

CURRÍCULO OFICIAL

Qlik Sense Arquitetura de Dados

Janeiro de 2016 - Segunda Edição

As informações contidas neste documento, incluindo *URLs* e outras referências a *sites* da Internet estão sujeitos a mudanças sem prévio aviso. A menos que informado explicitamente, todas as referências a empresas, organizações, produtos e nomes de domínios, além de endereços de e-mail, logomarcas e eventos são fictícios e não possuem qualquer vínculo com dados reais. Este documento está coberto pelas leis de autorias e nenhuma parte ou em todo poderá ser reproduzida, armazenada, obtida eletronicamente ou transmitida (mecânica ou eletronicamente) em qualquer que seja o formato, nem tão pouco fotografado, sem a expressa autorização do fornecedor.

Conteúdo

Introdução	6
Fontes de Consulta.....	6
Portal da QlikTech	6
Help Online (ou Ajuda Online)	7
Agenda	7
Questões a desenvolver durante o curso	8
Estrutura de Arquivos.....	8
Instalação do Qlik Sense Desktop	9
Lição 1 - Plano de Projeto.....	9
Objetivo da lição.....	9
O que é um Plano de Projeto?	9
Plano de Projeto	10
Medidas.....	10
Indicadores Chave de Performance	10
Dimensões.....	10
Tendências	10
Filtros de Seleção	10
Segurança	10
Descrição dos Dados	10
Laboratório 01 - Plano de Projeto	10
Lição 2 - Estrutura dos Dados e AQL	11
Objetivo da Lição.....	11
Estruturas de Dados	11
Exemplo de Modelo Associativo	12
Origem de Dados para Carga	12
Comandos de Carga de Dados	13
Fontes de Dados.....	14
Criar Conexão OLE DB	14
Passos para criação da Conexão OLE DB.....	14
Laboratório 02 - Estrutura de Dados.....	16
Lição 3 - Script de Carga	16
Objetivo da Lição.....	16
Carregar as Tabelas	16

Analizador de dados	17
Laboratório 03 - Script de Carga	18
Lição 4 - Preparação de Dados.....	18
Objetivo da Lição.....	18
Carga de Dados – Excel – Funcionários	19
Carga de Dados – Excel – Escritórios.....	19
Referência Circular	20
Verificador de Sintaxe	21
Carga de Dados – AS/400 – Arquivo DIF	22
Qualificando Campos	22
Laboratório 04 - Preparação de Dados	23
Lição 5 - Load Resident, Chave Sintética e Função Exists()	23
Objetivo da Lição.....	23
Load Resident.....	23
Chave Sintética.....	24
Função Exists	25
Laboratório 05 - Load Resident, Chave Sintética e Função Exists()	26
Lição 6 - Campos Chave.....	27
Objetivo da Lição.....	27
Campos Chave	27
Frequência.....	27
Qualificador Distinct.....	27
Adicionando Contadores.....	28
Laboratório 06 - Campos Chave	29
Lição 7 - Cálculos, Include e Dimensões de Tempo	29
Objetivo da Lição.....	29
Cálculos no Script	29
Comando Include	30
Dimensões de Tempo.....	30
Derivação de Campos.....	30
Declare	30
Derive	31
Load Inline	31
Mapping	32

Autogenerate	34
Laboratório 07 - Cálculos, Include e Dimensões de Tempo	35
Lição 8 - Tratamento de dados não padronizados	35
Objetivo da Lição.....	35
Etapa de Transformação	36
Crosstable.....	37
Generic Load	38
Laboratório 08 - Tratamento de dados não padronizados	39
Lição 9 - Arquivos QVD, QVX e QVW.....	40
Objetivo da Lição.....	40
Nesta lição, você vai aprender a:	40
O que são Arquivos QVD	40
Formato	40
Uso	40
Aumentar a velocidade de Carga	40
Consolidando dados de vários aplicativos.	40
Diminuir a carga sobre Servidores de Base de dados.	41
Carga Incremental	41
Criação de QVD	41
Criação Manual a partir do Script	41
Criação Automática a partir do Script	42
O que são Arquivos QVX	43
O que são Arquivos QVW	43
Laboratório 09 - Arquivos QVD	44
Lição 10 - Script Avançado.....	44
Objetivo da Lição.....	44
Script Avançado.....	44
Join	44
Agregação.....	46
Previous e Order by.....	47
Uso de Variáveis	49
Laboratório 10 - Script Avançado.....	50
Lição 11 - Concatenação.....	51
Objetivo da Lição.....	51

Concatenação Automática	51
Concatenação Forçada	52
Prevenir Concatenação	53
Concatenação na prática.....	54
Laboratório 11 - Concatenação	55
Lição 12 - Uso de mapas.....	57
Objetivo da Lição.....	57
Keyhole Markup Language (kml).....	57
Preparando os dados	57
Montagem do Gráfico	58
Laboratório 12 – Uso de Mapas	60
Lição 13 - Depuração.....	60
Objetivo da Lição.....	60
Depuração	60
Laboratório 13 - Depurar	61
Gabarito dos Laboratórios	63
Lição 1 - Plano de Projeto.....	63
Lição 2 - Estrutura de Dados	63
Lição 3 - Script de Carga	63
Lição 4 - Preparação de Dados	63
Lição 5 - Load Resident, Chave Sintética e Função Exists()	63
Lição 6 - Campos Chave.....	63
Lição 7 - Cálculos, Include e Dimensões de Tempo.....	63
Lição 8 - Tratamento de dados não padronizados	63
Lição 9 - Arquivos QVD	63
Lição 10 - Script Avançado	63
Lição 11 - Concatenação.....	63
Lição 12 - Uso de Mapas	63
Lição 13 - Depuração	63

Introdução

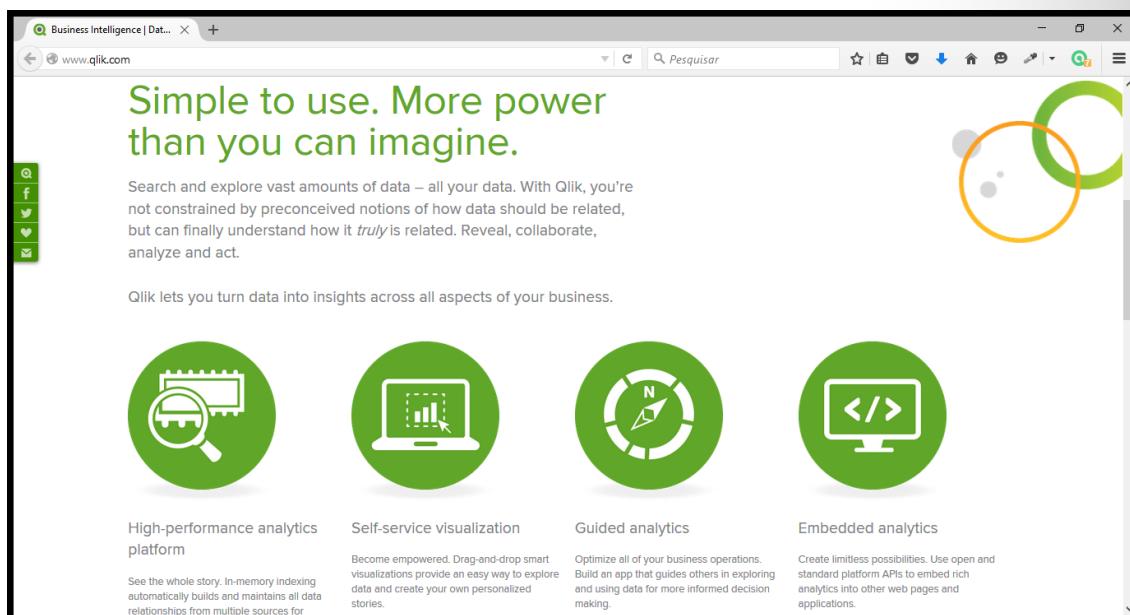
Fontes de Consulta

Vamos começar este curso respondendo à seguinte pergunta: onde posso ter acesso a informações sobre os produtos da Qlik?

Destacamos três fontes de consulta, a seguir:

Portal da QlikTech

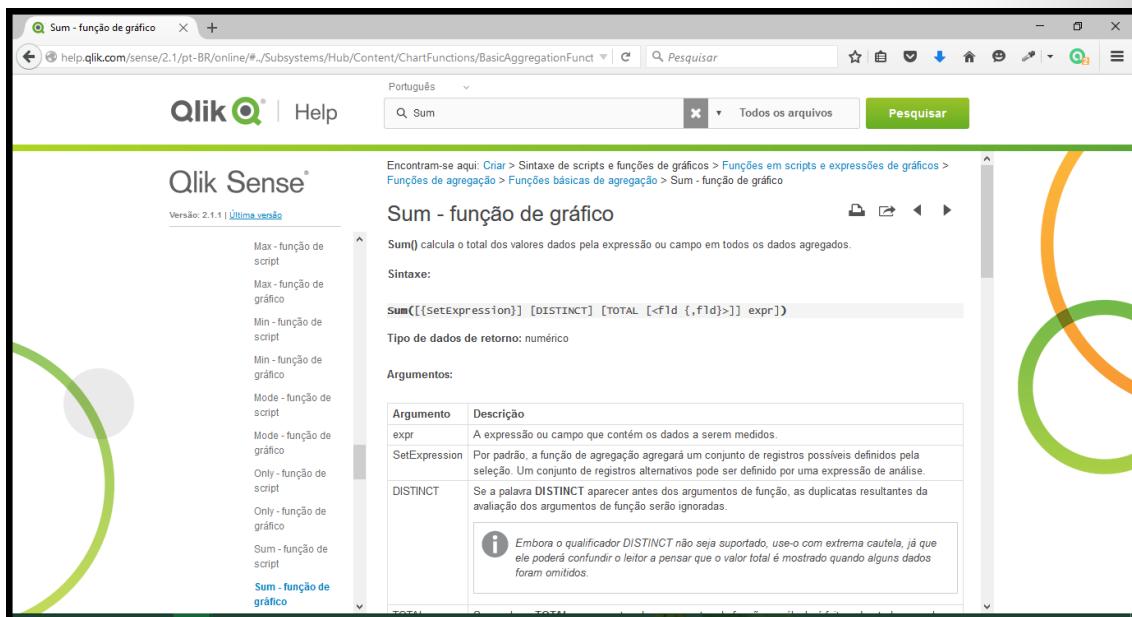
<http://www.qlik.com>



Este portal possui várias informações sobre os produtos e a Qlik. O que queremos destacar a você é a seção **Community**, que é a comunidade de Usuários e Desenvolvedores dos produtos da Qlik. Lá há grupos de usuários, fóruns, blogs, projetos para download e artigos técnicos (PDF). Para ter acesso aos recursos da **Community**, você deve fazer um cadastro no portal. Qualquer usuário pode solicitar seu cadastro.

Help Online (ou Ajuda Online)

Este portal, disponível em vários idiomas, inclusive o português, é bem completo. Diferencia-se por possuir exemplos para a maioria das funções.



Agenda

Durante o Curso Desenvolvedor I, abordaremos os seguintes temas:

- Estrutura de Arquivos
- Instalação do Qlik Sense Desktop
- Plano de Projeto
- Estrutura dos dados e AQL
- Script e Carga
- Preparação de Dados
- Load Residente, Chave Sintética e Função Exists()
- Campos Chave
- Load Inline, Autogenerate e Mapping
- Tratamento de dados não padronizados
- Arquivos QVD, QVW e QVX
- Script Avançado
- Concatenação
- Uso de Mapas
- Depuração

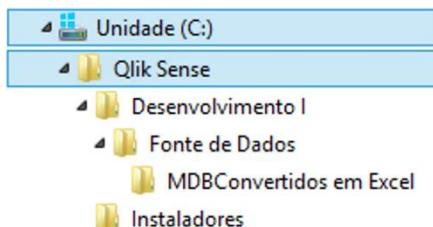
Questões a desenvolver durante o curso

Vamos ver agora algumas questões que serão desenvolvidas em nosso curso.

- Como interpretar um plano de projeto?
- Como acessar dados e integrá-los com dados de outras fontes?
- Quais são algumas das técnicas para desenvolver modelos de dados?
- Que recursos estão disponíveis para visualizar a estrutura de dados em várias fontes?
- Como e por que criar através do editor de carga e o gerenciador de dados?
- Quais os padrões a adotar para tornar minha aplicação eficiente, comprehensível e de manutenção simples?
- Como tornar uma aplicação segura?
- O que fazer quando tenho um problema?

Estrutura de Arquivos

Para este treinamento, usaremos instaladores e fontes de dados que estão dentro das pastas conforme o modelo a seguir:



Verifique o diretório **Desenvolvimento I\Fonte de Dados** com os seguintes arquivos:

Nome	Tipo	Tamanho
MDBConvertidos em Excel	Pasta de arquivos	
Brasil-Estados.kml	KML	318 KB
Email.txt	Documento de Texto	2 KB
Fornecedores.dif	Arquivo DIF	6 KB
FuncionariosEscritorios.xls	Planilha do Microsoft Excel 97-2003	33 KB
FuncionariosNovos.xls	Planilha do Microsoft Excel 97-2003	23 KB
Info.xls	Planilha do Microsoft Excel 97-2003	16 KB
MunicipiosDetalhes.qvd	Arquivo QVD	267 KB
Orcamento.xls	Planilha do Microsoft Excel 97-2003	29 KB
Pedidos.txt	Documento de Texto	52 KB
Produtos1.xlsx	Planilha do Microsoft Excel	12 KB
Produtos2.xlsx	Planilha do Microsoft Excel	12 KB
Produtos3.xlsx	Planilha do Microsoft Excel	12 KB
QWT.mdb	Arquivo MDB	2.460 KB
TodosMunicipios.kml	KML	23.487 KB

Verifique se o diretório **\Desenvolvimento I** contém o arquivo: **QWT BI Plano de Projeto.doc**

Instalação do Qlik Sense Desktop

Para este treinamento, usaremos instaladores da versão 2.1.1 do Qlik Sense Desktop.

Requisitos:

Windows 7 64bits ou superior e não pode ser versões Server.

8 GB RAM e 500MB de espaço livre em disco (*dependendo do volume de dados, talvez precise mais*).

Na pasta instaladores, execute o arquivo **Qlik_Sense/Desktop_setup.exe** e siga as instruções.

A instalação é simples como qualquer outro software que já tenha instalado em seu computador.

Se desejar, pode executar a instalação do **Qlik ODBC Connector Package x64**, um excelente recurso, disponibilizados pela Qlik, para obter drives de diferentes fontes de dados.

Lição 1 - Plano de Projeto

Objetivo da lição

Nesta lição, você vai aprender a:

- Necessidade de fazer um planejamento antes de começar a desenvolver o projeto.
- Apresentar um modelo de plano de projeto

O que é um Plano de Projeto?

Um plano de projeto é um documento guia com a finalidade de orientar o caminho a ser seguido durante o curso para alcançar um objetivo.

Temos observado que projetos construídos sem um planejamento inicial passam por constantes mudanças durante seu ciclo de vida, causando problemas como:

- Carga de dados desnecessários;
- Tempo de desenvolvimento perdido;
- Problemas no modelo de dados associados, como chaves sintéticas e referências circulares.

Apresentamos neste curso um modelo de plano de projeto, denominado Plano de Projeto QWT, que possui as seguintes características:

- O projeto não está configurado com gráficos, responsabilidades etc.
- Ele existe para ajudar a definir um objetivo a ser completado durante este curso.
- Vamos utilizar o plano de projeto como um guia para desenvolver o script de carga necessário para disponibilizar uma aplicação.
- O nome do arquivo com a definição do plano de projeto é **QWT BI Plano de Projeto.doc**.

Plano de Projeto

Veja agora as seções do Plano de Projeto QWT.

Medidas

Aqui encontraremos algumas das expressões que serão necessárias na aplicação.

Alguns desses cálculos serão usados no arquivo de carga, outros serão usados nos objetos (gráficos, texto, filtros, tabelas) que compõem a interface com o usuário.

Indicadores Chave de Performance

Esta seção inclui Indicadores Chave de Performance que podem ser mostrados através de objetos na aplicação.

Dimensões

Esta seção inclui a lista de algumas das dimensões chave a serem utilizadas durante o desenvolvimento dessa aplicação.

Tendências

Disponibiliza uma importante lista das dimensões que trabalham com campos “tempo” que serão necessários para uma análise histórica dos dados.

Filtros de Seleção

Inclui uma lista de campos necessários para executar seleções e filtros sobre os dados carregados na aplicação.

Segurança

Contém as necessidades referentes à segurança de acesso a uma aplicação.

Descrição dos Dados

Disponibiliza localização e descrição dos campos para cada fonte de dados.

Nesta lição você aprendeu o que é um plano de projeto e qual a importância de fazer um planejamento antes de começar a desenvolver o projeto. O que você achou? Está fácil, concorda? Vamos dar andamento aos nossos estudos!

Vamos praticar um pouco? Exercite o que você aprendeu aqui!

Laboratório 01 - Plano de Projeto

1. Faça um pequeno esboço de um projeto que você pretende desenvolver para a empresa que você trabalha ou para um cliente seu utilizando o modelo do Plano de Projeto QWT.
2. Com suas palavras, explique o que é um Plano de Projeto.
3. Relacione as colunas em relação às seções do Plano de Projeto QWT.
 - (A) Medidas
 - (B) Indicadores Chave de Performance

- (C) Dimensões
- (D) Tendências
- (E) Filtros de Seleção
- (F) Segurança
- (G) Descrição dos Dados

- () Esta seção inclui a lista de algumas das dimensões chave a serem utilizadas durante o desenvolvimento dessa aplicação.
- () Alguns desses cálculos serão usados no arquivo de carga, outros serão usados nos objetos (gráficos, texto, filtros, tabelas) que compõem a interface com o usuário.
- () Disponibiliza uma importante lista das dimensões que trabalham com campos “tempo” que serão necessários para uma análise histórica dos dados.
- () Inclui uma lista de campos necessários para executar seleções e filtros sobre os dados carregados na aplicação.
- () Disponibiliza localização e descrição dos campos para cada fonte de dados.
- () Contém as necessidades referentes à segurança de acesso a uma aplicação.
- () Esta seção inclui Indicadores Chave de Performance que podem ser mostrados através de objetos na aplicação.

Lição 2 - Estrutura dos Dados e AQL

Objetivo da Lição

Nesta lição, você vai aprender a:

- Conhecer as Estruturas de Dados que podem ser fontes de dados para aplicação.
- Conhecer a maneira de como trazer esses dados para os projetos.

Estruturas de Dados

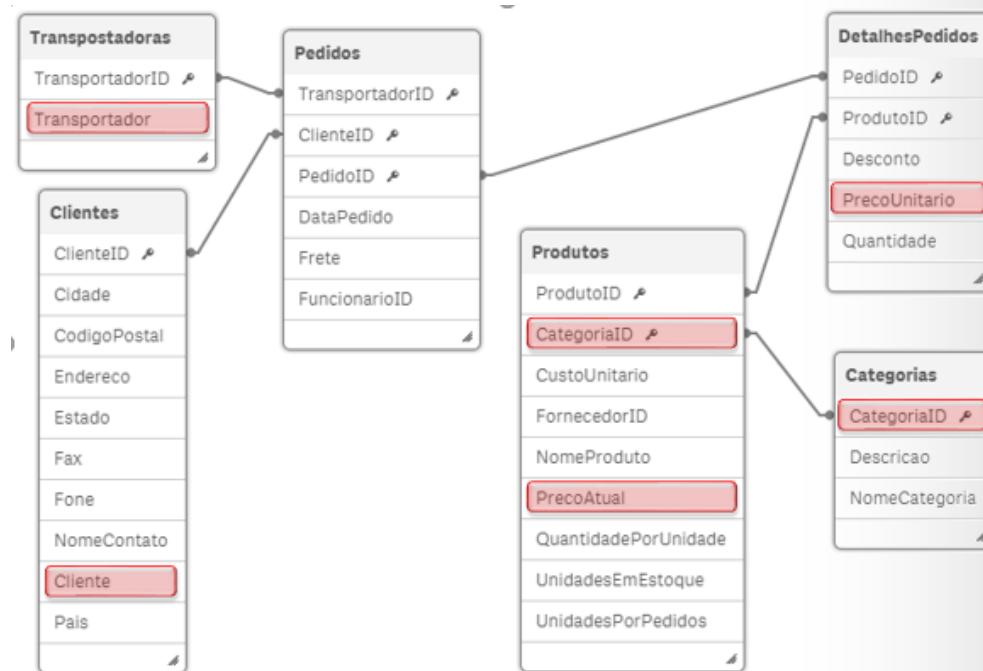
Existem várias classificações técnicas para Dados, como SGBD, Bancos Transacionais, Sistemas de Arquivos, entre outros. Para um fácil entendimento para este curso, podemos classificar os dados (fontes para projetos) em dois grupos:

- Bases de dados relacionais
 - Microsoft SQL Server, Microsoft Access, Oracle, DB2, Sybase, Informix, Teradata entre outros.
- Arquivos delimitados, fixos ou padronizados
 - Estrutura de dados comum para alimentar aplicações são arquivos de texto (csv, skv), planilhas no formato Excel e padrões XML, HTML, KML.

Uma vez que os dados foram carregados para o projeto, temos as seguintes definições:

- Cada coluna de uma tabela, que é carregada, torna-se um campo na base de dados associativa (também referenciada como base de dados AQL).
- Campos que aparecem em mais de uma tabela e tenham mesma identificação serão automaticamente associados.
- Cada campo pode ser apresentado na forma de filtros na aplicação.
- Quando uma seleção é feita em um campo ou gráfico, a pesquisa é feita através de toda a base de dados associativa por conexões lógicas. Como resultado desta pesquisa, os valores associados com sua seleção serão identificados.

Exemplo de Modelo Associativo



Origem de Dados para Carga

A lista a seguir mostra os tipos e origens de dados para as aplicações. Veja:

- O resultado de uma consulta SQL, usando qualquer conector compatível com a origem
- Arquivos delimitados (csv, txt, tab, qvo, mem, skv, prn, log)
- Arquivos de registro fixo (fix, dat)
- Arquivos de formato AS/400 (dif)
- Documentos QlikView (qvw)
- Qlik Data (qvd)
- Qlik Data eXchange (qvx)
- Excel (xls, xlw, xlsx, xlsxm)
- HTML (html, htm, php)
- XML (xml)

- KML (kml)
- Qlik DataMarket

Comandos de Carga de Dados

Os comandos a seguir são usados para conexão ao banco de dados e para seleção de tabelas.

- **Connect:** comando para conectar ao Banco de Dados, utilizando ODBC ou OLEDB.
- **Select:** comando SQL para selecionar dados de uma Tabela. O comando Connect deve ter sido executado antes do Select.
- **Load:** comando para seleção de dados.
- **As:** comando para renomear campos.

Não é necessário decorar sintaxe de string de conexão ao banco ou comando Select. Existe um assistente para montar essas linhas de código.

Exemplos de Connect:

```
ODBC connect to [SQLDATA;database=SQL1] (UserId is sa, Password is admin);
ODBC CONNECT TO [MS AccessDatabase;DBQ=data\sampledataba.mdb];
ODBC connect to [COSQL01;DATABASE=SALES DATA;Trusted_Connection=Yes];
LIB CONNECT TO 'OLEDB-QWT';
```

Exemplos de Select:

```
SQL SELECT * FROM CLIENTES;

SQL SELECT DISTINCT I.AddressID, Name, Address,
FROM [Invoice] I, [Address] A
WHERE I.InvoiceType is not null and I.InvoiceDate >= '2001-01-01',
and I.AddressID = A.AddressID;
```

Exemplos de Load:

```
Load *
from c:\userfiles\data2.txt (ansi, txt, delimiter is '\t', embedded labels);

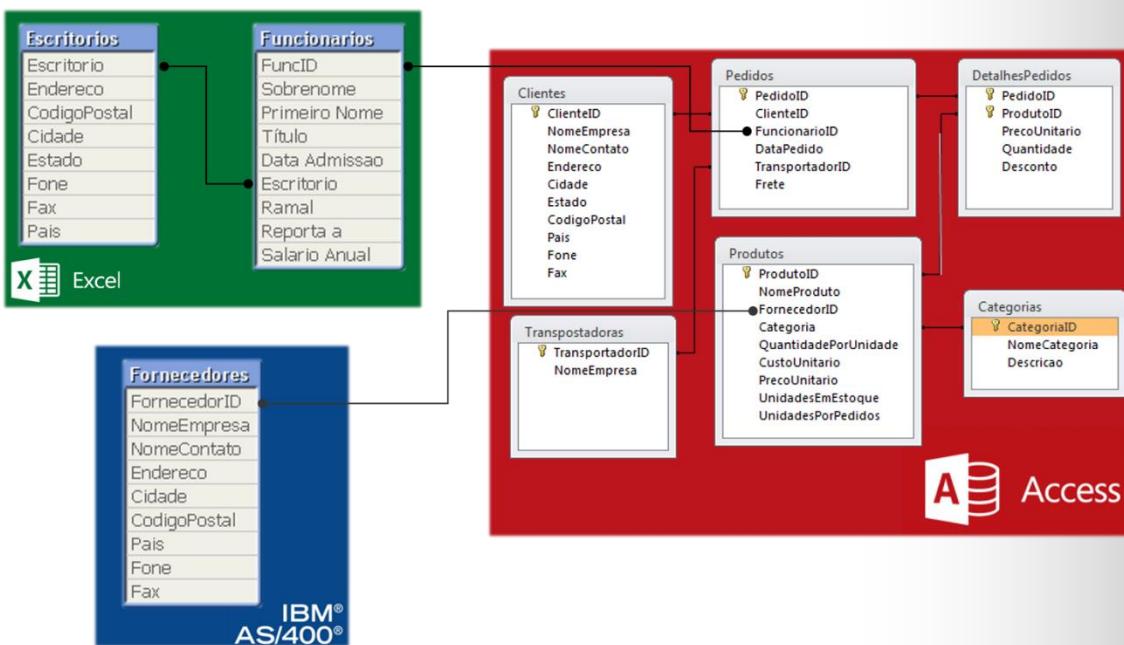
Load
RecNo() as A,
rand() as B
Autogenerate(10000);

Load
A,
B,
A*B+D as E
Resident tab1;

Load *
Inline [
CatID, Category
0, Regular
1,Occasional
2,Permanent];
```

Fontes de Dados

Veja que a imagem a seguir nos mostra que não importa de onde vieram os dados depois da leitura. Dados são dados e são trata-os sem distinção.



Criar Conexão OLE DB

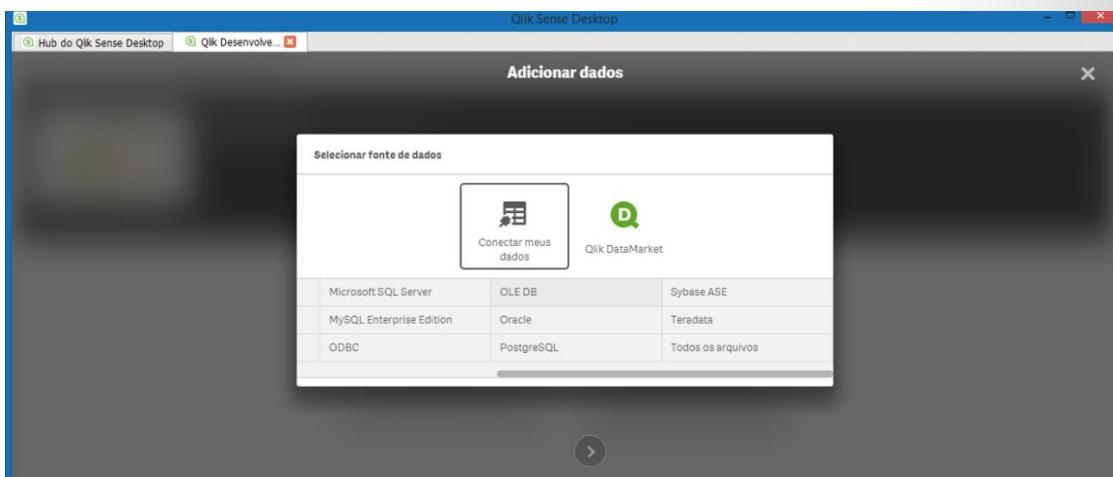
Para este curso, usaremos um banco de dados do Access, denominado **QWT.mdb**.

Para que possamos ter acesso a esse banco de dados, utilizaremos uma conexão OLE DB aos dados fonte.

Passos para criação da Conexão OLE DB

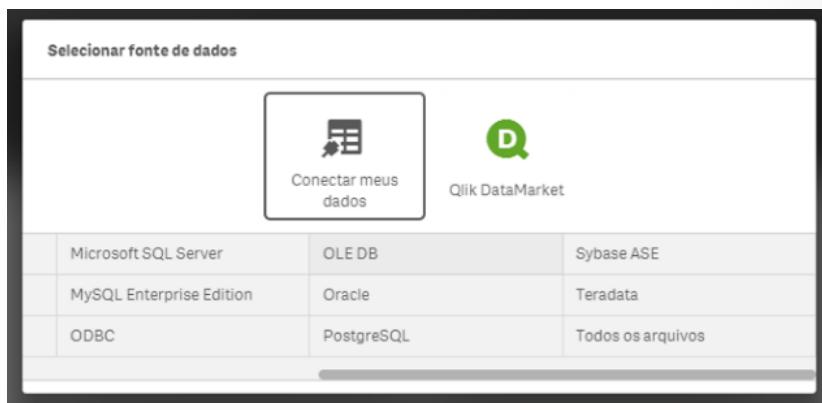
Vamos ver agora o passo a passo para a criação da conexão OLE DB após a criação de uma aplicação.

1. Crie um novo aplicativo com o nome de **Qlik Desenvolvedor I**.
2. Vá em **Adicionar dados** e verá a tela inicial do assistente.

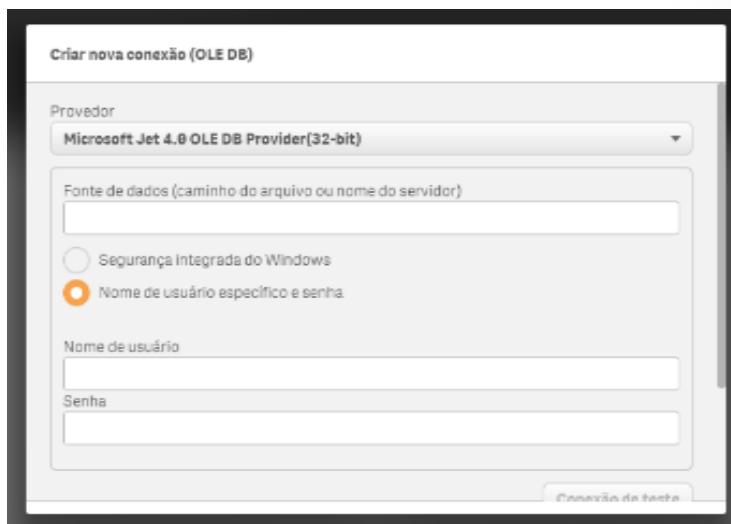


Qlik® - Material do Estudante Qlik Developer

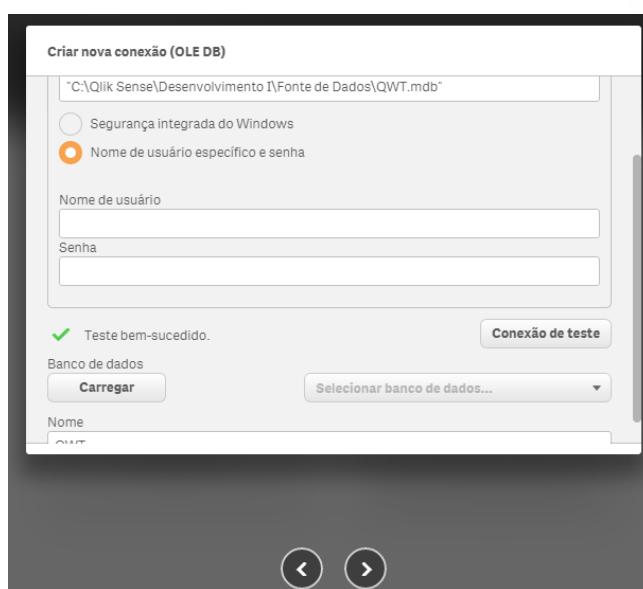
3. Posicione em **Conectar meus dados** e clique no botão **OLE DB**.



4. Selecione o **Provedor Microsoft Jet 4.0 OLE DB Provider(32-bits)**.



5. Informe o caminho da fonte de dados, teste a conexão, dê o nome **OLEDB-QWT** e clique em próximo.



Nesta segunda lição, você conheceu as estruturas de Dados que podem ser fontes de dados para Projetos e conheceu também a maneira de como o trazer esses dados para os projetos. Obteve exemplo de Dados Associados, aprendeu sobre a origem e os comandos de Dados para Carga, viu as Fontes de Dados e aprendeu a criar uma conexão OLE DB. Você conseguiu compreender todo o conteúdo estudado até aqui? Vamos prosseguir!

Vamos praticar? Exercite o que você aprendeu aqui!

Laboratório 02 - Estrutura de Dados

1. Liste pelo menos cinco tipos de Origens de Dados que podem ser carregados em Projetos.
2. Explique como faz para conectar nos diferentes tipos de Bancos de Dados (Oracle, SQL Server, DB2 etc.).
3. Relacione as colunas em relação aos comandos usados para conexão ao banco de dados e tabelas.
 - (A) Connect
 - (B) Select
 - (C) Load
 - (D) As

() Comando SQL para selecionar dados de uma Tabela. O comando Connect deve ter sido executado antes.

() Comando para conectar ao Banco de Dados, utilizando ODBC ou OLE DB.

() Comando para renomear campos.

() Comando para seleção de dados.

Lição 3 - Script de Carga

Objetivo da Lição

Nesta lição, você vai aprender a:

- Carregar as tabelas que serão utilizadas no treinamento.
- Fazer o tratamento inicial dos dados.

Carregar as Tabelas

Agora vamos ver os passos para carregar as primeiras tabelas na aplicação.

Se após a criação da aplicação e criação da conexão na lição acima, você fechou o aplicativo, abra a aplicação e vá em Adicionar Dados no menu, do contrário, siga os passos abaixo.

- Selecione todos os campos nas tabelas **Categorias, Clientes, DetalhesPedidos, Pedidos, Produtos e Transportadoras**, marcando-as cada uma conforme abaixo.

Selecionar dados

Banco de dados Visualizar dados Metadados Filtrar campos

Proprietário Cid... ClienteID CódigoPostal Endereço Estado... Fax Fone NomeCo...

Tabelas Filtrar tabelas

Categorias	10	Modehuset Abel	Dennis Olsen	Smagsløget 45	Århus	8200	Denmark
Clientes	10	Stephanies	Staffan Blond	Gatauppochner. 1	Stockholm	11111	Sweden
DetalhesPedidos	5	Eintrach GS	Albert von Einstein	Obere Str. 57	Berlin	12209	Germany
Pedidos	6	La Tienda de la Esquina	Paco el Maco	Avda. de la Constitución 2222	México D.F.	05021	Mexico
Produtos	9	3 La Ropa Vieja	Sancho Panza	C/ Ritual de lo Habitual 2312	México D.F.	05023	Mexico
Transportadoras	2	4 Dr. Jim's Trousers	Carl Montgomery	120 Hanover Sq.	London	WA1 1DP	UK
MSysACEs		5 Urras Shop	Urra Gurra Aktersnurra	Berguvsvägen 8	Luleå	S-958 22	Sweden
		6 Man Kleider	Herman Hinschler	Forsterstr. 57	Mannheim	68306	Germany
		7 Menage à Trois	Julie Binoché	24, place Kléber	Strasbourg	67000	France
		8 Las Corbatas	Julio Iglesias	C/ de Don Quijote, 67	Madrid	28023	Spain
		9 La Legion Mercenaire	Bernard de Gaulle	12, rue des Bouchers	Marseille	13008	France
		10 Big Foot Shoes	James Hendersson	23 Tsawassen Blvd.	Tsawassen	BC	T2F 8M4
		11 Shoe Expert	David Foot	Fauntleroy Circus	London	EC2 5NT	UK
		12 Los Pantalones Magicos	Victoria Abril	Plaza de Mayo 6	Buenos Aires	1010	Argentina
		13 Los Sombreros Gigantes	Speedy Gonzales	El Barrio Chino 12	México D.F.	05021	Mexico
		14 Das Alpen Shoe	Alfred Neumann	Hauptstr. 29	Bern	3012	Switzerland

Analizar dados Carregar e concluir

- Pode-se visualizar os dados de cada tabela ao clicar no nome correspondente ao lado esquerdo.
- Clique em **Analizar dados**.

Analizador de dados

Vamos agora analisar os 12 pares de tabelas e os 9 avisos gerados pelo analisador de dados.

Recomendações de associações de tabela 9 avisos

DetalhesPedidos - Clientes 1 de 12

Atual	Recomendações
Sem associação	ProdutoID ClienteID 91%
	Quantidade ClienteID 71%

Essas duas tabelas não estão associadas, mas contêm dados parecidos.
Selecione qual associação de tabela recomendada você deseja usar.

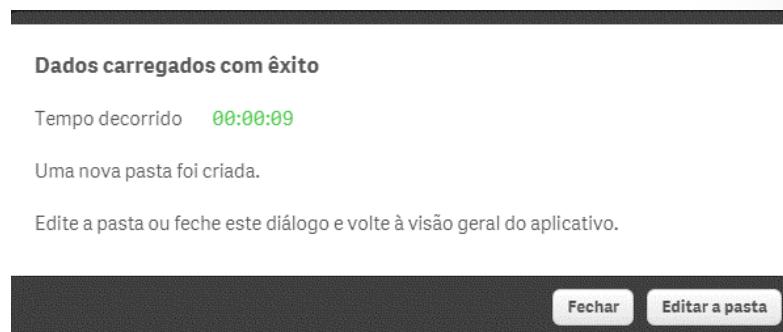
DetalhesPedidos **Clientes**

Desconto	PedidoID	PrecoUnitario	ProdutoID	Quantidade	Cidade	ClienteID	CódigoPostal	Endereço	Estado	Fax	Fone
0	10248	28,62	11	12	Århus	94	8200	Smagsløget 45	-	86 22 33 44	86 21
0	10248	14,04	42	10	Stockholm	127	11111	Gatauppochner. 1	-	(8) 11 22 33	(8) 10
0	10248	12,71	72	5	Berlin	1	12209	Obere Str. 57	-	030-0076545	030-6
0	10249	22,8	14	9	México D.F.	2	05021	Avda. de la Constitución 2222	-	(5) 555-3745	(5) 55
0	10249	101,2	51	40	México D.F.	3	05023	C/ Ritual de lo Habitual 2312	-	-	(5) 55
0	10250	9,59	41	10	London	4	WA1 1DP	120 Hanover Sq.	-	(171) 555-6750	(171)
0,15	10250	97,67	51	35	Luleå	5	S-958 22	Berguvsvägen 8	-	00011234567	0071

Carregar e concluir

- **Transportadoras - Clientes:** Volte a tela anterior e modifique o nome do campo NomeEmpresa para **Transportador** e o campo NomeEmpresa existe na tabela de Clientes para **Cliente**, não queremos confundir nem fazer chave entre essas tabelas. Clique em **Analizar dados** novamente.
- **Produtos - Categorias:** Nessa mesma tela, aceite a recomendação de **Categoria** e **CategorialD** que tem **100%** de compatibilidade entre as tabelas e renomeie os campos para **CategorialD**.
- **Produtos - DetalhesPedidos:** Volte a tela anterior e modifique o nome do campo PrecoUnitario para **PrecoAtual**, não queremos fazer chave composta com os dois

- campos, somente o campo ProdutoID faz chave nessa tabela. Clique em **Analizar dados** novamente.
- Os demais pares, não há o que alterar, mas é importante observá-los em busca de possíveis inconsistências.
 - Clique em **Carregar e concluir**.



Nesta lição você aprendeu a carregar as tabelas que serão utilizadas no treinamento e a fazer o tratamento inicial dos dados. Ainda restou alguma dúvida? Lembre-se de prosseguir somente se tudo estiver muito claro para você!

Exercite o que você aprendeu aqui!

Laboratório 03 - Script de Carga

1. Para fixar bem o processo de conexão ao banco de dados e carregamento de tabelas, crie uma outra aplicação:
 - a) Faça a conexão com o banco de dados através de OLE DB.
 - b) Carregue as tabelas.
 - c) Avalie todos os pares e avisos.
2. Complete as lacunas com a alternativa correta.

O analisador de dados é usado para _____ os dados seguindo as regras de negócio e evitando a geração de possíveis problemas de _____ inválidas e associações entre dados que não tem relação entre eles.

- a. modelar; chaves
- b. inserir; informações
- c. documentar; dados
- d. documentar; informações
- e. explicar; chaves

Lição 4 - Preparação de Dados

Objetivo da Lição

Nesta lição, você vai aprender a:

- Carregar dados de outras fontes.

- Simular problemas no script que podem ocorrer durante o trabalho no dia a dia, bem como opções para a solução desses problemas.

Carga de Dados – Excel – Funcionários

Até aqui carregamos dados através de uma conexão OLE DB. A partir de agora, vamos carregar dados de outras fontes, como Excel (xls) e arquivo AS/400 (dif).

Vamos fazer a carga da tabela Funcionários e da tabela Escritórios. Ambas estão no arquivo FuncionariosEscritorios.xls.

Ao abrir o arquivo Excel, você pode observar que existem duas planilhas:

- Funcionarios: Identifica os Funcionários.
- Escritorios: Identifica os Escritórios onde os Funcionários trabalham.

	A	B	C	D	E	F	G	H	I
1	FuncID	Sobrenome	Primeiro Nome	Título	Data Admissão	Escritorio	Ramal	Reporta a	Salário Anual
2	1	Roll	Frank	Sales Representative	10/01/03	5	501	4	61000,00
3	2	Presley	Erik	President	09/14/93	1	101		180000,00
4	3	Carsson	Rob	Sales Representative	10/01/94	1	102	4	63000,00
5	4	Callins	Joan	Sales Manager	09/03/94	3	301	2	120000,00
6	5	Hendrix	Ingrid	Sales Representative	10/17/95	3	302	4	61300,00

O campo chave nessa tabela é FuncID, que permitirá associar essa informação aos demais dados já carregados.

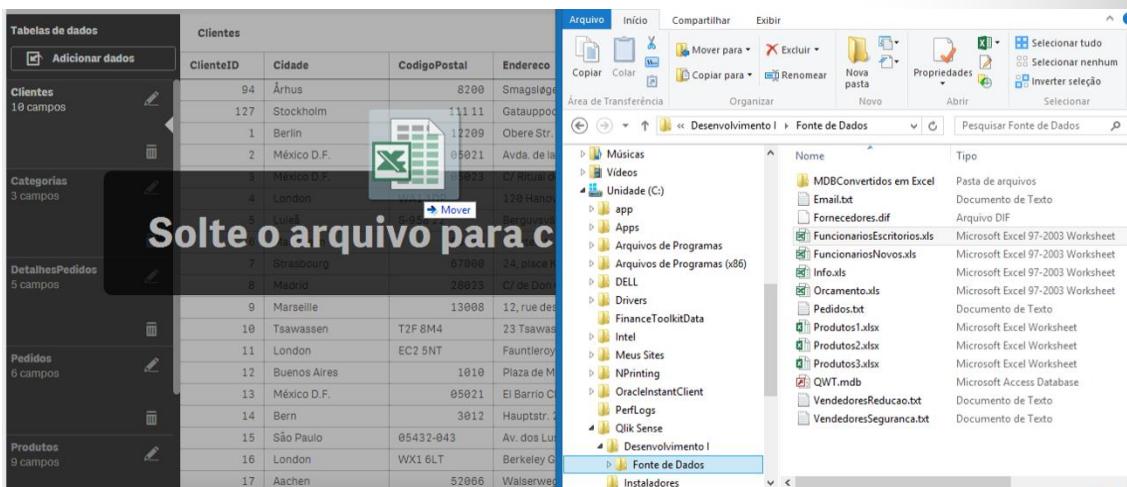
Carga de Dados – Excel – Escritórios

A tabela de Escritórios está no mesmo arquivo do Excel FuncionariosEscritorios.xls, mas na pasta Escritórios, como já citamos.

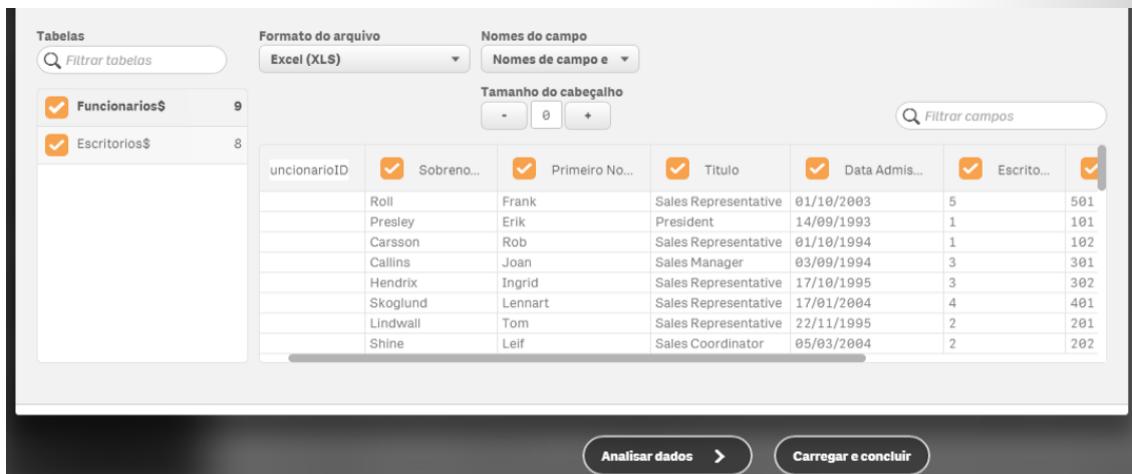
O campo chave nessa tabela é Escritorio e os valores desse campo serão associados ao do campo Escritorio na tabela Funcionarios.

	A	B	C	D	E	F	G	H
1	Escritorio	Endereço	Código Postal	Cidade	Estado	Fone	Fax	País
2	1	Citygatan. 1	111 11	Stockholm		(8) 100 100	(8) 100 110	Sweden
3	2	Mellangatan. 33	222 22	Lund		(46) 200 200	(46) 200 222	Sweden
4	3	66 Rue de Qlik	3456	Paris		(9) 344 344	(9) 344 362	France
5	4	Boulevard 3	7890	Nice		(67) 366 366	(67) 366 234	France
6	5	234 Sun Street	98122	Seattle	WA	(206) 567 5670	(206) 567 5690	USA

Arraste e solte o arquivo **FuncionariosEscritórios.xls**, na aplicação e o assistente de carga será exibido.



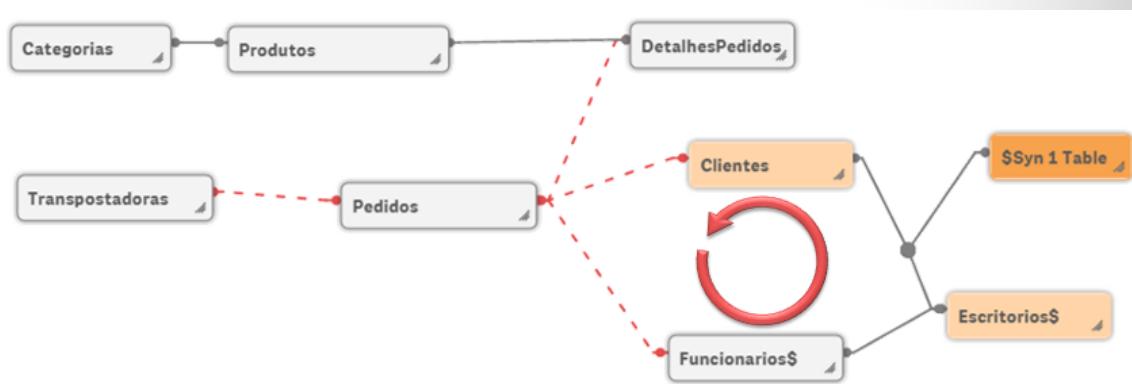
Marque a planilha **Funcionarios\$** e mude o rótulo de **FuncID** para **FuncionarioID**, também marque a planilha **Escrítorios\$** e clique em **Carregar e concluir**.



Referência Circular

Ao fazer a carga dos dados, aparece tela de alerta para a existência de uma **Referência Circular** e **Chave sintética**.

Estruturas com referências circulares, ou seja, quando a cadeia se torna um anel, precisam ser evitadas.



Uma referência circular é um sinal de um modelo de dados incorreto, no qual dois campos semelhantes, que têm interpretações ligeiramente diferentes, são tratados como um só.

Quando o indexador descobre a referência circular durante a execução do script, as tabelas são parcialmente desconectadas.

Sempre que ocorrer uma referência circular, você deve se perguntar: - Eu consigo resolver isso renomeando campos? Se sua resposta for sim, significa que os nomes dos campos iguais que causaram a referência circular referem-se a dados que não deveriam estar associados, ou seja, refletem informações distintas, apesar de ter o mesmo nome.

Entretanto, se realmente existir a necessidade de os dados estarem associados, você não poderá renomear os campos. A solução será trabalhar de modo diferente a modelagem dos dados, implementando um modelo estrela, utilizando uma tabela de ligação, por exemplo. No entanto, essa técnica para implementar um modelo estrela utilizando uma tabela de ligação será visto em outro curso.

Para resolver essa referência circular, vamos alterar a carga da planilha Escritórios renomeando o nome dos campos, conforme abaixo.

No **Gerenciador de dados**, edite a tabela **Escritorios\$** incluindo a palavra **Escrítório** em cada rótulo, exceto no campo chave.

Código País	Escritório Cidade	Escritório Estado	Escritório Fone	Escritório Endereço	Escritório País
	Stockholm		(8) 100 100	(8) 100 110	Sweden
	Lund		(46) 200 200	(46) 200 222	Sweden
	Paris		(9) 344 344	(9) 344 362	France
	Nice		(67) 366 366	(67) 366 234	France
	Seattle	WA	(206) 567 5670	(206) 567 5690	USA

Verificador de Sintaxe

```

Trimestres << AUTOGENERATE (12)
Linhas buscadas: 12
Conectando a OLEDB-QWT
Conectado
Clientes << Clientes
Linhas buscadas: 92
Conectando a OLEDB-QWT
Conectado
Categorias << Categorias
Linhas buscadas: 8
Conectando a OLEDB-QWT
  
```

Até a versão 9 do QlikView, o produto mais conhecido da Qlik, para localizarmos possíveis erros no script era necessário realizar a carga dos dados. Mesmo com a possibilidade de fazer a carga parcial, ainda era um processo demorado, pois, se houvesse mais de um erro, o próximo erro só seria detectado na próxima recarga. O Verificador de Sintaxe, disponível atualmente, apontará o local onde está o erro. Isso facilitará em muito o processo de desenvolvimento.

O que está achando do andamento de nosso curso? Está tranquilo até aqui? Caso sinta necessidade, faça uma revisão do conteúdo!

Carga de Dados – AS/400 – Arquivo DIF

Os dados sobre os Fornecedores da empresa serão extraídos de um arquivo DIF com o nome **Fornecedores.dif**. Podemos visualizar um arquivo DIF utilizando o Excel, veja a seguir:

A	B	C	D	E	F	G	H	I	
1	FornecedorID	NomeEmpresa	NomeContato	Endereco	Cidade	CodigoPostal	País	Fone	Fax
2	1	Sunny Clouds	Peter Smith	49 Gilbert Street	London	EC1 4SD	UK	(171) 555-1234	(171) 555-2345
3	2	Big Little	James Brown	P.O. Box 7	New Orleans	70117	USA	(100) 555-1234	(100) 555-2345
4	3	Dressed for Success	Glenn Miller	707 Oxford Street	Ann Arbor	48104	USA	(313) 555-1234	(313) 555-2345
5	4	Nitsurukihin	James Suzuki	9-8 Sekimachi	Tokyo	100	Japan	(03) 3555-1234	(03) 3555-2345

DIF significa Data Interchange Format e é usado, por exemplo, quando arquivos texto são transferidos a partir de ambiente AS/400.

Para abrir um arquivo DIF, vá no **Editor de carga de dados**, crie uma seção para organizar melhor o código, se preferir e digite o código abaixo:

```

Fornecedores:
LOAD FornecedorID,
      NomeEmpresa,
      NomeContato,
      Endereco,
      Cidade,
      CodigoPostal,
      Pais,
      Fone,
      Fax
FROM [lib://Fonte de Dados/Fornecedores.dif]
(dif, codepage is 1252, embedded labels);

```

Carregue os dados e observe que novamente ocorreu a mensagem alertando para a **Referência Circular**. Dessa vez resolveremos de outra maneira.

Qualificando Campos

Para resolver essa Referência Circular, você não deve mudar o nome dos campos manualmente, utilize a técnica de Qualificar os campos da tabela lida. Qualificar campos significa identificar os campos acrescentando o nome da tabela antes do nome dos campos.

Insira as linhas de código abaixo, antes do comando de carregamento da tabela Fornecedores.

```

Qualify *;
Unqualify FornecedorID;

```

Como o campo FornecedorID é usado para associar a outra tabela, não deverá utilizar o qualificador da tabela.

Isso é especificado usando o comando.

```
Unqualify FornecedorID;
```

Após a tabela Fornecedores ter sido carregada, temos de adicionar o seguinte comando para que os outros campos carregados posteriormente não utilizem o qualificador de tabela:

```
Unqualify *;
```

A figura a seguir mostra como ficará o bloco inteiro de leitura da tabela Fornecedores. Veja:

```
Qualify *;  
Unqualify FornecedorID;  
  
Fornecedores:  
LOAD FornecedorID,  
    NomeEmpresa,  
    NomeContato,  
    Endereco,  
    Cidade,  
    CodigoPostal,  
    Pais,  
    Fone,  
    Fax  
FROM [lib://Fonte de Dados/Fornecedores.dif]  
(dif, codepage is 1252, embedded labels);  
  
Unqualify *;
```

Nesta lição você aprendeu a carregar dados de outras fontes e a simular problemas no script que podem ocorrer durante o trabalho no dia a dia, bem como opções para a solução desses problemas. Vamos adiante!

Exercite o que você aprendeu aqui!

Laboratório 04 - Preparação de Dados

1. Aplique o comando Qualify para todos os campos da tabela Categorias. Tome o cuidado, no entanto, para manter a associação de Categorias com Produtos, depois remova esse comando para darmos seguimento ao curso.
2. O que significa qualificar campos?

Lição 5 - Load Resident, Chave Sintética e Função Exists()

Objetivo da Lição

Nesta lição, você vai aprender a:

- Criar uma tabela de dados diretamente no script através do Load Resident.
- Simular problemas no script que podem ocorrer durante o trabalho no dia a dia, bem como opções para a solução desses problemas.
- Entender o que são e como podemos evitar o uso de chaves sintéticas.
- Função Exists para condicionar o preenchimento de tabelas.

Load Resident

Agora você vai aprender a criar uma tabela usando como base uma tabela previamente carregada (Resident).

Vai aprender também a segmentar o script de carga em diferentes seções, para facilitar a leitura e a manutenção.

Siga os passos:

- Abra o **Editor de carga de dados**.
- Adicione uma nova seção clicando no ícone **Criar nova seção**. Use **Vendedores** como nome dessa nova aba.
- Adicione um comando de carga para uma nova tabela, mas, dessa vez, em vez de usar o Assistente para criar o código, digite o código abaixo:

```
/* Tabela Vendedores */  
Vendedores:  
LOAD FuncionarioID,  
    Sobrenome,  
    [Primeiro Nome],  
    Título as TituloVendedor  
Resident Funcionarios$;
```

Agora queremos limitar a carga dos registros dos empregados para somente aqueles que podemos identificar como pessoal de vendas.

- Para isso, você precisa fazer outra alteração no código do script:
- Primeiro, remova o ponto e vírgula (;) localizado após **Resident Funcionarios\$**.

Depois, adicione a condição **Where** após o comando **Resident**, como a seguir:

```
Where Left(Titulo, 3) = 'Sal'  
      or Titulo = 'President';
```

Chave Sintética

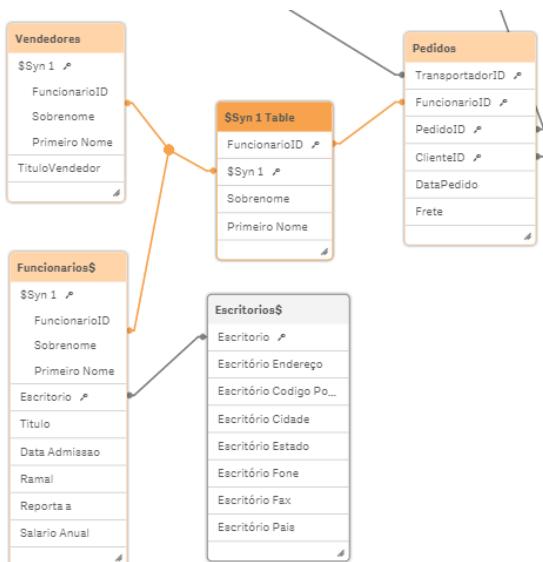
É indesejável que tenhamos várias chaves comuns através das tabelas da estrutura de dados.

Essa situação pode levar o indexador a usar chaves complexas (chaves sintéticas) ao gerar a estrutura de dados.

Chaves sintéticas é um tipo de recurso pesado, e geralmente o uso desse tipo de recurso pode tornar lento o processo de cálculo e, em casos extremos, sobrecarregar uma aplicação. Também tornam um documento difícil de entender e manter.

Quando você criou a tabela que gera o campo **TituloVendedor**, inadvertidamente foi criada uma chave sintética entre as tabelas **Funcionários** e **Vendedores**.

A chave sintética é gerada em uma nova tabela, que pode ser identificada no Visualizador do modelo de dados.



Como podemos ver, existe uma tabela para chave sintética composta pelos campos **Primeiro Nome, Sobrenome e FuncionarioID**.

Neste caso, não existe razão para que você utilize os campos nome como uma chave adicional entre essas tabelas, já que o campo FuncionarioID já serve para esse propósito.

Para corrigir essa situação, você utilizará a técnica de mudar os nomes dos campos novamente.

Mude o nome dos campos que não necessite como campos chave.

Na leitura da tabela **Vendedores**, vamos renomear os campos conforme abaixo:

```
/* Tabela Vendedores */
Vendedores:
LOAD FuncionarioID,
[Primeiro Nome] & ' ' & Sobrenome as Vendedor,
Titulo as TituloVendedor
Resident Funcionarios$
Where Left(Titulo,3) = 'Sal'
or Titulo = 'President';
```

Após carregar os dados, você poderá confirmar que a chave sintética não mais existe. Você pode confirmar isso usando o visualizador de tabelas novamente.

Função Exists

Você utilizou um método simples que permitiu identificar os Vendedores em um conjunto de Funcionários. Para este caso específico, foi adequado, mas podemos chegar ao mesmo resultado usando uma solução mais interessante e mais confiável.

Agora, você definirá que os vendedores são aqueles que estão incluídos nos nossos dados de vendas. Serão considerados vendedores aqueles que forem identificados no campo **FuncionarioID** na tabela **Pedidos**.

Para começar, vamos criar um campo com o nome **FuncionarioVendedor** para identificar os vendedores com segurança.

Utilize a linha de comando abaixo para a tabela **Pedidos**, imediatamente após o campo **FuncionarioID**.

```
FuncionarioID,  
FuncionarioID as FuncionarioVendedor,
```

Modifique a cláusula **Where** na tabela Vendedores utilizando a função **Exists()**.

```
/* Tabela Vendedores */  
Vendedores:  
LOAD FuncionarioID,  
    [Primeiro Nome] & ' ' & Sobrenome as Vendedor,  
    Titulo as TituloVendedor  
Resident Funcionarios$  
Where Exists(FuncionarioVendedor, FuncionarioID);
```

A condição na cláusula Where verifica se existem dados carregados que sejam equivalentes aos valores do campo **FuncionarioVendedor**.

Como o nome indica, a função Exists() pode ser utilizada para verificar se um valor específico existe em um dado campo dos dados carregados até esse momento.

Lembre-se de cuidar com a ordem com que você define a sequência de comandos, já que os campos de referência devem ser alimentados antes da verificação de valores.

Neste exemplo, a tabela Pedidos tem de ser carregada antes da tabela Vendedores para que essa condição funcione de forma apropriada. Após completar essas modificações, carregue os dados para ser verificada a funcionalidade.

Nesta lição, você aprendeu a criar uma tabela de dados diretamente no editor de carga de dados através do Load Resident, entendeu o que são e como podemos evitar o uso de chaves sintéticas e viu a Função Exists, que serve para condicionar o preenchimento de tabelas. Lembre-se: em caso de dúvida, retorne ao conteúdo e faça uma revisão dos pontos que achar necessário.

Exercite o que você aprendeu aqui!

Laboratório 05 - Load Resident, Chave Sintética e Função Exists()

1. Faça uma nova aplicação utilizando os dados desse curso e crie uma seção no editor de carga de dados com o nome “Produtos Vendidos”. Crie uma tabela com os seguintes campos:
 - o ProdutoID,
 - o PedidoID (Renomear para Pedido) e
 - o PrecoUnitario (Renomear para ValorUnitario)

Use as Funções Resident e Exists para criar essa nova tabela.

2. Complete a frase a seguir:

A frequência de um campo está associada ao _____, ou seja, é o número de vezes que um valor aparece no conjunto de _____.

- a. dado; campos
- b. valor; obras
- c. valor; dados
- d. número; dados

Lição 6 - Campos Chave

Objetivo da Lição

Nesta lição, você vai aprender a:

- Conhecer os problemas de cálculo de frequência (contagem) sobre campos chave.
- Saber técnicas para evitar esses problemas.

Campos Chave

Campos chave são aqueles que são comuns em duas ou mais tabelas (campos associados).

Frequência

A frequência de um campo está associada ao valor, ou seja, é o número de vezes que um valor aparece no conjunto de dados.

O problema de calcular a frequência em um campo chave é que o indexador não saberá qual tabela utilizar para contar a incidência dos dados.

- Supondo que temos uma tabela chamada **Pedidos** com **1.000** diferentes números de pedidos (**PedidoID**).
- Temos também uma tabela com nome **PedidosExternos**, que contém **200** números de pedidos.
- Esses números também são encontrados na tabela **Pedidos**.
- As duas tabelas são associadas através do campo em comum **PedidoID**.

O problema está quando queremos saber exatamente o número de pedidos únicos.

O correto é **1.000, 200 ou 1.200?**

Com base nas informações que temos, sabemos que é correto responder que existem **1.000** pedidos únicos, mas isso não é tão claro assim no modelo associativo.

Se olhássemos as propriedades do campo no modelo, seria exibido **1.200** valores **não nulos** e **1.000** valores totais distintos.

Também é importante saber que será impossível usar, nos gráficos e itens mestres, funções para calcular frequência de campos associados.

Qualificador Distinct

Para calcular corretamente a frequência de um campo chave, o uso do **Distinct** garante que contaremos somente uma vez a incidência de cada valor possível do campo. Por exemplo: se queremos saber quantos Pedidos únicos existem:

Em vez de utilizarmos a expressão: **Count(PedidoID)**, adicionamos o **distinct** da seguinte maneira: **Count(DISTINCT PedidoID)**.

Adicionando Contadores

Uma solução relativamente simples para o problema dos campos chave e o cálculo de frequência de dados é carregar o campo que queremos calcular a frequência uma vez mais com outro nome.

Esse problema pode ser resolvido da seguinte forma:

```
/* Exemplo */
[Pedidos]:
LOAD [PedidoID],
[PedidoID] as CódigoPedido,
...
```

Em geral, é boa prática evitar o uso de campos chave em listas e expressões. Campos chave devem ser usados para ligar tabelas, e não para mostrar dados em uma aplicação.

Agora o novo campo (não associado) pode ser utilizado em um filtro, que mostrará a frequência ou o gráfico com funções para calcular frequência.

O novo nome de campo pode facilmente ser tratado ao atribuirmos uma outra identificação por forma a não confundir os usuários.

Observe a tabela abaixo e os resultados apresentados:

Vendedor	Produto	ClienteID	Quantidade
Frank Roll	Lace Shoes	51	3
Frank Roll	Lace Shoes	82	3
Joan Callins	Casual Boots	84	4
Lennart Skoglund	Casual Boots	14	3
Rob Carsson	Lace Shoes	89	3
Tom Lindwall	Lace Shoes	24	6
Expressões		Resultados	
Count(Vendedor)			5
Count(Distinct Vendedor)			5
Count(Produto)			2
Count(Distinct Produto)			2
Count(Quantidade)			6
Count(Distinct Quantidade)			3
NumericCount(Quantidade)			6
Sum(Quantidade)			22

Nesta lição, você aprendeu a identificar os problemas de cálculo de frequência (contagem) sobre campos chave e conheceu técnicas para evitar esses problemas. Até aqui tudo certo? Então vamos prosseguir!

Vamos praticar!

Laboratório 06 - Campos Chave

1. Modifique o script na sua aplicação para incluir um campo com nome **CodigoProduto**, com base no campo **ProdutOID** na tabela **Produtos** e carregue os dados.
2. Crie uma tabela dinâmica, com:

Dimensões:

NomeProduto e NomeCategoria, nessa ordem.

Expressões:

Count(ProdutOID), Count(Distinct ProdutOID) e Count(CodigoProduto)

3. Mostre somas parciais por NomeProduto e NomeCategoria
4. Duplique o campo **PedidoID** com o nome **CodigoPedido** em **Pedidos** e carregue os dados.
5. Crie uma tabela dinâmica, com:

Dimensões:

NomeCategoria e NomeProduto, nessa ordem.

Expressões:

Count(PedidoID), Count(Distinct PedidoID) e Count(CodigoPedido)

6. Mostre somas parciais por NomeCategoria e NomeProduto.
7. Explique com suas palavras o que são campos chave.

Lição 7 - Cálculos, Include e Dimensões de Tempo

Objetivo da Lição

Nesta lição, você vai aprender a:

- Fazer cálculos diretamente do script.
- Embutir código a partir de arquivos externos a aplicação.
- Utilizar técnicas diferentes para criar as dimensões de tempo como: Declare e Derive, Load Inline, criando dados no script, Autogenerate criando registros automaticamente e usando tabelas mapeadas (Mapping).

Cálculos no Script

É possível usar diversos operadores e funções no script para efetuar os mais variados cálculos, tanto diretamente no Load quanto para definições de variáveis.

Vamos criar a medida **Venda Líquida** na memória, com dados da tabela **Detalhes Pedidos** fazendo os cálculos conforme script abaixo.

```
[DetalhesPedidos]:
LIB CONNECT TO [OLEDB-QWT];
LOAD [Desconto],
[PedidoID],
[PrecoUnitario],
[ProdutoID],
[Quantidade],
[PrecoUnitario]*[Quantidade]*(1-[Desconto]) as VendaLiquida;
SQL SELECT `Desconto`,
`PedidoID`,
`PrecoUnitario`,
`ProdutoID`,
`Quantidade`
FROM `DetalhesPedidos`;
```

Comando Include

É possível fazer a inclusão de partes de script usando a função include, isso é particularmente útil para reaproveitamento de código, para versionamento e fácil migração de aplicações entre ambientes de teste, homologação e produção.

Vamos incluir, após a carga das planilhas, o pedaço de script que está no arquivo **Email.txt** a nossa aplicação.

```
$ (Include=[lib://Fonte de Dados/Email.txt]);
```

Dimensões de Tempo

Existem dimensões de tempo previstas no Plano de Projeto, precisamos criar **Ano, Mês, Dia, Trimestre e Mês Ano**.

Para criar **Ano, Mês, Dia e Mês Ano**, vamos usar um novo recurso chamado **Campos Derivados**, que possibilita criar campo e definições de grupo.

O **Trimestre**, vamos usar uma tabela **Inline**, depois iremos recriar a mesma tabela usando o **Autogenerate**.

Derivação de Campos

Declare

É usado para criar campo e definições de grupo, em que você pode definir as relações entre campos ou funções. Um conjunto de definições de campo pode ser usado para gerar campos derivados automaticamente, o que pode ser usado como dimensões. Você pode usar Declare para definir uma nova definição de campo ou criar uma definição de campo com base em uma definição já existente.

Sintaxe:

```
definition_name:
Declare [Field[s]] Definition [Tagged tag_list ]
[Parameters parameter_list ]
Fields field_list;
```

Derive

É usado para gerar campos derivados com base em uma definição de campo criada com um comando **Declare**. Você pode especificar a partir de quais campos de dados derivar os campos ou derivá-los explicitamente ou implicitamente com base em tags de campos.

Sintaxe:

```
Derive [Field[s]] From [Field[s]] field_list Using definition;
Derive [Field[s]] From Explicit [Tag[s]] tag_list Using definition;
Derive [Field[s]] From Implicit [Tag[s]] Using definition;
```

Crie uma nova seção na aplicação com o nome de **Calendário**, inclua o script abaixo e carregue os dados.

```
[Cal]:
Declare Field Definition Tagged '$date'
Parameters
  fmy = 1
Fields
  Year($1) as Ano Tagged ('$numeric','$integer'),
  Month($1) as Mês Tagged ('$numeric','$integer'),
  MonthName($1) as [Mês Ano]
    Tagged ('$numeric','$integer','$timestamp','$date'),
  Day($1) as Dia Tagged ('$numeric','$integer'),
  Date($1) as Data
    Tagged ('$numeric','$integer','$timestamp','$date'),
  Week($1) as Semana Tagged ('$numeric','$integer'),
  Weekday($1) as [Dia Semana] Tagged ('$numeric','$integer'),
  DayNumberOfYear($1, fmy) as [Número dia no ano]
    Tagged ('$numeric');

Derive Fields From Fields [DataPedido],[Data Admissao] Using Cal;
```

Load Inline

Em alguns casos, pode ser interessante inserir dados diretamente através do script. Geralmente fazemos isso para criar tabelas de Tipos, Status ou para resolver questões onde na tabela aparecem códigos como:

1 = Significa Ativo

2 = Significa Inativo

Se sabemos que o usuário não conhece o significado desses códigos, podemos criar uma tabela associada com a descrição desses códigos. Isso é feito com a ajuda do comando Load Inline.

Inclua o script abaixo logo após a declaração do **Derive** que fizemos anteriormente.

```
Trimestres:  
Load * Inline [  
    Mês, Trimestre  
    1, T1  
    2, T1  
    3, T1  
    4, T2  
    5, T2  
    6, T2  
    7, T3  
    8, T3  
    9, T3  
    10, T4  
    11, T4  
    12, T4  
];
```

Observe que o comando Load Inline contém os nomes dos campos e dados entre colchetes.

Observe também que os nomes dos campos estão localizados na primeira linha, e que os valores estão separados por vírgulas.

A tabela assim inserida associa números dos meses ao trimestre correspondente. Quando o script é executado, o novo campo Trimestre é gerado.

Mapping

A tabela Trimestres é útil, já que executa a ligação do mês ao respectivo trimestre. Entretanto, o campo se torna um campo chave quando associado e isso provavelmente causará problemas mais tarde. Existem algumas soluções para isso, como você verá a seguir.

O objetivo agora é trazer o campo Trimestre para dentro da tabela Pedidos. Para isso, você vai usar o recurso Mapping Table (Tabelas Mapeadas).

O prefixo Mapping é utilizado em um comando Load ou Select para criar uma Mapping Table.

Tabelas lidas através de Mapping Load ou Mapping Select serão armazenadas em área de memória separada e usada somente como Mapping Table durante a execução do script.

Após a execução do script, elas serão automaticamente eliminadas.

Mapping Table tem duas colunas:

- A primeira contém valor de comparação;
- A segunda, o resultado da comparação.

As duas colunas têm de receber nomes, mas os nomes são irrelevantes no modelo de dados. Os nomes dessas colunas não definem ligação com nomes de campos em tabelas “normais”, ou seja, pode ter o mesmo nome que outros campos em outras tabelas na memória ou que serão carregadas.

Quando tabelas mapeadas são usadas para mapear um valor de campo ou expressão, aquele valor será comparado com um valor na primeira coluna.

Se encontrado, o valor original será substituído por um valor correspondente na segunda coluna da tabela Mapping. Caso contrário, não haverá substituição.

Sintaxe: Mapping (load statement | select statement)

Agora, modifique o comando Load da tabela Trimestres em um comando de carregamento Mapping, como a seguir:

```

Trimestres:
Mapping Load * Inline [
    Mês, Trimestre
    1, T1
    2, T1
    3, T1
    4, T2
    5, T2
    6, T2
    7, T3
    8, T3
    9, T3
    10, T4
    11, T4
    12, T4
];

```

Ao realizar a carga novamente, poderá observar que a tabela Trimestres e seus campos não se encontram mais disponíveis, já que tabelas Mapping só existem durante o processo de carga.

Entretanto, podemos usar a tabela Trimestres no nosso script, mas lembre-se: desde que seja utilizado após essa definição no script. Para isso, utilizaremos a função ApplyMap().

Sintaxe: ApplyMap('mapname',expr,[.defaultexpr])

Essa função identifica qualquer definição em uma tabela Mapping, previamente carregada.

Mapname é o nome do mapa previamente carregada por um comando Mapping Load ou Mapping Select. Observe que o nome obrigatoriamente deve estar entre aspas.

Expr é a expressão ou somente o campo que o resultado deve ser mapeado.

Defaultexpr é uma definição opcional, que será usada como valor padrão, caso não exista valor associado a expr. Se um valor padrão não é indicado, o valor de expr retorna tal como ele é.

Vamos adicionar uma função ApplyMap() para a tabela Pedidos, com base no valor numérico do campo Mês. Essa função refere-se à tabela Trimestres.

A função deverá ficar como a seguir:

```

[Pedidos]:
LIB CONNECT TO [OLEDB-QWT];
LOAD [ClienteID],
    [DataPedido],
    ApplyMap('Trimestres', Num(Month([DataPedido]))) as Trimestre,
    [Frete],
    [FuncionarioID],
    [FuncionarioID] as FuncionarioVendedor,
    [PedidoID],
    [PedidoID] as CódigoPedido,
    [TransportadorID];
SQL SELECT `ClienteID`,
    `DataPedido`,
    `Frete`,
    `FuncionarioID`,
    `PedidoID`,
    `TransportadorID`;
FROM `Pedidos`;

```

Carregue os dados e veja o resultado.

Autogenerate

Outra forma de gerar dados é utilizar o comando Autogenerate no script de carga.

Quando você especifica Autogenerate no script de carga, isso automaticamente irá gerar um determinado número de registros.

Somente constantes e funções sem parâmetros são permitidas na utilização do Autogenerate.

Frequentemente, as funções RecNo() ou RowNo() são utilizadas para gerar um identificador (número único) para cada linha.

Faça alteração no script conforme abaixo.

```
Trimestres:  
Mapping Load RowNo() as Mês,  
    'T' & Ceil(RowNo() / 3) as Trimestre  
Autogenerate (12);
```

O **Autogenerate(12)** especifica que será criada uma tabela com 12 linhas.

A função **RowNo()** irá retornar o número da linha que está sendo criada na tabela lógica, iniciando com 1.

A função **Ceil()** vai arredondar o número indicado para o próximo inteiro superior.

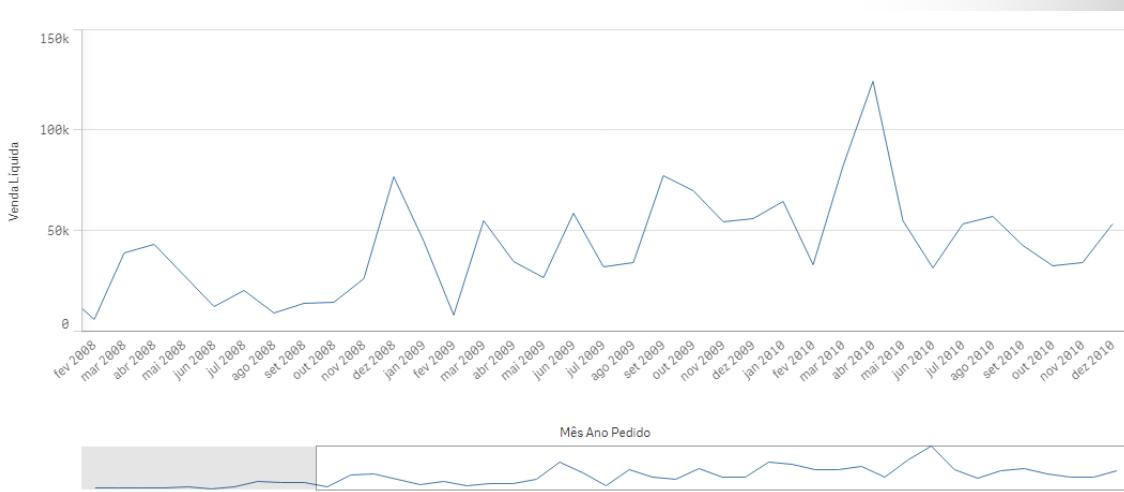
O caractere **&** é usado para concatenação.

Ficou fácil? Lembre-se: qualquer dificuldade que sintia, você deve retornar ao conteúdo, para depois seguir adiante!

Vamos exercitar! Realize as atividades referentes à Lição 7.

Laboratório 07 - Cálculos, Include e Dimensões de Tempo

1. Use o Mapping na tabela Transportadoras e aplique esse mapeamento para a tabela de Pedidos. Assim, os nomes das transportadoras estarão presentes diretamente na tabela Pedidos.
2. Crie um item mestre **dimensão**, com o campo derivado **DataPedido.Cal.Mês Ano**, dê o nome de **Mês Ano Pedido** e faça um item mestre **medida**, com a **soma** do campo **VendaLíquida**, dê o nome de **Venda Líquida** e uma tag **Vlr Venda**.
3. Faça um gráfico de linhas com os **Itens Mestres**: Mês Ano Pedido e Venda Líquida.



- Quais são as formas de gerar dados diretamente no script de carga?

Lição 8 - Tratamento de dados **não** padronizados

Objetivo da Lição

Nesta lição, você vai aprender a:

- Ler dados de arquivos não padronizados, contendo cabeçalho, rodapé, linhas vazias, linhas de totais e células mescladas.

Etapa de Transformação

Você pode observar que todos os dados que foram carregados até aqui estavam razoavelmente estruturados e puderam ser carregados através dos comandos Select ou Load.

Entretanto, nem todos os dados estão tão formatados para serem carregados.

Nesta parte do curso, vamos explorar algumas opções disponíveis para carregar dados em formatos não padronizados.

Um arquivo Excel (Orcamento.xls) contendo dados financeiros pode ser encontrado no diretório Fonte de Dados.

Esse arquivo contém dados em duas planilhas:

- Os dados existentes na primeira planilha podem ser facilmente carregados sem qualquer manipulação.
- Na segunda planilha, entretanto, os dados são apresentados em um formato frequentemente encontrado nas organizações.

Abra o arquivo Orcamento.xls que se encontra no diretório Fonte de Dados e veja a planilha **Orcamento**.

Esse formato de dados apresenta alguns desafios, mas pelo menos já podemos verificar que uma carga irá gerar campos para cada ano, em vez de valores anuais em cada campo Ano, como seria desejável em nossa aplicação.

	A	B	C	F	G	H	
	QWT Orçamento						
	Escriptorio	Metrica	2007	2008	2009	2010	Total
1		ReceitaOrcamento	\$40.000	\$90.000	\$153.000	\$202.000	\$485.000
		CustoOrcamento	\$65.000	\$75.000	\$105.000	\$160.000	\$405.000
		ReceitaReal	\$57.521	\$136.073	\$238.352	\$172.170	\$604.116
		CustoReal	\$73.000	\$80.000	\$102.000	\$145.000	\$400.000
2		ReceitaOrcamento	\$35.000	\$65.000	\$155.000	\$220.000	\$475.000
		CustoOrcamento	\$50.000	\$60.000	\$120.000	\$130.000	\$360.000
		ReceitaReal	\$25.991	\$75.568	\$205.078	\$157.298	\$463.935
		CustoReal	\$46.000	\$55.000	\$125.000	\$132.000	\$358.000
3		ReceitaOrcamento	\$15.000	\$30.000	\$60.000	\$70.000	\$175.000
		CustoOrcamento	\$45.000	\$46.000	\$48.000	\$50.000	\$189.000
		ReceitaReal	\$6.021	\$53.540	\$53.446	\$58.826	\$171.833
		CustoReal	\$43.000	\$43.000	\$45.000	\$46.000	\$177.000
4		ReceitaOrcamento	\$15.000	\$25.000	\$29.000	\$65.000	\$134.000
		CustoOrcamento	\$30.000	\$40.000	\$38.000	\$37.000	\$145.000
		ReceitaReal	\$6.492	\$24.772	\$32.779	\$34.355	\$98.398
		CustoReal	\$29.000	\$37.000	\$32.000	\$31.000	\$129.000
5		ReceitaOrcamento	\$13.000	\$38.000	\$60.000	\$72.000	\$183.000
		CustoOrcamento	\$22.000	\$27.000	\$31.000	\$34.000	\$114.000
		ReceitaReal	\$25.616	\$58.156	\$87.656	\$67.939	\$239.367
		CustoReal	\$20.000	\$26.000	\$33.000	\$32.000	\$111.000
		Total	\$662.641	\$1.085.109	\$1.753.311	\$1.916.588	\$5.417.649
		Média	\$33.132	\$54.255	\$87.666	\$95.829	\$270.882

Agora que você conheceu os desafios para alcançar dados padronizados, siga os passos:

- Arraste o arquivo **Orçamento.xls** para a aplicação e mude o tamanho do cabeçalho para 1.
- Observe os campos **Escriptorio** e **Total**, teremos de tratar isso diretamente no script.
- Carregue os dados e vá para o **Editor de carga de dados**.
- Desbloqueie a seção que foi gerada automaticamente e faça as alterações no script conforme abaixo.

```
[Orcamento$]:
LOAD [Escriptorio],
[Metrica],
[2007],
[2008],
[2009],
[2010]
FROM [lib://Fonte de Dados/Orcamento.xls]
(biff, embedded labels, header is 1 lines, table is Orcamento$, filters(
Replace(1, top, StrCnd(null)),
Rotate(left),
Remove(Row, Pos(Top, 1)),
Rotate(right)
)) Where IsNum([Escriptorio]);
```

Importante:

- Replace:** Preenche a célula com o valor superior (**top**) se estiver vazia (**null**).
- Rotate:** Gira os dados para a esquerda (**left**).
- Remove:** Apaga a primeira (**Top, 1**) linha (**Row**).
- Rotate:** Gira os dados para a direita (**right**), de volta a posição inicial.
- Where:** Com a função **IsNum** é mantida somente as linhas com números no campo escritório.

Achou difícil? Verifique o script atentamente e tire suas dúvidas. Vamos prosseguir!

Crosstable

Agora você está pronto para trabalhar com os **campos qualificadores**.

Um campo qualificador é simplesmente um campo normal, com dados em uma coluna.

Um campo qualificador será mantido como está em uma carga de tabela cruzada.

- Neste caso, os campos Escritorio e Metrica são campos qualificadores. Informe então que serão dois campos qualificadores.

Na carga de tabela cruzada, todos os campos qualificadores, obrigatoriamente, devem estar posicionados antes dos atributos e dos dados.

- Vamos usar o nome **AnoOrcamento** para identificar o campo de **Atributo** e o nome **Montante** para identificar o campo de **Dados**.

```
[Orcamento$]:
CrossTable(AnoOrcamento, Montante, 2)
LOAD [Escritorio],
    [Metrica],
    [2007],
    [2008],
    [2009],
    [2010]
FROM [lib://Fonte de Dados/Orcamento.xls]
(biff, embedded labels, header is 1 lines, table is Orcamento$, filters(
Replace(1, top, StrCnd(null)),
Rotate(left),
Remove(Row, Pos(Top, 1)),
Rotate(right)
)) Where IsNum([Escritorio]);
```

Queremos que as novas colunas incluídas **antes do total** na planilha também sejam carregadas, então, vamos fazer a alteração, incluindo o asterisco no script de carga que deve ficar como a seguir:

```
[Orcamento$]:
CrossTable(AnoOrcamento, Montante, 2)
LOAD *
FROM [lib://Fonte de Dados/Orcamento.xls]
(biff, embedded labels, header is 1 lines, table is Orcamento$, filters(
Replace(1, top, StrCnd(null)),
Rotate(left),
Remove(Row, Pos(Top, 1)),
Rotate(right)
)) Where IsNum([Escritorio]);
```

Observe no modelo de dados, que o campo **Montante** contém todos os valores para Orçamento e Real, mas preferimos ter esses valores em campos diferentes.

Para isso, necessitamos implementar um outro tipo de carregamento.

Generic Load

Uma base de dados genérica é aquela na qual os nomes dos campos são armazenados como valores de campos em uma coluna.

Tabelas de dados

Adicionar dados	
Lendo usando o script de carga de dados.	Orcamento\$
Escrítorios\$	Escritorio
8 campos	Metrica
Criada usando o script de carga de dados.	AnoOrcamento
EmailFuncionarios	Montante
2 campos	
Criada usando o script de carga de dados.	
Fornecedores	
9 campos	
Criada usando o script de carga de dados.	
Vendedores	
3 campos	
Criada usando o script de carga de dados.	
Orcamento\$	
4 campos	
Criada usando o script de carga de dados.	

Logo abaixo da carga do **Orçamento** com o comando **CrossTable** que você fez a pouco, inclua o script abaixo.

```
[Orcamento/Real] :
Generic Load
    Escritorio,
    AnoOrcamento,
    Metrica,
    Montante
Resident [Orcamento$];

DROP TABLE [Orcamento$];
```

Importante:

Metrica e Montante.

Obrigatoriamente o **penúltimo** campo deve conter a informação genérica e o **último** campo o valor propriamente.

Nesta lição, você aprendeu a ler dados de arquivos não padronizados, contendo cabeçalho, rodapé, linhas vazias, linhas de totais e células mescladas. Ficou tudo claro até aqui? Ótimo, vamos prosseguir!

Vamos praticar? Realize as atividades referentes à Lição 8.

Laboratório 08 - Tratamento de dados **não** padronizados

1. Na pasta Exercícios, existe um arquivo do Excel (tab01-IBGE.xls). É um dos arquivos disponíveis no site do IBGE e mostra um ranking dos 100 maiores municípios brasileiros em relação ao PIB (Produto Interno Bruto), dados de 2011.
2. Crie uma nova aplicação com o nome **100 Maiores Municípios Brasileiro**, lendo esse arquivo e extraiendo somente os campos:
 - o Cidade/Estado
 - o Valor do PIB
3. Use os recursos da etapa de transformação para não ler linhas e colunas desnecessárias
4. Depois da leitura do arquivo, utilize funções de manipulação de texto para separar a cidade do estado. Seu instrutor poderá ajudá-lo nessa tarefa.

Dica:

Use as funções `TextBetween()`, `left()`, `right()`, `len()` e/ou a função `subfield()` para separar a Cidade do Estado.

5. Sobre Crosstable, complete a lacuna com a alternativa correta:

Um campo qualificado é simplesmente um campo _____, com dados em uma _____.

- a) normal; coluna
- b) aberto; coluna
- c) normal; linha
- d) fechado; linha
- e) normal; linha

Lição 9 - Arquivos QVD, QVX e QVW

Objetivo da Lição

Nesta lição, você vai aprender a:

- Utilizar o poderoso recurso de geração de arquivos de alta performance.
- Entender os benefícios e as diferentes técnicas de trabalhar com QVD.

O que são Arquivos QVD

Um arquivo QVD (QlikView Data) é um arquivo que contém uma tabela de dados exportada dos produtos da Qlik. O QVD está em um formato nativo e só pode ser gravado e lido por esses produtos. O formato de arquivo é otimizado para agilização na leitura de dados de um script, ao mesmo tempo, é muito compacto. A leitura de dados de um arquivo QVD é geralmente de 10 a 100 vezes mais rápida do que a leitura de outras fontes de dados.

Formato

O arquivo QVD é uma tentativa de selar o compromisso do desempenho ideal dos produtos Qlik na leitura e gravação de arquivos e da representação compacta.

Um arquivo QVD contém exatamente uma tabela. Em termos conceituais, é muito semelhante a qualquer arquivo de tipo csv, dif, biff ou fix, e consiste em três partes:

- Um cabeçalho XML composto de modo apropriado (com o conjunto de caracteres UTF-8), que descreve os campos da tabela, o layout das informações subsequentes e alguns outros metadados.
- Tabelas de símbolos em um formato com bytes.
- Dados da tabela em um formato com bits.

Uso

Veja agora para que são usados arquivos QVD.

Aumentar a velocidade de Carga

Ao armazenar (buffering) dados com baixo nível de alterações ou sem alterações em arquivos QVD, a execução do script de carga pode tornar-se consideravelmente mais rápida quando trabalha com grandes conjuntos de dados.

Quando desenvolve aplicações, geralmente é necessário executar o script repetidamente. Ao utilizar dados armazenados em situações de repetição de carga, o arquivo QVD permite reduzir significativamente o tempo de espera mesmo em se tratado de conjuntos de dados de menor dimensão.

Consolidando dados de vários aplicativos.

Com o comando de script binary, é possível carregar dados de um único aplicativo em outro aplicativo, mas com os arquivos QVD, um script é capaz de combinar dados de vários aplicativos. Isso possibilita que os aplicativos consolidem dados semelhantes de diferentes unidades de negócios, por exemplo.

Diminuir a carga sobre Servidores de Base de dados.

O volume de dados lidos de fontes de dados externas pode também ser bastante reduzido. Isso reduz a carga de trabalho dos bancos de dados externos e o tráfego de rede. Além disso, quando vários scripts compartilham os mesmos dados, basta carregá-los uma vez do banco de dados de origem em um arquivo QVD. Os outros aplicativos podem usar os mesmos dados por meio desse arquivo QVD.

Carga Incremental

Em muitos casos comuns, a funcionalidade QVD pode ser usada para a carga incremental, para carregar apenas novos registros de um banco de dados crescente.

Criação de QVD

Os arquivos QVD podem ser criados:

- Manual a partir do script.
- Buffer, criação automática a partir do script.

Em seguida, você poderá ver uma descrição melhor cada uma das técnicas de criação de QVD.

Criação Manual a partir do Script

Um arquivo QVD pode ser criado por um comando **Store** no script. Esse é o comando que criará um arquivo QVD nomeado explicitamente.

Sintaxe:

- A <tabela> é uma tabela rotulada e já carregada do script.
- O <nome_do_arquivo> é interpretado de modo semelhante aos nomes nos comandos **load**, ou seja, os comandos **directory** se aplicam. Os campos da lista de campos podem ser renomeados usando a sintaxe As.

Exemplos:

```

Store Clientes
  into [lib://Fonte de Dados/Clientes.qvd] (qvd) ;

Store Produtos
  into [lib://Fonte de Dados/Produtos.csv] (txt, delimiter is ',');

Store [FuncionarioID],[Primeiro Nome] as Nome,[Escritorio]
  From [Funcionarios$]
  into [lig://Fonte de Dados/Funcionarios.skv] (txt, delimiter is ';' );

```

Ao incluir um comando **Store**, nenhuma alteração pode ser notada na sua aplicação e também que nenhuma nova tabela ou campo passou a existir.

O comando adicionado ao script não tem um efeito além de executar mais um comando.

Uma vez que esse script é executado, um novo arquivo de dados pode ser lido para a nossa aplicação ou mesmo qualquer outra aplicação com acesso à pasta onde está armazenado o arquivo.

Para fazer a carga de um arquivo QVD, basta arrastar ele para a aplicação ou qualquer outra forma válida de carga de arquivos de dados na aplicação.

Criação Automática a partir do Script

Os arquivos QVD podem ser criados e mantidos automaticamente usando o prefixo buffer. Esse prefixo pode ser usado com a maioria dos comandos Load e/ou Select no script. Ele indica se um arquivo QVD será usado para armazenar em cache/buffer o resultado do comando.

O nome atribuído ao arquivo QVD é calculado e normalmente armazenado em:

```

C:\ProgramData\Qlik\Sense\Engine\Buffers (instalação do servidor)
ou
C:\Users\{user}\Documents\Qlik\Sense\Buffers (Qlik Sense Desktop)

```

Sintaxe:

Buffer [(option [, option])] (loadstatement | selectstatement)

option::= incremental | **stale** [**after**] amount [(**days** | **hours**)]

Em que **option** é uma das seguintes:

Incremental

Permite a leitura apenas de parte de um arquivo subjacente. O tamanho anterior do arquivo é armazenado no cabeçalho XML no arquivo QVD. Isso é útil principalmente em arquivos de log. Todos os registros carregados anteriormente são lidos no arquivo QVD, ao passo que os novos registros subsequentes são lidos na fonte original, por fim, é criado um arquivo QVD atualizado.

Stale [After] amount [(days | hours)]

É geralmente usada com fontes de bancos de dados em que não há data/hora simples nos dados originais. Especifica-se por quanto tempo os dados capturados no último QVD poderão ser utilizados.

Exemplo:

```
[Clientes]:  
Buffer (Stale After 30 Days)  
SQL SELECT `Cidade`,  
          `ClienteID`,  
          `CodigoPostal`,  
          `Endereco`,  
          `Estado`,  
          `Fax`,  
          `Fone`,  
          `NomeContato`,  
          `NomeEmpresa`,  
          `Pais`  
FROM Clientes;
```

Vamos agora revisar o comando de carga da tabela **Clientes** para usar o método automático de geração de arquivo QVD.

Sabemos pelo nosso plano de projeto que os dados da tabela **Clientes** é atualizada semanalmente, então temos necessidade de ler e atualizar os dados a cada sete dias.

Vamos modificar então o nosso script de carga para adicionar o prefixo buffer ao nosso comando original de carga para Clientes.

Agora adicione o prefixo **Buffer (Stale After 7 Days)** ao comando de carga para **Clientes** similar ao exemplo a cima.

O que são Arquivos QVX

Um arquivo QVX (QlikView Data eXchanger) contém metadados que descrevem uma tabela de dados e os dados reais.

Diferente do formato QVD, que é patenteado e otimizado para transformações mínimas, o formato QVX é público e requer algumas transformações ao exportar dados de formatos de base de dados tradicionais.

Os arquivos QVX são carregados no script com o comando Load normalmente.

A documentação necessária para o desenvolvimento de arquivos QVX estão disponíveis nos materiais da Qlik.

O que são Arquivos QVW

Um arquivo QVW (QlikView Worksheet) pode ser carregado com o comando **Binary**, só podemos carregar dados de uma única aplicação QlikView em outra, de uma única aplicação QlikView em uma aplicação Qlik Sense ou de uma aplicação Qlik Sense em outra do mesmo Qlik Sense, porém nesse último caso, o comando **Binary** será em um arquivo QVF, equivalente o QVW.

Somente um comando **Binary** é permitido e obrigatoriamente é o primeiro comando executado pelo script. Quando executado, o script carrega os dados do arquivo correspondente, no entanto, sem usar a configuração do layout.

Nesta lição você aprendeu a utilizar Arquivos QVD, QVX e QVW, que é um poderoso recurso dos produtos da Qlik. Você entendeu também os benefícios e as diferentes técnicas de trabalhar com QVD.

Vamos exercitar! Realize as atividades referentes à Lição 9.

Laboratório 09 - Arquivos QVD

1. Modifique o script da sua aplicação para usar o buffer nas tabelas **Pedidos** e **DetalhesPedidos** utilizando a especificação **Stale**. Utilize o prazo de uma semana para determinar a frequência com que essas tabelas necessitam de novas cargas.
2. Crie uma nova aplicação e faça uma carga binária de arquivo QVW ou QVF de sua escolha.
3. Assinale apenas as alternativas que são usados arquivos QVD.
 Diminuir a carga sobre Servidores de Base de dados.
 Carga Incremental.
 Diminuir a velocidade de Carga.
 Aumentar a velocidade de Carga.
 Aumentar a carga sobre Servidores de Base de dados.

Lição 10 - Script Avançado

Objetivo da Lição

Nesta lição, você vai aprender a:

- Conhecer e aplicar técnicas de tratamento de dados que serão utilizados nos processos de Transformação dos dados extraídos no script.

Script Avançado

Nesta lição usaremos algumas técnicas avançadas de script para criar uma nova tabela com dados agregados, provenientes de duas novas tabelas. Iremos também comparar os valores de um campo em relação ao registro anterior.

Para trazer dados de duas tabelas diferentes, usaremos o comando **Join**. Já para agregar os dados, usaremos o comando de agregação **Sum** juntamente com a cláusula **Group by**. E para comparar os dados em relação ao registro anterior usaremos a função **Previous** juntamente com o **Order by**.

Finalmente, você aprenderá a utilizar variáveis para otimizar a manutenção do script.

Join

O comando Join é utilizado para unir dados que estão em tabelas diferentes.

O exemplo que veremos a seguir usará o Join na criação de uma nova tabela que utiliza dados de outras duas tabelas existentes.

Vamos entender o caso que aplicaremos o Join.

Para compararmos as Vendas dos Clientes entre os dois anos anteriores, poderíamos simplesmente selecionar os anos individualmente na interface da aplicação e comparar o gráfico resultante ou incluir Ano como dimensão em um gráfico calculando a Venda Líquida.

Isso pode ser suficiente para algumas de nossas necessidades, mas para determinar a média de performance do Cliente, vamos criar dados adicionais (uma nova tabela) demonstrando anualmente a performance da Venda Líquida no script de carga.

Assim, você poderá determinar o crescimento ou a redução da performance por Cliente.

Utilize o campo calculado **VendaLiquida** da tabela **DetalhesPedidos**, mas restrinja para dois anos que são apresentados no campo **DataPedido** e para os **Clientes** no campo **ClientesID** na tabela **Pedidos**.

Para usar as duas tabelas, temos de combiná-las em uma única tabela.

Já que os dados que necessitamos já se encontram carregados em memória, vamos utilizar o comando **Join Load** do indexador sobre essas duas tabelas que estão residentes na memória (**DetalhesPedidos** e **Pedidos**).

Siga os passos a seguir para criar essa nova tabela.

- Adicione uma nova seção ao script, identificando como Variação Venda.
- Certifique-se que seja a última seção do script, basta arrastar para o fim.
- Primeiro, necessitamos carregar uma nova tabela lógica com base na tabela residente **Pedidos**.
- Utilize **PedidosPorAnoOrigem** como nome para essa tabela, conforme o script abaixo.

```
/* Variação de Vendas de dois anos */
PedidosPorAnoOrigem:
Load ClienteID,
    PedidoID,
    Year(DataPedido) as Ano
Resident Pedidos
Where Year(DataPedido) = 2009
    or Year(DataPedido) = 2010;
```

- Os campos **ClienteID** e **Ano** serão usados para agrupar os dados.
- O campo **PedidoID** será usado para fazer a ligação com a tabela **DetalhesPedidos**, quando você executar o **Join** para aquela tabela.
- A cláusula **Where** limitará os dados para apenas os anos de **2009** e **2010**.

- Adicione os dados da **Venda Líquida** a esta tabela através de um **Join Load**.
- Adicione o seguinte comando ao script:

```
/* Junção da Venda Líquida à tabela que contém apenas dois anos de vendas */
Left Join (PedidosPorAnoOrigem)
Load PedidoID,
        VendaLiquida
Resident DetalhesPedidos;
```

O comando **Left Join Load** foi utilizado porque só queremos utilizar a ligação em pedidos armazenados com base na cláusula **Where** especificada no carregamento anterior.

Você deve assegurar que o campo **PedidoID** está incluído nesse comando **Load** para termos associação com os registros da tabela **PedidosPorAnoOrigem**.

O comportamento padrão do **Join** é o de um **full outer join**, por essa razão, se não existirem campos nas duas tabelas que passaram pelo comando **Join**, teremos um produto cartesiano de registros.

Já que estamos especificando **PedidoID** em ambas tabelas, e estamos especificando como um **Left Join**, somente os registros que combinem pelo campo **PedidoID** existente na tabela **PedidosPorAnoOrigem** serão incluídos.

Incluímos o campo **VendaLiquida** porque esse é o campo que queremos acrescentar na tabela **PedidosPorAnoOrigem** que estamos criando.

Para que possamos visualizar os dados da nova tabela que geramos, podemos acessar o visualizador do modelo de dados.

Agregação

Para agrupar ou agregar dados, você deve utilizar a cláusula **Group by** no comando **Load**, juntamente com o comando de agregação **Sum**.

Neste caso, necessitamos agregar os dados da tabela **PedidosPorAnoOrigem** usando os campos **ClienteID** e **Ano**.

Para implementar essa agregação, siga os passos:

- Adicione o seguinte comando logo após o código que fizemos o join:

```
/* Agregação por ClienteID e Ano */
PedidosPorAno:
Load ClienteID,
        Ano,
        Sum(VendaLiquida) as VendaLiquidaAnual
Resident PedidosPorAnoOrigem
Group by ClienteID, Ano;

Drop Table PedidosPorAnoOrigem;
```

- Eliminamos a tabela **PedidosPorAnoOrigem** que utilizamos no processo do comando **Join**, com o comando **Drop Table**, pois ela se tornou desnecessária, já que temos a informação que precisamos.

Se você observar no modelo de dados, o campo **ClienteID**, visualizará dois valores associados nesse novo campo (as duas somas da Venda Líquida agregados para cada ano).

Previous e Order by

Para calcular a alteração anual de cada cliente, você pode carregar esses dados novamente, usando a cláusula **Order by**, para ordenar os registros apropriadamente, e a função **Previous**, para acessar os dados do registro anterior.

Adicione o script abaixo, após a última linha do código que fizemos a cima.

```
/* Crescimento / Queda das vendas por ClienteID e Ano */
VariacaoVenda:
Load ClienteID,
    Ano,
    VendaLiquidaAnual,
    If(ClienteID = Previous(ClienteID),
        If(VendaLiquidaAnual > Previous(VendaLiquidaAnual),
            'Crescimento','Queda')
        ,null()) as VariacaoVendaIndicador,
    If(ClienteID = Previous(ClienteID),
        VendaLiquidaAnual - Previous(VendaLiquidaAnual)
        ,null()) as VariacaoVendaValor
Resident PedidosPorAno
Order By ClienteID, Ano;

Drop Table PedidosPorAno;
```

Também já eliminamos a tabela **PedidosPorAno**, por ela também já ser desnecessária daqui por diante.

Observe a cláusula **Order by** no fim do comando. Isso fará com que os registros sejam lidos na ordem necessária: **ClienteID** e **Ano**.

O padrão de ordenação é ascendente, então o ano de **2009** para um dado cliente (se existir) será carregado primeiro.

A primeira <condição>:

ClienteID = Previous(ClienteID)

Irá comparar o valor do registro corrente do campo **ClienteID** com o registro anterior que não foi descartado pela cláusula **Where**.

Em outras palavras, essa condição verifica se ambos os registros são do mesmo cliente.

A segunda <condição>:

VendaLiquidaAnual > Previous(VendaLiquidaAnual)

Compara o valor corrente do campo **VendaLiquidaAnual** com o valor anterior.

Então, se o cliente é o mesmo, e **VendaLiquidaAnual** é maior em **2010** do que em **2009**, o resultado do campo **VariacaoVendaValor** será “Crescimento”.

Se **ClienteID** é diferente, este campo será alimentado com o resultado da função **Null**.

O campo **VariacaoVendaValor** usa uma função **If** similar, é uma estrutura de **If's** em cadeia.

Neste caso, se cliente é o mesmo entre registros, **VendaLiquidaAnual** em **2010** e **VendaLiquidaAnual** em **2009** serão calculados, de outro modo, o campo será alimentado com o resultado da função **Null**.

Uso de Variáveis

É importante conceber seus scripts de carga para que sejam flexíveis e simples de manter. Uma das formas de se alcançar esse objetivo é o uso de variáveis.

Podem ser atribuídas com uma **constante** (número ou texto), ou pode ser atribuído um valor através de uma **expressão**.

Utilize o comando **Set** para atribuir um valor diretamente, e o comando **Let** para atribuir um valor através de uma expressão.

A utilização de nomes de variáveis únicas e que permitam uma fácil identificação também é considerada uma boa prática.

Variáveis definidas no script de carga estão disponíveis na interface para o usuário. Neste caso, vamos atribuir um valor ao ano.

Anteriormente, escrevemos diretamente o valor 2009 e 2010 no script.

Primeiro, vamos atribuir um valor direto a uma variável, e então você mudará para uma dinâmica com base na data corrente do sistema.

Siga os passos:

- Na primeira seção do script, geralmente com o nome de **Main**, inclua o código abaixo, preferencialmente na última linha dessa seção (*É uma boa prática usar essa seção apenas para definições de variáveis*).

```
SET vAnoAnteriorCompleto = 2010;
```
- No script onde fizemos a carga da tabela **PedidosPorAnoOrigem**, precisamos editar a cláusula **Where**.

```
/* Variação de Vendas de dois anos */  
PedidosPorAnoOrigem:  
    Load ClienteID,  
        PedidoID,  
        Year(DataPedido) as Ano  
    Resident Pedidos  
    Where Year(DataPedido) = $(vAnoAnteriorCompleto)-1  
        or Year(DataPedido) = $(vAnoAnteriorCompleto);
```
- Modifique-a conforme abaixo.

```
/* Variação de Vendas de dois anos */  
PedidosPorAnoOrigem:  
    Load ClienteID,  
        PedidoID,  
        Year(DataPedido) as Ano  
    Resident Pedidos  
    Where Year(DataPedido) = $(vAnoAnteriorCompleto)-1  
        or Year(DataPedido) = $(vAnoAnteriorCompleto);
```

Será alimentado o valor de **vAnoAnteriorCompleto** quando for executado. Neste caso, o valor será **2010**.

- Não deverá haver modificações nos valores dos campos após essa nova versão do script ser executada.

Em vez de exigir uma alteração todos os anos para manter esse script, você pode definir um valor para a variável usando a data de sistema para alimentar esse valor.

Já que essa atribuição irá requerer uma expressão, devemos modificar o nosso comando **Set** por um comando **Let**.

- Modifique o comando que define a variável **vAnoAnteriorCompleto** para o comando **Let** conforme abaixo.

```
LET vAnoAnteriorCompleto = Year(Today())-5;
```

Esse comando utilizará o ano atual definido no sistema (a data de execução do script) e subtrair 5 para obter um período de ano anterior que possui informações para análise.

Nesta lição você conheceu e aprendeu a aplicar técnicas de tratamento de dados que serão utilizados nos processos de transformação dos dados extraídos no script. Lembre-se, em caso de dúvidas retome o conteúdo, para depois seguir seu estudo com tranquilidade.

Vamos exercitar! Realize as atividades referentes à Lição 10.

Laboratório 10 - Script Avançado

1. Crie uma pasta na aplicação com o nome de **Variação da Venda**.
 - a. Inclua uma tabela dinâmica com a dimensão **Cliente** e **Ano**.
 - b. Inclua a expressão com a **Soma da Venda Líquida Anual** e outra expressão com a **Soma da Variação do Valor da Venda** e o rótulo de **Variação da Venda Líquida de 2009 para 2010**.
2. Configure a expressão da cor do fundo da variação da venda com a condicional abaixo.

```
If(VariacaoVendaIndicador='Crescimento',
    RGB(134,219,134),
    If(VariacaoVendaIndicador='Queda',
        RGB(255,104,104)
    )
)
```

Após alguns ajustes teremos a seguinte tabela:

Variação da Venda						
Indicador	Crescimento / Queda das Vendas de 2008 à 2010					
	Ano		Medidas			
	2008		2009		2010	
Valor Variação	Venda	Valor Variação Ano Anterior	Venda	Valor Variação Ano Anterior	Venda	Valor Variação Ano Anterior
R\$ 46337,86	-	-	R\$ 193,36	R\$ 0,00	-	-
R\$ 44357,88	R\$ 2.213,50	R\$ 0,00	R\$ 3.347,35	R\$ 1.133,86	R\$ 1.923,87	-R\$ 1.423,48
R\$ 26412,03	-	-	R\$ 873,75	R\$ 0,00	R\$ 658,07	-R\$ 214,78
R\$ 25698,82	-	-	R\$ 2.859,95	R\$ 0,00	R\$ 478,23	-R\$ 2.381,73
R\$ 23365,85	-	-	R\$ 26,70	R\$ 0,00	R\$ 8.651,14	R\$ 8.624,44
R\$ 23084,88	R\$ 1.421,16	R\$ 0,00	-	-	R\$ 8.393,44	R\$ 6.972,28
R\$ 21681,14	-	-	R\$ 1.569,13	R\$ 0,00	-	-
	R\$ 24.842,59	R\$ 0,00	R\$ 38.334,12	R\$ 13.491,53	R\$ 43.180,66	R\$ 4.846,54
	R\$ 17.656,53	R\$ 0,00	R\$ 3.971,48	-R\$ 13.685,08	R\$ 14.689,08	R\$ 10.717,60
	R\$ 7.468,90	R\$ 0,00	R\$ 27.831,97	R\$ 19.563,07	R\$ 17.815,10	-R\$ 14.216,68

3. Assinale a alternativa que completa a seguinte frase:

Para calcular a alteração anual de cada cliente, você pode carregar esses dados novamente, usando a cláusula _____, para ordenar os registros apropriadamente, e a função _____, para acessar os dados do registro anterior.

- a. Order by; ClientID
- b. Order by; Previous()
- c. Previous(); Order by
- d. Venda_Liquida; Order by
- e. ClientID; Previous()

Lição 11 - Concatenação

Objetivo da Lição

Nesta lição, você vai aprender a:

- Conhecer esse poderoso recurso de união de dados, verificando os casos onde a concatenação se dará automaticamente.
- Forçar ou prevenir (evitar) a concatenação.

Concatenação Automática

Se os nomes e o número de campos das duas tabelas são exatamente iguais, automaticamente, será concatenado o resultado dos diferentes comandos Load ou Select em uma tabela.

Exemplos:

```
LOAD a, b, c from table1.csv;
```

```
LOAD a, c, b from table2.csv;
```

```
LOAD c, b, a from table3.csv;
```

A tabela lógica resultante terá os campos **a**, **b** e **c**.

O número de registros é a somatória do número de registros da tabela 1, tabela 2 e tabela 3.

As figuras a seguir exemplificam como se aplica a concatenação automática:

Tabela 1, primeira tabela lida

A	B	C
A1	B1	C1
A2	B2	C2
A3	B3	C3

Tabela 2, segunda tabela lida

A	C	B
A10	C10	B10
A11	C11	B11

Tabela resultante

A	B	C
A1	B1	C1
A2	B2	C2
A3	B3	C3
A10	B10	C10
A11	B11	C11

Observe que o rótulo da tabela resultante será o mesmo da primeira tabela lida.

Importante:

- A **quantidade** e os nomes dos campos devem ser exatamente os **mesmos**.
- Independente da ordem dos dois comandos e dos campos, o resultado será o mesmo.

Concatenação Forçada

Mesmo se duas ou mais tabelas não tenham exatamente o mesmo conjunto de campos, ainda será possível forçar a concatenação das duas tabelas. Isso é feito usando o prefixo concatenate no script, que concatena uma tabela a outra nomeada ou à última tabela lógica criada anteriormente.

Exemplo 1:

```
LOAD a, b, c from table1.csv;
Concatenate LOAD a, c from table2.csv;
Concatenate LOAD b, a from table3.csv;
```

A tabela lógica resultante tem os campos **a**, **b** e **c**.

O número de registros na tabela resultante é a soma do número de registros na tabela 1, tabela 2 e na tabela 3.

O valor do campo **b** nos registros vindos da tabela 2 e o do campo **c** na tabela 3 é **NULL**.

As figuras a seguir exemplificam como se aplica a concatenação automática:

Tabela 1, primeira tabela lida

A	B	C
A1	B1	C1
A2	B2	C2
A3	B3	C3

Tabela 2, segunda tabela lida

A	C
A10	C10
A11	C11

Tabela resultante

A	B	C
A1	B1	C1
A2	B2	C2
A3	B3	C3
A10	-	C10
A11	-	C11

Importante:

- A **quantidade** e os nomes dos campos podem ser **diferentes**.
- A menos que o nome de uma tabela carregada anteriormente seja especificado no comando **concatenate**, o prefixo **concatenate** utilizará a última tabela criada anteriormente.
- Dessa forma, a ordem dos dois comandos é **muito importante**.

Prevenir Concatenação

Se duas tabelas têm o mesmo conjunto de campos, isso levaria a que elas fossem automaticamente concatenadas.

Podemos prevenir essa situação com o prefixo **Noconcatenate**.

A utilização desse recurso previne a concatenação de qualquer tabela lógica com o mesmo conjunto de campos:

Sintaxe:

```
NoConcatenate( loadstatement | selectstatement )
```

Exemplo:

```
LOAD a, b, c from table1.csv;  
NoConcatenate LOAD a, c, b from table2.csv;  
NoConcatenate LOAD c, b, a from table3.csv;
```

Será gerada três tabelas interna resultante com os campos **a**, **b** e **c**.

O número de registros será correspondente a origem em cada tabela.

Será gerada uma chave sintética entre as três tabelas, ou seja, haverá quatro tabelas no modelo de dados

Importante:

- A **quantidade** e os nomes dos campos devem ser exatamente os **mesmos**, se forem diferentes não há o que ser evitado, logo o **NoConcatenate** não deve ser usado.
- A concatenação será evitada independentemente de onde a definição da tabela carregada anteriormente esteja.
- Dessa forma, a ordem dos dois comandos é **muito importante**.

Concatenação na prática

Nos nossos dados, temos disponível um conjunto adicional de colaboradores que não existiam no arquivo **FuncionariosEscritorios.xls**.

Para inserir esses dados, vamos modificar nosso script de carga. Siga os passos:

- Localize o script de carga onde fizemos a carga da planilha **Funcionários** do arquivo do Excel **FuncionariosEscritorios.xls**.
- Modifique a cláusula **From** conforme abaixo.

```
[Funcionarios$]:
LOAD [FuncID] AS [FuncionarioID],
[Sobrenome],
[Primeiro Nome],
[Titulo],
[Data Admissao],
[Escriptorio],
[Ramal],
[Reporta a],
[Salario Anual]
FROM [lib://Fonte de Dados/Funcionarios*.xls]
(biff, embedded labels, table is Funcionarios$);
```

- Carregue os dados e observe no visualizador de tabelas, que como os dados tinham exatamente a mesma estrutura, as duas tabelas foram automaticamente concatenadas, mesmo estando em pastas de trabalho diferentes.

Nesta lição você conheceu a concatenação e verificou os casos onde ela se dará automaticamente. Aprendeu também a forçar ou prevenir (evitar) a concatenação. Está tudo certo até aqui? Vamos prosseguir.

Vamos praticar? Realize as atividades referentes à Lição 11.

Laboratório 11 - Concatenação

1. Crie uma nova aplicação e dê o nome de **Concatenação**.
 - a. Abra o Excel, BrOffice ou Google Docs, como preferir, e crie um arquivo **Produtos1.xls** com os seguintes dados:

Código	Nome	Categoria	Valor
1	Arroz	Alimento	4
2	Feijão	Alimento	5
3	Camisa	Vestuário	20

- b. Altere os dados conforme abaixo e salve como **Produtos2.xls**.

Código	Nome	Categoria	Valor
4	Bermuda	Vestuário	10
5	Macarrão	Alimento	3
6	Sapato	Calçados	30

- c. Altere os dados conforme abaixo e salve como **Produtos3.xls**.

Código	Nome	Valor
7	Refrigerante	4
8	Cerveja	200
9	Picanha	5

* Observe que este último arquivo está sem a coluna **Categoria**.

- d. Arraste o primeiro arquivo **Produtos1.xls** para a aplicação, deixe todos os campos marcados e **carregue os dados**.
- e. Carregue os dados e crie uma pasta, na aplicação, com uma lista para cada campo e também um objeto tabela com todos os campos.

- f. Arraste o segundo arquivo **Produtos2.xls**, deixe todos os campos marcados e **carregue os dados**.
- g. Verifique os dados nas listas e na tabela, verifique também o **Visualizador do Modelo de Dados**.
- h. Arraste também o terceiro arquivo **Produtos3.xls** para a aplicação, deixe todos os campos marcados e **carregue os dados**.
- i. Verifique que na aplicação os dados foram carregados corretamente, mas veja como ficou o **Visualizador do Modelo de Dados**.
- j. Vá ao **Editor de Carga de Dados**, desbloqueie a seção que foi gerada automaticamente e acrescente o prefixo **Concatenate** antes do **Load** do arquivo **Produtos3.xls**.
- k. Depois de carregar os dados, veja como ficou o visualizador de modelo de dados.

Plan1

Código	Nome	Categoria	Valor
1	Arroz	Alimento	4
2	Feijão	Alimento	5
3	Camisa	Vestuário	20
-	-	-	-
4	Bermuda	Vestuário	10
5	Macarrão	Alimento	3
6	Sapato	Calçados	30

2. Observe a seguinte explicação e assinale a que conceito corresponde.

Se duas ou mais tabelas não têm exatamente o mesmo número de campos, ainda é possível forçar a concatenação das duas tabelas. Isso é feito com o prefixo **concatenate** no script, que concatenará uma tabela com outra com outro nome ou com uma tabela lógica previamente criada.

- Concatenação automática.
- Prevenir concatenação.
- Concatenação de campos.
- Concatenação forçada.
- Concatenação de tabelas.

Lição 12 - Uso de mapas

Objetivo da Lição

Nesta lição, você vai aprender a:

- O que é necessário para analisar dados geograficamente.
- Utilizar mapas para fazer descoberta de informações ainda mais relevantes.

Keyhole Markup Language (kml)

É um padrão, baseado em XML e serve para representar anotações geográficas e visualização de conteúdos existentes. Esse padrão foi desenvolvido para uso com o [Google Earth](#), que era originalmente chamado de Keyhole Earth Viewer. Criado por [Keyhole, Inc](#), e que mais tarde foi adquirida pelo Google em 2004.

Arquivos KML são muito frequentemente distribuídos como pacotes **KMZ**, que são arquivos KML compactados com outros tipos de arquivos juntos.

Exemplo de KML:

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Placemark>
    <name>Toccato Tecnologia</name>
    <description>Fica no Corporate Park,
    em Santo Antônio de Lisboa.</description>
    <Point>
      <coordinates>-48.513048,-27.507459,0</coordinates>
    </Point>
    <Polygon>
      <tessellate>1</tessellate>
      <outerBoundaryIs>
        <LinearRing>
          <coordinates>
            -48.51307633112951,-27.50741903932523,0
            -48.51305369723773,-27.50750287100037,0
            -48.51296202023312,-27.50749215859556,0
            -48.51297896887919,-27.50740413745671,0
            -48.51307633112951,-27.50741903932523,0
          </coordinates>
        </LinearRing>
      </outerBoundaryIs>
    </Polygon>
  </Placemark>
</kml>
```

Para mais informações: <https://developers.google.com/kml/documentation>

Preparando os dados

Iremos usar os arquivos **TodosMunicipios.kml**, **Brasil-Estados.kml** e **MunicipiosDetalhes.qvd** presentes na pasta do curso.

- Arraste o arquivo **TodosMunicipios.kml** para a aplicação e defina os respectivos nomes dos campos: **Código IBGE**, **Municípios Ponto** e **Municípios Área** conforme abaixo e carregue os dados.

Código IBGE	Municípios Ponto	Municípios Área
1100015	[-62.3893014000,-12.4730069850]	[-62.8916198400,-12.8584379700][-62.8886478400,-12.86276400]
1100023	[-63.8192445200,-9.9786509500]	[-63.1816610000,-9.7521679800][-63.0335649600,-9.7514219700]
1100031	[-60.6444964200,-13.5057345300]	[-60.9184551000,-13.5479160000][-60.9181430400,-13.55339700]
1100049	[-61.3976786000,-11.3029209900]	[-61.7952711000,-11.3974329600][-61.7825098800,-11.40158400]
1100056	[-61.3117211400,-13.1836099950]	[-61.5084678800,-13.0039229700][-61.4790106800,-13.01098200]
1100064	[-60.5375406000,-13.1365239750]	[-60.6514910400,-12.9808999800][-60.6253471200,-12.97258700]
1100072	[-61.0788704400,-12.8806119900]	[-61.3375419600,-12.6261559800][-61.3313161200,-12.63758600]
1100080	[-64.0691386200,-12.0853184500]	[-64.5940181600,-12.2153020200][-64.5862028400,-12.18463100]
1100088	[-60.8228656200,-11.3683780350]	[-61.2111981600,-11.5287018200][-61.2082586400,-11.52857100]
1100106	[-64.4741058600,-11.3219035200]	[-65.3809550400,-10.4286859200][-65.3598118800,-10.43378100]
1100114	[-62.6331696000,-10.6484160150]	[-62.9196338400,-10.8375219900][-62.9141146800,-10.83518600]
1100122	[-61.8125956200,-10.4483835900]	[-62.1228559400,-10.6989900100][-62.0194798800,-10.64633600]
1100130	[-62.0110999800,-9.1897974900]	[-62.5542271200,-9.6703779600][-62.5365608400,-9.6743379600]
1100148	[-62.1973881000,-11.5598264850]	[-62.4387020400,-11.6153100000][-62.4253100400,-11.59718600]
1100155	[-62.1764446200,-10.5078204900]	[-62.5247290800,-10.7090769800][-62.4739071600,-10.66454600]
1100189	[-60.9378694200,-11.8621544850]	[-61.5484508400,-11.6187059700][-61.4368749600,-11.61806600]

- Se observarmos o modelo de dados, o KML está “solto”, vamos arrastar também o arquivo QVD com os detalhes dos municípios, alterar os campos, **UF** para **Estado** e **Município** para **Cidade**.
- Observe que o campo **Código IBGE** já está igual ao que foi definido no KML.
- Arraste também o arquivo **Brasil-Estados.kml** e defina os respectivos nomes dos campos: **Estado**, **Estado Ponto** e **Estado Área** e carregue os dados.
- É importante lembrar de salvar a aplicação sempre, pois estamos em um ambiente de páginas de internet, mesmo usando a versão desktop.

Montagem do Gráfico

Vamos agora montar o gráfico que exibirá nosso mapa.

- Crie uma pasta com o nome **Mapa**, arraste o **gráfico mapa**, inclua a dimensão **Cliente** e selecione o campo geográfico **Municípios Área** (**área**).

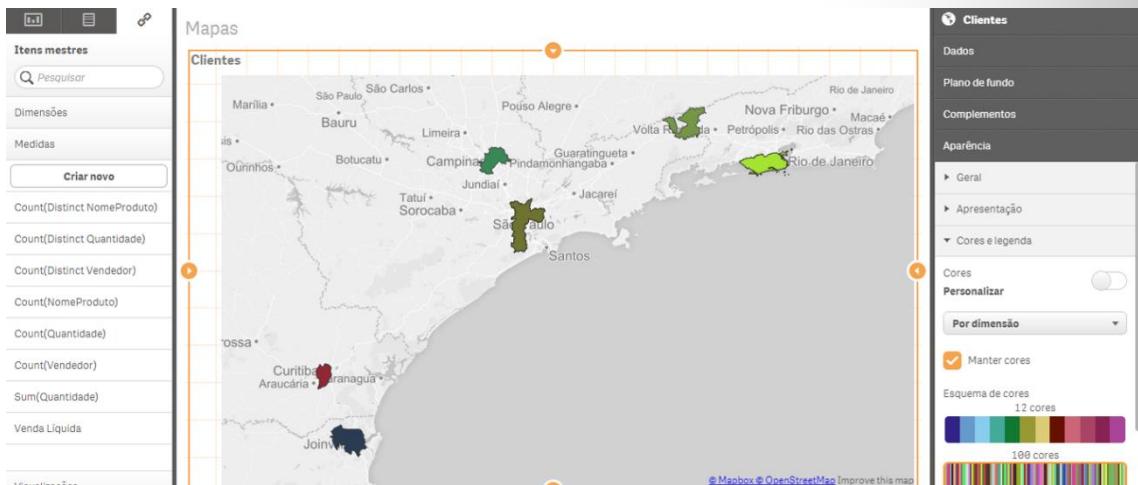
- Há uma limitação quanto a quantidade de itens que podem ser plotados no gráfico de mapa, para contornar essa limitação, podemos restringir a carga aos municípios os quais temos clientes ou segmentar os dados geoespaciais.
- Abra o **editor de carga de dados** e desbloqueie a seção que foi gerada automaticamente com os dados do KML.
- Renomeie a seção com o nome de **Geolocalização** e faça as alterações no script conforme abaixo.

```
[MunicipiosDetalhes]:
Left Keep (Clientes)
LOAD [UF] AS [Estado],
    [Municipio] AS [Cidade],
    [Altitude],
    [Código IBGE],
    [Microregião],
    [Area (km²)],
    [Mesoregião]
FROM [lib://Fonte de Dados/MunicipiosDetalhes.qvd]
(qvd);

[KMLMunicípios]:
Left Keep (MunicipiosDetalhes)
LOAD [TodosMunicípios.Name] AS [Código IBGE],
    [TodosMunicípios.Point] AS [Municípios Ponto],
    [TodosMunicípios.Area] AS [Municípios Área]
FROM [lib://Fonte de Dados/TodosMunicípios.kml]
(kml, Table is [Munic.shp/Features]);
```

A ordem é importante, por tanto, confira se primeiro carregou os detalhes para só depois carregar o KML.

- Carregue os dados e volte a pasta **Mapas** para fazer algumas configurações.
- Ative a opção de **mostrar o plano de fundo** e mude a **aparência** para que as **cores** sejam mantidas **por dimensão** e o esquema seja de **100 cores**, conforme abaixo.



Vamos praticar? Realize a atividade referente a lição 12.

Laboratório 12 – Uso de Mapas

1. Crie um mapa do Brasil de área, mostrando a venda dos vendedores com esquema de cores que mostre, quanto mais vermelho, o estado que vende menos e azul o que vende mais?
2. Assinale **V** em todas as alternativas corretas e **F** nas incorretas.
 - () Para montar mapas de polígonos, precisamos apenas da latitude e longitude do município.
 - () Para montar mapas de polígonos, precisamos da latitude e longitude de cada vértice do polígono.
 - () Para criar mais de um mapa em uma única pasta, precisamos refazer o script de carga.
 - () Para criar um mapa, basta arrastar o gráfico do tipo mapa para a pasta e configurar a dimensão por área ou por ponto.

Lição 13 - Depuração

Objetivo da Lição

Nesta lição, você vai aprender a:

- Conhecer os recursos do Depurador de Script para saber utilizá-lo para localizar problemas no script.

Depuração

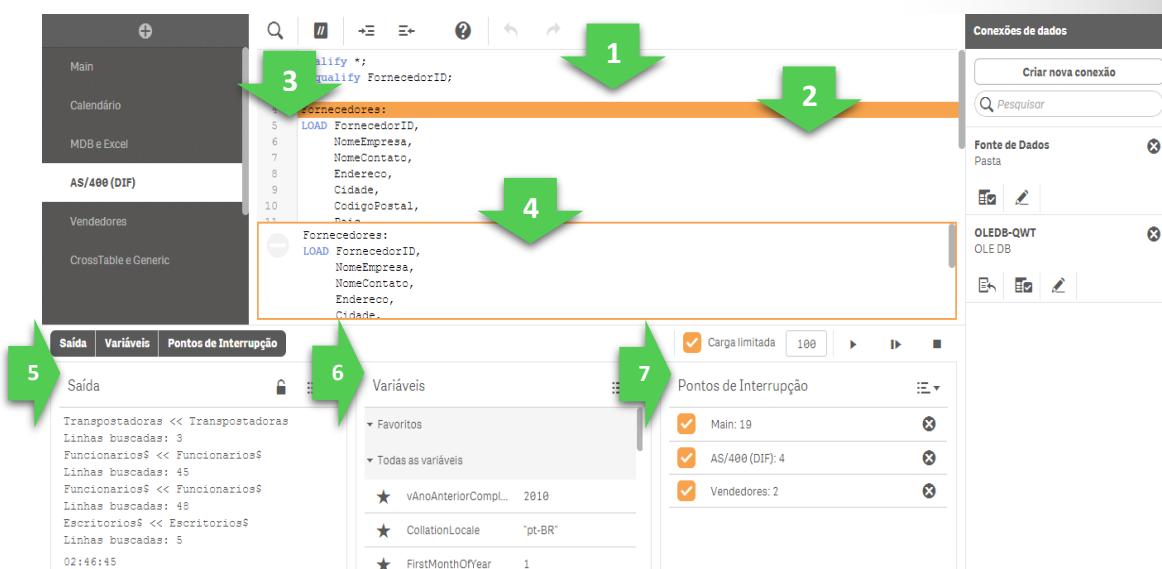
Você pode usar os utilitários de depuração no editor de script de carga para percorrer a execução do seu script usando pontos de interrupção, podendo assim inspecionar os valores variáveis e a saída da execução do script.

É possível selecionar se você deseja ver uma ou todas as Saídas, Variáveis e Pontos de interrupção.

Importante:

Você **não** pode **criar** conexões, **editar** conexões, **selecionar** dados, **salvar** o script ou **carregar** dados enquanto estiver executando no modo de depuração, isto é, desde quando começou a execução da depuração até que o script seja executado ou a execução termine.

Para entrar no modo de depuração, basta clicar no ícone que está ao lado direito, **Mostrar Painel de Depuração**.



A imagem mostra ao centro o script (1).

A posição da execução é marcada por uma linha laranja (2).

É possível definir pontos de interrupção **clicando no número da linha**, à esquerda de um comando (3).

Após clicar no número da linha, o **ponto de interrupção** é marcado por um ícone laranja (3).

Um quadrado sobre o script (4) mostra o comando que está prestes a ser executado no modo **incremento** ou que está sendo executado no modo **executar**.

A aba **Saída** (5) mostra o status e possíveis mensagens de erro, basicamente as mesmas informações que no diálogo de progresso do script.

A aba **Variáveis** (6) mostra todas as variáveis e respectivos valores, as variáveis alteradas são mostradas em ordem de alteração da mais recente a mais antiga mudança.

A aba **Pontos de Interrupção** (7) mostra todas as pausas incluídas no script, suas respectivas seções e opções de desativar ou excluir os pontos.

A depuração pode ser executada de forma incremental, direta ou interrompida a qualquer momento.

Nesta lição, você conheceu os recursos do Depurador de Script, para saber utilizá-lo para localizar problemas no script.

Hora de atividade! Realize as atividades referentes à Lição 13.

Laboratório 13 - Depurar

1. Usando a imagem a cima, que mostra a tela de depuração, preencha corretamente as opções abaixo com os números possíveis.
 - () Exibe todas as variáveis e respectivos valores.
 - () Ao clicar no número da linha, é criado um ponto de interrupção.
 - () Script de carga de dados.

- () Exibe o status e possíveis mensagens de erros.
- () Exibe todos os pontos de interrupção com opções de desativar ou remover o ponto.

Gabarito dos Laboratórios

Lição 1 - Plano de Projeto

2. Um plano de projeto é um documento guia com a finalidade de orientar o caminho a ser seguido durante o curso para alcançar um objetivo.
3. C, A, D, E, G, D, B.

Lição 2 - Estrutura de Dados

3. B, A, D, C

Lição 3 - Script de Carga

2. A

Lição 4 - Preparação de Dados

2. Qualificar campos significa identificar os campos concatenando o nome da tabela com o nome dos campos.

Lição 5 - Load Resident, Chave Sintética e Função Exists()

2. C

Lição 6 - Campos Chave

7. Campos chave são aqueles que são comuns em duas ou mais tabelas (campos associados).

Lição 7 - Cálculos, Include e Dimensões de Tempo

4. Utilizando o comando Autogenerate no script de carga ou o Load Inline.

Lição 8 - Tratamento de dados não padronizados

5. A

Lição 9 - Arquivos QVD

3. Diminuir a carga sobre Servidores de Base de dados; Carga Incremental; Aumentar a velocidade de Carga.

Lição 10 - Script Avançado

3. B

Lição 11 - Concatenação

2. Concatenação forçada.

Lição 12 - Uso de Mapas

2. F; V; F; V

Lição 13 - Depuração

1. 6; 3; 1; 5; 7.