Minha Experiência com o Qlikview

Elaborado por: Henrique Figueiredo de Souza	Data: 27/02/2013

Sumário

1. O Qlikview

O QlikView é a primeira plataforma de business intelligence associativa in-memory. Funciona como a nossa mente. Faz associações ao ligar dados de diversas fontes com apenas alguns cliques.
O usuário pode escolher uma questão relacionada com o seu negócio, criar uma tabela para encontrar a resposta, fazer seleções para ver associações e alterar a tabela ou criar outra instantaneamente. Pode testar e fazer protótipos e aprender simultaneamente, sem tirar os olhos dos dados e sem interromper a linha de raciocínio. Além disto, é uma plataforma única de execução e gestão simples.

As empresas podem começar com uma única versão do QlikView Desktop e expandirem para múltiplos servidores que suportam milhares de usuários ou milhões de registros. Podem executar o QlikView da forma que o usuário desejar: um cliente Windows instalado, clientes baseados em browser (incluindo um cliente Ajax sem necessidade de instalação), aparelhos móveis ou relatórios PDF enviados para uma caixa de correio eletrônica.

O PODER DO QLIKVIEW

- **Consolidar** - A tecnologia in-memory patenteada do QlikView combina dados rapidamente a partir de qualquer fonte. Utiliza o SAP, o Salesforce ou outras aplicações empresariais. Tem dados em formato Excel, XML ou CSV. Tem dados armazenados em Oracle, Microsoft SQL Server ou MySQL. Tem um armazém de dados implementado. Não há problema, o QlikView pensa em tudo.
- **Pesquisa** - A pesquisa associativa do QlikView possibilita uma experiência de pesquisa familiar que apresenta respostas empresariais rápidas. Experimente e insira os seus termos de busca. A pesquisa associativa do QlikView apresenta resultados instantâneos à medida em que se digita. A sua interface intuitiva sublinha relações importantes nos seus dados, mostrando não só as associações, como também dados que não estão relacionados.
- **Visualizar** - Esta é a parte divertida. Em apenas alguns segundos, o usuário visualiza seus dados da forma que quiser. Colocando em tabelas, quadros, gráficos – de todos os tipos imagináveis. Ampliando ou minimizando a visualização. Com o QlikView, os dados podem ser analisados sob uma perspectiva completamente diferente.

Fonte: http://www.ammc.com.br/qlikview/

2. Os Papeis do Qlikview

Conhecedor do Negócio
- Domina as regras do negócio, conhece a fundo os mínimos detalhes sobre o seu negócio.

Administrador do banco de dados (DBA)
- indivíduo que possui o acesso e os direitos absolutos as dados do banco.

Especialista Qlikview
- Transforma o negócio em documento Qlikview elaborado de acordo com o aval da perspectiva do conhecedor do negócio.

Usuário
- Indivíduo que vai fazer uso das facilidades advindas do documento Qlikview construído.

A fonte do negócio pode estar em planilhas eletrônicas e/ou num banco de dados de algum sistema.

O Trabalho do especialista Qlikview deverá ser desenvolvido juntamente com o conhecedor do negócio, obedecendo aos padrões, técnicas e procedimentos consistentes, voltados para as atividades de construção do documento Qlikview, obedecendo aos padrões de qualidade e visando às competências do negócio a ser construído, buscando-se:
- Criar um documento Qlikview
- Disponibilizar as informações pertinentes
- Apoiar o processo de construção

3. As fases de um projeto Qlikview

[Iniciação]

- Ter bem definido as pessoas que vão representar os papeis do qlikview; (Conhecedor do Negócio)
- Definir o cronograma incial do projeto; (Conhecedor do Negócio)

[Elaboração] (Modelagem do negócio, definição dos requisitos, análise inicial)

- Definir num documento as regras do negócio a ser construído, limitar o escopo do negócio neste documento; (Conhecedor do Negócio)
- É muito importante que o nome por extenso dos campos de cada tabela estejam escritos no documento de negócio; (Conhecedor do Negócio)
- Gerar o script da estrutura das tabelas de onde serão providos os dados, para termos conhecimento absoluto das chaves primárias e estrangeiras; (DBA)
- Listar o tamanho de todas as tabelas do escopo definido em termos de Kbytes e quantidade de linhas; (DBA)
- A partir do estudo dessas informações é que poderemos saber, aproximadamente o espaço necessário em disco e em memória que o servidor do qlikview precisará; (Especialista Qlikview)

[Construção] (Implementação, análise e projeto)

- Caso tudo esteja bem definido anteriormente, começamos com as três etapas de desenvolvimento dos documentos QVW;
- A primeira etapa do desenvolvimento consiste na extração dos dados do banco de dados e o armazenamento das tabelas ou visões nos arquivos QVD; (Especialista Qlikview/DBA)
- A segunda etapa consiste na transformação dos dados que estão no modelo relacional do banco de dados para o modelo associativo do qlikview e é onde definimos muitas regras do negócio; (Especialista Qlikview/Conhecedor do Negócio/DBA)
- A terceira etapa consiste na elaboração dos paineis visuais, onde serão organizados as informações em períodos, filtros, gráficos, analise comparativa, analise periódica, seleção dos últimos filtros...; (Especialista Qlikview/Conhecedor do Negócio)

[Transição] (teste e implantação)

- Nesta etapa colocamos o projeto construído anteriormente no servidor do qlikview e realizamos alguns testes, por exemplo para saber se o aspecto visual num navegador web não estrapolou os limites de tela; (Especialista Qlikview)
- Podemos definir as restrições de segurança para cada documento qlikview e o que cada usuário poderá ou não fazer dentro do documento; (Especialista Qlikview)
- Realizar vários testes para saber se o que foi construído no documento do qlikview está de acordo com as informações das regras de negócio; (Conhecedor do Negócio/Usuário)

4. Boas práticas para desenvolver um projeto em QlikView

Agradecimentos
Pelas explicações das boas práticas eu dou crédito total para **Eronita Van Leijden da SEFAZ** e pela explicação da parte avançada para **Paulo Souza da Toccato**, eu simplesmente organizei as ideias deles e algumas minhas num documento para ficar mais organizado.

1. Modularizar os documentos separando o documento com o script de carga e transformação em relação ao documento com o layout de tela, usando o comando **Bynary** para chamar o documento QVW com o código dentro do documento de tela.

```
Binary c:\diretorio\documento.qvw;
```

2. Manter as mesmas tabelas existentes nos projetos e salvando elas no formato de arquivo QVD.

```
//Após ao comando de carga da tabela Clientes, adicione o seguinte
comando Store:
Store Clientes into Diretorio\Clientes.qvd;

//Para carregar a tabela Clientes a partir do arquivo QVD
Clientes:
```

```
LOAD ClienteID,
     NomeEmpresa
FROM
[Diretorio\Clientes.qvd]
(qvd);
```

3. Usar o comando **Mapping Load** para tabelas com baixa atualização.

```
//Onde o campo [Regiao] faz a ligação da tabela mapeada com a nova
coluna que será criada.

MapaRegioes:
Mapping Load * inline [
      Regiao, DescricaoRegiao
      J,Japan
      E,Europe
      U,America
      G,German
      F,France
      K,Kelvin
];

ApplyMap('MapaRegioes',Regiao) as Regiao,
```

4. Quando for fazer uma chave composta, concatenar os campos para se tornarem um único campo, com uma única chave para simplificar as associações das tabelas e melhorar o entendimento das associações.

```
//Concatenação dos campos [Cod_Item] e [Ano] para formar a chave
composta [Chave Variação]
text([Cod Item]) & '|' & Ano as [Chave Variação],
```

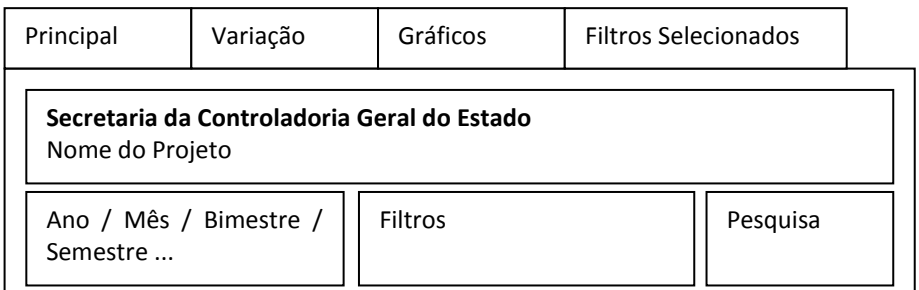
5. Como faz o agendamento da carga no Qlikview?

O agendamento é programado dentro do Qlikview Server, através do Qlikview Management Console, e dentro dele acesso a aba “User Documents”, escolhe-se um documento/projeto QlikView e na aba “Reload” será possível fazer o agendamento da carga daquele documento.

6. Realizar **o controle de versão** dos documentos Qlikview, utilizando um software de versionamento como o CVS ou o SVN.

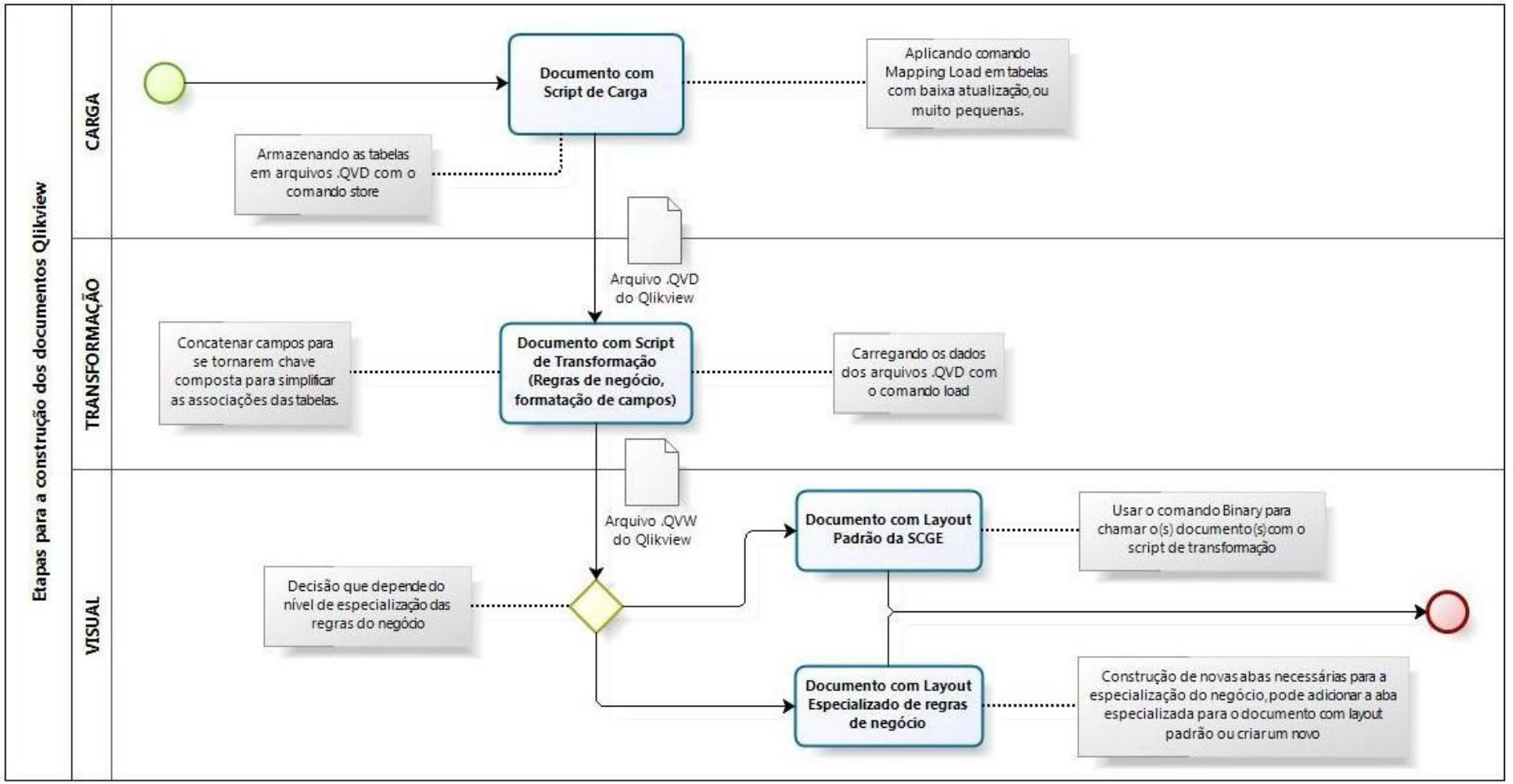
5. O Layout padrão para um documento Qlikview

- Sempre quebrar alguma data importante em (Ano / Mês/ Bimestre / Semestre ...);
- Colocar o objeto de seleção múltipla para servir como filtro principal e não usar campos de data, de valor numérico;
- Colocar o objeto Container com gráficos, exemplos de gráficos:
  - Comparativo Total
  - Pulverização
  - Pareto Total
  - Comparativo por Ano
- Colocar o objeto seleções atuais - para indicar qual informação foi filtrada/selecionada;
- Colocar o objeto de pesquisa – para facilitar a busca por qualquer informação dentro do documento;
- Definir um layout padronizado para os projetos, com as abas (Principal/Variação/Gráficos/Filtros Selecionados).



6. Etapas para a construção dos documentos Qlikview

1. **Documento com Script de Carga**
    a. Armazenando as tabelas em arquivos .QVD com o comando store;
    b. Aplicando comando Mapping Load em tabelas com baixa atualização, ou muito pequenas.
2. **Documento com Script de Transformação** (Regras de negócio, formatação de campos)
    a. Carregando os dados dos arquivos .QVD com o comando load;
    b. Concatenar campos para se tornarem chave composta para simplificar as associações das tabelas.
3. **Documento com Layout Padrão**
    a. Usar o comando binary para chamar o(s) documento(s) com o script de transformação.
4. **Documento com Layout Especializado de regras de negócio**
    a. Pode adicionar a aba especializada no documento com layout padrão ou criar um novo;
    b. Construção de novas abas necessárias para a especialização do negócio.


Etapas para a construção dos documentos Qlikview
CARGA
TRANSFORMAÇÃO
VISUAL
Documento com Script de Carga
Aplicando comando Mapping Load em tabelas com baixa atualização,ou muito pequenas.
Armazenando as tabelas em arquivos .QVD com o comando store
Arquivo .QVD do Qlikview
Concatenar campos para se tornarem chave composta para simplificar as associações das tabelas.
Documento com Script de Transformação (Regras de negócio, formatação de campos)
Carregando os dados dos arquivos .QVD com o comando load
Arquivo .QVW do Qlikview
Documento com Layout Padrão da SCGE
Usar o comando Binary para chamar o(s) documento(s) com o script de transformação
Decisão que depende do nível de especialização das regras do negócio
Documento com Layout Especializado de regras de negócio
Construção de novas abas necessárias para a especialização do negócio, pode adicionar a aba especializada para o documento com layout padrão ou criar um novo

7. Como é feita a integração do documento Qlikview com algum sistema?

Existem duas abordagens:

1) **Passando parâmetros pela URL do navegador**:

http://URL/**documento.qvw**&select=COMPONENTE1,PARAMETRO11,PARAMETRO12,PARAMETRO13...[COMPONENTE2,PARAMETRO21,PARAMETRO22,PARAMETRO23...]

Exemplo:

http://192.168.1.4/QvAJAXZfc/opendoc.htm?document=Apresentacoes/ExemploIntegracao.qvw&select=LB44,2010

O Qlickview segue o padrão normal de passagem de parâmetros para uma URL, no caso dele para um documento .QVW

O documento → 192.168.1.4/QvAJAXZfc/opendoc.htm?document=Apresentacoes/ExemploIntegracao.qvw

A seleção do componente LB44 → select=LB44

Depois do componente LB44 segue os seus parâmetros depois da vírgula → select=LB44,2010

SIM é possível passar mais de um componente e para cada componente mais de um parâmetro.

2) **Através de código javascript**:

Exemplos retirados da própria documentação do Qlikview.

Exemplo 1: Capturar o evento quando a seleção está sendo feita por um listbox.

```
var doc;
var lb;
qvInit = function () {
    doc = Qv.GetCurrentDocument();
    lb = doc.GetObject("LB36");
    lb.SetOnUpdateComplete(listboxUpdated);
}
listboxUpdated = function () {
    var selected = this.Data.GetSelected();
    //"selected" is now an array of objects
    alert(selected.length + " selected items");
    //loop through the array
    for (var i = 0; i < selected.length; i++)
    {
        //get the "text" property
        var text = selected[i].text;
    }
}
Qv.InitWorkBench({ View: "Movies Database", BodyOnLoadFunctionNames: "qvInit"
});
```

Exemplo 2: Limpar as seleções de um listbox. A função "clearSelection" deve ser chamada do seu código.

```
var doc;
var lb;
qvInit = function () {
    doc = Qv.GetCurrentDocument();
    lb = doc.GetObject("LB36");
}
clearSelection = function () {
```

```
    lb.Data.ClearSelections();
}
Qv.InitWorkBench({ View: "Movies Database", BodyOnLoadFunctionNames: "qvInit"
});
```

8. Dicas de Projeto Qlikview

Copiar e Colar para evitar retrabalho (esta é a dica mais importante)
Primeiro faça um documento qlikview padrão que contenha todos os objetos que possam sempre ser copiados para outros projetos. Para copiar um objeto para outra aba, simplesmente segure o botão esquerdo do mouse + a tecla Ctrl e para copiar a referência ligada de um objeto para outra aba, segure o botão esquerdo do mouse + a tecla Ctrl + a tecla Shift.

Lembrar sempre que for possível **realizar todas as transformações dos dados pelo código** e não amarrado às propriedades de um documento específico, apesar da facilidade que o qlikview Personal Edition oferece.

Importante **criar uma explicação descritiva para cada campo** (preferencialmente um dicionário de dados) e também criar um rótulo mais descritivo para cada campo, ex: codabx as [código de absorção X]

Lembrar que os campos são associados pelos nomes que recebem, por isso sempre é necessário renomear campos com o mesmo nome em tabelas diferentes, qualificando-os.

Usar o **caminho relativo**, usando o comando **Directory**, isso pode parecer óbvio, porém tive algum retrabalho quando mudava o projeto de pasta.

Caso necessite usar no mesmo documento mais de uma tabela unidas numa única tabela e ao mesmo tempo as mesmas tabelas separadas, é necessário que nas tabelas separadas exista um qualificador individualizando os campos que são comuns com a tabela para evitar a associação dos campos, o comando **qualify** pode ajudar.

Qual a diferença entre grupo cíclico e grupo hierárquico?

Grupos hierárquicos (drill-down) - Os Grupos hierárquicos são usados para criar hierarquias de campos que permitem as hierarquias em gráficos. Quando vários campos formam uma hierarquia natural, faz sentido criar um grupo hierárquico. Ex: Tempo: Ano, Trimestre, Mês ou Geografia: Continente, País, Estado, Cidade

Grupos cíclicos não são hierárquicos e normalmente são usados apenas como uma forma conveniente de permitir que o usuário alterne os campos de dimensão do gráfico com um simples clique do mouse.

Prefira sempre formatar os campos na linha de código mesmo e não dentro das propriedades gerais do documento, pois eles ficam mais visíveis no código da aplicação do que ocultos dentro do documento Qlikview.

O Qlikview vai primeiro interpretar datas e horas de acordo com o formato padrão para data e hora e caso esta data/hora não estiver no formato padrão, ele vai interpretar a data com o formato “yyyy-MM-dd” e a hora com o formato “hh:mm:ss.ms”.

Como converter campos para ponto flutuante importados com vírgula

```
Num(Replace(Replace([Valor],',',''),'.',','),'###.###.###.##0,00')    as    [Valor
Formatado]
```

Sempre usar o objeto de Pesquisa em todas as telas de análises

Análise Comparativa
Na análise comparativa geralmente comparamos graficamente as expressões principais através de dimensões relacionadas, incluindo entre os gráficos a temporalidade dessas dimensões. A construção dos gráficos, em geral, é bem simples e diretamente associada com cada dimensão e expressões principais.

Análise de Variação Temporal

Na análise de variação temporal, comparamos a agregação de expressões principais através de alguma dimensão de tempo, o que geralmente é feito mensalmente, pois reflete melhor a realidade na maioria dos casos. A construção dos gráficos finais se torna mais complicada, pois inicialmente durante a carga e transformação dos dados precisamos criar uma tabela separada que reflete a variação da dimensão tempo e também precisamos de uma chave única que ligue esta nova tabela com a tabela que contêm as dimensões que serão filtradas posteriormente pelo usuário.

Para demonstrar algumas dessas dicas criei um projeto exemplo em Qlikview com dados fictícios, não sei se vocês conseguiram abrir o mesmo, abaixo listo o conteúdo de cada aba com o código do projeto:

```
/*
Usar mapping load inline para campos código sem descrição do arquivo carregado
O inline que dizer que os dados ficam no documento qlikview
*/

MapaMeses:
Mapping Load * inline [
	CodigoMes, DescricaoMes
	1, Janeiro
	2, Fevereiro
	3, Março
	4, Abril
	5, Maio
	6, Junho
	7, Julho
	8, Agosto
	9, Setembro
	10, Outubro
	11, Novembro
	12, Dezembro
];

/*
Usar mapping load inline para campos código sem descrição do arquivo carregado
O inline que dizer que os dados ficam no documento qlikview
*/

MapaCategorias:
Mapping Load * inline [
	CodigoCategoria, DescricaoCategoria
	1, 'Agência de Viagens'
	2, 'Alimentos e Bebidas'
	3, 'Áudio'
	4, 'Automotivo'
	5, 'Bebês'
	6, 'Beleza e Saúde'
	7, 'Blockbuster'
	8, 'Brinquedos'
	9, 'Câmeras e Filmadoras'
	10, 'Cama, Mesa e Banho'
	11, 'CDs + DVDs Musicais'
	12, 'Celulares e Telefones'
	13, 'DVDs e Blu-Ray'
	14, 'Eletrodomésticos'
	15, 'Eletrônicos'
	16, 'Eletroportáteis'
	17, 'Esporte e Lazer'
	18, 'Ferramentas e Jardim'
	19, 'Games'
	20, 'Informática'
	21, 'Informática Acessórios'
```

```
      22, 'Livros'
      23, 'Móveis e Decoração'
      24, 'Papelaria'
      25, 'Perfumaria'
      26, 'Pet Shop'
      27, 'Utilidades Domésticas'
];

/*
Usar mapping load para campos código sem descrição do arquivo carregado
Perceba que aqui eu não uso o comando inline, estou carregando os dados da
própria planilha

Usando o comando Directory para não amarrar o caminho do arquivo em um diretório
*/

Directory;
MapaProdutos:
Mapping LOAD
      CodigoProduto,
      DescricaoProduto
FROM
DadosProjetoQlikview.xlsx
(ooxml, embedded labels, table is Produtos);


 /*
Tabela fato principal do projeto, ela é quem detenhe o campos que podem ser
filtrados.
Aqui foi criado um campo chave para ligar com a tabela de variação,
sempre fica mais fácil juntar dois campos para formar uma chave única.
*/

Directory;

[Vendas de Produtos]:
LOAD Ano,
     CodigoCategoria,
     ApplyMap('MapaCategorias',CodigoCategoria) as Categoria,
     CodigoProduto,
     ApplyMap('MapaProdutos',CodigoProduto) as Produto,
     Mes as [Codigo do Mês],
     ApplyMap('MapaMeses',Mes) as Mês,
     [Valor de Compra],
     [Valor de Venda],
     [CodigoProduto] & '|' & Ano as [Chave Variação]
FROM
DadosProjetoQlikview.xlsx
(ooxml, embedded labels, table is Dados);


/*
A tabela temporária somente serve para separar os dados de um mês específico,
neste caso o mês de janeiro
A TabelaVariacao vai armazenar o resultado de cada tabela temporária mês a mês,
indicando o valor de venda como o nome do mês de janeiro,
para que a tabela de variação possa agrupar as linhas mês a mês.

O uso do comando NoConcatenate indica que a primeira carga da TabelaVariacao não
será concatenada com nenhum carga anterior

E finalmente usando o comando drop table apagamos a tabela temporaria para que
ele sempre armazene o mês corrente.
*/
```

```
Directory;

[Temp]:
LOAD Ano,
     CodigoCategoria,
     ApplyMap('MapaCategorias',CodigoCategoria) as Categoria,
     CodigoProduto,
     ApplyMap('MapaProdutos',CodigoProduto) as Produto,
     Mes as [Codigo do Mês],
     ApplyMap('MapaMeses',Mes) as Mês,
     [Valor de Compra],
     [Valor de Venda] as Janeiro,
     [CodigoProduto] & '|' & Ano as [Chave Variação]
FROM
DadosProjetoQlikview.xlsx
(ooxml, embedded labels, table is Dados)
where Mes = 1;

[TabelaVariacao]:
NoConcatenate
LOAD *
RESIDENT Temp;

drop table Temp;


/*
Este código irá se repetir para todos os meses, com a diferença que agora
estamos concatenando os meses seguintes dentro da TabelaVariacao
e apagando a tabela temporária.
Estamos fazendo todo esse esforço para criarmos temporariamente na memória uma
tabela que servirá de base para a análise de variação
*/

Directory;

[Temp]:
LOAD Ano,
     CodigoCategoria,
     ApplyMap('MapaCategorias',CodigoCategoria) as Categoria,
     CodigoProduto,
     ApplyMap('MapaProdutos',CodigoProduto) as Produto,
     Mes as [Codigo do Mês],
     ApplyMap('MapaMeses',Mes) as Mês,
     [Valor de Compra],
     [Valor de Venda] as Fevereiro,
     [CodigoProduto] & '|' & Ano as [Chave Variação]
FROM
DadosProjetoQlikview.xlsx
(ooxml, embedded labels, table is Dados)
where Mes = 2;

Concatenate ([TabelaVariacao])
LOAD *
RESIDENT Temp;

drop table Temp;


/*
O resultado final é o agrupamento da variação dos meses em uma tabela nova
ligada a tabela de fatos [Vendas de Produtos]
O uso do comando Resident serve para carregarmos os dados vindos de uma tabela
```

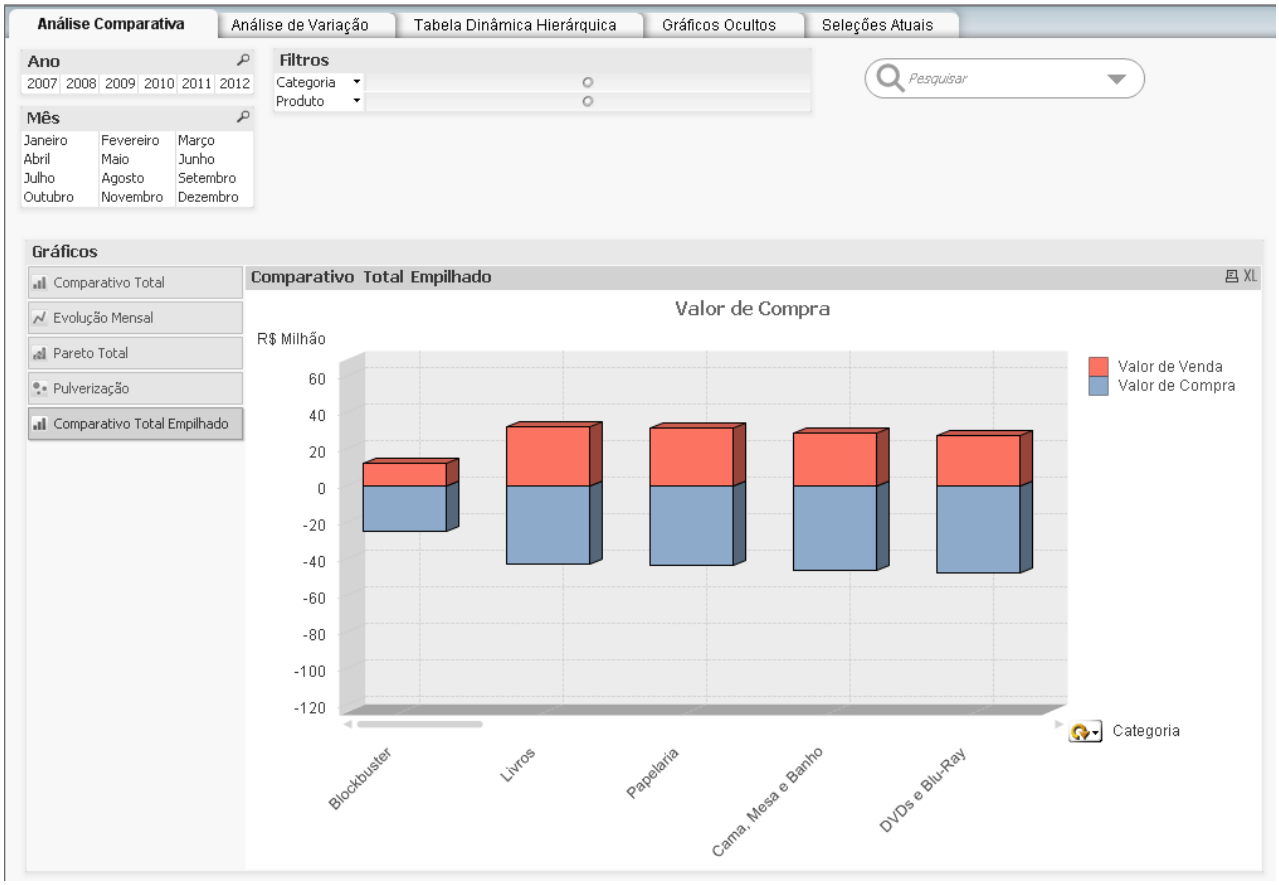
```
que só existe na memória.
Usamos o comando Group by para agruparmos a variação pelo codigo do produto e
pelo Ano, usando a nova Chave de variação única
E finalmente apagamos a TabelaVariacao que estava somente servindo como buffer
temporário dos dados.
*/

[Variações por Produto]:
LOAD
     [CodigoProduto] & '|' & Ano                    as [Chave Variação],
     (sum (Fevereiro) / sum (Janeiro)) - 1          as [Variação
Janeiro/Fevereiro],
     (sum (Março) / sum (Fevereiro)) - 1       as [Variação Fevereiro/Março],
     (sum (Abril) / sum (Março)) - 1                as [Variação Março/Abril],
     (sum (Maio) / sum (Abril)) - 1            as [Variação Abril/Maio],
     (sum (Junho) / sum (Maio)) - 1            as [Variação Maio/Junho],
     (sum (Julho) / sum (Junho)) - 1                as [Variação Junho/Julho],
     (sum (Agosto) / sum (Julho)) - 1               as [Variação Julho/Agosto],
     (sum (Setembro) / sum (Agosto)) - 1       as [Variação Agosto/Setembro],
     (sum (Outubro) / sum (Setembro)) - 1      as [Variação Setembro/Outubro],
     (sum (Novembro) / sum (Outubro)) - 1      as [Variação Outubro/Novembro],
     (sum (Dezembro) / sum (Novembro)) - 1          as [Variação
Novembro/Dezembro],
     (sum (Janeiro) / sum (Dezembro)) - 1      as [Variação Dezembro/Janeiro]
RESIDENT
      TabelaVariacao
GROUP BY
      [CodigoProduto] & '|' & Ano;

DROP TABLE TabelaVariacao;

/*
Id: identificação do aluno
Turma:      turma a que o aluno foi colocado (A ou B)
Sexo: F se feminino, M se masculino
Idade:      idade, em anos
Altura: altura em metros
Peso: peso em quilogramas
Filhos: número de filhos na família
Fuma: hábito de fumar, sim ou não
Toler:      tolerância ao cigarro: (I) Indiferente, (P) Incomoda pouco e (M)
Incomoda muito
Exerc:      horas de atividade física, por semana
Cine: número de vezes que vai ao cinema, por semana
OpCine: opinião a respeito das salas de cinema na cidade: (B) regular a boa e
(M) muito boa
TV:   horas gastas assistindo TV, por semana
OpTV: opinião a respeito da qualidade da programação na TV: (R) ruim, (M) média,
(B) boa e (N) não sabe

Neste exemplo demonstramos
O comando Qualify que qualifica os campos da tabela como tabela.campo
O comando If em linha
A formatação dos campo Altura e Peso, onde a separação original está com ponto e
não vírgula, por isso a necessidade da formatação
*/

Directory;

QUALIFY *;

Alunos:
LOAD Id as Identificação,
     Turma,
```

```
     If(Sexo='F','Feminino','Masculino') as Sexo,
     Idade,
     Num(Replace(Replace(Alt,',',''),'.',','),'###.###.###.##0,00') as Altura,
     Num(Replace(Replace(Peso,',',''),'.',','),'###.###.###.##0,00') as Peso,
     Filhos as [Número de Filhos na Família],
     If(Fuma='SIM','Sim','Não') as [Hábito de Fumar],
     If(Upper(Toler)='I','Indiferente',If(Upper(Toler)='P','Incomoda
pouco',If(Upper(Toler)='M','Incomoda muito',''))) as [Tolerância ao Cigarro],
     Exerc as [Horas de Atividade Física],
     Cine as [Número vezes que vai ao Cinema],
     If(OpCine='B','regular a boa',If(OpCine='M','muito boa','')) as [Opinião
Sala Cinema],
     TV as [Horas assistindo TV],
     If(OpTV='R','ruim',If(OpTV='M','média',If(OpTV='B','boa','não sabe'))) as
[Qualidade Programação TV]
FROM
DadosProjetoQlikview.xlsx
(ooxml, embedded labels, table is Alunos);

UNQUALIFY *;


/*
Para transformar uma tabela cruzada em uma tabela simples use o prefixo
CrossTable com o comando LOAD
*/

Directory;
TabelaCruzada:
CrossTable ([TabelaCruzada - Mês], [TabelaCruzada - Vendas], 2)
LOAD Vendedor as [TabelaCruzada - Vendedor],
     Ano as [TabelaCruzada - Ano],
     Janeiro,
     Fevereiro,
     Março,
     Abril,
     Maio,
     Junho
FROM
DadosProjetoQlikview.xlsx
(ooxml, embedded labels, table is TabelaCruzada);



/*
Usamos o comando Sample para carregar uma amostra aleatório de linhas
A função pick() retorna a n-ésima expressão na lista.
A função Rand() gera valores aleatórios
*/


Directory;
TabelaVirtual1:
Sample 0.15 LOAD
      mid(CodigoProduto,IterNo(),1) as CodigoV1,
      DescricaoProduto as DescricaoV1,
      pick(IterNo(), 'Vendido', 'Comprado', 'Alugado', 'Quebrado') as EstadoV1,
      Rand() * 100 as AleatorioV1,
      MakeDate( round(rand()*2000), pick(IterNo(),1,2,3,4,5,6,7,8,9,10,11,12),
round(rand()*10) ) as DataV1
      //MakeTime( hh [, mm [, ss [.fff ]]] )
FROM
DadosProjetoQlikview.xlsx
(ooxml, embedded labels, table is Produtos)
```

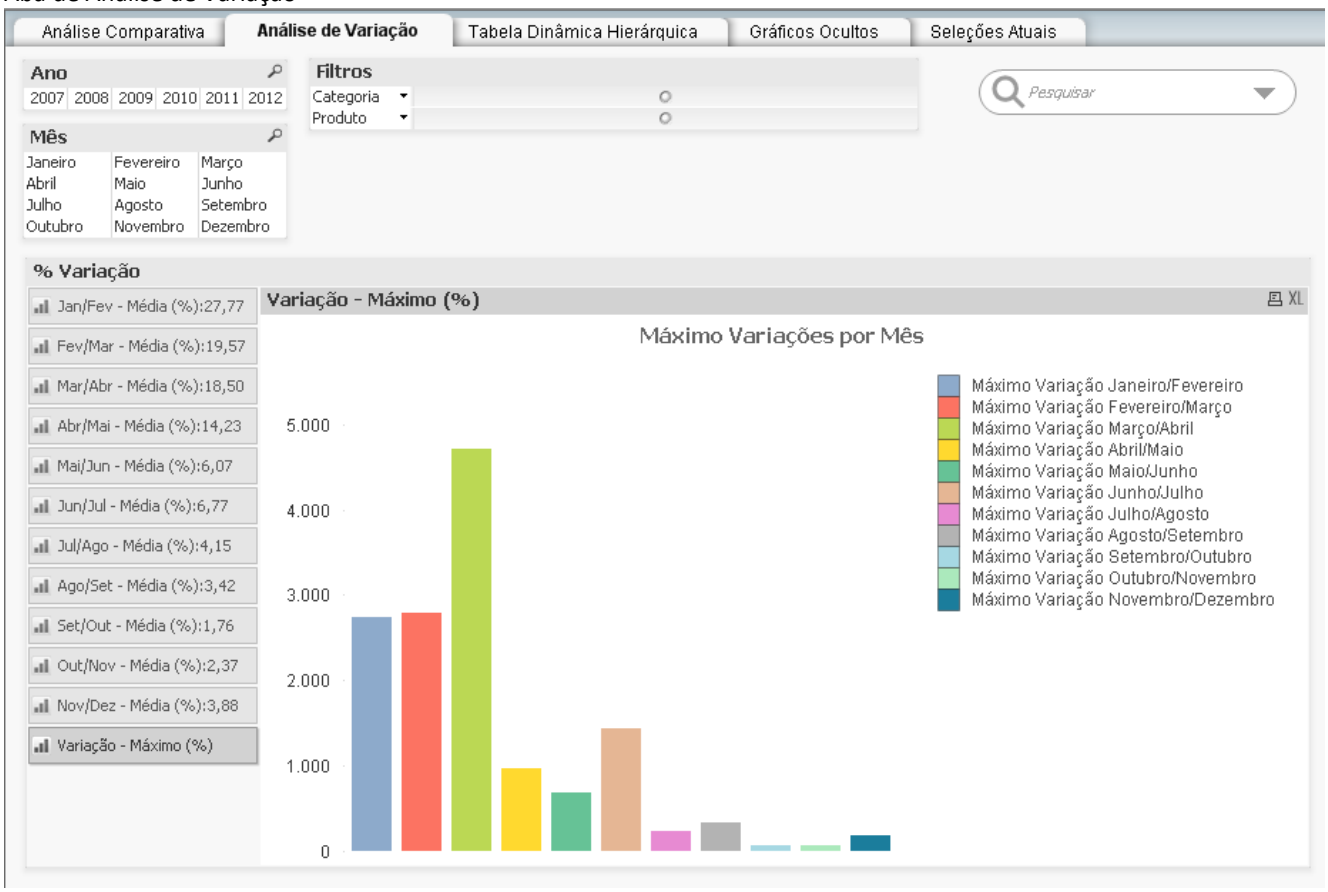
```
while mid(CodigoProduto,IterNo(),1)<>'';


TabelaVirtual2:
Load
      RecNo() as CodigoV2,
      Rand() * 10000 as NumeroV2
autogenerate(2000);
```

Para quem não conseguiu abrir o projeto, aqui estão algumas telas de como ele ficou:

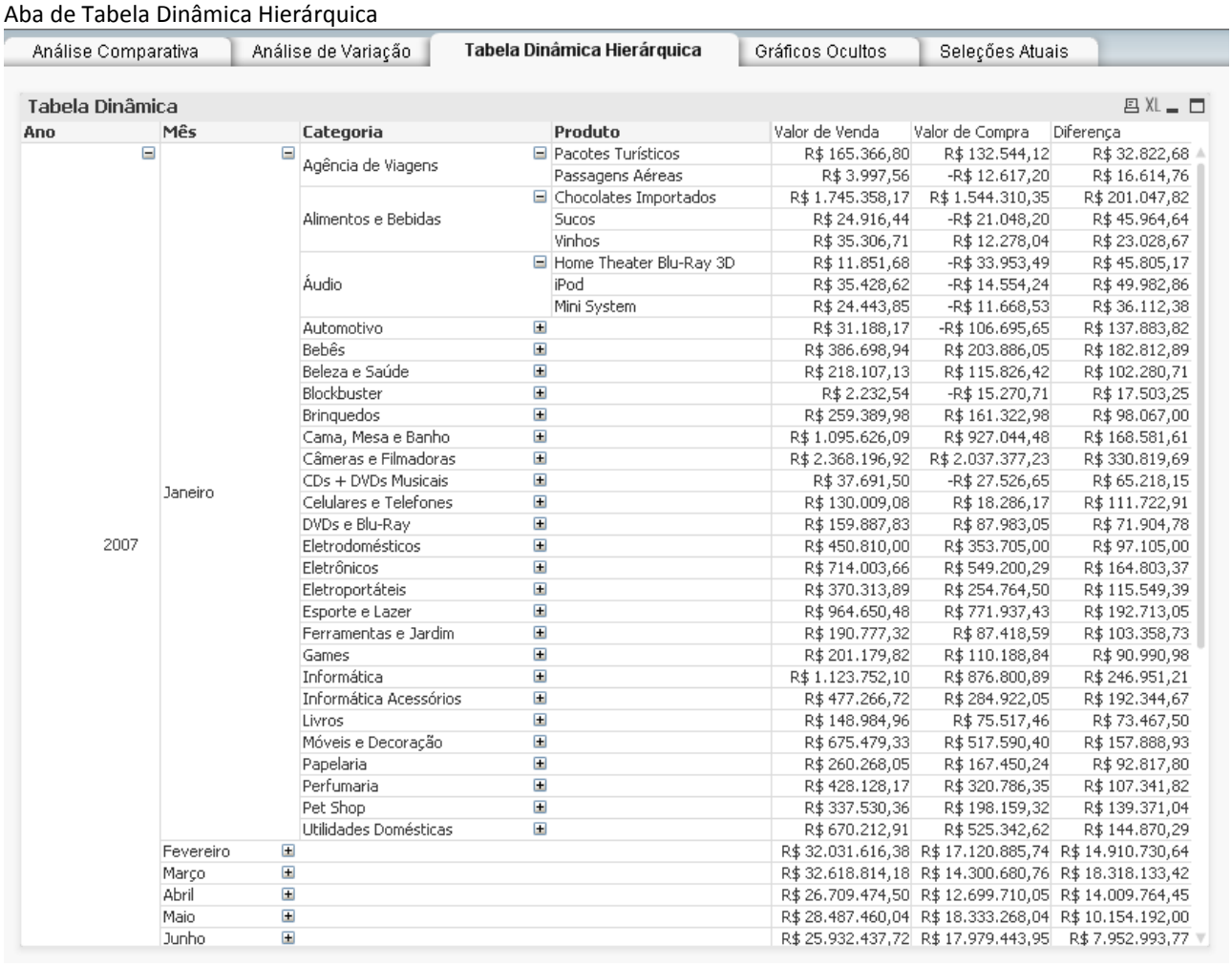
Aba de Análise Comparativa



Aba de Análise de Variação



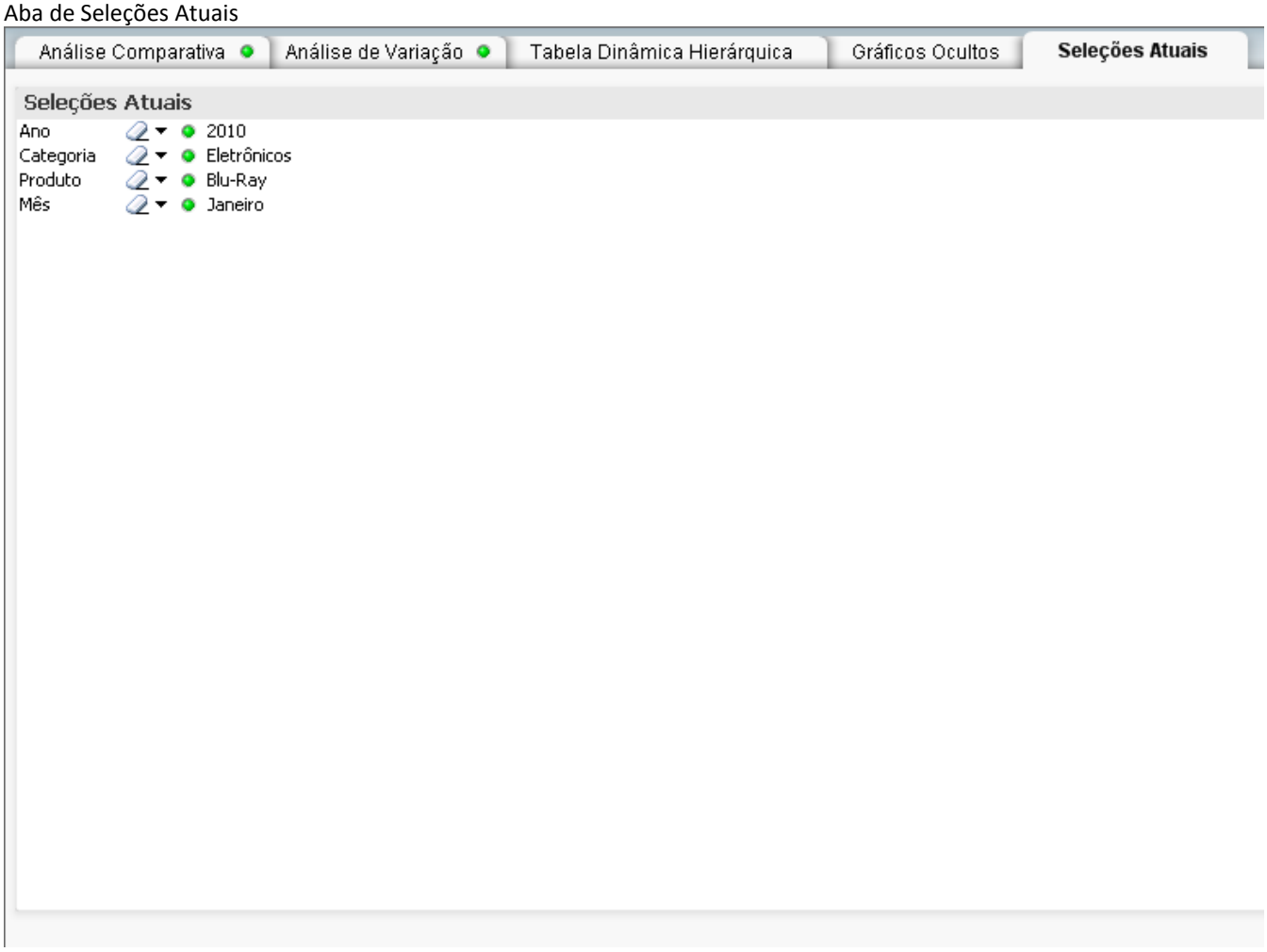
Aba de Tabela Dinâmica Hierárquica



Ano	Mês	Categoria	Produto	Valor de Venda	Valor de Compra	Diferença
2007	Janeiro	Agência de Viagens	Pacotes Turísticos	R$ 165.366,80	R$ 132.544,12	R$ 32.822,68
			Passagens Aéreas	R$ 3.997,56	-R$ 12.617,20	R$ 16.614,76
		Alimentos e Bebidas	Chocolates Importados	R$ 1.745.358,17	R$ 1.544.310,35	R$ 201.047,82
			Sucos	R$ 24.916,44	-R$ 21.048,20	R$ 45.964,64
			Vinhos	R$ 35.306,71	R$ 12.278,04	R$ 23.028,67
		Áudio	Home Theater Blu-Ray 3D	R$ 11.851,68	-R$ 33.953,49	R$ 45.805,17
			iPod	R$ 35.428,62	-R$ 14.554,24	R$ 49.982,86
			Mini System	R$ 24.443,85	-R$ 11.668,53	R$ 36.112,38
		Automotivo		R$ 31.188,17	-R$ 106.695,65	R$ 137.883,82
		Bebês		R$ 386.698,94	R$ 203.886,05	R$ 182.812,89
		Beleza e Saúde		R$ 218.107,13	R$ 115.826,42	R$ 102.280,71
		Blockbuster		R$ 2.232,54	-R$ 15.270,71	R$ 17.503,25
		Brinquedos		R$ 259.389,98	R$ 161.322,98	R$ 98.067,00
		Cama, Mesa e Banho		R$ 1.095.626,09	R$ 927.044,48	R$ 168.581,61
		Câmeras e Filmadoras		R$ 2.368.196,92	R$ 2.037.377,23	R$ 330.819,69
		CDs + DVDs Musicais		R$ 37.691,50	-R$ 27.526,65	R$ 65.218,15
		Celulares e Telefones		R$ 130.009,08	R$ 18.286,17	R$ 111.722,91
		DVDs e Blu-Ray		R$ 159.887,83	R$ 87.983,05	R$ 71.904,78
		Eletrodomésticos		R$ 450.810,00	R$ 353.705,00	R$ 97.105,00
		Eletrônicos		R$ 714.003,66	R$ 549.200,29	R$ 164.803,37
		Eletroportáteis		R$ 370.313,89	R$ 254.764,50	R$ 115.549,39
		Esporte e Lazer		R$ 964.650,48	R$ 771.937,43	R$ 192.713,05
		Ferramentas e Jardim		R$ 190.777,32	R$ 87.418,59	R$ 103.358,73
		Games		R$ 201.179,82	R$ 110.188,84	R$ 90.990,98
		Informática		R$ 1.123.752,10	R$ 876.800,89	R$ 246.951,21
		Informática Acessórios		R$ 477.266,72	R$ 284.922,05	R$ 192.344,67
		Livros		R$ 148.984,96	R$ 75.517,46	R$ 73.467,50
		Móveis e Decoração		R$ 675.479,33	R$ 517.590,40	R$ 157.888,93
		Papelaria		R$ 260.268,05	R$ 167.450,24	R$ 92.817,80
		Perfumaria		R$ 428.128,17	R$ 320.786,35	R$ 107.341,82
		Pet Shop		R$ 337.530,36	R$ 198.159,32	R$ 139.371,04
		Utilidades Domésticas		R$ 670.212,91	R$ 525.342,62	R$ 144.870,29
	Fevereiro			R$ 32.031.616,38	R$ 17.120.885,74	R$ 14.910.730,64
	Março			R$ 32.618.814,18	R$ 14.300.680,76	R$ 18.318.133,42
	Abril			R$ 26.709.474,50	R$ 12.699.710,05	R$ 14.009.764,45
	Maio			R$ 28.487.460,04	R$ 18.333.268,04	R$ 10.154.192,00
	Junho			R$ 25.932.437,72	R$ 17.979.443,95	R$ 7.952.993,77

Aba de Gráficos Ocultos

Aba de Seleções Atuais



9. Os Scripts Qlikview para Salvar QVDs

9.1.criarTabelaPropriedades

```
/*============================================================================
Procedimento:     criarTabelaPropriedades
Autor:            Henrique Figueiredo de Souza
Data Criação:     28/01/2013
Descrição:
            Cria uma tabela virtual para demonstrar as propriedades
            do documento QlikView.
============================================================================*/

sub criarTabelaPropriedades

      let conexao = ConnectString();
      let data = now();
      let autor = Author();
      let plataformaCliente = ClientPlatform();
      let nomeComputador = ComputerName();
      let nomeDocumento = DocumentName();
      let caminhoDocumento = DocumentPath();
      let tituloDocumento = DocumentTitle();
      let numeroRelatorios = NoOfReports();
      let numeroTabelas = NoOfTables();
      let usuarioSO = OSUser();
      let versaoQlikView = QlikViewVersion();
      let usuarioQlikView = QVUser();
      let tempoRecarga = ReloadTime();

      TabelaPropriedades:
      Load * inline [
            Propriedade, Valor
            String de Conexão, $(conexao)
            Data, $(data)
            Autor, $(autor)
            Plataforma Cliente, $(plataformaCliente)
            Nome do Computador, $(nomeComputador)
            Nome do Documento, $(nomeDocumento)
            Caminho do Documento, $(caminhoDocumento)
            Título do Documento, $(tituloDocumento)
            Número de Relatórios, $(numeroRelatorios)
            Número de Tabelas, $(numeroTabelas)
            Usuário do Sistema Operacional, $(usuarioSO)
            Versão do QlikView, $(versaoQlikView)
            Usuário do QlikView, $(usuarioQlikView)
            Última Recarga do Script, $(tempoRecarga)
      ];

end sub;
```

9.2.listarNomesTabelasBanco

```
/*============================================================================
Procedimento:     listarNomesTabelasBanco
Autor:            Henrique Figueiredo de Souza
Data Criação:     28/01/2013
Descrição:
            Realiza as consultas no banco para retornar os nomes e os
```

```
            tamanhos das tabelas do banco, estando essas tabelas vazias
            ou não.
==============================================================================*/

sub listarNomesTabelasBanco(bListarDoBanco)


      if (bListarDoBanco=true()) then

            TabelaNomesBancoOriginal:
            SQL SELECT segment_name NOME, round(sum(bytes/1024),2) TAMANHO
            FROM user_segments
            WHERE segment_type = 'TABLE'
            GROUP BY segment_name
            ORDER BY 2 desc;

            let vNumeroLinhas = NoOfRows('TabelaNomesBancoOriginal');

            if  vNumeroLinhas = 0 then

                  drop Table TabelaNomesBancoOriginal;

                  TabelaNomesBancoOriginal:
                  SQL SELECT TABLE_NAME NOME, 0 TAMANHO
                  FROM user_tables
                  ORDER BY 1;
            end if

      else

            TabelaNomesBancoOriginal:
            Load * inline [
                  NOME, TAMANHO
                  TABELA1, 40
            ];

      end if

      TabelaNomesBancoNova:
      LOAD
            NOME,
            TAMANHO,
            Num((TAMANHO/1024),'###.###.###.##0,00') as [TAMANHO_MB],
            Num((TAMANHO/1024/1024),'###.###.###.##0,00') as [TAMANHO_GB]
      Resident TabelaNomesBancoOriginal;

      drop Table TabelaNomesBancoOriginal;

end sub;
```

9.3.duracaoEntreHoras

```
/*==============================================================================
Procedimento:     duracaoEntreHoras
Autor:            Henrique Figueiredo de Souza
Data Criação:     30/01/2013
Parametros:
                  pHoraAnterior     Hora inicial do intervalo.
                  pHoraPosterior    Hora final do intervalo.
                  rDuracao          retorna a duracao em horas.
Descrição:
                  Calcula a diferença entre duas horas e retorna a duração.
```

```
==========================================================================*/

sub duracaoEntreHoras(pHoraAnterior, pHoraPosterior, rDuracao)

      let vHora = Hour('$(pHoraAnterior)');
      let vMinuto = Minute('$(pHoraAnterior)');
      let vSegundo = Second('$(pHoraAnterior)');

      let vDuracaoAnterior = (vHora*3600) + (vMinuto*60) + vSegundo;

      let vHora = Hour('$(pHoraPosterior)');
      let vMinuto = Minute('$(pHoraPosterior)');
      let vSegundo = Second('$(pHoraPosterior)');

      let vDuracaoPosterior = (vHora*3600) + (vMinuto*60) + vSegundo;

      let vDiferenca = vDuracaoPosterior - vDuracaoAnterior;

      if vDiferenca > 0 then

            let vHora = div(vDiferenca, 3600);
            let vResto = (vDiferenca-(vHora*3600));
            let vMinuto = div(vResto,60);
            let vSegundo = (vResto-(vMinuto*60));

            let rDuracao = MakeTime(vHora, vMinuto, vSegundo);

      else

            let rDuracao = MakeTime(0, 0, 0);

      end if

end sub;
```

9.4.duracaoEntreDatas

```
/*==========================================================================
Procedimento:     duracaoEntreDatas
Autor:            Henrique Figueiredo de Souza
Data Criação:     30/01/2013
Parametros:
            pDataAnterior     Data/Hora inicial do intervalo.
            pDataPosterior    Data/Hora final do intervalo.
            rDuracao          retorna a duracao em Dias/Horas.
Descrição:
            Calcula a diferença entre duas datas e retorna a duração.

*ATENÇÃO*
            AS DATAS TEM QUE ESTAR NO INTERVALO DO MESMO MÊS.
==========================================================================*/

sub duracaoEntreDatas(pDataAnterior, pDataPosterior, rDuracao)

      let vDia = Day('$(pDataAnterior)');
      let vHora = Hour('$(pDataAnterior)');
      let vMinuto = Minute('$(pDataAnterior)');
      let vSegundo = Second('$(pDataAnterior)');

      let vDuracaoAnterior = (vDia*86400) + (vHora*3600) + (vMinuto*60) +
```

```
vSegundo;

      let vDia = Day('$(pDataPosterior)');
      let vHora = Hour('$(pDataPosterior)');
      let vMinuto = Minute('$(pDataPosterior)');
      let vSegundo = Second('$(pDataPosterior)');

      let vDuracaoPosterior = (vDia*86400) + (vHora*3600) + (vMinuto*60) +
vSegundo;

      let vDiferenca = vDuracaoPosterior - vDuracaoAnterior;

      if vDiferenca > 0 then

            let vDia = div(vDiferenca, 86400);
            let vResto = (vDiferenca-(vDia*86400));
            let vHora = div(vResto, 3600);
            let vResto = (vResto-(vHora*3600));
            let vMinuto = div(vResto,60);
            let vSegundo = (vResto-(vMinuto*60));

            let rDuracao = vDia & ' dia(s) ' & MakeTime(vHora, vMinuto,
vSegundo);

      else

            let rDuracao = '0 dia(s) ' & MakeTime(0, 0, 0);

      end if

end sub;
```

9.5.salvarArquivosQVD

```
/*============================================================================
Procedimento:     salvarArquivosQVD
Autor:            Henrique Figueiredo de Souza
Data Criação:     28/01/2013
Parametros:
            pDiretorio              Nome do diretorio onde serão salvos os
                                    arquivos QVD.
            pNomeEsquema                  Nome do esquema do banco de dados.
            pIncluirCampoDataCarga  Se vai incluir o campo de data da
                                    carga nas tabelas.
            pVerficarTabelaVazia    Não salva o arquivo QVD se a tabela
                                    estiver vazia.
            pAtualizarQVD                 Se o arquivo QVD existir no diretório
                                    sobrepor o arquivo existente.
Descrição:
            Consulta a tabela em memória TabelaNomesBancoNova para salvar
            os arquivos QVD dentro do diretório passado como parâmetro.
============================================================================*/

sub salvarArquivosQVD(pDiretorio,pNomeEsquema,pIncluirCampoDataCarga,
pVerficarTabelaVazia,pAtualizarQVD)

      Let vQtdTabelas = NoOfRows('TabelaNomesBancoNova');
      let vQlikviewCargaDatetime = Now();

      LET vNomeTabela = Peek('NOME', 0, 'TabelaNomesBancoNova');

      TabelaNomesDuracao:
```

```
	LOAD NOME,
		'0' as DURACAO
	Resident TabelaNomesBancoNova
	Where NOME = '$(vNomeTabela)';


	For i = 0 To $(vQtdTabelas) - 1
		LET vNomeTabela = Peek('NOME', $(i), 'TabelaNomesBancoNova');
		let vDataAnterior = now();

		if (pIncluirCampoDataCarga=true()) then

			$(vNomeTabela):
			SQL SELECT * FROM $(pNomeEsquema).$(vNomeTabela) WHERE rownum
< 0;

			LET vQtdColunas = NoOfFields('$(vNomeTabela)');
			LET vNomeColunas = FieldName(1,'$(vNomeTabela)');

			For j = 2 To $(vQtdColunas)
				LET vNomeColunas = '$(vNomeColunas)' & ', ' &
FieldName($(j),'$(vNomeTabela)');
			Next j

			LET vNomeColunas = chr(39) & '$(vQlikviewCargaDatetime)' &
chr(39) & ' QLIKVIEW_CARGA_DATETIME, $(vNomeColunas)';

			DROP Table $(vNomeTabela);

			$(vNomeTabela):
			SQL SELECT $(vNomeColunas) FROM
$(pNomeEsquema).$(vNomeTabela);

		else

			$(vNomeTabela):
			SQL SELECT * FROM $(pNomeEsquema).$(vNomeTabela);

		end if


		let vSalvarQVD = false();

		if (pVerficarTabelaVazia=true()) then
			if NoOfRows('$(vNomeTabela)') > 0 then
				let vSalvarQVD = true();
			else
				let vSalvarQVD = false();
			end if
		else
			let vSalvarQVD = true();
		end if

		if not isnull(QVDCreateTime('$(pDiretorio)$(vNomeTabela).qvd')) then
			if (pAtualizarQVD=true()) then
				let vSalvarQVD = true();
			else
				let vSalvarQVD = false();
			end if
		end if

		if (vSalvarQVD=true()) then

			STORE $(vNomeTabela) INTO $(pDiretorio)$(vNomeTabela).qvd;
```

```
                  let vDataPosterior = now();

                  call duracaoEntreDatas(vDataAnterior, vDataPosterior,
vDuracao);

                  Concatenate(TabelaNomesDuracao)
                  LOAD '$(vNomeTabela)' as NOME,
                        '$(vDuracao)' as DURACAO
                  Resident TabelaNomesBancoNova
                  Where NOME = '$(vNomeTabela)';

            end if

            DROP Table $(vNomeTabela);
      Next i

      Inner Join(TabelaNomesBancoNova)
      LOAD
            NOME,
            DURACAO
      Resident TabelaNomesDuracao;

      drop table TabelaNomesDuracao;

      TabelaNomesBanco:
      LOAD
            RecNo()-1 as [Ordem do Registro],
            NOME,
            TAMANHO,
            TAMANHO_MB as [Tamanho da Tabela em Mb],
            TAMANHO_GB as [Tamanho da Tabela em Gb],
            DURACAO
      Resident TabelaNomesBancoNova
      Where DURACAO <> '0';

      drop table TabelaNomesBancoNova;

end sub;
```

9.6.criarTabelaArquivosQVD

```
/*============================================================================
Procedimento:     criarTabelaArquivosQVD
Autor:            Henrique Figueiredo de Souza
Data Criação:     28/01/2013
Parametros:
            pDiretorio        Nome do diretorio onde serão salvos os
                              arquivos QVD.
Descrição:
            Consulta a tabela em memória TabelaNomesBanco para criar
            uma nova tabela em memória que demonstra o percentual de
            redução dos arquivos QVD em relação aos arquivos do banco
            de dados, além de outras informações úteis.
============================================================================*/

sub criarTabelaArquivosQVD(pDiretorio)

      TabelaArquivosQVD:
      load NOME as [Nome da Tabela],
            DURACAO as [Duração Salvar QVD],
            Num(TAMANHO,'###.###.###.##0,00') as [Tamanho da Tabela em Kb],
```

```
            Num((FileSize('$(pDiretorio)' &
                  peek('NOME', RecNo()-1, 'TabelaNomesBanco') & '.qvd')/1024),
                  '###.###.###.##0,00') as [Tamanho Arquivo QVD em Kb],
            Timestamp(QvdCreateTime('$(pDiretorio)' &
                  peek('NOME', RecNo()-1, 'TabelaNomesBanco') & '.qvd'),
                  'DD/MM/YYYY hh:mm:ss') as [Hora Criação do QVD],
            QvdNoOfFields('$(pDiretorio)' &
                  peek('NOME', RecNo()-1, 'TabelaNomesBanco') & '.qvd')
                  as [Número de Campos do QVD],
            QvdNoOfRecords('$(pDiretorio)' &
                  peek('NOME', RecNo()-1, 'TabelaNomesBanco') & '.qvd')
                  as [Número de Registros do QVD],
            QvdTableName('$(pDiretorio)' &
                  peek('NOME', RecNo()-1, 'TabelaNomesBanco') & '.qvd')
                  as [Nome da Tabela do QVD],
            Num(100-((Num((FileSize('$(pDiretorio)' &
                  peek('NOME', RecNo()-1, 'TabelaNomesBanco') & '.qvd')/1024),
                  '###.###.###.##0,00')*100)/
                  Num(peek('TAMANHO', RecNo()-1, 'TabelaNomesBanco'),
                  '###.###.###.##0,00')),'###.###.###.##0,00') as [% de Redução]
      Resident TabelaNomesBanco;

end sub
```

9.7.SalvarQVD

```
/*============================================================================
Autor:            Henrique Figueiredo de Souza
Data Criação:     28/01/2013
Descrição:
            Chama os procedimentos na ordem adequada para salvar todos
            os arquivos QVD em disco e criar as tabelas em memória.
============================================================================*/

call listarNomesTabelasBanco(false());
call salvarArquivosQVD('qvd\','BANCO',true(),false(),true());
call criarTabelaArquivosQVD('qvd\');
call criarTabelaPropriedades;

/*
let nome = QvdTableName('qvd\DADOS.qvd');
let num = MsgBox(nome);

let temp = Input('Digite um valor', 'Input box');
let num = MsgBox(temp);

*/
```

10. Os Scripts Qlikview para Carregar QVDs

10.1. listarNomesTabelasBanco

```
/*============================================================================
Procedimento:     listarNomesTabelasBanco
Autor:            Henrique Figueiredo de Souza
Data Criação:     28/01/2013
Descrição:
            Realiza as consultas no banco para retornar os nomes e os
            tamanhos das tabelas do banco, estando essas tabelas vazias
            ou não.
```

```
==========================================================================*/

sub listarNomesTabelasBanco

      TabelaNomesBanco:
      SQL SELECT segment_name NOME, round(sum(bytes/1024),2) TAMANHO
      FROM user_segments
      WHERE segment_type = 'TABLE'
      GROUP BY segment_name
      ORDER BY 2 desc;

      let vNumeroLinhas = NoOfRows('TabelaNomesBanco');

      if  vNumeroLinhas = 0 then

            drop Table TabelaNomesBanco;

            TabelaNomesBanco:
            SQL SELECT TABLE_NAME NOME, 0 TAMANHO
            FROM user_tables
            ORDER BY 1;
      end if

end sub;
```

10.2. carregarArquivosQVD

```
/*==========================================================================
Procedimento:     carregarArquivosQVD
Autor:            Henrique Figueiredo de Souza
Data Criação:     28/01/2013
Parametros:
            pDiretorio        Nome do diretorio onde serão salvos os
                              arquivos QVD.
Descrição:
            Carrega na memória os arquivos QVD salvos dentro do
            diretório passado como parâmetro.
==========================================================================*/

sub carregarArquivosQVD(pDiretorio)

      Let vQtdTabelas = NoOfRows('TabelaNomesBanco');

      QUALIFY *;

      For i = 0 To $(vQtdTabelas) - 1
            LET vNomeTabela = Peek('NOME', $(i), 'TabelaNomesBanco');
            $(vNomeTabela):
            LOAD * From $(pDiretorio)$(vNomeTabela).qvd (qvd);
      Next i

      UNQUALIFY *;

end sub;
```

10.3. CarregarQVDs

```
call listarNomesTabelasBanco;
call carregarArquivosQVD('qvd\');
drop table TabelaNomesBanco;
```

11. O Pacote Qlikview para o ORACLE

```
SET SERVEROUTPUT ON
EXEC PacoteQlikview.ListarTabelasColunas(TRUE);
EXEC PacoteQlikview.MontarTodosLoad('Henrique Figueiredo de Souza', FALSE);
EXEC PacoteQlikview.MontarTodosMappingLoad('Henrique Figueiredo de Souza',
  FALSE, TRUE);
EXEC PacoteQlikview.ListarTabelasRestricoes;
EXEC PacoteQlikview.TabelasNaoPossuem_PK_nemFK;
EXEC PacoteQlikview.TabelasPossuemSomente_PK_naoFK;
EXEC PacoteQlikview.TabelasPossuem_FK;
EXEC PacoteQlikview.TabelasPossuemSomente_FK_naoPK;
EXEC PacoteQlikview.DescomentarTabelasColunas;

begin
DBMS_OUTPUT.put_line (PacoteQlikview.AplicarMapa('TABELA', 'COLUNA'));
end;

DROP PACKAGE BODY PacoteQlikview;
DROP PACKAGE PacoteQlikview;

CREATE OR REPLACE PACKAGE PacoteQlikview
AS
      PROCEDURE MontarLoad(p_tabela IN varchar2, p_autor IN varchar2,
          p_mapeada IN boolean, p_2_colunas IN boolean);
      PROCEDURE MontarTodosLoad(p_autor IN varchar2, p_vazia IN boolean);
      PROCEDURE MontarTodosMappingLoad(p_autor IN varchar2, p_vazia IN boolean,
          p_2_colunas IN boolean);
      FUNCTION AplicarMapa(p_tabela IN varchar2, p_coluna IN varchar2)
            RETURN VARCHAR2;
      PROCEDURE ListarTabelasColunas(p_comentario IN boolean);
      FUNCTION ColunasDaRestricaoDaTabela(p_tabela IN varchar2,
          p_restricao IN varchar2) RETURN VARCHAR2;
      PROCEDURE ListarTabelasRestricoes;
      PROCEDURE ListarTabelaRestricao(p_tabela IN varchar2);
      PROCEDURE TabelasNaoPossuem_PK_nemFK;
      PROCEDURE TabelasPossuemSomente_PK_naoFK;
      PROCEDURE TabelasPossuem_FK;
      PROCEDURE TabelasPossuemSomente_FK_naoPK;
      FUNCTION LimitaTamanhoTexto(p_str IN VARCHAR2, p_limite IN integer)
            RETURN VARCHAR2;
      PROCEDURE DescomentarTabelasColunas;
END PacoteQlikview;
/
CREATE OR REPLACE PACKAGE BODY PacoteQlikview
AS
...
END PacoteQlikview;
/
```

11.1. O Procedimento MontarLoad

```
/*================================================================================
Procedimento:     MontarLoad
Autor:            Henrique Figueiredo de Souza
Data Criação:     03/02/2013
Parametros:
            p_tabela    Nome da tabela a gerar o LOAD.
            p_autor           Nome do autor a ser mostrado no comentário.
            p_mapeada   TRUE para mapping load, FALSE para sem mapping load
            p_2_colunas Limita a saida a somente a duas colunas.
Descrição:
```

```
            Monta um LOAD em formato qlikview de uma tabela
================================================================================*/

PROCEDURE MontarLoad(p_tabela IN varchar2, p_autor IN varchar2,
p_mapeada IN boolean, p_2_colunas IN boolean)
IS
      v_tabela VARCHAR2(61);
      v_coluna VARCHAR2(32676);
      v_comenta VARCHAR2(8000);
      v_rotulo VARCHAR2(4000);
      v_rotulo_tabela VARCHAR2(4000);
      v_mapa VARCHAR2(4000);
      v_nomecoluna VARCHAR2(4000);
      v_cr CHAR(2);
      v_conta INTEGER;
BEGIN
      v_cr := chr(13) || chr(10);
      v_coluna := '';
      v_conta := 0;

      FOR c_coluna IN (SELECT utc.table_name, utc.column_name, utc.column_id,
                        (select ucc.comments from user_col_comments ucc
                        where ucc.table_name=utc.table_name
                        and ucc.column_name=utc.column_name) comments,
                        (select utc1.comments from user_tab_comments utc1
                        where utc1.table_name=ut.table_name
                          and utc1.table_type='TABLE') tab_comments
                  FROM user_tables ut, user_tab_columns utc
                  where ut.table_name=utc.table_name and ut.table_name=p_tabela
                  order by utc.column_id)
      LOOP
            v_tabela := c_coluna.table_name;
            v_rotulo := c_coluna.comments;
            v_rotulo_tabela := LimitaTamanhoTexto(c_coluna.tab_comments, 10);
            v_mapa := AplicarMapa(v_tabela, c_coluna.column_name);

            if length(v_mapa) > 0 then
                  v_nomecoluna := v_mapa;
            else
                  v_nomecoluna := c_coluna.column_name;
            end if;

            if v_rotulo is null then
                  v_coluna := v_coluna || ', ' || v_cr || '     ' ||
                  v_nomecoluna;
            else
                  v_coluna := v_coluna || ', ' || v_cr || '     ' ||
                  v_nomecoluna || ' as [' || v_rotulo || ']';
            end if;

            if p_2_colunas=TRUE then
                  v_conta := v_conta + 1;

                  if v_conta=2 then
                        EXIT;
                  end if;
            end if;

      END LOOP;

      if length(v_coluna) > 0 then

            v_coluna := substr(v_coluna, 3, length(v_coluna));
```

```
            v_comenta := v_cr ||

'/*============================================================================
' || v_cr;
            if p_mapeada=TRUE then
                  v_comenta := v_comenta || 'Procedimento:' || chr(9) ||
      'Carga mapeada em memória do arquivo QVD' || v_cr;
            else
                  v_comenta := v_comenta || 'Procedimento:' || chr(9) ||
      'Carga em memória do arquivo QVD' || v_cr;
            end if;
            v_comenta := v_comenta || 'ArquivoQVD:' || chr(9) || chr(9)
    || v_tabela || v_cr;
            v_comenta := v_comenta || 'Autor:' || chr(9) || chr(9) || chr(9)
    || p_autor || v_cr;
            v_comenta := v_comenta || 'Data Criação:' || chr(9)
    || to_char(sysdate, 'dd/mm/yyyy') || v_cr;

            if v_rotulo_tabela is not null then
                  if  length(v_rotulo_tabela) > 0 then
                        v_comenta := v_comenta || 'Descrição:' || chr(9) ||
                        v_rotulo_tabela || v_cr;
                  end if;
            end if;

            DBMS_OUTPUT.put_line(v_comenta || 'Restrições:');

            ListarTabelaRestricao(p_tabela);

            DBMS_OUTPUT.put_line(

      '============================================================================
====*/' || v_cr);

            if p_mapeada=TRUE then
                  DBMS_OUTPUT.put_line ('Mapa' || v_tabela || ':' || v_cr ||
                  'Mapping LOAD' || v_coluna || v_cr || 'FROM' ||
                  v_cr || '$(vDiretorio)'
                  || v_tabela || '.qvd' || v_cr || '(qvd);' || v_cr);
            else
                  DBMS_OUTPUT.put_line (v_tabela || ':' || v_cr ||
                  'LOAD' || v_coluna || v_cr || 'FROM' ||
                  v_cr || '$(vDiretorio)'
                  || v_tabela || '.qvd' || v_cr || '(qvd);' || v_cr);
            end if;

      end if;

EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Um erro foi encontrado - '
   ||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

11.2. O Procedimento MontarTodosLoad

```
/*============================================================================
Procedimento:     MontarTodosLoad
Autor:            Henrique Figueiredo de Souza
Data Criação:     03/02/2013
Parametros:
            p_autor           Nome do autor a ser mostrado no comentário.
```

```
            p_vazia           Se TRUE verificar se a tabela está vazia.
Descrição:
            Monta todos os LOAD em formato qlikview de todas as tabelas
================================================================================*/

PROCEDURE MontarTodosLoad(p_autor IN varchar2, p_vazia IN boolean)
IS
      v_tabela VARCHAR2(61);
      v_conta INTEGER;
BEGIN
      FOR c_tabela IN (SELECT ut.table_name
                  FROM user_tables ut order by ut.table_name)
      LOOP
            v_tabela := c_tabela.table_name;

            if p_vazia = TRUE then
                  EXECUTE IMMEDIATE 'SELECT COUNT(*) FROM ' ||
                  v_tabela INTO v_conta;

                  if v_conta > 0 then
                        MontarLoad(v_tabela, p_autor, FALSE, FALSE);
                  end if;
            else
                  MontarLoad(v_tabela, p_autor, FALSE, FALSE);
            end if;

      END LOOP;

EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Um erro foi encontrado - '
   ||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

11.3. O Procedimento MontarTodosMappingLoad

```
/*================================================================================
Procedimento:     MontarTodosMappingLoad
Autor:            Henrique Figueiredo de Souza
Data Criação:     03/02/2013
Parametros:
            p_autor           Nome do autor a ser mostrado no comentário.
            p_vazia           Se 1 verificar se a tabela está vazia.
            p_2_colunas Limita a saida a somente a duas colunas.
Descrição:
            Monta todos os Mapping LOAD em formato qlikview
            de todas as tabelas que possuem SOMENTE Chave Primária.
================================================================================*/

PROCEDURE MontarTodosMappingLoad(p_autor IN varchar2,
p_vazia IN boolean, p_2_colunas IN boolean)
IS
      v_tabela VARCHAR2(61);
      v_conta INTEGER;
BEGIN
      FOR c_tabela IN (select uo.OBJECT_NAME
            from (select uo1.OBJECT_NAME, uo1.CREATED
          from user_objects uo1
              WHERE uo1.OBJECT_TYPE='TABLE' AND uo1.STATUS='VALID'
                  AND uo1.OBJECT_NAME IN (SELECT DISTINCT uc1.TABLE_NAME
      FROM user_constraints uc1 WHERE uc1.CONSTRAINT_TYPE
      IN ('P','R'))) uo
```

```
            WHERE uo.OBJECT_NAME NOT IN
    (SELECT DISTINCT uc.TABLE_NAME FROM user_constraints uc
    WHERE uc.CONSTRAINT_TYPE='R')
            ORDER BY TO_CHAR(uo.CREATED,'DD/MM/YYYY HH24:MI:SS:SSSSS') ASC)
      LOOP
            v_tabela := c_tabela.OBJECT_NAME;

            if p_vazia = TRUE then
                  EXECUTE IMMEDIATE 'SELECT COUNT(*) FROM ' ||
      v_tabela INTO v_conta;

                  if v_conta > 0 then
                        MontarLoad(v_tabela, p_autor, TRUE, p_2_colunas);
                  end if;
            else
                  MontarLoad(v_tabela, p_autor, TRUE, p_2_colunas);
            end if;
      END LOOP;

EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Um erro foi encontrado - '
   ||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

11.4. A Função AplicarMapa

```
/*=============================================================================
Procedimento:     AplicarMapa
Autor:            Henrique Figueiredo de Souza
Data Criação:     03/02/2013
Parametros:
            p_tabela    Nome da tabela referenciada pela FK.
            p_coluna    Nome da coluna que possui uma FK.
Descrição:
            Aplicar o mapa carregado, onde a coluna é uma
            Chave Estrangeira e tabela é a tabela referenciada pela
            chave estrangeira da coluna.
=============================================================================*/

FUNCTION AplicarMapa(p_tabela IN varchar2, p_coluna IN varchar2)
RETURN VARCHAR2
IS
      v_mapa VARCHAR2(8000);
      v_tabela VARCHAR2(61);
      v_tabela_mapa VARCHAR2(61);

      CURSOR c_mapa is
            SELECT (SELECT distinct uc1.TABLE_NAME
    FROM user_constraints uc1
    WHERE uc1.CONSTRAINT_NAME=uc.R_CONSTRAINT_NAME) R_TABLE_NAME
            FROM user_constraints uc, user_cons_columns ucc
            WHERE uc.TABLE_NAME=ucc.TABLE_NAME
            AND uc.CONSTRAINT_NAME=ucc.CONSTRAINT_NAME
            AND uc.CONSTRAINT_TYPE='R'
            AND uc.TABLE_NAME=p_tabela
            AND ucc.COLUMN_NAME=p_coluna
            and ucc.POSITION IS NOT NULL;

      CURSOR c_tabela is
            select uo.OBJECT_NAME
            from (select uo1.OBJECT_NAME, uo1.CREATED
```

```
      from user_objects uo1
              WHERE uo1.OBJECT_TYPE='TABLE' AND uo1.STATUS='VALID'
                  AND uo1.OBJECT_NAME IN (SELECT DISTINCT uc1.TABLE_NAME
        FROM user_constraints uc1
        WHERE uc1.CONSTRAINT_TYPE IN ('P','R'))) uo
            WHERE uo.OBJECT_NAME NOT IN (SELECT DISTINCT uc.TABLE_NAME
      FROM user_constraints uc WHERE uc.CONSTRAINT_TYPE='R')
            AND uo.OBJECT_NAME=v_tabela_mapa;

BEGIN
      v_mapa := '';

      -- verifica se a coluna da tabela é uma Chave Estrangeira
      open c_mapa;
      fetch c_mapa into v_tabela_mapa;

      if c_mapa%found then

            open c_tabela;
            fetch c_tabela into v_tabela;

            -- verifica se tabela possui somente Chave Primaria
            if c_tabela%found then
                  close c_tabela;

                  v_mapa:= 'ApplyMap(' || chr(39) || 'Mapa' ||
      v_tabela_mapa || chr(39) || ',' || p_coluna || ')';
            else
                  close c_tabela;
            end if;

            close c_mapa;
      else
            close c_mapa;
      end if;

      RETURN v_mapa;

EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Um erro foi encontrado - '
   ||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

11.5. O Procedimento ListarTabelasColunas

```
/*================================================================================
Procedimento:     ListarTabelasColunas
Autor:            Henrique Figueiredo de Souza
Data Criação:     03/02/2013
Parametros:
            p_comentario      Se TRUE incluir comando para gerar comentario.
Descrição:
            Lista todas a tabelas e colunas do banco de dados
================================================================================*/

PROCEDURE ListarTabelasColunas(p_comentario IN boolean)
IS
      v_tabela_anterior VARCHAR2(61);
      v_coluna VARCHAR2(8000);
      v_cr CHAR(2);
      v_conta integer;
```

```
BEGIN
      v_cr := chr(13) || chr(10);
      v_tabela_anterior := '';
      v_conta := 0;

      FOR c_tabela IN (SELECT utc.table_name, utc.column_name
                  FROM user_tables ut, user_tab_columns utc
                  where ut.table_name=utc.table_name
                  order by utc.table_name, utc.column_id)
      LOOP

            if v_tabela_anterior<>c_tabela.table_name then
                  DBMS_OUTPUT.put_line (v_coluna);
                  v_coluna:='';
                  v_conta := 0;
            end if;

            if p_comentario=TRUE then

                  if v_conta=0 then
                        v_coluna := 'comment on table ' ||
        c_tabela.table_name || ' is ' || chr(39) || chr(9)
        || chr(9) || chr(39) || ';' || v_cr || v_coluna;
                        v_conta := 1;
                  end if;

                  v_coluna := v_coluna || 'comment on column ' ||
      c_tabela.table_name || '.' || c_tabela.column_name ||
      ' is ' || chr(39) || chr(9) || chr(9) || chr(39)
      || ';' || v_cr;
            else
                  v_coluna := v_coluna || c_tabela.table_name
      || '.' || c_tabela.column_name || v_cr;
            end if;

            v_tabela_anterior := c_tabela.table_name;

      END LOOP;

      DBMS_OUTPUT.put_line (v_coluna);

EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Um erro foi encontrado - '
   ||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

11.6. A Função ColunasDaRestricaoDaTabela

```
/*=============================================================================
Procedimento:     ColunasDaRestricaoDaTabela
Autor:            Henrique Figueiredo de Souza
Data Criação:     03/02/2013
Parametros:
            p_tabela    Nome da tabela referenciada pela FK.
            p_restricao Nome da restrição.
Descrição:
            Retorna as colunas de uma tabela e sua restrição.
=============================================================================*/

FUNCTION ColunasDaRestricaoDaTabela(p_tabela IN varchar2,
p_restricao IN varchar2)
```

```
RETURN VARCHAR2
IS
      v_coluna VARCHAR2(8000);
BEGIN

      v_coluna := '';

      FOR c_tabela IN (SELECT ucc.COLUMN_NAME
    FROM user_cons_columns ucc where ucc.TABLE_NAME=p_tabela
    and ucc.CONSTRAINT_NAME=p_restricao
    and ucc.POSITION IS NOT NULL)
      LOOP

            v_coluna := v_coluna || ', ' || c_tabela.column_name;

      END LOOP;

  v_coluna := substr(v_coluna, 3, length(v_coluna));

  RETURN v_coluna;

EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Um erro foi encontrado - '
   ||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

11.7. O Procedimento ListarTabelasRestricoes

```
/*================================================================================
Procedimento:     ListarTabelasRestricoes
Autor:            Henrique Figueiredo de Souza
Data Criação:     03/02/2013
Descrição:
            Lista todas as tabelas e suas restrições.
================================================================================*/

PROCEDURE ListarTabelasRestricoes
IS
      v_tabela_anterior VARCHAR2(61);
      v_tabela VARCHAR2(8000);
      v_restricao VARCHAR2(8000);
      v_cr CHAR(2);
BEGIN
      v_cr := chr(13) || chr(10);
      v_tabela_anterior := '';
      v_restricao := '';

      FOR c_tabela IN (SELECT uc.TABLE_NAME, uc.CONSTRAINT_NAME,
    uc.CONSTRAINT_TYPE, uc.R_CONSTRAINT_NAME,
                  (SELECT distinct uc1.TABLE_NAME
      FROM user_constraints uc1
      WHERE uc1.CONSTRAINT_NAME=uc.R_CONSTRAINT_NAME) R_TABLE_NAME
                  FROM user_constraints uc
                  WHERE uc.CONSTRAINT_TYPE in ('P','R')
                  order by uc.TABLE_NAME, uc.CONSTRAINT_TYPE)
      LOOP

            if v_tabela_anterior<>c_tabela.table_name then
                  DBMS_OUTPUT.put_line (v_restricao);
                  v_restricao:='';
            end if;
```

```
            if c_tabela.CONSTRAINT_TYPE='P' then
                  v_restricao := v_restricao || c_tabela.CONSTRAINT_NAME
      || ' = ' || c_tabela.table_name || '(' ||
      ColunasDaRestricaoDaTabela(c_tabela.table_name,
      c_tabela.CONSTRAINT_NAME) || ')' || v_cr;
            else
                  v_restricao := v_restricao || c_tabela.CONSTRAINT_NAME
      || ' = ' || c_tabela.table_name || '(' ||
      ColunasDaRestricaoDaTabela(c_tabela.table_name,
      c_tabela.CONSTRAINT_NAME) || ') --> ' ||
      c_tabela.R_TABLE_NAME || '(' ||
      ColunasDaRestricaoDaTabela(c_tabela.R_TABLE_NAME,
      c_tabela.R_CONSTRAINT_NAME) || ')' || v_cr;
            end if;

            v_tabela_anterior := c_tabela.table_name;

      END LOOP;

      DBMS_OUTPUT.put_line (v_restricao);

EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Um erro foi encontrado - '
   ||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

11.8. O Procedimento ListarTabelaRestricao

```
/*================================================================================
Procedimento:     ListarTabelaRestricao
Autor:            Henrique Figueiredo de Souza
Data Criação:     03/02/2013
Parametros:
            p_tabela    Nome da tabela.
Descrição:
            Lista restrições de uma tabela.
================================================================================*/

PROCEDURE ListarTabelaRestricao(p_tabela IN varchar2)
IS
      v_tabela_anterior VARCHAR2(61);
      v_tabela VARCHAR2(8000);
      v_restricao VARCHAR2(8000);
      v_cr CHAR(2);
      v_tam INTEGER;
BEGIN
      v_cr := chr(13) || chr(10);
      v_tabela_anterior := '';
      v_restricao := '';

      FOR c_tabela IN (SELECT uc.TABLE_NAME, uc.CONSTRAINT_NAME,
                  uc.CONSTRAINT_TYPE, uc.R_CONSTRAINT_NAME,
                  (SELECT distinct uc1.TABLE_NAME FROM user_constraints uc1
                        WHERE uc1.CONSTRAINT_NAME=uc.R_CONSTRAINT_NAME)
R_TABLE_NAME
                  FROM user_constraints uc
                  WHERE uc.CONSTRAINT_TYPE in ('P','R') AND
uc.TABLE_NAME=p_tabela
                  order by uc.TABLE_NAME, uc.CONSTRAINT_TYPE)
      LOOP
```

```
            if v_tabela_anterior<>c_tabela.table_name then
                  DBMS_OUTPUT.put_line (v_restricao);
                  v_restricao:='';
            end if;

            if c_tabela.CONSTRAINT_TYPE='P' then
                  v_restricao := v_restricao || c_tabela.table_name || '(' ||
                  ColunasDaRestricaoDaTabela(c_tabela.table_name,
                  c_tabela.CONSTRAINT_NAME) || ')' || v_cr;
            else
                  v_restricao := v_restricao || c_tabela.table_name || '(' ||
                  ColunasDaRestricaoDaTabela(c_tabela.table_name,
                  c_tabela.CONSTRAINT_NAME) || ') --> ' ||
                  c_tabela.R_TABLE_NAME || '(' ||
                  ColunasDaRestricaoDaTabela(c_tabela.R_TABLE_NAME,
                  c_tabela.R_CONSTRAINT_NAME) || ')' || v_cr;
            end if;

            v_tabela_anterior := c_tabela.table_name;

      END LOOP;

      v_tam := length(v_restricao)-2;

      if v_tam > 0 then
            v_restricao := substr(v_restricao, 1, v_tam);
            DBMS_OUTPUT.put_line (v_restricao);
      end if;

EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Um erro foi encontrado - '
                  ||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

11.9. O Procedimento TabelasNaoPossuem_PK_nemFK

```
/*================================================================================
Procedimento:     TabelasNaoPossuem_PK_nemFK
Autor:            Henrique Figueiredo de Souza
Data Criação:     03/02/2013
Descrição:
            Lista todas as tabelas que NÃO possuem Chave Primária e
            não possuem Chave Estrangeira, ordenados por data de criação.
================================================================================*/

PROCEDURE TabelasNaoPossuem_PK_nemFK
IS
      v_tabela VARCHAR2(8000);
      v_cr CHAR(2);
BEGIN
      v_cr := chr(13) || chr(10);
      v_tabela := '';

      FOR c_objeto IN (select uo.OBJECT_NAME
                  from user_objects uo
                  WHERE uo.OBJECT_TYPE='TABLE' AND uo.STATUS='VALID'
                  AND uo.OBJECT_NAME NOT IN (SELECT DISTINCT uc.TABLE_NAME
      FROM user_constraints uc
      WHERE uc.CONSTRAINT_TYPE IN ('P','R'))
```

```
                ORDER BY TO_CHAR(uo.CREATED,'DD/MM/YYYY HH24:MI:SS:SSSSS')
ASC)
      LOOP
            v_tabela := v_tabela || c_objeto.OBJECT_NAME || v_cr;

      END LOOP;

      DBMS_OUTPUT.put_line (v_tabela);

EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Um erro foi encontrado - '
   ||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

11.10. O Procedimento TabelasPossuemSomente_PK_naoFK

```
/*================================================================================
Procedimento:     TabelasPossuemSomente_PK_naoFK
Autor:            Henrique Figueiredo de Souza
Data Criação:     03/02/2013
Descrição:
            Lista todas as tabelas que possuem SOMENTE Chave Primária e
            não possuem Chave Estrangeira, ordenados por data de criação.
================================================================================*/

PROCEDURE TabelasPossuemSomente_PK_naoFK
IS
      v_restricao VARCHAR2(8000);
      v_cr CHAR(2);
BEGIN
      v_cr := chr(13) || chr(10);

      FOR c_objeto IN (select uo.OBJECT_NAME
            from (select uo1.OBJECT_NAME, uo1.CREATED
    from user_objects uo1
              WHERE uo1.OBJECT_TYPE='TABLE' AND uo1.STATUS='VALID'
                  AND uo1.OBJECT_NAME IN (SELECT DISTINCT uc1.TABLE_NAME
      FROM user_constraints uc1
      WHERE uc1.CONSTRAINT_TYPE IN ('P','R'))) uo
            WHERE uo.OBJECT_NAME NOT IN (SELECT DISTINCT uc.TABLE_NAME
    FROM user_constraints uc WHERE uc.CONSTRAINT_TYPE='R')
            ORDER BY TO_CHAR(uo.CREATED,'DD/MM/YYYY HH24:MI:SS:SSSSS') ASC)
      LOOP
            v_restricao := '';

            FOR c_tabela IN (SELECT uc.TABLE_NAME, uc.CONSTRAINT_NAME,
    uc.CONSTRAINT_TYPE, uc.R_CONSTRAINT_NAME,
                        (SELECT distinct uc1.TABLE_NAME
        FROM user_constraints uc1
        WHERE uc1.CONSTRAINT_NAME=uc.R_CONSTRAINT_NAME) R_TABLE_NAME
                        FROM user_constraints uc
                        WHERE uc.TABLE_NAME=c_objeto.OBJECT_NAME)
            LOOP

                  if c_tabela.CONSTRAINT_TYPE='P' then
                        v_restricao := v_restricao || c_tabela.CONSTRAINT_NAME
        || ' = ' || c_tabela.table_name || '(' ||
        ColunasDaRestricaoDaTabela(c_tabela.table_name,
        c_tabela.CONSTRAINT_NAME) || ')' || v_cr;
                  end if;
```

```
            END LOOP;

            DBMS_OUTPUT.put_line (v_restricao);

      END LOOP;

EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Um erro foi encontrado - '
   ||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

11.11. O Procedimento TabelasPossuem_FK

```
/*================================================================================
Procedimento:     TabelasPossuem_FK
Autor:            Henrique Figueiredo de Souza
Data Criação:     03/02/2013
Descrição:
            Lista todas as tabelas que possuem Chave Estrangeira e podem
            ou não possuir Chave Primária, ordenados por data de criação.
================================================================================*/

PROCEDURE TabelasPossuem_FK
IS
      v_restricao VARCHAR2(8000);
      v_cr CHAR(2);
BEGIN
      v_cr := chr(13) || chr(10);

      FOR c_objeto IN (select uo.OBJECT_NAME
                  from user_objects uo
                  WHERE uo.OBJECT_TYPE='TABLE' AND uo.STATUS='VALID'
                  AND uo.OBJECT_NAME IN (SELECT DISTINCT uc.TABLE_NAME
      FROM user_constraints uc WHERE uc.CONSTRAINT_TYPE='R')
                  ORDER BY TO_CHAR(uo.CREATED,'DD/MM/YYYY HH24:MI:SS:SSSSS')
ASC)
      LOOP
            v_restricao := '';

            FOR c_tabela IN (SELECT uc.TABLE_NAME, uc.CONSTRAINT_NAME,
    uc.CONSTRAINT_TYPE, uc.R_CONSTRAINT_NAME,
                        (SELECT distinct uc1.TABLE_NAME
        FROM user_constraints uc1
        WHERE uc1.CONSTRAINT_NAME=uc.R_CONSTRAINT_NAME) R_TABLE_NAME
                        FROM user_constraints uc
                        WHERE uc.TABLE_NAME=c_objeto.OBJECT_NAME)
            LOOP

                  if c_tabela.CONSTRAINT_TYPE='R' then
                        v_restricao := v_restricao || c_tabela.CONSTRAINT_NAME
        || ' = ' || c_tabela.table_name || '(' ||
        ColunasDaRestricaoDaTabela(c_tabela.table_name,
        c_tabela.CONSTRAINT_NAME) || ') --> ' ||
        c_tabela.R_TABLE_NAME || '(' ||
        ColunasDaRestricaoDaTabela(c_tabela.R_TABLE_NAME,
        c_tabela.R_CONSTRAINT_NAME) || ')' || v_cr;
            end if;

            END LOOP;

            DBMS_OUTPUT.put_line (v_restricao);
```

```
      END LOOP;

EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Um erro foi encontrado - '
   ||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

11.12. O Procedimento TabelasPossuemSomente_FK_naoPK

```
/*================================================================================
Procedimento:     TabelasPossuemSomente_FK_naoPK
Autor:            Henrique Figueiredo de Souza
Data Criação:     03/02/2013
Descrição:
            Lista todas as tabelas que possuem SOMENTE Chave Estrangeira
            e não possuem Chave Primária, ordenados por data de criação.
================================================================================*/

PROCEDURE TabelasPossuemSomente_FK_naoPK
IS
      v_restricao VARCHAR2(8000);
      v_cr CHAR(2);
BEGIN
      v_cr := chr(13) || chr(10);

      FOR c_objeto IN (select uo.OBJECT_NAME
                  from (select uo1.OBJECT_NAME, uo1.CREATED
      from user_objects uo1
                     WHERE uo1.OBJECT_TYPE='TABLE' AND uo1.STATUS='VALID'
                         AND uo1.OBJECT_NAME NOT IN (SELECT DISTINCT
uc1.TABLE_NAME
        FROM user_constraints uc1 WHERE uc1.CONSTRAINT_TYPE='P')) uo
                  WHERE uo.OBJECT_NAME IN (SELECT DISTINCT uc.TABLE_NAME
      FROM user_constraints uc WHERE uc.CONSTRAINT_TYPE='R')
                  ORDER BY TO_CHAR(uo.CREATED,'DD/MM/YYYY HH24:MI:SS:SSSSS')
ASC)
      LOOP
            v_restricao := '';

            FOR c_tabela IN (SELECT uc.TABLE_NAME, uc.CONSTRAINT_NAME,
    uc.CONSTRAINT_TYPE, uc.R_CONSTRAINT_NAME,
                        (SELECT distinct uc1.TABLE_NAME FROM user_constraints
uc1
        WHERE uc1.CONSTRAINT_NAME=uc.R_CONSTRAINT_NAME) R_TABLE_NAME
                        FROM user_constraints uc
                        WHERE uc.TABLE_NAME=c_objeto.OBJECT_NAME)
            LOOP

                  if c_tabela.CONSTRAINT_TYPE='R' then
                        v_restricao := v_restricao || c_tabela.CONSTRAINT_NAME
        || ' = ' || c_tabela.table_name || '(' ||
        ColunasDaRestricaoDaTabela(c_tabela.table_name,
        c_tabela.CONSTRAINT_NAME) || ') --> ' ||
        c_tabela.R_TABLE_NAME || '(' ||
        ColunasDaRestricaoDaTabela(c_tabela.R_TABLE_NAME,
        c_tabela.R_CONSTRAINT_NAME) || ')' || v_cr;
            end if;

            END LOOP;
```

```
            DBMS_OUTPUT.put_line (v_restricao);

      END LOOP;

EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Um erro foi encontrado - '
   ||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

11.13. A Função LimitaTamanhoTexto

```
/*=============================================================================
Procedimento:     LimitaTamanhoTexto
Autor:            Henrique Figueiredo de Souza
Data Criação:     05/02/2013
Parametros:
            p_str       Texto a ser limitado.
            p_limite    Limite de palavras por linhas.
Descrição:
            Limita o texto fornecido a uma determinada quantidades
            de palavras por linha.
=============================================================================*/

FUNCTION LimitaTamanhoTexto(p_str IN VARCHAR2,
p_limite IN integer)
RETURN VARCHAR2
AS
      v_palavras PLS_INTEGER := 0;
      v_tamanho PLS_INTEGER := NVL(LENGTH(p_str),0);
      v_dentro_uma_palavra BOOLEAN;
      v_texto VARCHAR2(32676);
BEGIN
      v_texto := '';
      FOR i IN 1..v_tamanho + 1
      LOOP
            v_texto := v_texto || SUBSTR(p_str, i, 1);

            IF ASCII(SUBSTR(p_str, i, 1)) < 33 OR i > v_tamanho THEN
                  IF v_dentro_uma_palavra THEN
                        v_palavras := v_palavras + 1;
                        v_dentro_uma_palavra := FALSE;

                        if v_palavras=p_limite then
                              v_texto := v_texto || chr(13) || chr(10);
                              v_palavras := 0;
                        end if;
                  END IF;
            ELSE
                  v_dentro_uma_palavra := TRUE;
            END IF;
      END LOOP;

      RETURN v_texto;

EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Um erro foi encontrado - '
            ||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

11.14. O Procedimento DescomentarTabelasColunas

```
/*================================================================================
Procedimento:     DescomentarTabelasColunas
Autor:            Henrique Figueiredo de Souza
Data Criação:     09/02/2013
Descrição:
            Descomentar todas a tabelas e colunas do banco de dados
================================================================================*/

PROCEDURE DescomentarTabelasColunas
IS
      v_tabela_anterior VARCHAR2(61);
      v_coluna VARCHAR2(8000);
      v_cr CHAR(2);
      v_conta integer;
BEGIN
      v_cr := chr(13) || chr(10);
      v_tabela_anterior := '';
      v_conta := 0;

      FOR c_tabela IN (SELECT utc.table_name, utc.column_name
                  FROM user_tables ut, user_tab_columns utc
                  where ut.table_name=utc.table_name
                  order by utc.table_name, utc.column_id)
      LOOP

            if v_tabela_anterior<>c_tabela.table_name then
                  DBMS_OUTPUT.put_line (v_coluna);
                  v_coluna:='';
                  v_conta := 0;
            end if;

            if v_conta=0 then
                  v_coluna := 'comment on table ' || c_tabela.table_name
      || ' is ' || chr(39) || chr(39) || ';' || v_cr || v_coluna;
                  v_conta := 1;
            end if;

            v_coluna := v_coluna || 'comment on column ' ||
    c_tabela.table_name || '.' || c_tabela.column_name ||
    ' is ' || chr(39) || chr(39) || ';' || v_cr;

            v_tabela_anterior := c_tabela.table_name;

      END LOOP;

      DBMS_OUTPUT.put_line (v_coluna);

EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Um erro foi encontrado - '
   ||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

11.15. A Função ContaPalavra para o ORACLE

```
CREATE OR REPLACE FUNCTION ContaPalavra(str IN VARCHAR2)
RETURN PLS_INTEGER
AS
      palavras PLS_INTEGER := 0;
      tamanho PLS_INTEGER := NVL(LENGTH(str),0);
```

```
      dentro_uma_palavra BOOLEAN;
BEGIN
      FOR i IN 1..tamanho + 1
      LOOP
            IF ASCII(SUBSTR(str, i, 1)) < 33 OR i > tamanho THEN
                  IF dentro_uma_palavra THEN
                        palavras := palavras + 1;
                        dentro_uma_palavra := FALSE;
                  END IF;
            ELSE
                  dentro_uma_palavra := TRUE;
            END IF;
      END LOOP;
      RETURN palavras;
END;
```

12. Os Procedimentos Qlikview para o SQLSERVER

```
exec dbo.QV_MontarTodosLoad 'Henrique Figueiredo de Souza', 0
exec dbo.QV_MontarTodosMappingLoad 'Henrique Figueiredo de Souza', 0, 1

select dbo.QV_AplicarMapa('ext_pergunta', 'cod_tipo_atendimento');
exec dbo.QV_ListarTabelasColunas 1;
exec dbo.QV_ListarTabelasRestricoes;
exec dbo.QV_TabelasNaoPossuem_PK_nemFK;
exec dbo.QV_TabelasSomente_PK_naoFK;
exec dbo.QV_TabelasPossuem_FK;
exec dbo.QV_TabelasSomente_FK_naoPK;
EXEC dbo.QV_DescomentarTabelasColunas;

DROP FUNCTION QV_LimitaTamanhoTexto;
DROP FUNCTION QV_AplicarMapa;
DROP PROCEDURE QV_MontarLoad;
DROP PROCEDURE QV_MontarTodosMappingLoad;
DROP PROCEDURE QV_MontarTodosLoad;
DROP PROCEDURE dbo.QV_ContarLinhasTabela
DROP FUNCTION QV_ColunasDaRestricaoDaTabela;
DROP PROCEDURE QV_ListarTabelasColunas;
DROP PROCEDURE QV_ListarTabelasRestricoes;
DROP PROCEDURE QV_ListarTabelaRestricao;
DROP PROCEDURE QV_TabelasNaoPossuem_PK_nemFK;
DROP PROCEDURE QV_TabelasSomente_PK_naoFK;
DROP PROCEDURE QV_TabelasPossuem_FK;
DROP PROCEDURE QV_TabelasSomente_FK_naoPK;
DROP PROCEDURE QV_DescomentarTabelasColunas;

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
```

12.1. O Procedimento QV_CarregarTamanhoTabela

```
CREATE TABLE QV_TabelasTamanho(
name        sysname     not null,
rows        int   null,
reserved    varchar(25) null,
data        varchar(25) null,
index_size  varchar(25) null,
unused            varchar(25) null,
CONSTRAINT PK_TabelasTamanho PRIMARY KEY(name)
```

```
);
go

create view QV_TamanhoTabela
as
select name as 'NomeTabela',
   rows as 'Linhas',
   convert(int, replace(reserved, ' KB','')) as Tamanho_KB,
   convert(int, replace(data, ' KB',''))as Dados_KB,
   convert(int, replace(index_size, ' KB',''))as Index_KB,
   convert(int, replace(unused, ' KB',''))as Nao_Utilizado
from QV_TabelasTamanho
where name<>'QV_TabelasTamanho';
go

drop procedure QV_CarregarTamanhoTabela;


SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

/*============================================================================
Procedimento:     QV_CarregarTamanhoTabela
Autor:            Henrique Figueiredo de Souza
Data Criação:     08/02/2013
Descrição:
            Carrega a tabela QV_TabelasTamanho com o tamanho e quantidade
            de linhas de cada tabela.
============================================================================*/

create procedure QV_CarregarTamanhoTabela
as
begin
      declare @vname sysname
      declare cp1 cursor local fast_forward read_only for
            select name from sysobjects where type = 'U' order by name

      delete from QV_TabelasTamanho;

      open cp1

      while 1 = 1
      begin
            fetch next from cp1 into @vname

            if @@fetch_status <> 0
                  break

            insert into QV_TabelasTamanho(name, rows, reserved, data,
    index_size, unused)

            exec sp_spaceused @vname
      end

      close cp1
      deallocate cp1
end
go

exec QV_CarregarTamanhoTabela;

select * from QV_TamanhoTabela
```

```
order by convert(int, replace(Tamanho_KB, ' KB','')) desc;

SET ROWCOUNT 1;
select NomeTabela as NOME, Tamanho_KB as TAMANHO,
ROW_NUMBER() OVER (ORDER BY
convert(int, replace(Tamanho_KB, ' KB','')) desc) AS ROWNUM
from QV_TamanhoTabela
order by convert(int, replace(Tamanho_KB, ' KB','')) desc;
```

12.2. A Função QV_LimitaTamanhoTexto

```
/*============================================================================
Procedimento:     QV_LimitaTamanhoTexto
Autor:            Henrique Figueiredo de Souza
Data Criação:     08/02/2013
Parametros:
            p_str       Texto a ser limitado.
            p_limite    Limite de palavras por linhas.
Descrição:
            Limita o texto fornecido a uma determinada quantidades
            de palavras por linha.
============================================================================*/

CREATE FUNCTION QV_LimitaTamanhoTexto (@p_str VARCHAR(8000),
@p_limite integer)
RETURNS VARCHAR(8000)
AS
BEGIN
      DECLARE     @i INTEGER = 1;
      DECLARE     @v_palavras INTEGER = 0;
      DECLARE     @v_tamanho INTEGER = ISNULL(LEN(@p_str),0);
      DECLARE     @v_dentro_uma_palavra BIT;
      DECLARE     @v_texto VARCHAR(8000);

      SET @v_texto = '';
      WHILE @i <= (@v_tamanho + 1)
      BEGIN
            SET @v_texto = @v_texto + SUBSTRING(@p_str, @i, 1);

            IF (ASCII(SUBSTRING(@p_str, @i, 1)) < 33)
        OR (@i > @v_tamanho)
            BEGIN
                  IF @v_dentro_uma_palavra=1
                  BEGIN
                        SET @v_palavras = @v_palavras + 1;
                        SET @v_dentro_uma_palavra = 0;

                        if @v_palavras=@p_limite
                        BEGIN
                              SET @v_texto = @v_texto + CHAR(13) + CHAR(10);
                              SET @v_palavras = 0;
                        END
                  END
            END
            ELSE
            BEGIN
                  SET @v_dentro_uma_palavra = 1;
            END

            SET @i = @i + 1;
      END
```

```
      RETURN @v_texto;
END
GO
```

12.3. A Função QV_AplicarMapa

```
/*=============================================================================
Procedimento:     QV_AplicarMapa
Autor:            Henrique Figueiredo de Souza
Data Criação:     09/02/2013
Parametros:
            p_tabela    Nome da tabela referenciada pela FK.
            p_coluna    Nome da coluna que possui uma FK.
Descrição:
            Aplicar o mapa carregado, onde a coluna é uma
            Chave Estrangeira e tabela é a tabela referenciada pela
            chave estrangeira da coluna.
=============================================================================*/

CREATE FUNCTION QV_AplicarMapa(@p_tabela VARCHAR(128),
@p_coluna VARCHAR(128))
RETURNS VARCHAR(8000)
AS
BEGIN
      DECLARE @v_mapa VARCHAR(8000);
      DECLARE @v_tabela VARCHAR(128);
      DECLARE @v_tabela_mapa VARCHAR(128);
      DECLARE @v_nome_tabela VARCHAR(128);

      DECLARE c_mapa cursor LOCAL FAST_FORWARD READ_ONLY FOR
            SELECT (SELECT distinct tc1.TABLE_NAME
    FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
                        WHERE tc1.CONSTRAINT_NAME=(
        SELECT rc.UNIQUE_CONSTRAINT_NAME
        FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
                        WHERE rc.CONSTRAINT_NAME=ccu.CONSTRAINT_NAME))
R_TABLE_NAME
            FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc,
    INFORMATION_SCHEMA.KEY_COLUMN_USAGE ccu
            WHERE tc.TABLE_NAME=ccu.TABLE_NAME
            AND tc.CONSTRAINT_NAME=ccu.CONSTRAINT_NAME
            AND tc.CONSTRAINT_TYPE='FOREIGN KEY'
            AND tc.TABLE_NAME=@p_tabela
            AND ccu.COLUMN_NAME=@p_coluna;

      SET @v_mapa = '';

      -- verifica se a coluna da tabela é uma Chave Estrangeira
      OPEN c_mapa;
      FETCH NEXT FROM c_mapa INTO @v_tabela_mapa

      IF (@@FETCH_STATUS = 0)
      BEGIN
            DECLARE c_tabela cursor LOCAL FAST_FORWARD READ_ONLY FOR
                  SELECT t.name
                  from (SELECT t1.name, t1.create_date FROM sys.tables t1
                  WHERE t1.name in (SELECT DISTINCT tc1.TABLE_NAME
      FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
                        WHERE tc1.CONSTRAINT_TYPE IN ('PRIMARY KEY','FOREIGN
KEY'))) t
                  WHERE t.name not in (SELECT DISTINCT tc1.TABLE_NAME
      FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
```

```
                        WHERE tc1.CONSTRAINT_TYPE='FOREIGN KEY')
                  AND t.name=@v_tabela_mapa;

            OPEN c_tabela;
            FETCH NEXT FROM c_tabela INTO @v_tabela

            -- verifica se tabela possui somente Chave Primaria
            IF (@@FETCH_STATUS = 0)
            BEGIN
                  CLOSE c_tabela;
                  DEALLOCATE c_tabela;

                  SET @v_mapa = 'ApplyMap(' + char(39) + 'Mapa' +
        @v_tabela_mapa + char(39) + ',' + @p_coluna + ')';
            END
            ELSE
            BEGIN
                  CLOSE c_tabela;
                  DEALLOCATE c_tabela;
            END;
      END;

      CLOSE c_mapa;
      DEALLOCATE c_mapa;

      RETURN @v_mapa;
END
GO
```

12.4. O Procedimento QV_MontarLoad

```
/*==============================================================================
Procedimento:     QV_MontarLoad
Autor:            Henrique Figueiredo de Souza
Data Criação:     09/02/2013
Parametros:
            p_tabela    Nome da tabela a gerar o LOAD.
            p_autor           Nome do autor a ser mostrado no comentário.
            p_mapeada   TRUE para mapping load, FALSE para sem mapping load
            p_2_colunas Limita a saida a somente a duas colunas.
Descrição:
            Monta um LOAD em formato qlikview de uma tabela
==============================================================================*/

CREATE PROCEDURE QV_MontarLoad(@p_tabela varchar(128),
@p_autor varchar(80), @p_mapeada BIT, @p_2_colunas BIT)
AS
BEGIN
      DECLARE @v_tabela VARCHAR(128);
      DECLARE @v_coluna VARCHAR(8000);
      DECLARE @v_comenta VARCHAR(8000);
      DECLARE @v_rotulo VARCHAR(4000);
      DECLARE @v_rotulo_tabela VARCHAR(4000);
      DECLARE @v_mapa VARCHAR(4000);
      DECLARE @v_nomecoluna VARCHAR(4000);
      DECLARE @v_cr CHAR(2);
      DECLARE @v_conta INTEGER;

      DECLARE @v_nome_tabela VARCHAR(128);
      DECLARE @v_nome_coluna VARCHAR(128);
      DECLARE @v_ordem_coluna INTEGER;
      DECLARE @v_comentario VARCHAR(8000);
```

```
  DECLARE @v_comentario_tabela VARCHAR(8000);

  DECLARE c_coluna cursor LOCAL FAST_FORWARD READ_ONLY FOR
        SELECT c.TABLE_NAME, c.COLUMN_NAME, c.ORDINAL_POSITION,
        (SELECT CONVERT(VARCHAR(8000),value)
              FROM fn_listextendedproperty('COMENTARIO', 'schema',
  'dbo', 'table', c.TABLE_NAME, 'column',
  c.COLUMN_NAME)) as COLUMN_COMMENT,
        (SELECT CONVERT(VARCHAR(8000),value)
              FROM fn_listextendedproperty('COMENTARIO', 'schema',
  'dbo', 'table', c.TABLE_NAME, default,
  default)) as TABLE_COMMENT
        FROM INFORMATION_SCHEMA.COLUMNS c
        WHERE c.TABLE_NAME=@p_tabela
        ORDER BY c.ORDINAL_POSITION

  SET @v_cr = char(13) + char(10);
  SET @v_coluna = '';
  SET @v_conta = 0;

  OPEN c_coluna;
  FETCH NEXT FROM c_coluna INTO @v_nome_tabela,
@v_nome_coluna, @v_ordem_coluna, @v_comentario,
@v_comentario_tabela;

  WHILE (@@FETCH_STATUS = 0)
  BEGIN
        SET @v_tabela = @v_nome_tabela;
        SET @v_rotulo = @v_comentario;
        SET @v_rotulo_tabela = dbo.QV_LimitaTamanhoTexto(
  @v_comentario_tabela, 10);
        SET @v_mapa = dbo.QV_AplicarMapa(@v_tabela, @v_nome_coluna);

        if len(@v_mapa) > 0
        begin
              SET @v_nomecoluna = @v_mapa;
        end
        else
        begin
              SET @v_nomecoluna = @v_nome_coluna;
        end;

        if @v_rotulo is null
        begin
              SET @v_coluna = @v_coluna + ', ' + @v_cr + '     ' +
    @v_nomecoluna;
        end
        else
        begin
              SET @v_coluna = @v_coluna + ', ' + @v_cr + '     ' +
    @v_nomecoluna + ' as [' + @v_rotulo + ']';
        end;

        if @p_2_colunas=1
        begin
              SET @v_conta = @v_conta + 1;

              if @v_conta=2
              begin
                    BREAK;
              end;
        end;

        FETCH NEXT FROM c_coluna INTO @v_nome_tabela,
```

```
      @v_nome_coluna, @v_ordem_coluna, @v_comentario,
      @v_comentario_tabela;
      END

      CLOSE c_coluna;
      DEALLOCATE c_coluna;

      if len(@v_coluna) > 0
      begin

            SET @v_coluna = substring(@v_coluna, 3, len(@v_coluna));

            SET @v_comenta = @v_cr +

'/*============================================================================
' + @v_cr;

            if @p_mapeada=1
            begin
                  SET @v_comenta = @v_comenta + 'Procedimento:' + char(9) +
      'Carga mapeada em memória do arquivo QVD' + @v_cr;
            end
            else
            begin
                  SET @v_comenta = @v_comenta + 'Procedimento:' + char(9) +
      'Carga em memória do arquivo QVD' + @v_cr;
            end;
            SET @v_comenta = @v_comenta + 'ArquivoQVD:' + char(9) +
    char(9) + @v_tabela + @v_cr;
            SET @v_comenta = @v_comenta + 'Autor:' + char(9) +
    char(9) + char(9) + @p_autor + @v_cr;
            SET @v_comenta = @v_comenta + 'Data Criação:' + char(9)
    + convert(CHAR, GETDATE(), 103) + @v_cr;

            if @v_rotulo_tabela is not null
            begin
                  if  len(@v_rotulo_tabela) > 0
                  begin
                        SET @v_comenta = @v_comenta + 'Descrição:' + char(9)
        + @v_rotulo_tabela + @v_cr;
                  end;
            end;

            PRINT @v_comenta + 'Restrições:';

            EXECUTE dbo.QV_ListarTabelaRestricao @p_tabela;

            PRINT
'============================================================================*/
' + @v_cr;

            if @p_mapeada=1
            begin
                  PRINT 'Mapa' + @v_tabela + ':' + @v_cr +
      'Mapping LOAD' + @v_coluna + @v_cr + 'FROM' + @v_cr +
                        '$(vDiretorio)' + @v_tabela + '.qvd' +
        @v_cr + '(qvd);' + @v_cr;
            end
            else
            begin
                  PRINT @v_tabela + ':' + @v_cr +
      'LOAD' + @v_coluna + @v_cr +
      'FROM' + @v_cr + '$(vDiretorio)' + @v_tabela +
      '.qvd' + @v_cr + '(qvd);' + @v_cr;
```

```
            end;

      end;

END
GO
```

12.5. O Procedimento QV_ContarLinhasTabela

```
/*=====================================================================================
Procedimento:     QV_ContarLinhasTabela
Autor:            Henrique Figueiredo de Souza
Data Criação:     09/02/2013
Parametros:
            p_tabela    Nome da tabela referenciada pela FK.
Descrição:
            Retorna a quantidade de linhas de uma tabela.
=====================================================================================*/

CREATE PROCEDURE QV_ContarLinhasTabela(@p_tabela VARCHAR(128))
AS
BEGIN
      DECLARE @v_conta BIGINT;
      DECLARE @sqlStatement NVARCHAR(200);
      DECLARE @tmpContaTabela table (linhas bigint);

      SET @sqlStatement = 'SELECT COUNT(*) FROM ' + @p_tabela;

      insert into @tmpContaTabela exec (@sqlStatement);

      select @v_conta=linhas from @tmpContaTabela

      PRINT @v_conta;
END
GO
```

12.6. O Procedimento QV_MontarTodosLoad

```
/*=====================================================================================
Procedimento:     QV_MontarTodosLoad
Autor:            Henrique Figueiredo de Souza
Data Criação:     09/02/2013
Parametros:
            @p_autor          Nome do autor a ser mostrado no comentário.
            @p_vazia          Se 1 verificar se a tabela está vazia.
Descrição:
            Monta todos os LOAD em formato qlikview de todas as tabelas
=====================================================================================*/

CREATE PROCEDURE QV_MontarTodosLoad(@p_autor varchar(80),
@p_vazia BIT)
AS
BEGIN
      DECLARE @v_tabela VARCHAR(128);
      DECLARE @v_conta BIGINT;
      DECLARE @sqlStatement NVARCHAR(200);
      DECLARE @tmpContaTabela table (linhas bigint);
      DECLARE c_tabela cursor LOCAL FAST_FORWARD READ_ONLY FOR
            SELECT t.TABLE_NAME FROM INFORMATION_SCHEMA.TABLES t
            WHERE t.TABLE_TYPE='BASE TABLE'
    AND t.TABLE_NAME<>'QV_TabelasTamanho'
```

```
            ORDER BY t.TABLE_NAME;

      OPEN c_tabela;
      FETCH NEXT FROM c_tabela INTO @v_tabela;

      WHILE (@@FETCH_STATUS = 0)
      BEGIN
            if @p_vazia = 1
            begin
                  SET @sqlStatement = 'SELECT COUNT(*) FROM ' + @v_tabela;
                  insert into @tmpContaTabela exec (@sqlStatement);
                  select @v_conta=linhas from @tmpContaTabela

                  if @v_conta > 0
                  begin
                        EXECUTE dbo.QV_MontarLoad @v_tabela, @p_autor, 0, 0
                  end;
            end
            else
            begin
                  EXECUTE dbo.QV_MontarLoad @v_tabela, @p_autor, 0, 0
            end;

            FETCH NEXT FROM c_tabela INTO @v_tabela;
      END;

      CLOSE c_tabela;
      DEALLOCATE c_tabela;

END
GO
```

12.7. O Procedimento QV_MontarTodosMappingLoad

```
/*=============================================================================
Procedimento:     QV_MontarTodosMappingLoad
Autor:            Henrique Figueiredo de Souza
Data Criação:     09/02/2013
Parametros:
            p_autor           Nome do autor a ser mostrado no comentário.
            p_vazia           Se 1 verificar se a tabela está vazia.
            p_2_colunas Limita a saida a somente a duas colunas.
Descrição:
            Monta todos os Mapping LOAD em formato qlikview
            de todas as tabelas que possuem SOMENTE Chave Primária.
=============================================================================*/

CREATE PROCEDURE QV_MontarTodosMappingLoad(@p_autor varchar(80), @p_vazia BIT,
@p_2_colunas BIT)
AS
BEGIN
      DECLARE @v_tabela VARCHAR(128);
      DECLARE @v_conta BIGINT;
      DECLARE @sqlStatement NVARCHAR(200);
      DECLARE @tmpContaTabela table (linhas bigint);
      DECLARE c_tabela cursor LOCAL FAST_FORWARD READ_ONLY FOR
            SELECT t.name from (
    SELECT t1.name, t1.create_date FROM sys.tables t1
            WHERE t1.name in (SELECT DISTINCT tc1.TABLE_NAME
    FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
    WHERE tc1.CONSTRAINT_TYPE IN ('PRIMARY KEY','FOREIGN KEY'))) t
            WHERE t.name not in (SELECT DISTINCT tc1.TABLE_NAME
```

```
    FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
    WHERE tc1.CONSTRAINT_TYPE='FOREIGN KEY')
            AND t.name<>'QV_TabelasTamanho'
            ORDER BY convert(datetime,t.create_date,126) ASC;

      OPEN c_tabela;
      FETCH NEXT FROM c_tabela INTO @v_tabela;

      WHILE (@@FETCH_STATUS = 0)
      BEGIN
            if @p_vazia = 1
            begin
                  SET @sqlStatement = 'SELECT COUNT(*) FROM ' + @v_tabela;
                  insert into @tmpContaTabela exec (@sqlStatement);
                  select @v_conta=linhas from @tmpContaTabela

                  if @v_conta > 0
                  begin
                        EXECUTE dbo.QV_MontarLoad @v_tabela, @p_autor, 1,
@p_2_colunas
                  end;
            end
            else
            begin
                  EXECUTE dbo.QV_MontarLoad @v_tabela, @p_autor, 1, @p_2_colunas
            end;

            FETCH NEXT FROM c_tabela INTO @v_tabela;
      END;

      CLOSE c_tabela;
      DEALLOCATE c_tabela;

END
GO
```

12.8. A Função QV_ColunasDaRestricaoDaTabela

```
/*=============================================================================
Procedimento:     QV_ColunasDaRestricaoDaTabela
Autor:            Henrique Figueiredo de Souza
Data Criação:     08/02/2013
Parametros:
            p_tabela    Nome da tabela referenciada pela FK.
            p_restricao Nome da restrição.
Descrição:
            Retorna as colunas de uma tabela e sua restrição.
=============================================================================*/

CREATE FUNCTION QV_ColunasDaRestricaoDaTabela(
@p_tabela varchar(128), @p_restricao varchar(128))
RETURNS VARCHAR(8000)
AS
BEGIN
      DECLARE     @v_coluna VARCHAR(8000);
      DECLARE @v_nome_coluna varchar(128);
      DECLARE c_tabela cursor LOCAL FAST_FORWARD READ_ONLY FOR
            SELECT ccu.COLUMN_NAME
    FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE ccu
            where ccu.TABLE_NAME=@p_tabela
    and ccu.CONSTRAINT_NAME=@p_restricao;
```

```
	SET @v_coluna = '';

	OPEN c_tabela;
	FETCH NEXT FROM c_tabela INTO @v_nome_coluna;

	WHILE (@@FETCH_STATUS = 0)
	BEGIN
		SET @v_coluna = @v_coluna + ', ' + @v_nome_coluna;

		FETCH NEXT FROM c_tabela INTO @v_nome_coluna;
	END

	CLOSE c_tabela;
	DEALLOCATE c_tabela;

	set @v_coluna = substring(@v_coluna, 3, len(@v_coluna));

	RETURN @v_coluna;
END
GO
```

12.9. O Procedimento QV_ListarTabelasColunas

```
/*================================================================================
Procedimento:	QV_ListarTabelasColunas
Autor:		Henrique Figueiredo de Souza
Data Criação:	09/02/2013
Parametros:
		p_comentario	Se 1 incluir comando para gerar comentario.
Descrição:
		Lista todas a tabelas e colunas do banco de dados
================================================================================*/

CREATE PROCEDURE QV_ListarTabelasColunas(@p_comentario BIT)
AS
BEGIN
	DECLARE @v_nome_tabela varchar(128);
	DECLARE @v_nome_coluna varchar(128);
	DECLARE @v_tabela_anterior VARCHAR(61);
	DECLARE @v_coluna VARCHAR(8000);
	DECLARE @v_cr CHAR(2);
	DECLARE @v_conta integer;
	DECLARE c_tabela cursor LOCAL FAST_FORWARD READ_ONLY FOR
		SELECT c.TABLE_NAME, c.COLUMN_NAME
    FROM INFORMATION_SCHEMA.COLUMNS c
		ORDER BY c.TABLE_NAME, c.ORDINAL_POSITION

	SET @v_cr = char(13) + char(10);
	SET @v_tabela_anterior = '';
	SET @v_conta = 0;

	OPEN c_tabela;
	FETCH NEXT FROM c_tabela INTO @v_nome_tabela,
    @v_nome_coluna;

	WHILE (@@FETCH_STATUS = 0)
	BEGIN
		if @v_tabela_anterior<>@v_nome_tabela
		begin
			PRINT @v_coluna;
			SET @v_coluna = '';
			SET @v_conta = 0;
```

```
            end;

            if @p_comentario=1
            begin
                  if @v_conta=0
                  begin
                        SET @v_coluna = 'EXEC sys.sp_addextendedproperty @name='
+
        char(39) + 'COMENTARIO' + char(39) + ', @value=' +
        char(39) + char(9) + char(9) + char(39) +
        ', @level0type=' + char(39) + 'SCHEMA' + char(39) +
        ', @level0name=' + char(39) + 'dbo' + char(39) +
                        ', @level1type=' + char(39) + 'TABLE' + char(39) +
        ', @level1name=' + char(39) + @v_nome_tabela +
        char(39) + @v_cr + @v_coluna;

                        SET @v_conta = 1;
                  end;

                  SET @v_coluna = @v_coluna +
      'EXEC sys.sp_addextendedproperty @name=' + char(39) +
      'COMENTARIO' + char(39) + ', @value=' + char(39) +
                  char(9) + char(9) + char(39) + ', @level0type=' +
      char(39) + 'SCHEMA' + char(39) + ', @level0name=' +
      char(39) + 'dbo' + char(39) + ', @level1type=' + char(39) +
      'TABLE' + char(39) + ', @level1name=' + char(39) +
      @v_nome_tabela + char(39) + ', @level2type=' +
                  char(39) + 'COLUMN' + char(39) + ', @level2name='  +
      char(39) + @v_nome_coluna + char(39) + @v_cr;
            end
            else
            begin
                  SET @v_coluna = @v_coluna + @v_nome_tabela + '.' +
        @v_nome_coluna + @v_cr;
            end;

            SET @v_tabela_anterior = @v_nome_tabela;

            FETCH NEXT FROM c_tabela INTO @v_nome_tabela,
      @v_nome_coluna;
      END

      CLOSE c_tabela;
      DEALLOCATE c_tabela;

      PRINT @v_coluna;

END
GO
```

12.10. O Procedimento QV_ListarTabelasRestricoes

```
/*==============================================================================
Procedimento:     QV_ListarTabelasRestricoes
Autor:            Henrique Figueiredo de Souza
Data Criação:     08/02/2013
Descrição:
            Lista todas as tabelas e suas restrições.
==============================================================================*/

CREATE PROCEDURE QV_ListarTabelasRestricoes
AS
```

```
BEGIN
      DECLARE @v_tabela_anterior VARCHAR(128);
      DECLARE @v_tabela VARCHAR(8000);
      DECLARE @v_nome_tabela VARCHAR(128);
      DECLARE @v_nome_restricao VARCHAR(128);
      DECLARE @v_tipo_restricao VARCHAR(128);
      DECLARE @v_restricao_ref VARCHAR(128);
      DECLARE @v_tabela_ref VARCHAR(128);
      DECLARE @v_restricao VARCHAR(8000);
      DECLARE @v_cr CHAR(2);

      SET @v_cr = char(13) + char(10);
      SET @v_tabela_anterior = '';
      SET @v_restricao = '';

      DECLARE c_tabela CURSOR LOCAL FAST_FORWARD READ_ONLY FOR
            SELECT tc.TABLE_NAME, tc.CONSTRAINT_NAME, tc.CONSTRAINT_TYPE,
            (SELECT rc.UNIQUE_CONSTRAINT_NAME
    FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
    WHERE rc.CONSTRAINT_NAME=tc.CONSTRAINT_NAME) R_CONSTRAINT_NAME,
            (SELECT distinct tc1.TABLE_NAME
    FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
                  WHERE tc1.CONSTRAINT_NAME=(SELECT rc.UNIQUE_CONSTRAINT_NAME
      FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
                        WHERE rc.CONSTRAINT_NAME=tc.CONSTRAINT_NAME))
R_TABLE_NAME
            FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc
            WHERE tc.CONSTRAINT_TYPE in ('PRIMARY KEY','FOREIGN KEY')
            ORDER BY tc.TABLE_NAME, tc.CONSTRAINT_TYPE;

      OPEN c_tabela;
      FETCH NEXT FROM c_tabela INTO @v_nome_tabela,
    @v_nome_restricao, @v_tipo_restricao, @v_restricao_ref,
    @v_tabela_ref

      WHILE (@@FETCH_STATUS = 0)
      BEGIN
            if @v_tabela_anterior<>@v_nome_tabela
            begin
                  PRINT @v_restricao;
                  SET @v_restricao = '';
            end;

            if @v_tipo_restricao='PRIMARY KEY'
            begin
                  set @v_restricao = @v_restricao + @v_nome_restricao +
      ' = ' + @v_nome_tabela + '(' +
                  dbo.QV_ColunasDaRestricaoDaTabela(@v_nome_tabela,
      @v_nome_restricao) + ')' + @v_cr;
            end
            else
            begin
                  set @v_restricao = @v_restricao + @v_nome_restricao +
      ' = ' + @v_nome_tabela + '(' +
                  dbo.QV_ColunasDaRestricaoDaTabela(@v_nome_tabela,
      @v_nome_restricao) + ') --> ' + @v_tabela_ref + '(' +
                  dbo.QV_ColunasDaRestricaoDaTabela(@v_tabela_ref,
      @v_restricao_ref) + ')' + @v_cr;
            end;

            FETCH NEXT FROM c_tabela INTO @v_nome_tabela,
    @v_nome_restricao, @v_tipo_restricao, @v_restricao_ref,
    @v_tabela_ref
      END
```

```
	CLOSE c_tabela;
	DEALLOCATE c_tabela;

	PRINT @v_restricao;

END
GO
```

12.11. O Procedimento QV_ListarTabelaRestricao

```
/*================================================================================
Procedimento:	QV_ListarTabelaRestricao
Autor:		Henrique Figueiredo de Souza
Data Criação:	08/02/2013
Parametros:
		p_tabela	Nome da tabela.
Descrição:
		Lista restrições de uma tabela.
================================================================================*/

CREATE PROCEDURE QV_ListarTabelaRestricao(@p_tabela varchar(128))
AS
BEGIN
	DECLARE @v_tabela_anterior VARCHAR(128);
	DECLARE @v_tabela VARCHAR(8000);
	DECLARE @v_nome_tabela VARCHAR(128);
	DECLARE @v_nome_restricao VARCHAR(128);
	DECLARE @v_tipo_restricao VARCHAR(128);
	DECLARE @v_restricao_ref VARCHAR(128);
	DECLARE @v_tabela_ref VARCHAR(128);
	DECLARE @v_restricao VARCHAR(8000);
	DECLARE @v_cr CHAR(2);
	DECLARE @v_tam INTEGER;

	SET @v_cr = char(13) + char(10);
	SET @v_tabela_anterior = '';
	SET @v_restricao = '';

	DECLARE c_tabela CURSOR LOCAL FAST_FORWARD READ_ONLY FOR
		SELECT tc.TABLE_NAME, tc.CONSTRAINT_NAME, tc.CONSTRAINT_TYPE,
		(SELECT rc.UNIQUE_CONSTRAINT_NAME
		FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
		WHERE rc.CONSTRAINT_NAME=tc.CONSTRAINT_NAME) R_CONSTRAINT_NAME,
		(SELECT distinct tc1.TABLE_NAME
			FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
			WHERE tc1.CONSTRAINT_NAME=(SELECT rc.UNIQUE_CONSTRAINT_NAME
			FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
				WHERE rc.CONSTRAINT_NAME=tc.CONSTRAINT_NAME))
R_TABLE_NAME
		FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc
		WHERE tc.CONSTRAINT_TYPE in ('PRIMARY KEY','FOREIGN KEY')
		AND tc.TABLE_NAME=@p_tabela
		ORDER BY tc.TABLE_NAME, tc.CONSTRAINT_TYPE;

	OPEN c_tabela;
	FETCH NEXT FROM c_tabela INTO @v_nome_tabela, @v_nome_restricao,
	@v_tipo_restricao, @v_restricao_ref, @v_tabela_ref

	WHILE (@@FETCH_STATUS = 0)
	BEGIN
		if @v_tabela_anterior<>@v_nome_tabela
```

```
            begin
                  PRINT @v_restricao;
                  SET @v_restricao = '';
            end;

            if @v_tipo_restricao='PRIMARY KEY'
            begin
                  set @v_restricao = @v_restricao + @v_nome_tabela + '(' +
                  dbo.QV_ColunasDaRestricaoDaTabela(@v_nome_tabela,
                  @v_nome_restricao) + ')' + @v_cr;
            end
            else
            begin
                  set @v_restricao = @v_restricao + @v_nome_tabela + '(' +
                  dbo.QV_ColunasDaRestricaoDaTabela(@v_nome_tabela,
                  @v_nome_restricao) + ') --> ' + @v_tabela_ref + '(' +
                  dbo.QV_ColunasDaRestricaoDaTabela(@v_tabela_ref,
                  @v_restricao_ref) + ')' + @v_cr;
            end;

            FETCH NEXT FROM c_tabela INTO @v_nome_tabela, @v_nome_restricao,
            @v_tipo_restricao, @v_restricao_ref, @v_tabela_ref
      END

      CLOSE c_tabela;
      DEALLOCATE c_tabela;

      set @v_tam = len(@v_restricao)-2;

      if @v_tam > 0
      begin
            set @v_restricao = substring(@v_restricao, 1, @v_tam);
            PRINT @v_restricao;
      end;

END
GO
```

12.12. O Procedimento QV_TabelasNaoPossuem_PK_nemFK

```
/*================================================================================
Procedimento:     QV_TabelasNaoPossuem_PK_nemFK
Autor:            Henrique Figueiredo de Souza
Data Criação:     08/02/2013
Descrição:
            Lista todas as tabelas que NÃO possuem Chave Primária e
            não possuem Chave Estrangeira, ordenados por data de criação.
================================================================================*/

CREATE PROCEDURE QV_TabelasNaoPossuem_PK_nemFK
AS
BEGIN
      DECLARE @v_nome_objeto VARCHAR(128);
      DECLARE @v_tabela VARCHAR(8000);
      DECLARE @v_cr CHAR(2);
      DECLARE c_objeto CURSOR LOCAL FAST_FORWARD READ_ONLY FOR
            SELECT t.name FROM sys.tables t
            WHERE t.name not in (SELECT DISTINCT tc.TABLE_NAME
    FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc
    WHERE tc.CONSTRAINT_TYPE IN ('PRIMARY KEY','FOREIGN KEY'))
            AND t.name<>'QV_TabelasTamanho'
            ORDER BY convert(datetime,t.create_date,126) ASC;
```

```
	set @v_cr = char(13) + char(10);
	set @v_tabela = '';

	OPEN c_objeto;
	FETCH NEXT FROM c_objeto INTO @v_nome_objeto;

	WHILE (@@FETCH_STATUS = 0)
	BEGIN
		set @v_tabela = @v_tabela + @v_nome_objeto + @v_cr;

		FETCH NEXT FROM c_objeto INTO @v_nome_objeto;
	END

	CLOSE c_objeto;
	DEALLOCATE c_objeto;

	PRINT @v_tabela;
END
GO
```

12.13. O Procedimento QV_TabelasSomente_PK_naoFK

```
/*================================================================================
Procedimento:	QV_TabelasSomente_PK_naoFK
Autor:		Henrique Figueiredo de Souza
Data Criação:	08/02/2013
Descrição:
		Lista todas as tabelas que possuem SOMENTE Chave Primária e
		não possuem Chave Estrangeira, ordenados por data de criação.
================================================================================*/

CREATE PROCEDURE QV_TabelasSomente_PK_naoFK
AS
BEGIN
	DECLARE @v_nome_objeto VARCHAR(128);
	DECLARE @v_nome_tabela VARCHAR(128);
	DECLARE @v_nome_restricao VARCHAR(128);
	DECLARE @v_tipo_restricao VARCHAR(128);
	DECLARE @v_restricao_ref VARCHAR(128);
	DECLARE @v_tabela_ref VARCHAR(128);
	DECLARE @v_restricao VARCHAR(8000);
	DECLARE @v_cr CHAR(2);
	DECLARE c_objeto CURSOR LOCAL FAST_FORWARD READ_ONLY FOR
		SELECT t.name from (SELECT t1.name, t1.create_date
    FROM sys.tables t1
		WHERE t1.name in (SELECT DISTINCT tc1.TABLE_NAME
    FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
    WHERE tc1.CONSTRAINT_TYPE IN ('PRIMARY KEY','FOREIGN KEY'))) t
		WHERE t.name not in (SELECT DISTINCT tc1.TABLE_NAME
    FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
    WHERE tc1.CONSTRAINT_TYPE='FOREIGN KEY')
		AND t.name<>'QV_TabelasTamanho'
		ORDER BY convert(datetime,t.create_date,126) ASC;

	set @v_cr = char(13) + char(10);

	OPEN c_objeto;
	FETCH NEXT FROM c_objeto INTO @v_nome_objeto;

	WHILE (@@FETCH_STATUS = 0)
	BEGIN
```

```
            SET @v_restricao = '';

            DECLARE c_tabela CURSOR LOCAL FAST_FORWARD READ_ONLY FOR
                  SELECT tc.TABLE_NAME, tc.CONSTRAINT_NAME, tc.CONSTRAINT_TYPE,
                  (SELECT rc.UNIQUE_CONSTRAINT_NAME
      FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
      WHERE rc.CONSTRAINT_NAME=tc.CONSTRAINT_NAME) R_CONSTRAINT_NAME,
                  (SELECT distinct tc1.TABLE_NAME
      FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
                        WHERE tc1.CONSTRAINT_NAME=(
        SELECT rc.UNIQUE_CONSTRAINT_NAME
        FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
                              WHERE rc.CONSTRAINT_NAME=tc.CONSTRAINT_NAME))
R_TABLE_NAME
                  FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc
                  WHERE tc.TABLE_NAME=@v_nome_objeto;

            OPEN c_tabela;
            FETCH NEXT FROM c_tabela INTO @v_nome_tabela,
    @v_nome_restricao, @v_tipo_restricao, @v_restricao_ref,
    @v_tabela_ref

            WHILE (@@FETCH_STATUS = 0)
            BEGIN
                  if @v_tipo_restricao='PRIMARY KEY'
                  begin
                        set @v_restricao = @v_restricao + @v_nome_restricao
        + ' = ' + @v_nome_tabela + '(' +
                        dbo.QV_ColunasDaRestricaoDaTabela(@v_nome_tabela,
        @v_nome_restricao) + ')' + @v_cr;
                  end;

                  FETCH NEXT FROM c_tabela INTO @v_nome_tabela,
      @v_nome_restricao, @v_tipo_restricao,
      @v_restricao_ref, @v_tabela_ref
            END

            CLOSE c_tabela;
            DEALLOCATE c_tabela;


            PRINT @v_restricao;

            FETCH NEXT FROM c_objeto INTO @v_nome_objeto;
      END

      CLOSE c_objeto;
      DEALLOCATE c_objeto;

END
GO
```

12.14. O Procedimento QV_TabelasPossuem_FK

```
/*================================================================================
Procedimento:     QV_TabelasPossuem_FK
Autor:            Henrique Figueiredo de Souza
Data Criação:     08/02/2013
Descrição:
            Lista todas as tabelas que possuem Chave Estrangeira e podem
            ou não possuir Chave Primária, ordenados por data de criação.
================================================================================*/
```

```
CREATE PROCEDURE QV_TabelasPossuem_FK
AS
BEGIN
      DECLARE @v_nome_objeto VARCHAR(128);
      DECLARE @v_nome_tabela VARCHAR(128);
      DECLARE @v_nome_restricao VARCHAR(128);
      DECLARE @v_tipo_restricao VARCHAR(128);
      DECLARE @v_restricao_ref VARCHAR(128);
      DECLARE @v_tabela_ref VARCHAR(128);
      DECLARE @v_restricao VARCHAR(8000);
      DECLARE @v_cr CHAR(2);
      DECLARE c_objeto CURSOR LOCAL FAST_FORWARD READ_ONLY FOR
            SELECT t.name FROM sys.tables t
            WHERE t.name in (SELECT DISTINCT tc.TABLE_NAME
    FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc
    WHERE tc.CONSTRAINT_TYPE='FOREIGN KEY')
            AND t.name<>'QV_TabelasTamanho'
            ORDER BY convert(datetime,t.create_date,126) ASC;

      set @v_cr = char(13) + char(10);

      OPEN c_objeto;
      FETCH NEXT FROM c_objeto INTO @v_nome_objeto;

      WHILE (@@FETCH_STATUS = 0)
      BEGIN
            SET @v_restricao = '';

            DECLARE c_tabela CURSOR LOCAL FAST_FORWARD READ_ONLY FOR
                  SELECT tc.TABLE_NAME, tc.CONSTRAINT_NAME, tc.CONSTRAINT_TYPE,
                  (SELECT rc.UNIQUE_CONSTRAINT_NAME
      FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
      WHERE rc.CONSTRAINT_NAME=tc.CONSTRAINT_NAME) R_CONSTRAINT_NAME,
                  (SELECT distinct tc1.TABLE_NAME
      FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
                        WHERE tc1.CONSTRAINT_NAME=(
        SELECT rc.UNIQUE_CONSTRAINT_NAME
        FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
                              WHERE rc.CONSTRAINT_NAME=tc.CONSTRAINT_NAME))
R_TABLE_NAME
                  FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc
                  WHERE tc.TABLE_NAME=@v_nome_objeto;

            OPEN c_tabela;
            FETCH NEXT FROM c_tabela INTO @v_nome_tabela,
    @v_nome_restricao, @v_tipo_restricao,
    @v_restricao_ref, @v_tabela_ref

            WHILE (@@FETCH_STATUS = 0)
            BEGIN
                  if @v_tipo_restricao='FOREIGN KEY'
                  begin
                        set @v_restricao = @v_restricao + @v_nome_restricao
        + ' = ' + @v_nome_tabela + '(' +
                        dbo.QV_ColunasDaRestricaoDaTabela(@v_nome_tabela,
        @v_nome_restricao) + ') --> ' + @v_tabela_ref + '(' +
                        dbo.QV_ColunasDaRestricaoDaTabela(@v_tabela_ref,
        @v_restricao_ref) + ')' + @v_cr;
                  end;

                  FETCH NEXT FROM c_tabela INTO @v_nome_tabela,
      @v_nome_restricao, @v_tipo_restricao,
      @v_restricao_ref, @v_tabela_ref
```

```
            END

            CLOSE c_tabela;
            DEALLOCATE c_tabela;


            PRINT @v_restricao;

            FETCH NEXT FROM c_objeto INTO @v_nome_objeto;
      END

      CLOSE c_objeto;
      DEALLOCATE c_objeto;

END
GO
```

12.15. O Procedimento QV_TabelasSomente_FK_naoPK

```
/*================================================================================
Procedimento:     QV_TabelasSomente_FK_naoPK
Autor:            Henrique Figueiredo de Souza
Data Criação:     08/02/2013
Descrição:
            Lista todas as tabelas que possuem SOMENTE Chave Estrangeira
            e não possuem Chave Primária, ordenados por data de criação.
================================================================================*/

CREATE PROCEDURE QV_TabelasSomente_FK_naoPK
AS
BEGIN
      DECLARE @v_nome_objeto VARCHAR(128);
      DECLARE @v_nome_tabela VARCHAR(128);
      DECLARE @v_nome_restricao VARCHAR(128);
      DECLARE @v_tipo_restricao VARCHAR(128);
      DECLARE @v_restricao_ref VARCHAR(128);
      DECLARE @v_tabela_ref VARCHAR(128);
      DECLARE @v_restricao VARCHAR(8000);
      DECLARE @v_cr CHAR(2);
      DECLARE c_objeto CURSOR LOCAL FAST_FORWARD READ_ONLY FOR
            SELECT t.name
            from (SELECT t1.name, t1.create_date FROM sys.tables t1
                  WHERE t1.name not in (SELECT DISTINCT tc1.TABLE_NAME
      FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
      WHERE tc1.CONSTRAINT_TYPE='PRIMARY KEY')) t
            WHERE t.name in (SELECT DISTINCT tc1.TABLE_NAME
    FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
    WHERE tc1.CONSTRAINT_TYPE='FOREIGN KEY')
            AND t.name<>'QV_TabelasTamanho'
            ORDER BY convert(datetime,t.create_date,126) ASC;

      set @v_cr = char(13) + char(10);

      OPEN c_objeto;
      FETCH NEXT FROM c_objeto INTO @v_nome_objeto;

      WHILE (@@FETCH_STATUS = 0)
      BEGIN
            SET @v_restricao = '';

            DECLARE c_tabela CURSOR LOCAL FAST_FORWARD READ_ONLY FOR
                  SELECT tc.TABLE_NAME, tc.CONSTRAINT_NAME, tc.CONSTRAINT_TYPE,
```

```
                  (SELECT rc.UNIQUE_CONSTRAINT_NAME
      FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
      WHERE rc.CONSTRAINT_NAME=tc.CONSTRAINT_NAME) R_CONSTRAINT_NAME,
                  (SELECT distinct tc1.TABLE_NAME
      FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
                        WHERE tc1.CONSTRAINT_NAME=(SELECT
rc.UNIQUE_CONSTRAINT_NAME
        FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
                              WHERE rc.CONSTRAINT_NAME=tc.CONSTRAINT_NAME))
R_TABLE_NAME
                  FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc
                  WHERE tc.TABLE_NAME=@v_nome_objeto;

            OPEN c_tabela;
            FETCH NEXT FROM c_tabela INTO @v_nome_tabela,
    @v_nome_restricao, @v_tipo_restricao,
    @v_restricao_ref, @v_tabela_ref

            WHILE (@@FETCH_STATUS = 0)
            BEGIN
                  if @v_tipo_restricao='FOREIGN KEY'
                  begin
                        set @v_restricao = @v_restricao + @v_nome_restricao
      + ' = ' + @v_nome_tabela + '(' +
                        dbo.QV_ColunasDaRestricaoDaTabela(@v_nome_tabela,
      @v_nome_restricao) + ') --> ' + @v_tabela_ref + '(' +
                        dbo.QV_ColunasDaRestricaoDaTabela(@v_tabela_ref,
      @v_restricao_ref) + ')' + @v_cr;
                  end;

                  FETCH NEXT FROM c_tabela INTO @v_nome_tabela,
      @v_nome_restricao, @v_tipo_restricao,
      @v_restricao_ref, @v_tabela_ref
            END

            CLOSE c_tabela;
            DEALLOCATE c_tabela;


            PRINT @v_restricao;

            FETCH NEXT FROM c_objeto INTO @v_nome_objeto;
      END

      CLOSE c_objeto;
      DEALLOCATE c_objeto;

END
GO
```

12.16. O Procedimento QV_DescomentarTabelasColunas

```
/*================================================================================
Procedimento:     QV_DescomentarTabelasColunas
Autor:            Henrique Figueiredo de Souza
Data Criação:     09/02/2013
Descrição:
            Descomentar todas a tabelas e colunas do banco de dados
================================================================================*/

CREATE PROCEDURE QV_DescomentarTabelasColunas
AS
```

```
BEGIN
      DECLARE @v_nome_tabela varchar(128);
      DECLARE @v_nome_coluna varchar(128);
      DECLARE @v_tabela_anterior VARCHAR(61);
      DECLARE @v_coluna VARCHAR(8000);
      DECLARE @v_cr CHAR(2);
      DECLARE @v_conta integer;
      DECLARE c_tabela cursor LOCAL FAST_FORWARD READ_ONLY FOR
            SELECT c.TABLE_NAME, c.COLUMN_NAME
    FROM INFORMATION_SCHEMA.COLUMNS c
            ORDER BY c.TABLE_NAME, c.ORDINAL_POSITION

      SET @v_cr = char(13) + char(10);
      SET @v_tabela_anterior = '';
      SET @v_conta = 0;

      OPEN c_tabela;
      FETCH NEXT FROM c_tabela INTO @v_nome_tabela,
    @v_nome_coluna;

      WHILE (@@FETCH_STATUS = 0)
      BEGIN
            if @v_tabela_anterior<>@v_nome_tabela
            begin
                  PRINT @v_coluna;
                  SET @v_coluna = '';
                  SET @v_conta = 0;
            end;

            if @v_conta=0
            begin
                  SET @v_coluna = 'EXEC sys.sp_dropextendedproperty @name='
      + char(39) + 'COMENTARIO' + char(39) +
                  ', @level0type=' + char(39) + 'SCHEMA' + char(39) +
      ', @level0name=' + char(39) + 'dbo' + char(39) +
                  ', @level1type=' + char(39) + 'TABLE' + char(39) +
      ', @level1name=' + char(39) + @v_nome_tabela
      + char(39) + @v_cr + @v_coluna;

                  SET @v_conta = 1;
            end;

            SET @v_coluna = @v_coluna +
    'EXEC sys.sp_dropextendedproperty @name=' + char(39) +
    'COMENTARIO' + char(39) +
            ', @level0type=' + char(39) + 'SCHEMA' + char(39) +
    ', @level0name=' + char(39) + 'dbo' + char(39) +
            ', @level1type=' + char(39) + 'TABLE' + char(39) +
    ', @level1name=' + char(39) + @v_nome_tabela + char(39) +
            ', @level2type=' + char(39) + 'COLUMN' + char(39) +
    ', @level2name='  + char(39) + @v_nome_coluna
    + char(39) + @v_cr;

            SET @v_tabela_anterior = @v_nome_tabela;

            FETCH NEXT FROM c_tabela INTO @v_nome_tabela,
      @v_nome_coluna;
      END

      CLOSE c_tabela;
      DEALLOCATE c_tabela;

      PRINT @v_coluna;
```

```
END
GO
```

13. As Funções Qlikview para o POSTGRESQL

```
SELECT QV_AplicarMapa('pergunta_duvida','codigo_duvida');
SELECT QV_MontarTodosLoad('Henrique Figueiredo de Souza', true);
SELECT QV_MontarTodosMappingLoad('Henrique Figueiredo de Souza', FALSE, TRUE);
SELECT QV_ListarTabelaRestricao('pergunta_duvida');
SELECT QV_ListarTabelasColunas(false);
SELECT QV_ListarTabelasRestricoes();
SELECT QV_TabelasNaoPossuem_PK_nemFK();
SELECT QV_TabelasSomente_PK_naoFK();
SELECT QV_TabelasPossuem_FK();
SELECT QV_TabelasSomente_FK_naoPK();
SELECT QV_DescomentarTabelasColunas();


DROP FUNCTION QV_LimitaTamanhoTexto(varchar, integer);
DROP FUNCTION QV_AplicarMapa(varchar, varchar);
DROP FUNCTION QV_ListarTabelaRestricao(varchar);
DROP FUNCTION QV_MontarLoad(VARCHAR, VARCHAR, boolean, boolean);
DROP FUNCTION QV_MontarTodosMappingLoad(VARCHAR, boolean, boolean);
DROP FUNCTION QV_MontarTodosLoad(VARCHAR, boolean);
DROP FUNCTION QV_ContarLinhasTabela(varchar);
DROP FUNCTION QV_ColunasDaRestricaoDaTabela(varchar, varchar);
DROP FUNCTION QV_ListarTabelasColunas(boolean);
DROP FUNCTION QV_ListarTabelasRestricoes();
DROP FUNCTION QV_TabelasNaoPossuem_PK_nemFK();
DROP FUNCTION QV_TabelasSomente_PK_naoFK();
DROP FUNCTION QV_TabelasPossuem_FK();
DROP FUNCTION QV_TabelasSomente_FK_naoPK();
DROP FUNCTION QV_DescomentarTabelasColunas();
```

13.1. A Função QV_LimitaTamanhoTexto

```
/*================================================================================
Procedimento:     QV_LimitaTamanhoTexto
Autor:            Henrique Figueiredo de Souza
Data Criação:     13/02/2013
Parametros:
            p_str       Texto a ser limitado.
            p_limite    Limite de palavras por linhas.
Descrição:
            Limita o texto fornecido a uma determinada quantidades
            de palavras por linha.
================================================================================*/

CREATE OR REPLACE FUNCTION QV_LimitaTamanhoTexto(p_str IN VARCHAR,
p_limite IN integer)
RETURNS VARCHAR AS
$BODY$
DECLARE
      v_palavras INTEGER := 0;
      v_tamanho INTEGER := coalesce(LENGTH(p_str),0);
      v_dentro_uma_palavra BOOLEAN;
      v_texto VARCHAR(32676);
BEGIN
      v_texto := '';
      FOR i IN 1..v_tamanho + 1
      LOOP
```

```
            v_texto := v_texto || SUBSTR(p_str, i, 1);

            IF ASCII(SUBSTR(p_str, i, 1)) < 33 OR i > v_tamanho THEN
                  IF v_dentro_uma_palavra THEN
                        v_palavras := v_palavras + 1;
                        v_dentro_uma_palavra := FALSE;

                        if v_palavras=p_limite then
                              v_texto := v_texto || chr(13) || chr(10);
                              v_palavras := 0;
                        end if;
                  END IF;
            ELSE
                  v_dentro_uma_palavra := TRUE;
            END IF;
      END LOOP;

      RETURN v_texto;

END;
$BODY$
LANGUAGE plpgsql;
```

13.2. A Função QV_AplicarMapa

```
/*=============================================================================
Procedimento:     QV_AplicarMapa
Autor:            Henrique Figueiredo de Souza
Data Criação:     13/02/2013
Parametros:
            p_tabela    Nome da tabela referenciada pela FK.
            p_coluna    Nome da coluna que possui uma FK.
Descrição:
            Aplicar o mapa carregado, onde a coluna é uma
            Chave Estrangeira e tabela é a tabela referenciada pela
            chave estrangeira da coluna.
=============================================================================*/

CREATE OR REPLACE FUNCTION QV_AplicarMapa(p_tabela IN varchar,
p_coluna IN varchar)
RETURNS VARCHAR AS
$BODY$
DECLARE
      v_mapa VARCHAR(8000);
      v_tabela VARCHAR(61);
      v_tabela_mapa VARCHAR(61);

      c_mapa CURSOR IS
            SELECT (SELECT distinct tc1.TABLE_NAME
                  FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
                  WHERE tc1.CONSTRAINT_NAME=(SELECT rc.UNIQUE_CONSTRAINT_NAME
                        FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
                  WHERE rc.CONSTRAINT_NAME=ccu.CONSTRAINT_NAME)) R_TABLE_NAME
            FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc,
            INFORMATION_SCHEMA.KEY_COLUMN_USAGE ccu
            WHERE tc.TABLE_NAME=ccu.TABLE_NAME
            AND tc.CONSTRAINT_SCHEMA='public'
            AND tc.CONSTRAINT_NAME=ccu.CONSTRAINT_NAME
            AND tc.CONSTRAINT_TYPE='FOREIGN KEY'
            AND tc.TABLE_NAME=p_tabela
            AND ccu.COLUMN_NAME=p_coluna;
```

```
	c_tabela CURSOR IS
		select t.table_name
		from (select t1.table_name
			from information_schema.tables t1
			where t1.table_schema='public' and t1.table_type='BASE TABLE'
			AND t1.table_name in (SELECT DISTINCT tc1.TABLE_NAME
			FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
			WHERE tc1.CONSTRAINT_SCHEMA='public'
			and tc1.CONSTRAINT_TYPE IN ('PRIMARY KEY','FOREIGN KEY'))) t
		WHERE t.table_name not in (SELECT DISTINCT tc1.TABLE_NAME
			FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
			WHERE tc1.CONSTRAINT_SCHEMA='public'
			and tc1.CONSTRAINT_TYPE='FOREIGN KEY')
		AND t.table_name=v_tabela_mapa;
BEGIN
	v_mapa := '';

	-- verifica se a coluna da tabela é uma Chave Estrangeira
	open c_mapa;
	fetch c_mapa into v_tabela_mapa;

	if (v_tabela_mapa is not null) and (length(v_tabela_mapa) > 0) then

		open c_tabela;
		fetch c_tabela into v_tabela;

		-- verifica se tabela possui somente Chave Primaria
		if (v_tabela is not null) and (length(v_tabela) > 0) then
			close c_tabela;

			v_mapa:= 'ApplyMap(' || chr(39) || 'Mapa' ||
			v_tabela_mapa || chr(39) || ',' || p_coluna || ')';
		else
			close c_tabela;
		end if;

		close c_mapa;
	else
		close c_mapa;
	end if;

	RETURN v_mapa;

END;
$BODY$
LANGUAGE plpgsql;
```

13.3. A Função QV_ColunasDaRestricaoDaTabela

```
/*=========================================================================
Procedimento:	QV_ColunasDaRestricaoDaTabela
Autor:		Henrique Figueiredo de Souza
Data Criação:	14/02/2013
Parametros:
		p_tabela    Nome da tabela referenciada pela FK.
		p_restricao Nome da restrição.
Descrição:
		Retorna as colunas de uma tabela e sua restrição.
=========================================================================*/

CREATE OR REPLACE FUNCTION QV_ColunasDaRestricaoDaTabela(p_tabela IN varchar,
p_restricao IN varchar)
```

```
RETURNS VARCHAR AS
$BODY$
DECLARE
	v_coluna VARCHAR(8000);
	c_tabela RECORD;
BEGIN
	v_coluna := '';

	FOR c_tabela IN (
		SELECT ccu.COLUMN_NAME FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE ccu
		where ccu.TABLE_NAME=p_tabela and ccu.CONSTRAINT_NAME=p_restricao)
	LOOP

		v_coluna := v_coluna || ', ' || c_tabela.column_name;

	END LOOP;

  v_coluna := substr(v_coluna, 3, length(v_coluna));

  RETURN v_coluna;

END;
$BODY$
LANGUAGE plpgsql;
```

13.4. A Função QV_ListarTabelaRestricao

```
/*================================================================================
Procedimento:	QV_ListarTabelaRestricao
Autor:		Henrique Figueiredo de Souza
Data Criação:	14/02/2013
Parametros:
		p_tabela	Nome da tabela.
Descrição:
		Lista restrições de uma tabela.
================================================================================*/

CREATE OR REPLACE FUNCTION QV_ListarTabelaRestricao(p_tabela IN varchar)
RETURNS VARCHAR AS
$BODY$
DECLARE
	v_tabela_anterior VARCHAR(61);
	v_tabela VARCHAR(8000);
	v_restricao VARCHAR(8000);
	v_cr CHAR(2);
	v_tam INTEGER;
	c_tabela RECORD;
	v_retorno VARCHAR(32676);
BEGIN
	v_cr := chr(13) || chr(10);
	v_tabela_anterior := '';
	v_restricao := '';
	v_retorno := '';

	FOR c_tabela IN (
		SELECT tc.TABLE_NAME, tc.CONSTRAINT_NAME, tc.CONSTRAINT_TYPE,
		(SELECT rc.UNIQUE_CONSTRAINT_NAME
		FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
		WHERE rc.CONSTRAINT_NAME=tc.CONSTRAINT_NAME) R_CONSTRAINT_NAME,
		(SELECT distinct tc1.TABLE_NAME
			FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
			WHERE tc1.CONSTRAINT_NAME=(SELECT rc.UNIQUE_CONSTRAINT_NAME
```

```
                  FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
                  WHERE rc.CONSTRAINT_NAME=tc.CONSTRAINT_NAME)) R_TABLE_NAME
            FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc
            WHERE tc.CONSTRAINT_TYPE in ('PRIMARY KEY','FOREIGN KEY')
            AND tc.CONSTRAINT_SCHEMA='public' AND tc.TABLE_NAME=p_tabela
            ORDER BY tc.TABLE_NAME, tc.CONSTRAINT_TYPE DESC)
      LOOP

            if v_tabela_anterior<>c_tabela.table_name then

                  if v_restricao is not null then
                        --RAISE NOTICE '%', v_restricao;
                        v_retorno := v_retorno || v_restricao;
                  end if;
                  v_restricao:='';
            end if;

            if c_tabela.CONSTRAINT_TYPE='PRIMARY KEY' then
                  v_restricao := v_restricao || c_tabela.table_name || '(' ||
                  QV_ColunasDaRestricaoDaTabela(c_tabela.table_name,
                  c_tabela.CONSTRAINT_NAME) || ')' || v_cr;
            else
                  v_restricao := v_restricao || c_tabela.table_name || '(' ||
                  QV_ColunasDaRestricaoDaTabela(c_tabela.table_name,
                  c_tabela.CONSTRAINT_NAME) || ') --> ' ||
                  c_tabela.R_TABLE_NAME || '(' ||
                  QV_ColunasDaRestricaoDaTabela(c_tabela.R_TABLE_NAME,
                  c_tabela.R_CONSTRAINT_NAME) || ')' || v_cr;
            end if;

            v_tabela_anterior := c_tabela.table_name;

      END LOOP;

      v_tam := length(v_restricao)-2;

      if v_tam > 0 then
            v_restricao := substr(v_restricao, 1, v_tam);

            --RAISE NOTICE '%', v_restricao;
            v_retorno := v_retorno || v_restricao;
      end if;

      RETURN      v_retorno;
END;
$BODY$
LANGUAGE plpgsql;
```

13.5. A Função QV_MontarLoad

```
/*================================================================================
Procedimento:     QV_MontarLoad
Autor:            Henrique Figueiredo de Souza
Data Criação:     14/02/2013
Parametros:
            p_tabela    Nome da tabela a gerar o LOAD.
            p_autor           Nome do autor a ser mostrado no comentário.
            p_mapeada   TRUE para mapping load, FALSE para sem mapping load
            p_2_colunas Limita a saida a somente a duas colunas.
Descrição:
            Monta um LOAD em formato qlikview de uma tabela
================================================================================*/
```

```
CREATE OR REPLACE FUNCTION QV_MontarLoad(p_tabela IN VARCHAR, p_autor IN
VARCHAR,
p_mapeada IN boolean, p_2_colunas IN boolean)
RETURNS VARCHAR AS
$BODY$
DECLARE
	v_tabela VARCHAR(61);
	v_coluna VARCHAR(32676);
	v_comenta VARCHAR(8000);
	v_rotulo VARCHAR(4000);
	v_rotulo_tabela VARCHAR(4000);
	v_mapa VARCHAR(4000);
	v_nomecoluna VARCHAR(4000);
	v_cr CHAR(2);
	v_conta INTEGER;
	c_coluna RECORD;
	v_retorno VARCHAR(32676);
BEGIN
	v_cr := chr(13) || chr(10);
	v_coluna := '';
	v_conta := 0;
	v_retorno := '';

	FOR c_coluna IN (
		SELECT CL.TABLE_NAME, CL.COLUMN_NAME, CL.ORDINAL_POSITION,
		(SELECT PG_CATALOG.COL_DESCRIPTION(OID,CL.ORDINAL_POSITION::INT)
		FROM PG_CATALOG.PG_CLASS C WHERE C.RELNAME=CL.TABLE_NAME) AS
COMMENTS,
		(SELECT PG_CATALOG.OBJ_DESCRIPTION(OID) FROM PG_CATALOG.PG_CLASS C
		WHERE C.RELNAME=CL.TABLE_NAME) AS TAB_COMMENTS
		FROM INFORMATION_SCHEMA.COLUMNS CL
		WHERE CL.TABLE_NAME=p_tabela AND CL.TABLE_SCHEMA='public'
		ORDER BY CL.ORDINAL_POSITION)
	LOOP
		v_tabela := c_coluna.table_name;
		v_rotulo := c_coluna.comments;
		v_rotulo_tabela := QV_LimitaTamanhoTexto(c_coluna.tab_comments, 10);
		v_mapa := QV_AplicarMapa(v_tabela, c_coluna.column_name);

		if length(v_mapa) > 0 then
			v_nomecoluna := v_mapa;
		else
			v_nomecoluna := c_coluna.column_name;
		end if;

		if v_rotulo is null then
			v_coluna := v_coluna || ', ' || v_cr
			|| '	' || v_nomecoluna;
		else
			v_coluna := v_coluna || ', ' || v_cr
			|| '	' || v_nomecoluna || ' as [' || v_rotulo || ']';
		end if;

		if p_2_colunas=TRUE then
			v_conta := v_conta + 1;

			if v_conta=2 then
				EXIT;
			end if;
		end if;

	END LOOP;
```

```
      if length(v_coluna) > 0 then

            v_coluna := substr(v_coluna, 3, length(v_coluna));

            v_comenta := v_cr ||

      '/*============================================================================
======' || v_cr;

            if p_mapeada=TRUE then
                  v_comenta := v_comenta || 'Procedimento:' || chr(9) ||
                  'Carga mapeada em memória do arquivo QVD' || v_cr;
            else
                  v_comenta := v_comenta || 'Procedimento:' || chr(9) ||
                  'Carga em memória do arquivo QVD' || v_cr;
            end if;
            v_comenta := v_comenta || 'ArquivoQVD:' || chr(9) || chr(9) ||
v_tabela || v_cr;
            v_comenta := v_comenta || 'Autor:' || chr(9) || chr(9) || chr(9) ||
p_autor || v_cr;
            v_comenta := v_comenta || 'Data Criação:' || chr(9) ||
                              to_char(current_timestamp, 'DD/MM/YYYY') || v_cr;

            if v_rotulo_tabela is not null then
                  if  length(v_rotulo_tabela) > 0 then
                        v_comenta := v_comenta || 'Descrição:' || chr(9) ||
                        v_rotulo_tabela || v_cr;
                  end if;
            end if;

            --RAISE NOTICE '%', v_comenta || 'Restrições:' || v_cr;
            v_retorno := v_retorno || v_comenta || 'Restrições:' || v_cr;

            v_retorno := v_retorno || QV_ListarTabelaRestricao(p_tabela) ||
v_cr;

            --RAISE NOTICE '%',
            v_retorno := v_retorno ||

      '============================================================================
====*/' || v_cr;

            if p_mapeada=TRUE then
                  --RAISE NOTICE '%',
                  v_retorno := v_retorno || 'Mapa' || v_tabela || ':' || v_cr ||
                  'Mapping LOAD' || v_coluna || v_cr || 'FROM' || v_cr ||
'$(vDiretorio)'
                  || v_tabela || '.qvd' || v_cr || '(qvd);' || v_cr;
            else
                  --RAISE NOTICE '%',
                  v_retorno := v_retorno || v_tabela || ':' || v_cr ||
                  'LOAD' || v_coluna || v_cr || 'FROM' || v_cr ||
'$(vDiretorio)'
                  || v_tabela || '.qvd' || v_cr || '(qvd);' || v_cr;
            end if;

      end if;

      RETURN      v_retorno;
END;
$BODY$
LANGUAGE plpgsql;
```

13.6. A Função QV_MontarTodosMappingLoad

```
/*================================================================================
Procedimento:     QV_MontarTodosMappingLoad
Autor:            Henrique Figueiredo de Souza
Data Criação:     14/02/2013
Parametros:
            p_autor           Nome do autor a ser mostrado no comentário.
            p_vazia           Se 1 verificar se a tabela está vazia.
            p_2_colunas Limita a saida a somente a duas colunas.
Descrição:
            Monta todos os Mapping LOAD em formato qlikview
            de todas as tabelas que possuem SOMENTE Chave Primária.
================================================================================*/

CREATE OR REPLACE FUNCTION QV_MontarTodosMappingLoad(p_autor IN varchar,
p_vazia IN boolean, p_2_colunas IN boolean)
RETURNS VARCHAR AS
$BODY$
DECLARE
      v_tabela VARCHAR(61);
      v_conta INTEGER;
      c_tabela RECORD;
      v_retorno VARCHAR(32676);
      v_sql VARCHAR(200);
BEGIN
      v_retorno := '';

      FOR c_tabela IN (
            select t.table_name
            from (select t1.table_name
                  from information_schema.tables t1
                  where t1.table_schema='public'
                  and t1.table_type='BASE TABLE'
                  AND t1.table_name in (SELECT DISTINCT tc1.TABLE_NAME
                  FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
                  WHERE tc1.table_schema='public'
                  and tc1.CONSTRAINT_TYPE IN ('PRIMARY KEY','FOREIGN KEY'))) t
            WHERE t.table_name not in (SELECT DISTINCT tc1.TABLE_NAME
            FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
            WHERE tc1.table_schema='public'
            and tc1.CONSTRAINT_TYPE='FOREIGN KEY'))
      LOOP
            v_tabela := c_tabela.table_name;

            if p_vazia = TRUE then
                  v_sql := 'SELECT COUNT(*) FROM ' || v_tabela;
                  EXECUTE v_sql INTO v_conta;

                  if v_conta > 0 then
                        v_retorno := v_retorno ||
                        QV_MontarLoad(v_tabela, p_autor, TRUE, p_2_colunas);
                  end if;
            else
                  v_retorno := v_retorno ||
                  QV_MontarLoad(v_tabela, p_autor, TRUE, p_2_colunas);
            end if;
      END LOOP;

      RETURN      v_retorno;
END;
$BODY$
LANGUAGE plpgsql;
```

13.7. A Função QV_MontarTodosLoad

```
/*=============================================================================
Procedimento:     QV_MontarTodosLoad
Autor:            Henrique Figueiredo de Souza
Data Criação:     14/02/2013
Parametros:
            p_autor           Nome do autor a ser mostrado no comentário.
            p_vazia           Se TRUE verificar se a tabela está vazia.
Descrição:
            Monta todos os LOAD em formato qlikview de todas as tabelas
=============================================================================*/

CREATE OR REPLACE FUNCTION QV_MontarTodosLoad(p_autor IN varchar,
p_vazia IN boolean)
RETURNS VARCHAR AS
$BODY$
DECLARE
      v_tabela VARCHAR(61);
      v_conta INTEGER;
      c_tabela RECORD;
      v_retorno VARCHAR(32676);
      v_sql VARCHAR(200);
BEGIN
      v_retorno := '';

      FOR c_tabela IN (
            SELECT t.TABLE_NAME FROM INFORMATION_SCHEMA.TABLES t
            WHERE t.TABLE_TYPE='BASE TABLE' AND t.TABLE_SCHEMA='public'
            ORDER BY t.TABLE_NAME)
      LOOP
            v_tabela := c_tabela.table_name;

            if p_vazia = TRUE then
                  v_sql := 'SELECT COUNT(*) FROM ' || v_tabela;
                  EXECUTE v_sql INTO v_conta;

                  if v_conta > 0 then
                        v_retorno := v_retorno ||
                        QV_MontarLoad(v_tabela,
                                    p_autor, FALSE, FALSE);
                  end if;
            else
                  v_retorno := v_retorno ||
                  QV_MontarLoad(v_tabela, p_autor, FALSE, FALSE);
            end if;

      END LOOP;

      RETURN v_retorno;
END;
$BODY$
LANGUAGE plpgsql;
```

13.8. A Função QV_ListarTabelasColunas

```
/*=============================================================================
Procedimento:     QV_ListarTabelasColunas
Autor:            Henrique Figueiredo de Souza
Data Criação:     14/02/2013
```

```
Parametros:
            p_comentario      Se TRUE incluir comando para gerar comentario.
Descrição:
            Lista todas a tabelas e colunas do banco de dados
================================================================================*/

CREATE OR REPLACE FUNCTION QV_ListarTabelasColunas(p_comentario IN boolean)
RETURNS VARCHAR AS
$BODY$
DECLARE
      v_tabela_anterior VARCHAR(61);
      v_coluna VARCHAR(8000);
      v_cr CHAR(2);
      v_conta integer;
      c_tabela RECORD;
      v_retorno VARCHAR(32676);
BEGIN
      v_cr := chr(13) || chr(10);
      v_tabela_anterior := '';
      v_conta := 0;
      v_retorno := '';

      FOR c_tabela IN (
            SELECT c.TABLE_NAME, c.COLUMN_NAME
            FROM INFORMATION_SCHEMA.COLUMNS c, information_schema.tables t
            WHERE C.TABLE_SCHEMA='public' AND C.TABLE_SCHEMA=T.TABLE_SCHEMA
            AND c.TABLE_NAME=t.TABLE_NAME and t.table_type='BASE TABLE'
            ORDER BY c.TABLE_NAME, c.ORDINAL_POSITION)
      LOOP

            if v_tabela_anterior<>c_tabela.table_name then

                  if v_coluna is not null then
                        --RAISE NOTICE '%', v_coluna || v_cr;
                        v_retorno := v_retorno || v_coluna || v_cr;
                  end if;

                  v_coluna:='';
                  v_conta := 0;
            end if;

            if p_comentario=TRUE then

                  if v_conta=0 then
                        v_coluna := 'comment on table ' ||
                        c_tabela.table_name || ' is '
                        || chr(39) || chr(9) || chr(9) ||
                        chr(39) || ';' || v_cr || v_coluna;
                        v_conta := 1;
                  end if;

                  v_coluna := v_coluna || 'comment on column ' ||
                  c_tabela.table_name || '.' || c_tabela.column_name
                  || ' is ' || chr(39) || chr(9) || chr(9) ||
                  chr(39) || ';' || v_cr;
            else
                  v_coluna := v_coluna || c_tabela.table_name || '.'
                  || c_tabela.column_name || v_cr;
            end if;

            v_tabela_anterior := c_tabela.table_name;

      END LOOP;
```

```
	--RAISE NOTICE '%', v_coluna || v_cr;
	v_retorno := v_retorno || v_coluna || v_cr;

	RETURN	v_retorno;
END;
$BODY$
LANGUAGE plpgsql;
```

13.9. A Função QV_ListarTabelasRestricoes

```
/*================================================================================
Procedimento:	QV_ListarTabelasRestricoes
Autor:		Henrique Figueiredo de Souza
Data Criação:	14/02/2013
Descrição:
		Lista todas as tabelas e suas restrições.
================================================================================*/

CREATE OR REPLACE FUNCTION QV_ListarTabelasRestricoes()
RETURNS VARCHAR AS
$BODY$
DECLARE
	v_tabela_anterior VARCHAR(61);
	v_tabela VARCHAR(8000);
	v_restricao VARCHAR(8000);
	v_cr CHAR(2);
	c_tabela RECORD;
	v_retorno VARCHAR(32676);
BEGIN
	v_cr := chr(13) || chr(10);
	v_tabela_anterior := '';
	v_restricao := '';
	v_retorno := '';

	FOR c_tabela IN (
		SELECT tc.TABLE_NAME, tc.CONSTRAINT_NAME, tc.CONSTRAINT_TYPE,
		(SELECT rc.UNIQUE_CONSTRAINT_NAME
		FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
		WHERE rc.CONSTRAINT_NAME=tc.CONSTRAINT_NAME) R_CONSTRAINT_NAME,
		(SELECT distinct tc1.TABLE_NAME
			FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
			WHERE tc1.CONSTRAINT_NAME=(SELECT rc.UNIQUE_CONSTRAINT_NAME
			FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
			WHERE rc.CONSTRAINT_NAME=tc.CONSTRAINT_NAME)) R_TABLE_NAME
		FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc
		WHERE tc.CONSTRAINT_TYPE in ('PRIMARY KEY','FOREIGN KEY')
		AND tc.CONSTRAINT_SCHEMA='public'
		ORDER BY tc.TABLE_NAME, tc.CONSTRAINT_TYPE DESC)
	LOOP

		if v_tabela_anterior<>c_tabela.table_name then

			if v_restricao is not null then
				--RAISE NOTICE '%', v_restricao;
				v_retorno := v_retorno || v_restricao || v_cr;
			end if;

			v_restricao:='';
		end if;

		if c_tabela.CONSTRAINT_TYPE='PRIMARY KEY' then
			v_restricao := v_restricao || c_tabela.CONSTRAINT_NAME
```

```
                  || ' = ' || c_tabela.table_name || '(' ||
                  QV_ColunasDaRestricaoDaTabela(c_tabela.table_name,
                  c_tabela.CONSTRAINT_NAME) || ')' || v_cr;
            else
                  v_restricao := v_restricao || c_tabela.CONSTRAINT_NAME
                  || ' = ' || c_tabela.table_name || '(' ||
                  QV_ColunasDaRestricaoDaTabela(c_tabela.table_name,
                  c_tabela.CONSTRAINT_NAME) || ') --> ' ||
                  c_tabela.R_TABLE_NAME || '(' ||
                  QV_ColunasDaRestricaoDaTabela(c_tabela.R_TABLE_NAME,
                  c_tabela.R_CONSTRAINT_NAME) || ')' || v_cr;
            end if;

            v_tabela_anterior := c_tabela.table_name;

      END LOOP;

      --RAISE NOTICE '%', v_restricao;
      v_retorno := v_retorno || v_restricao || v_cr;

      RETURN      v_retorno;
END;
$BODY$
LANGUAGE plpgsql;
```

13.10. A Função QV_TabelasNaoPossuem_PK_nemFK

```
/*================================================================================
Procedimento:     QV_TabelasNaoPossuem_PK_nemFK
Autor:            Henrique Figueiredo de Souza
Data Criação:     14/02/2013
Descrição:
            Lista todas as tabelas que NÃO possuem Chave Primária e
            não possuem Chave Estrangeira, ordenados por data de criação.
================================================================================*/

CREATE OR REPLACE FUNCTION TabelasNaoPossuem_PK_nemFK()
RETURNS VARCHAR AS
$BODY$
DECLARE
      v_tabela VARCHAR(32676);
      v_cr CHAR(2);
      c_objeto RECORD;
BEGIN
      v_cr := chr(13) || chr(10);
      v_tabela := '';

      FOR c_objeto IN (
            select t.table_name
            from information_schema.tables t
            where t.table_schema='public' and t.table_type='BASE TABLE'
            AND t.table_name not in (SELECT DISTINCT tc.TABLE_NAME
            FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc
            WHERE tc.table_schema='public'
            and tc.CONSTRAINT_TYPE IN ('PRIMARY KEY','FOREIGN KEY')))
      LOOP
            v_tabela := v_tabela || c_objeto.OBJECT_NAME || v_cr;

      END LOOP;

      --RAISE NOTICE '%', v_tabela;
```

```
	RETURN	v_tabela;
END;
$BODY$
LANGUAGE plpgsql;
```

13.11. A Função QV_TabelasSomente_PK_naoFK

```
/*================================================================================
Procedimento:	QV_TabelasSomente_PK_naoFK
Autor:		Henrique Figueiredo de Souza
Data Criação:	14/02/2013
Descrição:
		Lista todas as tabelas que possuem SOMENTE Chave Primária e
		não possuem Chave Estrangeira, ordenados por data de criação.
================================================================================*/

CREATE OR REPLACE FUNCTION QV_TabelasSomente_PK_naoFK()
RETURNS VARCHAR AS
$BODY$
DECLARE
	v_restricao VARCHAR(8000);
	v_cr CHAR(2);
	v_retorno VARCHAR(32676);
	c_objeto RECORD;
	c_tabela RECORD;
BEGIN
	v_cr := chr(13) || chr(10);
	v_retorno := '';

	FOR c_objeto IN (
		select t.table_name
		from (select t1.table_name
			from information_schema.tables t1
			where t1.table_schema='public'
			and t1.table_type='BASE TABLE'
			AND t1.table_name in (SELECT DISTINCT tc1.TABLE_NAME
			FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
			WHERE tc1.table_schema='public'
			and tc1.CONSTRAINT_TYPE IN ('PRIMARY KEY','FOREIGN KEY'))) t
		WHERE t.table_name not in (SELECT DISTINCT tc1.TABLE_NAME
		FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
		WHERE tc1.table_schema='public'
		and tc1.CONSTRAINT_TYPE='FOREIGN KEY'))
	LOOP
		v_restricao := '';

		FOR c_tabela IN (
			SELECT tc.TABLE_NAME, tc.CONSTRAINT_NAME, tc.CONSTRAINT_TYPE,
			(SELECT rc.UNIQUE_CONSTRAINT_NAME
			FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
		WHERE rc.CONSTRAINT_NAME=tc.CONSTRAINT_NAME) R_CONSTRAINT_NAME,
			(SELECT distinct tc1.TABLE_NAME
			FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
			WHERE tc1.CONSTRAINT_NAME=(SELECT rc.UNIQUE_CONSTRAINT_NAME
				FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
			WHERE rc.CONSTRAINT_NAME=tc.CONSTRAINT_NAME)) R_TABLE_NAME
			FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc
			WHERE tc.CONSTRAINT_SCHEMA='public'
			AND tc.TABLE_NAME=c_objeto.table_name)
		LOOP

			if c_tabela.CONSTRAINT_TYPE='PRIMARY KEY' then
```

```
                        v_restricao := v_restricao ||
                        c_tabela.CONSTRAINT_NAME || ' = '
                        || c_tabela.table_name || '(' ||
                        QV_ColunasDaRestricaoDaTabela(c_tabela.table_name,
                        c_tabela.CONSTRAINT_NAME) || ')' || v_cr;
                  end if;

            END LOOP;

            --RAISE NOTICE '%', v_restricao;
            v_retorno := v_retorno || v_restricao;

      END LOOP;

      RETURN      v_retorno;
END;
$BODY$
LANGUAGE plpgsql;
```

13.12. A Função QV_TabelasPossuem_FK

```
/*================================================================================
Procedimento:     QV_TabelasPossuem_FK
Autor:            Henrique Figueiredo de Souza
Data Criação:     14/02/2013
Descrição:
            Lista todas as tabelas que possuem Chave Estrangeira e podem
            ou não possuir Chave Primária, ordenados por data de criação.
================================================================================*/

CREATE OR REPLACE FUNCTION QV_TabelasPossuem_FK()
RETURNS VARCHAR AS
$BODY$
DECLARE
      v_restricao VARCHAR(8000);
      v_cr CHAR(2);
      v_retorno VARCHAR(32676);
      c_objeto RECORD;
      c_tabela RECORD;
BEGIN
      v_cr := chr(13) || chr(10);
      v_retorno := '';

      FOR c_objeto IN (
            select t.table_name
            from information_schema.tables t
            where t.table_schema='public'
            and t.table_type='BASE TABLE'
            AND t.table_name in (SELECT DISTINCT tc.TABLE_NAME
            FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc
            WHERE tc.table_schema='public'
            and tc.CONSTRAINT_TYPE='FOREIGN KEY'))
      LOOP
            v_restricao := '';

            FOR c_tabela IN (
                  SELECT tc.TABLE_NAME, tc.CONSTRAINT_NAME, tc.CONSTRAINT_TYPE,
                  (SELECT rc.UNIQUE_CONSTRAINT_NAME
                  FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
            WHERE rc.CONSTRAINT_NAME=tc.CONSTRAINT_NAME) R_CONSTRAINT_NAME,
                  (SELECT distinct tc1.TABLE_NAME
                        FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
```

```
                  WHERE tc1.CONSTRAINT_NAME=(SELECT rc.UNIQUE_CONSTRAINT_NAME
                        FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
                  WHERE rc.CONSTRAINT_NAME=tc.CONSTRAINT_NAME)) R_TABLE_NAME
                  FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc
                  WHERE tc.CONSTRAINT_SCHEMA='public'
                  AND tc.TABLE_NAME=c_objeto.table_name)
            LOOP

                  if c_tabela.CONSTRAINT_TYPE='FOREIGN KEY' then
                        v_restricao := v_restricao ||
                        c_tabela.CONSTRAINT_NAME || ' = '
                        || c_tabela.table_name || '(' ||
                        QV_ColunasDaRestricaoDaTabela(c_tabela.table_name,
                        c_tabela.CONSTRAINT_NAME) || ') --> ' ||
                        c_tabela.R_TABLE_NAME || '(' ||
                        QV_ColunasDaRestricaoDaTabela(c_tabela.R_TABLE_NAME,
                        c_tabela.R_CONSTRAINT_NAME) || ')' || v_cr;
            end if;

            END LOOP;

            --RAISE NOTICE '%', v_restricao;
            v_retorno := v_retorno || v_restricao;

      END LOOP;

      RETURN      v_retorno;
END;
$BODY$
LANGUAGE plpgsql;
```

13.13. A Função QV_TabelasSomente_FK_naoPK

```
/*================================================================================
Procedimento:     QV_TabelasSomente_FK_naoPK
Autor:            Henrique Figueiredo de Souza
Data Criação:     14/02/2013
Descrição:
            Lista todas as tabelas que possuem SOMENTE Chave Estrangeira
            e não possuem Chave Primária, ordenados por data de criação.
================================================================================*/

CREATE OR REPLACE FUNCTION QV_TabelasSomente_FK_naoPK()
RETURNS VARCHAR AS
$BODY$
DECLARE
      v_restricao VARCHAR(8000);
      v_cr CHAR(2);
      v_retorno VARCHAR(32676);
      c_objeto RECORD;
      c_tabela RECORD;
BEGIN
      v_cr := chr(13) || chr(10);
      v_retorno := '';

      FOR c_objeto IN (
            select t.table_name
            from (select t1.table_name
                  from information_schema.tables t1
                  where t1.table_schema='public'
                  and t1.table_type='BASE TABLE'
                  AND t1.table_name not in (SELECT DISTINCT tc1.TABLE_NAME
```

```
                  FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
                  WHERE tc1.table_schema='public'
                  and tc1.CONSTRAINT_TYPE='PRIMARY KEY')) t
            WHERE t.table_name in (SELECT DISTINCT tc1.TABLE_NAME
            FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
            WHERE tc1.table_schema='public'
            and tc1.CONSTRAINT_TYPE='FOREIGN KEY'))
      LOOP
            v_restricao := '';

            FOR c_tabela IN (
                  SELECT tc.TABLE_NAME, tc.CONSTRAINT_NAME, tc.CONSTRAINT_TYPE,
                  (SELECT rc.UNIQUE_CONSTRAINT_NAME
                  FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
            WHERE rc.CONSTRAINT_NAME=tc.CONSTRAINT_NAME) R_CONSTRAINT_NAME,
                  (SELECT distinct tc1.TABLE_NAME
                        FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
                  WHERE tc1.CONSTRAINT_NAME=(SELECT rc.UNIQUE_CONSTRAINT_NAME
                        FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
                  WHERE rc.CONSTRAINT_NAME=tc.CONSTRAINT_NAME)) R_TABLE_NAME
                  FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc
                  WHERE tc.table_schema='public'
                  AND tc.TABLE_NAME=c_objeto.table_name)
            LOOP

                  if c_tabela.CONSTRAINT_TYPE='FOREIGN KEY' then
                        v_restricao := v_restricao ||
                        c_tabela.CONSTRAINT_NAME || ' = '
                        || c_tabela.table_name || '(' ||
                        QV_ColunasDaRestricaoDaTabela(c_tabela.table_name,
                        c_tabela.CONSTRAINT_NAME) || ') --> ' ||
                        c_tabela.R_TABLE_NAME || '(' ||
                        QV_ColunasDaRestricaoDaTabela(c_tabela.R_TABLE_NAME,
                        c_tabela.R_CONSTRAINT_NAME) || ')' || v_cr;
            end if;

            END LOOP;

            --RAISE NOTICE '%', v_restricao;
            v_retorno := v_retorno || v_restricao;

      END LOOP;

      RETURN      v_retorno;
END;
$BODY$
LANGUAGE plpgsql;
```

13.14. A Função QV_DescomentarTabelasColunas

```
/*================================================================================
Procedimento:     QV_DescomentarTabelasColunas
Autor:            Henrique Figueiredo de Souza
Data Criação:     14/02/2013
Descrição:
            Descomentar todas a tabelas e colunas do banco de dados
================================================================================*/

CREATE OR REPLACE FUNCTION QV_DescomentarTabelasColunas()
RETURNS VARCHAR AS
$BODY$
DECLARE
```

```
	v_tabela_anterior VARCHAR(61);
	v_cr CHAR(2);
	v_coluna VARCHAR(32676);
	v_retorno VARCHAR(32676);
	c_tabela RECORD;
	v_conta integer;
BEGIN
	v_cr := chr(13) || chr(10);
	v_coluna := '';
	v_retorno := '';
	v_tabela_anterior := '';
	v_conta := 0;

	FOR c_tabela IN (
		SELECT c.TABLE_NAME, c.COLUMN_NAME
		FROM INFORMATION_SCHEMA.COLUMNS c, information_schema.tables t
		WHERE C.TABLE_SCHEMA='public' AND C.TABLE_SCHEMA=T.TABLE_SCHEMA
		AND c.TABLE_NAME=t.TABLE_NAME and t.table_type='BASE TABLE'
		ORDER BY c.TABLE_NAME, c.ORDINAL_POSITION)
	LOOP

		if v_tabela_anterior<>c_tabela.table_name then

			if v_coluna is not null then
				--RAISE NOTICE '%', v_coluna || v_cr;
				v_retorno := v_retorno || v_coluna || v_cr;
			end if;

			v_coluna:='';
			v_conta := 0;
		end if;

		if v_conta=0 then
			v_coluna := 'comment on table ' || c_tabela.table_name ||
			' is ' || chr(39) || chr(39) || ';' || v_cr || v_coluna;
			v_conta := 1;
		end if;

		v_coluna := v_coluna || 'comment on column ' ||
		c_tabela.table_name || '.' || c_tabela.column_name
		|| ' is ' || chr(39) || chr(39) || ';' || v_cr;

		v_tabela_anterior := c_tabela.table_name;

	END LOOP;

	--RAISE NOTICE '%', v_coluna || v_cr;
	v_retorno := v_retorno || v_coluna || v_cr;

	RETURN	v_retorno;
END;
$BODY$
LANGUAGE plpgsql;
```

13.15. A Função QV_ContarLinhasTabela

```
/*=============================================================================
Procedimento:	QV_ContarLinhasTabela
Autor:		Henrique Figueiredo de Souza
Data Criação:	14/02/2013
Parametros:
		p_tabela	Nome da tabela referenciada pela FK.
```

```
Descrição:
            Retorna a quantidade de linhas de uma tabela.
================================================================================*/

CREATE OR REPLACE FUNCTION QV_ContarLinhasTabela(p_tabela VARCHAR)
RETURNS BIGINT AS
$BODY$
DECLARE
      v_conta BIGINT;
      v_sql VARCHAR(200);
BEGIN
      v_sql := 'SELECT COUNT(*) FROM ' || p_tabela;
      EXECUTE v_sql INTO v_conta;
      RETURN v_conta;
END;
$BODY$
LANGUAGE plpgsql;
```

14. Os Procedimentos Qlikview para o MYSQL

```
SELECT QV_LimitaTamanhoTexto('ALFA BETA GAMA TETA', 3)
SELECT QV_AplicarMapa('pergunta','codigo_tipo_pergunta')
SELECT QV_ColunasDaRestricaoDaTabela('pergunta','PERGUNTA_TIPO_PERGUNTA_FK');
SELECT QV_ListarTabelaRestricao('pergunta');
SELECT QV_MontarLoad('pergunta','Henrique Figueiredo de Souza', FALSE, FALSE);

CALL QV_ContarLinhasTabela('pergunta', @resultado);
SELECT @resultado;

CALL QV_MontarTodosMappingLoad('Henrique Figueiredo de Souza', FALSE, TRUE,
@resultado);
SELECT @resultado;

CALL QV_MontarTodosLoad('Henrique Figueiredo de Souza', TRUE, @resultado);
SELECT @resultado;

SELECT QV_ListarTabelasColunas(TRUE);
SELECT QV_ListarTabelasRestricoes();
SELECT QV_TabelasNaoPossuem_PK_nemFK();
SELECT QV_TabelasSomente_PK_naoFK();
SELECT QV_TabelasPossuem_FK();
SELECT QV_TabelasSomente_FK_naoPK();
SELECT QV_DescomentarTabelasColunas();

DROP FUNCTION ContaPalavra;
DROP FUNCTION QV_LimitaTamanhoTexto;
DROP FUNCTION QV_AplicarMapa;
DROP FUNCTION QV_ColunasDaRestricaoDaTabela;
DROP FUNCTION QV_ListarTabelaRestricao;
DROP FUNCTION QV_MontarLoad;
DROP PROCEDURE QV_ContarLinhasTabela;
DROP PROCEDURE QV_MontarTodosMappingLoad;
DROP PROCEDURE QV_MontarTodosLoad;
DROP FUNCTION QV_ListarTabelasColunas;
DROP FUNCTION QV_ListarTabelasRestricoes;
DROP FUNCTION QV_TabelasNaoPossuem_PK_nemFK;
DROP FUNCTION QV_TabelasSomente_PK_naoFK;
DROP FUNCTION QV_TabelasPossuem_FK;
DROP FUNCTION QV_TabelasSomente_FK_naoPK;
DROP FUNCTION QV_DescomentarTabelasColunas;
```

14.1. A Função ContaPalavra

```
DELIMITER $$
CREATE FUNCTION ContaPalavra(str VARCHAR(21845))
RETURNS INTEGER
BEGIN
	DECLARE i INTEGER DEFAULT 1;
	DECLARE palavras INTEGER DEFAULT 0;
	DECLARE tamanho INTEGER DEFAULT IFNULL(LENGTH(str),0);
	DECLARE dentro_uma_palavra BOOLEAN;

	WHILE i <= (tamanho + 1) DO

		IF ASCII(SUBSTR(str, i, 1)) < 33 OR i > tamanho THEN
			IF dentro_uma_palavra THEN
				SET palavras = palavras + 1;
				SET dentro_uma_palavra = FALSE;
			END IF;
		ELSE
			SET dentro_uma_palavra = TRUE;
		END IF;

		SET i = i + 1;
	END WHILE;

	RETURN palavras;
END
$$
```

14.2. A Função QV_LimitaTamanhoTexto

```
/*================================================================================
Procedimento:	QV_LimitaTamanhoTexto
Autor:		Henrique Figueiredo de Souza
Data Criação:	16/02/2013
Parametros:
		p_str		Texto a ser limitado.
		p_limite	Limite de palavras por linhas.
Descrição:
		Limita o texto fornecido a uma determinada quantidades
		de palavras por linha.
================================================================================*/
DELIMITER $$
CREATE FUNCTION QV_LimitaTamanhoTexto(p_str TEXT, p_limite INTEGER)
RETURNS TEXT
BEGIN
	DECLARE i INTEGER DEFAULT 1;
	DECLARE v_palavras INTEGER DEFAULT 0;
	DECLARE v_tamanho INTEGER DEFAULT IFNULL(LENGTH(p_str),0);
	DECLARE v_dentro_uma_palavra BOOLEAN;
	DECLARE v_texto TEXT;

	SET v_texto = '';
	WHILE i <= (v_tamanho + 1) DO
		SET v_texto = CONCAT(v_texto, SUBSTR(p_str, i, 1));

		IF ASCII(SUBSTR(p_str, i, 1)) < 33 OR i > v_tamanho THEN
			IF v_dentro_uma_palavra THEN
				SET v_palavras = v_palavras + 1;
				SET v_dentro_uma_palavra = FALSE;

				if v_palavras=p_limite then
```

```
                        SET v_texto = CONCAT(v_texto, char(13), char(10));
                        SET v_palavras = 0;
                    end if;
                END IF;
            ELSE
                SET v_dentro_uma_palavra = TRUE;
            END IF;

            SET i = i + 1;
        END WHILE;

        RETURN v_texto;
END
$$
```

14.3. A Função QV_AplicarMapa

```
/*================================================================================
Procedimento:     QV_AplicarMapa
Autor:            Henrique Figueiredo de Souza
Data Criação:     16/02/2013
Parametros:
            p_tabela    Nome da tabela referenciada pela FK.
            p_coluna    Nome da coluna que possui uma FK.
Descrição:
            Aplicar o mapa carregado, onde a coluna é uma
            Chave Estrangeira e tabela é a tabela referenciada pela
            chave estrangeira da coluna.
================================================================================*/
DELIMITER $$
CREATE FUNCTION QV_AplicarMapa(p_tabela varchar(64),
p_coluna varchar(64))
RETURNS TEXT
BEGIN
        DECLARE v_mapa TEXT;
        DECLARE v_tabela VARCHAR(64);
        DECLARE v_tabela_mapa VARCHAR(64);

        SELECT (SELECT rc.REFERENCED_TABLE_NAME
                    FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
                WHERE rc.CONSTRAINT_NAME=ccu.CONSTRAINT_NAME) R_TABLE_NAME
                INTO v_tabela_mapa
                FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc,
                INFORMATION_SCHEMA.KEY_COLUMN_USAGE ccu
                WHERE tc.TABLE_NAME=ccu.TABLE_NAME
                AND tc.CONSTRAINT_SCHEMA=schema()
                AND tc.CONSTRAINT_NAME=ccu.CONSTRAINT_NAME
                AND tc.CONSTRAINT_TYPE='FOREIGN KEY'
                AND tc.TABLE_NAME=p_tabela
                AND ccu.COLUMN_NAME=p_coluna;

        select t.table_name INTO v_tabela
                from (select t1.table_name
                        from information_schema.tables t1
                        where t1.table_schema=schema() and t1.table_type='BASE TABLE'
                        AND t1.table_name in (SELECT DISTINCT tc1.TABLE_NAME
                        FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
                        WHERE tc1.CONSTRAINT_SCHEMA=schema()
                        and tc1.CONSTRAINT_TYPE IN ('PRIMARY KEY','FOREIGN KEY'))) t
                WHERE t.table_name not in (SELECT DISTINCT tc1.TABLE_NAME
                        FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
                        WHERE tc1.CONSTRAINT_SCHEMA=schema()
                        and tc1.CONSTRAINT_TYPE='FOREIGN KEY')
                AND t.table_name=v_tabela_mapa;
```

```
	SET v_mapa = '';

	/* verifica se a coluna da tabela é uma Chave Estrangeira */
	if (v_tabela_mapa is not null) and (length(v_tabela_mapa) > 0) then

		-- verifica se tabela possui somente Chave Primaria
		if (v_tabela is not null) and (length(v_tabela) > 0) then
			SET v_mapa = CONCAT('ApplyMap(', char(39), 'Mapa',
			v_tabela_mapa, char(39), ',', p_coluna, ')');
		end if;
	end if;

	RETURN v_mapa;
END
$$
```

14.4. A Função QV_ColunasDaRestricaoDaTabela

```
/*==============================================================================
Procedimento:	QV_ColunasDaRestricaoDaTabela
Autor:		Henrique Figueiredo de Souza
Data Criação:	16/02/2013
Parametros:
		p_tabela    Nome da tabela referenciada pela FK.
		p_restricao Nome da restrição.
Descrição:
		Retorna as colunas de uma tabela e sua restrição.
==============================================================================*/
DELIMITER $$
CREATE FUNCTION QV_ColunasDaRestricaoDaTabela(p_tabela varchar(64),
p_restricao varchar(64))
RETURNS TEXT
BEGIN
	DECLARE h_tabela BOOLEAN DEFAULT FALSE;
	DECLARE v_coluna TEXT;
	DECLARE v_nome_coluna varchar(64);
	DECLARE c_tabela CURSOR FOR
		SELECT ccu.COLUMN_NAME
		FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE ccu
		where ccu.TABLE_NAME=p_tabela
		and ccu.CONSTRAINT_NAME=p_restricao;
	DECLARE CONTINUE HANDLER FOR NOT FOUND SET h_tabela = TRUE;

	SET v_coluna = '';

	OPEN c_tabela;

	loop_tabela: LOOP
		FETCH c_tabela INTO v_nome_coluna;
		IF h_tabela THEN
			LEAVE loop_tabela;
		END IF;

		SET v_coluna = CONCAT(v_coluna, ', ', v_nome_coluna);

	END LOOP loop_tabela;

	CLOSE c_tabela;

	SET v_coluna = substr(v_coluna, 3, length(v_coluna));
```

```
	RETURN v_coluna;

END
$$
```

14.5. A Função QV_ListarTabelaRestricao

```
/*================================================================================
Procedimento:	QV_ListarTabelaRestricao
Autor:		Henrique Figueiredo de Souza
Data Criação:	16/02/2013
Parametros:
		p_tabela	Nome da tabela.
Descrição:
		Lista restrições de uma tabela.
================================================================================*/
DELIMITER $$
CREATE FUNCTION QV_ListarTabelaRestricao(p_tabela varchar(64))
RETURNS TEXT
BEGIN
	DECLARE h_tabela BOOLEAN DEFAULT FALSE;
	DECLARE	v_restricao TEXT;
	DECLARE v_cr CHAR(1) DEFAULT char(10);
	DECLARE	v_tam INTEGER;
	DECLARE v_nome_tabela VARCHAR(64);
	DECLARE v_nome_restricao VARCHAR(64);
	DECLARE v_tipo_restricao VARCHAR(64);
	DECLARE v_restricao_ref VARCHAR(64);
	DECLARE v_tabela_ref VARCHAR(64);
	DECLARE c_tabela CURSOR FOR
		SELECT tc.TABLE_NAME, tc.CONSTRAINT_NAME, tc.CONSTRAINT_TYPE,
		(SELECT rc.UNIQUE_CONSTRAINT_NAME
		FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
		WHERE rc.CONSTRAINT_NAME=tc.CONSTRAINT_NAME) R_CONSTRAINT_NAME,
		(SELECT rc1.REFERENCED_TABLE_NAME
			FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc1
		WHERE rc1.CONSTRAINT_NAME=tc.CONSTRAINT_NAME) R_TABLE_NAME
		FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc
		WHERE tc.CONSTRAINT_TYPE in ('PRIMARY KEY','FOREIGN KEY')
		AND tc.CONSTRAINT_SCHEMA=schema() AND tc.TABLE_NAME=p_tabela
		ORDER BY tc.TABLE_NAME, tc.CONSTRAINT_TYPE DESC;
	DECLARE CONTINUE HANDLER FOR NOT FOUND SET h_tabela = TRUE;

	SET v_restricao = '';

	OPEN c_tabela;

	loop_tabela: LOOP
		FETCH c_tabela INTO v_nome_tabela, v_nome_restricao,
		v_tipo_restricao, v_restricao_ref, v_tabela_ref;
		IF h_tabela THEN
			LEAVE loop_tabela;
		END IF;

		if v_tipo_restricao='PRIMARY KEY' then
			SET v_restricao = CONCAT(v_restricao, v_nome_tabela, '(',
			QV_ColunasDaRestricaoDaTabela(v_nome_tabela,
			v_nome_restricao), ')', v_cr);
		else
			SET v_restricao = CONCAT(v_restricao, v_nome_tabela, '(',
			QV_ColunasDaRestricaoDaTabela(v_nome_tabela,
			v_nome_restricao), ') --> ', v_tabela_ref, '(',
			QV_ColunasDaRestricaoDaTabela(v_tabela_ref,
			v_restricao_ref), ')', v_cr);
```

```
            end if;

      END LOOP loop_tabela;

      CLOSE c_tabela;

      SET v_tam = length(v_restricao)-2;

      if v_tam > 0 then
            SET v_restricao = substr(v_restricao, 1, v_tam);
      end if;

      RETURN v_restricao;
END
$$
```

14.6. A Função QV_MontarLoad

```
/*==============================================================================
Procedimento:     QV_MontarLoad
Autor:            Henrique Figueiredo de Souza
Data Criação:     17/02/2013
Parametros:
            p_tabela    Nome da tabela a gerar o LOAD.
            p_autor           Nome do autor a ser mostrado no comentário.
            p_mapeada   TRUE para mapping load, FALSE para sem mapping load
            p_2_colunas Limita a saida a somente a duas colunas.
Descrição:
            Monta um LOAD em formato qlikview de uma tabela
==============================================================================*/
DELIMITER $$
CREATE FUNCTION QV_MontarLoad(p_tabela VARCHAR(64), p_autor VARCHAR(128),
p_mapeada boolean, p_2_colunas boolean)
RETURNS TEXT
BEGIN
      DECLARE h_coluna BOOLEAN DEFAULT FALSE;
      DECLARE v_tabela VARCHAR(64);
      DECLARE v_coluna TEXT DEFAULT '';
      DECLARE v_comenta VARCHAR(8000);
      DECLARE v_rotulo VARCHAR(4000);
      DECLARE v_rotulo_tabela TEXT;
      DECLARE v_mapa TEXT;
      DECLARE v_nomecoluna TEXT;
      DECLARE v_cr CHAR(1) DEFAULT char(10);
      DECLARE     v_conta INTEGER DEFAULT 0;
      DECLARE     v_retorno TEXT;

      DECLARE v_nome_tabela VARCHAR(64);
      DECLARE v_nome_coluna VARCHAR(64);
      DECLARE v_ordem_coluna INTEGER;
      DECLARE v_comentario VARCHAR(1024);
      DECLARE v_comentario_tabela VARCHAR(2048);

      DECLARE c_coluna CURSOR FOR
            SELECT c.TABLE_NAME, c.COLUMN_NAME, c.ORDINAL_POSITION,
            (select column_comment from information_schema.columns
            where table_name=c.TABLE_NAME
            and column_name=c.COLUMN_NAME) as COLUMN_COMMENT,
            (select table_comment from information_schema.tables
            where table_name=c.TABLE_NAME) as TABLE_COMMENT
            FROM INFORMATION_SCHEMA.COLUMNS c
            WHERE c.TABLE_NAME=p_tabela
            AND c.TABLE_SCHEMA=schema()
            ORDER BY c.ORDINAL_POSITION;
```

```
	DECLARE CONTINUE HANDLER FOR NOT FOUND SET h_coluna = TRUE;

	SET v_retorno = '';
	OPEN c_coluna;

	loop_coluna: LOOP
		FETCH c_coluna INTO v_nome_tabela, v_nome_coluna,
		v_ordem_coluna, v_comentario, v_comentario_tabela;
		IF h_coluna THEN
			LEAVE loop_coluna;
		END IF;

		SET v_tabela = v_nome_tabela;
		SET v_rotulo = v_comentario;
		SET v_rotulo_tabela =
QV_LimitaTamanhoTexto(IFNULL(v_comentario_tabela,''), 10);
		SET v_mapa = QV_AplicarMapa(v_tabela, v_nome_coluna);

		if length(v_mapa) > 0 then
			SET v_nomecoluna = v_mapa;
		else
			SET v_nomecoluna = v_nome_coluna;
		end if;

		if length(v_rotulo) = 0 then
			SET v_coluna = CONCAT(v_coluna, ', ', v_cr,
			'	', v_nomecoluna);
		else
			SET v_coluna = CONCAT(v_coluna, ', ', v_cr,
			'	', v_nomecoluna, ' as [', v_rotulo, ']');
		end if;

		if p_2_colunas=TRUE then
			SET v_conta = v_conta + 1;

			if v_conta=2 then
				LEAVE loop_coluna;
			end if;
		end if;

	END LOOP loop_coluna;

	CLOSE c_coluna;

	if length(v_coluna) > 0 then

		SET v_coluna = substr(v_coluna, 3, length(v_coluna));

		SET v_comenta = CONCAT(v_cr,

	'/*============================================================================
======', v_cr);

		if p_mapeada=TRUE then
			SET v_comenta = CONCAT(v_comenta, 'Procedimento:', char(9),
			'Carga mapeada em memória do arquivo QVD', v_cr);
		else
			SET v_comenta = CONCAT(v_comenta, 'Procedimento:', char(9),
			'Carga em memória do arquivo QVD', v_cr);
		end if;
		SET v_comenta = CONCAT(v_comenta, 'ArquivoQVD:', char(9), char(9),
v_tabela, v_cr);
		SET v_comenta = CONCAT(v_comenta, 'Autor:', char(9), char(9),
char(9), p_autor, v_cr);
```

```
            SET v_comenta = CONCAT(v_comenta, 'Data Criação:', char(9),
                              DATE_FORMAT(NOW(),'%d/%m/%Y'), v_cr);

            if  length(v_rotulo_tabela) > 0 then
                  SET v_comenta = CONCAT(v_comenta, 'Descrição:', char(9),
                  v_rotulo_tabela, v_cr);
            end if;

            SET v_retorno = CONCAT(v_retorno, v_comenta, 'Restrições:', v_cr);

            SET v_retorno = CONCAT(v_retorno,
QV_ListarTabelaRestricao(p_tabela), v_cr);

            SET v_retorno = CONCAT(v_retorno,

      '================================================================================
====*/', v_cr);

            if p_mapeada=TRUE then
                  SET v_retorno = CONCAT(v_retorno, 'Mapa', v_tabela, ':', v_cr,
                  'Mapping LOAD', v_coluna, v_cr, 'FROM', v_cr, '$(vDiretorio)',
                  v_tabela, '.qvd', v_cr, '(qvd);', v_cr);
            else
                  SET v_retorno = CONCAT(v_retorno, v_tabela, ':', v_cr,
                  'LOAD', v_coluna, v_cr, 'FROM', v_cr, '$(vDiretorio)',
                  v_tabela, '.qvd', v_cr, '(qvd);', v_cr);
            end if;

      end if;

      RETURN      v_retorno;
END
$$
```

14.7. O Procedimento QV_ContarLinhasTabela

```
/*================================================================================
Procedimento:     QV_ContarLinhasTabela
Autor:            Henrique Figueiredo de Souza
Data Criação:     17/02/2013
Parametros:
            p_tabela    Nome da tabela referenciada pela FK.
Descrição:
            Retorna a quantidade de linhas de uma tabela.
================================================================================*/
DELIMITER $$
CREATE PROCEDURE QV_ContarLinhasTabela(p_tabela VARCHAR(64),
OUT p_resultado BIGINT)
BEGIN
      SET @v_sql = CONCAT('SELECT COUNT(*) INTO @v_conta FROM ', p_tabela);
      PREPARE stmt FROM @v_sql;
      EXECUTE stmt;
      DEALLOCATE PREPARE stmt;

      SET p_resultado = @v_conta;
END
$$
```

14.8. O Procedimento QV_MontarTodosMappingLoad

```
/*================================================================================
Procedimento:     QV_MontarTodosMappingLoad
Autor:            Henrique Figueiredo de Souza
```

```
Data Criação:     17/02/2013
Parametros:
            p_autor           Nome do autor a ser mostrado no comentário.
            p_vazia           Se 1 verificar se a tabela está vazia.
            p_2_colunas Limita a saida a somente a duas colunas.
Descrição:
            Monta todos os Mapping LOAD em formato qlikview
            de todas as tabelas que possuem SOMENTE Chave Primária.
================================================================================*/
DELIMITER $$
CREATE PROCEDURE QV_MontarTodosMappingLoad(p_autor varchar(128),
p_vazia boolean, p_2_colunas boolean, OUT p_resultado TEXT)
BEGIN
      DECLARE h_tabela BOOLEAN DEFAULT FALSE;
      DECLARE v_tabela VARCHAR(64);
      DECLARE v_retorno TEXT DEFAULT '';
      DECLARE c_tabela CURSOR FOR
            select t.table_name
            from (select t1.table_name
                  from information_schema.tables t1
                  where t1.table_schema=schema()
                  and t1.table_type='BASE TABLE'
                  AND t1.table_name in (
                  SELECT DISTINCT tc1.TABLE_NAME
                  FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
                  WHERE tc1.table_schema=schema()
                  and tc1.CONSTRAINT_TYPE IN ('PRIMARY KEY','FOREIGN KEY'))) t
            WHERE t.table_name not in (SELECT DISTINCT tc1.TABLE_NAME
            FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
            WHERE tc1.table_schema=schema()
            and tc1.CONSTRAINT_TYPE='FOREIGN KEY');
      DECLARE CONTINUE HANDLER FOR NOT FOUND SET h_tabela = TRUE;

      SET v_retorno = '';
      OPEN c_tabela;

      loop_tabela: LOOP
            FETCH c_tabela INTO v_tabela;
            IF h_tabela THEN
                  LEAVE loop_tabela;
            END IF;

            if p_vazia = TRUE then
                  SET @v_sql = CONCAT('SELECT COUNT(*) INTO @v_conta FROM ',
                                    v_tabela);
                  PREPARE stmt FROM @v_sql;
                  EXECUTE stmt;
                  DEALLOCATE PREPARE stmt;

                  if @v_conta > 0 then
                        SET v_retorno = CONCAT(v_retorno,
                        QV_MontarLoad(v_tabela, p_autor, TRUE, p_2_colunas));
                  end if;
            else
                  SET v_retorno = CONCAT(v_retorno,
                  QV_MontarLoad(v_tabela, p_autor, TRUE, p_2_colunas));
            end if;
      END LOOP loop_tabela;

      CLOSE c_tabela;

      SET p_resultado = v_retorno;
END
$$
```

14.9. O Procedimento QV_MontarTodosLoad

```
/*================================================================================
Procedimento:     QV_MontarTodosLoad
Autor:            Henrique Figueiredo de Souza
Data Criação:     14/02/2013
Parametros:
            p_autor           Nome do autor a ser mostrado no comentário.
            p_vazia           Se TRUE verificar se a tabela está vazia.
Descrição:
            Monta todos os LOAD em formato qlikview de todas as tabelas
================================================================================*/
DELIMITER $$
CREATE PROCEDURE QV_MontarTodosLoad(p_autor varchar(128),
p_vazia boolean, OUT p_resultado TEXT)
BEGIN
      DECLARE h_tabela BOOLEAN DEFAULT FALSE;
      DECLARE v_tabela VARCHAR(64);
      DECLARE v_retorno TEXT;
      DECLARE c_tabela CURSOR FOR
            SELECT t.TABLE_NAME FROM INFORMATION_SCHEMA.TABLES t
            WHERE t.TABLE_TYPE='BASE TABLE' AND t.TABLE_SCHEMA=schema()
            ORDER BY t.TABLE_NAME;
      DECLARE CONTINUE HANDLER FOR NOT FOUND SET h_tabela = TRUE;

      SET v_retorno = '';
      OPEN c_tabela;

      loop_tabela: LOOP
            FETCH c_tabela INTO v_tabela;
            IF h_tabela THEN
                  LEAVE loop_tabela;
            END IF;

            if p_vazia = TRUE then
                  SET @v_sql = CONCAT('SELECT COUNT(*) INTO @v_conta FROM ',
                                    v_tabela);
                  PREPARE stmt FROM @v_sql;
                  EXECUTE stmt;
                  DEALLOCATE PREPARE stmt;

                  if @v_conta > 0 then
                        SET v_retorno = CONCAT(v_retorno,
                        QV_MontarLoad(v_tabela,
                                    p_autor, FALSE, FALSE));
                  end if;
            else
                  SET v_retorno = CONCAT(v_retorno,
                  QV_MontarLoad(v_tabela, p_autor, FALSE, FALSE));
            end if;

      END LOOP loop_tabela;

      CLOSE c_tabela;

      SET p_resultado = v_retorno;
END
$$
```

14.10. A Função QV_ListarTabelasColunas

```
/*================================================================================
Procedimento:     QV_ListarTabelasColunas
```

```
Autor:            Henrique Figueiredo de Souza
Data Criação:     14/02/2013
Parametros:
            p_comentario      Se TRUE incluir comando  para gerar comentario.
Descrição:
            Lista todas a tabelas e colunas do banco de dados
================================================================================*/
DELIMITER $$
CREATE FUNCTION QV_ListarTabelasColunas(p_comentario boolean)
RETURNS TEXT
BEGIN
      DECLARE h_tabela BOOLEAN DEFAULT FALSE;
      DECLARE     v_tabela_anterior VARCHAR(64) DEFAULT '';
      DECLARE     v_nome_tabela VARCHAR(64);
      DECLARE     v_nome_coluna VARCHAR(64);
      DECLARE     v_tipo_coluna VARCHAR(64);
      DECLARE     v_coluna TEXT;
      DECLARE v_cr CHAR(1) DEFAULT char(10);
      DECLARE v_conta integer DEFAULT 0;
      DECLARE v_retorno TEXT DEFAULT '';
      DECLARE c_tabela CURSOR FOR
            SELECT c.TABLE_NAME, c.COLUMN_NAME, c.COLUMN_TYPE
            FROM INFORMATION_SCHEMA.COLUMNS c, information_schema.tables t
            WHERE C.TABLE_SCHEMA=schema() AND C.TABLE_SCHEMA=T.TABLE_SCHEMA
            AND c.TABLE_NAME=t.TABLE_NAME and t.table_type='BASE TABLE'
            ORDER BY c.TABLE_NAME, c.ORDINAL_POSITION;
      DECLARE CONTINUE HANDLER FOR NOT FOUND SET h_tabela = TRUE;

      OPEN c_tabela;

      loop_tabela: LOOP
            FETCH c_tabela INTO v_nome_tabela, v_nome_coluna, v_tipo_coluna;
            IF h_tabela THEN
                  LEAVE loop_tabela;
            END IF;

            if v_tabela_anterior<>v_nome_tabela then

                  if v_coluna is not null then
                        SET v_retorno = CONCAT(v_retorno, v_coluna, v_cr);
                  end if;

                  SET v_coluna='';
                  SET v_conta = 0;
            end if;

            if p_comentario=TRUE then

                  if v_conta=0 then
                        SET v_coluna = CONCAT('ALTER TABLE ',
                        v_nome_tabela, ' COMMENT '
                        , char(39), char(9), char(9),
                        char(39), ';', v_cr, v_coluna);
                        SET v_conta = 1;
                  end if;

                  SET v_coluna = CONCAT(v_coluna, 'ALTER TABLE ',
                  v_nome_tabela, ' MODIFY ', v_nome_coluna, ' ',
                  v_tipo_coluna, ' COMMENT ', char(39), char(9),
                  char(9), char(39), ';', v_cr);
            else
                  SET v_coluna = CONCAT(v_coluna, v_nome_tabela, '.'
                  , v_nome_coluna, v_cr);
            end if;
```

```
            SET v_tabela_anterior = v_nome_tabela;

      END LOOP loop_tabela;

      CLOSE c_tabela;

      SET v_retorno = CONCAT(v_retorno, v_coluna, v_cr);

      RETURN      v_retorno;
END;
$$
```

14.11. A Função QV_ListarTabelasRestricoes

```
/*================================================================================
Procedimento:     QV_ListarTabelasRestricoes
Autor:            Henrique Figueiredo de Souza
Data Criação:     14/02/2013
Descrição:
            Lista todas as tabelas e suas restrições.
================================================================================*/
DELIMITER $$
CREATE FUNCTION QV_ListarTabelasRestricoes()
RETURNS TEXT
BEGIN
      DECLARE h_tabela BOOLEAN DEFAULT FALSE;
      DECLARE v_tabela_anterior VARCHAR(61);
      DECLARE v_restricao TEXT DEFAULT '';
      DECLARE v_cr CHAR(1) DEFAULT char(10);
      DECLARE v_retorno TEXT DEFAULT '';
      DECLARE v_nome_tabela VARCHAR(64);
      DECLARE v_nome_restricao VARCHAR(64);
      DECLARE v_tipo_restricao VARCHAR(64);
      DECLARE v_restricao_ref VARCHAR(64);
      DECLARE v_tabela_ref VARCHAR(64);
      DECLARE c_tabela CURSOR FOR
            SELECT tc.TABLE_NAME, tc.CONSTRAINT_NAME, tc.CONSTRAINT_TYPE,
            (SELECT rc.UNIQUE_CONSTRAINT_NAME
            FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
            WHERE rc.CONSTRAINT_NAME=tc.CONSTRAINT_NAME) R_CONSTRAINT_NAME,
            (SELECT rc1.REFERENCED_TABLE_NAME
                  FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc1
            WHERE rc1.CONSTRAINT_NAME=tc.CONSTRAINT_NAME) R_TABLE_NAME
            FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc
            WHERE tc.CONSTRAINT_TYPE in ('PRIMARY KEY','FOREIGN KEY')
            AND tc.CONSTRAINT_SCHEMA=schema()
            ORDER BY tc.TABLE_NAME, tc.CONSTRAINT_TYPE DESC;
      DECLARE CONTINUE HANDLER FOR NOT FOUND SET h_tabela = TRUE;

      OPEN c_tabela;

      loop_tabela: LOOP
            FETCH c_tabela INTO v_nome_tabela, v_nome_restricao,
            v_tipo_restricao, v_restricao_ref, v_tabela_ref;
            IF h_tabela THEN
                  LEAVE loop_tabela;
            END IF;

            if v_tabela_anterior<>v_nome_tabela then

                  if v_restricao is not null then
                        SET v_retorno = CONCAT(v_retorno, v_restricao, v_cr);
                  end if;
```

```
                SET v_restricao='';
            end if;

            if v_tipo_restricao='PRIMARY KEY' then
                SET v_restricao = CONCAT(v_restricao, v_nome_restricao,
                ' = ', v_nome_tabela, '(',
                QV_ColunasDaRestricaoDaTabela(v_nome_tabela,
                v_nome_restricao), ')', v_cr);
            else
                SET v_restricao = CONCAT(v_restricao, v_nome_restricao,
                ' = ', v_nome_tabela, '(',
                QV_ColunasDaRestricaoDaTabela(v_nome_tabela,
                v_nome_restricao), ') --> ',
                v_tabela_ref, '(',
                QV_ColunasDaRestricaoDaTabela(v_tabela_ref,
                v_restricao_ref), ')', v_cr);
            end if;

            SET v_tabela_anterior = v_nome_tabela;

      END LOOP loop_tabela;

      CLOSE c_tabela;

      SET v_retorno = CONCAT(v_retorno, v_restricao, v_cr);

      RETURN      v_retorno;
END;
$$
```

14.12. A Função QV_TabelasNaoPossuem_PK_nemFK

```
/*================================================================================
Procedimento:     QV_TabelasNaoPossuem_PK_nemFK
Autor:            Henrique Figueiredo de Souza
Data Criação:     14/02/2013
Descrição:
            Lista todas as tabelas que NÃO possuem Chave Primária e
            não possuem Chave Estrangeira, ordenados por data de criação.
================================================================================*/
DELIMITER $$
CREATE FUNCTION QV_TabelasNaoPossuem_PK_nemFK()
RETURNS TEXT
BEGIN
      DECLARE h_tabela BOOLEAN DEFAULT FALSE;
      DECLARE     v_tabela TEXT DEFAULT '';
      DECLARE     v_nome_tabela varchar(64);
      DECLARE v_cr CHAR(1) DEFAULT char(10);
      DECLARE c_tabela CURSOR FOR
            select t.table_name
            from information_schema.tables t
            where t.table_schema=schema() and t.table_type='BASE TABLE'
            AND t.table_name not in (SELECT DISTINCT tc.TABLE_NAME
            FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc
            WHERE tc.table_schema=schema()
            and tc.CONSTRAINT_TYPE IN ('PRIMARY KEY','FOREIGN KEY'));
      DECLARE CONTINUE HANDLER FOR NOT FOUND SET h_tabela = TRUE;

      OPEN c_tabela;

      loop_tabela: LOOP
```

```
            FETCH c_tabela INTO v_nome_tabela;
            IF h_tabela THEN
                  LEAVE loop_tabela;
            END IF;

            SET v_tabela = CONCAT(v_tabela, v_nome_tabela, v_cr);

      END LOOP loop_tabela;

      CLOSE c_tabela;

      RETURN      v_tabela;
END;
$$
```

14.13. A Função QV_TabelasSomente_PK_naoFK

```
/*================================================================================
Procedimento:     QV_TabelasSomente_PK_naoFK
Autor:            Henrique Figueiredo de Souza
Data Criação:     14/02/2013
Descrição:
            Lista todas as tabelas que possuem SOMENTE Chave Primária e
            não possuem Chave Estrangeira, ordenados por data de criação.
================================================================================*/
DELIMITER $$
CREATE FUNCTION QV_TabelasSomente_PK_naoFK()
RETURNS TEXT
bloco1: BEGIN
      DECLARE h_tabela BOOLEAN DEFAULT FALSE;
      DECLARE     v_restricao TEXT DEFAULT '';
      DECLARE v_cr CHAR(1) DEFAULT char(10);
      DECLARE v_retorno TEXT DEFAULT '';
      DECLARE v_tabela VARCHAR(64);
      DECLARE c_tabela CURSOR FOR
            select t.table_name
            from (select t1.table_name
                  from information_schema.tables t1
                  where t1.table_schema=schema()
                  and t1.table_type='BASE TABLE'
                  AND t1.table_name in (SELECT DISTINCT tc1.TABLE_NAME
                  FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
                  WHERE tc1.table_schema=schema()
                  and tc1.CONSTRAINT_TYPE IN ('PRIMARY KEY','FOREIGN KEY'))) t
            WHERE t.table_name not in (SELECT DISTINCT tc1.TABLE_NAME
            FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
            WHERE tc1.table_schema=schema()
            and tc1.CONSTRAINT_TYPE='FOREIGN KEY');
      DECLARE CONTINUE HANDLER FOR NOT FOUND SET h_tabela = TRUE;

      OPEN c_tabela;

      loop_tabela: LOOP
            FETCH c_tabela INTO v_tabela;
            IF h_tabela THEN
                  LEAVE loop_tabela;
            END IF;

            SET v_restricao = '';

            bloco2: BEGIN
                  DECLARE h_restricao BOOLEAN DEFAULT FALSE;
```

```
                DECLARE v_nome_tabela VARCHAR(64);
                DECLARE v_nome_restricao VARCHAR(64);
                DECLARE v_tipo_restricao VARCHAR(64);
                DECLARE v_restricao_ref VARCHAR(64);
                DECLARE v_tabela_ref VARCHAR(64);
                DECLARE c_restricao CURSOR FOR
                      SELECT tc.TABLE_NAME, tc.CONSTRAINT_NAME,
                      tc.CONSTRAINT_TYPE,
                      (SELECT rc.UNIQUE_CONSTRAINT_NAME
                      FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
                      WHERE rc.CONSTRAINT_NAME=tc.CONSTRAINT_NAME)
                      R_CONSTRAINT_NAME,
                      (SELECT rc1.REFERENCED_TABLE_NAME
                      FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc1
                      WHERE rc1.CONSTRAINT_NAME=tc.CONSTRAINT_NAME)
                      R_TABLE_NAME
                      FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc
                      WHERE tc.CONSTRAINT_SCHEMA=schema()
                      AND tc.TABLE_NAME=v_tabela;
                DECLARE CONTINUE HANDLER FOR NOT FOUND SET h_restricao = TRUE;

                OPEN c_restricao;

                loop_restricao: LOOP
                      FETCH c_restricao INTO v_nome_tabela, v_nome_restricao,
                      v_tipo_restricao, v_restricao_ref, v_tabela_ref;
                      IF h_restricao THEN
                            CLOSE c_restricao;
                            LEAVE loop_restricao;
                      END IF;

                      if v_tipo_restricao='PRIMARY KEY' then
                            SET v_restricao = CONCAT(v_restricao,
                            v_nome_restricao, ' = ', v_nome_tabela, '(',
                            QV_ColunasDaRestricaoDaTabela(v_nome_tabela,
                            v_nome_restricao), ')', v_cr);
                      end if;

                END LOOP loop_restricao;
          END bloco2;

          SET v_retorno = CONCAT(v_retorno, v_restricao);

    END LOOP loop_tabela;

    CLOSE c_tabela;

    RETURN      v_retorno;
END bloco1
$$
```

14.14. A Função QV_TabelasPossuem_FK

```
/*================================================================================
Procedimento:     QV_TabelasPossuem_FK
Autor:            Henrique Figueiredo de Souza
Data Criação:     14/02/2013
Descrição:
          Lista todas as tabelas que possuem Chave Estrangeira e podem
          ou não possuir Chave Primária, ordenados por data de criação.
================================================================================*/
DELIMITER $$
CREATE FUNCTION QV_TabelasPossuem_FK()
RETURNS TEXT
```

```
bloco1: BEGIN
	DECLARE h_tabela BOOLEAN DEFAULT FALSE;
	DECLARE	v_restricao TEXT DEFAULT '';
	DECLARE v_cr CHAR(1) DEFAULT char(10);
	DECLARE v_retorno TEXT DEFAULT '';
	DECLARE v_tabela VARCHAR(64);
	DECLARE c_tabela CURSOR FOR
		select t.table_name
		from information_schema.tables t
		where t.table_schema=schema()
		and t.table_type='BASE TABLE'
		AND t.table_name in (SELECT DISTINCT tc.TABLE_NAME
		FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc
		WHERE tc.table_schema=schema()
		and tc.CONSTRAINT_TYPE='FOREIGN KEY');
	DECLARE CONTINUE HANDLER FOR NOT FOUND SET h_tabela = TRUE;

	OPEN c_tabela;

	loop_tabela: LOOP
		FETCH c_tabela INTO v_tabela;
		IF h_tabela THEN
			LEAVE loop_tabela;
		END IF;

		SET v_restricao = '';

		bloco2: BEGIN
			DECLARE h_restricao BOOLEAN DEFAULT FALSE;
			DECLARE v_nome_tabela VARCHAR(64);
			DECLARE v_nome_restricao VARCHAR(64);
			DECLARE v_tipo_restricao VARCHAR(64);
			DECLARE v_restricao_ref VARCHAR(64);
			DECLARE v_tabela_ref VARCHAR(64);
			DECLARE c_restricao CURSOR FOR
				SELECT tc.TABLE_NAME, tc.CONSTRAINT_NAME,
				tc.CONSTRAINT_TYPE,
				(SELECT rc.UNIQUE_CONSTRAINT_NAME
				FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
				WHERE rc.CONSTRAINT_NAME=tc.CONSTRAINT_NAME)
				R_CONSTRAINT_NAME,
				(SELECT rc1.REFERENCED_TABLE_NAME
				FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc1
				WHERE rc1.CONSTRAINT_NAME=tc.CONSTRAINT_NAME)
				R_TABLE_NAME
				FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc
				WHERE tc.CONSTRAINT_SCHEMA=schema()
				AND tc.TABLE_NAME=v_tabela;
			DECLARE CONTINUE HANDLER FOR NOT FOUND SET h_restricao = TRUE;

			OPEN c_restricao;

			loop_restricao: LOOP
				FETCH c_restricao INTO v_nome_tabela, v_nome_restricao,
				v_tipo_restricao, v_restricao_ref, v_tabela_ref;
				IF h_restricao THEN
					CLOSE c_restricao;
					LEAVE loop_restricao;
				END IF;

				if v_tipo_restricao='FOREIGN KEY' then
					SET v_restricao = CONCAT(v_restricao,
					v_nome_restricao, ' = ',
					v_nome_tabela, '(',
```

```
                              QV_ColunasDaRestricaoDaTabela(v_nome_tabela,
                              v_nome_restricao), ') --> ', v_tabela_ref, '(',
                              QV_ColunasDaRestricaoDaTabela(v_tabela_ref,
                              v_restricao_ref), ')', v_cr);
                        end if;

                  END LOOP loop_restricao;
            END bloco2;

            SET v_retorno = CONCAT(v_retorno, v_restricao);

      END LOOP loop_tabela;

      CLOSE c_tabela;

      RETURN      v_retorno;
END bloco1
$$
```

14.15. A Função QV_TabelasSomente_FK_naoPK

```
/*================================================================================
Procedimento:     QV_TabelasSomente_FK_naoPK
Autor:            Henrique Figueiredo de Souza
Data Criação:     14/02/2013
Descrição:
            Lista todas as tabelas que possuem SOMENTE Chave Estrangeira
            e não possuem Chave Primária, ordenados por data de criação.
================================================================================*/
DELIMITER $$
CREATE FUNCTION QV_TabelasSomente_FK_naoPK()
RETURNS TEXT
bloco1: BEGIN
      DECLARE h_tabela BOOLEAN DEFAULT FALSE;
      DECLARE     v_restricao TEXT DEFAULT '';
      DECLARE v_cr CHAR(1) DEFAULT char(10);
      DECLARE v_retorno TEXT DEFAULT '';
      DECLARE v_tabela VARCHAR(64);
      DECLARE c_tabela CURSOR FOR
            select t.table_name
            from (select t1.table_name
                  from information_schema.tables t1
                  where t1.table_schema=schema()
                  and t1.table_type='BASE TABLE'
                  AND t1.table_name not in (SELECT DISTINCT tc1.TABLE_NAME
                  FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
                  WHERE tc1.table_schema=schema()
                  and tc1.CONSTRAINT_TYPE='PRIMARY KEY')) t
            WHERE t.table_name in (SELECT DISTINCT tc1.TABLE_NAME
            FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc1
            WHERE tc1.table_schema=schema()
            and tc1.CONSTRAINT_TYPE='FOREIGN KEY');
      DECLARE CONTINUE HANDLER FOR NOT FOUND SET h_tabela = TRUE;

      OPEN c_tabela;

      loop_tabela: LOOP
            FETCH c_tabela INTO v_tabela;
            IF h_tabela THEN
                  LEAVE loop_tabela;
            END IF;

            SET v_restricao = '';
```

```
            bloco2: BEGIN
                  DECLARE h_restricao BOOLEAN DEFAULT FALSE;
                  DECLARE v_nome_tabela VARCHAR(64);
                  DECLARE v_nome_restricao VARCHAR(64);
                  DECLARE v_tipo_restricao VARCHAR(64);
                  DECLARE v_restricao_ref VARCHAR(64);
                  DECLARE v_tabela_ref VARCHAR(64);
                  DECLARE c_restricao CURSOR FOR
                        SELECT tc.TABLE_NAME, tc.CONSTRAINT_NAME,
                        tc.CONSTRAINT_TYPE,
                        (SELECT rc.UNIQUE_CONSTRAINT_NAME
                        FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
                        WHERE rc.CONSTRAINT_NAME=tc.CONSTRAINT_NAME)
                        R_CONSTRAINT_NAME,
                        (SELECT rc1.REFERENCED_TABLE_NAME
                        FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc1
                        WHERE rc1.CONSTRAINT_NAME=tc.CONSTRAINT_NAME)
                        R_TABLE_NAME
                        FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc
                        WHERE tc.table_schema=schema()
                        AND tc.TABLE_NAME=v_tabela;
                  DECLARE CONTINUE HANDLER FOR NOT FOUND SET h_restricao = TRUE;

                  OPEN c_restricao;

                  loop_restricao: LOOP
                        FETCH c_restricao INTO v_nome_tabela, v_nome_restricao,
                        v_tipo_restricao, v_restricao_ref, v_tabela_ref;
                        IF h_restricao THEN
                              CLOSE c_restricao;
                              LEAVE loop_restricao;
                        END IF;

                        if v_tipo_restricao='FOREIGN KEY' then
                              SET v_restricao = CONCAT(v_restricao,
                              v_nome_restricao, ' = ', v_nome_tabela, '(',
                              QV_ColunasDaRestricaoDaTabela(v_nome_tabela,
                              v_nome_restricao), ') --> ',
                              v_tabela_ref, '(',
                              QV_ColunasDaRestricaoDaTabela(v_tabela_ref,
                              v_restricao_ref), ')', v_cr);
                        end if;

                  END LOOP loop_restricao;
            END bloco2;

            SET v_retorno = CONCAT(v_retorno, v_restricao);

      END LOOP loop_tabela;

      CLOSE c_tabela;

      RETURN      v_retorno;
END bloco1
$$
```

14.16. A Função QV_DescomentarTabelasColunas

```
/*================================================================================
Procedimento:     QV_DescomentarTabelasColunas
Autor:            Henrique Figueiredo de Souza
Data Criação:     14/02/2013
Descrição:
            Descomentar todas a tabelas e colunas do banco de dados
```

```
================================================================================*/
DELIMITER $$
CREATE FUNCTION QV_DescomentarTabelasColunas()
RETURNS TEXT
BEGIN
	DECLARE h_tabela BOOLEAN DEFAULT FALSE;
	DECLARE	v_restricao TEXT DEFAULT '';
	DECLARE v_cr CHAR(1) DEFAULT char(10);
	DECLARE v_retorno TEXT DEFAULT '';
	DECLARE v_tabela_anterior VARCHAR(64);
	DECLARE v_coluna TEXT DEFAULT '';
	DECLARE v_conta integer DEFAULT 0;
	DECLARE v_nome_tabela VARCHAR(64);
	DECLARE v_nome_coluna VARCHAR(64);
	DECLARE v_tipo_coluna VARCHAR(64);
	DECLARE c_tabela CURSOR FOR
		SELECT c.TABLE_NAME, c.COLUMN_NAME, c.COLUMN_TYPE
		FROM INFORMATION_SCHEMA.COLUMNS c, information_schema.tables t
		WHERE C.TABLE_SCHEMA=schema() AND C.TABLE_SCHEMA=T.TABLE_SCHEMA
		AND c.TABLE_NAME=t.TABLE_NAME and t.table_type='BASE TABLE'
		ORDER BY c.TABLE_NAME, c.ORDINAL_POSITION;
	DECLARE CONTINUE HANDLER FOR NOT FOUND SET h_tabela = TRUE;

	OPEN c_tabela;

	loop_tabela: LOOP
		FETCH c_tabela INTO v_nome_tabela, v_nome_coluna, v_tipo_coluna;
		IF h_tabela THEN
			LEAVE loop_tabela;
		END IF;

		if v_tabela_anterior<>v_nome_tabela then

			if v_coluna is not null then
				SET v_retorno = CONCAT(v_retorno, v_coluna, v_cr);
			end if;

			SET v_coluna='';
			SET v_conta = 0;
		end if;

		if v_conta=0 then
			SET v_coluna = CONCAT('ALTER TABLE ', v_nome_tabela,
			' COMMENT ', char(39), char(39), ';', v_cr, v_coluna);
			SET v_conta = 1;
		end if;

		SET v_coluna = CONCAT(v_coluna, 'ALTER TABLE  ', v_nome_tabela,
		' MODIFY ', v_nome_coluna, ' ', v_tipo_coluna,
		' COMMENT ', char(39), char(39), ';', v_cr);

		SET v_tabela_anterior = v_nome_tabela;

	END LOOP loop_tabela;

	CLOSE c_tabela;

	SET v_retorno = CONCAT(v_retorno, v_coluna, v_cr);

	RETURN	v_retorno;
END
$$
```

15. Os Procedimentos Qlikview para o SYBASE SQL Anywhere

```
SELECT QV_ContaPalavra('ALFA BETA GAMA TETA');
SELECT QV_LimitaTamanhoTexto('ALFA BETA GAMA TETA', 2);
SELECT QV_AplicarMapa('pergunta','codigo_tipo_pergunta');
SELECT QV_ColunasDaRestricaoDaTabela('pergunta','PERGUNTA_TIPO_PERGUNTA_FK');
SELECT QV_ListarTabelaRestricao('pergunta');
SELECT QV_MontarLoad('pergunta','Henrique Figueiredo de Souza', 0, 0);

begin
DECLARE resultado bigint;
CALL QV_ContarLinhasTabela('TIPO_PERGUNTA', resultado);
SELECT resultado;
end;

begin
DECLARE resultado TEXT;
CALL QV_MontarTodosMappingLoad('Henrique Figueiredo de Souza', 0, 1, resultado);
SELECT resultado;
end;

begin
DECLARE resultado TEXT;
CALL QV_MontarTodosLoad('Henrique Figueiredo de Souza', 0, resultado);
SELECT resultado;
end;

begin
DECLARE resultado TEXT;
CALL QV_ListarTabelasColunas(1, resultado);
SELECT resultado;
end;

begin
DECLARE resultado TEXT;
CALL QV_ListarTabelasRestricoes(resultado);
SELECT resultado;
end;

begin
DECLARE resultado TEXT;
CALL QV_TabelasNaoPossuem_PK_nemFK(resultado);
SELECT resultado;
end;

begin
DECLARE resultado TEXT;
CALL QV_TabelasSomente_PK_naoFK(resultado);
SELECT resultado;
end;

begin
DECLARE resultado TEXT;
CALL QV_TabelasPossuem_FK(resultado);
SELECT resultado;
end;

begin
DECLARE resultado TEXT;
CALL QV_TabelasSomente_FK_naoPK(resultado);
SELECT resultado;
end;
```

```
begin
DECLARE resultado TEXT;
CALL QV_DescomentarTabelasColunas(resultado);
SELECT resultado;
end;

DROP FUNCTION QV_ContaPalavra;
DROP FUNCTION QV_LimitaTamanhoTexto;
DROP FUNCTION QV_AplicarMapa;
DROP FUNCTION QV_ColunasDaRestricaoDaTabela;
DROP FUNCTION QV_ListarTabelaRestricao;
DROP FUNCTION QV_MontarLoad;
DROP PROCEDURE QV_ContarLinhasTabela;
DROP PROCEDURE QV_MontarTodosMappingLoad;
DROP PROCEDURE QV_MontarTodosLoad;
DROP PROCEDURE QV_ListarTabelasColunas;
DROP PROCEDURE QV_ListarTabelasRestricoes;
DROP PROCEDURE QV_TabelasNaoPossuem_PK_nemFK;
DROP PROCEDURE QV_TabelasSomente_PK_naoFK;
DROP PROCEDURE QV_TabelasPossuem_FK;
DROP PROCEDURE QV_TabelasSomente_FK_naoPK;
DROP PROCEDURE QV_DescomentarTabelasColunas;
```

15.1. A Função QV_ContaPalavra

```
/*================================================================================
Procedimento:     QV_ContaPalavra
Autor:            Henrique Figueiredo de Souza
Data Criação:     21/02/2013
Parametros:
            p_str       Texto a ser contado as palavras.
Descrição:
            Conta as palavras de um texto.
================================================================================*/

CREATE FUNCTION QV_ContaPalavra(IN str VARCHAR(32676))
RETURNS BIGINT
BEGIN
      DECLARE i INTEGER DEFAULT 1;
      DECLARE palavras INTEGER DEFAULT 0;
      DECLARE tamanho INTEGER;
      DECLARE dentro_uma_palavra BIT;

      SET tamanho =  NULLIF(LENGTH(str),0);

      WHILE i <= (tamanho + 1) LOOP
            IF (ascii(substr(str, i, 1)) < 33) OR (i > tamanho) THEN
                  IF dentro_uma_palavra = 1 THEN
                        SET palavras = palavras + 1;
                        SET dentro_uma_palavra = 0;
                  ENDIF;
            ELSE
                  SET dentro_uma_palavra = 1;
            ENDIF;

            SET i = i + 1;
      END LOOP;

      RETURN palavras;
END;
```

15.2. A Função QV_LimitaTamanhoTexto

```
/*=============================================================================
Procedimento:     QV_LimitaTamanhoTexto
Autor:            Henrique Figueiredo de Souza
Data Criação:     21/02/2013
Parametros:
            p_str       Texto a ser limitado.
            p_limite    Limite de palavras por linhas.
Descrição:
            Limita o texto fornecido a uma determinada quantidades
            de palavras por linha.
=============================================================================*/

CREATE FUNCTION QV_LimitaTamanhoTexto(IN p_str TEXT, IN p_limite INTEGER)
RETURNS TEXT
BEGIN
      DECLARE i INTEGER DEFAULT 1;
      DECLARE v_palavras INTEGER DEFAULT 0;
      DECLARE v_tamanho INTEGER;
      DECLARE v_dentro_uma_palavra BIT;
      DECLARE v_texto TEXT;

    SET v_tamanho = NULLIF(LENGTH(p_str),0);

      SET v_texto = '';
      WHILE i <= (v_tamanho + 1) LOOP
            SET v_texto = v_texto || SUBSTR(p_str, i, 1);

            IF ASCII(SUBSTR(p_str, i, 1)) < 33 OR i > v_tamanho THEN
                  IF v_dentro_uma_palavra=1 THEN
                        SET v_palavras = v_palavras + 1;
                        SET v_dentro_uma_palavra = 0;

                        if v_palavras=p_limite then
                              SET v_texto = v_texto || char(13) || char(10);
                              SET v_palavras = 0;
                        end if;
                  END IF;
            ELSE
                  SET v_dentro_uma_palavra = 1;
            END IF;

            SET i = i + 1;
      END LOOP;

      RETURN v_texto;
END;
```

15.3. A Função QV_AplicarMapa

```
/*=============================================================================
Procedimento:     QV_AplicarMapa
Autor:            Henrique Figueiredo de Souza
Data Criação:     21/02/2013
Parametros:
            p_tabela    Nome da tabela referenciada pela FK.
            p_coluna    Nome da coluna que possui uma FK.
Descrição:
            Aplicar o mapa carregado, onde a coluna é uma
            Chave Estrangeira e tabela é a tabela referenciada pela
            chave estrangeira da coluna.
```

```
==========================================================================*/

CREATE FUNCTION QV_AplicarMapa(p_tabela varchar(128),
p_coluna varchar(128))
RETURNS TEXT
BEGIN
      DECLARE v_mapa TEXT;
      DECLARE v_tabela VARCHAR(128);
      DECLARE v_tabela_mapa VARCHAR(128);

      select pk_tab.table_name INTO v_tabela_mapa
      from SYS.SYSFKEY as fk
      join SYS.SYSTAB as fk_tab on fk_tab.table_id = fk.foreign_table_id
      join SYS.SYSUSER as fk_up on fk_up.user_id = fk_tab.creator
      join SYS.SYSTAB as pk_tab on pk_tab.table_id = fk.primary_table_id
      join SYS.SYSUSER as pk_up on pk_up.user_id = pk_tab.creator
      join SYS.SYSIDX as ix on ix.table_id = fk.foreign_table_id
      and ix.index_id = fk.foreign_index_id
      join SYS.SYSIDXCOL as fkc
      join SYS.SYSTABCOL as fk_col
      on(fkc.table_id = fk_col.table_id
      and fkc.column_id = fk_col.column_id)
      where fkc.table_id = fk.foreign_table_id
      and fkc.index_id = fk.foreign_index_id
      and fk_tab.table_name=p_tabela
      and fk_col.column_name=p_coluna;

      SELECT t.table_name INTO v_tabela
      FROM (SELECT t1.table_name, t1.object_id
            FROM SYS.SYSTAB t1
            where t1.creator=(SELECT user_id FROM SYS.SYSUSER
                  WHERE user_name=user_name())
            and t1.table_type_str='BASE'
            AND t1.object_id in (SELECT DISTINCT table_object_id
            from sys.SYSCONSTRAINT
            WHERE constraint_type IN ('P','F'))) t
      where t.object_id not in (SELECT DISTINCT table_object_id
      from sys.SYSCONSTRAINT
      WHERE constraint_type='F')
      AND t.table_name=v_tabela_mapa;

      SET v_mapa = '';

      /* verifica se a coluna da tabela é uma Chave Estrangeira */
      if (v_tabela_mapa is not null) and (length(v_tabela_mapa) > 0) then

            -- verifica se tabela possui somente Chave Primaria
            if (v_tabela is not null) and (length(v_tabela) > 0) then
                  SET v_mapa = 'ApplyMap(' || char(39) || 'Mapa' ||
                  v_tabela_mapa || char(39) || ',' || p_coluna || ')';
            end if;
      end if;

      RETURN v_mapa;
END;
```

15.4. A Função QV_ColunasDaRestricaoDaTabela

```
/*==========================================================================
Procedimento:     QV_ColunasDaRestricaoDaTabela
Autor:            Henrique Figueiredo de Souza
Data Criação:     21/02/2013
Parametros:
```

```
            p_tabela    Nome da tabela referenciada pela FK.
            p_restricao Nome da restrição.
Descrição:
            Retorna as colunas de uma tabela e sua restrição.
================================================================================*/

CREATE FUNCTION QV_ColunasDaRestricaoDaTabela(p_tabela varchar(128),
p_restricao varchar(128))
RETURNS TEXT
BEGIN
      DECLARE v_coluna TEXT;
      DECLARE v_nome_coluna varchar(128);
      DECLARE c_tabela CURSOR FOR
            SELECT tc.column_name
            from sys.SYSIDXCOL ic, SYS.SYSTAB t,
                  sys.SYSTABCOL tc, sys.SYSIDX i
            where ic.table_id=t.table_id
            and tc.table_id=t.table_id
            and ic.column_id=tc.column_id
            and t.creator=(SELECT user_id
                  FROM SYS.SYSUSER
                  WHERE user_name=user_name())
            and ic.index_id=i.index_id
            and i.table_id=t.table_id
            AND t.table_name=p_tabela
            and i.index_name=p_restricao;

      SET v_coluna = '';

      OPEN c_tabela;

      loop_tabela: LOOP
            FETCH c_tabela INTO v_nome_coluna;
            IF SQLCODE <> 0 THEN
                  LEAVE loop_tabela;
            END IF;

            SET v_coluna = v_coluna || ', ' || v_nome_coluna;

      END LOOP loop_tabela;

      CLOSE c_tabela;

      SET v_coluna = substr(v_coluna, 3, length(v_coluna));

      RETURN v_coluna;

END;
```

15.5. A Função QV_ListarTabelaRestricao

```
/*================================================================================
Procedimento:     QV_ListarTabelaRestricao
Autor:            Henrique Figueiredo de Souza
Data Criação:     21/02/2013
Parametros:
            p_tabela    Nome da tabela.
Descrição:
            Lista restrições de uma tabela.
================================================================================*/

CREATE FUNCTION QV_ListarTabelaRestricao(p_tabela varchar(128))
```

```
RETURNS TEXT
BEGIN
	DECLARE	v_restricao TEXT;
	DECLARE v_cr CHAR(2);
	DECLARE	v_tam INTEGER;
	DECLARE v_nome_tabela VARCHAR(128);
	DECLARE v_nome_restricao VARCHAR(128);
	DECLARE v_tipo_restricao VARCHAR(128);
	DECLARE v_restricao_ref VARCHAR(128);
	DECLARE v_tabela_ref VARCHAR(128);
	DECLARE c_tabela CURSOR FOR
		SELECT t.table_name,
		sc.constraint_name, sc.constraint_type,
		(SELECT c1.constraint_name
		from sys.SYSCONSTRAINT c1, SYS.SYSTAB t1
		WHERE t1.object_id=c1.table_object_id
		and c1.constraint_type='P'
		and t1.table_name=(SELECT k.primary_tname
		FROM SYS.SYSFOREIGNKEYS k
		WHERE k.primary_creator=user_name()
		and k.role=sc.constraint_name
		and k.foreign_tname=t.table_name)) as r_constraint_name,
		(SELECT fk.primary_tname
		FROM SYS.SYSFOREIGNKEYS fk
		WHERE fk.primary_creator=user_name()
		and fk.role=sc.constraint_name
		and fk.foreign_tname=t.table_name) as r_table_name
		from sys.SYSCONSTRAINT sc, SYS.SYSTAB t
		WHERE sc.constraint_type IN ('P','F')
		AND t.object_id=sc.table_object_id
		AND t.creator=(SELECT user_id
			FROM SYS.SYSUSER
			WHERE user_name=user_name())
		AND t.table_type_str='BASE'
		AND t.table_name=p_tabela
		ORDER BY t.table_name, sc.constraint_type DESC;

	SET v_restricao = '';
	SET v_cr = char(13) || char(10);

	OPEN c_tabela;

	loop_tabela: LOOP
		FETCH c_tabela INTO v_nome_tabela, v_nome_restricao,
		v_tipo_restricao, v_restricao_ref, v_tabela_ref;
		IF SQLCODE <> 0 THEN
			LEAVE loop_tabela;
		END IF;

		if v_tipo_restricao='P' then
			SET v_restricao = v_restricao || v_nome_tabela || '(' ||
			QV_ColunasDaRestricaoDaTabela(v_nome_tabela,
			v_nome_restricao) || ')' || v_cr;
		else
			SET v_restricao = v_restricao || v_nome_tabela || '(' ||
			QV_ColunasDaRestricaoDaTabela(v_nome_tabela,
			v_nome_restricao) || ') --> ' || v_tabela_ref || '(' ||
			QV_ColunasDaRestricaoDaTabela(v_tabela_ref,
			v_restricao_ref) || ')' || v_cr;
		end if;

	END LOOP loop_tabela;

	CLOSE c_tabela;
```

```
	SET v_tam = length(v_restricao)-2;

	if v_tam > 0 then
		SET v_restricao = substr(v_restricao, 1, v_tam);
	end if;

	RETURN v_restricao;
END;
```

15.6. A Função QV_MontarLoad

```
/*==============================================================================
Procedimento:	QV_MontarLoad
Autor:		Henrique Figueiredo de Souza
Data Criação:	17/02/2013
Parametros:
		p_tabela	Nome da tabela a gerar o LOAD.
		p_autor		Nome do autor a ser mostrado no comentário.
		p_mapeada	TRUE para mapping load, FALSE para sem mapping load
		p_2_colunas Limita a saida a somente a duas colunas.
Descrição:
		Monta um LOAD em formato qlikview de uma tabela
==============================================================================*/

CREATE FUNCTION QV_MontarLoad(p_tabela VARCHAR(128), p_autor VARCHAR(128),
p_mapeada BIT, p_2_colunas BIT)
RETURNS TEXT
BEGIN
	DECLARE v_tabela VARCHAR(128);
	DECLARE v_coluna TEXT DEFAULT '';
	DECLARE v_comenta VARCHAR(8000);
	DECLARE v_rotulo VARCHAR(4000);
	DECLARE v_rotulo_tabela TEXT;
	DECLARE v_mapa TEXT;
	DECLARE v_nomecoluna TEXT;
	DECLARE v_cr CHAR(2);
	DECLARE 	v_conta INTEGER DEFAULT 0;
	DECLARE 	v_retorno TEXT;

	DECLARE v_nome_tabela VARCHAR(128);
	DECLARE v_nome_coluna VARCHAR(128);
	DECLARE v_ordem_coluna INTEGER;
	DECLARE v_comentario VARCHAR(1024);
	DECLARE v_comentario_tabela VARCHAR(2048);

	DECLARE c_coluna CURSOR FOR
		SELECT t.table_name, tc.column_name, tc.column_id,
		(SELECT remarks FROM SYSREMARK r
		WHERE tc.object_id = r.object_id) as COLUMN_COMMENT,
		(SELECT remarks FROM SYSREMARK r
		WHERE t.object_id = r.object_id) as TABLE_COMMENT
		from SYS.SYSTAB t, sys.SYSTABCOL tc
		where tc.table_id=t.table_id
		and t.creator=(SELECT user_id FROM SYS.SYSUSER
			WHERE user_name=user_name())
		and t.table_name=p_tabela
		order by tc.column_id;

	SET v_retorno = '';
	SET v_cr = char(13) || char(10);
	OPEN c_coluna;
```

```
      loop_coluna: LOOP
            FETCH c_coluna INTO v_nome_tabela, v_nome_coluna,
            v_ordem_coluna, v_comentario, v_comentario_tabela;
            IF SQLCODE <> 0 THEN
                  LEAVE loop_coluna;
            END IF;

            SET v_tabela = v_nome_tabela;
            SET v_rotulo = v_comentario;
            SET v_rotulo_tabela = QV_LimitaTamanhoTexto(
                  NULLIF(v_comentario_tabela,''), 10);
            SET v_mapa = QV_AplicarMapa(v_tabela, v_nome_coluna);

            if length(v_mapa) > 0 then
                  SET v_nomecoluna = v_mapa;
            else
                  SET v_nomecoluna = v_nome_coluna;
            end if;

            if v_rotulo IS NULL then
                  SET v_coluna = v_coluna || ', ' || v_cr ||
                  '     ' || v_nomecoluna;
            else
                  SET v_coluna = v_coluna || ', ' || v_cr ||
                  '     ' || v_nomecoluna || ' as [' || v_rotulo || ']';
            end if;

            if p_2_colunas=1 then
                  SET v_conta = v_conta + 1;

                  if v_conta=2 then
                        LEAVE loop_coluna;
                  end if;
            end if;

      END LOOP loop_coluna;

      CLOSE c_coluna;

      if length(v_coluna) > 0 then

            SET v_coluna = substr(v_coluna, 3, length(v_coluna));

            SET v_comenta = v_cr ||

      '/*============================================================================
======' || v_cr;

            if p_mapeada=1 then
                  SET v_comenta = v_comenta || 'Procedimento:' || char(9) ||
                  'Carga mapeada em memória do arquivo QVD' || v_cr;
            else
                  SET v_comenta = v_comenta || 'Procedimento:' || char(9) ||
                  'Carga em memória do arquivo QVD' || v_cr;
            end if;
            SET v_comenta = v_comenta || 'ArquivoQVD:' ||
            char(9) || char(9) || v_tabela || v_cr;
            SET v_comenta = v_comenta || 'Autor:' || char(9) ||
            char(9) || char(9) || p_autor || v_cr;
            SET v_comenta = v_comenta || 'Data Criação:' || char(9) ||
                              DATEFORMAT(getdate(),'dd/mm/yyyy') || v_cr;

            if  length(v_rotulo_tabela) > 0 then
```

```
                  SET v_comenta = v_comenta || 'Descrição:' || char(9) ||
                  v_rotulo_tabela || v_cr;
            end if;

            SET v_retorno = v_retorno || v_comenta || 'Restrições:' || v_cr;

            SET v_retorno = v_retorno ||
            QV_ListarTabelaRestricao(p_tabela) || v_cr;

            SET v_retorno = v_retorno ||

      '================================================================================
====*/' || v_cr;

            if p_mapeada=1 then
                  SET v_retorno = v_retorno ||
                  'Mapa' || v_tabela || ':' || v_cr ||
                  'Mapping LOAD' || v_coluna || v_cr ||
                  'FROM' || v_cr || '$(vDiretorio)' ||
                  v_tabela || '.qvd' || v_cr || '(qvd);' || v_cr;
            else
                  SET v_retorno = v_retorno || v_tabela ||
                  ':' || v_cr || 'LOAD' || v_coluna || v_cr ||
                  'FROM' || v_cr || '$(vDiretorio)' ||
                  v_tabela || '.qvd' || v_cr || '(qvd);' || v_cr;
            end if;

      end if;

      RETURN      v_retorno;
END;
```

15.7. O Procedimento QV_ContarLinhasTabela

```
/*================================================================================
Procedimento:     QV_ContarLinhasTabela
Autor:            Henrique Figueiredo de Souza
Data Criação:     21/02/2013
Parametros:
            p_tabela    Nome da tabela referenciada pela FK.
Descrição:
            Retorna a quantidade de linhas de uma tabela.
================================================================================*/

CREATE PROCEDURE QV_ContarLinhasTabela(p_tabela VARCHAR(64),
OUT p_resultado BIGINT)
BEGIN
      DECLARE v_sql VARCHAR(200);

      SET v_sql = 'SELECT COUNT(*) FROM ' || p_tabela;
      BEGIN
            DECLARE c_tabela CURSOR USING v_sql;
            OPEN c_tabela;
            FETCH c_tabela INTO p_resultado;
            CLOSE c_tabela;
      END
END;
```

15.8. O Procedimento QV_MontarTodosMappingLoad

```
/*================================================================================
```

```
Procedimento:     QV_MontarTodosMappingLoad
Autor:            Henrique Figueiredo de Souza
Data Criação:     21/02/2013
Parametros:
            p_autor           Nome do autor a ser mostrado no comentário.
            p_vazia           Se 1 verificar se a tabela está vazia.
            p_2_colunas Limita a saida a somente a duas colunas.
Descrição:
            Monta todos os Mapping LOAD em formato qlikview
            de todas as tabelas que possuem SOMENTE Chave Primária.
================================================================================*/

CREATE PROCEDURE QV_MontarTodosMappingLoad(p_autor varchar(128),
p_vazia BIT, p_2_colunas BIT, OUT p_resultado TEXT)
BEGIN
      DECLARE v_tabela VARCHAR(128);
      DECLARE v_retorno TEXT DEFAULT '';
      DECLARE c_tabela CURSOR FOR
            SELECT t.table_name
            FROM (SELECT t1.table_name, t1.object_id
                  FROM SYS.SYSTAB t1
                  where t1.creator=(SELECT user_id FROM SYS.SYSUSER
                        WHERE user_name=user_name())
                  and t1.table_type_str='BASE'
                  AND t1.object_id in (SELECT DISTINCT table_object_id
                  from sys.SYSCONSTRAINT
                  WHERE constraint_type IN ('P','F'))) t
            where t.object_id not in (SELECT DISTINCT table_object_id
            from sys.SYSCONSTRAINT
            WHERE constraint_type='F');
      DECLARE v_sql VARCHAR(200);
      DECLARE v_conta BIGINT;

      SET v_retorno = '';
      OPEN c_tabela;

      loop_tabela: LOOP
            FETCH c_tabela INTO v_tabela;
            IF SQLCODE <> 0 THEN
                  LEAVE loop_tabela;
            END IF;

            if p_vazia = 1 then
                  SET v_sql = 'SELECT COUNT(*) FROM ' || v_tabela;
                  BEGIN
                        DECLARE c_conta CURSOR USING v_sql;
                        OPEN c_conta;
                        FETCH c_conta INTO v_conta;
                        CLOSE c_conta;
                  END;

                  if v_conta > 0 then
                        SET v_retorno = v_retorno ||
                        QV_MontarLoad(v_tabela, p_autor, 1, p_2_colunas);
                  end if;
            else
                  SET v_retorno = v_retorno ||
                  QV_MontarLoad(v_tabela, p_autor, 1, p_2_colunas);
            end if;
      END LOOP loop_tabela;

      CLOSE c_tabela;

      SET p_resultado = v_retorno;
```

```
END;
```

15.9. O Procedimento QV_MontarTodosLoad

```
/*================================================================================
Procedimento:     QV_MontarTodosLoad
Autor:            Henrique Figueiredo de Souza
Data Criação:     21/02/2013
Parametros:
            p_autor           Nome do autor a ser mostrado no comentário.
            p_vazia           Se TRUE verificar se a tabela está vazia.
Descrição:
            Monta todos os LOAD em formato qlikview de todas as tabelas
================================================================================*/

CREATE PROCEDURE QV_MontarTodosLoad(p_autor varchar(128),
p_vazia BIT, OUT p_resultado TEXT)
BEGIN
      DECLARE v_tabela VARCHAR(128);
      DECLARE v_retorno TEXT;
      DECLARE c_tabela CURSOR FOR
            SELECT t.table_name
            FROM SYS.SYSTAB t
            where t.creator=(SELECT user_id FROM SYS.SYSUSER
                  WHERE user_name=user_name())
            and t.table_type_str='BASE'
            ORDER BY t.TABLE_NAME;
      DECLARE v_sql VARCHAR(200);
      DECLARE v_conta BIGINT;

      SET v_retorno = '';
      OPEN c_tabela;

      loop_tabela: LOOP
            FETCH c_tabela INTO v_tabela;
            IF SQLCODE <> 0 THEN
                  LEAVE loop_tabela;
            END IF;

            if p_vazia = 1 then
                  SET v_sql = 'SELECT COUNT(*) FROM ' || v_tabela;
                  BEGIN
                        DECLARE c_conta CURSOR USING v_sql;
                        OPEN c_conta;
                        FETCH c_conta INTO v_conta;
                        CLOSE c_conta;
                  END;

                  if v_conta > 0 then
                        SET v_retorno = v_retorno ||
                        QV_MontarLoad(v_tabela, p_autor, 0, 0);
                  end if;
            else
                  SET v_retorno = v_retorno ||
                  QV_MontarLoad(v_tabela, p_autor, 0, 0);
            end if;

      END LOOP loop_tabela;

      CLOSE c_tabela;

      SET p_resultado = v_retorno;
```

```
END;
```

15.10. O Procedimento QV_ListarTabelasColunas

```
/*================================================================================
Procedimento:     QV_ListarTabelasColunas
Autor:            Henrique Figueiredo de Souza
Data Criação:     21/02/2013
Parametros:
            p_comentario      Se TRUE incluir comando  para gerar comentario.
Descrição:
            Lista todas a tabelas e colunas do banco de dados
================================================================================*/

CREATE PROCEDURE QV_ListarTabelasColunas(p_comentario BIT, OUT p_resultado TEXT)
BEGIN
      DECLARE     v_tabela_anterior VARCHAR(128) DEFAULT '';
      DECLARE     v_nome_tabela VARCHAR(128);
      DECLARE     v_nome_coluna VARCHAR(128);
      DECLARE     v_coluna TEXT;
      DECLARE v_cr CHAR(2);
      DECLARE v_conta integer DEFAULT 0;
      DECLARE v_retorno TEXT DEFAULT '';
      DECLARE c_tabela CURSOR FOR
            SELECT t.table_name, tc.column_name
            from SYS.SYSTAB t, sys.SYSTABCOL tc
            where tc.table_id=t.table_id
            and t.creator=(SELECT user_id FROM SYS.SYSUSER
                  WHERE user_name=user_name())
            and t.table_type_str='BASE'
            order by t.table_name, tc.column_id;

      SET v_cr = char(13) || char(10);
      OPEN c_tabela;

      loop_tabela: LOOP
            FETCH c_tabela INTO v_nome_tabela, v_nome_coluna;
            IF SQLCODE <> 0 THEN
                  LEAVE loop_tabela;
            END IF;

            if v_tabela_anterior<>v_nome_tabela then

                  if v_coluna is not null then
                        SET v_retorno = v_retorno || v_coluna || v_cr;
                  end if;

                  SET v_coluna='';
                  SET v_conta = 0;
            end if;

            if p_comentario=1 then

                  if v_conta=0 then
                        SET v_coluna = 'comment on table ' ||
                        v_nome_tabela || ' is '
                        || char(39) || char(9) || char(9) ||
                        char(39) || ';' || v_cr || v_coluna;
                        SET v_conta = 1;
                  end if;

                  SET v_coluna = v_coluna || 'comment on column ' ||
```

```
                  v_nome_tabela || '.' || v_nome_coluna
                  || ' is ' || char(39) || char(9) || char(9) ||
                  char(39) || ';' || v_cr;
            else
                  SET v_coluna = v_coluna || v_nome_tabela || '.'
                  || v_nome_coluna || v_cr;
            end if;

            SET v_tabela_anterior = v_nome_tabela;

      END LOOP loop_tabela;

      CLOSE c_tabela;

      SET p_resultado = v_retorno || v_coluna || v_cr;
END;
```

15.11. O Procedimento QV_ListarTabelasRestricoes

```
/*================================================================================
Procedimento:     QV_ListarTabelasRestricoes
Autor:            Henrique Figueiredo de Souza
Data Criação:     21/02/2013
Descrição:
            Lista todas as tabelas e suas restrições.
================================================================================*/

CREATE PROCEDURE QV_ListarTabelasRestricoes(OUT p_resultado TEXT)
BEGIN
      DECLARE v_tabela_anterior VARCHAR(61);
      DECLARE v_restricao TEXT DEFAULT '';
      DECLARE v_cr CHAR(2);
      DECLARE v_retorno TEXT DEFAULT '';
      DECLARE v_nome_tabela VARCHAR(64);
      DECLARE v_nome_restricao VARCHAR(64);
      DECLARE v_tipo_restricao VARCHAR(64);
      DECLARE v_restricao_ref VARCHAR(64);
      DECLARE v_tabela_ref VARCHAR(64);
      DECLARE c_tabela CURSOR FOR
            SELECT t.table_name,
            sc.constraint_name, sc.constraint_type,
            (SELECT c1.constraint_name
            from sys.SYSCONSTRAINT c1, SYS.SYSTAB t1
            WHERE t1.object_id=c1.table_object_id
            and c1.constraint_type='P'
            and t1.table_name=(SELECT k.primary_tname
            FROM SYS.SYSFOREIGNKEYS k
            WHERE k.primary_creator=user_name()
            and k.role=sc.constraint_name
            and k.foreign_tname=t.table_name)) as r_constraint_name,
            (SELECT fk.primary_tname
            FROM SYS.SYSFOREIGNKEYS fk
            WHERE fk.primary_creator=user_name()
            and fk.role=sc.constraint_name
            and fk.foreign_tname=t.table_name) as r_table_name
            from sys.SYSCONSTRAINT sc, SYS.SYSTAB t
            WHERE sc.constraint_type IN ('P','F')
            AND t.object_id=sc.table_object_id
            AND t.creator=(SELECT user_id
                  FROM SYS.SYSUSER
                  WHERE user_name=user_name())
            AND t.table_type_str='BASE'
```

```
            ORDER BY t.table_name, sc.constraint_type DESC;

      SET v_cr = char(13) || char(10);
      OPEN c_tabela;

      loop_tabela: LOOP
            FETCH c_tabela INTO v_nome_tabela, v_nome_restricao,
            v_tipo_restricao, v_restricao_ref, v_tabela_ref;
            IF SQLCODE <> 0 THEN
                  LEAVE loop_tabela;
            END IF;

            if v_tabela_anterior<>v_nome_tabela then

                  if v_restricao is not null then
                        SET v_retorno = v_retorno || v_restricao || v_cr;
                  end if;

                  SET v_restricao='';
            end if;

            if v_tipo_restricao='P' then
                  SET v_restricao = v_restricao || v_nome_restricao ||
                  ' = ' || v_nome_tabela || '(' ||
                  QV_ColunasDaRestricaoDaTabela(v_nome_tabela,
                  v_nome_restricao) || ')' || v_cr;
            else
                  SET v_restricao = v_restricao || v_nome_restricao ||
                  ' = ' || v_nome_tabela || '(' ||
                  QV_ColunasDaRestricaoDaTabela(v_nome_tabela,
                  v_nome_restricao) || ') --> ' ||
                  v_tabela_ref || '(' ||
                  QV_ColunasDaRestricaoDaTabela(v_tabela_ref,
                  v_restricao_ref) || ')' || v_cr;
            end if;

            SET v_tabela_anterior = v_nome_tabela;

      END LOOP loop_tabela;

      CLOSE c_tabela;

      SET p_resultado = v_retorno || v_restricao || v_cr;
END;
```

15.12. O Procedimento QV_TabelasNaoPossuem_PK_nemFK

```
/*================================================================================
Procedimento:     QV_TabelasNaoPossuem_PK_nemFK
Autor:            Henrique Figueiredo de Souza
Data Criação:     22/02/2013
Descrição:
            Lista todas as tabelas que NÃO possuem Chave Primária e
            não possuem Chave Estrangeira, ordenados por data de criação.
================================================================================*/

CREATE PROCEDURE QV_TabelasNaoPossuem_PK_nemFK(OUT p_resultado TEXT)
BEGIN
      DECLARE     v_tabela TEXT DEFAULT '';
      DECLARE     v_nome_tabela varchar(128);
      DECLARE v_cr CHAR(2);
      DECLARE c_tabela CURSOR FOR
```

```
            SELECT t.table_name
            FROM SYS.SYSTAB t
            where t.creator=(SELECT user_id FROM SYS.SYSUSER
                  WHERE user_name=user_name())
            and t.table_type_str='BASE'
            AND t.object_id not in (SELECT DISTINCT table_object_id
            from sys.SYSCONSTRAINT
            WHERE constraint_type IN ('P','F'));

      SET v_cr = char(13) || char(10);
      OPEN c_tabela;

      loop_tabela: LOOP
            FETCH c_tabela INTO v_nome_tabela;
            IF SQLCODE <> 0 THEN
                  LEAVE loop_tabela;
            END IF;

            SET v_tabela = v_tabela || v_nome_tabela || v_cr;

      END LOOP loop_tabela;

      CLOSE c_tabela;

      SET p_resultado = v_tabela;
END;
```

15.13. O Procedimento QV_TabelasSomente_PK_naoFK

```
/*================================================================================
Procedimento:     QV_TabelasSomente_PK_naoFK
Autor:            Henrique Figueiredo de Souza
Data Criação:     22/02/2013
Descrição:
            Lista todas as tabelas que possuem SOMENTE Chave Primária e
            não possuem Chave Estrangeira, ordenados por data de criação.
================================================================================*/

CREATE PROCEDURE QV_TabelasSomente_PK_naoFK(OUT p_resultado TEXT)
BEGIN
      DECLARE v_restricao TEXT DEFAULT '';
      DECLARE v_cr CHAR(2);

      SET v_cr = char(13) || char(10);

      FOR loop_objeto as c_objeto CURSOR FOR
            SELECT t.table_name as objeto_name
            FROM (SELECT t1.table_name,
                  t1.object_id, t1.last_modified_at
                  FROM SYS.SYSTAB t1
                  where t1.creator=(SELECT user_id
                        FROM SYS.SYSUSER
                        WHERE user_name=user_name())
                  and t1.table_type_str='BASE'
                  AND t1.object_id in (
                  SELECT DISTINCT table_object_id
                  from sys.SYSCONSTRAINT
                  WHERE constraint_type IN ('P','F'))) t
            where t.object_id not in (
            SELECT DISTINCT table_object_id
            from sys.SYSCONSTRAINT
            WHERE constraint_type='F')
```

```
            order by last_modified_at asc
      DO
            SET v_restricao = '';

            FOR loop_tabela as c_tabela CURSOR FOR
                  SELECT t.table_name,
                  sc.constraint_name, sc.constraint_type,
                  (SELECT fk.primary_tname
                  FROM SYS.SYSFOREIGNKEYS fk
                  WHERE fk.primary_creator=user_name()
                  and fk.role=sc.constraint_name
                  and fk.foreign_tname=t.table_name) as r_table_name
                  from sys.SYSCONSTRAINT sc, SYS.SYSTAB t
                  WHERE sc.constraint_type IN ('P','F')
                  AND t.object_id=sc.table_object_id
                  AND t.creator=(SELECT user_id
                        FROM SYS.SYSUSER
                        WHERE user_name=user_name())
                  AND t.table_type_str='BASE'
                  AND t.table_name=objeto_name
            DO
                  if constraint_type='P' then
                        SET v_restricao = v_restricao || CONSTRAINT_NAME || ' =
'
                        || table_name || '(' ||
                        QV_ColunasDaRestricaoDaTabela(table_name,
                        CONSTRAINT_NAME) || ')' || v_cr;
                  end if;

            END FOR;

            SET p_resultado = p_resultado || v_restricao;

      END FOR;

END;
```

15.14. O Procedimento QV_TabelasPossuem_FK

```
/*================================================================================
Procedimento:     QV_TabelasPossuem_FK
Autor:            Henrique Figueiredo de Souza
Data Criação:     22/02/2013
Descrição:
            Lista todas as tabelas que possuem Chave Estrangeira e podem
            ou não possuir Chave Primária, ordenados por data de criação.
================================================================================*/

CREATE PROCEDURE QV_TabelasPossuem_FK(OUT p_resultado TEXT)
BEGIN
      DECLARE v_restricao TEXT DEFAULT '';
      DECLARE v_cr CHAR(2);

      SET v_cr = char(13) || char(10);

      FOR loop_objeto as c_objeto CURSOR FOR
            SELECT t.table_name as objeto_nome, t.last_modified_at
            FROM SYS.SYSTAB t
            where t.creator=(SELECT user_id FROM SYS.SYSUSER
                  WHERE user_name=user_name())
            and t.table_type_str='BASE'
            AND t.object_id in (SELECT DISTINCT table_object_id
```

```
            from sys.SYSCONSTRAINT
            WHERE constraint_type='F')
            order by t.last_modified_at ASC
      DO
            SET v_restricao = '';

            FOR loop_tabela as c_tabela CURSOR FOR
                  SELECT t.table_name,
                  sc.constraint_name, sc.constraint_type,
                  (SELECT c1.constraint_name
                  from sys.SYSCONSTRAINT c1, SYS.SYSTAB t1
                  WHERE t1.object_id=c1.table_object_id
                  and c1.constraint_type='P'
                  and t1.table_name=(SELECT k.primary_tname
                  FROM SYS.SYSFOREIGNKEYS k
                  WHERE k.primary_creator=user_name()
                  and k.role=sc.constraint_name
                  and k.foreign_tname=t.table_name)) as r_constraint_name,
                  (SELECT fk.primary_tname
                  FROM SYS.SYSFOREIGNKEYS fk
                  WHERE fk.primary_creator=user_name()
                  and fk.role=sc.constraint_name
                  and fk.foreign_tname=t.table_name) as r_table_name
                  from sys.SYSCONSTRAINT sc, SYS.SYSTAB t
                  WHERE t.object_id=sc.table_object_id
                  AND t.creator=(SELECT user_id
                        FROM SYS.SYSUSER
                        WHERE user_name=user_name())
                  AND t.table_type_str='BASE'
                  AND t.table_name=objeto_nome
            DO
                  if CONSTRAINT_TYPE='F' then
                        SET v_restricao = v_restricao ||
                        CONSTRAINT_NAME || ' = '
                        || table_name || '(' ||
                        QV_ColunasDaRestricaoDaTabela(table_name,
                        CONSTRAINT_NAME) || ') --> ' ||
                        R_TABLE_NAME || '(' ||
                        QV_ColunasDaRestricaoDaTabela(R_TABLE_NAME,
                        R_CONSTRAINT_NAME) || ')' || v_cr;
                  end if;

            END FOR;

            SET p_resultado = p_resultado || v_restricao;

      END FOR;

END;
```

15.15. O Procedimento QV_TabelasSomente_FK_naoPK

```
/*================================================================================
Procedimento:     QV_TabelasSomente_FK_naoPK
Autor:            Henrique Figueiredo de Souza
Data Criação:     22/02/2013
Descrição:
            Lista todas as tabelas que possuem SOMENTE Chave Estrangeira
            e não possuem Chave Primária, ordenados por data de criação.
================================================================================*/

CREATE PROCEDURE QV_TabelasSomente_FK_naoPK(OUT p_resultado TEXT)
BEGIN
```

```
DECLARE v_restricao TEXT DEFAULT '';
DECLARE v_cr CHAR(2);

SET v_cr = char(13) || char(10);

FOR loop_objeto as c_objeto CURSOR FOR
      SELECT t.table_name as objeto_nome, t.last_modified_at
      FROM (SELECT t1.table_name, t1.object_id, t1.last_modified_at
            FROM SYS.SYSTAB t1
            where t1.creator=(SELECT user_id FROM SYS.SYSUSER
                  WHERE user_name=user_name())
            and t1.table_type_str='BASE'
            AND t1.object_id not in (SELECT DISTINCT table_object_id
            from sys.SYSCONSTRAINT
            WHERE constraint_type='P')) t
      where t.object_id in (SELECT DISTINCT table_object_id
      from sys.SYSCONSTRAINT
      WHERE constraint_type='F')
      order by t.last_modified_at ASC
DO
      SET v_restricao = '';

      FOR loop_tabela as c_tabela CURSOR FOR
            SELECT t.table_name,
            sc.constraint_name, sc.constraint_type,
            (SELECT c1.constraint_name
            from sys.SYSCONSTRAINT c1, SYS.SYSTAB t1
            WHERE t1.object_id=c1.table_object_id
            and c1.constraint_type='P'
            and t1.table_name=(SELECT k.primary_tname
            FROM SYS.SYSFOREIGNKEYS k
            WHERE k.primary_creator=user_name()
            and k.role=sc.constraint_name
            and k.foreign_tname=t.table_name)) as r_constraint_name,
            (SELECT fk.primary_tname
            FROM SYS.SYSFOREIGNKEYS fk
            WHERE fk.primary_creator=user_name()
            and fk.role=sc.constraint_name
            and fk.foreign_tname=t.table_name) as r_table_name
            from sys.SYSCONSTRAINT sc, SYS.SYSTAB t
            WHERE t.object_id=sc.table_object_id
            AND t.creator=(SELECT user_id
                  FROM SYS.SYSUSER
                  WHERE user_name=user_name())
            AND t.table_type_str='BASE'
            AND t.table_name=objeto_nome
      DO
            if CONSTRAINT_TYPE='F' then
                  SET v_restricao = v_restricao ||
                  CONSTRAINT_NAME || ' = '
                  || table_name || '(' ||
                  QV_ColunasDaRestricaoDaTabela(table_name,
                  CONSTRAINT_NAME) || ') --> ' ||
                  R_TABLE_NAME || '(' ||
                  QV_ColunasDaRestricaoDaTabela(R_TABLE_NAME,
                  R_CONSTRAINT_NAME) || ')' || v_cr;
            end if;

      END FOR;

      SET p_resultado = p_resultado || v_restricao;

END FOR;
```

```
END;
```

15.16. O Procedimento QV_DescomentarTabelasColunas

```
/*================================================================================
Procedimento:     QV_DescomentarTabelasColunas
Autor:            Henrique Figueiredo de Souza
Data Criação:     22/02/2013
Descrição:
            Descomentar todas a tabelas e colunas do banco de dados
================================================================================*/

CREATE PROCEDURE QV_DescomentarTabelasColunas(OUT p_resultado TEXT)
BEGIN
      DECLARE v_tabela_anterior VARCHAR(128) DEFAULT '';
      DECLARE v_coluna TEXT DEFAULT '';
      DECLARE v_cr CHAR(2);
      DECLARE v_conta integer DEFAULT 0;
      DECLARE v_retorno TEXT DEFAULT '';

      SET v_cr = char(13) || char(10);
      SET p_resultado = '';

      FOR loop_tabela as c_tabela CURSOR FOR
            SELECT t.table_name, tc.column_name
            from SYS.SYSTAB t, sys.SYSTABCOL tc
            where tc.table_id=t.table_id
            and t.creator=(SELECT user_id FROM SYS.SYSUSER
            WHERE user_name=user_name())
            order by t.table_name, tc.column_id
      DO
            if v_tabela_anterior<>table_name then

                  if v_coluna is not null then
                        SET v_retorno = v_retorno
                        || v_coluna || v_cr;
                  end if;

                  SET v_coluna='';
                  SET v_conta= 0;
            end if;

            if v_conta=0 then
                  SET v_coluna = 'comment on table ' ||
                  table_name || ' is null;' ||
                  v_cr || v_coluna;
                  SET v_conta = 1;
            end if;

            SET v_coluna = v_coluna || 'comment on column '
            || table_name || '.' || column_name
            || ' is null;' || v_cr;

            SET v_tabela_anterior = table_name;

      END FOR;

      SET p_resultado = v_retorno || v_coluna || v_cr;

END;
```

16. Os Procedimentos Qlikview para o SYBASE Adaptive Server Enterprise

```
sp_helpdb
go

sp_helpdb master
go

dump transaction master with truncate_only
go

sp_helpdevice master
go

sp_helpdevice
go

disk resize name = 'master', size = '1024M'
disk resize name = 'sybmgmtdev', size = '1024M'
disk resize name = 'sysprocsdev', size = '1024M'
disk resize name = 'systemdbdev', size = '1024M'
disk resize name = 'tempdbdev', size = '1024M'

alter database master on master = 2048
go

alter database tempdb on tempdbdev = 2048
go

alter database model on master = 2048
go

alter database sybmgmtdb on sybmgmtdev = 2048
go

alter database sybsystemdb on systemdbdev = 2048
go

alter database sybsystemprocs on sysprocsdev = 2048
go

alter database BANCO on master = 500
go

alter database BANCO on tempdbdev = 500
go

alter database BANCO on sybmgmtdev = 500
go

alter database BANCO on systemdbdev = 500
go

alter database BANCO on sysprocsdev = 500
go

sp_helpdb BANCO
```

```
SELECT dbo.QV_ContaPalavra('ALFA BETA GAMA TETA')
SELECT dbo.QV_LimitaTamanhoTexto('ALFA BETA GAMA TETA', 2)
SELECT dbo.QV_AplicarMapa('PERGUNTA','CODIGO_TIPO_PERGUNTA')
SELECT dbo.QV_ColunasDaRestricaoDaTabela('PERGUNTA','PERGUNTA_TIPO_PERGUNTA_FK')
EXEC dbo.QV_ListarTabelaRestricao 'PERGUNTA'
EXEC dbo.QV_MontarLoad 'PERGUNTA','Henrique Figueiredo de Souza', 0, 0

declare @texto varchar(16384)
set @texto = dbo.QV_LimitaTamanhoTexto('ALFA BETA GAMA TETA', 2)
print '%1!', @texto

SELECT dbo.QV_ContarLinhasTabela('PLANILHA_FUNCIONARIO')
EXEC dbo.QV_MontarTodosMappingLoad 'Henrique Figueiredo de Souza', 0, 1
EXEC dbo.QV_MontarTodosLoad 'Henrique Figueiredo de Souza', 1
EXEC dbo.QV_ListarTabelasColunas 1
EXEC dbo.QV_ListarTabelasRestricoes
EXEC dbo.QV_TabelasNaoPossuem_PK_nemFK
EXEC dbo.QV_TabelasSomente_PK_naoFK
EXEC dbo.QV_TabelasPossuem_FK
EXEC dbo.QV_TabelasSomente_FK_naoPK
EXEC dbo.QV_DescomentarTabelasColunas


DROP FUNCTION QV_ContaPalavra
DROP FUNCTION QV_LimitaTamanhoTexto
DROP FUNCTION QV_AplicarMapa
DROP FUNCTION QV_ColunasDaRestricaoDaTabela
DROP PROCEDURE QV_ListarTabelaRestricao
DROP PROCEDURE QV_MontarLoad
DROP FUNCTION QV_ContarLinhasTabela
DROP PROCEDURE QV_MontarTodosMappingLoad
DROP PROCEDURE QV_MontarTodosLoad
DROP PROCEDURE QV_ListarTabelasColunas
DROP PROCEDURE QV_ListarTabelasRestricoes
DROP PROCEDURE QV_TabelasNaoPossuem_PK_nemFK
DROP PROCEDURE QV_TabelasSomente_PK_naoFK
DROP PROCEDURE QV_TabelasPossuem_FK
DROP PROCEDURE QV_TabelasSomente_FK_naoPK
DROP PROCEDURE QV_DescomentarTabelasColunas
```

16.1. As Visões para auxiliar os procedimentos

```
DROP VIEW QV_TabelasChavesPrimarias
DROP VIEW QV_TabelasRestricoes
DROP VIEW QV_TabelasColunasRestricoes1
DROP VIEW QV_TabelasColunasRestricoes2
DROP VIEW QV_TabelasNao_PK_nemFK
DROP VIEW QV_TabelasPossuemSo_PK_naoFK
DROP VIEW QV_Tabelas_FK
DROP VIEW QV_TabelasPossuemSo_FK_naoPK


CREATE VIEW QV_TabelasChavesPrimarias
AS
select o.name table_name, i.name constraint_name, i.keycnt column_count,
index_col(o.name, i.indid, 1, o.uid) column_name, 1 ordem
from sysindexes i, sysobjects o
where i.id = o.id and i.indid > 0 and i.status2 & 2 = 2
and i.status & 2048 = 2048 and o.type = 'U' and i.keycnt>=1
union
select o.name table_name, i.name constraint_name, i.keycnt,
index_col(o.name, i.indid, 2, o.uid) column_name, 2
```

```
from sysindexes i, sysobjects o
where i.id = o.id and i.indid > 0 and i.status2 & 2 = 2
and i.status & 2048 = 2048 and o.type = 'U' and i.keycnt>=2
union
select o.name table_name, i.name constraint_name, i.keycnt,
index_col(o.name, i.indid, 3, o.uid) column_name, 3
from sysindexes i, sysobjects o
where i.id = o.id and i.indid > 0 and i.status2 & 2 = 2
and i.status & 2048 = 2048 and o.type = 'U' and i.keycnt>=3
union
select o.name table_name, i.name constraint_name, i.keycnt,
index_col(o.name, i.indid, 4, o.uid) column_name, 4
from sysindexes i, sysobjects o
where i.id = o.id and i.indid > 0 and i.status2 & 2 = 2
and i.status & 2048 = 2048 and o.type = 'U' and i.keycnt>=4
union
select o.name table_name, i.name constraint_name, i.keycnt,
index_col(o.name, i.indid, 5, o.uid) column_name, 5
from sysindexes i, sysobjects o
where i.id = o.id and i.indid > 0 and i.status2 & 2 = 2
and i.status & 2048 = 2048 and o.type = 'U' and i.keycnt>=5
union
select o.name table_name, i.name constraint_name, i.keycnt,
index_col(o.name, i.indid, 6, o.uid) column_name, 6
from sysindexes i, sysobjects o
where i.id = o.id and i.indid > 0 and i.status2 & 2 = 2
and i.status & 2048 = 2048 and o.type = 'U' and i.keycnt>=6
union
select o.name table_name, i.name constraint_name, i.keycnt,
index_col(o.name, i.indid, 7, o.uid) column_name, 7
from sysindexes i, sysobjects o
where i.id = o.id and i.indid > 0 and i.status2 & 2 = 2
and i.status & 2048 = 2048 and o.type = 'U' and i.keycnt>=7
union
select o.name table_name, i.name constraint_name, i.keycnt,
index_col(o.name, i.indid, 8, o.uid) column_name, 8
from sysindexes i, sysobjects o
where i.id = o.id and i.indid > 0 and i.status2 & 2 = 2
and i.status & 2048 = 2048 and o.type = 'U' and i.keycnt>=8
union
select o.name table_name, i.name constraint_name, i.keycnt,
index_col(o.name, i.indid, 9, o.uid) column_name, 9
from sysindexes i, sysobjects o
where i.id = o.id and i.indid > 0 and i.status2 & 2 = 2
and i.status & 2048 = 2048 and o.type = 'U' and i.keycnt>=9


CREATE VIEW QV_TabelasRestricoes
AS
SELECT so.name table_name,
i.name constraint_name,
'PRIMARY KEY' constraint_type,
null r_constraint_name,
null r_table_name
FROM sysobjects so, sysindexes i
WHERE so.id = i.id
AND so.type = 'U'
UNION
SELECT so.name table_name,
object_name(r.constrid) constraint_name,
'FOREIGN KEY' constraint_type,
(SELECT i.name FROM sysindexes i
WHERE i.id = r.reftabid) r_constraint_name,
object_name(r.reftabid) r_table_name
```

```
FROM sysobjects so, sysreferences r
WHERE so.id = r.tableid
AND so.type = 'U'
GO

CREATE VIEW QV_TabelasColunasRestricoes1
AS
SELECT object_name(r.tableid) table_name, c.name column_name,
object_name(r.constrid) constraint_name,
'FOREIGN KEY' constraint_type,
(SELECT i.name FROM sysindexes i WHERE i.id = r.reftabid) r_constraint_name,
object_name(r.reftabid) r_table_name
FROM   sysreferences r, syscolumns c
WHERE  c.id = r.tableid
AND c.colid IN (r.fokey1, r.fokey2, r.fokey3, r.fokey4,
r.fokey5, r.fokey6, r.fokey7, r.fokey8, r.fokey9,
r.fokey10, r.fokey11, r.fokey12, r.fokey13,
r.fokey14, r.fokey15, r.fokey16)
GO

CREATE VIEW QV_TabelasColunasRestricoes2
AS
SELECT object_name(r.tableid) table_name, object_name(r.constrid)
constraint_name,
'FOREIGN KEY' constraint_type,
(SELECT i.name FROM sysindexes i WHERE i.id = r.reftabid) r_constraint_name,
object_name(r.reftabid) r_table_name,
rc.name r_column_name
FROM   sysreferences r, syscolumns rc
WHERE  rc.id = r.reftabid
AND rc.colid IN (r.refkey1, r.refkey2, r.refkey3, r.refkey4,
r.refkey5, r.refkey6, r.refkey7, r.refkey8, r.refkey9,
r.refkey10, r.refkey11, r.refkey12, r.refkey13,
r.refkey14, r.refkey15, r.refkey16)
GO

/* LISTAR TODAS AS TABELAS QUE NÃO POSSUEM PK E NEM FK */
CREATE VIEW QV_TabelasNao_PK_nemFK
AS
SELECT t.name table_name
FROM sysobjects t
WHERE t.type = 'U'
AND t.name not in (SELECT DISTINCT table_name FROM QV_TabelasRestricoes)
GO

/* LISTAR TODAS AS TABELAS QUE POSSUEM SOMENTE PK E NÃO POSSUEM FK */
CREATE VIEW QV_TabelasPossuemSo_PK_naoFK
AS
SELECT t.table_name
FROM (SELECT DISTINCT table_name
    FROM QV_TabelasRestricoes
    WHERE constraint_type IN ('PRIMARY KEY','FOREIGN KEY')) t
WHERE t.table_name not in (SELECT DISTINCT table_name
  FROM QV_TabelasRestricoes
  WHERE constraint_type='FOREIGN KEY')
GO

/* LISTAR TODAS AS TABELAS QUE POSSUEM FK */
CREATE VIEW QV_Tabelas_FK
AS
SELECT DISTINCT table_name
  FROM QV_TabelasRestricoes
  WHERE constraint_type='FOREIGN KEY'
GO
```

```
/* LISTAR TODAS AS TABELAS QUE POSSUEM SOMENTE FK E NÃO POSSUEM PK */
CREATE VIEW QV_TabelasPossuemSo_FK_naoPK
AS
SELECT t.name table_name
FROM (SELECT t.name
      FROM sysobjects t
      WHERE t.type = 'U'
      AND t.name not in (SELECT DISTINCT table_name
      FROM QV_TabelasRestricoes
      WHERE constraint_type='PRIMARY KEY')) t
WHERE t.name in (SELECT DISTINCT table_name
  FROM QV_TabelasRestricoes
  WHERE constraint_type='FOREIGN KEY')
GO
```

16.2. A tabela de comentários para auxiliar os procedimentos

```
drop TABLE QV_Comentarios

CREATE TABLE QV_Comentarios(
id int not null,
colid smallint not null,
type char(1) not null,
comment varchar(4000) null,
CONSTRAINT QV_Comentarios_PK PRIMARY KEY(id, colid))

select comment
from QV_Comentarios
where id=1 and colid=0 and type='T'

select ('insert into QV_Comentarios(id, colid, type, comment) values(' +
rtrim(convert(char,so.id)) + ',0 ,' + char(39) + 'T' +
char(39) + ', ' + char(39) + ' ' + char(39) + ')') comando
from sysobjects so
where so.type='U'


select ('delete from QV_Comentarios where id=' +
rtrim(convert(char,so.id)) + ' and colid=0 and type=' +
char(39) + 'T' + char(39))  comando
from sysobjects so
where so.type='U'


select comment
from QV_Comentarios
where id=1 and colid=1 and type='C'


select ('insert into QV_Comentarios(id, colid, type, comment) values(' +
rtrim(convert(char,so.id)) + ',' + rtrim(convert(char,sc.colid))
+ ',' + char(39) + 'C' + char(39) + ', ' + char(39) + ' '
+ char(39) + ')') comando
from sysobjects so, syscolumns sc
WHERE sc.id = so.id
and so.type = 'U'


select ('delete from QV_Comentarios where id=' +
rtrim(convert(char,so.id)) + ' and colid=' +
rtrim(convert(char,sc.colid)) + ' and type=' +
```

```
char(39) + 'C' + char(39))  comando
from sysobjects so, syscolumns sc
WHERE sc.id = so.id
and so.type = 'U'
```

16.3. A Função QV_ContaPalavra

```
/*=====================================================================================
Procedimento:     QV_ContaPalavra
Autor:            Henrique Figueiredo de Souza
Data Criação:     23/02/2013
Parametros:
            p_str       Texto a ser contado as palavras.
Descrição:
            Conta as palavras de um texto.
=====================================================================================*/

CREATE FUNCTION QV_ContaPalavra(@str VARCHAR(16384))
RETURNS BIGINT AS
BEGIN
      DECLARE @i INTEGER
      DECLARE @palavras INTEGER
      DECLARE @tamanho INTEGER
      DECLARE @dentro_uma_palavra BIT

      SET @i = 1
      SET @palavras = 0
      SET @tamanho =  COALESCE(LEN(@str),0)

      WHILE @i <= (@tamanho + 1)
      BEGIN
            IF (ascii(substring(@str, @i, 1)) < 33) OR (@i > @tamanho)
            BEGIN
                  IF @dentro_uma_palavra = 1
                  BEGIN
                        SET @palavras = @palavras + 1
                        SET @dentro_uma_palavra = 0
                  END
            END
            ELSE
            BEGIN
                  SET @dentro_uma_palavra = 1
            END

            SET @i = @i + 1
      END

      RETURN @palavras
END
GO
```

16.4. A Função QV_LimitaTamanhoTexto

```
/*=====================================================================================
Procedimento:     QV_LimitaTamanhoTexto
Autor:            Henrique Figueiredo de Souza
Data Criação:     23/02/2013
Parametros:
            p_str       Texto a ser limitado.
            p_limite    Limite de palavras por linhas.
Descrição:
            Limita o texto fornecido a uma determinada quantidades
```

```
            de palavras por linha.
================================================================================*/

CREATE FUNCTION QV_LimitaTamanhoTexto(@p_str VARCHAR(16384), @p_limite INTEGER)
RETURNS VARCHAR(16384) AS
BEGIN
      DECLARE @v_i INTEGER
      DECLARE @v_palavras INTEGER
      DECLARE @v_tamanho INTEGER
      DECLARE @v_dentro_uma_palavra BIT
      DECLARE @v_texto VARCHAR(16384)

      SET @v_i = 1
      SET @v_palavras = 0
      SET @v_tamanho =  COALESCE(LEN(@p_str),0)

      WHILE @v_i <= (@v_tamanho + 1)
      BEGIN
            SET @v_texto = @v_texto + SUBSTRING(@p_str, @v_i, 1)

            IF ASCII(SUBSTRING(@p_str, @v_i, 1)) < 33 OR @v_i > @v_tamanho
            BEGIN
                  IF @v_dentro_uma_palavra=1
                  BEGIN
                        SET @v_palavras = @v_palavras + 1
                        SET @v_dentro_uma_palavra = 0

                        if @v_palavras=@p_limite
                        BEGIN
                              SET @v_texto = @v_texto + char(13) + char(10)
                              SET @v_palavras = 0
                        END
                  END
            END
            ELSE
            BEGIN
                  SET @v_dentro_uma_palavra = 1
            END

            SET @v_i = @v_i + 1
      END

      RETURN @v_texto
END
GO
```

16.5. A Função QV_AplicarMapa

```
/*================================================================================
Procedimento:     QV_AplicarMapa
Autor:            Henrique Figueiredo de Souza
Data Criação:     23/02/2013
Parametros:
            p_tabela    Nome da tabela referenciada pela FK.
            p_coluna    Nome da coluna que possui uma FK.
Descrição:
            Aplicar o mapa carregado, onde a coluna é uma
            Chave Estrangeira e tabela é a tabela referenciada pela
            chave estrangeira da coluna.
================================================================================*/

CREATE FUNCTION QV_AplicarMapa(@p_tabela varchar(255),
```

```
@p_coluna varchar(255))
RETURNS VARCHAR(16384) AS
BEGIN
      DECLARE @v_mapa VARCHAR(16384)
      DECLARE @v_tabela VARCHAR(255)
      DECLARE @v_tabela_mapa VARCHAR(255)

      SELECT @v_tabela_mapa=object_name(r.reftabid)
      FROM sysreferences r, syscolumns c
      WHERE c.id = r.tableid
      AND c.colid IN (r.fokey1, r.fokey2, r.fokey3, r.fokey4,
      r.fokey5, r.fokey6, r.fokey7, r.fokey8, r.fokey9,
      r.fokey10, r.fokey11, r.fokey12, r.fokey13,
      r.fokey14, r.fokey15, r.fokey16)
      AND object_name(r.tableid)=@p_tabela
      AND c.name=@p_coluna

      SELECT @v_tabela=t.table_name
      FROM (SELECT DISTINCT table_name
          FROM QV_TabelasRestricoes
          WHERE constraint_type IN ('PRIMARY KEY','FOREIGN KEY')) t
      WHERE t.table_name not in (SELECT DISTINCT table_name
        FROM QV_TabelasRestricoes
        WHERE constraint_type='FOREIGN KEY')
      AND t.table_name=@v_tabela_mapa

      /* verifica se a coluna da tabela é uma Chave Estrangeira */
      if (@v_tabela_mapa is not null) and (len(@v_tabela_mapa) > 0)
      begin
            -- verifica se tabela possui somente Chave Primaria
            if (@v_tabela is not null) and (len(@v_tabela) > 0)
            begin
                  SET @v_mapa = 'ApplyMap(' + char(39) + 'Mapa' +
                  @v_tabela_mapa + char(39) + ',' + @p_coluna + ')'
            end
      end

      RETURN @v_mapa
END
GO
```

16.6. A Função QV_ColunasDaRestricaoDaTabela

```
/*==============================================================================
Procedimento:     QV_ColunasDaRestricaoDaTabela
Autor:            Henrique Figueiredo de Souza
Data Criação:     23/02/2013
Parametros:
            p_tabela    Nome da tabela referenciada pela FK.
            p_restricao Nome da restrição.
Descrição:
            Retorna as colunas de uma tabela e sua restrição.
==============================================================================*/

CREATE FUNCTION QV_ColunasDaRestricaoDaTabela(@p_tabela varchar(255),
@p_restricao varchar(255))
RETURNS VARCHAR(16384) AS
BEGIN
      DECLARE @v_coluna VARCHAR(16384)
      DECLARE @v_nome_coluna varchar(255)
      DECLARE c_tabela CURSOR FOR
              SELECT column_name
```

```
            FROM QV_TabelasChavesPrimarias
          WHERE table_name=@p_tabela
          AND constraint_name=@p_restricao
            UNION
          SELECT c.name column_name
          FROM sysreferences r, syscolumns c
          WHERE c.id = r.tableid
          AND c.colid IN (r.fokey1, r.fokey2, r.fokey3, r.fokey4,
          r.fokey5, r.fokey6, r.fokey7, r.fokey8, r.fokey9,
          r.fokey10, r.fokey11, r.fokey12, r.fokey13,
          r.fokey14, r.fokey15, r.fokey16)
          AND object_name(r.tableid)=@p_tabela
          AND object_name(r.constrid)=@p_restricao
          UNION
          SELECT rc.name r_column_name
          FROM   sysreferences r, syscolumns rc
          WHERE  rc.id = r.reftabid
          AND rc.colid IN (r.refkey1, r.refkey2, r.refkey3, r.refkey4,
          r.refkey5, r.refkey6, r.refkey7, r.refkey8, r.refkey9,
          r.refkey10, r.refkey11, r.refkey12, r.refkey13,
          r.refkey14, r.refkey15, r.refkey16)
          AND object_name(r.reftabid)=@p_tabela
          AND (SELECT i.name FROM sysindexes i
          WHERE i.id = r.reftabid)=@p_restricao

    OPEN c_tabela
    FETCH c_tabela INTO @v_nome_coluna

    WHILE (@@sqlstatus !=2) AND (@@sqlstatus != 1)
    BEGIN
          SET @v_coluna = @v_coluna + ', ' + @v_nome_coluna

          FETCH c_tabela INTO @v_nome_coluna
    END

    CLOSE c_tabela
    DEALLOCATE CURSOR c_tabela

    SET @v_coluna = substring(@v_coluna, 3, len(@v_coluna))

    RETURN @v_coluna

END
GO
```

16.7. O Procedimento QV_ListarTabelaRestricao

```
/*=============================================================================
Procedimento:     QV_ListarTabelaRestricao
Autor:            Henrique Figueiredo de Souza
Data Criação:     23/02/2013
Parametros:
            p_tabela    Nome da tabela.
Descrição:
            Lista restrições de uma tabela.
=============================================================================*/

CREATE PROCEDURE QV_ListarTabelaRestricao(@p_tabela varchar(255))
AS
BEGIN
      DECLARE     @v_restricao VARCHAR(16384)
      DECLARE @v_cr CHAR(2)
```

```
DECLARE     @v_tam INTEGER
DECLARE @v_nome_tabela VARCHAR(255)
DECLARE @v_nome_restricao VARCHAR(255)
DECLARE @v_tipo_restricao VARCHAR(255)
DECLARE @v_restricao_ref VARCHAR(255)
DECLARE @v_tabela_ref VARCHAR(255)
DECLARE c_tabela CURSOR FOR
      SELECT so.name table_name,
      i.name constraint_name,
      'PRIMARY KEY' constraint_type,
      null r_constraint_name,
      null r_table_name
      FROM sysobjects so, sysindexes i
      WHERE so.id = i.id
      AND so.type = 'U'
      AND so.name=@p_tabela
      UNION
      SELECT so.name table_name,
      object_name(r.constrid) constraint_name,
      'FOREIGN KEY' constraint_type,
      (SELECT i.name FROM sysindexes i
      WHERE i.id = r.reftabid) r_constraint_name,
      object_name(r.reftabid) r_table_name
      FROM sysobjects so, sysreferences r
      WHERE so.id = r.tableid
      AND so.type = 'U'
      AND so.name=@p_tabela
      ORDER BY so.name, 3 DESC

SET @v_cr = char(13) + char(10)

OPEN c_tabela
FETCH c_tabela INTO @v_nome_tabela, @v_nome_restricao,
@v_tipo_restricao, @v_restricao_ref, @v_tabela_ref

WHILE (@@sqlstatus !=2) AND (@@sqlstatus != 1)
BEGIN
      if @v_tipo_restricao='PRIMARY KEY'
      begin
            SET @v_restricao = @v_restricao + @v_nome_tabela + '(' +
            dbo.QV_ColunasDaRestricaoDaTabela(@v_nome_tabela,
            @v_nome_restricao) + ')' + @v_cr
      end
      else
      begin
            SET @v_restricao = @v_restricao + @v_nome_tabela + '(' +
            dbo.QV_ColunasDaRestricaoDaTabela(@v_nome_tabela,
            @v_nome_restricao) + ') --> ' + @v_tabela_ref + '(' +
            dbo.QV_ColunasDaRestricaoDaTabela(@v_tabela_ref,
            @v_restricao_ref) + ')' + @v_cr
      end

      FETCH c_tabela INTO @v_nome_tabela, @v_nome_restricao,
      @v_tipo_restricao, @v_restricao_ref, @v_tabela_ref
END

CLOSE c_tabela
DEALLOCATE CURSOR c_tabela

SET @v_tam = len(@v_restricao)-2

if @v_tam > 0
begin
      SET @v_restricao = substring(@v_restricao, 1, @v_tam)
```

```
	end

	PRINT '%1!', @v_restricao
END
GO
```

16.8. O Procedimento QV_MontarLoad

```
/*==============================================================================
Procedimento:	QV_MontarLoad
Autor:		Henrique Figueiredo de Souza
Data Criação:	24/02/2013
Parametros:
		p_tabela	Nome da tabela a gerar o LOAD.
		p_autor		Nome do autor a ser mostrado no comentário.
		p_mapeada	TRUE para mapping load, FALSE para sem mapping load
		p_2_colunas Limita a saida a somente a duas colunas.
Descrição:
		Monta um LOAD em formato qlikview de uma tabela
==============================================================================*/

CREATE PROCEDURE QV_MontarLoad(@p_tabela VARCHAR(255), @p_autor VARCHAR(128),
@p_mapeada BIT, @p_2_colunas BIT)
AS
BEGIN
	DECLARE @v_temp VARCHAR(16384)
	DECLARE @v_tabela VARCHAR(255)
	DECLARE @v_coluna VARCHAR(16384)
	DECLARE @v_comenta VARCHAR(8000)
	DECLARE @v_rotulo VARCHAR(4000)
	DECLARE @v_rotulo_tabela VARCHAR(16384)
	DECLARE @v_mapa VARCHAR(16384)
	DECLARE @v_nomecoluna VARCHAR(16384)
	DECLARE @v_cr CHAR(2)
	DECLARE	@v_conta INTEGER

	DECLARE @v_nome_tabela VARCHAR(255)
	DECLARE @v_nome_coluna VARCHAR(255)
	DECLARE @v_ordem_coluna INTEGER
	DECLARE @v_comentario VARCHAR(1024)
	DECLARE @v_comentario_tabela VARCHAR(2048)

	DECLARE c_coluna CURSOR FOR
		SELECT so.name as table_name,
		sc.name as column_name, sc.colid as column_id,
		(select comment
		from QV_Comentarios
		where id=so.id and colid=sc.colid
		and type='C') as COLUMN_COMMENT,
		(select comment
		from QV_Comentarios
		where id=so.id and colid=0
		and type='T') as TABLE_COMMENT
		FROM sysobjects so, syscolumns sc
		WHERE sc.id = so.id
		and so.type = 'U'
		AND so.name=@p_tabela
		order by so.name, sc.colid

	SET @v_conta = 0
	SET @v_cr = char(13) + char(10)
```

```
	OPEN c_coluna
	FETCH c_coluna INTO @v_nome_tabela, @v_nome_coluna,
	@v_ordem_coluna, @v_comentario, @v_comentario_tabela

	WHILE (@@sqlstatus !=2) AND (@@sqlstatus != 1)
	BEGIN
		SET @v_tabela = @v_nome_tabela
		SET @v_rotulo = @v_comentario
		SET @v_rotulo_tabela =
dbo.QV_LimitaTamanhoTexto(COALESCE(@v_comentario_tabela,''), 10)
		SET @v_mapa = dbo.QV_AplicarMapa(@v_tabela, @v_nome_coluna)

		if @v_mapa is not null
		begin
			SET @v_nomecoluna = @v_mapa
		end
		else
		begin
			SET @v_nomecoluna = @v_nome_coluna
		end

		if @v_rotulo is null
		begin
			SET @v_coluna = @v_coluna + ', ' + @v_cr +
			'	' + @v_nomecoluna
		end
		else
		begin
			SET @v_coluna = @v_coluna + ', ' + @v_cr +
			'	' + @v_nomecoluna + ' as [' + @v_rotulo + ']'
		end

		if @p_2_colunas=1
		begin
			SET @v_conta = @v_conta + 1

			if @v_conta=2
			begin
				BREAK
			end
		end

		FETCH c_coluna INTO @v_nome_tabela, @v_nome_coluna,
		@v_ordem_coluna, @v_comentario, @v_comentario_tabela
	END

	CLOSE c_coluna
	DEALLOCATE CURSOR c_coluna

	if len(@v_coluna) > 0
	begin

		SET @v_coluna = substring(@v_coluna, 3, len(@v_coluna))

		SET @v_comenta = @v_cr +

	'/*============================================================================
======' + @v_cr

		if @p_mapeada=1
		begin
			SET @v_comenta = @v_comenta + 'Procedimento:' + char(9) +
			'Carga mapeada em memória do arquivo QVD' + @v_cr
		end
```

```
            else
            begin
                  SET @v_comenta = @v_comenta + 'Procedimento:' + char(9) +
                  'Carga em memória do arquivo QVD' + @v_cr
            end
            SET @v_comenta = @v_comenta + 'ArquivoQVD:' +
            char(9) + char(9) + @v_tabela + @v_cr
            SET @v_comenta = @v_comenta + 'Autor:' +
            char(9) + char(9) + char(9) + @p_autor + @v_cr
            SET @v_comenta = @v_comenta + 'Data Criação:' +
            char(9) + convert(char, getdate(), 103) + @v_cr

            if  @v_rotulo_tabela<>''
            begin
                  SET @v_comenta = @v_comenta + 'Descrição:' + char(9) +
                  @v_rotulo_tabela + @v_cr
            end

            PRINT '%1!%2!%3!', @v_comenta, 'Restrições:', @v_cr

            EXECUTE dbo.QV_ListarTabelaRestricao @p_tabela

            PRINT '%1!%2!',

      '================================================================================
====*/', @v_cr

            if @p_mapeada=1
            begin
                  set @v_temp =
                  'Mapa' + @v_tabela + ':' + @v_cr +
                  'Mapping LOAD' + @v_coluna + @v_cr +
                  'FROM' + @v_cr + '$(vDiretorio)' +
                  @v_tabela + '.qvd' + @v_cr + '(qvd)' + @v_cr
            end
            else
            begin
                  set @v_temp =  @v_tabela +
                  ':' + @v_cr + 'LOAD' + @v_coluna + @v_cr +
                  'FROM' + @v_cr + '$(vDiretorio)' +
                  @v_tabela + '.qvd' + @v_cr + '(qvd)' + @v_cr
            end

            PRINT '%1!', @v_temp
      end

END
GO
```

16.9. A Função QV_ContarLinhasTabela

```
/*================================================================================
Procedimento:     QV_ContarLinhasTabela
Autor:            Henrique Figueiredo de Souza
Data Criação:     24/02/2013
Parametros:
            p_tabela    Nome da tabela referenciada pela FK.
Descrição:
            Retorna a quantidade de linhas de uma tabela.
================================================================================*/

CREATE FUNCTION QV_ContarLinhasTabela(@p_tabela VARCHAR(255))
```

```
RETURNS BIGINT AS
BEGIN
      declare @v_conta BIGINT
      declare @v_sql varchar(200)

      set @v_sql = 'SELECT @v_conta=COUNT(*) FROM ' + @p_tabela
      execute(@v_sql)

      return @v_conta
END
GO
```

16.10. O Procedimento QV_MontarTodosMappingLoad

```
/*==============================================================================
Procedimento:     QV_MontarTodosMappingLoad
Autor:            Henrique Figueiredo de Souza
Data Criação:     24/02/2013
Parametros:
            p_autor          Nome do autor a ser mostrado no comentário.
            p_vazia          Se 1 verificar se a tabela está vazia.
            p_2_colunas Limita a saida a somente a duas colunas.
Descrição:
            Monta todos os Mapping LOAD em formato qlikview
            de todas as tabelas que possuem SOMENTE Chave Primária.
==============================================================================*/

CREATE PROCEDURE QV_MontarTodosMappingLoad(@p_autor varchar(128),
@p_vazia BIT, @p_2_colunas BIT)
AS
BEGIN
      DECLARE @v_sql VARCHAR(200)
      DECLARE @v_conta INTEGER
      DECLARE @v_tabela VARCHAR(255)
      DECLARE c_tabela CURSOR FOR
            SELECT t.table_name
            FROM (SELECT DISTINCT table_name
                FROM QV_TabelasRestricoes
                WHERE constraint_type IN ('PRIMARY KEY','FOREIGN KEY')) t
            WHERE t.table_name not in (SELECT DISTINCT table_name
              FROM QV_TabelasRestricoes
              WHERE constraint_type='FOREIGN KEY')

      SET @v_conta = 0

      OPEN c_tabela
      FETCH c_tabela INTO @v_tabela

      WHILE (@@sqlstatus !=2) AND (@@sqlstatus != 1)
      BEGIN
            if @p_vazia = 1
            begin
                  SET @v_sql = 'SELECT @v_conta=COUNT(*) FROM ' +
                                    @v_tabela
                  EXECUTE(@v_sql)

                  if @v_conta > 0
                  begin
                        EXECUTE dbo.QV_MontarLoad @v_tabela, @p_autor, 1,
@p_2_colunas
                  end
            end
```

```
            else
            begin
                  EXECUTE dbo.QV_MontarLoad @v_tabela, @p_autor, 1, @p_2_colunas
            end

            FETCH c_tabela INTO @v_tabela
      END

      CLOSE c_tabela
      DEALLOCATE CURSOR c_tabela
END
GO
```

16.11. O Procedimento QV_MontarTodosLoad

```
/*================================================================================
Procedimento:     QV_MontarTodosLoad
Autor:            Henrique Figueiredo de Souza
Data Criação:     24/02/2013
Parametros:
            p_autor           Nome do autor a ser mostrado no comentário.
            p_vazia           Se TRUE verificar se a tabela está vazia.
Descrição:
            Monta todos os LOAD em formato qlikview de todas as tabelas
================================================================================*/

CREATE PROCEDURE QV_MontarTodosLoad(@p_autor varchar(128),
@p_vazia BIT)
AS
BEGIN
      DECLARE @v_sql VARCHAR(200)
      DECLARE @v_conta INTEGER
      DECLARE @v_tabela VARCHAR(255)
      DECLARE c_tabela CURSOR FOR
            SELECT so.name as table_name
            FROM sysobjects so
            WHERE so.type = 'U'
            order by so.name

      SET @v_conta = 0

      OPEN c_tabela
      FETCH c_tabela INTO @v_tabela

      WHILE (@@sqlstatus !=2) AND (@@sqlstatus != 1)
      BEGIN
            if @p_vazia = 1
            begin
                  SET @v_sql = 'SELECT @v_conta=COUNT(*) FROM ' +
                                    @v_tabela
                  EXECUTE(@v_sql)

                  if @v_conta > 0
                  begin
                        EXECUTE dbo.QV_MontarLoad @v_tabela,
                                    @p_autor, 0, 0
                  end
            end
            else
            begin
                  EXECUTE dbo.QV_MontarLoad @v_tabela, @p_autor, 0, 0
            end
```

```
            FETCH c_tabela INTO @v_tabela
      END

      CLOSE c_tabela
      DEALLOCATE CURSOR c_tabela
END
GO
```

16.12. O Procedimento QV_ListarTabelasColunas

```
/*=============================================================================
Procedimento:     QV_ListarTabelasColunas
Autor:            Henrique Figueiredo de Souza
Data Criação:     24/02/2013
Parametros:
            p_comentario      Se TRUE incluir comando  para gerar comentario.
Descrição:
            Lista todas a tabelas e colunas do banco de dados
=============================================================================*/

CREATE PROCEDURE QV_ListarTabelasColunas(@p_comentario BIT)
AS
BEGIN
      DECLARE     @v_tabela_anterior VARCHAR(255)
      DECLARE     @v_nome_tabela VARCHAR(255)
      DECLARE     @v_nome_coluna VARCHAR(255)
      DECLARE @v_id int
      DECLARE @v_colid smallint
      DECLARE     @v_coluna VARCHAR(16384)
      DECLARE @v_cr CHAR(2)
      DECLARE @v_conta integer
      DECLARE c_tabela CURSOR FOR
            SELECT so.name as table_name,
            sc.name as column_name,
            so.id, sc.colid
            FROM sysobjects so, syscolumns sc
            WHERE sc.id = so.id
            and so.type = 'U'
            order by so.name, sc.colid

      SET @v_conta = 0
      SET @v_cr = char(13) + char(10)

      OPEN c_tabela
      FETCH c_tabela INTO @v_nome_tabela, @v_nome_coluna,
            @v_id,  @v_colid

      WHILE (@@sqlstatus !=2) AND (@@sqlstatus != 1)
      BEGIN
            if @v_tabela_anterior<>@v_nome_tabela
            begin
                  if @v_coluna is not null
                  begin
                        PRINT '%1!%2!', @v_coluna, @v_cr
                  end

                  SET @v_coluna = null
                  SET @v_conta = 0
            end

            if @p_comentario=1
```

```
            begin
                  if @v_conta=0
                  begin
                        SET @v_coluna =
                        'insert into QV_Comentarios(id, '+
                        'colid, type, comment) values(' +
                        rtrim(convert(char,@v_id)) + ',0 ,' +
                        char(39) + 'T' + char(39) + ', ' +
                        char(39) + ' ' + char(39) + ')'
                        + @v_cr + @v_coluna
                        SET @v_conta = 1
                  end

                  SET @v_coluna = @v_coluna +
                  'insert into QV_Comentarios(id, colid, type, comment) values('
+
                  rtrim(convert(char,@v_id)) + ',' +
rtrim(convert(char,@v_colid))
                  + ',' + char(39) + 'C' + char(39) + ', '
                  + char(39) + ' ' + char(39) + ')' + @v_cr
            end
            else
            begin
                  SET @v_coluna = @v_coluna + @v_nome_tabela + '.'
                   + @v_nome_coluna + @v_cr
            end

            SET @v_tabela_anterior = @v_nome_tabela

            FETCH c_tabela INTO @v_nome_tabela, @v_nome_coluna,
                  @v_id,  @v_colid
      END

      CLOSE c_tabela
      DEALLOCATE CURSOR c_tabela

      PRINT '%1!%2!', @v_coluna, @v_cr
END
GO
```

16.13. O Procedimento QV_ListarTabelasRestricoes

```
/*================================================================================
Procedimento:     QV_ListarTabelasRestricoes
Autor:            Henrique Figueiredo de Souza
Data Criação:     24/02/2013
Descrição:
            Lista todas as tabelas e suas restrições.
================================================================================*/

CREATE PROCEDURE QV_ListarTabelasRestricoes
AS
BEGIN
      DECLARE @v_tabela_anterior VARCHAR(61)
      DECLARE @v_restricao VARCHAR(16384)
      DECLARE @v_cr CHAR(2)
      DECLARE @v_nome_tabela VARCHAR(255)
      DECLARE @v_nome_restricao VARCHAR(255)
      DECLARE @v_tipo_restricao VARCHAR(255)
      DECLARE @v_restricao_ref VARCHAR(255)
      DECLARE @v_tabela_ref VARCHAR(255)
      DECLARE c_tabela CURSOR FOR
```

```
      SELECT so.name table_name,
      i.name constraint_name,
      'PRIMARY KEY' constraint_type,
      null r_constraint_name,
      null r_table_name
      FROM sysobjects so, sysindexes i
      WHERE so.id = i.id
      AND so.type = 'U'
      UNION
      SELECT so.name table_name,
      object_name(r.constrid) constraint_name,
      'FOREIGN KEY' constraint_type,
      (SELECT i.name FROM sysindexes i
      WHERE i.id = r.reftabid) r_constraint_name,
      object_name(r.reftabid) r_table_name
      FROM sysobjects so, sysreferences r
      WHERE so.id = r.tableid
      AND so.type = 'U'
      ORDER BY so.name, 3 DESC

SET @v_cr = char(13) + char(10)

OPEN c_tabela
FETCH c_tabela INTO @v_nome_tabela, @v_nome_restricao,
@v_tipo_restricao, @v_restricao_ref, @v_tabela_ref

WHILE (@@sqlstatus !=2) AND (@@sqlstatus != 1)
BEGIN
      if @v_tabela_anterior<>@v_nome_tabela
      begin
            if @v_restricao is not null
            begin
                  PRINT '%1!%2!', @v_restricao, @v_cr
            end

            SET @v_restricao = null
      end

      if @v_tipo_restricao='PRIMARY KEY'
      begin
            SET @v_restricao = @v_restricao + @v_nome_restricao +
            ' = ' + @v_nome_tabela + '(' +
            dbo.QV_ColunasDaRestricaoDaTabela(@v_nome_tabela,
            @v_nome_restricao) + ')' + @v_cr
      end
      else
      begin
            SET @v_restricao = @v_restricao + @v_nome_restricao +
            ' = ' + @v_nome_tabela + '(' +
            dbo.QV_ColunasDaRestricaoDaTabela(@v_nome_tabela,
            @v_nome_restricao) + ') --> ' +
            @v_tabela_ref + '(' +
            dbo.QV_ColunasDaRestricaoDaTabela(@v_tabela_ref,
            @v_restricao_ref) + ')' + @v_cr
      end

      SET @v_tabela_anterior = @v_nome_tabela

      FETCH c_tabela INTO @v_nome_tabela, @v_nome_restricao,
      @v_tipo_restricao, @v_restricao_ref, @v_tabela_ref
END

CLOSE c_tabela
DEALLOCATE CURSOR c_tabela
```

```
	PRINT '%1!%2!', @v_restricao, @v_cr
END
GO
```

16.14. O Procedimento QV_TabelasNaoPossuem_PK_nemFK

```
/*================================================================================
Procedimento:	QV_TabelasNaoPossuem_PK_nemFK
Autor:		Henrique Figueiredo de Souza
Data Criação:	24/02/2013
Descrição:
		Lista todas as tabelas que NÃO possuem Chave Primária e
		não possuem Chave Estrangeira, ordenados por data de criação.
================================================================================*/

CREATE PROCEDURE QV_TabelasNaoPossuem_PK_nemFK
AS
BEGIN
	DECLARE	@v_tabela VARCHAR(16384)
	DECLARE	@v_nome_tabela varchar(64)
	DECLARE @v_cr CHAR(2)
	DECLARE c_tabela CURSOR FOR
		SELECT t.name table_name
		FROM sysobjects t
		WHERE t.type = 'U'
		AND t.name not in (
			SELECT DISTINCT table_name
			FROM QV_TabelasRestricoes)

	SET @v_cr = char(13) + char(10)

	OPEN c_tabela
	FETCH c_tabela INTO @v_nome_tabela

	WHILE (@@sqlstatus !=2) AND (@@sqlstatus != 1)
	BEGIN
		SET @v_tabela = @v_tabela + @v_nome_tabela + @v_cr

		FETCH c_tabela INTO @v_nome_tabela
	END

	CLOSE c_tabela
	DEALLOCATE CURSOR c_tabela

	PRINT '%1!', @v_tabela
END
GO
```

16.15. O Procedimento QV_TabelasSomente_PK_naoFK

```
/*================================================================================
Procedimento:	QV_TabelasSomente_PK_naoFK
Autor:		Henrique Figueiredo de Souza
Data Criação:	24/02/2013
Descrição:
		Lista todas as tabelas que possuem SOMENTE Chave Primária e
		não possuem Chave Estrangeira, ordenados por data de criação.
================================================================================*/

CREATE PROCEDURE QV_TabelasSomente_PK_naoFK
```

```
AS
bloco1: BEGIN
      DECLARE     @v_restricao VARCHAR(16384)
      DECLARE @v_cr CHAR(2)
      DECLARE @v_tabela VARCHAR(255)
      DECLARE c_tabela CURSOR FOR
            SELECT t.table_name
            FROM (SELECT DISTINCT table_name
                FROM QV_TabelasRestricoes
                WHERE constraint_type IN ('PRIMARY KEY','FOREIGN KEY')) t
            WHERE t.table_name not in (SELECT DISTINCT table_name
              FROM QV_TabelasRestricoes
              WHERE constraint_type='FOREIGN KEY')

      SET @v_cr = char(13) + char(10)

      OPEN c_tabela
      FETCH c_tabela INTO @v_tabela

      WHILE (@@sqlstatus !=2) AND (@@sqlstatus != 1)
      BEGIN
            SET @v_restricao = null

            bloco2: BEGIN
                  DECLARE @v_nome_tabela VARCHAR(255)
                  DECLARE @v_nome_restricao VARCHAR(255)
                  DECLARE @v_tipo_restricao VARCHAR(255)
                  DECLARE @v_restricao_ref VARCHAR(255)
                  DECLARE @v_tabela_ref VARCHAR(255)
                  DECLARE c_restricao CURSOR FOR
                        SELECT so.name table_name,
                        i.name constraint_name,
                        'PRIMARY KEY' constraint_type,
                        null r_constraint_name,
                        null r_table_name
                        FROM sysobjects so, sysindexes i
                        WHERE so.id = i.id
                        AND so.type = 'U'
                        AND so.name=@v_tabela
                        UNION
                        SELECT so.name table_name,
                        object_name(r.constrid) constraint_name,
                        'FOREIGN KEY' constraint_type,
                        (SELECT i.name FROM sysindexes i
                        WHERE i.id = r.reftabid) r_constraint_name,
                        object_name(r.reftabid) r_table_name
                        FROM sysobjects so, sysreferences r
                        WHERE so.id = r.tableid
                        AND so.type = 'U'
                        AND so.name=@v_tabela

                  OPEN c_restricao
                  FETCH c_restricao INTO @v_nome_tabela, @v_nome_restricao,
                  @v_tipo_restricao, @v_restricao_ref, @v_tabela_ref

                  WHILE (@@sqlstatus !=2) AND (@@sqlstatus != 1)
                  BEGIN
                        if @v_tipo_restricao='PRIMARY KEY'
                        begin
                              SET @v_restricao = @v_restricao +
                              @v_nome_restricao + ' = ' + @v_nome_tabela + '(' +
                              dbo.QV_ColunasDaRestricaoDaTabela(@v_nome_tabela,
                              @v_nome_restricao) + ')' + @v_cr
                        end
```

```
                        FETCH c_restricao INTO @v_nome_tabela,
@v_nome_restricao,
                        @v_tipo_restricao, @v_restricao_ref, @v_tabela_ref
                  END

                  CLOSE c_restricao
                  DEALLOCATE CURSOR c_restricao
            END

            PRINT '%1!', @v_restricao

            FETCH c_tabela INTO @v_tabela
      END

      CLOSE c_tabela
      DEALLOCATE CURSOR c_tabela
END
GO
```

16.16. O Procedimento QV_TabelasPossuem_FK

```
/*==========================================================================
Procedimento:     QV_TabelasPossuem_FK
Autor:            Henrique Figueiredo de Souza
Data Criação:     24/02/2013
Descrição:
            Lista todas as tabelas que possuem Chave Estrangeira e podem
            ou não possuir Chave Primária, ordenados por data de criação.
==========================================================================*/

CREATE PROCEDURE QV_TabelasPossuem_FK
AS
bloco1: BEGIN

      DECLARE     @v_restricao VARCHAR(16384)
      DECLARE @v_cr CHAR(2)
      DECLARE @v_retorno VARCHAR(16384)
      DECLARE @v_tabela VARCHAR(255)
      DECLARE c_tabela CURSOR FOR
            SELECT DISTINCT table_name
            FROM QV_TabelasRestricoes
            WHERE constraint_type='FOREIGN KEY'

      SET @v_cr = char(13) + char(10)

      OPEN c_tabela
      FETCH c_tabela INTO @v_tabela

      WHILE (@@sqlstatus !=2) AND (@@sqlstatus != 1)
      BEGIN
            SET @v_restricao = null

            bloco2: BEGIN
                  DECLARE @v_nome_tabela VARCHAR(255)
                  DECLARE @v_nome_restricao VARCHAR(255)
                  DECLARE @v_tipo_restricao VARCHAR(255)
                  DECLARE @v_restricao_ref VARCHAR(255)
                  DECLARE @v_tabela_ref VARCHAR(255)
                  DECLARE c_restricao CURSOR FOR
                        SELECT so.name table_name,
                        i.name constraint_name,
```

```
                        'PRIMARY KEY' constraint_type,
                        null r_constraint_name,
                        null r_table_name
                        FROM sysobjects so, sysindexes i
                        WHERE so.id = i.id
                        AND so.type = 'U'
                        AND so.name=@v_tabela
                        UNION
                        SELECT so.name table_name,
                        object_name(r.constrid) constraint_name,
                        'FOREIGN KEY' constraint_type,
                        (SELECT i.name FROM sysindexes i
                        WHERE i.id = r.reftabid) r_constraint_name,
                        object_name(r.reftabid) r_table_name
                        FROM sysobjects so, sysreferences r
                        WHERE so.id = r.tableid
                        AND so.type = 'U'
                        AND so.name=@v_tabela

                  OPEN c_restricao
                  FETCH c_restricao INTO @v_nome_tabela, @v_nome_restricao,
                  @v_tipo_restricao, @v_restricao_ref, @v_tabela_ref

                  WHILE (@@sqlstatus !=2) AND (@@sqlstatus != 1)
                  BEGIN
                        if @v_tipo_restricao='FOREIGN KEY'
                        begin
                              SET @v_restricao = @v_restricao +
                              @v_nome_restricao + ' = ' +
                              @v_nome_tabela + '(' +
                              dbo.QV_ColunasDaRestricaoDaTabela(@v_nome_tabela,
                              @v_nome_restricao) + ') --> ' + @v_tabela_ref +
                              '(' +
                              dbo.QV_ColunasDaRestricaoDaTabela(@v_tabela_ref,
                              @v_restricao_ref) + ')' + @v_cr
                        end

                        FETCH c_restricao INTO @v_nome_tabela,
                        @v_nome_restricao,
                        @v_tipo_restricao, @v_restricao_ref, @v_tabela_ref
                  END

                  CLOSE c_restricao
                  DEALLOCATE CURSOR c_restricao

            END

            PRINT '%1!', @v_restricao

            FETCH c_tabela INTO @v_tabela
      END

      CLOSE c_tabela
      DEALLOCATE CURSOR c_tabela
END
GO
```

16.17. O Procedimento QV_TabelasSomente_FK_naoPK

```
/*==========================================================================
Procedimento:     QV_TabelasSomente_FK_naoPK
Autor:            Henrique Figueiredo de Souza
```

```
Data Criação:     24/02/2013
Descrição:
            Lista todas as tabelas que possuem SOMENTE Chave Estrangeira
            e não possuem Chave Primária, ordenados por data de criação.
=========================================================================*/

CREATE PROCEDURE QV_TabelasSomente_FK_naoPK
AS
bloco1: BEGIN

      DECLARE     @v_restricao VARCHAR(16384)
      DECLARE @v_cr CHAR(2)
      DECLARE @v_tabela VARCHAR(255)
      DECLARE c_tabela CURSOR FOR
            SELECT t.name table_name
            FROM (SELECT t.name
                  FROM sysobjects t
                  WHERE t.type = 'U'
                  AND t.name not in (SELECT DISTINCT table_name
                  FROM QV_TabelasRestricoes
                  WHERE constraint_type='PRIMARY KEY')) t
            WHERE t.name in (SELECT DISTINCT table_name
              FROM QV_TabelasRestricoes
              WHERE constraint_type='FOREIGN KEY')

      SET @v_cr = char(13) + char(10)

      OPEN c_tabela
      FETCH c_tabela INTO @v_tabela

      WHILE (@@sqlstatus !=2) AND (@@sqlstatus != 1)
      BEGIN
            SET @v_restricao = null

            bloco2: BEGIN
                  DECLARE @v_nome_tabela VARCHAR(255)
                  DECLARE @v_nome_restricao VARCHAR(255)
                  DECLARE @v_tipo_restricao VARCHAR(255)
                  DECLARE @v_restricao_ref VARCHAR(255)
                  DECLARE @v_tabela_ref VARCHAR(255)
                  DECLARE c_restricao CURSOR FOR
                        SELECT so.name table_name,
                        i.name constraint_name,
                        'PRIMARY KEY' constraint_type,
                        null r_constraint_name,
                        null r_table_name
                        FROM sysobjects so, sysindexes i
                        WHERE so.id = i.id
                        AND so.type = 'U'
                        AND so.name=@v_tabela
                        UNION
                        SELECT so.name table_name,
                        object_name(r.constrid) constraint_name,
                        'FOREIGN KEY' constraint_type,
                        (SELECT i.name FROM sysindexes i
                        WHERE i.id = r.reftabid) r_constraint_name,
                        object_name(r.reftabid) r_table_name
                        FROM sysobjects so, sysreferences r
                        WHERE so.id = r.tableid
                        AND so.type = 'U'
                        AND so.name=@v_tabela

                  OPEN c_restricao
                  FETCH c_restricao INTO @v_nome_tabela, @v_nome_restricao,
```

```
                  @v_tipo_restricao, @v_restricao_ref, @v_tabela_ref

                  WHILE (@@sqlstatus !=2) AND (@@sqlstatus != 1)
                  BEGIN
                        if @v_tipo_restricao='FOREIGN KEY'
                        begin
                              SET @v_restricao = @v_restricao +
                              @v_nome_restricao + ' = ' + @v_nome_tabela + '(' +
                              dbo.QV_ColunasDaRestricaoDaTabela(@v_nome_tabela,
                              @v_nome_restricao) + ') --> ' +
                              @v_tabela_ref + '(' +
                              dbo.QV_ColunasDaRestricaoDaTabela(@v_tabela_ref,
                              @v_restricao_ref) + ')' + @v_cr
                        end

                        FETCH c_restricao INTO @v_nome_tabela,
@v_nome_restricao,
                        @v_tipo_restricao, @v_restricao_ref, @v_tabela_ref

                  END

                  CLOSE c_restricao
                  DEALLOCATE CURSOR c_restricao

            END

            PRINT '%1!', @v_restricao

            FETCH c_tabela INTO @v_tabela
      END

      CLOSE c_tabela
      DEALLOCATE CURSOR c_tabela
END
GO
```

16.18. O Procedimento QV_DescomentarTabelasColunas

```
/*=============================================================================
Procedimento:     QV_DescomentarTabelasColunas
Autor:            Henrique Figueiredo de Souza
Data Criação:     24/02/2013
Descrição:
            Descomentar todas a tabelas e colunas do banco de dados
=============================================================================*/

CREATE PROCEDURE QV_DescomentarTabelasColunas
AS
BEGIN
      DECLARE     @v_restricao VARCHAR(16384)
      DECLARE @v_cr CHAR(2)
      DECLARE @v_tabela_anterior VARCHAR(255)
      DECLARE @v_coluna VARCHAR(16384)
      DECLARE @v_conta integer
      DECLARE @v_nome_tabela VARCHAR(255)
      DECLARE @v_nome_coluna VARCHAR(255)
      DECLARE @v_id int
      DECLARE @v_colid smallint
      DECLARE c_tabela CURSOR FOR
            SELECT so.name as table_name,
            sc.name as column_name,
            so.id, sc.colid
```

```
            FROM sysobjects so, syscolumns sc
            WHERE sc.id = so.id
            and so.type = 'U'
            order by so.name, sc.colid

      SET @v_conta = 0
      SET @v_cr = char(13) + char(10)

      OPEN c_tabela
      FETCH c_tabela INTO @v_nome_tabela, @v_nome_coluna,
            @v_id, @v_colid

      WHILE (@@sqlstatus !=2) AND (@@sqlstatus != 1)
      BEGIN
            if @v_tabela_anterior<>@v_nome_tabela
            begin
                  if @v_coluna is not null
                  begin
                        PRINT '%1!%2!', @v_coluna, @v_cr
                  end

                  SET @v_coluna=null
                  SET @v_conta = 0
            end

            if @v_conta=0
            begin
                  SET @v_coluna = 'delete from QV_Comentarios where id=' +
                  rtrim(convert(char,@v_id)) + ' and colid=0 and type='
                  + char(39) + 'T' + char(39) + @v_cr + @v_coluna
                  SET @v_conta = 1
            end

            SET @v_coluna = @v_coluna + 'delete from QV_Comentarios where id='
            + rtrim(convert(char,@v_id)) + ' and colid=' +
            rtrim(convert(char,@v_colid)) + ' and type=' +
            char(39) + 'C' + char(39) + @v_cr

            SET @v_tabela_anterior = @v_nome_tabela

            FETCH c_tabela INTO @v_nome_tabela, @v_nome_coluna,
                  @v_id, @v_colid
      END

      CLOSE c_tabela
      DEALLOCATE CURSOR c_tabela

      PRINT '%1!%2!', @v_coluna, @v_cr
END
GO
```

17. O Pacote Qlikview para o IBM DB2

```
db2 -td/

CONNECT TO CAPACITA USER DB2ADMIN USING 12345678/

SET SERVEROUTPUT ON/

SELECT PacoteQlikview.ContaPalavra('ALFA BETA GAMA TETA') FROM SYSIBM.SYSDUMMY1;
SELECT PacoteQlikview.LimitaTamanhoTexto('ALFA BETA GAMA TETA', 3)
FROM SYSIBM.SYSDUMMY1;
```

```
SELECT PacoteQlikview.APLICARMAPA('PERGUNTA','CODIGO_TIPO_PERGUNTA')
FROM SYSIBM.SYSDUMMY1;
SELECT PacoteQlikview.COLUNASDARESTRICAODATABELA('PERGUNTA',
'PERGUNTA_TIPO_PERGUNTA_FK') FROM SYSIBM.SYSDUMMY1;
SELECT QV_ContarLinhasTabela('PERGUNTA') FROM SYSIBM.SYSDUMMY1;

CALL PacoteQlikview.LISTARTABELARESTRICAO('PERGUNTA')/
CALL PacoteQlikview.MontarLoad('PERGUNTA',
'Henrique Figueiredo de Souza', FALSE, FALSE)/
CALL PacoteQlikview.MontarTodosMappingLoad(
'Henrique Figueiredo de Souza', FALSE, TRUE)/
CALL PacoteQlikview.MontarTodosLoad('Henrique Figueiredo de Souza', TRUE)/
CALL PacoteQlikview.ListarTabelasColunas(TRUE)/
CALL PacoteQlikview.ListarTabelasRestricoes/
CALL PacoteQlikview.TabelasNaoPossuem_PK_nemFK/
CALL PacoteQlikview.TabelasSomente_PK_naoFK/
CALL PacoteQlikview.TabelasPossuem_FK/
CALL PacoteQlikview.TabelasSomente_FK_naoPK/
CALL PacoteQlikview.DescomentarTabelasColunas/

DROP PACKAGE BODY PacoteQlikview;
DROP PACKAGE PacoteQlikview;
DROP FUNCTION QV_ContarLinhasTabela;

CREATE OR REPLACE PACKAGE PacoteQlikview
AS
      FUNCTION ContaPalavra(p_str IN VARCHAR(32672))
              RETURN INTEGER;
      FUNCTION LimitaTamanhoTexto(p_str IN VARCHAR(32672),
            p_limite IN integer) RETURN VARCHAR(32672);
      FUNCTION AplicarMapa(p_tabela IN VARCHAR(128),
            p_coluna IN VARCHAR(128)) RETURN VARCHAR(32672);
      FUNCTION ColunasDaRestricaoDaTabela(p_tabela IN VARCHAR(128),
            p_restricao IN VARCHAR(128)) RETURN VARCHAR(32672);
      PROCEDURE ListarTabelaRestricao(p_tabela IN VARCHAR(128));
      PROCEDURE MontarLoad(p_tabela IN VARCHAR(128),
            p_autor IN VARCHAR(128), p_mapeada IN boolean,
            p_2_colunas IN boolean);
      PROCEDURE MontarTodosMappingLoad(p_autor IN VARCHAR(128),
            p_vazia IN boolean, p_2_colunas IN boolean);
      PROCEDURE MontarTodosLoad(p_autor IN VARCHAR(128),
            p_vazia IN boolean);
      PROCEDURE ListarTabelasColunas(p_comentario IN boolean);
      PROCEDURE ListarTabelasRestricoes;
      PROCEDURE TabelasNaoPossuem_PK_nemFK;
      PROCEDURE TabelasPossuemSomente_PK_naoFK;
      PROCEDURE TabelasPossuem_FK;
      PROCEDURE TabelasPossuemSomente_FK_naoPK;
      PROCEDURE DescomentarTabelasColunas;
END PacoteQlikview;


CREATE OR REPLACE PACKAGE BODY PacoteQlikview
AS
...
END PacoteQlikview;
```

17.1. A Função QV_ContarLinhasTabela

```
/*==============================================================================
Procedimento:     QV_ContarLinhasTabela
Autor:            Henrique Figueiredo de Souza
Data Criação:     27/02/2013
Parametros:
            p_tabela    Nome da tabela referenciada pela FK.
Descrição:
            Retorna a quantidade de linhas de uma tabela.
==============================================================================*/

CREATE OR REPLACE FUNCTION QV_ContarLinhasTabela(IN p_tabela VARCHAR(128))
RETURNS INTEGER
LANGUAGE SQL
BEGIN
  DECLARE v_conta INTEGER;
  DECLARE v_sql VARCHAR(1000);
  DECLARE c_tabela CURSOR FOR stmt;

  SET v_sql = 'SELECT COUNT(*) FROM ' CONCAT p_tabela;

  PREPARE stmt FROM v_sql;
  OPEN c_tabela;
  FETCH c_tabela INTO v_conta;
  CLOSE c_tabela;

  RETURN v_conta;
END;
```

17.2. A Função ContaPalavra

```
/*==============================================================================
Procedimento:     ContaPalavra
Autor:            Henrique Figueiredo de Souza
Data Criação:     27/02/2013
Parametros:
            p_str       Texto a ser contado as palavras.
Descrição:
            Conta as palavras de um texto.
==============================================================================*/

FUNCTION ContaPalavra (p_str IN VARCHAR(32672))
  RETURN INTEGER
AS
      v_palavras BIGINT := 0;
      v_tamanho INTEGER := NVL(LENGTH(p_str),0);
      v_dentro_uma_palavra BOOLEAN;
BEGIN
      FOR i IN 1..v_tamanho + 1
      LOOP
            IF ASCII(SUBSTR(p_str, i, 1)) < 33 OR i > v_tamanho THEN
                  IF v_dentro_uma_palavra THEN
                        v_palavras := v_palavras + 1;
                        v_dentro_uma_palavra := FALSE;
                  END IF;
            ELSE
                  v_dentro_uma_palavra := TRUE;
            END IF;
      END LOOP;
      RETURN v_palavras;
END;
```

17.3. A Função LimitaTamanhoTexto

```
/*=============================================================================
Procedimento:     LimitaTamanhoTexto
Autor:            Henrique Figueiredo de Souza
Data Criação:     27/02/2013
Parametros:
            p_str       Texto a ser limitado.
            p_limite    Limite de palavras por linhas.
Descrição:
            Limita o texto fornecido a uma determinada quantidades
            de palavras por linha.
=============================================================================*/

FUNCTION LimitaTamanhoTexto(p_str IN VARCHAR(32672),
p_limite IN integer)
RETURN VARCHAR(32672)
IS
      v_palavras INTEGER := 0;
      v_tamanho INTEGER := NVL(LENGTH(p_str),0);
      v_dentro_uma_palavra BOOLEAN;
      v_texto VARCHAR(32672);
BEGIN
      v_texto := '';
      FOR i IN 1..v_tamanho + 1
      LOOP
            v_texto := v_texto || SUBSTR(p_str, i, 1);

            IF ASCII(SUBSTR(p_str, i, 1)) < 33 OR i > v_tamanho THEN
                  IF v_dentro_uma_palavra THEN
                        v_palavras := v_palavras + 1;
                        v_dentro_uma_palavra := FALSE;

                        if v_palavras=p_limite then
                              v_texto := v_texto || chr(13) || chr(10);
                              v_palavras := 0;
                        end if;
                  END IF;
            ELSE
                  v_dentro_uma_palavra := TRUE;
            END IF;
      END LOOP;

      RETURN TRIM(v_texto);
END;
```

17.4. A Função AplicarMapa

```
/*=============================================================================
Procedimento:     AplicarMapa
Autor:            Henrique Figueiredo de Souza
Data Criação:     27/02/2013
Parametros:
            p_tabela    Nome da tabela referenciada pela FK.
            p_coluna    Nome da coluna que possui uma FK.
Descrição:
            Aplicar o mapa carregado, onde a coluna é uma
            Chave Estrangeira e tabela é a tabela referenciada pela
            chave estrangeira da coluna.
=============================================================================*/

FUNCTION AplicarMapa(p_tabela IN VARCHAR(128),
p_coluna IN VARCHAR(128))
RETURN VARCHAR(32672)
```

```
IS
      v_mapa VARCHAR(8000);
      v_tabela VARCHAR(128);
      v_tabela_mapa VARCHAR(128);

      CURSOR c_mapa is
            SELECT R.REFTABNAME
            FROM SYSCAT.REFERENCES R, SYSCAT.KEYCOLUSE KCU
            WHERE R.TABSCHEMA=KCU.TABSCHEMA
            AND R.TABNAME=KCU.TABNAME
            AND R.CONSTNAME=KCU.CONSTNAME
            AND R.TABSCHEMA=CURRENT SCHEMA
            AND R.OWNER=CURRENT USER
            AND R.OWNERTYPE='U'
            AND R.TABNAME=p_tabela
            AND KCU.COLNAME=p_coluna;

      CURSOR c_tabela is
            SELECT T.TABNAME
            FROM (SELECT T1.TABNAME, T1.CREATE_TIME,
                  T1.TABSCHEMA, T1.OWNER, T1.OWNERTYPE
                  FROM SYSCAT.TABLES T1
                  WHERE T1.TABNAME IN (
                        SELECT DISTINCT I.TABNAME
                        FROM SYSCAT.INDEXES I
                        WHERE I.INDSCHEMA=T1.TABSCHEMA
                        AND I.OWNER=T1.OWNER
                        AND I.OWNERTYPE=T1.OWNERTYPE
                        UNION
                        SELECT DISTINCT R.TABNAME
                        FROM SYSCAT.REFERENCES R
                        WHERE R.TABSCHEMA=T1.TABSCHEMA
                        AND R.OWNER=T1.OWNER
                        AND R.OWNERTYPE=T1.OWNERTYPE)
                  AND T1.TABSCHEMA=CURRENT SCHEMA
                  AND T1.OWNER=CURRENT USER
                  AND T1.OWNERTYPE='U' AND T1.TYPE='T') T
            WHERE T.TABNAME NOT IN (
                  SELECT DISTINCT R.TABNAME
                  FROM SYSCAT.REFERENCES R
                  WHERE R.TABSCHEMA=T.TABSCHEMA
                  AND R.OWNER=T.OWNER
                  AND R.OWNERTYPE=T.OWNERTYPE)
            AND T.TABNAME=v_tabela_mapa;

BEGIN
      v_mapa := '';

      -- verifica se a coluna da tabela é uma Chave Estrangeira
      open c_mapa;
      fetch c_mapa into v_tabela_mapa;

      if c_mapa%found then

            open c_tabela;
            fetch c_tabela into v_tabela;

            -- verifica se tabela possui somente Chave Primaria
            if c_tabela%found then
                  close c_tabela;

                  v_mapa:= 'ApplyMap(' || chr(39) || 'Mapa' ||
                  v_tabela_mapa || chr(39) || ',' || p_coluna || ')';
            else
```

```
                close c_tabela;
            end if;

            close c_mapa;
      else
            close c_mapa;
      end if;

      RETURN TRIM(v_mapa);

EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Um erro foi encontrado - '
                ||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

17.5. A Função ColunasDaRestricaoDaTabela

```
/*=============================================================================
Procedimento:     ColunasDaRestricaoDaTabela
Autor:            Henrique Figueiredo de Souza
Data Criação:     27/02/2013
Parametros:
            p_tabela    Nome da tabela referenciada pela FK.
            p_restricao Nome da restrição.
Descrição:
            Retorna as colunas de uma tabela e sua restrição.
=============================================================================*/

FUNCTION ColunasDaRestricaoDaTabela(
p_tabela IN VARCHAR(128), p_restricao IN VARCHAR(128))
RETURN VARCHAR(32672)
IS
      v_coluna VARCHAR(32672);
BEGIN

      v_coluna := '';

      FOR c_tabela IN (SELECT KCU.COLNAME COLUMN_NAME
            FROM SYSCAT.KEYCOLUSE KCU
            WHERE KCU.TABSCHEMA=CURRENT SCHEMA
            AND KCU.TABNAME=p_tabela
            AND KCU.CONSTNAME=p_restricao)
      LOOP

            v_coluna := v_coluna || ', ' || c_tabela.column_name;

      END LOOP;

  v_coluna := substr(v_coluna, 3, length(v_coluna));

  RETURN TRIM(v_coluna);

EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Um erro foi encontrado - '
                ||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

17.6. O Procedimento ListarTabelaRestricao

```
/*================================================================================
Procedimento:     ListarTabelaRestricao
Autor:            Henrique Figueiredo de Souza
Data Criação:     27/02/2013
Parametros:
            p_tabela    Nome da tabela.
Descrição:
            Lista restrições de uma tabela.
================================================================================*/

PROCEDURE ListarTabelaRestricao(p_tabela IN VARCHAR(128))
IS
      v_tabela_anterior VARCHAR(128);
      v_tabela VARCHAR(8000);
      v_restricao VARCHAR(8000);
      v_cr CHAR(2);
      v_tam INTEGER;
BEGIN
      v_cr := chr(13) || chr(10);
      v_tabela_anterior := '';
      v_restricao := '';

      FOR c_tabela IN (
            SELECT DISTINCT I.TABNAME TABLE_NAME, I.INDNAME CONSTRAINT_NAME,
            'PRIMARY KEY' CONSTRAINT_TYPE, NULL R_TABLE_NAME, NULL
R_CONSTRAINT_NAME
            FROM SYSCAT.INDEXES I
            WHERE I.INDSCHEMA=CURRENT SCHEMA
            AND I.OWNER=CURRENT USER
            AND I.OWNERTYPE='U'
            AND I.TABNAME=p_tabela
            UNION
            SELECT DISTINCT R.TABNAME, R.CONSTNAME,
            'FOREIGN KEY' CONSTRAINT_TYPE, R.REFTABNAME, R.REFKEYNAME
            FROM SYSCAT.REFERENCES R
            WHERE R.TABSCHEMA=CURRENT SCHEMA
            AND R.OWNER=CURRENT USER
            AND R.OWNERTYPE='U'
            AND R.TABNAME=p_tabela
            ORDER BY 1)
      LOOP

            if v_tabela_anterior<>c_tabela.table_name then
                  DBMS_OUTPUT.put_line (v_restricao);
                  v_restricao:='';
            end if;

            if c_tabela.CONSTRAINT_TYPE='PRIMARY KEY' then
                  v_restricao := v_restricao || c_tabela.table_name || '(' ||
                  ColunasDaRestricaoDaTabela(c_tabela.table_name,
                  c_tabela.CONSTRAINT_NAME) || ')' || v_cr;
            else
                  v_restricao := v_restricao || c_tabela.table_name || '(' ||
                  ColunasDaRestricaoDaTabela(c_tabela.table_name,
                  c_tabela.CONSTRAINT_NAME) || ') --> ' ||
                  c_tabela.R_TABLE_NAME || '(' ||
                  ColunasDaRestricaoDaTabela(c_tabela.R_TABLE_NAME,
                  c_tabela.R_CONSTRAINT_NAME) || ')' || v_cr;
            end if;

            v_tabela_anterior := c_tabela.table_name;

      END LOOP;
```

```
      v_tam := length(v_restricao)-2;

      if v_tam > 0 then
            v_restricao := substr(v_restricao, 1, v_tam);
            DBMS_OUTPUT.put_line (v_restricao);
      end if;

EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Um erro foi encontrado - '
                  ||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

17.7. O Procedimento MontarLoad

```
/*=============================================================================
Procedimento:     MontarLoad
Autor:            Henrique Figueiredo de Souza
Data Criação:     27/02/2013
Parametros:
            p_tabela    Nome da tabela a gerar o LOAD.
            p_autor           Nome do autor a ser mostrado no comentário.
            p_mapeada   TRUE para mapping load, FALSE para sem mapping load
            p_2_colunas Limita a saida a somente a duas colunas.
Descrição:
            Monta um LOAD em formato qlikview de uma tabela
=============================================================================*/

PROCEDURE MontarLoad(p_tabela IN VARCHAR(128),
p_autor IN VARCHAR(128), p_mapeada IN boolean, p_2_colunas IN boolean)
IS
      v_tabela VARCHAR(128);
      v_coluna VARCHAR(32672);
      v_comenta VARCHAR(8000);
      v_rotulo VARCHAR(4000);
      v_rotulo_tabela VARCHAR(4000);
      v_mapa VARCHAR(4000);
      v_nomecoluna VARCHAR(4000);
      v_cr CHAR(2);
      v_conta INTEGER;
BEGIN
      v_cr := chr(13) || chr(10);
      v_coluna := '';
      v_conta := 0;

      FOR c_coluna IN (
            SELECT T.TABNAME TABLE_NAME, C.COLNAME COLUMN_NAME, C.COLNO,
            C.REMARKS COMMENTS, T.REMARKS TAB_COMMENTS
            FROM SYSCAT.TABLES T, SYSCAT.COLUMNS C
            WHERE T.TABSCHEMA=C.TABSCHEMA AND T.TABNAME=C.TABNAME
            AND T.TABSCHEMA=CURRENT SCHEMA AND T.OWNER=CURRENT USER
            AND T.OWNERTYPE='U' AND T.TYPE='T'
            AND T.TABNAME=p_tabela
            ORDER BY T.TABNAME, C.COLNO)
      LOOP
            v_tabela := c_coluna.table_name;
            v_rotulo := c_coluna.comments;
            v_rotulo_tabela := LimitaTamanhoTexto(c_coluna.tab_comments, 10);
            v_mapa := AplicarMapa(v_tabela, c_coluna.column_name);

            if length(v_mapa) > 0 then
                  v_nomecoluna := v_mapa;
```

```
            else
                  v_nomecoluna := c_coluna.column_name;
            end if;

            if v_rotulo is null then
                  v_coluna := v_coluna || ', ' || v_cr
                  || '      ' || v_nomecoluna;
            else
                  v_coluna := v_coluna || ', ' || v_cr
                  || '      ' || v_nomecoluna ||
                  ' as [' || v_rotulo || ']';
            end if;

            if p_2_colunas=TRUE then
                  v_conta := v_conta + 1;

                  if v_conta=2 then
                        EXIT;
                  end if;
            end if;

      END LOOP;

      if length(v_coluna) > 0 then

            v_coluna := substr(v_coluna, 3, length(v_coluna));

            v_comenta := v_cr ||

      '/*=============================================================================
======' || v_cr;

            if p_mapeada=TRUE then
                  v_comenta := v_comenta || 'Procedimento:' || chr(9) ||
                  'Carga mapeada em memória do arquivo QVD' || v_cr;
            else
                  v_comenta := v_comenta || 'Procedimento:' || chr(9) ||
                  'Carga em memória do arquivo QVD' || v_cr;
            end if;
            v_comenta := v_comenta || 'ArquivoQVD:' ||
            chr(9) || chr(9) || v_tabela || v_cr;
            v_comenta := v_comenta || 'Autor:' || chr(9) ||
            chr(9) || chr(9) || p_autor || v_cr;
            v_comenta := v_comenta || 'Data Criação:' || chr(9) ||
                              to_char(sysdate, 'dd/mm/yyyy') || v_cr;

            if v_rotulo_tabela is not null then
                  if  length(v_rotulo_tabela) > 0 then
                        v_comenta := v_comenta || 'Descrição:' || chr(9) ||
                        v_rotulo_tabela || v_cr;
                  end if;
            end if;

            DBMS_OUTPUT.put_line(v_comenta || 'Restrições:');

            ListarTabelaRestricao(p_tabela);

            DBMS_OUTPUT.put_line(

      '=============================================================================
====*/' || v_cr);

            if p_mapeada=TRUE then
                  DBMS_OUTPUT.put_line ('Mapa' || v_tabela
```

```
                  || ':' || v_cr || 'Mapping LOAD' || v_coluna
                  || v_cr || 'FROM' || v_cr || '$(vDiretorio)'
                  || v_tabela || '.qvd' || v_cr || '(qvd);' || v_cr);
            else
                  DBMS_OUTPUT.put_line (v_tabela || ':'
                  || v_cr || 'LOAD' || v_coluna || v_cr ||
                  'FROM' || v_cr || '$(vDiretorio)'
                  || v_tabela || '.qvd' || v_cr || '(qvd);' || v_cr);
            end if;

      end if;

EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Um erro foi encontrado - '
                  ||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

17.8. O Procedimento MontarTodosMappingLoad

```
/*================================================================================
Procedimento:     MontarTodosMappingLoad
Autor:            Henrique Figueiredo de Souza
Data Criação:     27/02/2013
Parametros:
            p_autor           Nome do autor a ser mostrado no comentário.
            p_vazia           Se 1 verificar se a tabela está vazia.
            p_2_colunas Limita a saida a somente a duas colunas.
Descrição:
            Monta todos os Mapping LOAD em formato qlikview
            de todas as tabelas que possuem SOMENTE Chave Primária.
================================================================================*/

PROCEDURE MontarTodosMappingLoad(p_autor IN VARCHAR(128),
p_vazia IN boolean, p_2_colunas IN boolean)
IS
      v_tabela VARCHAR(128);
      v_conta INTEGER;
BEGIN
      FOR c_tabela IN (
            SELECT T.TABNAME TABLE_NAME
            FROM (SELECT T1.TABNAME, T1.CREATE_TIME,
                  T1.TABSCHEMA, T1.OWNER, T1.OWNERTYPE
                  FROM SYSCAT.TABLES T1
                  WHERE T1.TABNAME IN (
                        SELECT DISTINCT I.TABNAME
                        FROM SYSCAT.INDEXES I
                        WHERE I.INDSCHEMA=T1.TABSCHEMA
                        AND I.OWNER=T1.OWNER
                        AND I.OWNERTYPE=T1.OWNERTYPE
                        UNION
                        SELECT DISTINCT R.TABNAME
                        FROM SYSCAT.REFERENCES R
                        WHERE R.TABSCHEMA=T1.TABSCHEMA
                        AND R.OWNER=T1.OWNER
                        AND R.OWNERTYPE=T1.OWNERTYPE)
                  AND T1.TABSCHEMA=CURRENT SCHEMA
                  AND T1.OWNER=CURRENT USER
                  AND T1.OWNERTYPE='U' AND T1.TYPE='T') T
            WHERE T.TABNAME NOT IN (
                  SELECT DISTINCT R.TABNAME
                  FROM SYSCAT.REFERENCES R
```

```
                  WHERE R.TABSCHEMA=T.TABSCHEMA
                  AND R.OWNER=T.OWNER
                  AND R.OWNERTYPE=T.OWNERTYPE)
            ORDER BY VARCHAR_FORMAT(T.CREATE_TIME,
            'DD/MM/YYYY HH24:MI:SS:SSSSS') ASC)
      LOOP
            v_tabela := c_tabela.TABLE_NAME;

            if p_vazia = TRUE then
                  --EXECUTE IMMEDIATE 'SELECT COUNT(*) FROM ' ||
                  --    v_tabela INTO v_conta;
                  v_conta := QV_ContarLinhasTabela(v_tabela);

                  if v_conta > 0 then
                        MontarLoad(v_tabela, p_autor,
                              TRUE, p_2_colunas);
                  end if;
            else
                  MontarLoad(v_tabela, p_autor,
                        TRUE, p_2_colunas);
            end if;
      END LOOP;

EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Um erro foi encontrado - '
                  ||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

17.9. O Procedimento MontarTodosLoad

```
/*================================================================================
Procedimento:     MontarTodosLoad
Autor:            Henrique Figueiredo de Souza
Data Criação:     27/02/2013
Parametros:
            p_autor           Nome do autor a ser mostrado no comentário.
            p_vazia           Se TRUE verificar se a tabela está vazia.
Descrição:
            Monta todos os LOAD em formato qlikview de todas as tabelas
================================================================================*/

PROCEDURE MontarTodosLoad(p_autor IN VARCHAR(128),
p_vazia IN boolean)
IS
      v_tabela VARCHAR(128);
      v_conta INTEGER;
BEGIN
      FOR c_tabela IN (
            SELECT TABNAME TABLE_NAME
            FROM SYSCAT.TABLES
            WHERE TABSCHEMA=CURRENT SCHEMA
            AND OWNER=CURRENT USER
            AND OWNERTYPE='U'
            AND TYPE='T'
            ORDER BY TABNAME)
      LOOP
            v_tabela := c_tabela.table_name;

            if p_vazia = TRUE then
                  --EXECUTE IMMEDIATE 'SELECT COUNT(*) FROM ' ||
                  --    v_tabela INTO v_conta;
```

```
                  v_conta := QV_ContarLinhasTabela(v_tabela);

                  if v_conta > 0 then
                        MontarLoad(v_tabela,
                              p_autor, FALSE, FALSE);
                  end if;
            else
                  MontarLoad(v_tabela, p_autor, FALSE, FALSE);
            end if;

      END LOOP;

EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Um erro foi encontrado - '
                  ||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

17.10. O Procedimento ListarTabelasColunas

```
/*================================================================================
Procedimento:     ListarTabelasColunas
Autor:            Henrique Figueiredo de Souza
Data Criação:     27/02/2013
Parametros:
            p_comentario      Se TRUE incluir comando para gerar comentario.
Descrição:
            Lista todas a tabelas e colunas do banco de dados
================================================================================*/

PROCEDURE ListarTabelasColunas(p_comentario IN boolean)
IS
      v_tabela_anterior VARCHAR(128);
      v_coluna VARCHAR(8000);
      v_cr CHAR(2);
      v_conta integer;
BEGIN
      v_cr := chr(13) || chr(10);
      v_tabela_anterior := '';
      v_conta := 0;

      FOR c_tabela IN (
            SELECT T.TABNAME TABLE_NAME, C.COLNAME COLUMN_NAME
            FROM SYSCAT.TABLES T, SYSCAT.COLUMNS C
            WHERE T.TABSCHEMA=C.TABSCHEMA AND T.TABNAME=C.TABNAME
            AND T.TABSCHEMA=CURRENT SCHEMA AND T.OWNER=CURRENT USER
            AND T.OWNERTYPE='U' AND T.TYPE='T'
            ORDER BY T.TABNAME, C.COLNO)
      LOOP

            if v_tabela_anterior<>c_tabela.table_name then
                  DBMS_OUTPUT.put_line (v_coluna);
                  v_coluna:='';
                  v_conta := 0;
            end if;

            if p_comentario=TRUE then

                  if v_conta=0 then
                        v_coluna := 'comment on table ' ||
                        c_tabela.table_name || ' is '
                        || chr(39) || chr(9) || chr(9) ||
```

```
                        chr(39) || ';' || v_cr || v_coluna;
                        v_conta := 1;
                  end if;

                  v_coluna := v_coluna || 'comment on column ' ||
                  c_tabela.table_name || '.' || c_tabela.column_name
                  || ' is ' || chr(39) || chr(9) || chr(9) ||
                  chr(39) || ';' || v_cr;
            else
                  v_coluna := v_coluna || c_tabela.table_name || '.'
                  || c_tabela.column_name || v_cr;
            end if;

            v_tabela_anterior := c_tabela.table_name;

      END LOOP;

      DBMS_OUTPUT.put_line (v_coluna);

EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Um erro foi encontrado - '
                  ||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

17.11. O Procedimento ListarTabelasRestricoes

```
/*================================================================================
Procedimento:     ListarTabelasRestricoes
Autor:            Henrique Figueiredo de Souza
Data Criação:     27/02/2013
Descrição:
            Lista todas as tabelas e suas restrições.
================================================================================*/

PROCEDURE ListarTabelasRestricoes
IS
      v_tabela_anterior VARCHAR(128);
      v_tabela VARCHAR(8000);
      v_restricao VARCHAR(8000);
      v_cr CHAR(2);
BEGIN
      v_cr := chr(13) || chr(10);
      v_tabela_anterior := '';
      v_restricao := '';

      FOR c_tabela IN (
            SELECT DISTINCT I.TABNAME TABLE_NAME,
            I.INDNAME CONSTRAINT_NAME,
            'PRIMARY KEY' CONSTRAINT_TYPE,
            NULL R_CONSTRAINT_NAME, NULL R_TABLE_NAME
            FROM SYSCAT.INDEXES I
            WHERE I.INDSCHEMA=CURRENT SCHEMA
            AND I.OWNER=CURRENT USER
            AND I.OWNERTYPE='U'
            UNION
            SELECT DISTINCT R.TABNAME TABLE_NAME,
            R.CONSTNAME CONSTRAINT_NAME,
            'FOREIGN KEY' CONSTRAINT_TYPE,
            R.REFKEYNAME R_CONSTRAINT_NAME,
            R.REFTABNAME R_TABLE_NAME
            FROM SYSCAT.REFERENCES R
```

```
            WHERE R.TABSCHEMA=CURRENT SCHEMA
            AND R.OWNER=CURRENT USER
            AND R.OWNERTYPE='U'
            ORDER BY 1, 3 DESC)
      LOOP

            if v_tabela_anterior<>c_tabela.table_name then
                  DBMS_OUTPUT.put_line (v_restricao);
                  v_restricao:='';
            end if;

            if c_tabela.CONSTRAINT_TYPE='PRIMARY KEY' then
                  v_restricao := v_restricao ||
                  c_tabela.CONSTRAINT_NAME || ' = ' ||
                  c_tabela.table_name || '(' ||
                  ColunasDaRestricaoDaTabela(c_tabela.table_name,
                  c_tabela.CONSTRAINT_NAME) || ')' || v_cr;
            else
                  v_restricao := v_restricao ||
                  c_tabela.CONSTRAINT_NAME || ' = ' ||
                  c_tabela.table_name || '(' ||
                  ColunasDaRestricaoDaTabela(c_tabela.table_name,
                  c_tabela.CONSTRAINT_NAME) || ') --> ' ||
                  c_tabela.R_TABLE_NAME || '(' ||
                  ColunasDaRestricaoDaTabela(c_tabela.R_TABLE_NAME,
                  c_tabela.R_CONSTRAINT_NAME) || ')' || v_cr;
            end if;

            v_tabela_anterior := c_tabela.table_name;

      END LOOP;

      DBMS_OUTPUT.put_line (v_restricao);

EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Um erro foi encontrado - '
                  ||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

17.12. O Procedimento TabelasNaoPossuem_PK_nemFK

```
/*================================================================================
Procedimento:     TabelasNaoPossuem_PK_nemFK
Autor:            Henrique Figueiredo de Souza
Data Criação:     27/02/2013
Descrição:
            Lista todas as tabelas que NÃO possuem Chave Primária e
            não possuem Chave Estrangeira, ordenados por data de criação.
================================================================================*/

PROCEDURE TabelasNaoPossuem_PK_nemFK
IS
      v_tabela VARCHAR(8000);
      v_cr CHAR(2);
BEGIN
      v_cr := chr(13) || chr(10);
      v_tabela := '';

      FOR c_objeto IN (
            SELECT T.TABNAME TABLE_NAME
            FROM SYSCAT.TABLES T
```

```
            WHERE T.TABNAME NOT IN (
                  SELECT DISTINCT I.TABNAME
                  FROM SYSCAT.INDEXES I
                  WHERE I.INDSCHEMA=T.TABSCHEMA
                  AND I.OWNER=T.OWNER
                  AND I.OWNERTYPE=T.OWNERTYPE
                  UNION
                  SELECT DISTINCT R.TABNAME
                  FROM SYSCAT.REFERENCES R
                  WHERE R.TABSCHEMA=T.TABSCHEMA
                  AND R.OWNER=T.OWNER
                  AND R.OWNERTYPE=T.OWNERTYPE)
            AND T.TABSCHEMA=CURRENT SCHEMA
            AND T.OWNER=CURRENT USER
            AND T.OWNERTYPE='U'
            AND T.TYPE='T'
            ORDER BY VARCHAR_FORMAT(CREATE_TIME,
            'DD/MM/YYYY HH24:MI:SS:SSSSS') ASC)
      LOOP
            v_tabela := v_tabela || c_objeto.TABLE_NAME || v_cr;

      END LOOP;

      DBMS_OUTPUT.put_line (v_tabela);

EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Um erro foi encontrado - '
                  ||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

17.13. O Procedimento TabelasPossuemSomente_PK_naoFK

```
/*================================================================================
Procedimento:     TabelasPossuemSomente_PK_naoFK
Autor:            Henrique Figueiredo de Souza
Data Criação:     27/02/2013
Descrição:
            Lista todas as tabelas que possuem SOMENTE Chave Primária e
            não possuem Chave Estrangeira, ordenados por data de criação.
================================================================================*/

PROCEDURE TabelasSomente_PK_naoFK
IS
      v_restricao VARCHAR(8000);
      v_cr CHAR(2);
BEGIN
      v_cr := chr(13) || chr(10);

      FOR c_objeto IN (
            SELECT T.TABNAME TABLE_NAME
            FROM (SELECT T1.TABNAME, T1.CREATE_TIME,
                  T1.TABSCHEMA, T1.OWNER, T1.OWNERTYPE
                  FROM SYSCAT.TABLES T1
                  WHERE T1.TABNAME IN (
                        SELECT DISTINCT I.TABNAME
                        FROM SYSCAT.INDEXES I
                        WHERE I.INDSCHEMA=T1.TABSCHEMA
                        AND I.OWNER=T1.OWNER
                        AND I.OWNERTYPE=T1.OWNERTYPE
                        UNION
                        SELECT DISTINCT R.TABNAME
```

```
                        FROM SYSCAT.REFERENCES R
                        WHERE R.TABSCHEMA=T1.TABSCHEMA
                        AND R.OWNER=T1.OWNER
                        AND R.OWNERTYPE=T1.OWNERTYPE)
                  AND T1.TABSCHEMA=CURRENT SCHEMA
                  AND T1.OWNER=CURRENT USER
                  AND T1.OWNERTYPE='U' AND T1.TYPE='T') T
            WHERE T.TABNAME NOT IN (
                  SELECT DISTINCT R.TABNAME
                  FROM SYSCAT.REFERENCES R
                  WHERE R.TABSCHEMA=T.TABSCHEMA
                  AND R.OWNER=T.OWNER
                  AND R.OWNERTYPE=T.OWNERTYPE)
            ORDER BY VARCHAR_FORMAT(T.CREATE_TIME,
            'DD/MM/YYYY HH24:MI:SS:SSSSS') ASC)
      LOOP
            v_restricao := '';

            FOR c_tabela IN (
                  SELECT DISTINCT I.TABNAME TABLE_NAME,
                  I.INDNAME CONSTRAINT_NAME,
                  'PRIMARY KEY' CONSTRAINT_TYPE,
                  NULL R_TABLE_NAME, NULL R_CONSTRAINT_NAME
                  FROM SYSCAT.INDEXES I
                  WHERE I.INDSCHEMA=CURRENT SCHEMA
                  AND I.OWNER=CURRENT USER
                  AND I.OWNERTYPE='U'
                  AND I.TABNAME=c_objeto.TABLE_NAME
                  UNION
                  SELECT DISTINCT R.TABNAME, R.CONSTNAME,
                  'FOREIGN KEY' CONSTRAINT_TYPE,
                  R.REFTABNAME, R.REFKEYNAME
                  FROM SYSCAT.REFERENCES R
                  WHERE R.TABSCHEMA=CURRENT SCHEMA
                  AND R.OWNER=CURRENT USER
                  AND R.OWNERTYPE='U'
                  AND R.TABNAME=c_objeto.TABLE_NAME)
            LOOP

                  if c_tabela.CONSTRAINT_TYPE='PRIMARY KEY' then
                        v_restricao := v_restricao ||
                        c_tabela.CONSTRAINT_NAME || ' = '
                        || c_tabela.table_name || '(' ||
                        ColunasDaRestricaoDaTabela(
                        c_tabela.table_name,
                        c_tabela.CONSTRAINT_NAME) || ')' || v_cr;
                  end if;

            END LOOP;

            DBMS_OUTPUT.put_line (v_restricao);

      END LOOP;

EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Um erro foi encontrado - '
                  ||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

17.14. O Procedimento TabelasPossuem_FK

```
/*================================================================================
Procedimento:     TabelasPossuem_FK
Autor:            Henrique Figueiredo de Souza
Data Criação:     27/02/2013
Descrição:
            Lista todas as tabelas que possuem Chave Estrangeira e podem
            ou não possuir Chave Primária, ordenados por data de criação.
================================================================================*/

PROCEDURE TabelasPossuem_FK
IS
      v_restricao VARCHAR(8000);
      v_cr CHAR(2);
BEGIN
      v_cr := chr(13) || chr(10);

      FOR c_objeto IN (
            SELECT T.TABNAME TABLE_NAME
            FROM SYSCAT.TABLES T
            WHERE T.TABNAME IN (
                  SELECT DISTINCT R.TABNAME
                  FROM SYSCAT.REFERENCES R
                  WHERE R.TABSCHEMA=T.TABSCHEMA
                  AND R.OWNER=T.OWNER
                  AND R.OWNERTYPE=T.OWNERTYPE)
            AND T.TABSCHEMA=CURRENT SCHEMA
            AND T.OWNER=CURRENT USER
            AND T.OWNERTYPE='U'
            AND T.TYPE='T'
            ORDER BY VARCHAR_FORMAT(CREATE_TIME,
            'DD/MM/YYYY HH24:MI:SS:SSSSS') ASC)
      LOOP
            v_restricao := '';

            FOR c_tabela IN (
                  SELECT DISTINCT I.TABNAME TABLE_NAME,
                  I.INDNAME CONSTRAINT_NAME,
                  'PRIMARY KEY' CONSTRAINT_TYPE,
                  NULL R_TABLE_NAME, NULL R_CONSTRAINT_NAME
                  FROM SYSCAT.INDEXES I
                  WHERE I.INDSCHEMA=CURRENT SCHEMA
                  AND I.OWNER=CURRENT USER
                  AND I.OWNERTYPE='U'
                  AND I.TABNAME=c_objeto.TABLE_NAME
                  UNION
                  SELECT DISTINCT R.TABNAME, R.CONSTNAME,
                  'FOREIGN KEY' CONSTRAINT_TYPE,
                  R.REFTABNAME, R.REFKEYNAME
                  FROM SYSCAT.REFERENCES R
                  WHERE R.TABSCHEMA=CURRENT SCHEMA
                  AND R.OWNER=CURRENT USER
                  AND R.OWNERTYPE='U'
                  AND R.TABNAME=c_objeto.TABLE_NAME)
            LOOP

                  if c_tabela.CONSTRAINT_TYPE='FOREIGN KEY' then
                        v_restricao := v_restricao ||
                        c_tabela.CONSTRAINT_NAME || ' = '
                        || c_tabela.table_name || '(' ||
                        ColunasDaRestricaoDaTabela(
                        c_tabela.table_name,
                        c_tabela.CONSTRAINT_NAME) || ') --> ' ||
                        c_tabela.R_TABLE_NAME || '(' ||
                        ColunasDaRestricaoDaTabela(
```

```
                        c_tabela.R_TABLE_NAME,
                        c_tabela.R_CONSTRAINT_NAME) || ')' || v_cr;
            end if;

            END LOOP;

            DBMS_OUTPUT.put_line (v_restricao);

      END LOOP;

EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Um erro foi encontrado - '
                  ||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

17.15. O Procedimento TabelasPossuemSomente_FK_naoPK

```
/*================================================================================
Procedimento:     TabelasSomente_FK_naoPK
Autor:            Henrique Figueiredo de Souza
Data Criação:     27/02/2013
Descrição:
            Lista todas as tabelas que possuem SOMENTE Chave Estrangeira
            e não possuem Chave Primária, ordenados por data de criação.
================================================================================*/

PROCEDURE TabelasSomente_FK_naoPK
IS
      v_restricao VARCHAR(8000);
      v_cr CHAR(2);
BEGIN
      v_cr := chr(13) || chr(10);

      FOR c_objeto IN (
            SELECT T.TABNAME TABLE_NAME
            FROM (SELECT T1.TABNAME, T1.CREATE_TIME,
                  T1.TABSCHEMA, T1.OWNER, T1.OWNERTYPE
                  FROM SYSCAT.TABLES T1
                  WHERE T1.TABNAME NOT IN (
                        SELECT DISTINCT I.TABNAME
                        FROM SYSCAT.INDEXES I
                        WHERE I.INDSCHEMA=T1.TABSCHEMA
                        AND I.OWNER=T1.OWNER
                        AND I.OWNERTYPE=T1.OWNERTYPE)
                  AND T1.TABSCHEMA=CURRENT SCHEMA
                  AND T1.OWNER=CURRENT USER
                  AND T1.OWNERTYPE='U'
                  AND T1.TYPE='T') T
            WHERE T.TABNAME NOT IN (
                  SELECT DISTINCT R.TABNAME
                  FROM SYSCAT.REFERENCES R
                  WHERE R.TABSCHEMA=T.TABSCHEMA
                  AND R.OWNER=T.OWNER
                  AND R.OWNERTYPE=T.OWNERTYPE)
            ORDER BY VARCHAR_FORMAT(T.CREATE_TIME,
            'DD/MM/YYYY HH24:MI:SS:SSSSS') ASC)
      LOOP
            v_restricao := '';

            FOR c_tabela IN (
                  SELECT DISTINCT I.TABNAME TABLE_NAME,
```

```
                  I.INDNAME CONSTRAINT_NAME,
                  'PRIMARY KEY' CONSTRAINT_TYPE,
                  NULL R_TABLE_NAME, NULL R_CONSTRAINT_NAME
                  FROM SYSCAT.INDEXES I
                  WHERE I.INDSCHEMA=CURRENT SCHEMA
                  AND I.OWNER=CURRENT USER
                  AND I.OWNERTYPE='U'
                  AND I.TABNAME=c_objeto.TABLE_NAME
                  UNION
                  SELECT DISTINCT R.TABNAME, R.CONSTNAME,
                  'FOREIGN KEY' CONSTRAINT_TYPE,
                  R.REFTABNAME, R.REFKEYNAME
                  FROM SYSCAT.REFERENCES R
                  WHERE R.TABSCHEMA=CURRENT SCHEMA
                  AND R.OWNER=CURRENT USER
                  AND R.OWNERTYPE='U'
                  AND R.TABNAME=c_objeto.TABLE_NAME)
            LOOP

                  if c_tabela.CONSTRAINT_TYPE='R' then
                        v_restricao := v_restricao ||
                        c_tabela.CONSTRAINT_NAME || ' = '
                        || c_tabela.table_name || '(' ||
                        ColunasDaRestricaoDaTabela(c_tabela.table_name,
                        c_tabela.CONSTRAINT_NAME) || ') --> ' ||
                        c_tabela.R_TABLE_NAME || '(' ||
                        ColunasDaRestricaoDaTabela(c_tabela.R_TABLE_NAME,
                        c_tabela.R_CONSTRAINT_NAME) || ')' || v_cr;
            end if;

            END LOOP;

            DBMS_OUTPUT.put_line (v_restricao);

      END LOOP;

EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Um erro foi encontrado - '
                  ||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

17.16. O Procedimento DescomentarTabelasColunas

```
/*================================================================================
Procedimento:     DescomentarTabelasColunas
Autor:            Henrique Figueiredo de Souza
Data Criação:     27/02/2013
Descrição:
            Descomentar todas a tabelas e colunas do banco de dados
================================================================================*/

PROCEDURE DescomentarTabelasColunas
IS
      v_tabela_anterior VARCHAR(128);
      v_coluna VARCHAR(8000);
      v_cr CHAR(2);
      v_conta integer;
BEGIN
      v_cr := chr(13) || chr(10);
      v_tabela_anterior := '';
      v_conta := 0;
```

```
      FOR c_tabela IN (
            SELECT T.TABNAME TABLE_NAME, C.COLNAME COLUMN_NAME
            FROM SYSCAT.TABLES T, SYSCAT.COLUMNS C
            WHERE T.TABSCHEMA=C.TABSCHEMA AND T.TABNAME=C.TABNAME
            AND T.TABSCHEMA=CURRENT SCHEMA AND T.OWNER=CURRENT USER
            AND T.OWNERTYPE='U' AND T.TYPE='T'
            ORDER BY T.TABNAME, C.COLNO)
      LOOP

            if v_tabela_anterior<>c_tabela.table_name then
                  DBMS_OUTPUT.put_line (v_coluna);
                  v_coluna:='';
                  v_conta := 0;
            end if;

            if v_conta=0 then
                  v_coluna := 'comment on table ' || c_tabela.table_name ||
                  ' is ' || chr(39) || chr(39) || ';' || v_cr || v_coluna;
                  v_conta := 1;
            end if;

            v_coluna := v_coluna || 'comment on column ' ||
            c_tabela.table_name ||  '.' || c_tabela.column_name
            || ' is ' || chr(39) || chr(39) || ';' || v_cr;

            v_tabela_anterior := c_tabela.table_name;

      END LOOP;

      DBMS_OUTPUT.put_line (v_coluna);

EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Um erro foi encontrado - '
                  ||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

18. Os Procedimentos Qlikview para o Gupta SQLBase

```
SET SERVER SERVER1/12345678;
SHOW DATABASES ON SERVER SERVER1;
CONNECT CAPACITACAO;
SHOW CONNECT;

CREATE TABLE DUMMY(ID INTEGER);
INSERT INTO DUMMY(ID) VALUES(0);

ERASE QV_ContaPalavra;
ERASE QV_LimitaTamanhoTexto;
ERASE QV_AplicarMapa;
ERASE QV_ColunasDaRestricaoDaTabela;
ERASE QV_ListarTabelaRestricao;
ERASE QV_MontarLoad;
ERASE QV_ContarLinhasTabela;
ERASE QV_MontarTodosMappingLoad;
ERASE QV_MontarTodosLoad;
ERASE QV_ListarTabelasColunas;
ERASE QV_ListarTabelasRestricoes;
ERASE QV_TabelasNaoPossuem_PK_nemFK;
ERASE QV_TabelasSomente_PK_naoFK;
ERASE QV_TabelasPossuem_FK;
```

```
ERASE QV_TabelasSomente_FK_naoPK;
ERASE QV_DescomentarTabelasColunas;

EXECUTE QV_ContaPalavra
\
ALFA BETA GAMA TETA,,
/

EXECUTE QV_LimitaTamanhoTexto
\
ALFA BETA GAMA TETA,2,,
/


EXECUTE QV_AplicarMapa
\
PERGUNTA,CODIGO_TIPO_PERGUNTA,,
/


EXECUTE QV_ColunasDaRestricaoDaTabela
\
PERGUNTA,PERGUNTA_PK,,
/

EXECUTE QV_ListarTabelaRestricao
\
PERGUNTA,,
/

EXECUTE QV_MontarLoad
\
PERGUNTA,Henrique Figueiredo de Souza, 1, 1,
/

EXECUTE QV_ContarLinhasTabela
\
PERGUNTA,,
/

EXECUTE QV_MontarTodosMappingLoad
\
Henrique Figueiredo de Souza, 0, 1,,
/

EXECUTE QV_MontarTodosLoad
\
Henrique Figueiredo de Souza, 0,,
/

EXECUTE QV_ListarTabelasColunas
\
0,,
/

EXECUTE QV_ListarTabelasRestricoes
\
,,
/

EXECUTE QV_TabelasNaoPossuem_PK_nemFK
\
,,
/
```

```
EXECUTE QV_TabelasSomente_PK_naoFK
\
,,
/

EXECUTE QV_TabelasPossuem_FK
\
,,
/

EXECUTE QV_TabelasSomente_FK_naoPK
\
,,
/

EXECUTE QV_DescomentarTabelasColunas
\
,,
/
```

18.1. As Visões para auxiliar os procedimentos

```
DROP VIEW QV_TabelasNaoPossuem_PK_nemFK;
DROP VIEW QV_TabelasPossuemSo_PK_naoFK;
DROP VIEW QV_TabelasPossuem_FK;
DROP VIEW QV_TabelasPossuemSo_FK_naoPK;

--LISTAR TODAS AS TABELAS QUE NÃO POSSUEM PK E NEM FK
CREATE VIEW QV_TabelasNaoPossuem_PK_nemFK
AS
SELECT NAME
FROM SYSTABLES
WHERE TYPE='T' AND SYSTEM='N'
AND NAME NOT IN (select DISTINCT NAME
            from SYSTABCONSTRAINTS
            WHERE TYPE IN ('P','F'));

/* LISTAR TODAS AS TABELAS QUE POSSUEM PK e FK */
CREATE VIEW QV_TabelasPossuem_PK_e_FK
AS
SELECT NAME
FROM SYSTABLES
WHERE TYPE='T' AND SYSTEM='N'
AND NAME IN (select DISTINCT NAME
            from SYSTABCONSTRAINTS
            WHERE TYPE IN ('P','F'));


/* LISTAR TODAS AS TABELAS QUE POSSUEM SOMENTE PK E NÃO POSSUEM FK */
CREATE VIEW QV_TabelasPossuemSo_PK_naoFK
AS
SELECT NAME
FROM QV_TabelasPossuem_PK_e_FK
WHERE NAME NOT IN (select DISTINCT NAME
from SYSTABCONSTRAINTS WHERE TYPE='F');

/* LISTAR TODAS AS TABELAS QUE POSSUEM FK */
CREATE VIEW QV_TabelasPossuem_FK
AS
SELECT NAME
FROM SYSTABLES
```

```
WHERE TYPE='T' AND SYSTEM='N'
AND NAME IN (select DISTINCT NAME
             from SYSTABCONSTRAINTS
             WHERE TYPE='F');

--LISTAR TODAS AS TABELAS QUE NÃO POSSUEM PK
CREATE VIEW QV_TabelasNaoPossuem_PK
AS
SELECT NAME
FROM SYSTABLES
WHERE TYPE='T' AND SYSTEM='N'
AND NAME NOT IN (select DISTINCT NAME
             from SYSTABCONSTRAINTS
             WHERE TYPE='P');


/* LISTAR TODAS AS TABELAS QUE POSSUEM SOMENTE FK E NÃO POSSUEM PK */
CREATE VIEW QV_TabelasPossuemSo_FK_naoPK
AS
SELECT NAME
FROM QV_TabelasNaoPossuem_PK
WHERE NAME IN (select DISTINCT NAME
             from SYSTABCONSTRAINTS
             WHERE TYPE='F');
```

18.2. O Procedimento QV_ContaPalavra

```
/*================================================================================
Procedimento:  QV_ContaPalavra
Autor:    Henrique Figueiredo de Souza
Data Criação:  13/03/2013
Parametros:
    p_str    Texto a ser contado as palavras.
Descrição:
    Conta as palavras de um texto.
================================================================================*/

STORE QV_ContaPalavra
PROCEDURE: QV_ContaPalavra
Parameters
  Long String: sTexto
  Receive Number: nPalavras
Local variables
  Sql Handle: hSqlCur1
  String: sSql
  Boolean: bOk
  Number: nRetVal
  Number: nI
  Number: nCodigo
  Number: nTamanho
  Boolean: bDentro_uma_palavra
Actions
  Call SqlConnect ( hSqlCur1 )
  Set bDentro_uma_palavra = FALSE
  Set nI = 0
  Set nTamanho = 0
  Set nCodigo = 0
  Set nPalavras = 0
  Set sSql = 'SELECT @COALESCE(@LENGTH(:sTexto),0) FROM DUMMY into :nTamanho'
  Set bOk = SqlPrepareAndExecute ( hSqlCur1, sSql )
  If bOk
    Call SqlFetchNext ( hSqlCur1, nRetVal )
    If nRetVal = FETCH_Ok
```

```
      While nI <= (nTamanho + 1)
        Set sSql = 'SELECT @CODE(@MID(:sTexto, :nI, 1)) FROM DUMMY into
:nCodigo'
        Set bOk = SqlPrepareAndExecute ( hSqlCur1, sSql )
        If bOk
          Call SqlFetchNext ( hSqlCur1, nRetVal )
          If nRetVal = FETCH_Ok
            If (nCodigo < 33) OR (nI > nTamanho)
              If bDentro_uma_palavra = TRUE
                Set nPalavras = nPalavras + 1
                Set bDentro_uma_palavra = FALSE
            Else
              Set bDentro_uma_palavra = TRUE
        Set nI = nI + 1
  Call SqlDisconnect ( hSqlCur1 )
\
```

18.3. O Procedimento QV_LimitaTamanhoTexto

```
/*================================================================================
Procedimento:  QV_LimitaTamanhoTexto
Autor:    Henrique Figueiredo de Souza
Data Criação:  13/03/2013
Parametros:
    p_str    Texto a ser limitado.
    p_limite  Limite de palavras por linhas.
Descrição:
    Limita o texto fornecido a uma determinada quantidades
    de palavras por linha.
================================================================================*/

STORE QV_LimitaTamanhoTexto
PROCEDURE: QV_LimitaTamanhoTexto
  Parameters
    Long String: pTexto
    Number: pLimite
    Receive Long String: pRetorno
  Local variables
    Sql Handle: hSqlCur1
    String: sSql
    Boolean: bOk
    Number: nRetVal
    Number: nPalavras
    Number: nTamanho
    Boolean: bDentro_uma_palavra
    Number: nI
    Number: nCodigo
    String: sParte
    String: sCR
  Actions
    Call SqlConnect ( hSqlCur1 )
    Set nPalavras = 0
    Set nTamanho = 0
    Set bDentro_uma_palavra = FALSE
    Set pRetorno = ''
    Set nI = 0
    Set nCodigo = 0
    Set sSql = 'SELECT @COALESCE(@LENGTH(:pTexto),0) FROM DUMMY into :nTamanho'
    Set bOk = SqlPrepareAndExecute ( hSqlCur1, sSql )
    If bOk
      Call SqlFetchNext ( hSqlCur1, nRetVal )
      If nRetVal = FETCH_Ok
        While nI <= (nTamanho + 1)
          Set sSql = 'SELECT @MID(:pTexto, :nI, 1) FROM DUMMY into :sParte'
```

```
          Set bOk = SqlPrepareAndExecute ( hSqlCur1, sSql )
          If bOk
            Call SqlFetchNext ( hSqlCur1, nRetVal )
            If nRetVal = FETCH_Ok
              Set pRetorno = pRetorno || sParte
              Set sSql = 'SELECT @CODE(@MID(:pTexto, :nI, 1)) FROM DUMMY into
:nCodigo'
              Set bOk = SqlPrepareAndExecute ( hSqlCur1, sSql )
              If bOk
                Call SqlFetchNext ( hSqlCur1, nRetVal )
                If nRetVal = FETCH_Ok
                  If (nCodigo < 33) OR (nI > nTamanho)
                    If bDentro_uma_palavra = TRUE
                      Set nPalavras = nPalavras + 1
                      Set bDentro_uma_palavra = FALSE
                      If nPalavras = pLimite
                        Set sSql = 'SELECT @CHAR(13) || @CHAR(10) FROM DUMMY
into :sCR'
                        Call SqlImmediate( sSql )
                        Set pRetorno = pRetorno || sCR
                        Set nPalavras = 0
                  Else
                    Set bDentro_uma_palavra = TRUE
          Set nI = nI + 1
    Call SqlDisconnect ( hSqlCur1 )
\
```

18.4. O Procedimento QV_AplicarMapa

```
/*================================================================================
Procedimento:  QV_AplicarMapa
Autor:    Henrique Figueiredo de Souza
Data Criação:  13/03/2013
Parametros:
    p_tabela  Nome da tabela referenciada pela FK.
    p_coluna  Nome da coluna que possui uma FK.
Descrição:
    Aplicar o mapa carregado, onde a coluna é uma
    Chave Estrangeira e tabela é a tabela referenciada pela
    chave estrangeira da coluna.
================================================================================*/

STORE QV_AplicarMapa
PROCEDURE: QV_AplicarMapa
  Parameters
    String: pTabela
    String: pColuna
    Receive Long String: pTexto
  Local variables
    Sql Handle: hSqlCur1
    String: sSql
    Boolean: bOk
    Number: nRetVal
    String: sTabela
    String: sTabelaMapa
    String: sTAB
  Actions
    Call SqlConnect ( hSqlCur1 )
    Set pTexto = ''
    ! verifica se a coluna da tabela é uma Chave Estrangeira
    Set sSql = 'select REFDTBNAME \
        from SYSFKCONSTRAINTS \
        WHERE NAME=:pTabela \
        AND REFSCOLUMN=:pColuna INTO :sTabelaMapa'
```

```
    Set bOk = SqlPrepareAndExecute ( hSqlCur1, sSql )
    If bOk
      Call SqlFetchNext ( hSqlCur1, nRetVal )
      If nRetVal = FETCH_Ok
        Set sSql = 'SELECT NAME \
            FROM QV_TabelasPossuem_PK_e_FK \
            WHERE NAME NOT IN (select DISTINCT NAME \
              from SYSTABCONSTRAINTS WHERE TYPE=''F'') \
            AND NAME=:sTabelaMapa INTO :sTabela'
        Set bOk = SqlPrepareAndExecute ( hSqlCur1, sSql )
        If bOk
          Call SqlFetchNext ( hSqlCur1, nRetVal )
          If nRetVal = FETCH_Ok
            Set sSql = 'SELECT @CHAR(39) FROM DUMMY into :sTAB'
            Call SqlImmediate( sSql )
            Set pTexto = 'ApplyMap(' || sTAB || 'Mapa' || sTabelaMapa || sTAB ||
',' || pColuna || ')'
    Call SqlDisconnect ( hSqlCur1 )
\
```

18.5. O Procedimento QV_ColunasDaRestricaoDaTabela

```
/*================================================================================
Procedimento:  QV_ColunasDaRestricaoDaTabela
Autor:    Henrique Figueiredo de Souza
Data Criação:  13/03/2013
Parametros:
    p_tabela  Nome da tabela referenciada pela FK.
    p_restricao  Nome da restrição.
Descrição:
    Retorna as colunas de uma tabela e sua restrição.
================================================================================*/

STORE QV_ColunasDaRestricaoDaTabela
PROCEDURE: QV_ColunasDaRestricaoDaTabela
  Parameters
    String: pTabela
    String: pRestricao
    Receive Long String: pTexto
  Local variables
    Sql Handle: hSqlCur1
    String: sSql
    Boolean: bOk
    Number: nRetVal
    String: sNomeColuna
    String: sColuna
  Actions
    Call SqlConnect ( hSqlCur1 )
    Set sColuna = ''
    Set sSql = 'select P.COLNAME \
        FROM SYSINDEXES I, SYSPKCONSTRAINTS P \
        WHERE I.UNIQUERULE=''U'' \
        AND I.TBNAME=P.NAME \
        AND I.TBNAME=:pTabela \
        AND I.NAME=:pRestricao \
        UNION \
        select REFSCOLUMN \
        from SYSFKCONSTRAINTS \
        WHERE NAME=:pTabela \
        AND CONSTRAINT=:pRestricao INTO :sNomeColuna'
    Set bOk = SqlPrepareAndExecute ( hSqlCur1, sSql )
    If bOk
```

```
      Call SqlFetchNext ( hSqlCur1, nRetVal )
      While nRetVal != FETCH_EOF
        Set sColuna = sColuna || ', ' || sNomeColuna
        Call SqlFetchNext ( hSqlCur1, nRetVal )
    Set sSql = 'SELECT @MID(:sColuna, 2, @LENGTH(:sColuna)) FROM DUMMY into
:sColuna'
    Call SqlImmediate( sSql )
    Set pTexto = sColuna
    Call SqlDisconnect ( hSqlCur1 )
\
```

18.6. O Procedimento QV_ListarTabelaRestricao

```
/*==============================================================================
Procedimento:  QV_ListarTabelaRestricao
Autor:    Henrique Figueiredo de Souza
Data Criação:  13/03/2013
Parametros:
    p_tabela  Nome da tabela.
Descrição:
    Lista restrições de uma tabela.
==============================================================================*/

STORE QV_ListarTabelaRestricao
PROCEDURE: QV_ListarTabelaRestricao
  Parameters
    String: pTabela
    Receive Long String: pTexto
  Local variables
    Sql Handle: hSqlCur1
    String: sSql
    Boolean: bOk
    Number: nRetVal
    String: sRestricao
    String: sCR
    String: sNomeTabela
    String: sNomeRestricao
    String: sTipoRestricao
    String: sRestricaoRef
    String: sTabelaRef
    Number: nTam
    String: sRetorno1
    String: sRetorno2
  Actions
    Call SqlConnect ( hSqlCur1 )
    Set sRestricao = ''
    Set sSql = 'SELECT @CHAR(13) || @CHAR(10) FROM DUMMY into :sCR'
    Call SqlImmediate( sSql )
    Set sSql = 'SELECT DISTINCT TBNAME AS TABLE_NAME, NAME AS CONSTRAINT_NAME, \
        ''PRIMARY KEY'' AS CONSTRAINT_TYPE, NULL AS CONSTRAINT_REF, NULL AS
REFTABNAME \
        FROM SYSINDEXES \
        WHERE UNIQUERULE=''U'' \
        AND TBNAME=:pTabela \
        UNION \
        SELECT DISTINCT F.NAME, F.CONSTRAINT, ''FOREIGN KEY'', I.NAME,
F.REFDTBNAME \
        FROM SYSFKCONSTRAINTS F, SYSINDEXES I \
        WHERE F.REFDTBNAME=I.TBNAME \
        AND F.CREATOR=I.CREATOR \
        AND I.UNIQUERULE=''U'' \
        AND F.NAME=:pTabela \
        ORDER BY 2 DESC \
        INTO :sNomeTabela, :sNomeRestricao, \
```

```
        :sTipoRestricao, :sRestricaoRef, :sTabelaRef'
    Set bOk = SqlPrepareAndExecute ( hSqlCur1, sSql )
    If bOk
      Call SqlFetchNext ( hSqlCur1, nRetVal )
      While nRetVal != FETCH_EOF
        If sTipoRestricao = 'PRIMARY KEY'
          ! Call QV_ColunasDaRestricaoDaTabela(sNomeTabela, sNomeRestricao,
sRetorno1)
          Call SqlRetrieve( hSqlCur1, 'QV_ColunasDaRestricaoDaTabela',
':sNomeTabela, :sNomeRestricao, :sRetorno1', ':sRetorno1' )
          Call SqlExecute( hSqlCur1 )
          Set sRestricao = sRestricao || sNomeTabela || '(' || sRetorno1 || ')'
|| sCR
        Else
          ! Call QV_ColunasDaRestricaoDaTabela(sNomeTabela, sNomeRestricao,
sRetorno1)
          Call SqlRetrieve( hSqlCur1, 'QV_ColunasDaRestricaoDaTabela',
':sNomeTabela, :sNomeRestricao, :sRetorno1', ':sRetorno1' )
          Call SqlExecute( hSqlCur1 )
          ! Call QV_ColunasDaRestricaoDaTabela(sTabelaRef, sRestricaoRef,
sRetorno2)
          Call SqlRetrieve( hSqlCur1, 'QV_ColunasDaRestricaoDaTabela',
':sTabelaRef, :sRestricaoRef, :sRetorno2', ':sRetorno2' )
          Call SqlExecute( hSqlCur1 )
          Set sRestricao = sRestricao || sNomeTabela || '(' || sRetorno1 || ') -
-> ' || sTabelaRef || '(' || sRetorno2 || ')' || sCR
        Call SqlFetchNext ( hSqlCur1, nRetVal )
      Set sSql = 'SELECT @LENGTH(:sRestricao)-2 FROM DUMMY into :nTam'
      Call SqlImmediate( sSql )
      If nTam > 0
        Set sSql = 'SELECT @MID(:sRestricao, 0, :nTam) FROM DUMMY into
:sRestricao'
        Call SqlImmediate( sSql )
    Set pTexto = sRestricao
    Call SqlDisconnect ( hSqlCur1 )
\
```

18.7. O Procedimento QV_MontarLoad

```
/*==============================================================================
Procedimento:  QV_MontarLoad
Autor:    Henrique Figueiredo de Souza
Data Criação:  13/03/2013
Parametros:
    p_tabela  Nome da tabela a gerar o LOAD.
    p_autor    Nome do autor a ser mostrado no comentário.
    p_mapeada  TRUE para mapping load, FALSE para sem mapping load
    p_2_colunas  Limita a saida a somente a duas colunas.
Descrição:
    Monta um LOAD em formato qlikview de uma tabela
==============================================================================*/

STORE QV_MontarLoad
PROCEDURE: QV_MontarLoad
  Parameters
    String: pTabela
    String: pAutor
    Boolean: pMapeada
    Boolean: p2Colunas
    Receive Long String: pRetorno
  Local variables
    Sql Handle: hSqlCur1
    String: sSql
    Boolean: bOk
```

```
    Number: nRetVal
    String: sCR
    String: sTAB
    String: sTabela
    String: sColuna
    String: sComenta
    String: sRotulo
    String: sRotuloTabela
    String: sMapa
    String: sColunaNome
    Number: nConta
    String: sRetorno
    String: sNomeTabela
    String: sNomeColuna
    Number: sOrdemColuna
    String: sComentario
    String: sComentarioTabela
    String: sDataAtual
    String: sRetorno1
  Actions
    Call SqlConnect ( hSqlCur1 )
    Set sSql = 'SELECT @CHAR(13) || @CHAR(10) FROM DUMMY into :sCR'
    Call SqlImmediate( sSql )
    Set sSql = 'SELECT @CHAR(9) FROM DUMMY into :sTAB'
    Call SqlImmediate( sSql )
    Set sRetorno = ''
    Set sColuna = ''
    Set sMapa = ''
    Set sRotulo = ''
    Set nConta = 0
    Set sSql = 'SELECT T.NAME AS TABLE_NAME, \
        C.NAME AS COLUMN_NAME, C.COLNO, \
        C.REMARKS AS COLUMN_REMARKS, \
        T.REMARKS AS TABLE_REMARKS \
        FROM SYSTABLES T, SYSCOLUMNS C \
        WHERE T.NAME=C.TBNAME \
        AND T.TYPE=''T'' AND T.SYSTEM=''N'' \
        AND T.NAME=:pTabela \
        ORDER BY T.NAME, C.COLNO \
        INTO :sNomeTabela, :sNomeColuna, \
        :sOrdemColuna, :sComentario, :sComentarioTabela'
    Set bOk = SqlPrepareAndExecute ( hSqlCur1, sSql )
    If bOk
      Call SqlFetchNext ( hSqlCur1, nRetVal )
      Loop
        If nRetVal = FETCH_EOF
          Break
        Set sTabela = sNomeTabela
        Set sRotulo = sComentario
        ! Call QV_LimitaTamanhoTexto(sComentarioTabela, 10, sRotuloTabela)
        Call SqlRetrieve( hSqlCur1, 'QV_LimitaTamanhoTexto',
':sComentarioTabela, 10, :sRotuloTabela', ':sRotuloTabela' )
        Call SqlExecute( hSqlCur1 )
        ! Call QV_AplicarMapa(sTabela, sNomeColuna, sMapa)
        Call SqlRetrieve( hSqlCur1, 'QV_AplicarMapa', ':sTabela, :sNomeColuna,
:sMapa', ':sMapa' )
        Call SqlExecute( hSqlCur1 )
        If sMapa!=''
          Set sColunaNome = sMapa
        Else
          Set sColunaNome = sNomeColuna
        If sRotulo=''
          Set sColuna = sColuna || ', ' || sCR ||  '     ' || sColunaNome
        Else
```

```
          Set sColuna = sColuna || ', ' || sCR || '     ' || sColunaNome || ' as
[' || sRotulo || ']'
        If p2Colunas
          Set nConta = nConta + 1
          If nConta=2
            Break
        Call SqlFetchNext ( hSqlCur1, nRetVal )
    If sColuna!=''
      Set sSql = 'SELECT @MID(:sColuna, 2, @LENGTH(:sColuna)) FROM DUMMY into
:sColuna'
      Call SqlImmediate( sSql )
      Set sComenta = sCR ||
'/*=============================================================================
' || sCR
      If pMapeada
        Set sComenta = sComenta || 'Procedimento:' || sTAB || 'Carga mapeada em
memória do arquivo QVD' || sCR
      Else
        Set sComenta = sComenta || 'Procedimento:' || sTAB || 'Carga em memória
do arquivo QVD' || sCR
      Set sComenta = sComenta || 'ArquivoQVD:' || sTAB || sTAB || sTabela || sCR
      Set sComenta = sComenta || 'Autor:' || sTAB || sTAB || sTAB || pAutor ||
sCR
      Set sSql = 'SELECT @DATETOCHAR(SYSDATETIME, ''dd/mm/yyyy'') FROM DUMMY
INTO :sDataAtual'
      Call SqlImmediate( sSql )
      Set sComenta = sComenta || 'Data Criação:' || sTAB || sDataAtual || sCR
      If sRotuloTabela!=''
        Set sComenta = sComenta || 'Descrição:' || sTAB || sRotuloTabela || sCR
      Set sRetorno = sRetorno || sComenta || 'Restrições:' || sCR
      ! Call QV_ListarTabelaRestricao(pTabela, sRetorno1)
      Call SqlRetrieve( hSqlCur1, 'QV_ListarTabelaRestricao', ':pTabela,
:sRetorno1', ':sRetorno1' )
      Call SqlExecute( hSqlCur1 )
      Set sRetorno = sRetorno || sRetorno1 || sCR
      Set sRetorno = sRetorno ||
'=============================================================================*/
' || sCR
      If pMapeada
        Set sRetorno = sRetorno || 'Mapa' || sTabela || ':' || sCR || 'Mapping
LOAD' || sColuna || sCR || 'FROM' || sCR || '$(vDiretorio)' || sTabela || '.qvd'
|| sCR || '(qvd);' || sCR
      Else
        Set sRetorno = sRetorno || sTabela || ':' || sCR || 'LOAD' || sColuna ||
sCR || 'FROM' || sCR || '$(vDiretorio)' || sTabela || '.qvd' || sCR || '(qvd);'
|| sCR
    Set pRetorno = sRetorno
    Call SqlDisconnect ( hSqlCur1 )
\
```

18.8. O Procedimento QV_ContarLinhasTabela

```
/*=============================================================================
Procedimento:  QV_ContarLinhasTabela
Autor:    Henrique Figueiredo de Souza
Data Criação:  13/03/2013
Parametros:
    p_tabela  Nome da tabela referenciada pela FK.
Descrição:
    Retorna a quantidade de linhas de uma tabela.
=============================================================================*/
```

```
STORE QV_ContarLinhasTabela
PROCEDURE: QV_ContarLinhasTabela
  Parameters
    String: pTabela
    Receive Number: pTotal
  Local variables
    Sql Handle: hSqlCur1
    String: sSql
    Boolean: bOk
    Number: nRetVal
  Actions
    Call SqlConnect ( hSqlCur1 )
    Set sSql = 'SELECT COUNT(*) FROM ' || pTabela || ' INTO :pTotal'
    Set bOk = SqlPrepareAndExecute ( hSqlCur1, sSql )
    If bOk
      Call SqlFetchNext ( hSqlCur1, nRetVal )
      If nRetVal != FETCH_Ok
        Set pTotal = -1
    Call SqlDisconnect ( hSqlCur1 )
\
```

18.9. O Procedimento QV_MontarTodosMappingLoad

```
/*==============================================================================
Procedimento:  QV_MontarTodosMappingLoad
Autor:    Henrique Figueiredo de Souza
Data Criação:  13/03/2013
Parametros:
    p_autor    Nome do autor a ser mostrado no comentário.
    p_vazia    Se 1 verificar se a tabela está vazia.
    p_2_colunas  Limita a saida a somente a duas colunas.
Descrição:
    Monta todos os Mapping LOAD em formato qlikview
    de todas as tabelas que possuem SOMENTE Chave Primária.
==============================================================================*/

STORE QV_MontarTodosMappingLoad
PROCEDURE: QV_MontarTodosMappingLoad
  Parameters
    String: pAutor
    Boolean: pVazia
    Boolean: p2Colunas
    Receive String: pRetorno
  Local variables
    Sql Handle: hSqlCur1
    String: sSql
    Boolean: bOk
    Number: nRetVal
    String: sTabela
    Number: nConta
    String: sRetorno
    String: sRetorno1
  Actions
    Call SqlConnect ( hSqlCur1 )
    Set sRetorno = ''
    Set sSql = 'SELECT NAME \
        FROM QV_TabelasPossuem_PK_e_FK \
        WHERE NAME NOT IN (select DISTINCT NAME \
        from SYSTABCONSTRAINTS WHERE TYPE=''F'') INTO :sTabela'
    Set bOk = SqlPrepareAndExecute ( hSqlCur1, sSql )
    If bOk
      Call SqlFetchNext ( hSqlCur1, nRetVal )
      While nRetVal != FETCH_EOF
        If pVazia
```

```
          Set sSql = 'SELECT COUNT(*) FROM ' || sTabela || ' INTO :nConta'
          Set bOk = SqlPrepareAndExecute ( hSqlCur1, sSql )
          If bOk
            Call SqlFetchNext ( hSqlCur1, nRetVal )
            If nRetVal = FETCH_Ok
              If nConta > 0
                ! Call QV_MontarLoad(sTabela, pAutor, TRUE, p2Colunas,
sRetorno1)
                Call SqlRetrieve( hSqlCur1, 'QV_MontarLoad', ':sTabela, :pAutor,
TRUE, :p2Colunas, :sRetorno1', ':sRetorno1' )
                Call SqlExecute( hSqlCur1 )
                Set sRetorno = sRetorno || sRetorno1
        Else
          ! Call QV_MontarLoad(sTabela, pAutor, TRUE, p2Colunas, sRetorno1)
          Call SqlRetrieve( hSqlCur1, 'QV_MontarLoad', ':sTabela, :pAutor, TRUE,
:p2Colunas, :sRetorno1', ':sRetorno1' )
          Call SqlExecute( hSqlCur1 )
          Set sRetorno = sRetorno || sRetorno1
        Call SqlFetchNext ( hSqlCur1, nRetVal )
    Set pRetorno = sRetorno
    Call SqlDisconnect ( hSqlCur1 )
\
```

18.10. O Procedimento QV_MontarTodosLoad

```
/*================================================================================
Procedimento:  QV_MontarTodosLoad
Autor:    Henrique Figueiredo de Souza
Data Criação:  13/03/2013
Parametros:
    p_autor    Nome do autor a ser mostrado no comentário.
    p_vazia    Se TRUE verificar se a tabela está vazia.
Descrição:
    Monta todos os LOAD em formato qlikview de todas as tabelas
================================================================================*/

STORE QV_MontarTodosLoad
PROCEDURE: QV_MontarTodosLoad
  Parameters
    String: pAutor
    Boolean: pVazia
    Receive String: pRetorno
  Local variables
    Sql Handle: hSqlCur1
    String: sSql
    Boolean: bOk
    Number: nRetVal
    String: sTabela
    Number: nConta
    String: sRetorno
    String: sRetorno1
  Actions
    Call SqlConnect ( hSqlCur1 )
    Set sRetorno = ''
    Set sSql = 'SELECT T.NAME \
        FROM SYSTABLES T \
        WHERE T.TYPE=''T'' AND T.SYSTEM=''N'' \
        AND T.NAME<>''DUMMY'' \
        ORDER BY T.NAME INTO :sTabela'
    Set bOk = SqlPrepareAndExecute ( hSqlCur1, sSql )
    If bOk
      Call SqlFetchNext ( hSqlCur1, nRetVal )
      While nRetVal != FETCH_EOF
        If pVazia
```

```
          Set sSql = 'SELECT COUNT(*) FROM ' || sTabela || ' INTO :nConta'
          Set bOk = SqlPrepareAndExecute ( hSqlCur1, sSql )
          If bOk
            Call SqlFetchNext ( hSqlCur1, nRetVal )
            If nRetVal = FETCH_Ok
              If nConta > 0
                ! Call QV_MontarLoad(sTabela, pAutor, FALSE, FALSE, sRetorno1)
                Call SqlRetrieve( hSqlCur1, 'QV_MontarLoad', ':sTabela, :pAutor,
FALSE, FALSE, :sRetorno1', ':sRetorno1' )
                Call SqlExecute( hSqlCur1 )
                Set sRetorno = sRetorno || sRetorno1
        Else
          ! Call QV_MontarLoad(sTabela, pAutor, FALSE, FALSE, sRetorno1)
          Call SqlRetrieve( hSqlCur1, 'QV_MontarLoad', ':sTabela, :pAutor,
FALSE, FALSE, :sRetorno1', ':sRetorno1' )
          Call SqlExecute( hSqlCur1 )
          Set sRetorno = sRetorno || sRetorno1
        Call SqlFetchNext ( hSqlCur1, nRetVal )
    Set pRetorno = sRetorno
    Call SqlDisconnect ( hSqlCur1 )
\
```

18.11. O Procedimento QV_ListarTabelasColunas

```
/*================================================================================
Procedimento:  QV_ListarTabelasColunas
Autor:    Henrique Figueiredo de Souza
Data Criação:  13/03/2013
Parametros:
    p_comentario  Se TRUE incluir comando  para gerar comentario.
Descrição:
    Lista todas a tabelas e colunas do banco de dados
================================================================================*/

STORE QV_ListarTabelasColunas
PROCEDURE: QV_ListarTabelasColunas
  Parameters
    Boolean: pComentario
    Receive Long String: pRetorno
  Local variables
    Sql Handle: hSqlCur1
    String: sSql
    Boolean: bOk
    Number: nRetVal
    Long String: sRetorno
    String: sTabelaAnterior
    String: sNomeTabela
    String: sNomeColuna
    String: sColuna
    String: sCR
    Number: nConta
    String: sAPOSTROFO
    String: sTAB
  Actions
    Call SqlConnect ( hSqlCur1 )
    Set sTabelaAnterior=''
    Set sRetorno=''
    Set sSql = 'SELECT @CHAR(13) || @CHAR(10) FROM DUMMY into :sCR'
    Call SqlImmediate( sSql )
    Set sSql = 'SELECT @CHAR(9) || @CHAR(10) FROM DUMMY into :sTAB'
    Call SqlImmediate( sSql )
    Set sSql = 'SELECT @CHAR(39) || @CHAR(10) FROM DUMMY into :sAPOSTROFO'
    Call SqlImmediate( sSql )
    Set sSql = 'SELECT T.NAME AS TABLE_NAME, \
```

```
        C.NAME AS COLUMN_NAME \
        FROM SYSTABLES T, SYSCOLUMNS C \
        WHERE T.NAME=C.TBNAME \
        AND T.TYPE=''T'' AND T.SYSTEM=''N'' \
        AND T.NAME<>''DUMMY'' \
        ORDER BY T.NAME, C.COLNO INTO :sNomeTabela, :sNomeColuna'
    Set bOk = SqlPrepareAndExecute ( hSqlCur1, sSql )
    If bOk
      Call SqlFetchNext ( hSqlCur1, nRetVal )
      While nRetVal != FETCH_EOF
        If sTabelaAnterior!=sNomeTabela
          If sColuna!=''
            Set sRetorno = sRetorno || sColuna || sCR
          Set sColuna=''
          Set nConta=0
        If pComentario
          If nConta=0
            Set sColuna = 'comment on table ' || sNomeTabela || ' is ' ||
sAPOSTROFO || sTAB || sTAB || sAPOSTROFO || ';' || sCR || sColuna
            Set nConta = 1
          Set sColuna = sColuna || 'comment on column ' || sNomeTabela || '.' ||
sNomeColuna || ' is ' || sAPOSTROFO || sTAB || sTAB || sAPOSTROFO || ';' || sCR
        Else
          Set sColuna = sColuna || sNomeTabela || '.' || sNomeColuna || sCR
        Set sTabelaAnterior = sNomeTabela
        Call SqlFetchNext ( hSqlCur1, nRetVal )
    Set pRetorno = sRetorno || sColuna || sCR
    Call SqlDisconnect ( hSqlCur1 )
\
```

18.12. O Procedimento QV_ListarTabelasRestricoes

```
/*================================================================================
Procedimento:  QV_ListarTabelasRestricoes
Autor:    Henrique Figueiredo de Souza
Data Criação:  13/03/2013
Descrição:
    Lista todas as tabelas e suas restrições.
================================================================================*/

STORE QV_ListarTabelasRestricoes
PROCEDURE: QV_ListarTabelasRestricoes
  Parameters
    Receive Long String: pRetorno
  Local variables
    Sql Handle: hSqlCur1
    String: sSql
    Boolean: bOk
    Number: nRetVal
    Long String: sRetorno
    String: sTabelaAnterior
    String: sRestricao
    String: sCR
    String: sNomeTabela
    String: sNomeRestricao
    String: sTipoRestricao
    String: sRestricaoRef
    String: sTabelaRef
    String: sRetorno1
    String: sRetorno2
  Actions
    Call SqlConnect ( hSqlCur1 )
    Set sTabelaAnterior=''
    Set sRetorno=''
```

```
    Set sSql = 'SELECT @CHAR(13) || @CHAR(10) FROM DUMMY into :sCR'
    Call SqlImmediate( sSql )
    Set sSql = 'SELECT DISTINCT TBNAME AS TABLE_NAME, \
        NAME AS CONSTRAINT_NAME, \
        ''PRIMARY KEY'' AS CONSTRAINT_TYPE, \
        NULL AS CONSTRAINT_REF, NULL AS REFTABNAME \
        FROM SYSINDEXES \
        WHERE UNIQUERULE=''U'' \
        UNION \
        SELECT DISTINCT F.NAME, F.CONSTRAINT, \
        ''FOREIGN KEY'', I.NAME, F.REFDTBNAME \
        FROM SYSFKCONSTRAINTS F, SYSINDEXES I \
        WHERE F.REFDTBNAME=I.TBNAME \
        AND F.CREATOR=I.CREATOR \
        AND I.UNIQUERULE=''U'' \
        ORDER BY 2 DESC \
        INTO :sNomeTabela, :sNomeRestricao, \
        :sTipoRestricao, :sRestricaoRef, :sTabelaRef'
    Set bOk = SqlPrepareAndExecute ( hSqlCur1, sSql )
    If bOk
      Call SqlFetchNext ( hSqlCur1, nRetVal )
      While nRetVal != FETCH_EOF
        If sTabelaAnterior!=sNomeTabela
          If sRestricao!=''
            Set sRetorno = sRetorno || sRestricao || sCR
          Set sRestricao=''
        If sTipoRestricao = 'PRIMARY KEY'
          ! Call QV_ColunasDaRestricaoDaTabela(sNomeTabela, sNomeRestricao,
sRetorno1)
          Call SqlRetrieve( hSqlCur1, 'QV_ColunasDaRestricaoDaTabela',
':sNomeTabela, :sNomeRestricao, :sRetorno1', ':sRetorno1' )
          Call SqlExecute( hSqlCur1 )
          Set sRestricao = sRestricao || sNomeRestricao || ' = ' || sNomeTabela
|| '(' || sRetorno1 || ')' || sCR
        Else
          ! Call QV_ColunasDaRestricaoDaTabela(sNomeTabela, sNomeRestricao,
sRetorno1)
          Call SqlRetrieve( hSqlCur1, 'QV_ColunasDaRestricaoDaTabela',
':sNomeTabela, :sNomeRestricao, :sRetorno1', ':sRetorno1' )
          Call SqlExecute( hSqlCur1 )
          ! Call QV_ColunasDaRestricaoDaTabela(sTabelaRef, sRestricaoRef,
sRetorno2)
          Call SqlRetrieve( hSqlCur1, 'QV_ColunasDaRestricaoDaTabela',
':sTabelaRef, :sRestricaoRef, :sRetorno2', ':sRetorno2' )
          Call SqlExecute( hSqlCur1 )
          Set sRestricao = sRestricao || sNomeRestricao || ' = ' || sNomeTabela
|| '(' || sRetorno1 || ') --> ' || sTabelaRef || '(' || sRetorno2 || ')' || sCR
        Set sTabelaAnterior = sNomeTabela
        Call SqlFetchNext ( hSqlCur1, nRetVal )
    Set pRetorno = sRetorno || sRestricao || sCR
    Call SqlDisconnect ( hSqlCur1 )
\
```

18.13. O Procedimento QV_TabelasNaoPossuem_PK_nemFK

```
/*================================================================================
Procedimento:  QV_TabelasNaoPossuem_PK_nemFK
Autor:    Henrique Figueiredo de Souza
Data Criação:  13/03/2013
Descrição:
    Lista todas as tabelas que NÃO possuem Chave Primária e
    não possuem Chave Estrangeira, ordenados por data de criação.
================================================================================*/
```

```
STORE QV_TabelasNaoPossuem_PK_nemFK
PROCEDURE: QV_TabelasNaoPossuem_PK_nemFK
  Parameters
    Receive Long String: pRetorno
  Local variables
    Sql Handle: hSqlCur1
    String: sSql
    Boolean: bOk
    Number: nRetVal
    String: sCR
    Long String: sTabela
    String: sNomeTabela
  Actions
    Call SqlConnect ( hSqlCur1 )
    Set sTabela=''
    Set sSql = 'SELECT @CHAR(13) || @CHAR(10) FROM DUMMY into :sCR'
    Call SqlImmediate( sSql )
    Set sSql = 'SELECT NAME \
        FROM SYSTABLES \
        WHERE TYPE=''T'' AND SYSTEM=''N'' \
        AND NAME<>''DUMMY''
        AND NAME NOT IN (select DISTINCT NAME \
        from SYSTABCONSTRAINTS \
        WHERE TYPE IN (''P'',''F'')) INTO :sNomeTabela'
    Set bOk = SqlPrepareAndExecute ( hSqlCur1, sSql )
    If bOk
      Call SqlFetchNext ( hSqlCur1, nRetVal )
      While nRetVal != FETCH_EOF
        Set sTabela = sTabela || sNomeTabela || sCR
        Call SqlFetchNext ( hSqlCur1, nRetVal )
    Set pRetorno = sTabela
    Call SqlDisconnect ( hSqlCur1 )
\
```

18.14. O Procedimento QV_TabelasSomente_PK_naoFK

```
/*================================================================================
Procedimento:  QV_TabelasSomente_PK_naoFK
Autor:    Henrique Figueiredo de Souza
Data Criação:  13/03/2013
Descrição:
    Lista todas as tabelas que possuem SOMENTE Chave Primária e
    não possuem Chave Estrangeira, ordenados por data de criação.
================================================================================*/

STORE QV_TabelasSomente_PK_naoFK
PROCEDURE: QV_TabelasSomente_PK_naoFK
  Parameters
    Receive Long String: pRetorno
  Local variables
    Sql Handle: hSqlCur1
    Sql Handle: hSqlCur2
    String: sSql
    Boolean: bOk
    Number: nRetVal
    Number: nRetVal2
    String: sCR
    Long String: sRestricao
    String: sTabela
    String: sNomeTabela
    String: sNomeRestricao
    String: sTipoRestricao
```

```
    String: sRestricaoRef
    String: sTabelaRef
    String: sRetorno1
  Actions
    Call SqlConnect ( hSqlCur1 )
    Set sRestricao=''
    Set sSql = 'SELECT @CHAR(13) || @CHAR(10) FROM DUMMY into :sCR'
    Call SqlImmediate( sSql )
    Set sSql = 'SELECT NAME
        FROM QV_TabelasPossuem_PK_e_FK
        WHERE NAME NOT IN (select DISTINCT NAME
        from SYSTABCONSTRAINTS WHERE TYPE=''F'') INTO :sTabela'
    Set bOk = SqlPrepareAndExecute ( hSqlCur1, sSql )
    If bOk
      Call SqlFetchNext ( hSqlCur1, nRetVal )
      Loop externo
        If nRetVal = FETCH_EOF
          Break externo
        Set sRestricao=''
        !
        Call SqlConnect ( hSqlCur2 )
        Set sSql = 'SELECT DISTINCT TBNAME AS TABLE_NAME, \
            NAME AS CONSTRAINT_NAME, \
            ''PRIMARY KEY'' AS CONSTRAINT_TYPE, \
            NULL AS CONSTRAINT_REF, NULL AS REFTABNAME \
            FROM SYSINDEXES \
            WHERE UNIQUERULE=''U'' \
            AND TBNAME=:sTabela \
            UNION \
            SELECT DISTINCT F.NAME, F.CONSTRAINT, \
            ''FOREIGN KEY'', I.NAME, F.REFDTBNAME \
            FROM SYSFKCONSTRAINTS F, SYSINDEXES I \
            WHERE F.REFDTBNAME=I.TBNAME \
            AND F.CREATOR=I.CREATOR \
            AND I.UNIQUERULE=''U'' \
            AND F.NAME=:sTabela \
            ORDER BY 2 DESC \
            INTO :sNomeTabela, :sNomeRestricao, \
            :sTipoRestricao, :sRestricaoRef, :sTabelaRef'
        Set bOk = SqlPrepareAndExecute ( hSqlCur2, sSql )
        If bOk
          Call SqlFetchNext ( hSqlCur2, nRetVal2 )
          Loop interno
            If nRetVal2 = FETCH_EOF
              Break interno
            If sTipoRestricao = 'PRIMARY KEY'
              ! Call QV_ColunasDaRestricaoDaTabela(sNomeTabela, sNomeRestricao,
sRetorno1)
              Call SqlRetrieve( hSqlCur2, 'QV_ColunasDaRestricaoDaTabela',
':sNomeTabela, :sNomeRestricao, :sRetorno1', ':sRetorno1' )
              Call SqlExecute( hSqlCur2 )
              Set sRestricao = sRestricao || sNomeRestricao || ' = ' ||
sNomeTabela || '(' || sRetorno1 || ')' || sCR
            Call SqlFetchNext ( hSqlCur2, nRetVal2 )
        Set pRetorno = pRetorno || sRestricao
        Call SqlDisconnect ( hSqlCur2 )
        !
        Call SqlFetchNext ( hSqlCur1, nRetVal )
    Call SqlDisconnect ( hSqlCur1 )
\
```

18.15. O Procedimento QV_TabelasPossuem_FK

```
/*================================================================================
Procedimento:  QV_TabelasPossuem_FK
Autor:    Henrique Figueiredo de Souza
Data Criação:  14/03/2013
Descrição:
    Lista todas as tabelas que possuem Chave Estrangeira e podem
    ou não possuir Chave Primária, ordenados por data de criação.
================================================================================*/

STORE QV_TabelasPossuem_FK
PROCEDURE: QV_TabelasPossuem_FK
  Parameters
    Receive Long String: pRetorno
  Local variables
    Sql Handle: hSqlCur1
    Sql Handle: hSqlCur2
    String: sSql
    Boolean: bOk
    Number: nRetVal
    Number: nRetVal2
    String: sCR
    Long String: sRestricao
    String: sTabela
    String: sNomeTabela
    String: sNomeRestricao
    String: sTipoRestricao
    String: sRestricaoRef
    String: sTabelaRef
    String: sRetorno1
    String: sRetorno2
  Actions
    Call SqlConnect ( hSqlCur1 )
    Set sRestricao=''
    Set sSql = 'SELECT @CHAR(13) || @CHAR(10) FROM DUMMY into :sCR'
    Call SqlImmediate( sSql )
    Set sSql = 'SELECT NAME
        FROM SYSTABLES
        WHERE TYPE=''T'' AND SYSTEM=''N''
        AND NAME IN (select DISTINCT NAME
        from SYSTABCONSTRAINTS
        WHERE TYPE=''F'') INTO :sTabela'
    Set bOk = SqlPrepareAndExecute ( hSqlCur1, sSql )
    If bOk
      Call SqlFetchNext ( hSqlCur1, nRetVal )
      Loop externo
        If nRetVal = FETCH_EOF
          Break externo
        Set sRestricao=''
        !
        Call SqlConnect ( hSqlCur2 )
        Set sSql = 'SELECT DISTINCT TBNAME AS TABLE_NAME, \
            NAME AS CONSTRAINT_NAME, \
            ''PRIMARY KEY'' AS CONSTRAINT_TYPE, \
            NULL AS CONSTRAINT_REF, NULL AS REFTABNAME \
            FROM SYSINDEXES \
            WHERE UNIQUERULE=''U'' \
            AND TBNAME=:sTabela \
            UNION \
            SELECT DISTINCT F.NAME, F.CONSTRAINT, \
            ''FOREIGN KEY'', I.NAME, F.REFDTBNAME \
            FROM SYSFKCONSTRAINTS F, SYSINDEXES I \
```

```
            WHERE F.REFDTBNAME=I.TBNAME \
            AND F.CREATOR=I.CREATOR \
            AND I.UNIQUERULE=''U'' \
            AND F.NAME=:sTabela \
            ORDER BY 2 DESC \
            INTO :sNomeTabela, :sNomeRestricao, \
            :sTipoRestricao, :sRestricaoRef, :sTabelaRef'
        Set bOk = SqlPrepareAndExecute ( hSqlCur2, sSql )
        If bOk
          Call SqlFetchNext ( hSqlCur2, nRetVal2 )
          Loop interno
            If nRetVal2 = FETCH_EOF
              Break interno
            If sTipoRestricao = 'FOREIGN KEY'
              ! Call QV_ColunasDaRestricaoDaTabela(sNomeTabela, sNomeRestricao,
sRetorno1)
              Call SqlRetrieve( hSqlCur1, 'QV_ColunasDaRestricaoDaTabela',
':sNomeTabela, :sNomeRestricao, :sRetorno1', ':sRetorno1' )
              Call SqlExecute( hSqlCur1 )
              ! Call QV_ColunasDaRestricaoDaTabela(sTabelaRef, sRestricaoRef,
sRetorno2)
              Call SqlRetrieve( hSqlCur1, 'QV_ColunasDaRestricaoDaTabela',
':sTabelaRef, :sRestricaoRef, :sRetorno2', ':sRetorno2' )
              Call SqlExecute( hSqlCur1 )
              Set sRestricao = sRestricao || sNomeRestricao || ' = ' ||
sNomeTabela || '(' || sRetorno1 || ') --> ' || sTabelaRef || '(' || sRetorno2 ||
')' || sCR
            Call SqlFetchNext ( hSqlCur2, nRetVal2 )
        Set pRetorno = pRetorno || sRestricao
        Call SqlDisconnect ( hSqlCur2 )
        !
        Call SqlFetchNext ( hSqlCur1, nRetVal )
    Call SqlDisconnect ( hSqlCur1 )
\
```

18.16. O Procedimento QV_TabelasSomente_FK_naoPK

```
/*================================================================================
Procedimento:  QV_TabelasSomente_FK_naoPK
Autor:    Henrique Figueiredo de Souza
Data Criação:  14/03/2013
Descrição:
    Lista todas as tabelas que possuem SOMENTE Chave Estrangeira
    e não possuem Chave Primária, ordenados por data de criação.
================================================================================*/

STORE QV_TabelasSomente_FK_naoPK
PROCEDURE: QV_TabelasSomente_FK_naoPK
  Parameters
    Receive Long String: pRetorno
  Local variables
    Sql Handle: hSqlCur1
    Sql Handle: hSqlCur2
    String: sSql
    Boolean: bOk
    Number: nRetVal
    Number: nRetVal2
    String: sCR
    Long String: sRestricao
    String: sTabela
    String: sNomeTabela
    String: sNomeRestricao
    String: sTipoRestricao
    String: sRestricaoRef
```

```
    String: sTabelaRef
    String: sRetorno1
    String: sRetorno2
  Actions
    Call SqlConnect ( hSqlCur1 )
    Set sRestricao=''
    Set sSql = 'SELECT @CHAR(13) || @CHAR(10) FROM DUMMY into :sCR'
    Call SqlImmediate( sSql )
    Set sSql = 'SELECT NAME
        FROM QV_TabelasNaoPossuem_PK
        WHERE NAME IN (select DISTINCT NAME
        from SYSTABCONSTRAINTS
        WHERE TYPE=''F'') INTO :sTabela'
    Set bOk = SqlPrepareAndExecute ( hSqlCur1, sSql )
    If bOk
      Call SqlFetchNext ( hSqlCur1, nRetVal )
      Loop externo
        If nRetVal = FETCH_EOF
          Break externo
        Set sRestricao=''
        !
        Call SqlConnect ( hSqlCur2 )
        Set sSql = 'SELECT DISTINCT TBNAME AS TABLE_NAME, \
            NAME AS CONSTRAINT_NAME, \
            ''PRIMARY KEY'' AS CONSTRAINT_TYPE, \
            NULL AS CONSTRAINT_REF, NULL AS REFTABNAME \
            FROM SYSINDEXES \
            WHERE UNIQUERULE=''U'' \
            AND TBNAME=:sTabela \
            UNION \
            SELECT DISTINCT F.NAME, F.CONSTRAINT, \
            ''FOREIGN KEY'', I.NAME, F.REFDTBNAME \
            FROM SYSFKCONSTRAINTS F, SYSINDEXES I \
            WHERE F.REFDTBNAME=I.TBNAME \
            AND F.CREATOR=I.CREATOR \
            AND I.UNIQUERULE=''U'' \
            AND F.NAME=:sTabela \
            ORDER BY 2 DESC \
            INTO :sNomeTabela, :sNomeRestricao, \
            :sTipoRestricao, :sRestricaoRef, :sTabelaRef'
        Set bOk = SqlPrepareAndExecute ( hSqlCur2, sSql )
        If bOk
          Call SqlFetchNext ( hSqlCur2, nRetVal2 )
          Loop interno
            If nRetVal2 = FETCH_EOF
              Break interno
            If sTipoRestricao = 'FOREIGN KEY'
              ! Call QV_ColunasDaRestricaoDaTabela(sNomeTabela, sNomeRestricao,
sRetorno1)
              Call SqlRetrieve( hSqlCur1, 'QV_ColunasDaRestricaoDaTabela',
':sNomeTabela, :sNomeRestricao, :sRetorno1', ':sRetorno1' )
              Call SqlExecute( hSqlCur1 )
              ! Call QV_ColunasDaRestricaoDaTabela(sTabelaRef, sRestricaoRef,
sRetorno2)
              Call SqlRetrieve( hSqlCur1, 'QV_ColunasDaRestricaoDaTabela',
':sTabelaRef, :sRestricaoRef, :sRetorno2', ':sRetorno2' )
              Call SqlExecute( hSqlCur1 )
              Set sRestricao = sRestricao || sNomeRestricao || ' = ' ||
sNomeTabela || '(' || sRetorno1 || ') --> ' || sTabelaRef || '(' || sRetorno2 ||
')' || sCR
            Call SqlFetchNext ( hSqlCur2, nRetVal2 )
        Set pRetorno = pRetorno || sRestricao
        Call SqlDisconnect ( hSqlCur2 )
        !
```

```
        Call SqlFetchNext ( hSqlCur1, nRetVal )
    Call SqlDisconnect ( hSqlCur1 )
\
```

18.17. O Procedimento QV_DescomentarTabelasColunas

```
/*================================================================================
Procedimento:  QV_DescomentarTabelasColunas
Autor:    Henrique Figueiredo de Souza
Data Criação:  14/03/2013
Descrição:
    Descomentar todas a tabelas e colunas do banco de dados
================================================================================*/

STORE QV_DescomentarTabelasColunas
PROCEDURE: QV_DescomentarTabelasColunas
  Parameters
    Receive Long String: pRetorno
  Local variables
    Sql Handle: hSqlCur1
    String: sSql
    Boolean: bOk
    Number: nRetVal
    Long String: sRetorno
    String: sTabelaAnterior
    String: sNomeTabela
    String: sNomeColuna
    String: sColuna
    String: sCR
    Number: nConta
    String: sAPOSTROFO
  Actions
    Call SqlConnect ( hSqlCur1 )
    Set sTabelaAnterior=''
    Set sRetorno=''
    Set sSql = 'SELECT @CHAR(13) || @CHAR(10) FROM DUMMY into :sCR'
    Call SqlImmediate( sSql )
    Set sSql = 'SELECT @CHAR(39) || @CHAR(10) FROM DUMMY into :sAPOSTROFO'
    Call SqlImmediate( sSql )
    Set sSql = 'SELECT T.NAME AS TABLE_NAME, \
        C.NAME AS COLUMN_NAME \
        FROM SYSTABLES T, SYSCOLUMNS C \
        WHERE T.NAME=C.TBNAME \
        AND T.TYPE=''T'' AND T.SYSTEM=''N'' \
        AND T.NAME<>''DUMMY'' \
        ORDER BY T.NAME, C.COLNO INTO :sNomeTabela, :sNomeColuna'
    Set bOk = SqlPrepareAndExecute ( hSqlCur1, sSql )
    If bOk
      Call SqlFetchNext ( hSqlCur1, nRetVal )
      While nRetVal != FETCH_EOF
        If sTabelaAnterior!=sNomeTabela
          If sColuna!=''
            Set sRetorno = sRetorno || sColuna || sCR
          Set sColuna=''
          Set nConta=0
        If nConta=0
          Set sColuna = 'comment on table ' || sNomeTabela || ' is ' ||
sAPOSTROFO || sAPOSTROFO || ';' || sCR || sColuna
          Set nConta = 1
        Set sColuna = sColuna || 'comment on column ' || sNomeTabela || '.' ||
sNomeColuna || ' is ' || sAPOSTROFO || sAPOSTROFO || ';' || sCR
        Set sTabelaAnterior = sNomeTabela
        Call SqlFetchNext ( hSqlCur1, nRetVal )
    Set pRetorno = sRetorno || sColuna || sCR
```

```
    Call SqlDisconnect ( hSqlCur1 )
\
```