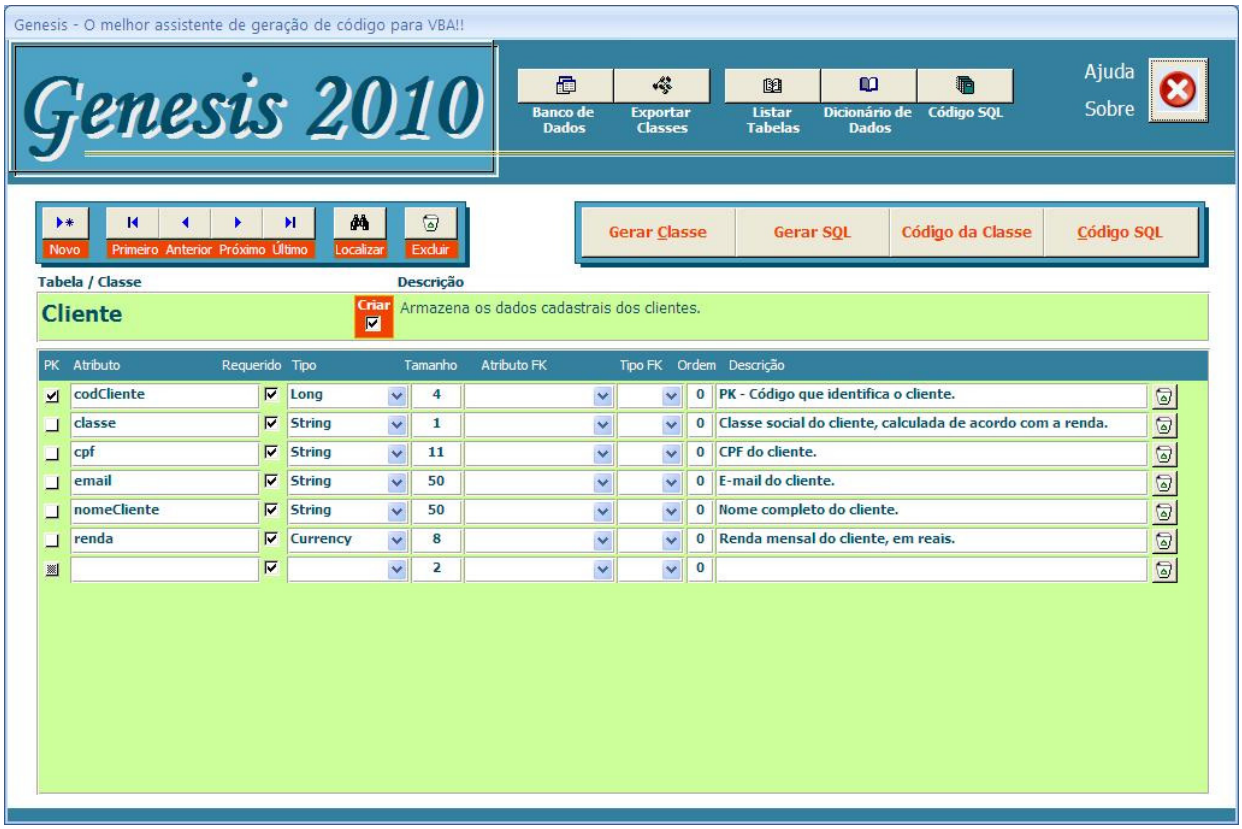
GENESIS 2010

A sua ferramenta de desenvolvimento para geração de códigos VBA e SQL

Autor: Plinio Mabesi
Contato: pliniomabesi@gmail.com



Frequentemente nós, profissionais e amadores da área de informática, nos deparamos com algumas tarefas extremamente chatas existentes no desenvolvimento de sistemas, algumas delas relacionadas aos trabalhos repetitivos e cansativos, como, por exemplo, a documentação de atributos e a programação de métodos básicos de classes, ou seja, aqueles que atribuem ou retornam valores, buscam, salvam ou excluem um objeto, entre outros.

Segundo criadores de sistemas análogos, o trabalho de programação pode ser reduzido, em média, até 60%, ao se utilizar os geradores de código. Isto permite que o profissional concentre-se apenas em implementar os métodos inerentes às regras do negócio, diminuindo consideravelmente o tempo necessário para a conclusão do projeto.

Algumas das ferramentas de geração de código existentes mantém o seu foco na criação de classes a partir de um Banco de Dados pronto, conectando-se ao mesmo e buscando as informações sobre a estrutura das tabelas e relacionamentos. Outras geram as classes a partir dos modelos de projeto. Existem também algumas especializadas em modelar o próprio esquema do Banco de Dados, fornecendo os script´s SQL para a criação do mesmo. Umas mais simples, outras nem tanto, a verdade é que nunca encontramos aquela que atende às nossas necessidades completamente.

O Genesis, que é uma opção a mais, é uma ferramenta de desenvolvimento que busca a união entre os esforços de documentação, criação do Banco de Dados e programação, auxiliando na construção do software, a partir dos dados inseridos pelo usuário uma única vez, nestes três aspectos básicos:

> **Geração do Banco de Dados**, com todas as tabelas e seus respectivos relacionamentos;
> **Geração do código das classes**, uma para cada tabela, contendo os atributos e os métodos básicos, além da classe de conexão com o BD e o módulo de importação de classes;

> **Geração dos relatórios de estrutura das tabelas, dicionário de dados e scripts de criação das tabelas**, podendo servir para análise do esquema ou fazer parte da documentação do sistema.

Para alcançarmos nosso objetivo de programar orientado a objetos, agora utilizando ferramentas que aumentem nossa produtividade, nesta etapa vamos conhecer um pouco mais do Genesis, e descobrir como ele poderá nos ajudar em nossas atividades.

Utilizando o Genesis

A ferramenta foi desenvolvida no padrão do Access 2000, para que possa ser utilizado pelo maior número de usuários. Além das bibliotecas padrão do Access, ele utiliza as seguintes referências, na ordem em que aparecem:

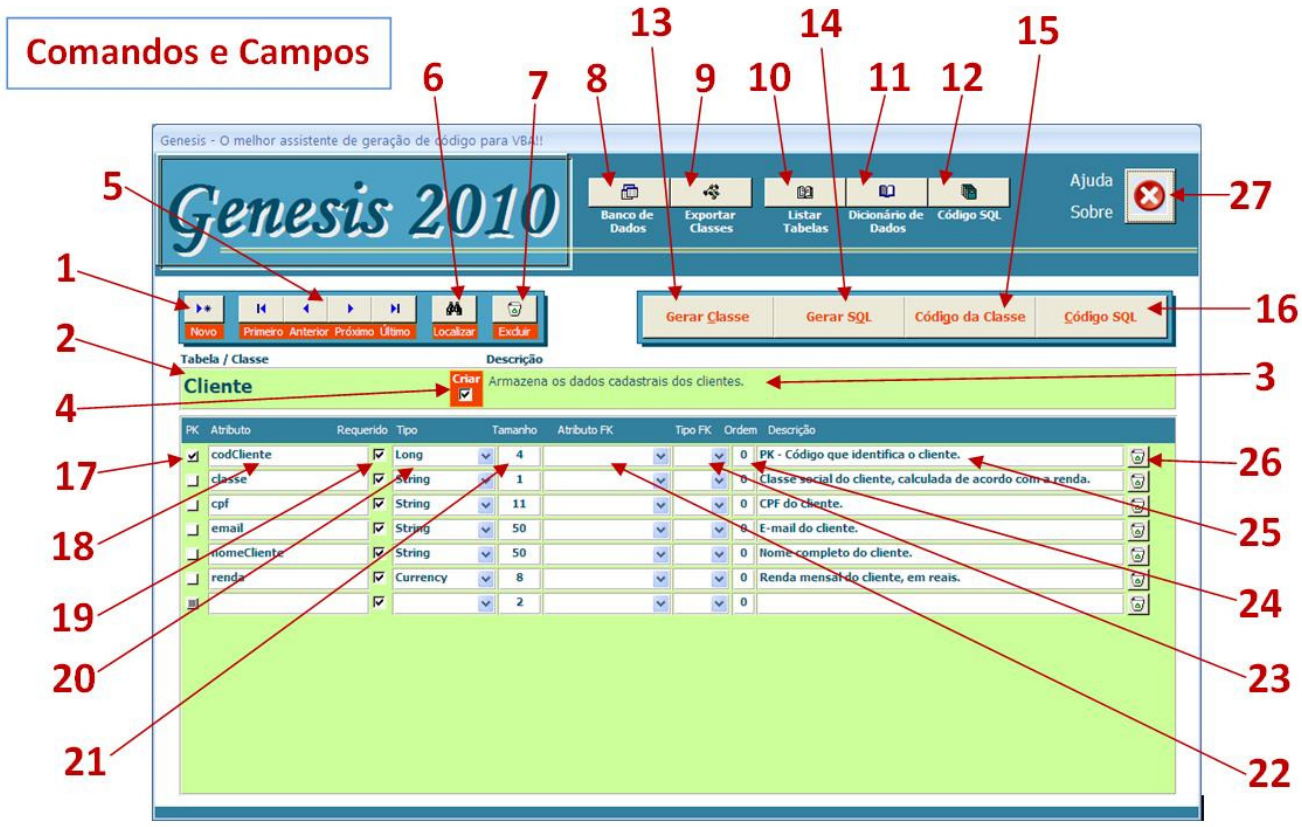
OLE Automation
Microsoft ActiveX Data Objects 2.1 Library
Microsoft DAO 3.6 Object Library

Caso sua versão possua referências diferentes, desmarque as existentes, que deverão estar indicadas como " AUSENTES ", e procure uma equivalente, a mais próxima possível. Normalmente elas têm o mesmo nome, alterando apenas o número da versão. Quando encontrá-la marque, compile e teste o programa até que encontre todas as bibliotecas necessárias.

Visando padronizar o estilo das classes e do banco de dados que serão criados, siga os padrões propostos para nomear tabelas e atributos, além de inserir descrições conforme será apresentado nos próximos tópicos. Caso vc utilize outro padrão a funcionalidade não será afetada, mas os atributos, as tabelas e as classes geradas pelo Genesis tem seus nomes modificados ou são acrescentadas iniciais que podem prejudicar a estética do sistema caso as regras não seja seguidas.

Como todo bom software livre o código do Genesis é aberto, e você pode realizar modificações a seu critério para personalizar a padronização dos nomes de tabelas, atributos e classes.

Campos e Comandos do Sistema
A figura abaixo identifica cada um dos botões de comando e campos da tela principal do Genesis. Utilize-a como referência para os tópicos seguintes, que esclarecerão com mais detalhes o funcionamento do sistema.



Trabalhando com Tabelas

Padrão proposto para nomes de tabelas:

> Primeira letra em maiúscula;
> Letras restantes em minúscula;
> Nomes compostos por mais de uma palavra separados por maiúscula;
> Não utilize acentos ou cedilha.

Ex: Departamento, Produto, Cliente, TipoCliente e CategoriaProduto

Quando o Genesis cria uma classe para a tabela é acrescentado o prefixo **cls** no início do nome da mesma, resultando em uma classe com a inicial minúscula e as separações em maiúscula. Os exemplos acima ficariam assim:

Ex: clsDepartamento, clsProduto, clsCliente, clsTipoCliente e clsCategoriaProduto

Caso você colocasse um nome, por exemplo, em minúscula, não haveria a separação das palavras. Nomes com underscore também prejudicariam a estética, resultando em nomes de classe da seguinte maneira:

Ex: clsdepartamento, clsPRODUTO, clsTipo_Cliente e clsCATEGORIA_PRODUTO

Para criar uma nova tabela clique no botão **Novo** (**1**). Será criado um registro em branco. Coloque o nome da tabela que deseja criar no campo **Tabela/Classe** (**2**), e a descrição da mesma ao lado, no campo **Descrição** (**3**), para que esta seja adicionada aos relatórios de documentação (estrutura da tabela e dicionário de dados) e ao relatório SQL.

Apenas as tabelas que estiverem com o campo **Criar** (**4**) marcado serão exportadas quando o botão **Exportar Classes** (**5**) for clicado. Para excluir uma tabela clique no botão **Excluir** (**7**). Se uma tabela for excluída todos os seus campos também serão automaticamente eliminados.

Para navegar entre tabelas e pesquisar utilize os **botões de navegação** (**5**) e **pesquisa** (**6**).

Depois de criada a tabela inclua os atributos.

Trabalhando com Campos da Tabela

Padrão proposto para os campos das tabelas:

> Primeira letra em minúscula;
> Letras restantes em minúscula;
> Nomes compostos por mais de uma palavra separados por maiúscula;
> Campos de chave primária ou estrangeira em que será utilizado um código inclua a inicial **cod** ou **id** no nome do campo;
> Não utilize acentos ou cedilha.

Ex: codDepartamento, descricao, valor, codCliente, idCliente, dataNascimento, mesEntradaProduto e categoriaProduto

Quando o Genesis cria um atributo de classe para um campo de tabela é acrescentado um prefixo que indica o tipo de dado, conforme a descrição a seguir:

> **String** (texto) => **str**
> **Date/Time** (data/hora) => **dtm**
> **Integer** (inteiro) => **int**
> **Long** (inteiro longo) => **lng**
> **Single** (decimal simples) => **sgl**
> **Double** (decimal duplo) => **dbl**
> **Currency** (valor monetário) => **cur**
> **Object** (objeto) => **obj**
> **Recordset** (conjunto de registros) => **rst**
> **Boolean** (lógico) => **boo**

Além disso a primeira letra do nome do atributo é convertida para maiúscula para manter o padrão. Os exemplos acima ficariam assim:

Ex: intCodDepartamento, strDescricao, curValor, intCodCliente, lngIdCliente, dtmDataNascimento, intMesEntradaProduto e strCategoriaProduto

Os próximos tópicos descrevem como configurar os campos da tabela a serem criados.

Chave Primária

Marque o campo **PK** (**17**) para indicar que o atributo é ou faz parte da chave primária da tabela. Ao indicar um campo como chave primária ele aparecerá na caixa de combinação do campo **Atributo FK** (**22**) de outras tabelas, para que possa ser usado como chave estrangeira.

Obs: Atributos do tipo Memo ou Object não podem ser chave primária.

Nome do Atributo

Informe no campo **Atributo** (**18**) neste campo o nome do atributo, de acordo com o padrão apresentado para nomes de campos de tabela.

Campo Obrigatório

Marque este campo para indicar que o atributo é obrigatório e não pode conter um valor nulo.

Tipo de Dado

Escolha o tipo de dado para o atributo. Os tipos possíveis são:^

> **String** => Texto com até 255 caracteres;
> **Memo** => Texto com tamanho indeterminado;
> **Date**/**Time** => Data e/ou hora, tamanho 8 bytes;
> **Integer** => Número inteiro com 2 bytes de tamanho (-32.768 a 32.767);
> **Long** => Número inteiro longo com 4 bytes de tamanho (-2.147.483.648 a 2.147.483.647);
> **Single** => Decimal com ponto flutuante com 4 bytes de tamanho (±3,402823E38 a ±1,401298E-45);
> **Double** => Decimal com ponto flutuante com 8 bytes de tamanho (±1,79769313486232E308 a ±4,94065645841247E-324);
> **Currency** => Número com ponto fixo com 15 dígitos à esquerda da vírgula e 4 dígitos à direita da vírgula, utilizado para cálculos de precisão por envolver valores em dinheiro (-922.337.203.685.477,5808 até 922.337.203.685.477,5807);
> **Object** => Endereços com tamanho de 4 bytes, que se referem a objetos (Imagens, documentos, planilhas, etc...);
> **Boolean** => Número com tamanho de 2 bytes que representa os valores lógicos True e False. A equivalência entre os valores e os números é 0 para False e qualquer outro número para True. Um valor True convertido para número retorna 1.

Tamanho do Campo

Informe no campo **Tamanho** (**21**) a quantidade máxima de caracteres para campos do tipo **String**. O tamanho mínimo permitido é 1 e o máximo é 255 caracteres. Caso seja colocado um valor fora do intervalo o Genesis informará o erro e colocará o valor padrão. Os outros tipos de campo não podem ter tamanho de campo diferente do padrão.

Estes são os tamanhos padrão para cada tipo de campo:

> **String** => 30;
> **Memo** => 0 (Não há limite definido!);
> **Currency**, **Double** e **Date**/**Time** => 8;
> **Integer** e **Boolean** => 2;
> **Object**, **Long** e **Single** => 4.

Chave Estrangeira (Foreign Key)

Indique no campo **Atributo FK** (**22**) o atributo (chave primária da tabela de destino) que servirá de referência para a chave estrangeira na tabela à qual estará vinculado.

Os possíveis atributos são listados e podem ser acessados inclusive a partir da mesma tabela. Isto ocorre quando, por exemplo, um atributo referencia outro atributo da mesma tabela, criando um auto-relacionamento. Esta situação poderia ocorrer em casos como o de uma empresa onde um funcionário é chefe de outro, ou um departamento é subordinado a outro, ou quando um produto é componente de outro, e a melhor forma de mapear este relacionamento seja uma tabela única.

Você só pode criar uma chave estrangeira que referencie um campo de chave primária de outra tabela, ou da mesma tabela, se os dois campos forem do mesmo tipo. Como exemplo um campo do tipo Integer não pode referenciar um String, e assim sucessivamente.

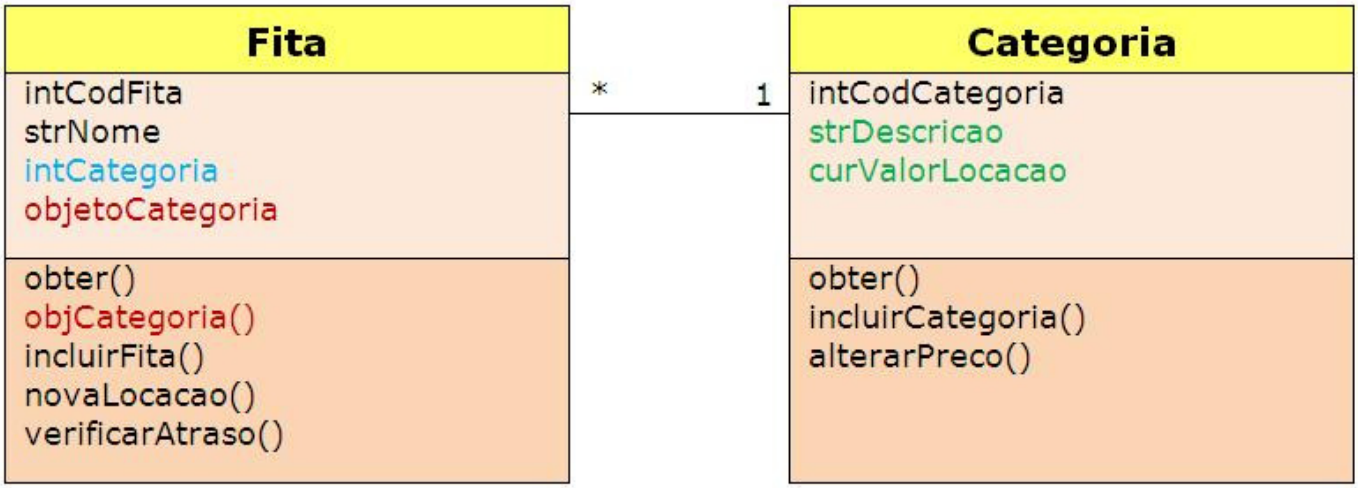
Ao vincular campos de chave estrangeira o Genesis criará os relacionamentos com integridade referencial, porém não incluirá os eventos em cascata para atualização e exclusão de dados. Caso seja necessária, esta configuração deverá ser feita manualmente pelo desenvolvedor no banco de dados criado pelo Genesis.

Tipo de Chave Estrangeira

Ao incluir uma chave estrangeira na tabela com certeza você fará uso desta ligação também nas classes. Logo devemos informar o tipo de chave estrangeira no campo **Tipo FK** (**23**). Para compreender a utilidade deste recurso, imagine os dois casos a seguir:

No primeiro temos uma relação entre **Fita** e **Categoria**, duas classes de uma locadora de fitas de vídeo. Nesta locadora o valor de locação não é colocado individualmente nas fitas. Elas são classificadas em categorias, como Lançamento, Catálogo, Infantil e Promoção, para que seja facilitado o trabalho de alteração de preços. Ao se modificar o valor de locação de uma categoria todas as fitas estarão atualizadas automaticamente.

1º Caso: Descrição de Tipo (DT)



Veja bem, no código da classe **Fita** deverá existir um método **novaLocacao()** ou algo parecido, que será responsável pela efetivação de uma locação de fita. Para calcular o valor da locação o método teria que instanciar a classe **Categoria**, verificar a que categoria a fita pertence, através do atributo **intCategoria**, e solicitar a ela o valor da locação.

Até aí tudo bem, mas então qual a utilidade do recurso **Descrição de Tipo**? Pois bem, este recurso cria automaticamente um atributo do tipo de objeto da classe que precisa ser instanciada (Categoria), e toda vez que o atributo de chave estrangeira **intCategoria** é atualizado, a referência à classe **Categoria** também é atualizada através do atributo objeto criado.

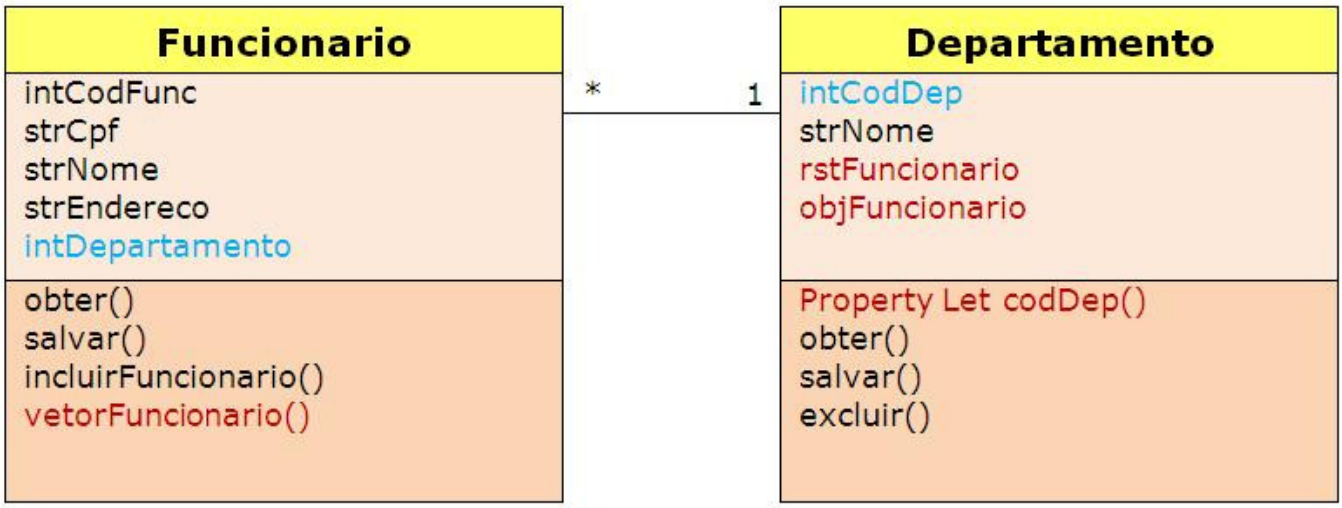
No exemplo dado os atributos da classe **Categoria** podem ser acessados através da classe **Fita**, da seguinte maneira:

```
'Declaramos algumas variáveis conforme a necessidade
Dim curPrecoLocacao As Currency
Dim strDescricao As String

'Declaramos e instanciamos o objeto Fita
Dim objFita As New clsFita

'Obtemos o objeto fita cujo código é 1012
objFita.obter(1012)

'Acessamos os atributos da classe Categoria
strDescricao = objFita.objCategoria.descricao
curPrecoLocacao = objFita.objCategoria.valorLocacao

'De dentro da classe Fita o acesso seria mais simples, da seguinte maneira
strDescricao = objCategoria.descricao
curPrecoLocacao = objCategoria.valorLocacao
```

Vamos analisar agora o segundo caso, o **Vetor de Registro (VR)**. A descrição de tipo serve para buscarmos um valor único em outra classe, mas imagine, como no exemplo abaixo, um **Departamento** que queira obter uma relação com todos os **Funcionários** que trabalham no mesmo. Uma solução seria criar um **Recordset**, executar uma consulta e navegar através dele para coletarmos os dados desejados.

Visando facilitar este trabalho, ao se definir uma chave estrangeira como **VR**, o Genesis inclui automaticamente um método **vetorNomeDaClasse()** na classe atual, cujo valor de retorno é exatamente um Recordset contendo todos os objetos que pertencem ao objeto que possui o método, neste caso todos os funcionários que pertencem ao departamento.

2º Caso: Vetor de Registro (VR)



Veja no modelo de classes que, após definir **intDepartamento** como chave estrangeira do tipo **VR**, foi criado o método **vetorFuncionario()** na classe **Funcionario**, o qual utiliza o valor do atributo para saber a que **Departamento** pertence o funcionário. A consulta é feita via SQL, diretamente no banco de dados, mas a atribuição dos valores foi originada anteriormente pelos atributos do objeto.

Além disso foram incluídos dois atributos na classe estrangeira, aquela que necessitava da relação de registros da outra classe. Foram criados um objeto da classe que contém o **VR** e um **Recordset** que recebe os registros retornados da consulta. A operação é executada toda vez que os atributos chave da classe estrangeira são atualizados. Neste caso quando **codDep** é alterado pelo procedimento **Property Let codDep(),** os atributos **objFuncionario** e **rstFuncionario** também são atualizados pelo procedimento.

A partir daí podemos navegar pelo **Recordset rstFuncionario** normalmente através da classe **Departamento**, ou acessar suas propriedades, da seguinte maneira:

```
'Navegando pelo Recordset, buscando o próximo registro
objDepartamento.rstFuncionario.moveNext

'Verificando a quantidade de registros de funcionários
objDepartamento.rstFuncionario.recordCount

'Acessando um atributo do funcionário
objDepartamento.rstFuncionario.Fields("nome")
```

*Obs: Perceba que ao acessarmos um atributo pelo recordset devemos utilizar o nome original do mesmo, pois o prefixo **str** adicionado existe somente dentro da classe.*

Devemos nos lembrar que em nenhum momento o programador necessita instanciar as classes ou os Recordsets para implementar estes recursos, pois o Genesis deixa todo o código pronto. Ao programador cabe apenas instanciar a classe com a qual vai trabalhar e fazer uso das facilidades apresentadas.

Importante:

> O recurso Descrição de Tipo só é implementado automaticamente quando a classe estrangeira possui chave primária única. Quando a classe estrangeira possui chave primária composta por mais de um atributo o Tipo FK será ignorado.
> O recurso Vetor de Registro só é implementado automaticamente quando as duas classes possuem chave primária única. Nas classes em que a chave primária é composta de mais de um atributo o Tipo FK será ignorado.

Ordem da Chave Estrangeira

Para definir a ordem das chaves estrangeiras no banco de dados a ser criado devemos informá-la no campo **Ordem (24).**

Este recurso tem a única, mas não menos importante, utilidade de separar chaves estrangeiras para uma mesma chave primária da tabela estrangeira. Isto ocorre quando temos dois ou mais atributos semelhantes em uma tabela fazendo referência ao mesmo atributo FK de outra tabela. Raramente isto acontece, mas no caso de ocorrer lembre-se de utilizar a **Ordem**, ou ocorrerá um erro tanto na criação da tabela quanto da classe.

Para ilustrar melhor a necessidade veja um exemplo prático. Em um hospital os médicos trabalham por escala de plantão. Digamos que o sistema do hospital mantém o registro das equipes que concorrem ao plantão diariamente. Suponha que o BD esteja organizado mais ou menos assim:

Tabela **Medico**

Código	Nome	Especialidade	CRM
1	Paulo Almeida	Clínico	123456
2	Adriana Mendes	Obstetra	456123
3	Otávio Santos	Pediatra	228789
4	Marcela Saraiva	Clínico	456984
5	Luciano Couto	Neurologista	228156

Tabela **Plantao**

Data	Turno	MedicoEscalado	MedicoSubstituto
20/01/2005	D	2	2
22/01/2005	N	1	3
27/01/2005	D	4	4
12/02/2005	N	5	5
14/02/2005	D	1	1
04/03/2005	N	2	4

Podemos notar claramente que o sistema guarda os registros do médico que estava escalado para o plantão e o médico que efetivamente tirou o plantão. Quando os dois códigos são iguais significa que não houve substituição. Mas o importante é perceber que tanto o atributo **MedicoEscalado** quanto **MedicoSubstituto** da tabela **Plantao** referenciam a mesma chave primária da tabela **Medico**, ou seja, o atributo **Codigo**.

Para que o Genesis não confunda um caso como este com um caso de **chave composta** por mais de um atributo, você deve diferenciar os atributos FK definindo **Ordens** diferentes para os mesmos. Poderíamos definir o campo Ordem para o atributo MedicoEscalado com o valor 1 e para MedicoSubstituto com o valor 2, por exemplo. Nos casos de chave composta deve ser mantida a mesma Ordem para todos os atributos, para o caso de apenas um conjunto de chave estrangeira, ou uma ordem para cada conjunto, no caso de haver mais de um.

Em resumo, a Ordem representa uma seqüência, ou conjunto, de chaves estrangeiras para uma mesma chave ou conjunto de chaves primárias.

Descrição do Atributo

Informe no campo Descrição **(25)** os detalhes que esclareçam a função do atributo, para que ela seja adicionada aos relatórios de documentação (estrutura da tabela e dicionário de dados) e aos comentários da classe. A descrição pode ter até 150 caracteres.

Lembre-se de utilizar texto que deixe claro a utilidade do atributo, e que esteja livre de erros de ortografia ou gramática, zelando, assim, pela estética da documentação e do sistema que será gerado.

Para campos de chave primária, a sugestão é colocar a sigla "**PK –** " (Primary Key) no início da descrição, para identificá-loS como tal. Já para campos de chave estrangeira, a sugestão é colocar a sigla "**FK –** " (Foreign Key).

Excluindo um Atributo

Para excluir um atributo basta clicar no botão que contem uma lixeira **(26)** e confirmar quando o sistema solicitar.

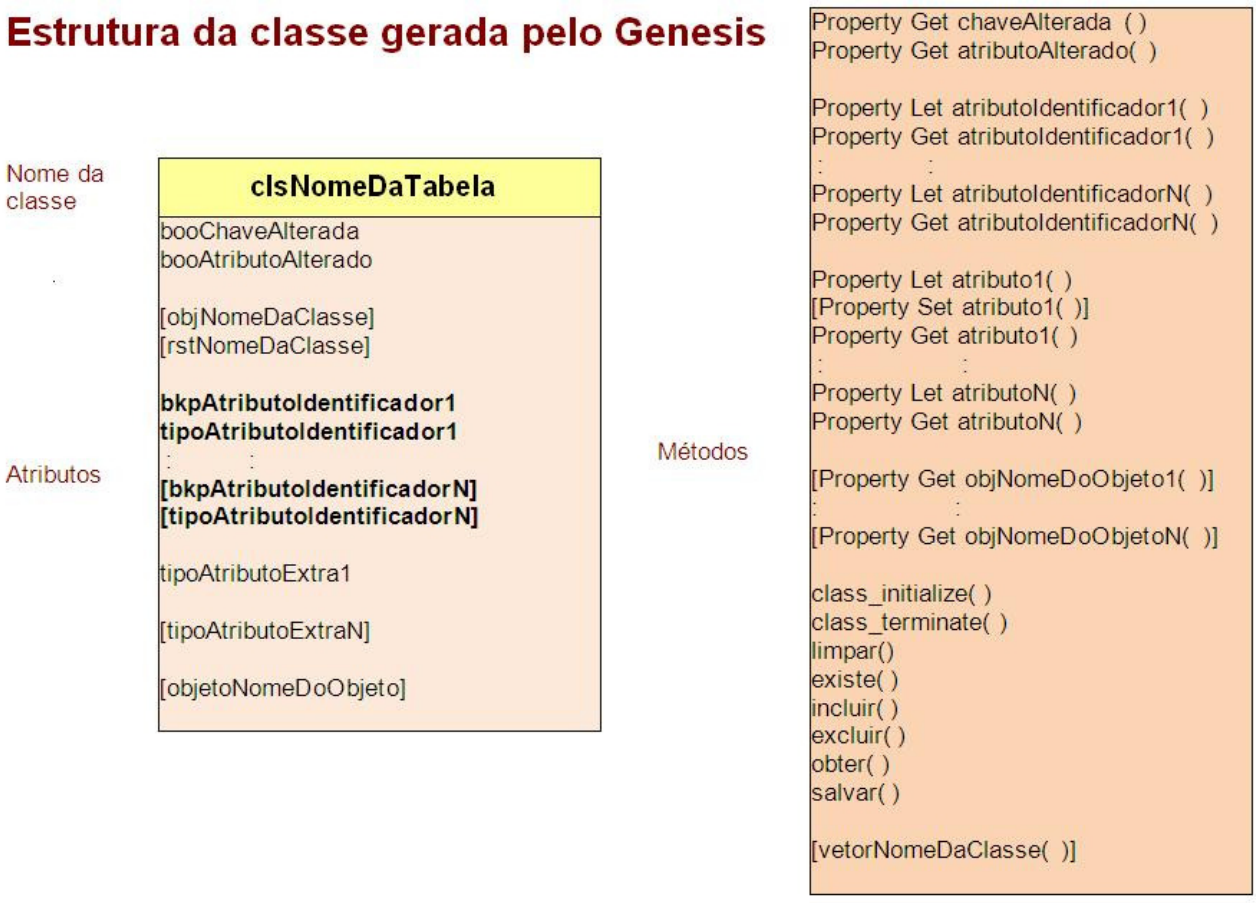
Gerando uma Classe

Para gerar o código da classe atual clique no botão **Gerar Classe (13)**. O código será criado e apresentado em um formulário independente. O campo **Criar (4)** não precisa estar marcado para a utilização deste comando.

Um conhecimento prévio dos conceitos de Orientação a Objetos auxilia na compreensão dos códigos desta parte. Em caso de dúvida consulte o restante do material ou uma das referências apresentadas anteriormente para se aprofundar no assunto.

Primeiramente vamos conhecer a estrutura completa de uma classe gerada pela ferramenta Genesis.

Estrutura da classe gerada pelo Genesis



Nome da Classe => Conforme descrito anteriormente, o nome da classe é criado adicionando-se o prefixo cls ao nome da tabela.

clsNomeDaTabela

Atributos => É criado um atributo para cada campo da tabela. Para cada campo de chave primária é criado também um atributo de backup, utilizado pelo método salvar(), pois no caso de ocorrer uma alteração de um atributo chave não seria mais possível atualizar o BD pelo novo valor. Neste caso o método salvar() não faria efeito, então ele utiliza o valor do backup para executar a consulta atualização, em seguida altera todos os atributos para os novos valores, caso a operação ocorra com sucesso. Além destes atributos particulares de cada classe, são criados dois atributos genéricos, booChaveAlterada e booAtributoAlterado, ambos lógicos, utilizados para indicar alterações nos valores de atributos chaves e atributos comuns, respectivamente. Estes atributos não possuem função específica, e podem ser utilizados pelo desenvolvedor conforme a necessidade, quando este quiser saber se uma chave ou atributo tiveram seus valores modificados. Os atributos do tipo objeto e recordset automáticos são criados caso exista algum atributo do tipo Descrição de Tipo (DT) ou Vetor de Registro (VR).

booChaveAlterada
booAtributoAlterado

bkpAtributoIdentificador1
tipoAtributoIdentificador1
: :
[bkpAtributoIdentificadorN]
[tipoAtributoIdentificadorN]

tipoAtributoExtra1

[tipoAtributoExtraN]

Métodos de acesso aos Atributos => Para cada atributo originado de um campo da tabela são criados dois métodos, um para atribuir valor (Property Let, ou Property Set para objetos) e outro para recuperar valor (Property Get). Dentro destes métodos podem existir funções extras, como as que implementam o VR, dependendo do contexto.

Property Get chaveAlterada ( )
Property Get atributoAlterado( )

Property Let atributoIdentificador1( )
Property Get atributoIdentificador1( )
: :
Property Let atributoIdentificadorN( )
Property Get atributoIdentificadorN( )

Property Let atributo1( )
[Property Set atributo1( )]
Property Get atributo1( )
: :
Property Let atributoN( )
Property Get atributoN( )

[Property Get objNomeDoObjeto1( )]
: :
[Property Get objNomeDoObjetoN( )]

Métodos construtor e destrutor => Os métodos construtor e destrutor são criados sem qualquer código executável. Apenas a estrutura deles é gerada. Para utilizá-los basta inserir código dentro deles. Todo código que estiver dentro do método **class_initialize()** será executado sempre que um objeto da classe for instanciado. Da mesma forma todo código dentro de um método **class_terminate()** será executado sempre que a classe encerrar a sua existência, mas antes que isto realmente ocorra.

class_initialize( )
class_terminate( )

Métodos de acesso ao Banco de Dados => São os métodos básicos de acesso ao banco de dados, utilizados para realizar a persistência dos objetos. O método **obter()** é utilizado para a recuperação de um objeto, recebendo como parâmetro o(s) valor(es) do(s) atributo(s) chave da classe, e realizando uma consulta a partir deste(s) valor(es). O método **incluir()** faz o oposto. Depois de definidos os valores dos atributos, este método os coleta e inclui o novo objeto no banco de dados, tornando-o persistente. O método **salvar()** grava as alterações feitas aos atributos do objeto no banco de dados. Um objeto só pode ser salvo se já existia anteriormente. Objetos novos devem ser incluídos antes que se possa salvá-los. Antes de salvar um objeto o método verifica se o mesmo já não existe no banco de dados, utilizando o método **existe()**, que apenas checa a existência de um registro com o mesmo valor de chave na respectiva tabela. O método **excluir()** apaga o registro referente ao objeto no banco de dados. Todos este métodos retornam verdadeiro caso a operação ocorra com sucesso, ou falso em caso contrário.

existe( )
incluir( )
excluir( )
obter( )
salvar( )

Limpar => Método que redefine todos os atributos de volta aos seus valores iniciais, ou seja, atribui o valor Nulo para todos os atributos e False para os atributos booChaveAlterada e booAtributoAlterado. O método é executado toda vez que um objeto é excluído com sucesso, imediatamente após o término da operação. Você pode utilizá-lo a qualquer momento para "limpar um objeto".

limpar()

Vetor de Registro (VR) e Descrição de Tipo (DT) => Os atributos e o método criados para a implementação da descrição de tipo e do vetor de registro foram detalhados no item que trata sobre o campo **TipoFK**. São os objetos que referenciam outra classe para obter valores de atributos da mesma, no caso da DT, ou uma relação de registros, no caso do VR. Estes atributos e o método são gerados apenas quando o usuário define uma chave estrangeira com um dos dois tipos.

[objNomeDaClasse]
[rstNomeDaClasse]

[objetoNomeDoObjeto]

[vetorNomeDaClasse( )]

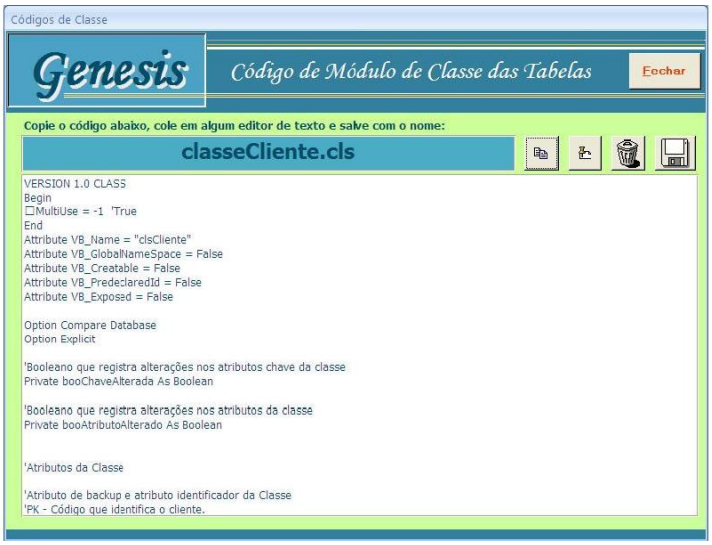
Gerando o Código SQL da Tabela

Para gerar o script SQL de criação da tabela clique no botão **Gerar SQL (14)**. O código será criado e uma mensagem de confirmação informará o sucesso da operação.

Visualizando o Código SQL da tabela e o Código da Classe

Para visualizar o script SQL de criação da tabela clique no botão **Código SQL (16)**. O código será apresentado em um formulário independente. O código pode ser copiado ou alterado, mas esta alteração não fará efeito na criação do Banco de Dados.

Para visualizar o código da classe clique no botão **Código da Classe (15)**. O código será apresentado em um formulário independente. O código pode ser copiado ou alterado, mas sua alteração não fará efeito na exportação das classes.



Visualizando o Relatório de Estrutura das Tabelas

Para visualizar o relatório de estrutura das tabelas clique no botão **Listar Tabelas (10)**. O relatório com as informações solicitadas será apresentado.

Visualizando o Dicionário de Dados

Para visualizar o dicionário de dados clique no botão **Dicionário de Dados (11)**. O relatório com as informações solicitadas será apresentado.

Visualizando o Relatório SQL

Para visualizar o relatório de código SQL das tabelas clique no botão **Código SQL (12)**. O relatório com as informações solicitadas será apresentado. Para exportá-lo para o Word utilize a função Publicar com o MS Word

Gerando o Banco de Dados

Para gerar o banco de dados completo clique no botão **Banco de Dados (8)**. A caixa de diálogo do Windows **Salvar Como** será apresentada. Escolha a pasta e o nome do novo banco de dados e clique em Salvar. Caso não ocorram erros uma mensagem será apresentada ao término do processamento informando que o BD foi gerado com sucesso.

Para colocar o projeto em andamento, exporte as classes para uma pasta e, em seguida, importe-as para o BD que acaba de ser criado. Para isso abra o editor do Visual Basic e vá ao menu **Arquivo** / **Importar Arquivo** e inclua as classes. Esta operação só permite a importação de uma classe por vez, então seja paciente. Para ajudar com esta importação, ou mesmo a atualização de classes, o Genesis inclui o módulo IRModulos, que contem as funções **importaClasses()** e **removeClasses()**, que serão detalhadas em um tópico a parte.

Na ocorrência de erros cada um deles será informado, sendo que isto não interrompe o processo de criação do BD. As mensagens tentarão informá-lo sobre o erro para que seja corrigido. Ao final será apresentada a mensagem de conclusão, na qual constará o número de erros ocorridos.

A mensagem final irá sugerir a você que verifique e organize a estrutura estética dos relacionamentos, pois eles são criados em ordem aleatória, fazendo com que as tabelas fiquem em posições que deixam as linhas dos relacionamentos cruzando-se entre si, de maneira totalmente caótica. Nada que interfira no funcionamento, mas para que você mantenha seu modelo adequadamente limpo, um pouco de ordem não fará mal.

A Classe de Conexão

A classe **aclConexaoBD** será incluída no banco de dados criado, possibilitando a utilização imediata a partir das outras classes ou qualquer outra função que necessite acesso às tabelas via SQL. As classes geradas pelo Genesis já utilizam a classe de conexão.

Importando e Removendo Módulos

A fim de evitar que os usuários tenham que importar ou remover módulos individualmente do sistema, utilizando o menu do editor do VBA, ou a combinação de teclas **Ctrl+M**, foi incluído no Genesis o módulo **IRModulos**, que possui duas funções, uma para importar e outra para remover módulos em grupo.

Os procedimentos devem ser executados manualmente pelo desenvolvedor, que deverá ajustar as variáveis que indicam o caminho da pasta onde se encontram as classes, posicionar o cursor dentro do procedimento desejado e teclar **F5** ou abrir o menu **Executar** => **Executar Sub/UserForm** no menu do editor do VBA.

Este é o código dos procedimentos do módulo:

```
Sub importaClasses()
'Como utilizar este procedimento para importar seus módulos:
'primeiro crie as classes utilizando o Genesis, exporte-as
'para uma pasta qualquer, altere o endereço da variável
'caminho abaixo, e execute o procedimento, posicionando o
cursor dentro do código e pressionando a tecla F5.

    Dim fs, pasta, arquivos, arquivo
    Dim caminho As String
    Dim nomeCompleto As String
    Dim nomeArquivo As String

    'Altere o caminho da pasta para o
    'local onde estão as suas classes
    'a serem importadas.
    caminho = "C:\MeuProjeto\Classes"

    If caminho <> "" Then

        Set fs = CreateObject("Scripting.FileSystemObject")
        Set pasta = fs.GetFolder(caminho)
        Set arquivos = pasta.Files
        For Each arquivo In arquivos
            nomeArquivo = arquivo.Name
            If Left(nomeArquivo, 3) = "cls" Then
                nomeCompleto = arquivo.Path
                Application.VBE.ActiveVBProject.VBComponents.Import nomeCompleto
            End If
        Next

    End If

End Sub

Sub removeClasses()
'Para remover os módulos execute o
'procedimento posicionando o cursor
'dentro do código e pressionando F5.

    Dim componentes, componente
    Dim nomeComponente As String

    Set componentes = Application.VBE.ActiveVBProject.VBComponents

    For Each componente In componentes
        nomeComponente = componente.Name
        If Left(nomeComponente, 3) = "cls" Then
            Application.VBE.ActiveVBProject.VBComponents.Remove componente
        End If
    Next

End Sub
```

Caso seja necessário realizar alguma alteração em qualquer dos atributos ou das tabelas, incluir ou retirar campos, ou mesmo corrigir descrições, remova as classes do projeto, faça as alterações no Genesis, exporte novamente as classes e importe-as novamente. Tudo isto pode ser feito em questão de minutos.

Tenha o cuidado de deletar as classes exportadas anteriormente, pois quando elas são sobrepostas e a nova classe for menor que a anterior, o final da primeira não será apagado, resultando em uma classe com erros no código.

Obs: Perceba que os procedimentos testam se o nome do componente é iniciado pelo prefixo cls. Logo somente módulos que se iniciam com o prefixo serão importados ou removidos. Para importar ou remover outros módulos basta ajustar o prefixo conforme a necessidade. Este modelo apenas foi ajustado para trabalhar com as classes criadas pelo Genesis, cujos nomes se iniciam por ***cls****.*

Exportando as classes

Para gerar e exportar todas as classes de uma só vez clique no botão **Exportar Classes (9)**. A caixa de diálogo do Windows **Procurar Pasta** será apresentada. Escolha a pasta para onde deseja exportar suas classes e clique em OK. Caso não ocorram erros uma mensagem será apresentada ao término do processamento informando que as classes foram exportadas com sucesso e exibindo o caminho completo da pasta escolhida.

Atenção: Somente as tabelas que estiverem com o campo **Criar (4)** marcado serão geradas e exportadas.

Encerrando o Sistema

Para encerrar o Genesis basta clicar no **botão de encerramento (27)**, localizado no canto superior direito da tela principal.

Download do Genesis

Será disponibilizado o link para download do Genesis, assim você poderá sempre obter a última versão do sistema, contendo as atualizações mais recentes.

Segue o link para download:

http://dl.getdropbox.com/u/507553/Genesis.zip

Utilizando este sistema com toda certeza seu trabalho de desenvolver projetos orientados a objetos no Access/VBA será reduzido consideravelmente. Planeje seu modelo de banco de dados, transcreva para o programa, exporte suas classes, gere seu banco de dados e importe as classes. Pronto, seu novo projeto está iniciado.

Curso de Utilização de Classes em Access/VBA

Caso tenha interesse visite o curso sobre uso de classes no Access. O curso é formado por uma série de artigos envolvendo conceitos de programação orientada a objetos, objetos no Access/VBA, modelagem de classes e de banco de dados, além de programação de diversas rotinas importantes para a maioria dos projetos de sistemas.

O curso poderá ser acessado através do seguinte endereço:

http://usandoaccess.com.br/tutoriais/tuto15.asp?id=1#inicio

Agradeço pela escolha e utilização do programa, assim como pelas críticas enviadas, que serão muito bem-vindas.

Obrigado.
Plinio Mabesi