UNIVERSIDADE FEDERAL DE PERNAMBUCO
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

EXTENSÕES DE UM CLIENTE OLAP PARA CONSULTAS GERENCIAIS A CUBOS EM UM BD MULTIDIMENSIONAL

TRABALHO DE GRADUAÇÃO - RELATÓRIO FINAL

PROFESSORA ORIENTADORA: ANA CAROLINA SALGADO (acs@cin.ufpe.br)
PROFESSOR AVALIADOR: FERNANDO DA FONSECA DE SOUZA (fdfd@cin.ufpe.br)

ALUNO: FÁBIO ÁVILA RÊGO PESSOA (farp@cin.ufpe.br)

Recife, 29 de outubro de 2002

Resumo

Este Trabalho de Graduação objetiva o desenvolvimento de uma aplicação cliente OLAP para o Microsoft Excel. O objetivo é disponibilizar vários recursos gerenciais importantes não disponíveis no PivotTable Service, o cliente OLAP que vem acoplado ao Microsoft Office.

A ferramenta será desenvolvida utilizando Visual Basic for Applications, a linguagem de programação nativa do Microsoft Excel. Será também utilizado MDX [12] para as consultas multidimensionais ao Analysis Server, Banco de Dados multidimensional utilizado neste projeto.

O projeto pretende implantar alguns recursos gerenciais importantes não disponíveis no PivotTable do Excel:

- Permitir a inserção de linhas e colunas entre os dados advindos do cubo [11] no BD multidimensional.
- Possibilitar escolha de ordenação dos dados advindos do cubo no BD multidimensional.
- Exibir os dados dos registros com maior valor em determinada medida.
- Possibilitar write-back nos dados do cubo, para possibilitar análise de cenários possíveis.
- Permitir drill-through para mostrar os dados de detalhe relacionados com determinado valor.

Agradecimentos

Este trabalho teve enormes contribuições de companheiros de trabalho, clientes e parceiros da empresa na qual sou diretor, a Ávila Sistemas Ltda. Através de ampla parceria de consultoria entre a Lanlink Informática Ltda. e a Ávila Sistemas, que possibilita contato sólido com muitos de seus grandes clientes, tive oportunidade de conhecer amplas demandas gerenciais e fortes necessidades de produtos para análise de dados e Business Intelligence. Fica aqui um sincero agradecimento a muitos que trabalham na Lanlink: Jailson Batista, Gerente de Suporte; Roberto Andrade, Diretor; Fábio Gambôa, Consultor e Gerente de Projetos; Libório Penz, Gerente de Negócios. Através da Lanlink tive também a oportunidade de ministrar palestras sobre Microsoft OLAP para grandes clientes do mercado local. Maior parceiro Microsoft do Nordeste, a Lanlink é um Centro de Treinamento Autorizado da Microsoft e um Parceiro de Suporte Autorizado da Microsoft.

Agradeço à minha orientadora, professora Ana Carolina Salgado, que auxiliou com diversos aspectos referentes à estruturação de documentos impressos e conteúdo dos mesmos. Ela teve papel fundamental para o projeto, principalmente nas etapas de levantamento inicial e escrita do relatório final.

Outra contribuição importante a este trabalho foi feita pelo Sr. Ricardo Rêgo, Diretor Financeiro da Companhia Pernambucana de Gás – Copergás, por conceder um Banco de Dados para construção da base de testes utilizada no projeto.

Fica aqui também registrado um forte agradecimento a Andrey Cavalcanti, consultor técnico na Microsoft – filial Recife. Além de companheiro de turma e bom amigo, Andrey forneceu valiosas dicas de documentos, links para informações públicas valiosas a partir do site da Microsoft e participou de importantes reuniões com clientes para estabelecer rumos em projetos usando a tecnologia Microsoft OLAP.

Gostaria também de agradecer ao pessoal interno de minha empresa, em particular a George Soares, que tem participado da implementação de projetos Microsoft OLAP através de nossa parceria com a Lanlink.

Naturalmente não poderia deixar de agradecer também a minha família por todo o suporte emocional e à minha amada noiva, Geisa Lira, que foi além de tudo companheira de trabalho na implantação de um projeto OLAP no início de 2002.

Índice

I Introdução e Contexto

Um dos segmentos mais expressivos dos aplicativos para Business Intelligence (BI) [9] é o que lida com OLAP (OnLine Analytical Processing) [8], um termo que tem sido usado para representar aplicações que possibilitam uma visão multidimensional de dados. O paradigma de consulta aos dados é diferente do modelo convencional baseado em um modelo puramente relacional e nas aplicações transacionais convencionais. Há diversos motivos para crer que a visão multidimensional de dados será amplamente popularizada em um futuro próximo:

- Principalmente no Brasil, a maioria das empresas desconhece o paradigma de consultas multidimensionais e os benefícios que ele pode trazer. Com a maior disseminação do paradigma, as empresas tenderão a implementar soluções de consulta gerenciais multidimensionais.
- Muitas empresas já possuem hoje um produto OLAP e não sabem disso. O Microsoft Excel [1][6] e o Microsoft Analysis Services[1] [2][6] são componentes de uma solução OLAP que inclui um BD multidimensional. Muitas empresas já possuem o Microsoft Excel porque ele é parte integrante do pacote integrado para escritórios mais vendido do mundo. O Analysis Services traz um BD multidimensional, parte integrante do Sistema de Banco de Dados da Microsoft, o SQL Server [3]. Sem custo adicional, o produto está disponível para uso desde que tenha sido adquirida uma licença do SQL Server. Muitos usuários do Sistema de Banco de Dados não estão cientes deste fato. Outro fato importante é que o usuário não necessariamente necessita de um SGBD multidimensional (o Analysis Services) para consultar dados no paradigma multidimensional. Apenas com o Microsoft Excel é possível obter este paradigma, permitindo acessar fontes de dados heterogêneas.
- Hoje há soluções OLAP com custo muito menor do que no passado. Os produtos estão mais poderosos e fáceis de usar [10]. O mercado é promissor e há muitos investimentos na área.
- Uma pesquisa do Grupo Gartner indica que os dados disponíveis estão crescendo exponencialmente, decisões críticas por semana têm triplicado nos últimos 5 anos e o pessoal analítico dedicado tem diminuído mais de 50% nas últimas décadas [7].

No segmento OLAP, muitos desenvolvedores optaram por desenvolver aplicativos clientes para consulta de dados multidimensionais no formato de um executável *standalone*, o que muitas vezes compromete a adoção do produto em larga escala pelo fato de o usuário necessitar instalar o produto à parte, o que gera uma demanda de suporte e treinamento adicionais. Entretanto, talvez o maior motivo da falta de adoção do produto seja o custo elevado de várias soluções existentes.

Uma grande motivação para este Trabalho de Graduação é poder gerar um suplemento do Microsoft Excel, sendo de fácil disseminação, com recursos gerenciais importantes. A intenção é que este Trabalho posteriormente dê origem a um produto com custo bastante inferior aos produtos existentes no mercado.

Este Trabalho de Graduação objetiva o desenvolvimento de uma aplicação cliente OLAP para o Microsoft Excel. O objetivo é disponibilizar vários recursos gerenciais importantes não disponíveis no PivotTable Service, o cliente OLAP que vem acoplado ao Microsoft Office.

A ferramenta será desenvolvida utilizando Visual Basic for Applications, a linguagem de programação nativa do Microsoft Excel. Será também utilizado MDX [12] para as consultas multidimensionais ao Analysis Server, Banco de Dados multidimensional utilizado neste projeto.

[1] Analysis Services é o nome do produto no Microsoft SQL Server 2000. Na versão 7.0, o produto chama-se OLAP Services.

O projeto pretende implantar no Excel alguns recursos gerenciais importantes:

- Permitir a inserção de linhas e colunas entre os dados advindos do cubo [11] no BD multidimensional.
- Possibilitar escolha de ordenação dos dados advindos do cubo no BD multidimensional.
- Exibir os dados dos registros com maior valor em determinada medida, permitindo agrupamento da soma dos valores de outros membros não presentes na lista.
- Possibilitar write-back nos dados do cubo, para possibilitar análise de cenários possíveis.
- Permitir drill-through para mostrar os dados de detalhe relacionados com determinado valor.

II Conceitos Básicos

Esta seção descreve sucintamente quais os conceitos básicos que são pré-requisitos para o entendimento dos resultados apresentados neste projeto. São também rapidamente descritas as tecnologias utilizadas no projeto, assim como os componentes já disponíveis que contribuíram fortemente para o alcance dos objetivos desejados.

Termos Importantes

Um **cubo** é um banco de dados multidimensional, uma estrutura de dados que armazena dados de forma redundante para obter o máximo de performance para consultas diversas. Um servidor pode armazenar vários cubos. Um **cubo** possui dimensões e medidas associadas, e os dados do cubo são armazenados nos cubos na forma de **células**. As células contêm instâncias de valores correspondentes ao cruzamento das diversas dimensões e medidas. Para cada combinação possível entre membros de dimensões, uma instância da medida pode estar sendo armazenada. Apesar do termo sugerir isto, um cubo não está restrito a três dimensões. Ele pode ter um número irrestrito de dimensões, determinado apenas pelo poder de processamento e armazenamento do hardware e do volume de dados a ser armazenado no cubo.

Na maioria das vezes, os cubos são alimentados a partir de dados originários de outros bancos de dados, de forma a conter cópia de dados a um nível apropriado de detalhamento. Normalmente, os cubos armazenam dados apenas a um nível adequado de detalhe, de forma que os dados de um ambiente de produção convencional são normalmente resumidos para alimentarem a estrutura de um cubo.

Um exemplo de cubo seria o cubo denominado **Vendas** que armazena os detalhes de venda de uma empresa.

Uma **dimensão** pode ser interpretada como um eixo do cubo. No exemplo de um cubo que armazena a informação de vendas da empresa, uma dimensão possível poderia ser a dimensão **Produto**, que representa os produtos efetivamente vendidos. Outro exemplo de dimensão poderia ser a dimensão **Loja**, que representa a loja onde o produto foi vendido.

Uma **medida** é um dado armazenado em uma célula do cubo. Exemplos possíveis de medidas que podem ser utilizadas no cubo de Vendas seria o **lucro** e o **total de vendas**. As medidas representam indicadores importantes que o analisados dos dados pretende acompanhar.

Um **membro** de uma dimensão é um elemento que compõe a dimensão. Por exemplo, membros da dimensão Produto poderiam ser os produtos propriamente ditos: Iogurte, Mostarda, Garrafa,

Vinho, amaciante, pão, cenoura, etc. Membros da dimensão ano podem ser: 1997, 1998, 2000, 2001, 2002, etc.

Uma **hierarquia** representa uma forma de classificar os membros de uma dimensão de maneira hierárquica. Por exemplo, uma hierarquia possível na dimensão **Tempo** pode ser a divisão entre ano, mês e dia. Outra possível hierarquia nesta dimensão poderia ser a divisão entre ano, semestre e trimestre.

Linguagem MDX

A linguagem MDX permite consultar dados em bancos de dados multidimensionais. Ela foi projetada com o objetivo de manter uma semelhança com a sintaxe da linguagem SQL, largamente conhecida e muito utilizada para consultas a bases de dados relacionais.

A linguagem MDX possui dois comandos particularmente úteis que foram usados neste projeto: o comando UPDATE CUBE, usado na implementação do recurso 3, e o comando DRILLTHROUGH, usado na implementação do recurso 4.

Comando UPDATE CUBE

O comando UPDATE CUBE atualiza o valor de uma célula do cubo. A célula pode ser atômica ou em uma hierarquia mais superior. É preciso informar uma medida e membros de cada dimensão para que a célula resultante do cruzamento destes membros seja atualizada. O comando pode ser ilustrado conforme o exemplo a seguir:

```
Update cube [Warehouse] set
([Measures].[Warehouse Sales]
,[Warehouse].[All Warehouses].[USA].[WA]
,[Store].[All Stores].[USA].[WA]
,[Product].[All Products].[Drink].[Beverages].[Carbonated Beverages]
,[Time].[1997].[Q2])
=130
Use_Equal_allocation
```

A primeira linha do comando informa qual cubo deve ser atualizado. Em seguida à cláusula **set** do comando é listada a medida a ser atualizada, como o primeiro item da lista de itens separados por vírgula. Em seguida, são listados os membros de cada dimensão para que a célula a ser atualizada seja determinada pelo cruzamento destes membros. Na penúltima linha acima, é indicado o novo valor que será atribuído à célula. A última cláusula do comando indica qual o valor a ser atribuído para as células que estão contribuindo para o valor da célula que está sendo atualizada. Se a célula que estiver sendo atualizada for o resultado da composição de outros sub-valores, os sub-valores abaixo serão atualizados para valores iguais e de forma que a soma de todos os sub-valores seja igual ao valor desejado. No exemplo acima, se o resultado do cruzamento dos membros indicados for uma célula que contenha uma síntese de 13 registros de dados de detalhe, todos os dados de nível inferior serão atualizados para um valor igual a 10, de forma que a soma deles dê igual ao valor desejado (130).

Comando DRILLTHROUGH

O comando DRILLTHROUGH serve para retornar os registros de detalhe associados com determinado cruzamento de membros de dimensões a partir de um cubo.

O comando permite que a aplicação cliente obtenha o conjunto de registros que foram utilizados para criar uma célula específica no cubo. Um outro comando MDX é usado para especificar a célula em questão. Se esta célula estiver no nível atômico (ou seja, no nível mais baixo de sua hierarquia), apenas um conjunto de registros é retornado. Se o cubo não estiver no nível atômico,

todos os registros que compõem os dados de origem dessa célula são retornados. O número total de registros retornados pode também ser afetado pelo uso da cláusula MAXROWS do comando. O valor especificado para a cláusula MAXROWS indica o número máximo de registros que deverá ser retornado pelo conjunto de registros resultante.

Pode-se observar o exemplo a seguir:

```
Drillthrough maxrows 2500 Select
{[Customers].[All Customers].[USA].[WA]} on 0
,{[Product].[All Products].[Food].[Baked Goods]} on 1
,{[Gender].[All Gender].[F]} on 2
,{[Store Type].[All Store Type].[Mid-Size Grocery]} on 3
,{[Education Level].[All Education Level].[Bachelors Degree]} on 4
,{[Promotions].[All Promotions]} on 5
,{[Promotion Media].[All Media]} on 6
From [Sales]
```

A primeira linha do comando acima indica que serão obtidos apenas os primeiros 2500 registros da consulta. Da segunda até a oitava linha são indicados cruzamentos de membros de dimensões. No exemplo, é feito um cruzamento entre membros das dimensões **Product**, **Customers**, **Store Type**, **Education Level**, **Promotion** e **Promotion Media**. A última linha do comando indica o cubo de onde os dados serão originados.

Recurso de Tabela Dinâmica (PivotTable) do Excel

Um relatório de tabela dinâmica, ou PivotTable, é um recurso nativo do Microsoft Excel. Ele corresponde a uma tabela interativa que combina e compara rapidamente grandes volumes de dados. É possível girar as linhas e colunas para ver diferentes resumos dos dados de origem e pode exibir os detalhes de áreas de interesse.

	A	B	C
1	**Esporte**	**Trimestre**	**Vendas**
2	Golfe	Trim3	R$1.500
3	Golfe	Trim4	R$2.000
4	Tênis	Trim3	R$600
5	Tênis	Trim4	R$1.500
6	Tênis	Trim3	R$4.070
7	Tênis	Trim4	R$5.000
8	Golfe	Trim3	R$6.430

	E	F	G
3	Soma das vendas	Trimestre	
	Esporte	Trim3	Trim4
4	Golfe	R$7.930	R$2.000
	Tênis	R$4.670	R$6.500
	Total geral	R$12.600	R$8.500

Um relatório de tabela dinâmica deve ser utilizado quando se deseja analisar totais relacionados, especialmente quando houver uma longa lista de valores a serem somados e se desejar comparar vários fatos sobre cada valor. No relatório ilustrado na figura, é possível ver com facilidade como as vendas de golfe do terceiro trimestre na célula F3 se comparam com as vendas de outro esporte ou trimestre, ou com o total de vendas. Como um relatório de tabela dinâmica é interativo, é possível alterar o modo de exibição dos dados para ver mais detalhes ou calcular diferentes resumos, como contagens ou médias.

Em um relatório da tabela dinâmica, cada coluna ou campo nos dados de origem se torna um campo de tabela dinâmica que resume várias linhas de informação. Na figura, a coluna **Esporte** se transforma no campo **Esporte** e cada registro de Golfe é resumido em um único item **Golfe**.

Um campo de dados, como Soma de Vendas, fornece os valores a serem resumidos. A célula F3 no relatório acima contém a soma dos valores de Vendas de cada linha dos dados de origem para as quais a coluna Esporte contém Golfe e a coluna Trimestre contém Tri3.

Para criar um relatório de tabela dinâmica, deve-se executar o **Assistente da tabela dinâmica e gráfico dinâmico**. No assistente, selecione os dados de origem desejados na lista de planilhas ou banco de dados externo. O assistente, então, fornecerá uma área de planilha para o relatório e

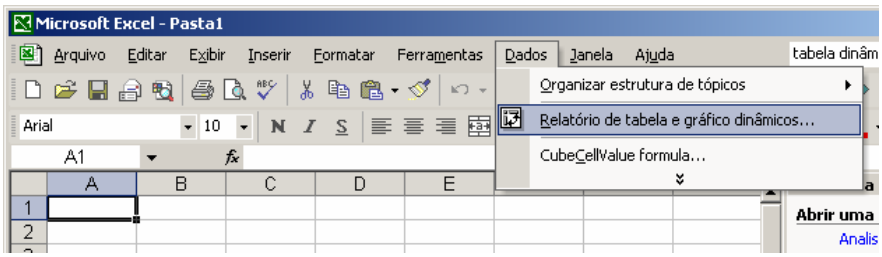
uma lista dos campos disponíveis. À medida que são arrastados os campos da janela de lista para as áreas estruturadas em tópicos, o Microsoft Excel resume e calcula o relatório automaticamente.

Ao usar uma conexão de dados do Office para recuperar dados externos para o relatório, é possível retornar os dados diretamente para um relatório da tabela dinâmica, sem executar o **Assistente da tabela dinâmica e gráfico dinâmico**. A conexão de dados do Office é o método recomendado para recuperação de dados externos para os relatórios quando não é necessário combinar dados de mais de uma tabela no banco de dados externo ou filtrar os dados para selecionar registros específicos antes de criar o relatório, bem como para recuperação de dados de bancos de dados OLAP.

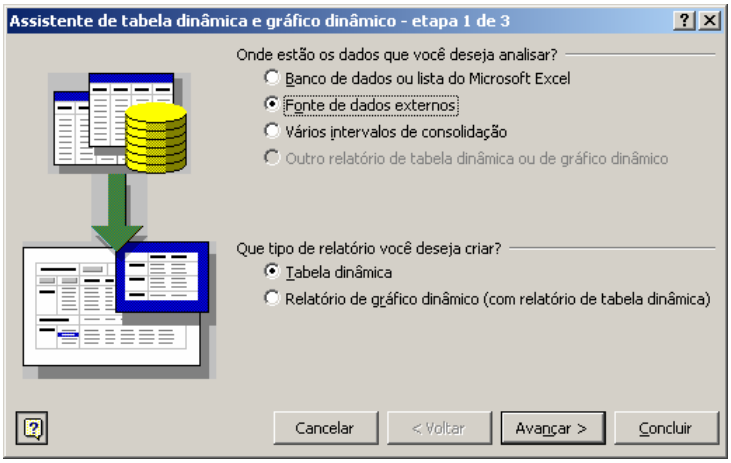
Depois de criar um relatório da tabela dinâmica, pode-se personalizá-lo para enfocar as informações desejadas: alterar o layout, alterar o formato ou pesquisar para baixo para exibir mais dados detalhados.

Uso da Tabela Dinâmica para acessar dados de um cubo

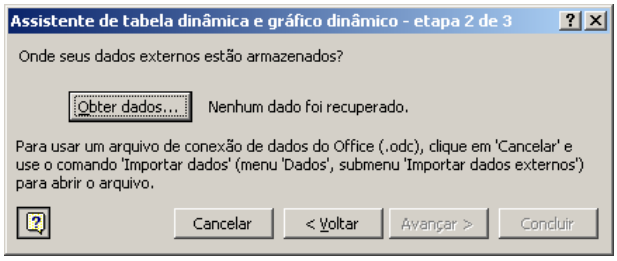
A seqüência de figuras abaixo ilustra, passo a passo, como ativar o recurso de Tabela Dinâmica para acessar dados de um cubo.



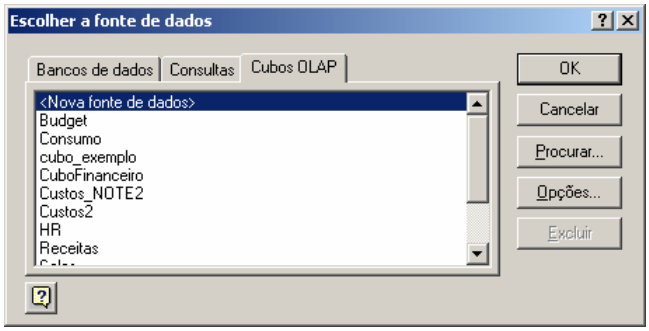
Etapa 1: Escolher a partir do menu "Dados", o item "Relatório de tabela e gráfico dinâmicos...". A caixa de diálogo "Assistente de tabela dinâmica e gráfico dinâmico – etapa 1 de 3" ilustrado pela figura abaixo é mostrada.



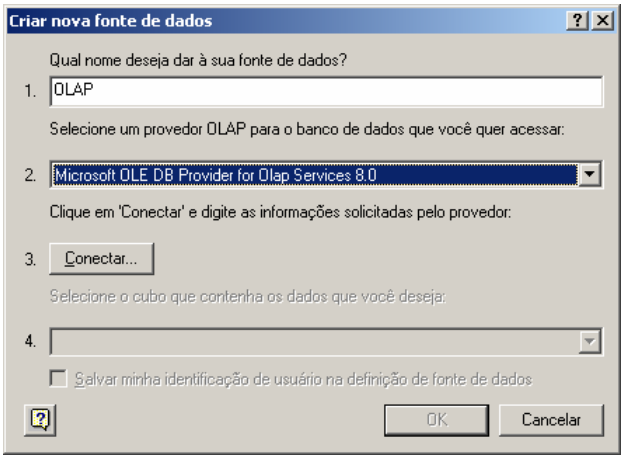
Etapa 2: Escolher a opção "Fonte de dados externos" e clicar em "Avançar". A caixa de diálogo "Assistente de tabela dinâmica e gráfico dinâmico – etapa 2 de 3" ilustrado pela figura abaixo é mostrada.



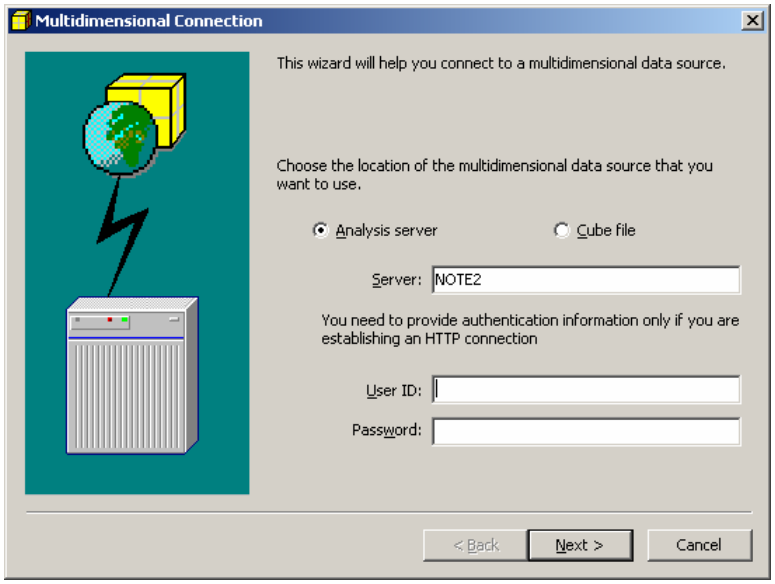
Etapa 3: Clicar sobre o botão "Obter dados". A caixa de diálogo "Escolher a fonte de dados" ilustrado pela figura abaixo é mostrada.



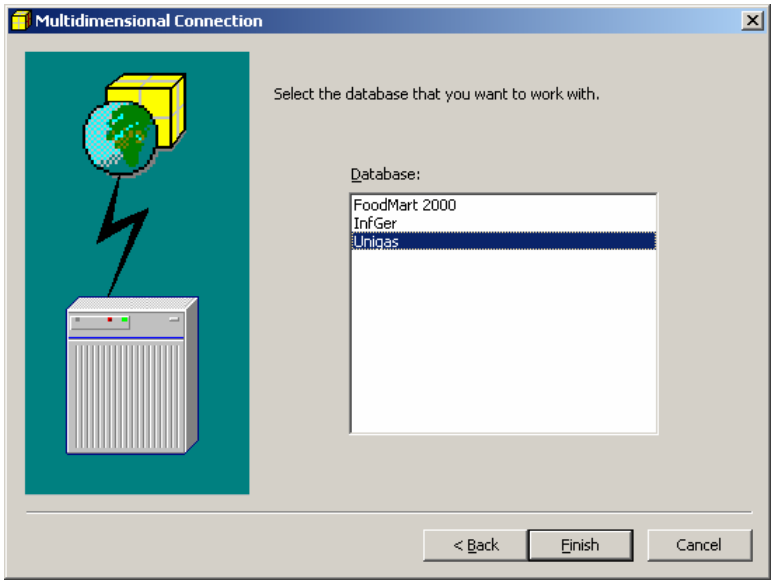
Etapa 4: Ao escolher a aba "Cubos OLAP", será exibida uma lista que mostra as conexões para cubos OLAP já configuradas previamente. Pode-se configurar uma nova conexão, escolhendo a primeira opção da lista, ou usar uma conexão já configurada, escolhendo outra opção da lista. Será detalhada neste procedimento a configuração de uma nova configuração, portanto a primeira opção será escolhida. Ao clicar em OK, a caixa de diálogo "Criar nova fonte de dados" será exibida, conforme ilustrado na figura abaixo.



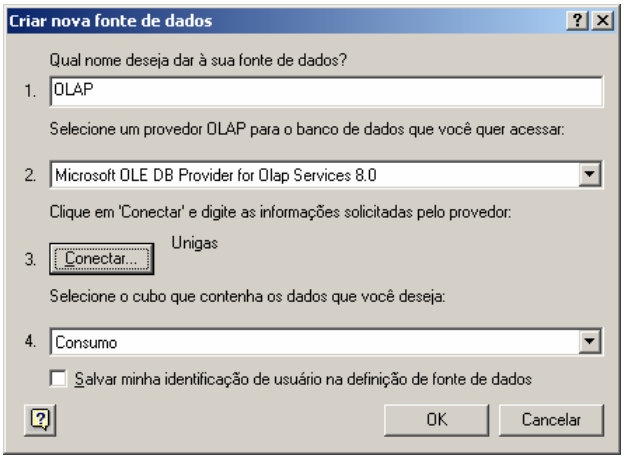
Etapa 5: Definir o nome para a fonte de dados e especificá-lo na caixa de diálogo 1. Neste exemplo, está se criando uma de nome "OLAP". O provedor deverá ser o "Microsoft OLE DB Provider for Olap Services 8.0". Ao clicar em "Conectar", a caixa de diálogo "Multidimensional Connection" será exibida, conforme ilustrado na figura abaixo.



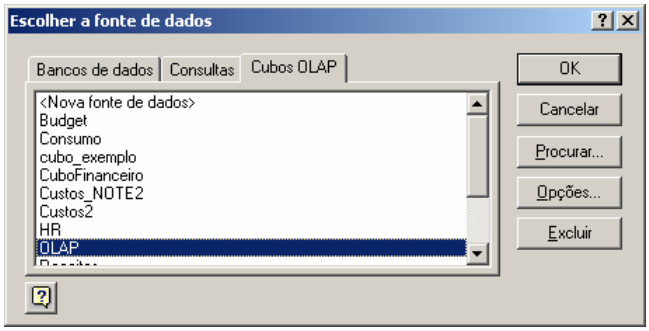
Etapa 6: Escolher a opção "Analysis Server" e especificar o nome do servidor OLAP que está armazenando o cubo. No exemplo, este servidor chama-se "NOTE2". Ao clicar em "Next", a caixa de diálogo para escolha do cubo será mostrada, conforme ilustrado na figura abaixo.



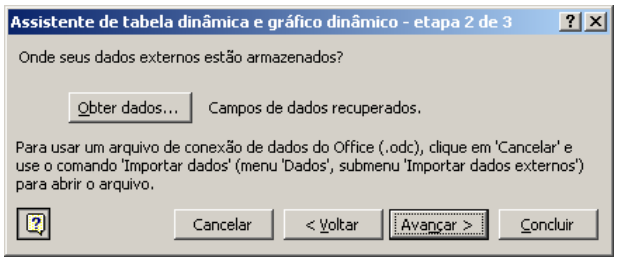
Etapa 7: Escolher a fonte de dados que irá prover as informações desejadas. Uma fonte de dados pode conter vários cubos. No exemplo, está se escolhendo a fonte de dados "Unigas". Ao clicar em Finish, a caixa de diálogo "Criar nova fonte de dados" será novamente mostrada, conforme ilustra a figura abaixo.



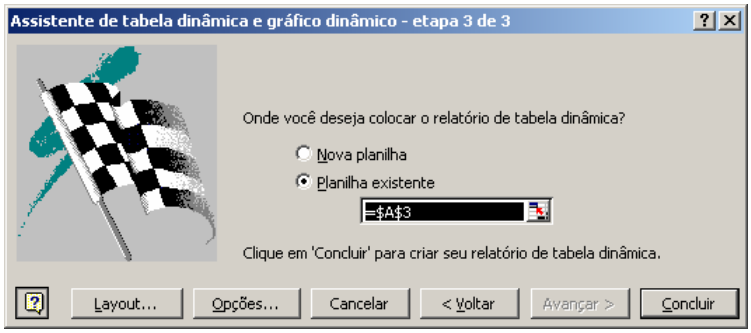
Etapa 8: A caixa de diálogo "Criar nova fonte de dados" mostra um dos cubos disponíveis para a fonte de dados selecionada na etapa anterior. O cubo mostrado é o cubo "Consumo", e pode-se escolher outro cubo disponível. No exemplo, é feita a escolha para o cubo "Consumo". Ao clicar em "OK", é exibida novamente a caixa de diálogo "Escolher a fonte de dados", com o nome da nova fonte de dados que acabou de ser criada, conforme ilustra a figura abaixo.



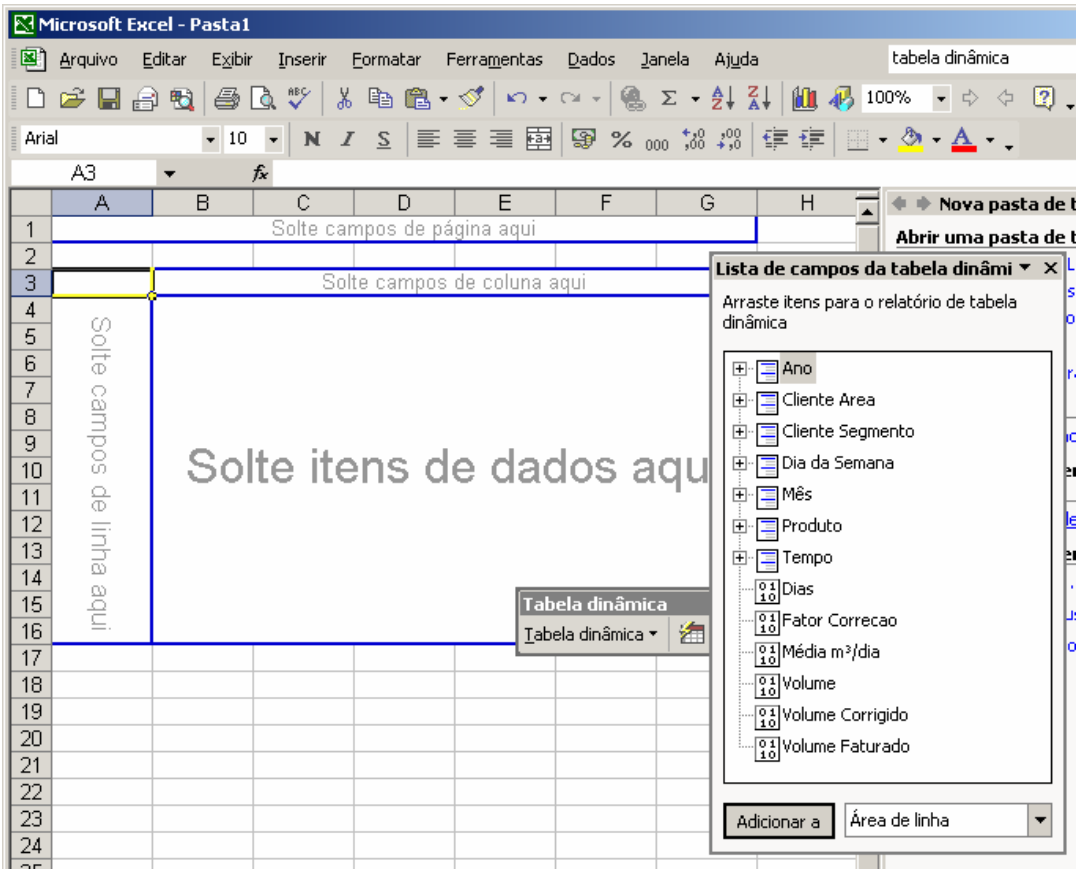
Etapa 9: Ao clicar em "OK", a caixa de diálogo "Assistente de tabela dinâmica e gráfico dinâmico – etapa 2 de 3" é exibida novamente, desta vez com os dados já recuperados, conforme indica a mensagem exibida nesta caixa de diálogo, ilustrada na figura abaixo.



Etapa 10: Ao clicar em "Avançar", é exibida a caixa de diálogo "Assistente de tabela dinâmica e gráfico dinâmico – etapa 3 de 3", ilustrada na figura abaixo.



Etapa 11: Ao clicar em "Concluir" a tabela dinâmica é disponibilizada na planilha para interação do usuário, conforme ilustra a figura abaixo. São mostradas as dimensões e medidas do cubo escolhido em uma janela separada, similar a uma barra de ferramentas do Excel. O usuário pode então arrastar e soltar com o mouse dimensões para a área de campos de coluna ou campos de linha e medidas para a área de itens de dados. No exemplo, as dimensões disponíveis são Ano, Cliente Área, Cliente Segmente, Dia da Semana, Mês, Produto e Tempo. As medidas disponíveis do exemplo são Dias, Fator Correção, Média m^3/dia, Volume, Volume Corrigido e Volume Faturado.



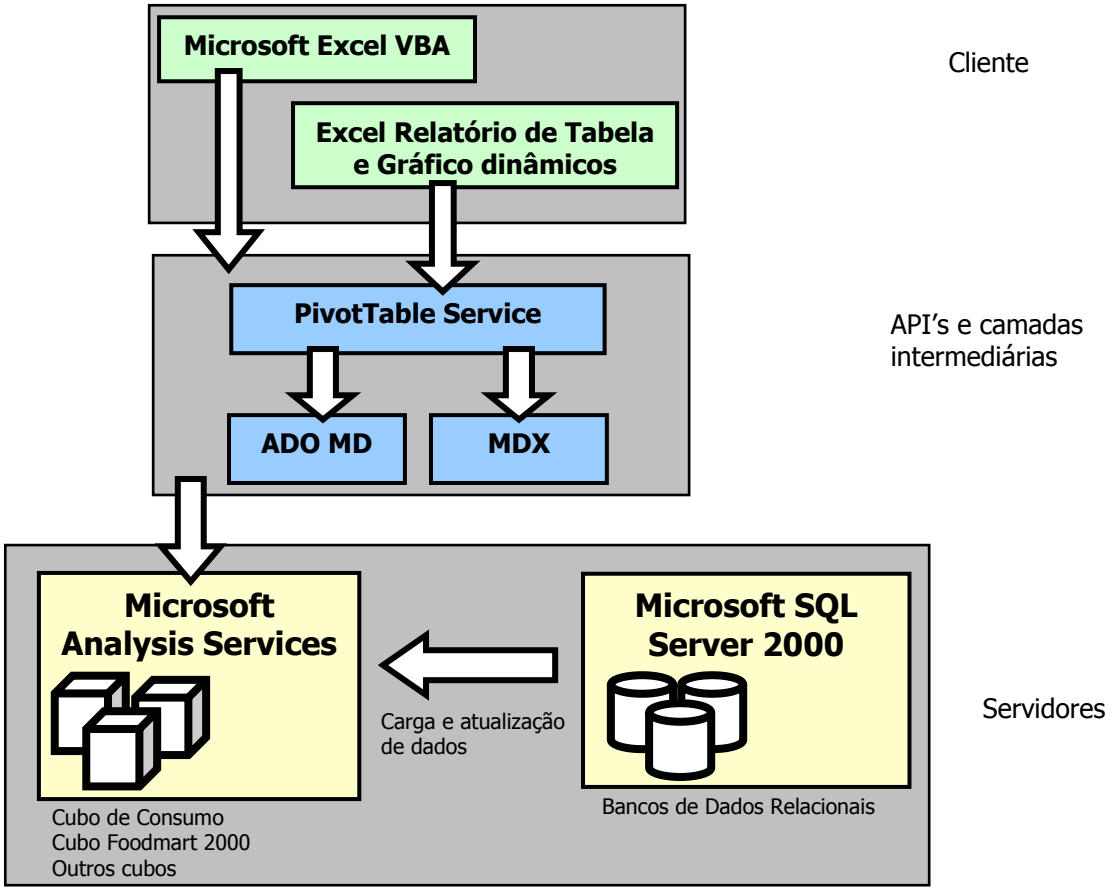
API ADOMD (ActiveX Data Objects Multidimensional)

A biblioteca ou API ActiveX Data Objects Multidimensional (ADOMD) contém um número de objetos que podem ser utilizados por programas para obter informações estruturais e de dados sobre fontes de dados multidimensionais. O objeto Cellset permite que sejam enviados comandos na linguagem MDX para serem processados por um servidor Analysis Server ou para arquivos locais de cubo.

III Componentes do Projeto

O projeto é composto por um número de componentes, necessários para o seu funcionamento adequado.

Os componentes utilizados no projeto se relacionam conforme demonstra o diagrama da figura.



Produtos de Software Necessários

Os requisitos mínimos de produtos de infra-estrutura que precisam estar instalados em um computador para o uso do produto final deste projeto são:

- Microsoft Windows NT/2000/XP
- Microsoft Office XP Standard ou superior
- Microsoft Analysis Services e SQL Server 2000.
- Microsoft Analysis Services Service Pack 1 ou superior.

Tecnologias, Linguagens e API's utilizadas

- Microsoft ActiveX Data Objects Multidimensional (ADOMD).
- Microsoft Excel Visual Basic for Applications (VBA)
- Linguagem MDX (Multidimensional Expressions)

Bancos de Dados OLAP

Alguns bancos OLAP são utilizados no projeto como exemplos e sua instalação é necessária para que os recursos apresentados funcionem adequadamente:

- Foodmart 2000
- Cubo de Consumo

Arquivos

O arquivo **AnalysisClient.xls** contém os códigos detalhados neste documento e alguns exemplos em quatro planilhas, ilustrando os recursos implementados.

IV Fases do Desenvolvimento do Projeto

Abaixo são detalhadas as etapas fundamentais para o desenvolvimento do projeto: Levantamento Inicial, Implementação, Testes e escrita do Relatório Final.

Levantamento inicial

Para o levantamento inicial, foram feitas diversas pesquisas na Web. As pesquisas possibilitaram verificar que há vários clientes OLAP disponíveis no mercado. Entretanto, o custo das soluções é alto e muitas vezes é necessário instalar um programa cliente adicional na máquina para realizar análises multidimensionais.

As empresas Business Objects, Cogno e outras possuem soluções completas para clientes OLAP, mas muitos deles são programas externos que precisam ser instalados. Alguns estão disponíveis já no Excel, mas o custo de aquisição do produto é muitas vezes proibitivo para a adoção do mesmo. Considerando que o mercado OLAP tem como demanda um público-alvo muito amplo no futuro, faz-se necessário ferramentas cliente de custo mais acessível.

Muitas pesquisas foram também realizadas no Help online do Microsoft Excel VBA e do Microsoft Analysis Services Books Online, documentação online integrante do servidor de BD multidimensional utilizado no projeto.

As pesquisas coletaram informações valiosas sobre OLAP, MDX, ADO MD e VBA. Foi feita a identificação de produtos de software e hardware existentes no mercado.

Preparação de Base de Testes

A base de testes foi construída baseada em um ambiente real, onde há dados históricos disponíveis por sete anos. O longo período de disponibilidade dos dados é fundamental para análises mais extensas sobre os dados. Foram implementadas algumas medidas de indicadores importantes, tais como a média de m^3/dia. Esta é uma medida calculada, que depende de uma grandeza variável, o número de dias em análise. A base de testes foi construída a partir de um Banco de Dados SQL Server, e as grandezas foram definidas baseadas apenas nos dados de consumo, envolvendo dimensões e membros periféricos a esses dados. Um cubo foi construído com sete dimensões e seis medidas. Há campos derivados e medidas hierárquicas.

Outra base de testes importante para a implementação deste projeto foi a base de exemplo disponibilizada junto com o Analysis Services da Microsoft. A base chama-se Foodmart 2000 e contém dados relacionados a uma rede fictícia de supermercados.

Implementação, Testes e Escrita do Relatório Final

O processo de implementação e testes foi na maioria das vezes concorrente. Após a implantação de determinado recurso, testes eram realizados sobre o mesmo com vários casos. A escrita do relatório final foi feita depois de realizados todos os trabalhos de pesquisa, implementação e testes.

V Recursos Implantados

Recurso 1: "Side Analysis"

O recurso de "Side Analysis", ou Análise ao Lado, foi implementado para possibilitar a inclusão de linhas e colunas no corpo dos dados advindos do cubo. Como isso não é possível diretamente a partir da Tabela Dinâmica, uma rotina constrói em uma nova planilha uma outra tabela mostrando os dados exibidos na Tabela Dinâmica da planilha original através de *links* para a tabela dinâmica. Os cabeçalhos de campo são exibidos conforme referência às células correspondentes na tabela dinâmica e os valores do corpo da tabela são obtidos através do uso da fórmula INFODADOSTABELADINÂMICA. Dessa forma, mesmo com o uso de filtros ou caso as células mudem de posição na tabela dinâmica, as referências continuam válidas. O uso da função de planilha INFODADOSTABELADINÂMICA permite uma referência a um cruzamento de dados da tabela dinâmica, ao invés de fazer referência a uma célula específica. Por exemplo, ao utilizar em uma célula do Excel a fórmula:

```
=INFODADOSTABELADINÂMICA("[Measures].[Média m³/dia]";Plan1!$A$3;"[Tempo]";"[Tempo].[All
Tempo].[1996]";"[Cliente].[Area]";"[Cliente].[Area].[All
Cliente].[Norte]";"[Produto]";"[Produto].[All Produto].[Gás Automotivo]")
```

Estará sendo exibido o valor de uma célula da tabela dinâmica, da medida "Média m^3/dia", cruzando os membros "1996" da dimensão Tempo, a área "Norte" e o Produto "Gás Automotivo". Ao não fazer uma célula da planilha no formato tradicional (por exemplo, "=F10"), o local exato da planilha onde o dado está situado fica pouco relevante para a exibição do mesmo, e há uma confiança maior que mudanças na planilha original não irão causar distorções no resultado da fórmula INFODADOSTABELADINÂMICA.

Outra vantagem da planilha de Análise ao Lado é permitir qualquer formatação e manuseio manual com a planilha que o usuário desejar.

Recursos Adicionais Implantados

Um recurso adicional importante foi implantado. Muitos usuários da tabela dinâmica como cliente de OLAP no Excel têm reinvidicado que a exibição dos cabeçalhos das linhas é feita propagando os dados ao longo das colunas. Entretanto, muitos usuários preferem a exibição dos cabeçalhos das linhas em uma única coluna.

Foi implantado um recurso adicional que preenche na planilha para "Side Analysis" os cabeçalhos de linha em uma única coluna. Com isso, os dados são exibidos conforme solicitação de muitos usuários de OLAP. Por exemplo, os dados exibidos na tabela dinâmica da seguinte forma (ocupando três colunas):

Gás Automotivo	Norte	POSTO BR-11
		POSTO CAENGA
		POSTO CARTAXO
		POSTO CEMOPEL
		POSTO CEMOPEL 2
		POSTO D. COELHO
		POSTO Duque Caxias
		POSTO M. DE NASSAU
		POSTO REAL
		POSTO São Carlos
		POSTO SUPERGAS
	Norte Total	

Serão exibidos para Análise ao Lado da seguinte forma (ocupando uma única coluna):

Gás Automotivo
Norte
POSTO BR-11
POSTO CAENGA
POSTO CARTAXO
POSTO CEMOPEL
POSTO CEMOPEL 2
POSTO D. COELHO
POSTO Duque Caxias
POSTO M. DE NASSAU
POSTO REAL
POSTO São Carlos
POSTO SUPERGAS

O espaçamento de identação dos itens é uma constante definida no código e é facilmente configurável. No exemplo da implementação, a identação é de quatro espaços.

Desafios da Implantação

O recurso de "Side Analysis" teve seu processo de implantação bastante demorado e complexo. Foram necessárias várias tentativas utilizando diversas propriedades dos objetos existentes no VBA do Excel para se chegar a uma solução satisfatória. Em algumas das tentativas realizadas, as soluções obtidas não tinham êxito em todos os casos. Alguns dos fatores se tornam grandes desafios para o sucesso da implantação deste recurso:

- O objeto PivotTable, a coleção PivotFields, o objeto PivotField, a coleção PivotItems e o objeto PivotItem, apesar de possuírem diversos métodos e propriedades, não dispõem de algumas propriedades fundamentais que permitiriam uma implantação mais fácil. Não existem propriedades que informe a ordem dos campos na escolha pelo usuário; é necessário escrever um programa que navegue pelas células da planilha para se chegar até essa informação. A coleção PivotFields não mantém a ordenação de acordo com a ordem dos campos exibida na tabela dinâmica.

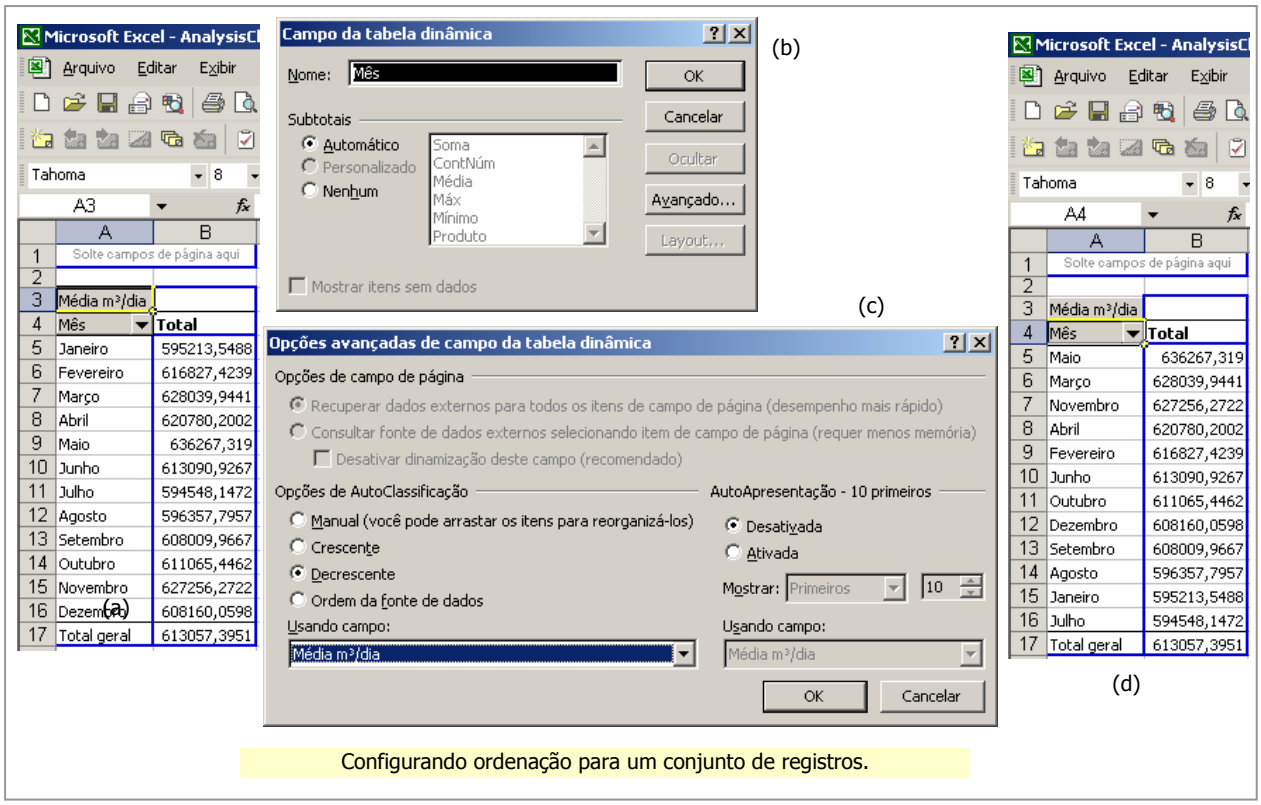
- Não existem métodos para hierarquizar os nomes dos membros (PivotItems) na coleção PivotFields. O nome de cada membro é obtido através de uma única propriedade String, não sendo dividido qual é o nome efetivo do membro e o nome da hierarquia anterior a este nome.

- É impossível descobrir qual o membro de uma coleção PivotItems apenas com o seu nome de exibição. Como apenas esses nomes são exibidos na planilha para análise paralela, é necessário que uma estrutura auxiliar seja construída para traçar a correspondência entre o nome e a célula que corresponde ao mesmo na tabela dinâmica na planilha original.

Recurso 2: Ordenação de dados e "n primeiros"

O recurso de ordenação de dados foi o mais simples de se obter. Durante as pesquisas iniciais, foi descoberto que este recurso já faz parte do Office XP, na opção "Avançada". Basta dar duplo clique sobre a dimensão desejada para ativar o recurso.

O exemplo a seguir demonstra a seqüência de passos necessária para ativar ordenação e exibição dos 5 primeiros registros em um conjunto exemplo de dados. A métrica utilizada é "Média m^3/dia" e apenas é exibida a dimensão "Mês" nas linhas, sem cruzamento com dimensões nas colunas.



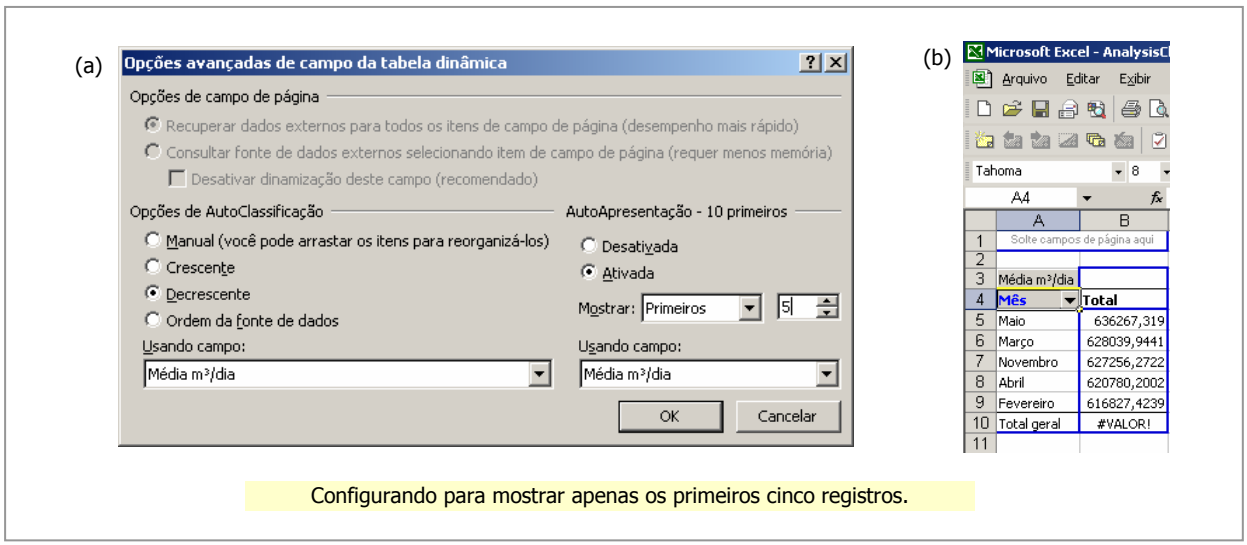
Configurando ordenação para um conjunto de registros.

A figura acima ilustra o processo de configuração para ordenação de dados do cubo. Na figura (a) estão sendo exibidos na tabela dinâmica o campo Mês e a métrica Média m^3/dia. Ao dar um duplo clique sobre a célula A4, que contém o nome do campo analisado nas linhas ("Mês"), a janela da figura (b) é mostrada. Ao clicar sobre o botão **Avançado**, é mostrada a tela da figura (c). Nesta tela, foi escolhido em "Opções de AutoClassificação" a opção 'Descrescente' e o campo Média m^3/dia. Ao pressionar em OK, é mostrada a planilha conforme mostra a figura (d), ordenada pela métrica das medidas de forma descrescente.

Ainda há possibilidade de se exibir os dados ordenados de forma crescente, usando a ordem da fonte de dados (a opção padrão) ou manual. Na opção manual, o usuário pode arrastar e soltar os itens para as posições desejadas.

A figura abaixo mostra como podem ser exibidos os primeiros n registros. Isso pode ser feito na tela de "Opções avançadas de campo da tabela dinâmica", na seção "AutoApresentação – 10 primeiros", que aparece após o usuário ter clicado sobre o botão "Avançado" do campo da tabela dinâmica. Na figura (a) abaixo, é mostrada a janela de "Opções avançadas de campo da tabela dinâmica". Foi escolhido para serem exibidos os primeiros 5 registros, usando o campo "Média

m^3/dia". Após clicar em OK, o usuário visualiza na planilha conforme mostrado na figura (b), mostrando apenas os primeiros cinco meses.



Configurando para mostrar apenas os primeiros cinco registros.

Recurso 3: Write-back e Cenários "e-se?"

Muitos usuários típicos de OLAP precisam realizar uma análise "e-se" com uma tabela dinâmica. O Analysis Services do SQL Server inclui um recurso chamado write-back, que permite que aplicações cliente gravem alterações para os dados de um cubo.

O Excel não tem suporte nativo para realizar operações de write-back. A implementação descrita a seguir permite utilizar um evento do Excel para capturar modificações feitas a uma tabela dinâmica, e posteriormente construir e submeter um comando MDX para o Analysis Services para atualizar um dado do cubo. A funcionalidade é feita para que o usuário apenas digite o valor desejado na célula da tabela dinâmica e pressione ENTER. O comando MDX é gerado e submetido para o Analysis Services para realizar a atualização adequada nos dados do cubo.

Recurso 4: Drillthrough

Os cubos OLAP que compõem os data warehouses contêm agregações de grandes quantidades de dados multidimensionais. Há ocasiões onde o usuário pretende visualizar os registros de detalhe que contribuem para um valor agregado mostrado na tabela dinâmica. O Analysis Services do SQL Server 2000 introduz um recurso de **drillthrough**, que permite realizar uma "abertura através" de um valor agregado para revelar os registros de detalhe que o mesmo representa.

O modelo de objetos do Excel não provê suporte nativo paras realizar operações de drillthrough. Foi feita uma implementação que realiza rápidas operações drillthrough usando a API do ADOMD (ActiveX Data Objects Multidimensional).

Um comando de menu chamado **OLAP Drillthrough** é adicionado ao menu de contexto da PivotTable. Quando o usuário seleciona o item de menu enquanto em uma área de dados de uma tabela dinâmica vinculada a um cubo OLAP, a consulta de drillthrough apropriada é submetida ao Analysis Services e os registros de detalhe são escritos em uma nova planilha.

O item de menu é criado no evento **Workbook_Open** da mesma forma que o item de menu **OLAP Side Analysis**.

VI Detalhes da Implementação

Recurso 1: "Side Analysis"

Um comando de menu chamado **OLAP Side Analysis** é adicionado ao menu de contexto da PivotTable. Ao clicar com o botão direito do mouse sobre a tabela dinâmica, é criada uma planilha adicional com os dados vinculados.

O item de menu é criado no evento **Workbook_Open**, junto com o item de menu **OLAP Drillthrough**.

```
Private Sub Workbook_Open()
   Call CriaItemMenuContexto("PivotTable context menu", "OLAP Drillthrough", "Drillthrough")
   Call CriaItemMenuContexto("PivotTable context menu", "OLAP Side Analysis", "SideAnalysis")
End Sub
```

A rotina auxiliar que adiciona o menu de contexto é listada a seguir:

```
Public Sub CriaItemMenuContexto(menu As String, capt As String, nomeMacro As String,
  Optional deleta As Boolean = False)
   Dim ptcon As CommandBar
   Dim cmdDrill As CommandBarControl

  'Consulte a seguinte lista para os menus do Excel:
  'http://support.microsoft.com/support/kb/articles/Q213/5/52.ASP
  'Título: XL2000: List of ID Numbers for Built-In CommandBar Controls
  Set ptcon = Application.CommandBars(menu)

  For Each btn In ptcon.Controls
    If btn.Caption = capt Then
      If deleta Then btn.Delete
      GoTo done
    End If
  Next btn

  ' Adiciona um item ao menu de contexto PivotTable
  Set cmdDrill = ptcon.Controls.Add(Type:=msoControlButton, temporary:=True)
  cmdDrill.Caption = capt
  cmdDrill.OnAction = nomeMacro

done:
End Sub
```

Abaixo é descrito o código da implantação do recurso de "Side Analysis". Há quatro rotinas principais: a principal cria a planilha adicional e realiza o preenchimento de todas as células do corpo da tabela; duas funções auxiliares que buscam o nome completo dos membros da coluna e linha; outra função auxiliar retorna uma string envolvida por aspas.

Rotina Principal

A primeira parte cria uma planilha com o nome SideAnalysis[n], onde [n] é um contador. Caso já exista a planilha SideAnalysis0, uma planilha será criada com o nome SideAnalysis1. Caso já exista a planilha SideAnalysis1, uma planilha será criada com o nome SideAnalysis2, e assim por diante.

As variáveis **po** e **pd** são globais e utilizadas por várias funções. Elas são definidas no início deste procedimento.

```
Public Sub SideAnalysis()
  Dim sh As Worksheet
  Dim NomeDestino As String, idx As Integer, idx1 As Integer

  idx = 0
  NomeDestino = "SideAnalysis"
```

```
  'Descobre um nome que garanta unicidade para a planilha que conterá os dados do destino
  For Each sh In Sheets
    If Left(sh.Name, Len(NomeDestino)) = NomeDestino Then
      idx1 = Val(Mid(sh.Name, Len(NomeDestino) + 1))
      If idx1 >= idx Then idx = idx1 + 1
    End If
  Next sh
  Set po = ActiveSheet
  Set pd = Sheets.Add(ActiveSheet)
  pd.Name = NomeDestino & idx
```

Em seguida, são preenchidos os valores dos cabeçalhos de coluna. É levado em consideração que os membros das linhas estarão dispostos em uma única coluna. Portanto, um recuo é necessário para exibição dos membros das colunas. A rotina teve que ser desenvolvida usando métodos que obtinham a posição das células (DataRange) onde os membros das colunas estão dispostos, porque os métodos que trazem os membros propriamente ditos (PivotItems) não estão dispostos em ordem conveniente. É também preenchida a coluna dos totais, pois normalmente ela não faz parte dos dados lidos através do método DataRange.

```
  Dim fld As PivotField
  Dim rg As Range
  Dim cl As Range

  For Each fld In po.PivotTables(1).ColumnFields
    For Each cl In Range(po.Name & "!" & fld.DataRange.Address)
      'Preenche os membros nas colunas. Leva em consideração o número de níveis das linhas.
      If cl <> "" Then
        pd.Range(cl.Address).Offset(0, 1 - po.PivotTables(1).rowFields.Count).Value = cl.Value
      End If
    Next cl
  Next fld
  'Se ficar faltando a coluna de totais, preenche ela também
  Set cl = po.PivotTables(1).DataLabelRange.Offset(1, po.PivotTables(1).rowFields.Count +
po.PivotTables(1).ColumnFields(1).DataRange.Count)
  If cl <> "" Then
    pd.Range(cl.Address).Offset(0, 1 - po.PivotTables(1).rowFields.Count).Value = cl.Value
  End If
  cl.Font.Bold = True
```

Em seguida, são preenchidos os membros das linhas. Estes membros são preenchidos utilizando apenas uma coluna, mostrando a identação de cada membro utilizando espaços. Durante o preenchimento, é construída uma matriz, rowMembers, para guardar as células da tabela dinâmica original. Esta matriz é definida como variável global e é utilizada em outras funções para traçar a correspondência entre os dados resultantes da planilha de Análise ao Lado e as células correspondentes da tabela dinâmica.

```
  'Preenche os membros nas linhas. O algoritmo de preenchimento é diferente do que é usado nas
colunas,
  'para linearizar os itens
  Dim ro As Integer, co As Integer, rod As Integer, rng As Range

  ro = 0: co = 0: rod = 0
  ReDim RowMembers(po.PivotTables(1).rowFields(1).DataRange.Count *
po.PivotTables(1).rowFields.Count)
  Set rng = Range(po.Name & "!" & po.PivotTables(1).rowFields(1).DataRange.Address)(1)
  While ro <= po.PivotTables(1).rowFields(1).DataRange.Count
    For co = 0 To po.PivotTables(1).rowFields.Count - 1
      If rng.Offset(ro, co) <> "" And rng.Offset(ro, co) <> 0
        And Right(po.Range(rng.Offset(ro, co).Address), 6) <> " Total" Then
        pd.Range(rng.Offset(rod, 0).Address).Formula = "=""" & Space(IDENT SPACE * co) & """ &
" & po.Name & "!" & rng.Offset(ro, co).Address
        Set RowMembers(rod) = rng.Offset(ro, co)
        If Left(pd.Range(rng.Offset(rod, 0).Address), 6) = "Total " Then
pd.Range(rng.Offset(rod, 0).Address).Font.Bold = True
```

```
        rod = rod + 1
      End If
    Next co
    ro = ro + 1
  Wend
```

Agora é feito o preenchimento das células do corpo da tabela, utilizando a fórmula GETPIVOTDATA (versão em inglês da fórmula INFODADOSTABELADINÂMICA). Um requerimento do VBA é que qualquer definição de fórmula a partir de código VBA deve ser feita usando a sintaxe das fórmulas em inglês, e o separador de parâmetros tem que ser a vírgula, ao invés do ponto-e-vírgula utilizado no Excel em português. Após o preenchimento do corpo da tabela, é feito o preenchimento dos totais de linha e coluna, e do total geral, separadamente.

```
  'Preenche as fórmulas da tabela do meio usando GETPIVOTDATA
  Dim pivotAddress As String, measure As String

  pivotAddress = po.Name & "!" & Range(Range(po.PivotTables(1).RowRange.Address)(1),
Range(po.PivotTables(1).ColumnRange.Address)(1))(1).Address
  measure = po.PivotTables(1).DataFields(1).CubeField.Name
  Set rng = Range(pd.Name & "!" & po.PivotTables(1).DataBodyRange.Address)(1).Offset(0, 1 -
po.PivotTables(1).rowFields.Count)
  For co = 0 To po.PivotTables(1).ColumnFields(1).DataRange.Count - 1
    For ro = 0 To po.PivotTables(1).rowFields(1).DataRange.Count - 1
      rng.Offset(ro, co).Formula = "=GETPIVOTDATA(" & Aspas(measure) & PARAM SEPARATOR
        & po.Name & "!" & po.PivotTables(1).DataLabelRange.Address & PARAM_SEPARATOR _
        & StrColumnMembers(po.PivotTables(1), co) & PARAM_SEPARATOR &
StrRowMembers(po.PivotTables(1), ro) & ")"
    Next ro
  Next co

  'Preenche os totais de linha
  co = po.PivotTables(1).ColumnFields(1).DataRange.Count
  For ro = 0 To po.PivotTables(1).rowFields(1).DataRange.Count - 1
    rng.Offset(ro, co).Formula = "=GETPIVOTDATA(" & Aspas(measure) & PARAM SEPARATOR
      & po.Name & "!" & po.PivotTables(1).DataLabelRange.Address & PARAM SEPARATOR
      & StrRowMembers(po.PivotTables(1), ro) & ")"
  Next ro

  ro = po.PivotTables(1).rowFields(1).DataRange.Count
  'Preenche os totais de coluna
  For co = 0 To po.PivotTables(1).ColumnFields(1).DataRange.Count - 1
    rng.Offset(ro, co).Formula = "=GETPIVOTDATA(" & Aspas(measure) & PARAM_SEPARATOR _
      & po.Name & "!" & po.PivotTables(1).DataLabelRange.Address & PARAM_SEPARATOR _
      & StrColumnMembers(po.PivotTables(1), co) & ")"
  Next co

  'Preenche o total geral
  rng.Offset(po.PivotTables(1).rowFields(1).DataRange.Count,
po.PivotTables(1).ColumnFields(1).DataRange.Count).Formula = _
    "=GETPIVOTDATA(" & Aspas(measure) & PARAM SEPARATOR
      & po.Name & "!" & po.PivotTables(1).DataLabelRange.Address & ")"
```

Funções Auxiliares

A função abaixo, **StrColumnMembers**, retorna o parâmetro de campos e membros para uma coluna específica já formatados para a função INFODADOSTABELADINÂMICA. Isso é obtido através da navegação dos itens correspondentes na tabela dinâmica. O algoritmo navega pelos campos da tabela dinâmica listando cada campo apenas uma vez. Caso haja campos hierárquicos que estejam expandidos na tabela dinâmica, os mesmos irão ser considerados na fórmula apenas uma vez.

```
Public Function StrColumnMembers(ByRef pt As PivotTable, col As Integer) As String
  Dim result As String, ro As Range, rng As Range
  Dim i As Integer
```

```
  Dim lastfield As String, lastMember As String

  'ro - guarda a célula do início dos headers das colunas na planilha da pivotTable
  Set ro = Range(po.Name & "!" & pt.ColumnRange.Address)(1)

  i = pt.ColumnFields.Count
  result = ""
  While i > 0
    Set rng = ro.Offset(i, col)
    If lastfield <> rng.PivotCell.PivotField.CubeField.Name Then
      lastfield = rng.PivotCell.PivotField.CubeField.Name
      lastMember = rng.PivotCell.PivotItem.Name
      result = result & Aspas(rng.PivotCell.PivotField.CubeField.Name) & PARAM_SEPARATOR &
Aspas(rng.PivotCell.PivotItem.Name) & PARAM SEPARATOR
    End If
    i = i - 1
  Wend
  StrColumnMembers = Left(result, Len(result) - 1)
End Function
```

A função abaixo, **StrRowMembers**, retorna o parâmetro de campos e membros para uma linha específica já formatados para a função INFODADOSTABELADINÂMICA. Isso é obtido através da navegação dos itens correspondentes na tabela dinâmica. O algoritmo é similar ao da função StrRowMembers, mas a função StrRowMembers utiliza a matriz previamente construída, **RowMembers**.

```
Public Function StrRowMembers(ByRef pt As PivotTable, rw As Integer) As String
  Dim result As String, rng As Range
  Dim i As Integer
  Dim lastfield As String, lastMember As String

  Set rng = RowMembers(rw)
  i = 1
  result = ""
  While i <= rng.Column
    If lastfield <> rng.Offset(0, 1 - i).PivotCell.PivotField.CubeField.Name Then
      lastfield = rng.Offset(0, 1 - i).PivotCell.PivotField.CubeField.Name
      lastMember = rng.Offset(0, 1 - i).PivotCell.PivotItem.Name
      result = result & Aspas(rng.Offset(0, 1 - i).PivotCell.PivotField.CubeField.Name) &
PARAM SEPARATOR & Aspas(rng.Offset(0, 1 - i).PivotCell.PivotItem.Name) & PARAM SEPARATOR
    End If
    i = i + 1
  Wend
  If Len(result) > 0 Then StrRowMembers = Left(result, Len(result) - 1)
End Function
```

A função auxiliar Aspas apenas retorna uma String com aspas envolvendo o parâmetro.

```
Public Function Aspas(st As String) As String
  Aspas = """" & st & """"
End Function
```

Recurso 3: Write-back e Cenários "e-se?"

Eventos

A rotina a seguir é executada no evento Worksheet_change, que é invocado quando uma célula da planilha foi modificada:

```
Private Sub Worksheet Change(ByVal Target As Range)
  Dim cell As PivotCell

  ' Err.Number retorna 0 quando este evento é chamado da planilha, e um valor diferente quando
  ' o 'desfazer' é invocado no procedimento Writeback
  If Err.Number <> 0 Then GoTo done

  ' Este evento é chamado novamente pelo procedimento Writeback.
  ' Isso sai dessa chamada, assim como outras que não interessam.
```

```
  If TypeName(Target.Value) <> "Double" Then GoTo done

  On Error GoTo done

  ' Atribui um objeto PivotCell. Se a mudança não for em uma PivotTable, a rotina de erro é
chamada.
  Set cell = Target.PivotCell

  On Error GoTo 0

  ' Se a célula da PivotTable estiver sendo modificada, então processe ele como uma
  ' operação de Writeback
  If cell.PivotCellType = xlPivotCellValue Then
    Call Writeback(cell, Target.Value)
  End If

done:
  On Error GoTo 0
End Sub
```

Procedimento Writeback

O procedimento de Writeback é o responsável pela construção e envio do comando de atualização do cubo, "UPDATE CUBE", para o Analysis Services.

A primeira parte do procedimento de Writeback aloca as variáveis a serem utilizadas pelo restante do procedimento e faz as conexões necessárias para enviar agregações de dados para o Analysis Services. O objeto **PivotCell** passado para a procedure é a chave para configurar os objetos **PivotTable**, **PivotCache**, ADO **Command** e ADO **Connection**.

```
Sub Writeback(ByVal pcell As PivotCell, ByVal novoValor As Double)
  Dim Cmd As New ADODB.Command
  Dim Conn As New ADODB.Connection
  Dim pt As PivotTable
  Dim pcache As PivotCache
  Dim pf As PivotField
  Dim headers(1 To 2) As PivotItemList
  Dim comandoMDX As String
  Dim tmp As String
  Dim lastHeader As String
  Dim i As Integer
  Dim k As Integer

  ' A maior parte do processamento ocorrerá fora do Excel, e o cursor do mouse deve indicar
isso.
  Application.Cursor = xlWait

  Set pt = pcell.Parent
  Set pcache = pt.PivotCache

  If Not pcache.IsConnected Then pcache.MakeConnection  'Garante que o cache do PivotTable
está conectado.

  Set Cmd.ActiveConnection = pcache.ADOConnection
  Set Conn = Cmd.ActiveConnection
```

Em seguida, a string que irá conter o comando completo para atualizar o cubo é inicializada. O primeiro laço constrói a parte do comando que faz referência aos campos de página ou filtros.

```
  ' A variável comandoMDX armazena o comando para enviar para o Analysis Services para
realizar a alocação.
  comandoMDX = pcell.DataField.Name & ","

  ' Adiciona cada campo de página (filtro) para a variável comandoMDX.
  If pt.PageFields.Count > 0 Then
    For Each pf In pt.PageFields
      comandoMDX = comandoMDX & pf.CurrentPageName & ","
    Next pf
```

```
  End If
```

A próxima seção adiciona os itens apropriados de linha e coluna para a variável comandoMDX. A variável headers é uma matriz de objetos PivotItems. O primeiro item da matriz é a coleção de objetos **RowItems** (propriedade da **PivotCell**) e o segundo item da matriz é a coleção de objetos **ColumnItems** (propriedade da **PivotCell**).

```
  Set headers(1) = pcell.RowItems
  Set headers(2) = pcell.ColumnItems

  For k = 1 To 2
    If headers(k).Count > 0 Then
      tmp = ""
      lastHeader = headers(k)(1).Parent.CubeField.Name

      ' Esse loop apenas adiciona para comandoMDX quando muda o nome do campo.
      For i = 1 To headers(k).Count
        If headers(k)(i).Parent.CubeField.Name = lastHeader Then
          tmp = headers(k)(i)
        Else
          comandoMDX = comandoMDX & tmp & ","
          lastHeader = headers(k)(i).Parent.CubeField.Name
          tmp = headers(k)(i)
        End If
      Next i

      ' O último item de linha é sempre o nível mais baixo e portanto adicionado a comandoMDX.
      tmp = headers(k)(headers(k).Count)
      comandoMDX = comandoMDX & tmp & ","
    End If
  Next k
```

Em seguida, o comando MDX final é montado, e depois é enviado para atualizar o valor agregado através do objeto ADO Command no Analysis Server. No final, a atualização é gravada (commit) e o PivotTable é atualizado para mostrar os subtotais onde este valor atualizado irá refletir.

```
  comandoMDX = Left(comandoMDX, Len(comandoMDX) - 1) 'Tira a vírgula

  ' Cria o comando final para enviar ao Analysis Server e executa o mesmo.
  comandoMDX = "Update cube " & "[" & pcache.CommandText & "]" & " set (" & _
           comandoMDX & ")=" & novoValor & " Use_Equal_allocation"
  Cmd.CommandText = comandoMDX
  Cmd.Execute

  On Error GoTo writefailed

  ' Abre um novo objeto Transaction e grava a modificação
  Conn.Attributes = adXactCommitRetaining
  Conn.CommitTrans

  pcache.Refresh   ' Atualiza a PivotTable para ver o efeito da alocação no contexto geral.
  GoTo cleanup     ' Limpa as variáveis na saída do procedimento
```

A parte final do código contém um tratamento de erros que mostra mensagens de erro e usa o método **Undo** para voltar atualizações feitas à célula original na planilha. O tratamento de erro é feito para alguns casos onde a atualização do dado não pode ser realizada.

```
writefailed:
  Select Case Err.Number
    Case -2147168234
      MsgBox "O writeback falhou por conta de um timeout."
    Case -2147168254
      MsgBox "O writeback falhou porque o cubo OLAP não está habilitado para escrita (write-
enabled)."
    Case Else
```

```
      MsgBox "Não foi possível gravar o valor atualizado para o Analysis Server."
  End Select
  Application.Undo
  GoTo cleanup

cleanup:
  Set Cmd = Nothing
  Set Conn = Nothing
  Application.Cursor = xlDefault
End Sub
```

Recurso 4: Drill-through

Uma única rotina principal, descrita abaixo, faz o trabalho de drillthrough.

A primeira parte do procedimento Drillthrough aloca as variáveis a serem utilizadas e realiza algumas verificações simples. Primeiro, uma variável é atribuída ao objeto PivotCell, que representa a célula ativa da tabela dinâmica. Em seguida, é verificado se a tabela dinâmica é conectada a uma fonte de dados OLAP, e se a célula ativa está na área de dados do relatório de tabela dinâmica. Se qualquer uma das verificações não for bem-sucedida, uma mensagem de erro é retornada para o usuário e nenhum ação é realizada. Se o objeto não estiver conectado, é forçada a conexão em seguida através do objeto **PivotCache**.

```
Sub Drillthrough()
  Dim Cat As ADOMD.Catalog
  Dim Conn As ADODB.Connection
  Dim qry As String
  Dim pcell As PivotCell
  Dim pt As PivotTable
  Dim i As Integer
  Dim rs As ADODB.Recordset
  Dim cntEixo As Integer
  Dim comandoMDX As String

  Set pcell = ActiveCell.PivotCell

  'Trata erros: Se a célula não for parte de uma PivotTable vinculada a OLAP, retorna erro
  'Se a célula não estiver na área de dados da PivotTable, retorna erro
  'Garante que o cache da PivotTable está conectado ao datasource
  If Not (pcell.PivotTable.PivotCache.OLAP) Then GoTo errmsg
  If pcell.PivotCellType <> xlPivotCellValue Then GoTo errmsg

  Set pt = pcell.PivotTable
  If Not pt.PivotCache.IsConnected Then
    pt.PivotCache.MakeConnection
  End If
```

Os comandos seguintes criam um objeto ADO Connection e outro ADO Catalog, e em seguida conecta ambos ao objeto **PivotCache**, cache da **PivotTable**.

```
  Set Cat = New ADOMD.Catalog
  Set Conn = New ADODB.Connection
  Set Cat.ActiveConnection = pt.PivotCache.ADOConnection
  Set Conn = pt.PivotCache.ADOConnection
```

A variável **comandoMDX** recebe o comando MDX montado para mostrar os registros de detalhe. A linha a seguir mostra o início desse comando.

```
  comandoMDX = "Drillthrough maxrows 2500 Select "
```

Em seguir, é construída a parte do comando MDX referente aos membros das linhas.

```
  'Faz o loop através dos itens de linha. Apenas os itens mais externos serão adicionados ao
comando MDX.
  For i = 1 To pcell.RowItems.Count - 1
```

```
    If pcell.RowItems(i).Parent.CubeField.Name <> pcell.RowItems(i + 1).Parent.CubeField.Name
Then
      comandoMDX = comandoMDX & "{" & pcell.RowItems(i) & "} on " & cntEixo & ", "
      cntEixo = cntEixo + 1
    End If
  Next i

  'Adiciona o item mais interno se mais de um item foi adicionado ao eixo das linhas
  If pcell.RowItems.Count > 0 Then
    comandoMDX = comandoMDX & "{" & pcell.RowItems(i) & "} on " & cntEixo & ", "
    cntEixo = cntEixo + 1
  End If
```

Agora, é construída a parte do comando MDX referente aos membros das colunas.

```
  'Faz o loop através dos itens de coluna. Apenas os itens mais externos serão adicionados ao
comando MDX.
  For i = 1 To pcell.ColumnItems.Count - 1
    If pcell.ColumnItems(i).Parent.CubeField.Name <> pcell.ColumnItems(i +
1).Parent.CubeField.Name Then
      comandoMDX = comandoMDX & "{" & pcell.RowItems(i) & "} on " & cntEixo & ", "
      cntEixo = cntEixo + 1
    End If
  Next i

  'Adiciona o item mais interno se mais de um item foi adicionado ao eixo das colunas
  If pcell.ColumnItems.Count > 0 Then
    comandoMDX = comandoMDX & "{" & pcell.ColumnItems(i) & "} on " & cntEixo & ", "
    cntEixo = cntEixo + 1
  End If
```

Em seguida, é construída a parte do comando MDX referente aos filtros ou campos de página da tabela dinâmica.

Depois disso, o comando é finalizado com o nome do cubo, o comando é executado e um objeto **QueryTable** é definido para conter o resultado.

```
  'Itera através dos itens de página visíveis (filtros)
  For i = 1 To pt.PageFields.Count
    comandoMDX = comandoMDX & "{" & pt.PageFields(i).CurrentPageName & "} on " & cntEixo & ",
"
    cntEixo = cntEixo + 1
  Next i
  comandoMDX = Left(comandoMDX, Len(comandoMDX) - 2)  'Remove o ", " do final.

  'Adiciona o nome do cubo ao comando MDX.
  comandoMDX = comandoMDX & " From " & "[" & pt.PivotCache.CommandText & "]"

  Set rs = New ADODB.Recordset
  On Error GoTo errmsg

  rs.Source = comandoMDX
  Set rs.ActiveConnection = Conn
  rs.Open
  On Error GoTo 0

  Set ws = Worksheets.Add  'Cria nova planilha

  'Adiciona um QueryTable à planilha. Conecta o QueryTable ao recordset já existente
  'com o resultado do comando MDX.
  With ws.QueryTables.Add(Connection:=rs, Destination:=ws.Range("A1"))
    .Refresh
  End With

  Exit Sub

errmsg:
  MsgBox "Não é possível realizar drillthrough desta seleção."
End Sub
```

VII Conclusão

Este trabalho traz contribuições importantes para as funcionalidades oferecidas pelo cliente de OLAP nativo do Microsoft Excel, através da Tabela Dinâmica. Os objetivos propostos foram efetivamente alcançados, e ainda há uma boa linha de evolução que pode ser seguida para realizar outros melhoramentos na ferramenta.

Foi particularmente gratificante poder trabalhar com ferramentas de BI, ainda mais quando se observa que o mercado de ferramentas gerenciais está sofrendo mudanças importantes com a introdução de ferramentas de BI. Os clientes e usuários diretos das ferramentas BI são presidentes e gerentes de empresas, e as soluções sempre trazem alto valor agregado a empresas, que aumentam sensivelmente a sua capacidade gerencial e capacidade de resposta a mudanças do mercado.

Sugestões de Evoluções Futuras

Recurso 1: "Side Analysis"

No recurso de **Side Analysis**, a implementação atual tem pouca robustez quando a disposição dos dados da tabela dinâmica é modificada, o que pode acontecer em um número de situações:

- Quando um membro de linha ou coluna tem seu nome modificado.
- Quando filtros forem aplicados à tabela dinâmica. Muito embora isso não cause distorção nos valores exibidos na planilha "Side Analysis", caso alguns itens deixem de ser mostrados por conta da escolha do filtro.
- Quando linhas ou colunas forem adicionadas ou removidas da tabela dinâmica por conseqüência de atualização dos dados.

Sugestões de evoluções futuras ao "Side Analysis" seriam para solucionar os problemas expostos acima.

Recurso 2: Ordenação de dados e "n primeiros"

Alguns usuários solicitam um recurso a mais nesta implementação. Para realizar determinadas análises, muitas vezes é necessário exibir também a consolidação do valores referentes aos outros membros que não estão na lista dos n primeiros. Por exemplo, se uma empresa tem 100 clientes e solicita a lista dos 10 clientes que têm maior lucratividade, o Sistema poderia mostrar além da lista dos 10 maiores clientes um 11º. registro correspondendo à soma dos outros 90.

VIII Bibliografia

[1] Microsoft Excel Online Documentation.

[2] Microsoft Analysis Services Books Online.

[3] Microsoft SQL Server 2000 Books Online.

[4] Extending Excel OLAP Functionality.
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnexcl2k2/html/odc_xlextendolap.asp

[5] Excel 2002 Add-in: OLAP CubeCellValue
http://office.microsoft.com/Downloads/2002/CubeCellValue.aspx

[6] Microsoft Business Intelligence
http://www.microsoft.com/business/bi

[7] Designing and Building Large Data Warehouses with SQL Server 2000
http://www.microsoft.com/singapore/downloads/Designing_and_Building_Large_Data_Warehouses_with_SQLServer.ppt; slides 15-17.

[8] OLAP – Webopedia.com
http://www.webopedia.com/TERM/O/OLAP.html

[9] DM Review: Business Intelligence: What is Business Intelligence?
http://www.dmreview.com/master.cfm?NavID=68&EdID=1924

[10] Visual OLAP for Microsoft Excel
http://www.visualolap.com/vo%20white%20paper.pdf; page 8.

[11] BI FAQ
http://www.iolap.com/bi_faq.htm

[12] What is MDX?
http://www.progsys.com/ARCHIVE/vbpj/0700/sidebar2.htm