Instituto de Ciências Matemáticas e de Computação

ISSN - 0103-2569

DBGen

Manual da Ferramenta

Renato Bueno
Mônica Ribeiro Porto Ferreira
Caetano Traina Junior

Nº 246

RELATÓRIOS TÉCNICOS DO ICMC

São Carlos
dezembro/2004

Sumário:

Lista de Figuras:

1. Introdução

No desenvolvimento de aplicações e em pesquisas na área de Banco de Dados, muitas vezes é necessário saber como se comportam os dados de uma determinada base de dados, para que se possam avaliar melhor ou até mesmo validar os resultados da aplicação de alguma técnica: o conhecimento prévio da distribuição dos dados na base é importante na concepção e nos testes de algoritmos em trabalhos de pesquisa.

Portanto, são necessários bancos de dados sintéticos, que obedeçam a distribuições conhecidas, e que possam ter características específicas escolhidas pelo desenvolvedor, de acordo com o propósito de sua utilização.

Para suprir essa necessidade de dados previamente conhecidos, foi desenvolvido um aplicativo gerador de dados sintéticos, chamado *Database Generator* - DBGen. Os dados gerados por esse aplicativo podem ser gravados em bases de dados, dando origem aos bancos de dados sintéticos.

Com o aplicativo podem ser geradas coleções de dados com diferentes distribuições, como por exemplo, em forma de planos, linhas, circunferências ou fractais. Os dados podem ter qualquer dimensionalidade, podendo gerar bancos de dados complexos.

O aplicativo foi desenvolvido junto ao Laboratório de Base de Dados e Imagens (GBdI) do ICMC, utilizando a ferramenta *Borland C++ Builder*[1]. Para o desenvolvimento do DBGen, foi utilizado como base o aplicativo *Measure Distance Exponent* – MDE (FALOUTSOS, 2000), desenvolvido pelo Prof. Dr. Caetano Traina Júnior e pela Profa. Dra. Agma Juci Machado Traina.

Nas seções seguintes, veremos uma descrição geral do aplicativo DBGen, assim como os comandos disponíveis, divididos por tipos. Primeiro serão analisados os comandos para síntese de dados, depois os comandos para configuração do aplicativo, e então os comandos para manipulação e visualização dos dados. A geração de fractais mediante um *L-System* (FLAKE, 1999) é discutida separadamente, onde são apresentados os comandos para configuração do *L-System* e são mostrados exemplos de fractais que podem ser gerados com o DBGen.

---

[1] THEOBALD, T. Borland C++ Builder 6 Overview: Rapid C++ e-business development with Web Services, Jan. 2002. Disponível em: <http://www.borland.com/products/white_papers/pdf/borland_cbuilder_overview.pdf>. Acesso em: 06 out. 2004.

2. DBGen: Sintetizador de bases de dados

O aplicativo DBGen tem a finalidade de gerar bases de dados sintéticas, obedecendo a algum modelo definido para distribuição de tais dados. Os dados gerados, que chamaremos de objetos, constituem coleções de objetos. A cada comando de síntese de objetos, uma nova coleção é criada[2] e passa a ser a coleção primária de objetos. Se existia previamente uma coleção primária de objetos, tal coleção passa a ser chamada de coleção secundária de objetos. O aplicativo, portanto, permite que duas coleções de objetos estejam simultaneamente armazenadas em memória. São permitidas também operações que atuam nas duas coleções, como a troca e a concatenação, para criação de coleções de dados mais complexas, acumulando os resultados de várias sínteses.

O aplicativo DBGen permite que os dados das coleções em memória sejam gravados em arquivos de texto e em bancos de dados. Também é possível visualizar graficamente as coleções geradas, com o auxílio da ferramenta *GNUPLOT*[3].

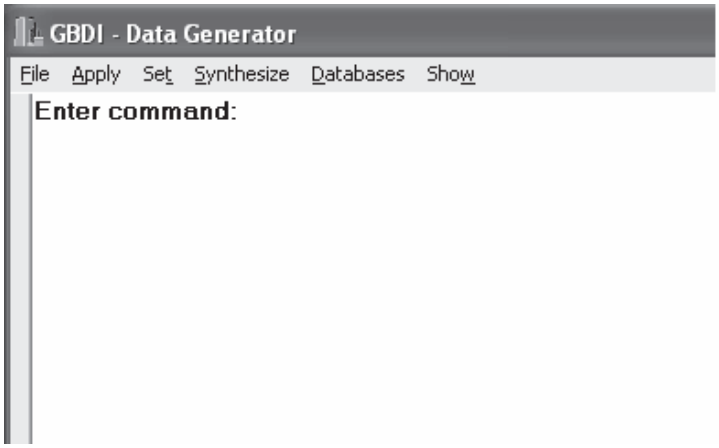

Figura 1: Tela Inicial do DBGen

O usuário interage com o aplicativo por meio de linhas de comandos. Tais linhas podem ser escritas completamente com todos os parâmetros requeridos ou parcialmente, quando o usuário é questionado sobre o valor de cada um dos parâmetros ainda não informados. A interface de linhas de comando facilita a utilização de *scripts,* isto é, a utilização de arquivos de texto salvos em formato *csv* (*comma separated values*) com as seqüências de comandos que podem ser carregados pelo aplicativo.

[2] A não ser nos comandos de modificações nos eixos, descritos posteriormente.
[3] GNUPLOT Homepage. Disponível em: <http://www.gnuplot.info/>. Acesso em 01 set. 2004.

Além da interface por meio de linhas de comandos, existe também a interface mediante menus, que facilita a interação de usuários inexperientes ou desconhecedores dos comandos do DBGen com a ferramenta. Todos os comandos disponíveis no modo linha-de-comando também estão disponíveis nos *menus* do aplicativo. Qualquer comando acionado pelo *menu* tem sua linha-de-comando equivalente escrita no *prompt* do aplicativo, familiarizando o usuário com os comandos. Na Figura 3 temos um exemplo de um comando acionado pelo *menu*, um exemplo de linha de código escrita parcialmente e um exemplo de linha de código escrita completamente no *prompt* de comandos.

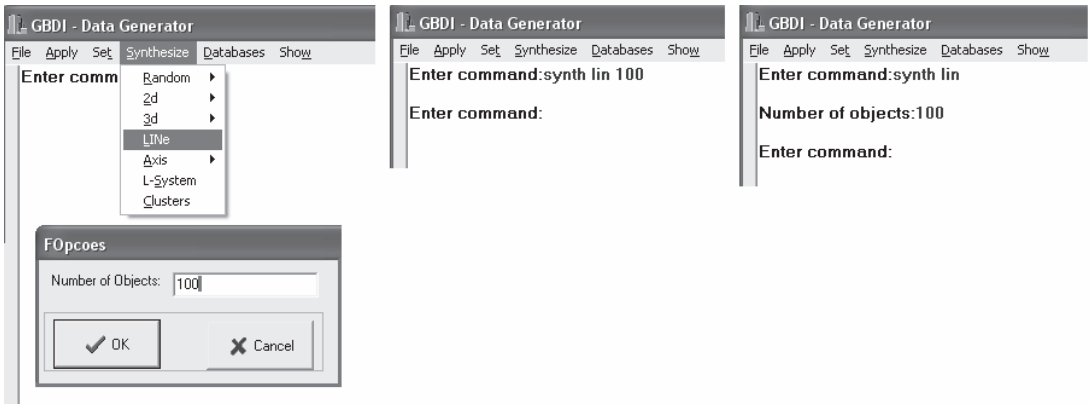

Figura 2: Comandos por meio de Menu e Linha de Comandos Total e Parcial

Quanto à síntese de dados, estão disponíveis várias opções para as distribuições das coleções, em especial, a utilização de um *L-system* para geração de fractais.

No diretório de execução do aplicativo é criado um arquivo de texto que registra todas as operações executadas pelo DBGen.

Apresentamos a seguir, descrições mais detalhadas do funcionamento do aplicativo, dos comandos disponíveis e também seus respectivos subcomandos e parâmetros.

2.1 Comandos para síntese de dados: SYNTHESIZE

O comando *Synthesize* é o comando responsável pela síntese dos objetos[4], ou seja, é por meio da utilização desse comando que são criados conjuntos de dados que obedecem alguma distribuição. O comando *Synthesize* possui vários subcomandos, mediante os quais são definidas as características dos conjuntos a serem criados. Como a maioria dos comandos do DBGen, o comando *Synthesize* possui uma forma abreviada, que é *SYNTH*.

[4] Chamaremos de Objetos cada um dos dados (tupla) de *D* dimensões gerados pelo DBGEN, onde D pode ser definido pelo comando SET DIMENSION.

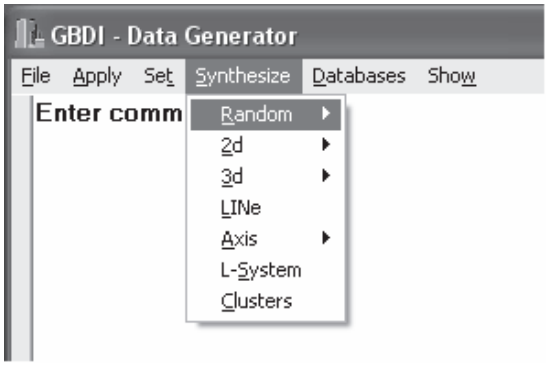

Figura 3: Menu do Comando Synthesize e Seus Subcomandos

Abaixo, listamos os principais subcomandos do comando *Synthesize*, onde todos têm sua forma abreviada mostrada em letras maiúsculas:

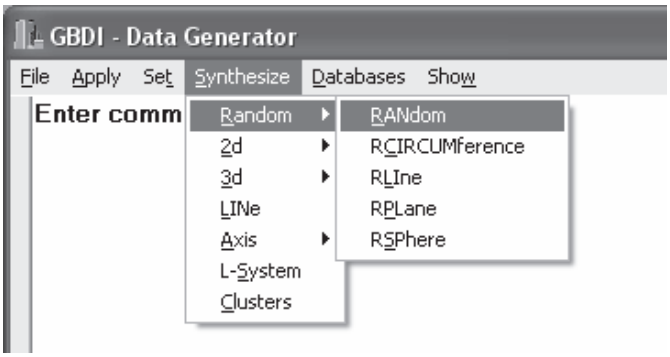

Figura 4: Submenu Random

- ***Random (RAN)***: o subcomando *random* gera números aleatórios distribuídos entre 0 e 1 em cada um dos eixos relativos a todas as dimensões dos novos objetos sintetizados. Portanto, o único parâmetro que é necessário é o número de objetos a serem gerados. Este subcomando pode ser encontrado no submenu *Random*.
- ***Rcircumference (RCIRCUM)***: o subcomando *rcircumference* gera objetos distribuídos aleatoriamente sobre uma circunferência imaginária com diâmetro 1 e centro na posição onde todos os eixos têm valores nulos e apenas os eixos x e y têm valor 0.5. É uma variação de *circumference*. Deve ser informado apenas o número de objetos a serem gerados. Este subcomando pode ser encontrado no submenu *Random*.
- ***Rline (RLI)***: o subcomando *rline* gera objetos distribuídos aleatoriamente sobre uma linha imaginária formada por objetos com os mesmos valores para todas as dimensões. Trata-se de uma variação do comando *line*. Na prática, para cada objeto é gerado um número aleatório entre 0 e 1, que preenche todos os eixos do objeto. Deve ser passado como

parâmetro apenas o número de pontos a serem gerados. Este subcomando pode ser encontrado no submenu *Random*.

- ***Rplane (RPL)***: o subcomando *rplane* gera objetos distribuídos aleatoriamente sobre o plano x, y. Trata-se de uma variação do comando *plane*. O único parâmetro que deve ser informado é o número de objetos a serem gerados. Este subcomando pode ser encontrado no submenu *Random*.
- ***Rsphere (RSP)***: o subcomando *rsphere* gera objetos distribuídos aleatoriamente por toda a superfície da espera (espera "oca") imaginária com raio 1. Trata-se de uma variação do comando *Sphere*. Deve ser informado o numero de objetos por circunferência a serem gerados. Esse comando só funciona se a opção *set dimension* estiver configurada com dimensão igual a 3. Este subcomando pode ser encontrado no submenu *Random.*

Nos comandos do submenu *Random*, deve-se configurar o tipo de distribuição desejada no comando *set disttype* (SET DIS). Caso não seja configurado o tipo da distribuição, o comando gerará os objetos com a distribuição padrão (distribuição uniforme).

- ***Line (LIN)***: o subcomando *line* gera uma linha reta, com um número de objetos que deve ser informado. Cada objeto da linha recebe o mesmo valor numérico (entre 0 e 1) para todas as suas dimensões (eixos). Dessa forma, uma linha reta pode ser vista com a seleção de quaisquer combinações de dimensões. Os objetos são distribuídos uniformemente pela linha sintetizada. Este subcomando pode ser encontrado no submenu *Line*.

Nas Figuras 5 podemos ver a diferenciação entre os comandos line e rline, ambos gerando 20 novos objetos com duas dimensões.

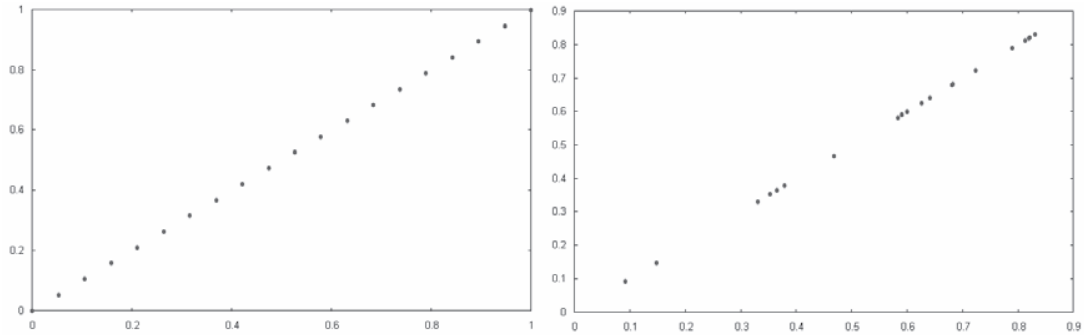

Figura 5: Resultado do comando: SYNTH LIN 20 e SYNTH RLI 20

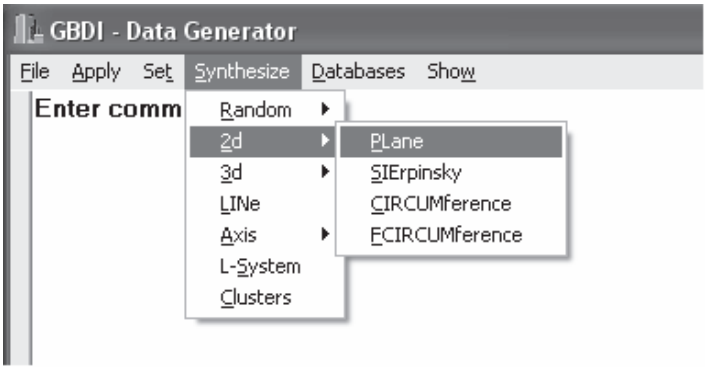

Figura 6: Submenu 2d

- ***Plane (PL)***: o subcomando *plane* gera um plano 2D usando duas dimensões dos objetos, ou seja, apenas dois eixos, (x, y)[5]. Com isso, para objetos com mais que duas dimensões, apenas os eixos relativos às duas dimensões primárias receberão valores diferentes de zero. O subcomando exige que seja informado o número de objetos que devem ser gerados em cada dimensão do plano. Tais objetos serão distribuídos uniformemente pelo plano sintetizado. Este subcomando pode ser encontrado no submenu *2d*.

- ***Sierpinsky (SIE)***: o subcomando *sierpinsky* gera triângulos de Sierpinsky e fractais similares, e armazena os vértices do fractal gerado como novos objetos. O triângulo de Sierpinsky foi descoberto pelo matemático Waclav Sierpinsky (1882-1969). O fractal é obtido através de um processo iterativo de divisão de um triângulo eqüilátero em quatro triângulos semelhantes, descartando-se o triângulo invertido. Para o próximo nível, o mesmo processo é aplicado em cada um dos três novos triângulos, e assim sucessivamente. Qualquer pequena parte da figura é semelhante à figura completa, apenas em escala diferente. Portanto, o fractal obtido é estritamente auto-semelhante, como podemos observar nos triângulos presentes na Figura 7.

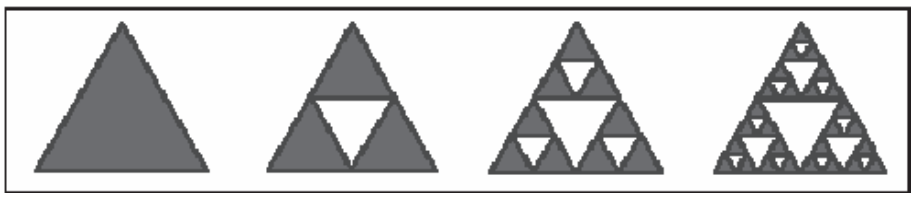

Figura 7: Triângulos de Sierpinsky (3 iterações)

Com o subcomando *sierpinsky*, também são admitidas figuras com mais vértices, como o quadrado presente na Figura 8.

[5] Os eixos x,y são chamados aqui de eixos primários, e podem ser definidos com o comando SET AXIS. O padrão é que sejam os dois primeiros eixos de um objeto.

Figura 8: Carpete de Sierpinsky (2 iterações)

São requisitados, portanto, o número de iterações desejado e o número de vértices da figura. Os fractais são gerados no plano x, y, ou seja, referente aos dois eixos primários. Este subcomando pode ser encontrado no submenu *2d*.

- ***Circumference (CIRCUM)***: o subcomando *circumference* gera uma circunferência sobre o plano x, y. Deve ser informado o número de objetos que devem ser gerados. Os objetos são distribuídos uniformemente pela circunferência. Para objetos com mais que duas dimensões, as demais dimensões (que não são relativas aos eixos x, y) receberão valor zero. A circunferência gerada tem diâmetro igual a 1 e tem seu centro localizado na posição onde todos os eixos têm valores nulos e apenas os eixos x e y têm valores 0.5. Este subcomando pode ser encontrado no submenu *2d*.

- ***Fcircumference (FCIRCUM)***: o subcomando *fcircumference* gera um círculo de diâmetro igual a 1 sobre o plano x, y, sendo os objetos gerados distribuídos aleatoriamente por todo interior do circulo e não somente sobre o seu contorno (circunferência). Deve ser informado o número de objetos que devem ser gerados. Na Figura 9, podemos ver a diferenciação entre os comandos *circumference* e *fcircumference*, ambos gerando 500 novos objetos com duas dimensões. Este subcomando pode ser encontrado no submenu *2d*.

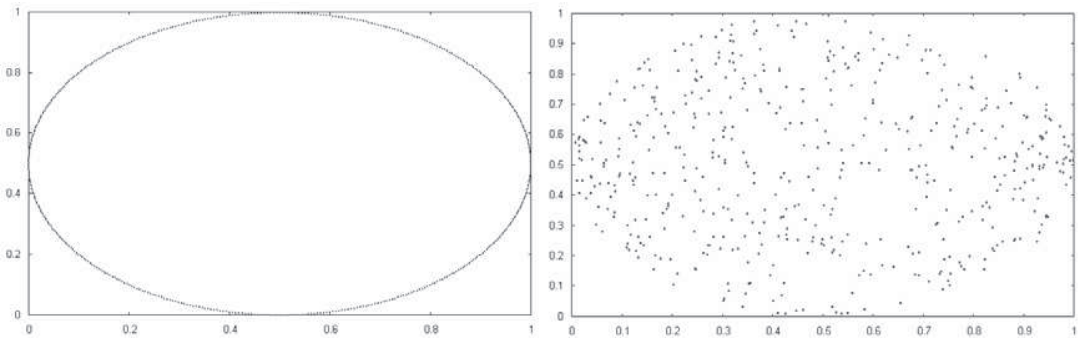

Figura 9: Resultado dos Comandos CIRCUM 500 e FCIRCUM 500

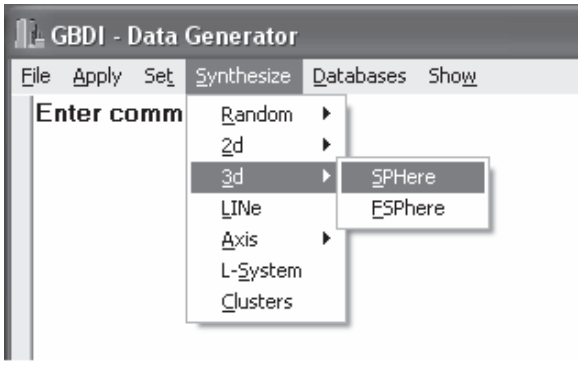

Figura 10: Submenu 3d

- ***Sphere (SPH)***: o subcomando *sphere* gera objetos aleatórios distribuídos uniformemente que fazem parte de uma espera imaginária com diâmetro 1, porém esses objetos estão distribuídos por toda a superfície da espera ("espera oca"). Deve ser informado o número de objetos a ser criado por circunferência. Esse subcomando só funciona se a opção *set dimension* estiver configurada com dimensão igual a 3. Este subcomando pode ser encontrado no submenu *3d*.

- ***Fsphere (FSP)***: o subcomando *fsphere* gera objetos distribuídos aleatoriamente pela espera imaginária com diâmetro 1, porém esses objetos aleatórios estão distribuídos por todo o interior da espera e não somente em sua superfície ("espera maciça"). Deve ser informado o número de objetos a ser criado pela ferramenta. Este comando só funciona se a opção *set dimension* estiver configurada com dimensão igual a 3. Na Figura 11, podemos ver a diferenciação entre os comandos *sphere* e *fsphere*, ambos gerando 1000 novos objetos com três dimensões. Este subcomando pode ser encontrado no submenu *3d*.

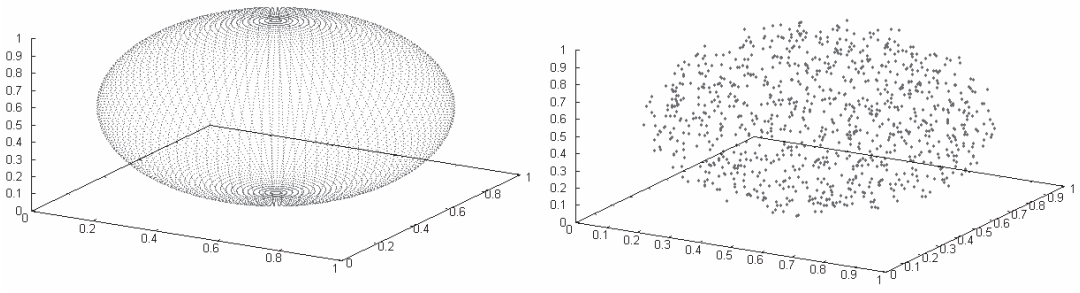

Figura 11: Resultado dos Comandos SPH 1000 e FSP 1000

- ***L-System (LSYS)***: O subcomando *lsystem* permite a criação de coleções de objetos baseados em fractais criados utilizando *L-System*. Um *L-System*, que foi inicialmente criado para descrever formas da natureza, é uma técnica para descrição de objetos complexos pela substituição de partes do objeto inicial por outras, utilizando algumas regras para reescrita,

chamadas de produções. A aplicação de um *L-System* torna possível a construção de fractais. O subcomando *lsystem* gera uma nova coleção de dados baseada nas configurações previamente selecionadas com o comando *lset*. Mais detalhes sobre o subcomando *lsystem*, sua configuração e exemplos podem ser encontrados na seção 2.4. Este subcomando pode ser encontrado no submenu *L-System*.

- ***Cluster (CL):*** O subcomando *cluster* cria uma coleção de objetos que são semelhantes uns aos outros (nuvem de agrupamento). Deve ser informado o valor inicial para o gerador de números aleatórios (*seed*); o número de dados a ser gerado (*data*); o número de conjuntos requeridos (*clust*) e a variação do cluster (*sigma*) - depende da distribuição escolhida. Se sigma for menor ou igual a 0, então a distribuição é uniforme. O tipo de distribuição que será usado para a criação da nuvem de agrupamento deve ser escolhido no menu *Set*, com a execução do comando *Disttype* e a dimensão deve ser escolhida no menu *Set* com a execução do comando *Dimension.* Na Figura 12, podemos ver o resultado da execução do comando *cluster* gerando 5 nuvens de cluster a partir 10000 objetos em três dimensões Este subcomando pode ser encontrado no submenu *Cluster.*

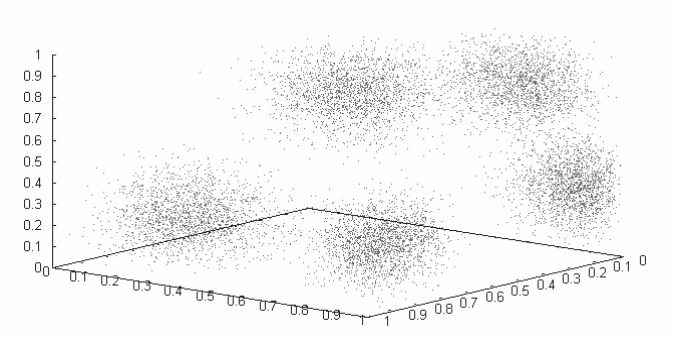

Figura 12: Resultado do Comando: SYNTH CL 15 10000 5 0.1

Os subcomandos anteriores geravam novos conjuntos de objetos, independentemente dos conjuntos gerados anteriormente. Os subcomandos que veremos a seguir modificam o último conjunto sintetizado, gerando então um novo conjunto. Portanto, dependem da existência de um conjunto de objetos já armazenados em memória. Tais comandos, que geram alterações relativas aos eixos dos objetos, são descritos abaixo:

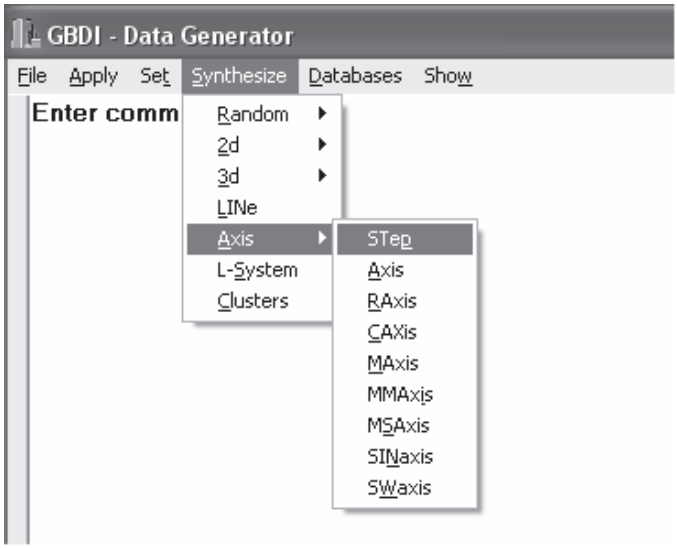

Figura 13: Submenu Axis

- ***Step (ST)*:** o subcomando *step* substitui os valores presentes no eixo escolhido por uma seqüência crescente de números que começa em 0 e cresce, a cada objeto, com a soma de um *passo* definido. O subcomando requer que seja informado um eixo e um valor para o passo. Portanto, o primeiro objeto recebe 0 no eixo escolhido, o segundo recebe o valor do *passo*, o terceiro o valor de 2 *passos*, o quarto três *passos* e assim por diante. Este subcomando pode ser encontrado no submenu *Axis.*

- ***Axis (AX)***: o subcomando *axis* modifica o valor de um eixo[6], substituindo-o por um valor específico em todos os objetos. Por isso, deve ser informado o eixo que será alvo da modificação e o valor que deverá substituir os valores atuais. A identificação dos eixos é feita pela forma numérica, sendo o primeiro eixo chamado de 1, o segundo chamado de 2, e assim por diante. Deve também ser fornecido o valor que substituirá os valores anteriores no eixo escolhido em todos os objetos da coleção primária. Este subcomando pode ser encontrado no submenu *Axis*.

- ***Raxis (RAX)*:** o subcomando *raxis* também modifica o valor de um eixo escolhido em todos os objetos existentes, como o comando *axis*. Mas aqui, ao invés de fornecer um valor para o eixo, um valor é escolhido aleatoriamente de uma distribuição uniforme entre 0 e 1 para cada um dos objetos existentes. O subcomando raxis causa um "espalhamento" dos objetos. Podemos ver um exemplo de uma seqüência de comandos utilizando *axis* e *raxis* na Figura 14. Este subcomando pode ser encontrado no submenu *Axis.*

[6] Um eixo equivale a uma dimensão, ou a uma coluna de uma tupla.

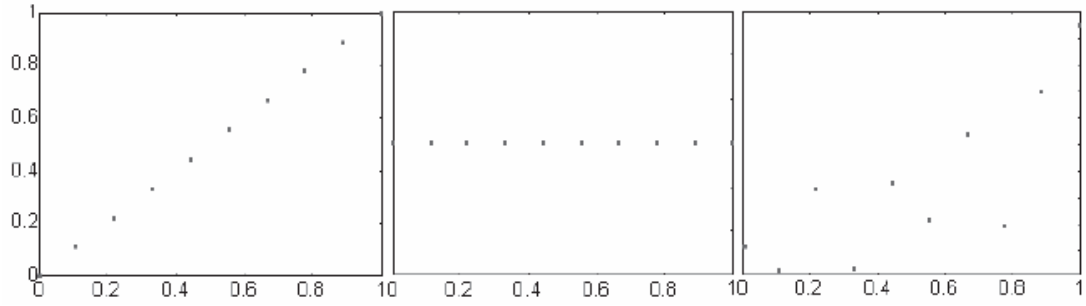

Figura 14: Seqüência de Comandos: SYNTH LIN 10, SYNTH AX 2 0.5, SYNTH RAX 2

- ***Caxis (CAX)*:** o subcomando *caxis* copia o valor de um eixo para outro, em cada um dos objetos existentes. É necessário indicar qual eixo será copiado e qual eixo receberá a cópia. Para cada um dos objetos, o valor presente no eixo indicado como destino receberá o valor presente no eixo indicado como origem. O subcomando *caxis* pode ser usado, por exemplo, para transformar objetos em duas dimensões em linhas retas, pois torna os valores presentes nos dois eixos iguais. Este subcomando pode ser encontrado no submenu *Axis.*

- ***Maxis (MAX)*:** o subcomando *maxis* multiplica o valor presente em um eixo dos objetos por um valor especificado. Portanto, deve ser especificado o eixo a ser multiplicado, assim como o valor pelo qual se pretende multiplicar. Este subcomando pode ser encontrado no submenu *Axis.*

- ***Mmaxis (MMAX)*:** o subcomando *mmaxis* é um pouco mais complexo, mas possibilita várias operações com os eixos. Devem ser fornecidos: um eixo destino (ED), um eixo de origem (EO), um eixo independente (EI) e um valor (VAL). O valor presente no eixo destino (ED) é substituído pela multiplicação de todos os outros parâmetros, ou seja, para cada objeto, ED = VAL * EO * EI. Com esse subcomando, podemos realizar várias operações envolvendo os eixos, como potenciação ou multiplicação entre eixos diferentes. Podemos ver um exemplo de uma seqüência de comandos utilizando *mmaxis* na Figura 15. Este subcomando pode ser encontrado no submenu *Axis.*

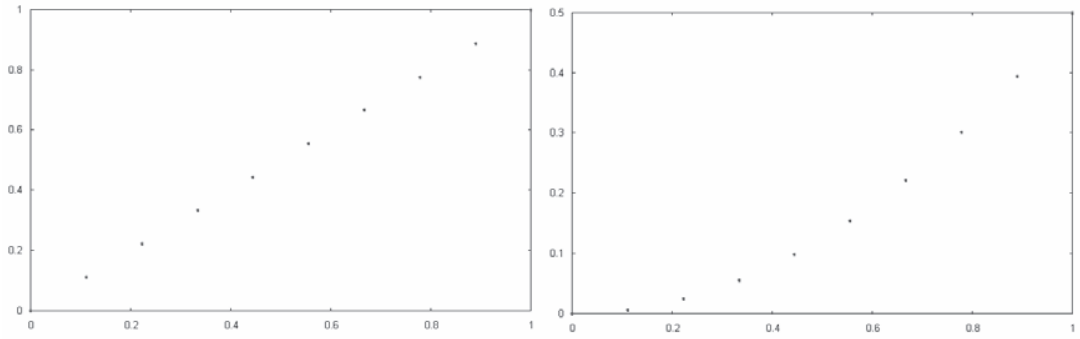

Figura 15: Seqüência de Comandos: SYNTH LIN 10, SYNTH MMAXIS 2 1 1 0.5

- ***Msaxis (MSAX)*:** o subcomando *msaxis* também requisita um eixo destino (ED), um eixo de origem (EO), um eixo independente (EI) e um valor (VAL). O subcomando *msaxis* substitui o valor presente no eixo destino (ED) de cada objeto, pela multiplicação do valor fornecido (VAL) pelo valor presente no eixo de origem (EO), somado ao valor do eixo independente (EI), isto é, ED = VAL * EO + EI. É uma variação do subcomando *mmaxis*, e pode também dar origem a outras operações com os eixos, que não são possíveis com os comandos anteriores. Podemos ver um exemplo de uma seqüência de comandos utilizando *msaxis* na Figura 16. Este subcomando pode ser encontrado no submenu *Axis.*

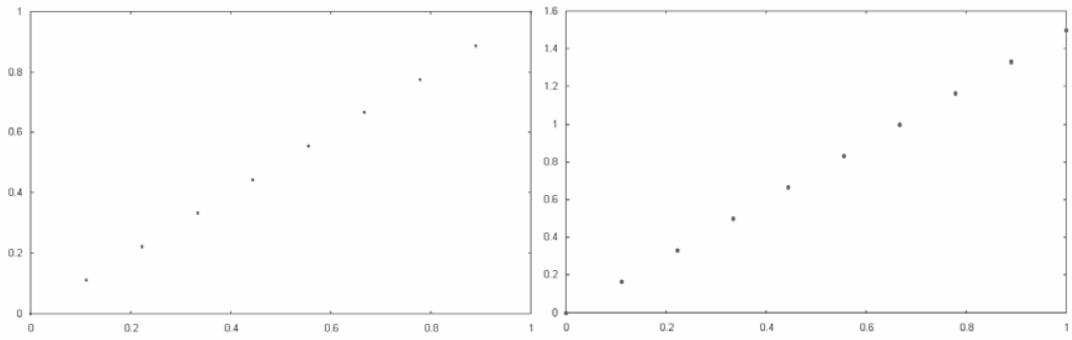

Figura 16: Seqüência de Comandos: SYNTH LIN 10, SYNTH MSAXIS 2 1 2 0.5

- ***Sinaxis (SIN)*:** o subcomando *sinaxis* substitui os valores presentes no eixo escolhido pelos senos dos ângulos do intervalo escolhido. É necessário informar o eixo escolhido, o valor em graus do ângulo inicial e o valor do ângulo final. O intervalo entre os ângulos inicial e final é dividido igualmente pelo número de objetos existentes, com cada objeto recebendo um ângulo pertencente ao intervalo. Então, o valor do eixo de cada objeto é substituído pelo seno do ângulo relativo a ele. Na Figura 17 podemos ver a aplicação do subcomando *sinaxis* sobre um conjunto de cinqüenta objetos, escolhendo-se o eixo 2 (eixo y) com ângulo inicial de 0^o e ângulo final de 360^o. Este subcomando pode ser encontrado no submenu *Axis.*

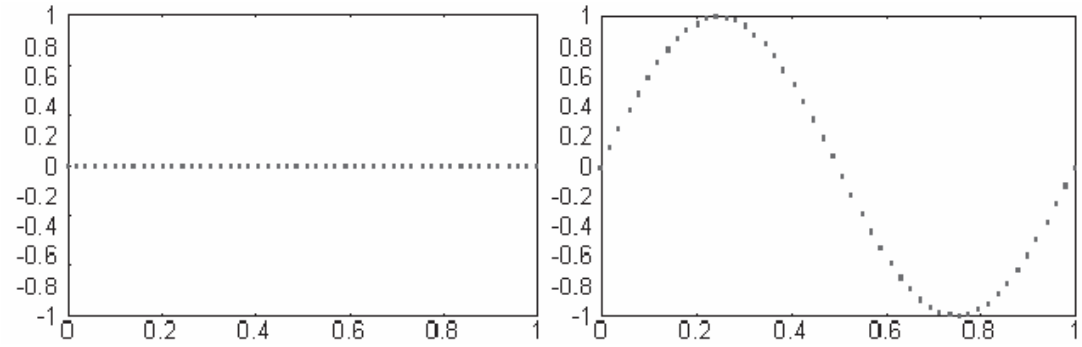

Figura 17: Seqüências de Comandos: SYNTH LIN 50, SYNTH AX 2 0.5, SYNTH ST 2 0, SYNTH SIN 2 0 360

- ***Swaxis (SW)*:** o subcomando *swpaxis* realiza a troca dos valores entre dois eixos. Precisa-se informar o eixo de origem e o eixo de destino. Então, para cada um dos objetos, o eixo de destino recebe o valor do eixo de origem e o eixo de origem recebe o valor do eixo de destino. Este subcomando pode ser encontrado no submenu *Axis*.

2.2 Comando para configuração: SET

O comando *SET* é responsável pela modificação das configurações do aplicativo DBGen. Com o comando *set* é possível: alterar a dimensão dos objetos a ser sintetizado; alterar o nome da coleção primária de objetos em memória; definir uma taxa de amostragem a ser aplicada, assim como configurar transformações que podem ser aplicadas aos objetos, como rotações, translações, escalas.

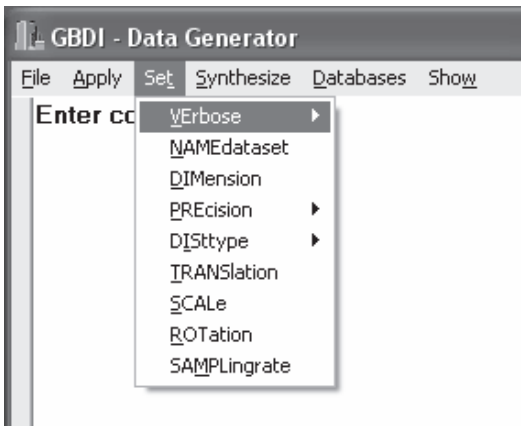

Figura 18: Menu do Comando Set e Seus Subcomandos

Como o comando *Synthesize*, o comando *SET* possui vários subcomandos, que serão listados abaixo, todos com sua forma abreviada em letras maiúsculas:

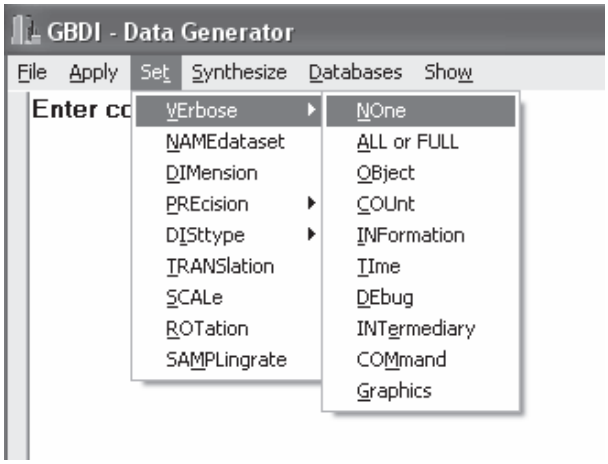

Figura 19: Subcomandos do Set Verbose (SET VE)

- ***Verbose (VE):*** o subcomando *verbose* define o nível de detalhamento das respostas do aplicativo. Possui várias opções: *None (NO ou NONE)*; *All or Full (ALL ou FULL); Object (OB); Count (COU); Information (INF); Time (TI); Debug (DE); Intermediary (INT); Command (COM)* e *Graphics (GRA)* sendo esta a mais importante das opções, pois configura o aplicativo para exibir graficamente as coleções quando o comando *dump (DUMP)* é usado.

- ***Namedataset (NAME)***: o subcomando *namedataset* altera o nome dado à coleção primária de objetos em memória. O nome da coleção pode ser visto com o comando *show datasets (SH DA)*, mas também está presente nos gráficos, nos arquivos de textos gerados e no arquivo de informações gerado pelo DBGen. Cada coleção, ao ser sintetizada com s*ynthesize (SYNTH),* recebe um nome padrão e auto-explicativo. Por exemplo, uma linha com trezentos pontos recebe o nome de *Line300*. O subcomando *namedataset* necessita de apenas um parâmetro, o novo nome da coleção primária. Na Figura 20, segue um exemplo do comando *set namedataset*.

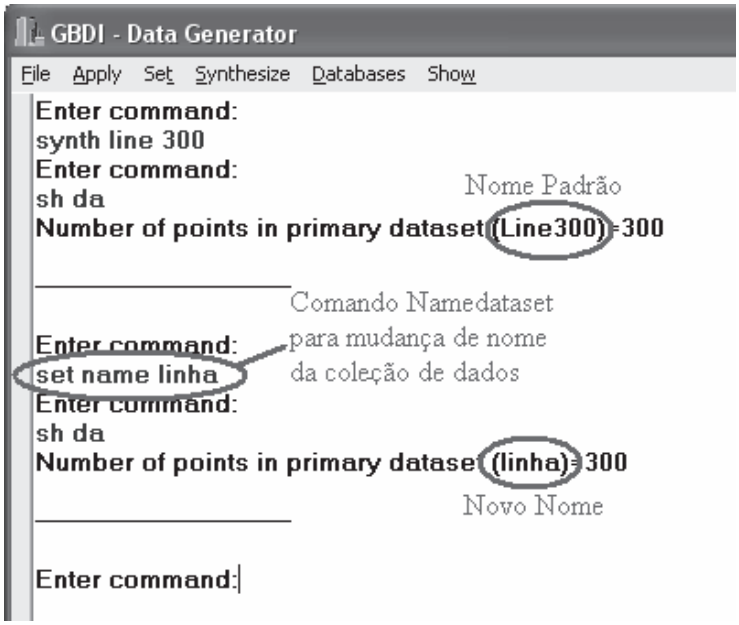

Figura 20: Exemplo de Utilização do Comando SET NAME

- ***Dimension (DIM)***: o subcomando *dimension* altera a dimensão dos objetos a serem sintetizados. O único parâmetro que deve ser informado é um número inteiro, referente à nova dimensionalidade dos objetos. Quando a dimensão é alterada com o comando *SET DIM*, as coleções presentes na memória são perdidas, ou seja, são zeradas. São zeradas também todas as transformações definidas anteriormente com o próprio comando *set*, assim como a taxa de amostragem. O subcomando *dimension* é importante para o sintetizador de dados, pois permite que sejam criados objetos com muitas dimensões, ou seja, tuplas com muitas colunas, que são necessárias na geração de bases de dados.

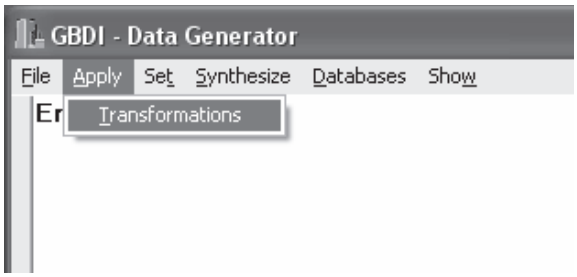

Figura 21: Comando Apply Transformations (AP TR)

- ***Precision (PRE):*** o subcomando *precision* configura os dados para serem gerados como variáveis aleatórias contínuas (*continuous*) ou discretas *(discrete)*. Este subcomando é usando junto com o subcomando *disttype* para configurar o tipo de distribuição de dados

(*uniform random*, *normal random* ou *chi-squared* random) que serão gerados aleatoriamente.

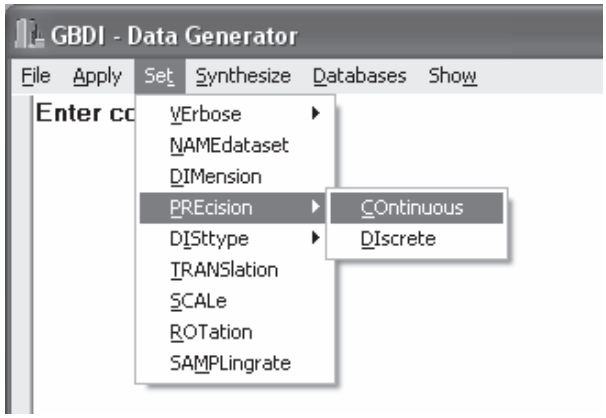

Figura 22: Subcomandos do Set Precision (SET PRE)

- ***Disttype (DIS):*** o subcomando *disttype* configura o tipo de distribuição que será adotada para a geração da coleção de dados. Tem três subopções: *Uniform Random*, que gera os números aleatórios entre 0 e 1; *Gauss (Normal) Random,* que gera números segundo a distribuição gaussiana (ou normal) e *Chi-Squared Random*, que gera números segundo a distribuição qui-quadrado. Deve ser usado junto com o comando *set precision* para que o tipo de distribuição (contínua ou discreta) seja configurado. Porém, as três subopções disponíveis na ferramenta são distribuições de variáveis aleatórias contínuas.

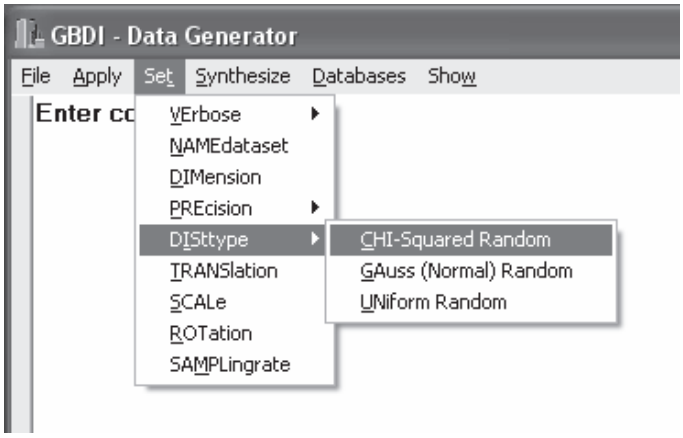

Figura 23: Subcomandos do Set Disttype (SET DIS)

- ***Translation (TRANS)*****:** o subcomando *translation* configura a transformação geométrica de translação em um dos eixos. É necessário informar o eixo onde se deseja fazer a translação e o valor *delta* que os objetos devem ser transladados. O comando *set translation*, assim como os dois subseqüentes *set scale* e *set rotation* apenas configuram as transformações geométricas a serem efetuadas. Para que estas transformações

sejam realmente efetuadas, deve-se executar o comando *Apply Transformations (AP TR)*.

- ***Scale (SCAL)*:** o subcomando *scale* configura a transformação geométrica de escala, a ser aplicada em um eixo escolhido, sobre a coleção primária. Os parâmetros necessários são: o eixo onde será realizada a transformação e o fator de escala. Igualmente ao subcomando *translation*, necessita da execução posterior do comando *Apply Transformations (AP TR)* para ser realmente executado.

- ***Rotation (ROT)*:** o subcomando *rotation* configura a transformação geométrica de rotação de um ângulo determinado sobre um eixo. Portanto, devem ser informados o eixo e o ângulo a ser rotacionado. Também necessita da execução posterior do comando *Apply Transformations (AP TR)*.

- ***Samplingrate (SAMPL)*:** o subcomando *samplingrate* permite que seja selecionada uma amostra da coleção primária. O único parâmetro a ser informado é a taxa de amostragem, que deve estar entre 0 e 1, e representa a porcentagem dos objetos atuais que devem ser mantidos, ou seja, a que parte da coleção primária atual deve corresponder a amostra. Esse comando é útil quando são utilizadas grandes coleções de dados. Também pode ser útil quando a coleção é muito homogênea e não se deseja tal homogeneidade. Apesar de não ser uma transformação geométrica, necessita também da execução posterior do comando *Apply Transformations (AP TR)*.

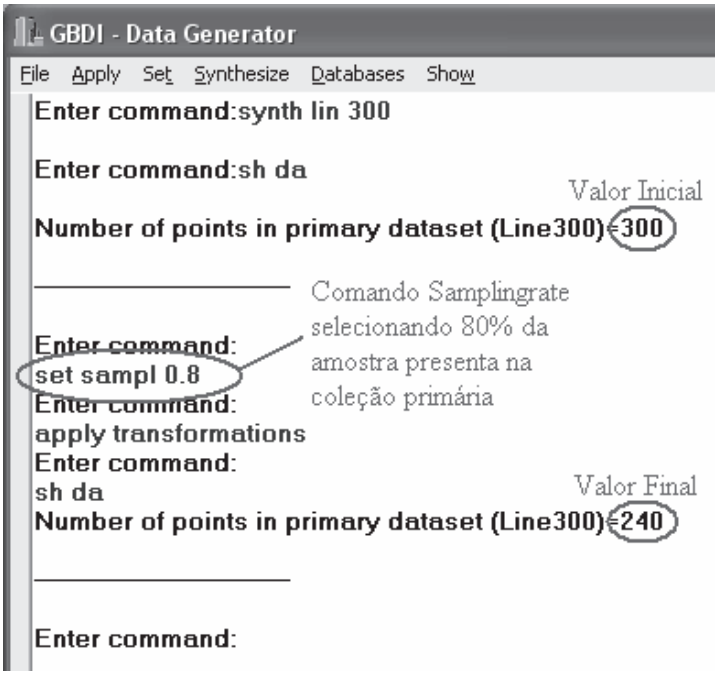

Figura 24: Exemplo do Comando SET SAMPL 0.8

2.3 Comandos para manipulação e visualização dos dados

Aqui trataremos de alguns outros comandos disponíveis no aplicativo DBGen, como comandos para manipulação das coleções, comandos para visualização das coleções, e informações sobre tais, e comandos para leitura e escrita em arquivos e banco de dados.

Como já foi dito antes, o aplicativo armazena duas coleções em memória, a coleção primária e a secundária, sendo que a coleção primária armazena os objetos gerados no último comando de síntese, ou lidos de uma base de dados ou de um arquivo texto, e a secundária armazena os objetos que pertenciam à coleção primária anteriormente. Os três próximos comandos permitem, respectivamente, a troca de conteúdo entre as duas coleções, a duplicação da coleção primária e a concatenação das duas coleções.

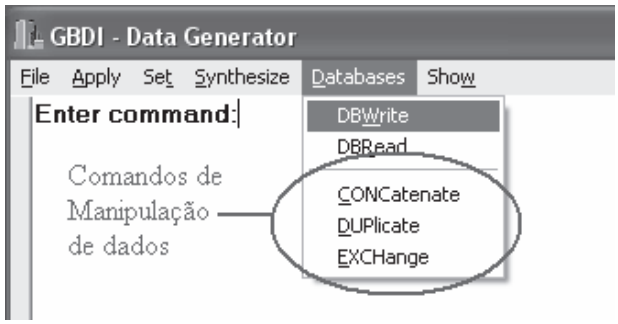

Figura 25: Comandos de Manipulação de Dados

- ***Concatenate (CONC)***: o comando *concatenate* permite a concatenação da coleção primária com a coleção secundária. Portanto, é necessário que as duas coleções estejam definidas. Tal comando é muito útil para síntese de coleções de dados mais complexas, pois permite que várias sínteses sejam acumuladas numa mesma coleção.

- ***Duplicate (DUP)***: o comando *duplicate* duplica a coleção primária na coleção secundária. É necessário que a coleção primária esteja definida. A coleção secundária se tornará idêntica à coleção primária, sendo seu conteúdo anterior perdido.

- ***Exchange (EXC)***: o comando *exchange* realiza a troca entre o conteúdo das duas coleções em memória, isto é, o conteúdo da coleção primária passa para a coleção secundária e o conteúdo da coleção secundária passa para a coleção primária. Para isso, é necessário que as duas coleções estejam definidas (as coleções são definidas por meio do comando *Synthesize*).

Os próximos comandos permitem visualizar as coleções presentes em memória, assim como informações sobre tais coleções.

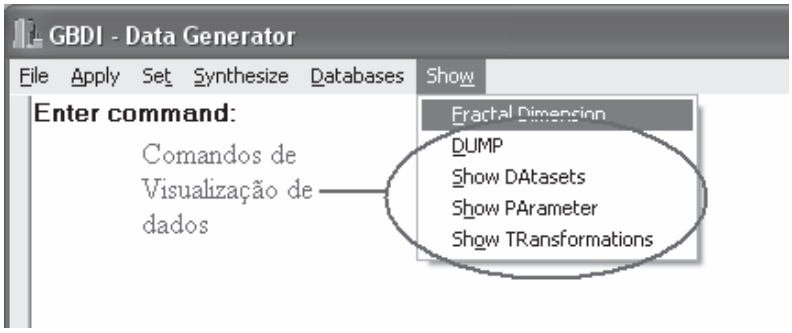

Figura 26: Comandos de Visualização de Dados

- ***Dump (DUMP)***: o comando *dump* permite a visualização das coleções em memória. Se o aplicativo estiver configurado para apresentação das coleções em modo texto, que é o padrão, será apresentada apenas a coleção primária, na forma textual. Já se o aplicativo estiver configurado para apresentação na forma gráfica (*set verbose graphics – set ve gra*), o comando *dump* apresenta a coleção primária e a secundária em forma de um gráfico de duas ou três dimensões, dependendo da dimensionalidade dos objetos das coleções. No caso de dimensionalidade maior que 3, pode ser usada a ferramenta *FastMapDB* para visualização das bases de dados criadas, pois tal ferramenta reduz a dimensionalidade dos dados para visualização. A visualização gráfica conta com o auxílio do aplicativo *GNUPLOT*[7]. Um arquivo temporário de visualização é criado e automaticamente aberto no aplicativo. Os dados das coleções primárias e secundárias são visualizados em cores diferentes no *GNUPLOT*, sendo que é possível alterar vários parâmetros relativos à visualização no próprio aplicativo.

- ***Show (SH)***: O comando *show* pode mostrar informações sobre a coleção primária e a secundária, sobre as transformações configuradas com o comando *Set* e sobre os parâmetros atuais das configurações do aplicativo DBGen. Com o subcomando *Datasets (DA)*, são mostradas informações sobre a coleção primária e a secundária, como o número de objetos e o nome de tais coleções. Já com o subcomando *Parameter (PA)*, podem ser visualizadas informações sobre a atual configuração do aplicativo, como a dimensionalidade das coleções em memória e o modo de visualização configurado no momento. Por fim, com o subcomando *Transformations (TR)*, pode-se ver a taxa de amostragem anteriormente definido pelo comando set *samplingrate*, assim como a matriz de

[7] GNUPLOT Homepage. Disponível em: <http://www.gnuplot.info/>. Acesso em 01 set. 2004.

transformação geométrica gerada pelas transformações configuradas (rotações, translações e escalas).

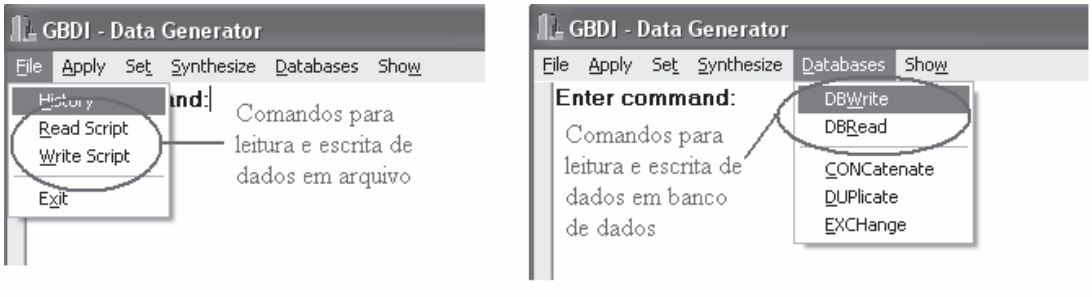

Figura 27: Comandos para Leitura e Escrita de Dados em Arquivos e em Banco de Dados

Por fim, existem os comandos para leitura e escrita dos dados em arquivos e em banco de dados. Tais comandos permitem tanto a armazenagem das coleções sintetizadas em arquivos texto, que podem ser carregados posteriormente, assim como em banco de dados, que é o objetivo do projeto. Iniciemos então com os comandos para gravação e leitura dos dados em arquivos texto:

- ***Write (WRI)***: O comando *write* grava todos os dados da coleção primária em um arquivo texto, assim como a seqüência inicial de comandos que carrega tais dados em memória na coleção primária - quando tal arquivo for carregado. A dimensão dos objetos do arquivo é armazenada inicialmente, assim como o nome da coleção. No arquivo gerado também está incluído o comando *mininsert (MINS)*, que é o responsável pela leitura dos dados quando o arquivo for carregado. O único parâmetro requerido pelo comando *write* é o nome do arquivo a ser gerado.
- ***Read***: O comando *read* lê os arquivos gerados com o comando *write*, assim como qualquer arquivo com seqüências de comandos do aplicativo.

Um arquivo, com uma seqüência de comandos, deve terminar com o comando *close*. Existe a opção *Write Script* no *menu File* que salva em um arquivo todos os comandos executados na atual execução do aplicativo.

Já a leitura de um arquivo gerado com o comando *write*, providencia a leitura dos dados armazenados no arquivo e a inclusão destes dados na coleção com o comando *minsert*, assim como configurações referentes à dimensão de tais dados e nome da coleção. Abaixo, na Figura 28, temos um exemplo de um arquivo gerado com o comando *write*:

```
set dimension 2
show parameter
minsert 5
0.000000  0.000000
0.250000  0.250000
0.500000  0.500000
0.750000  0.750000
1.000000  1.000000
set name Line5
close
```

Figura 28: Exemplo de Arquivo com Comandos Gerados com o Comando Write (WRI)

- **Dbwrite (DBW)**: O comando *dbwrite* grava os dados da coleção primária em uma base de dados. O usuário deve definir o nome do banco de dados, assim como as informações para *login*, o nome da tabela onde serão armazenados os dados e os nomes das colunas onde serão armazenadas as informações relativas a cada eixo dos objetos. Na figuras 29 há um exemplo da interface para escolha dos bancos de dados e das tabelas.

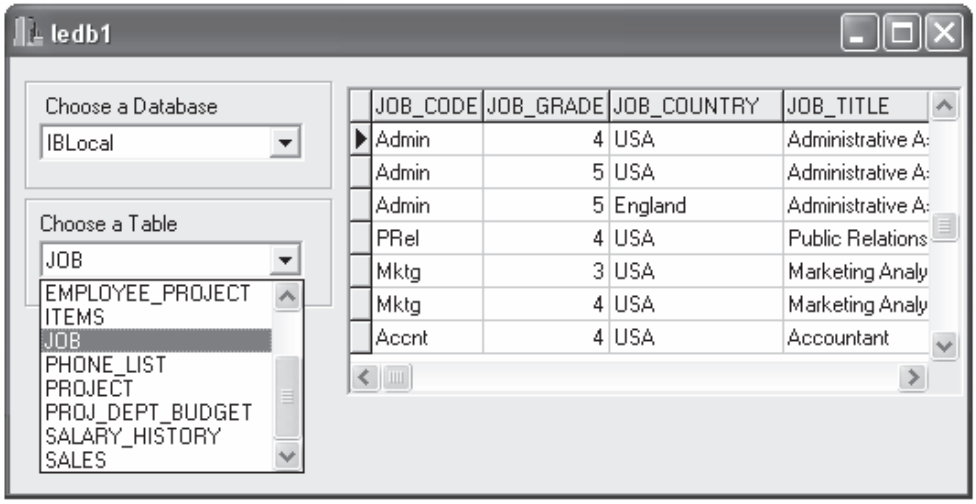

Figura 29: Interface com Bancos de dados

- ***Dbread (DBR)*:** O comando *dbread* faz a leitura dos dados armazenados em uma base de dados. Devem ser informados: o nome do banco de dados, o nome da tabela, e as colunas referentes a cada um dos eixos dos objetos, de acordo com a dimensionalidade dos objetos das coleções.

2.4 L-SYSTEM: Geração de Fractais

Os *L-SYSTEMS* (FLAKE, 1999) são sistemas para modelagem de objetos, muito utilizados para modelar fenômenos naturais, principalmente do reino vegetal, como plantas, árvores, flores e folhas. O nome *L-System* é uma homenagem a seu criador, o biólogo A. Lindenmayer.

Com a utilização de um *L-System*, podem ser modelados objetos complexos por meio da reescrita de partes do objeto, ou seja, partes mais simples do objeto são substituídas sucessivamente, dando origem a objetos mais complexos. Tais substituições obedecem a algumas regras de reescrita, chamadas de produções. O objeto inicial é chamado de axioma.

O aplicativo DBGen utiliza a interpretação gráfica de um *L-System*, conhecida como "gráficos tartaruga" que é um sistema para traduzir uma seqüência de símbolos de entrada em movimentos de um autômato, para gerar um gráfico. O autômato é a "tartaruga". Imagina-se que uma tartaruga está localizada em um plano bidimensional e, dependendo do símbolo de entrada, a tartaruga se move ou muda de direção.

O axioma, os símbolos de entrada e outros parâmetros podem ser configurados com o comando *LSET*, que veremos mais adiante.

O DBGen armazena os pontos dos vértices do gráfico gerado. Tais pontos são armazenados nos eixos (x, y), escolhidos com o comando *set axis*.

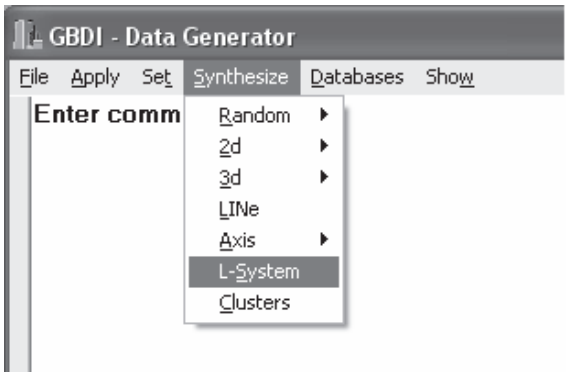

Figura 30: Comando L-System (LSYS)

2.4.1 Comando para configuração do L-System: LSET

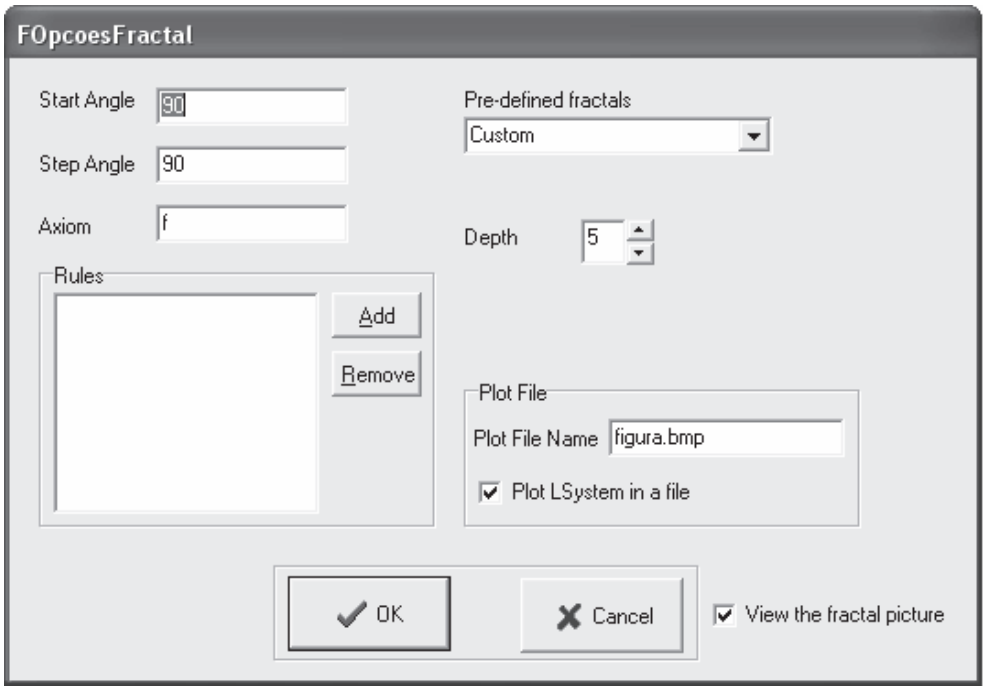

Figura 31: Configuração do L-System

O comando *LSET* é responsável pela configuração do *L-System* que será utilizado para síntese de dados, com o comando *SYNTH LSYS*. Os subcomandos de *LSET* são semelhantes aos parâmetros utilizados pela maioria dos geradores de fractais. Os principais subcomandos de *LSET* estão descritos abaixo:

- ***Rule***: O subcomando *rule* define as produções do *L-System*. As produções são as regras de reescrita dos objetos e devem ser da forma "X = seq", onde "X" é uma letra qualquer e "seq" é uma seqüência de comandos para os "gráficos tartaruga". Tais comandos, que são os símbolos de entrada, são descritos abaixo:
  - **F**: Move a tartaruga uma posição para frente, desenhando uma linha reta entre o ponto inicial e o final.
  - **G**: Move a tartaruga uma posição para frente, mas não desenha linha nenhuma.
  - **+**: Gira a tartaruga para a direita, com um ângulo determinado pelo comando *lset*.
  - **-**: Gira a tartaruga para a esquerda, com um ângulo determinado pelo comando *lset*.
  - **[**: Armazena a posição atual da tartaruga, assim como o seu ângulo, para serem recuperados posteriormente.

- **]**: Recupera o ângulo e a posição armazenados com o símbolo [.

Mais de uma regra podem ser definidas ao mesmo tempo.

- ***Axiom***: O subcomando *axiom* define o axioma inicial, a partir do qual será gerado o fractal. O axioma padrão é F.
- ***A0***: O subcomando *a0* define o ângulo inicial, ou seja, a orientação inicial da tartaruga no plano bidimensional. O valor padrão para a0 é 0 graus.
- ***Da***: o subcomando *da* define o ângulo de rotação dos comandos + e -. O padrão para DA é 0 graus.
- ***Depth (DE)***: O subcomando *depth* configura o número de recursões com o qual o gráfico do *L-System* será gerado. O padrão para depth é 4.
- ***File (FIL)***: Define se deve ser gerado um arquivo *Bitmap* cada vez que é um *Lsystem* é sintetizado.
- ***Namefile (NAM)***: Define o nome do arquivo *Bitmap* gerado.

Todas as configurações do *L-System* podem ser visualizadas com o comando *Lshow (LSYSSHOW)*, que mostra o valor atual dos parâmetros.

2.4.2 Exemplos de fractais gerados

Abaixo, temos alguns exemplos de fractais gerados a partir do *L-System*, seguidos pelos gráficos dos pontos gerados pelo DBGen.

Como primeiro exemplo, temos um fractal conhecido como *Snowflake* ou floco de neve. Para geração desse fractal, temos:

RULE: F = F----F----F----------F++F----F

AXIOM: F----F----F----F----F----F

DA: 18 graus

Na Figura 32 temos exemplos de *Snowflakes* gerados graficamente, com diversos níveis de recursão:

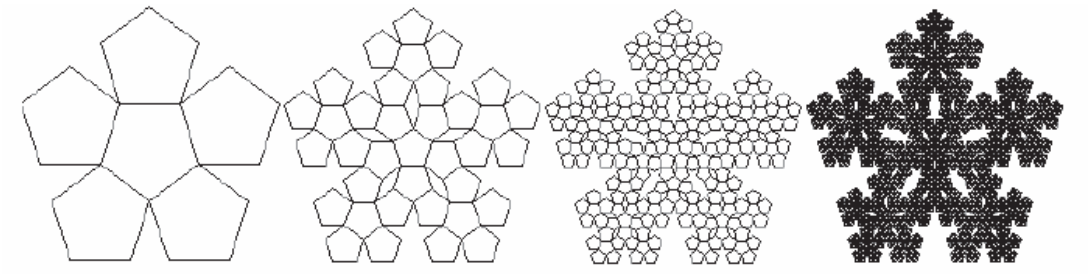

Figura 32: Fractais Snowflake, com 1, 2, 3 e 5 Níveis de Recursão

Para gerarmos no DBGen coleções relativas aos vértices dos dois fractais da Figura 33, usaremos a seguinte seqüência de comandos:

```
LSET RULE F = F----F----F----------F++F----F
LSET AXIOM F----F----F----F----F----F
LSET DA 18
SYNTH LSYS
```

Depois, com o comando *LSET DEPTH X*, escolheremos o nível de recursão do fractal a ser gerado. Na Figura 34, primeiro escolheu-se *X*=1 e depois *X*=2.

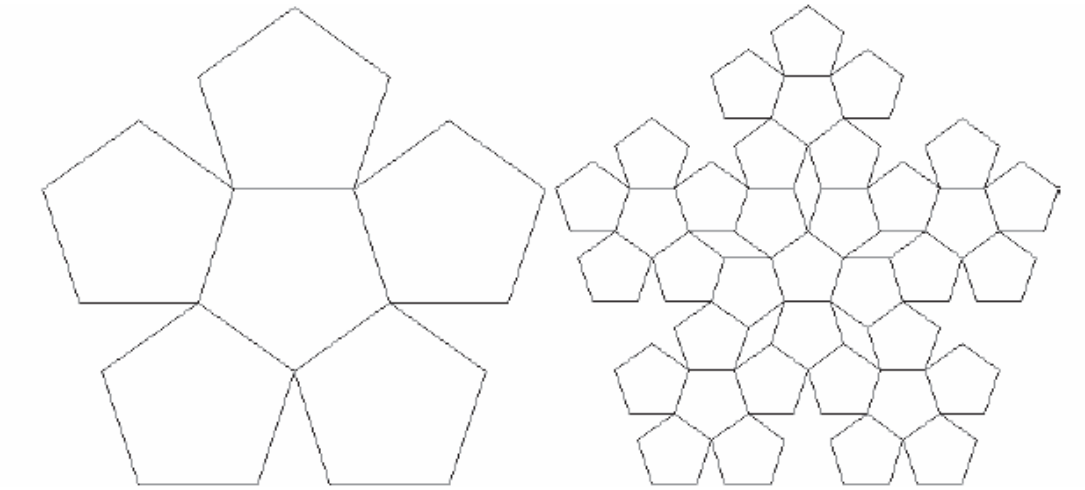

Figura 33: Snowflakes com 1 e 2 Níveis de Recursão (LSET DEPTH 1 e LSET DEPTH 2)

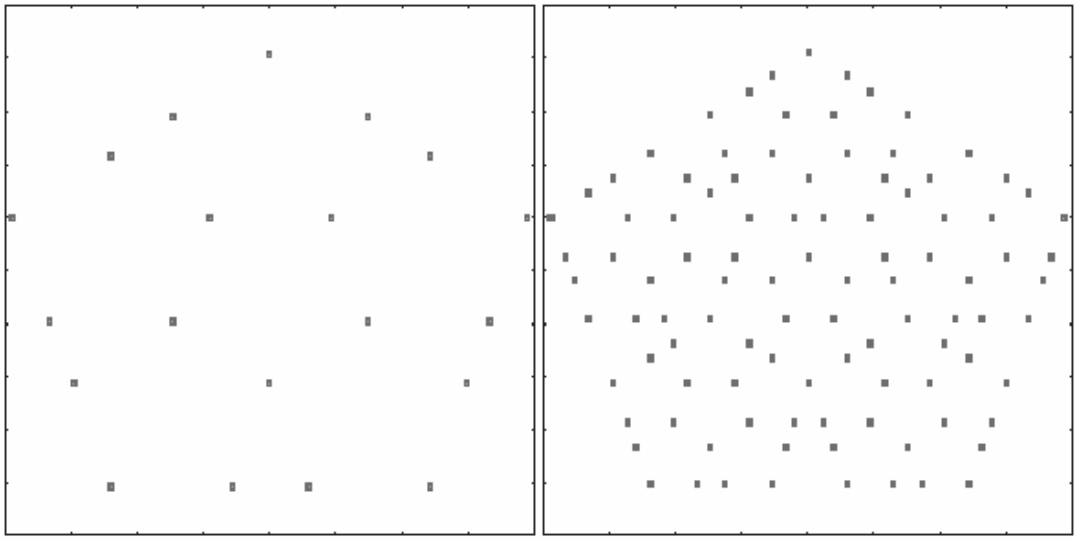

Figura 34: Pontos Gerados para Snowflakes com 1 e 2 Níveis de Recursão (LSET DEPTH 1 e LSET DEPTH 2)

Nesse próximo exemplo é mostrado um fractal conhecido como curva de Koch. A descrição da curva de Koch no *L-System* é:

RULE: F = F-F++F-F

DA: 60 graus

Na Figura 35, vemos a curva de Koch com 1, 2 e 4 níveis de recursão.

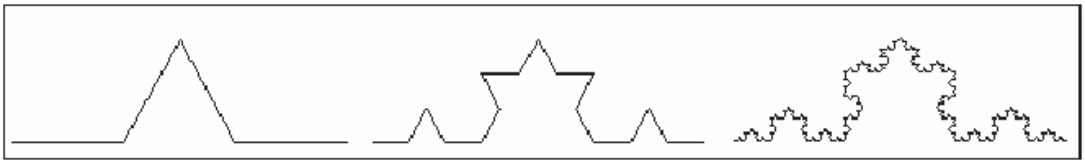

Figura 35: Curvas de Koch com 1, 2 e 4 Níveis de Recursão

No DBGen, para gerarmos coleções relativas aos vértices das curvas de Koch, temos a seguinte seqüência de comandos:

LSET RULE F = F-F++F-F

LSET DA 60

SYNTH LSYS

Com o comando *LSET DEPTH X*, escolhemos o nível de recursão do fractal a ser gerado. Na Figura 36, primeiro escolheu-se *X*=2 e depois *X*=3.

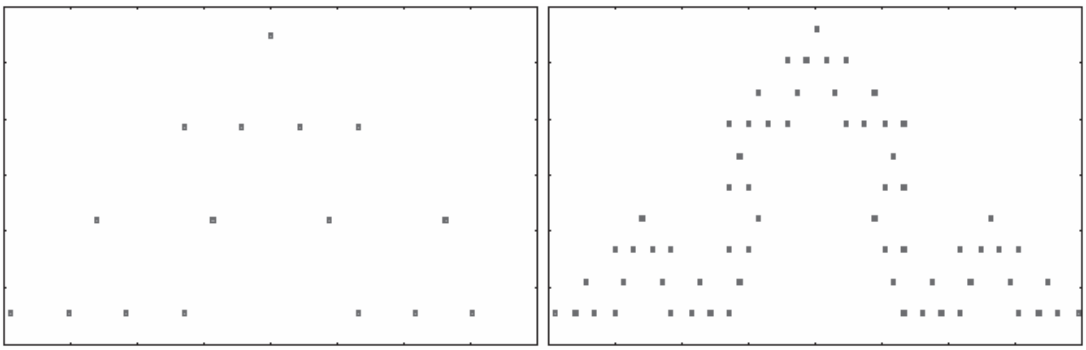

Figura 36: Pontos gerados: Curvas de Koch com 2 e 3 Níveis de Recursão (LSET DEPTH 2 e LSET DEPTH 3)

Nesse último exemplo, utilizamos os símbolos "[" e "]", para armazenar e recuperar a posição atual da "tartaruga".

Os fractais da Figura 37 têm a seguinte configuração:

RULE: F = F[-F][+F]F

AXIOM: F

DA: 45 graus

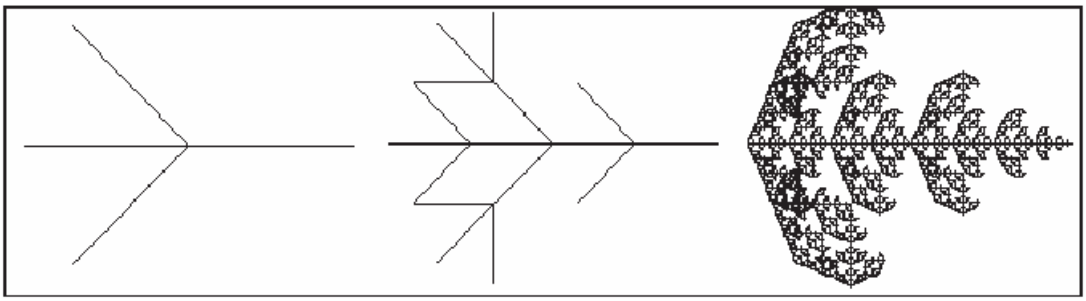

Figura 37: Fractais com 1, 2 e 6 Níveis de Recursão.

Para gerarmos coleções relativas aos fractais da Figura 38, temos a seguinte sucessão de comandos no DBGen:

```
LSET RULE F = F[-F][+F]F
LSET DA 45
SYNTH LSYS
```

Para gerarmos as coleções da Figura 38, com 4 e 7 níveis de recursão, foi usado também o comando *LSET DEPTH 4* e *LSET DEPTH 7*.

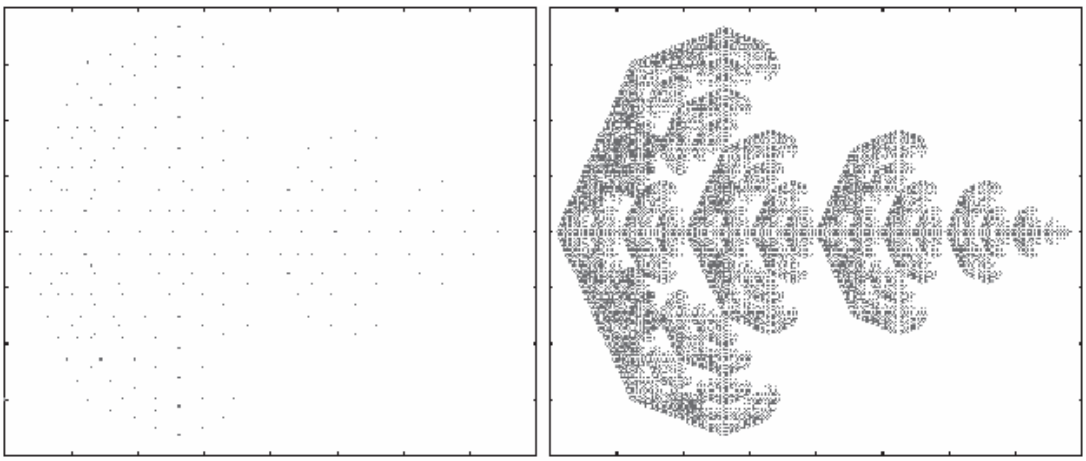

Figura 38: Coleções relativas ao Fractal da Figura 37 com LSET DEPTH 4 e LSET DEPTH 7

Anexo – Árvore de Comandos

File
- History
- Read Script (READ)
- Write Script (WRI)
- Exit

Apply (AP)
- Transformations (TR)

Set (SET)
- Verbose (VE)
  - None (NO ou NONE)
  - All or Full (ALL ou FULL)
  - Object (OB)
  - Count (COU)
  - Information (INF)
  - Time (TI)
  - Debug (DE)
  - Intermediary (INT)
  - Command (COM)
  - Graphics (GRA)
- Namedataset (NAME)
- Dimension (DIM)
- Precision (PRE)
  - Continuous (CO)
  - Discrete (DI)
- Disttype (DIS)
  - Chi-squared random (CHI)
  - Gauss (Normal) random (GA)
  - Uniform random (UN)
- Translation (TRANS)
- Scale (SCAL)
- Rotation (ROT)
- Samplingrate (SAMPL)

Synthesize (SYNTH)
- Random
  - Random (RAN)
  - Rcircumference (RCIRCUM)
  - Rline (RLI)
  - Rplane (RPL)
  - Rsphere (RSP)
- 2d
  - Plane (PL)
  - Sierpinsky (SIE)
  - Circumference (CIRCUM)
  - Fcircumference (FCIRCUM)

- 3d
  - Sphere (SPH)
  - Fsphere (FSP)
- Line (LIN)
- Axis
  - Step (ST)
  - Axis (AX)
  - Raxis (RAX)
  - Caxis (CAX)
  - Maxis (MAX)
  - Mmaxis (MMAX)
  - Msaxis (MSAX)
  - Sinaxis (SIN)
  - Swaxis (SW)
- L-System (LSYS)
- Cluster (CL)

Databases
- Dbwrite (DBW)
- Dbread (DBR)
- Concatenate (CONC)
- Duplicate (DUP)
- Exchange (EXC)

Show (SH)
- Fractal dimension
- Dump (DUMP)
- Show Datasets (DA)
- Show Parameter (PA)
- Show Transformations (TR)

Referências Bibliográficas

FALOUTSOS, C. et al. **Spatial Join Selectivity Using Power Laws**. In: SIGMOD Conference, 2000, Dallas (USA), 2000. p.177-188.

FLAKE, G.W. **The Computational Beauty of Nature – Computer Explorations of Fractals, Chaos, Complex Systems and Adaptations.** London: The MIT Press, 1999.