Análise visual de dados relacionais: uma abordagem interativa suportada por teoria dos grafos

Daniel Mário de Lima

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura:_____

Análise visual de dados relacionais: uma abordagem interativa suportada por teoria dos grafos

Daniel Mário de Lima

Orientador: **Prof. Dr. José Fernando Rodrigues Jr.**

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências - Ciências de Computação e Matemática Computacional. *EXEMPLAR DE DEFESA*.

USP – São Carlos
Novembro de 2013

FichaCcatalográficaCelaboradaCpelaCBibliotecaCProf.CAchilleCBassiC
eCSeçãoCTécnicaCdeCInformática,CICMC/USP,C
comCosCdadosCfornecidosCpelo(a)Cautor(a)

Lzêça Lima1fDanielfMário
Análisefvisualfdefdadosfrelacionaisífuma
abordagemfinterativafsuportadafporfteoriafdos
grafosfPfDanielfMáriofLima;forientadorfJosé
FernandofJr(fRodrigues(fTTfSãofCarlos1fçóGê(
hIfp(

TesefSDoutoradofTfProgramafdefPósTGraduaçãofem
CiênciasfdefComputaçãofefMatemáticafComputacionalãfTT
InstitutofdefCiênciasfMatemáticasfefdefComputação1
UniversidadefdefSãofPaulo1fçóGê(

G(fbancosfdefdadosfrelacionais(fç(fanálise
visual(fê(fvisualizaçãofdefgrafosfhierárquica(fI(
Rodrigues1fJoséfFernandofJr(1forient(fII(fTítulo(f

Dedicado aos meus avós *Dodó* (*in memoriam*) e *Dodô*, familiares, amigos e mestres que me acompanharam por esta longa jornada, e por muitos dias ainda haverão de me acompanhar.

Agradecimentos

Apesar do caráter individual solicitado pelos diversos trabalhos acadêmicos, a monografia é um exemplo de resultado concebido no seio do ambiente universitário, onde o principal benefício é o convívio – o constante diálogo, discussão de argumentos e exposição a novas ideias e horizontes. Por isso é com alegria que aqui presto os meus agradecimentos, que vão às diversas pessoas que me ofereceram palavras, ideias, sugestões, críticas e inspiração: ao meu orientador Prof. Júnior; aos professores e amigos do GBDI, de outros laboratórios do ICMC, e também a alguns amigos que conheci em outros institutos; aos funcionários da USP que me auxiliaram prontamente com as tarefas burocráticas; e aos vizinhos e amigos de São Carlos, pela agradável convivência no dia-a-dia da cidade. Agradeço também aos órgãos que prestaram apoio financeiro e material ao desenvolvimento deste estudo: o Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPQ, que financiou este trabalho sob o processo de número 132943/2011-5; a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – CAPES e a Fundação de Amparo a Pesquisa do Estado de São Paulo – FAPESP, que financiaram não apenas vários dos trabalhos relacionados a esta pesquisa, como também diversas outras pesquisas e materiais no GBDI.

"Porque a sabedoria serve de defesa,
como de defesa serve o dinheiro;
mas a excelência do conhecimento é que
a sabedoria dá vida ao seu possuidor."
(Bíblia Sagrada, Eclesiastes 7:12)

Resumo

Bancos de dados relacionais são fontes de dados rigidamente estruturadas, caracterizadas por relacionamentos complexos entre um conjunto de relações (tabelas). Entender tais relacionamentos é um desafio, porque os usuários precisam considerar múltiplas relações, entender restrições de integridade, interpretar vários atributos, e construir consultas SQL para cada tentativa de exploração. Neste cenário, introduz-se uma metodologia em duas etapas; primeiro utiliza-se um grafo organizado como uma estrutura hierárquica para modelar os relacionamentos do banco de dados, e então, propõe-se uma nova técnica de visualização para exploração relacional. Os resultados demonstram que a proposta torna a exploração de bases de dados significativamente simplificada, pois o usuário pode navegar visualmente pelos dados com pouco ou nenhum conhecimento sobre a estrutura subjacente. Além disso, a navegação visual de dados remove a necessidade de consultas SQL, e de toda complexidade que elas requerem. Acredita-se que esta abordagem possa trazer um paradigma inovador no que tange à compreensão de dados relacionais.
Palavras-chaves: bancos de dados relacionais, análise visual, visualização hierárquica de grafos.

Abstract

Relational databases are rigid-structured data sources characterized by complex relationships among a set of relations (tables). Making sense of such relationships is a challenging problem because users must consider multiple relations, understand their ensemble of integrity constraints, interpret dozens of attributes, and draw complex SQL queries for each desired data exploration. In this scenario, we introduce a twofold methodology; we use a hierarchical graph representation to efficiently model the database relationships and, on top of it, we designed a visualization technique for rapidly relational exploration. Our results demonstrate that the exploration of databases is deeply simplified as the user is able to visually browse the data with little or no knowledge about its structure, dismissing the need of complex SQL queries. We believe our findings will bring a novel paradigm in what concerns relational data comprehension.
Keywords: relational databases, visual analysis, hierarchical graph visualization.

Lista de ilustrações

Figura 1 – Especificações visuais no sistema Polaris. 26
Figura 2 – Visualização resultante de uma especificação visual. 27
Figura 3 – Propriedades visuais no sistema Polaris. 29
Figura 4 – Hierarquia de especificações visuais em um Zoom Graph. 30
Figura 5 – Visualização de gráficos de área empilhados. 31
Figura 6 – Elementos do *Cube Presentation Model*. 32
Figura 7 – Exemplo de *Table Lens* 33
Figura 8 – *Table Lens* associado ao CPM 33
Figura 9 – *Barsticks* representando dimensões na HDDV. 34
Figura 10 – *Roll-up* na HDDV. 34
Figura 11 – Árvore de decomposição aprimorada. 35
Figura 12 – Hierarquia de um cubo por extensão e intenção. 36
Figura 13 – Visualização de *Zoom Trees*. 37
Figura 14 – Seleção de atributos na *Zoom Tree*. 37
Figura 15 – Detalhamento de uma célula na *Zoom Tree*. 38
Figura 16 – Anticadeias. 38
Figura 17 – Separação de componentes biconexos. 39
Figura 18 – Correção de ramos degenerados. 39
Figura 19 – Visualização de grafo no sistema ASK-GraphView. 40

Figura 20 – Abordagens baseadas em nós interligados 45
Figura 21 – Outras abordagens baseadas em nós interligados 45
Figura 22 – Projeções em um espaço hiperbólico 46
Figura 23 – Desenho de árvores com preenchimento do espaço 46
Figura 24 – Comparando as abordagens de desenho de árvores 47
Figura 25 – Visualizações de grafos baseadas em nós e arestas 49
Figura 26 – Comparando as duas abordagens 50
Figura 27 – Abordagens híbridas para o desenho de grafos 50

Figura 28 – Visualização em níveis de detalhe de grafos agrupados 51

Figura 29 – SuperGrafo obtido a partir de um grafo particionado. 54
Figura 30 – Estrutura Graph-Tree em disco. 56
Figura 31 – Layout visual dos sistemas RMine e GMine 59
Figura 32 – Diagrama de classes da interface gráfica 60
Figura 33 – Diagrama de classes das estruturas de dados 60
Figura 34 – Construção de um SuperGrafo. 62
Figura 35 – Visualização hierárquica de um SuperGrafo. 63
Figura 36 – Distribuição do atributo *Pessoa.idade*. 64
Figura 37 – Cálculo de SuperAresta em uma Graph-Tree 67

Figura 38 – Diagrama do modelo relacional do banco de dados Tycho-USP. 70
Figura 39 – Sequência de interações ilustrando a expansão de SuperNós. 71
Figura 40 – Sequência de interações explorando os relacionamentos entre SuperNós. . 73
Figura 41 – Tempo de execução de cada cálculo de conectividade no RMine e consulta SQL correspondente no PostgreSQL. 75
Figura 42 – Tempo de execução acumulado dos cálculos de conectividade no RMine e consultas SQL correspondente no PostgreSQL. 75

Lista de tabelas

Tabela 1 – Tempo acumulado medido para a expansão de cada SuperNó. 75
Tabela 2 – Tempos de execução das tarefas de usuário com o cliente SQL pgAdmin e o sistema RMine, em segundos. 77
Tabela 3 – Teste de normalidade Kolmogorov-Smirnov sobre as amostras de tempo. . 77
Tabela 4 – Testes de significância unicaudais sobre pares $\langle x_S, x_R \rangle$ de amostras de tempo. 78

Sumário

Lista de ilustrações **13**

Lista de tabelas **15**

Sumário **17**

1 Introdução **19**
1.1 Motivação 21
1.2 Justificativa 21
1.3 Objetivos 22
1.4 Organização do trabalho 22

2 Análise visual de bancos de dados **25**
2.1 Análise visual de cubos de dados 25
2.2 Visualização hierárquica em cubos de dados 29
2.2.1 *Zoom Graph* 29
2.2.2 *Cube Presentation Model* 31
2.2.3 *Hierarchical Dynamic Dimensional Visualization* 33
2.2.4 *Enhanced Decomposition Tree* 35
2.2.5 *Zoom Tree* 36
2.3 Visualização hierárquica em grafos 37
2.4 Conclusão do capítulo 40

3 Visualização de grafos **43**
3.1 Layout visual 43
3.2 Desenho de árvores 44
3.2.1 Nós interligados 44
3.2.2 Preenchimento de espaço 45
3.2.3 Abordagens híbridas 47
3.3 Desenho de grafos 47
3.3.1 Vértices e arestas 48

3.3.2 Matrizes de adjacência . . . 49
3.3.3 Abordagens híbridas . . . 50
3.3.4 Outras abordagens . . . 51
3.4 Conclusão do capítulo . . . 51

4 Metodologia . . . 53
4.1 SuperGrafos . . . 53
4.2 Graph-Tree orientada a disco . . . 55
4.3 Sistema RMine . . . 57
4.4 Layout visual . . . 58
4.5 Desenvolvimento . . . 59
4.6 Particionamento Relacional . . . 61
4.7 Conclusão do capítulo . . . 66

5 Experimentos . . . 69
5.1 Dados e configurações . . . 69
5.2 Análise Visual . . . 70
5.3 Desempenho do protótipo . . . 74
5.4 Testes de usabilidade . . . 76
5.5 Conclusão do capítulo . . . 78

6 Conclusão . . . 79

Referências . . . 81

Capítulo

1 Introdução

Durante as últimas décadas, uma grande quantidade de informação tem sido gerada, fazendo com que seja comum encontrar grandes bancos de dados em vários tipos de aplicações. Exemplos desse crescimento são encontrados em dados gerados pela indústria, onde informações dos clientes, produtos e transações de múltiplos tipos são armazenadas de maneira relacional. Nesses bancos de dados, as entidades são descritas como atributos e se referenciam mutuamente em relacionamentos que definem uma forte coesão estrutural.

Os Sistemas Gerenciadores de Bancos de Dados (SGBD's) são as soluções para tais dados estruturados; eles provém um armazenamento inteligente, com poderosas capacidades de consulta, em aplicações que variam de comércio a educação. E, apesar de terem surgido soluções não-relacionais no Mercado – como bancos de dados NoSQL, os SGBD's ainda respondem pela maior fatia de mercado (AGRAWAL et al., 2008; ANTHES, 2010). Entretanto, mesmo que os SGBD's sejam engenhosamente projetados para o armazenamento de dados, eles não são adequados para a análise e interpretação – este é o problema debatido neste trabalho.

Quando o objetivo é obter informação útil, armazenar e recuperar dados são apenas uma parte do problema. Raciocinar sobre dados complexos e volumosos é uma tarefa árdua, uma dificuldade que demanda processos analíticos que encontrem padrões, estruturas incomuns, relacionamentos ocasionais, e outros tipos de conhecimento que possam auxiliar no suporte à decisão. Tais processos são usualmente realizados de maneira exploratória, supondo que o analista não sabe *a priori* o que procurar. Nestas circunstâncias, um ambiente visual rico e responsivo pode prover resultados interessantes, notadamente para dados estruturados.

Considerando esse cenário, técnicas baseadas em hierarquias têm sido estudadas para tratar problemas analíticos; hierarquias levam a informação a ser compreendida ao longo de níveis progressivamente restritivos de abrangência, permitindo que informações cada vez mais detalhadas sejam inspecionadas, no entanto, sem perder seu contexto. A visualização hierárquica, uma dessas técnicas, é bastante estudada e vem mostrando resultados promissores, sendo também muito utilizada em aplicações comerciais.

Uma abordagem direta para investigar dados estruturados é usar representações em grafo de tal maneira que nós e arestas correspondam, respectivamente, às instâncias de entidades e

de relacionamentos previstas no Modelo Entidade-Relacionamento (MER) de Chen (1976) e materializadas como um esquema relacional no SGBD. Dentro destas considerações, aqui é experimentada uma representação em grafos baseada em particionamento hierárquico, uma técnica que melhora a escalabilidade das visualizações baseadas em grafo.

Assim, esta proposta utiliza a estrutura relacional para gerar um grafo inicial que é hierarquicamente particionado de acordo com as entidades, atributos e valores encontrados no esquema do banco de dados. Este grafo hierarquicamente particionado, então, dá seguimento a uma visualização multi-nível composta de nós, grupos de nós, arestas e sumarizações a partir das quais a consulta e agregações interativas se estabelecem.

No contexto de um banco de dados relacional, este trabalho envolve-se em responder aos seguintes questionamentos:

- Como os dados (instâncias de entidades) estão distribuídos nas relações do banco de dados?
- Como as entidades do banco de dados estão relacionadas entre si?
- Como os vários atributos dessas instâncias influenciam nos relacionamentos entre as entidades?
- Como pode ser possível navegar os dados relacionais de forma rápida e interativa, considerando sua complexa estrutura?

Estas questões são respondidas usando-se particionamentos hierárquicos de grafo, criados a partir de ambos: estrutura e dados encontrados no banco de dados a ser analisado. Sobre este método é definido um inovador esquema visual/interativo instanciado em um protótipo operacional. Esta contribuição torna a exploração de relacionamentos entre entidades de dados intuitiva e computacionalmente rápida, mesmo ao se considerar grandes bases de dados. De acordo com esta técnica, a estrutura do banco de dados pode ser navegada por meio de caminhos de exploração pelos quais o usuário pode visualizar entidades e seus relacionamentos sem explicitamente definir consultas em Structured Query Language (SQL).

Utilizando esta ideia, este trabalho se baseia na estrutura hierárquica de grafo Graph-Tree, e no conceito formal de SuperGrafos (RODRIGUES et al., 2013), conceitos que definem um método de particionamento e representação hierárquica para grandes grafos. Neste trabalho, estes conceitos são adaptados e aplicados para a análise de dados relacionais concretizando uma nova contribuição cuja prova de conceito é definida em um sistema protótipo denominado RMine.

A hipótese do trabalho é a de que bancos de dados relacionais podem ser mais bem e mais rapidamente compreendidos por meio de representações visuais interativas baseadas em grafos hierarquicamente particionadossão processados sobre uma estrutura de armazenamento e recuperação de dados em disco.

1.1 Motivação

No contexto da análise de bancos de dados relacionais, várias dificuldades existem ao se considerar a estrutura do modelo MER. A representação de relacionamentos em relações separadas requer o uso de operadores de junção e eliminação de repetições, o que é um grande gargalo ao se trabalhar com grandes bases de dados. Outro problema é que é necessário um bom entendimento do modelo relacional e a linguagem de consulta SQL. A necessidade de se mapear mentalmente a estrutura dos dados à implementação física no modelo relacional impõe uma carga cognitiva ainda maior ao processo analítico.

Para resolver esses problemas, uma das ideias mais consolidadas é a modelagem dimensional em cubos de dados de Gray et al. (1996). Esse modelo foca na análise de uma relação de itens unívocos, como transações individuais, provendo a possibilidade de efetuar consultas em variados níveis de agregação, de uma maneira computacionalmente eficiente. Por esse motivo, é uma abordagem bastante estudada e utilizada em diversas aplicações de análise de dados, como demonstrado em diversos estudos (STOLTE; TANG; HANRAHAN, 2002), (MANIATIS et al., 2003a), (MANSMANN; SCHOLL, 2007) e (WANG et al., 2011).

No entanto, a modelagem em cubos de dados sofre uma limitação considerável: um banco de dados com várias entidades e relacionamentos precisa ser desnormalizado, fornecendo uma única relação que poderá ser utilizada para construção do cubo de dados. Este procedimento pode remover informações ou significado dos dados originais, como abordado por Džeroski (2003), levando à necessidade de se considerar técnicas para trabalhar diretamente com as múltiplas relações do banco de dados.

1.2 Justificativa

A exploração das estruturas encontradas em bancos de dados relacionais é tema de pesquisa em diversas áreas, o quê se denomina, de maneira geral, como Mineração de Dados Multi-Relacional (KNOBBE, 2004). Uma das maneiras de se representar bancos multi-relacionais é através de programação lógica indutiva (Inductive Logic Programming – ILP). Nessa abordagem, as tuplas são transformadas em uma tabela de proposições sobre as entidades, que pode assim ser exploradas por predicados lógicos, representando sequências de itens, estruturas locais e outros tipos de padrões. Há ainda a mineração multi-relacional de dados semi-estruturados, como bases representadas em XML (eXtensible Markup Language), contendo uma mistura de dados e informações estruturais que pode ser explorada por meio de linguagens de consulta estruturadas, como XQuery. Outra abordagem é a representação em grafos, a qual busca uma representação versátil dos dados multi-relacionais, como grafos de entidades ligadas por relacionamentos. Esse paradigma permite a utilização de diversas técnicas de representação, visualização e mineração de grafos diretamente sobre o contexto multi-relacional.

Uma das grandes vantagens da representação multi-relacional em grafos é a capacidade de manter diversas propriedades após certas transformações e processamentos. Um exemplo dessa característica é a filtragem de arestas por amostragem probabilística, como demonstrado por Tsourakakis et al. (2009). Ao selecionar aleatoriamente arestas com uma probabilidade determinada, várias medidas e sub-estruturas são mantidas em proporções conhecidas. Assim, considerando-se os muitos trabalhos que usam grafos como ferramenta de análise de dados (AUBER et al., 2003; ABELLO; HAM; KRISHNAN, 2006; SATO; MISUE; TANAKA, 2008; TOMINSKI; ABELLO; SCHUMANN, 2009; FORTUNATO, 2010; ARCHAMBAULT; MUNZNER; AUBER, 2011; MARTINS et al., 2012), decidiu-se usar esta modalidade analítica como metodologia de mineração visual de dados multi-relacional.

1.3 Objetivos

Como objetivo geral, este trabalho busca estudar a análise de bancos de dados relacionais sob uma perspectiva multi-relacional, investigando a aplicação da representação em grafos e a utilização de diversas técnicas dentre as revisadas na literatura. Os objetivos específicos deste trabalho são:

- Propor e desenvolver uma estrutura de dados orientada a disco capaz de armazenar e acessar informações de um grafo hierarquicamente particionado;
- Projetar o layout e a interação multi-resolução de um sistema exploratório de dados relacionais representados como um grafo hierárquico;
- Definir uma metodologia de análise de bancos dados relacionais usando técnicas hierárquicas sobre grafos;
- Desenvolver e avaliar a metodologia proposta por meio de um protótipo funcional considerados aspectos de potencial analítico, desempenho, e usabilidade;
- Demonstrar a aplicação da ferramenta em um banco de dados de interesse para o público acadêmico.

1.4 Organização do trabalho

Este documento é organizado em 5 capítulos. Os capítulos 2 e 3 revisam a literatura relacionada ao problema, especificamente a análise visual de dados relacionais e grafos no capítulo 2, e a visualização computacional de grafos no capítulo 3. Em seguida, o capítulo 4 com a metodologia proposta no sistema RMine, onde se definem: (a) um particionamento hierárquico para bancos de dados relacionais; (b) como este particionamento é representado em uma estrutura de grafos hierárquica; (c) e o ambiente de visualização utilizando esta estrutura de dados.

No capítulo 5 são apresentados alguns experimentos que exemplificam os principais aspectos desta abordagem e como ela simplifica o processo de manipulação de dados. Ao final, o capítulo 6 conclui o trabalho com uma breve discussão dos principais resultados alcançados, das limitações desta proposta, e também destacando algumas ideias para melhorias futuras.

Capítulo

2 Análise visual de bancos de dados

Analisar bancos de dados relacionais é um tópico há muito discutido, dada sua ampla utilização nas mais diversas operações e ambientes organizacionais. Nessa área, uma das abordagens mais conhecidas é a de *cubo de dados*, proposta por Gray et al. (1996). A modelagem em cubo de dados é amplamente utilizada em uma série de sistemas de apoio à decisão em empresas e organizações, e provê estruturas generalizadas para a execução de serviços OLAP (*on-line analytical processing*) sobre bancos de dados relacionais e data warehouses (THOMSEN, 2002).

Alternativamente às abordagens OLAP, foram desenvolvidos métodos de visualização interativa e multi-escala, preenchendo a lacuna da necessidade por interfaces mais simples e naturais. Isto ampliou o acesso à análise de dados a um público maior de usuários, permitindo que tarefas analíticas fossem efetuadas sobre esses sistemas, trazendo ao usuário a possibilidade de extrair conhecimento.

Uma das abordagens visual-interativas mais exploradas é o uso de visualizações baseadas em hierarquias de dados. O quê, mais especificamente neste trabalho, é abordado segundo hierárquicas definidas sobre o modelo de grafos, os quais fornecem o ferramental necessário à metodologia por meio do sistema proposto, o RMine. A seguir recapitulam-se estes tópicos, estabelecendo-se uma revisão da literatura relacionada à metologia proposta neste trabalho.

2.1 Análise visual de cubos de dados

O sistema Polaris desenvolvido por Stolte, Tang e Hanrahan (2002) é uma das mais referenciadas obras relacionadas à análise visual de cubos de dados, sendo tomada como a base de vários métodos de análise sobre bancos de dados relacionais. O sistema Polaris proporciona uma interface para desenvolver *especificações visuais* que correspondem a consultas analíticas sobre um cubo de dados.

As especificações visuais usadas no sistema Polaris são descrições utilizadas para executar ações sobre os registros do banco de dados e sobre a correspondente visualização. Partindo de um banco de dados a ser modelado em um cubo de dados, uma especificação visual é a

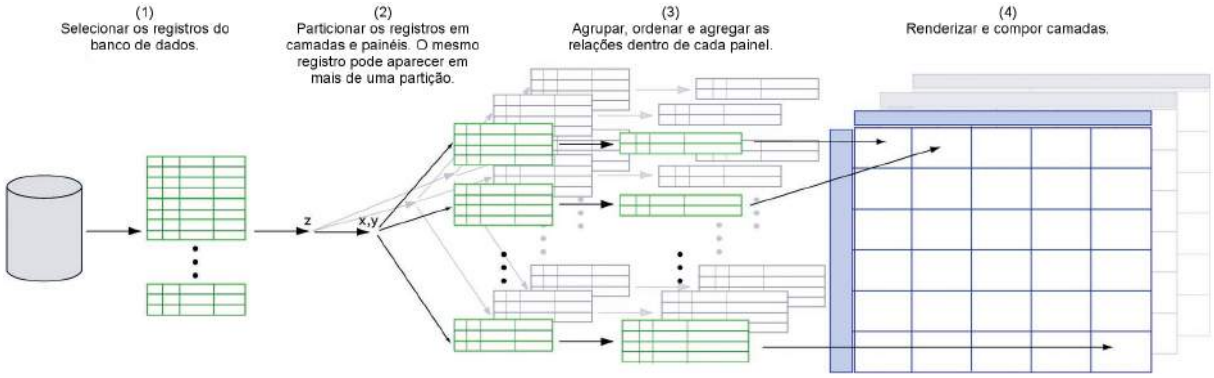


Figura 1 – Especificações visuais no sistema Polaris. Adaptado de Stolte, Tang e Hanrahan (2002).

associação dos atributos de uma tabela para cada um dos eixos desse cubo de dados, juntamente com as definições necessárias para as seguintes tarefas:

- Seleção dos registros (dados) a serem visualizados;
- Transformações nesses registros;
- Particionamentos, agregações e ordenações;
- Visualização apropriada e seus parâmetros.

Dessa maneira, o cubo de dados resultante no sistema Polaris é organizado de modo que cada célula apresenta a visualização (gráficos de dispersão, glifos, ou qualquer outra visualização) de faixas de dados específicas sobre os atributos selecionados, nas diferentes granularidades associadas às camadas do cubo de dados. Um exemplo da etapa final desse processo é a visualização apresentada na Figura 2, onde em um banco de dados de uma empresa é possível visualizar os lucros e vendas de diferentes produtos em cada localidade no decorrer do ano.

Assim, o sistema Polaris utiliza a representação dos dados em cubos de dados (etapas 1 a 3 da Figura 1) e múltiplas visualizações em uma grade tabular (etapa 4 da referida figura). Uma das motivações em se utilizar esta abordagem parte da estrutura essencialmente tabular dos cubos de dados. Segundo Stolte, Tang e Hanrahan (2002), as tabelas possuem várias características que as tornam particularmente efetivas para a análise de dados multi-dimensionais:

- **Multivariadas**: é possível representar diretamente as múltiplas dimensões de um conjunto de dados na estrutura de um cubo multi-dimensional;
- **Comparativas**: as tabelas dispõem os dados em pequenas células lado-a-lado, permitindo a comparação imediata;
- **Familiares**: são representações conhecidas e utilizadas historicamente por estatísticos e outros estudiosos para armazenar e visualizar dados.

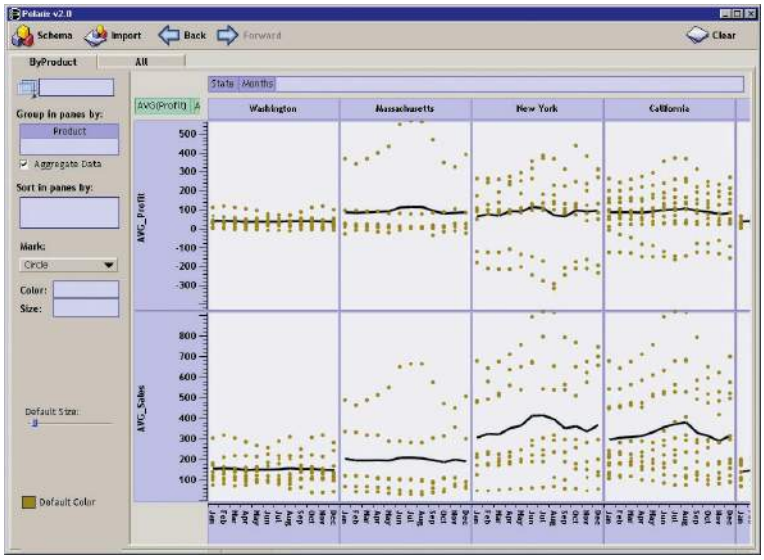


Figura 2 – Visualização resultante de uma especificação visual. Nesse exemplo cada coluna representa um estado cujas linhas são os gráficos das médias de vendas (sales) e lucro (profit) das vendas de cada produto nesse estado, por período mensal dentro de um ano. Adaptado de Stolte, Tang e Hanrahan (2002).

Uma segunda motivação para o uso de tabelas – e de cubos – é a possibilidade de utilizar operadores algébricos semelhantes aos da álgebra relacional. Assim, Stolte, Tang e Hanrahan (2002) definem os operadores de concatenação ($+$), aninhamento ($/$), e cruzamento ($\times$). A composição desses operadores é usada para definir as transformações, particionamentos e agregações de dados em uma especificação visual. Os operandos desses operadores são os atributos dos dados.

Nessa definição, atributos ordinais são representados como os respectivos conjuntos dos valores possíveis para cada atributo e os atributos quantitativos são representados por conjuntos unitários com os respectivos nomes dos atributos. Tomando A e B como atributos ordinais e P e Q como atributos quantitativos, eles são representados como:

$$\begin{aligned} A &= dominio(A) = \{a_1,\ a_2,\ \ldots,\ a_m\} \\ B &= dominio(B) = \{b_1,\ b_2,\ \ldots,\ b_n\} \\ P &= \{P\} \\ Q &= \{Q\}. \end{aligned}$$

A operação de concatenação ($+$) é uma união ordenada dos elementos dos operandos:

$$\begin{aligned} P + Q &= \{P\} + \{Q\} \\ &= \{P,\ Q\} \\ A + P &= \{a_1,\ a_2,\ \ldots,\ a_m\} + \{P\} \\ &= \{a_1,\ a_2,\ \ldots,\ a_m,\ P\} \\ A + B &= \{a_1,\ a_2,\ \ldots,\ a_m\} + \{b_1,\ b_2,\ \ldots,\ b_n\} \\ &= \{a_1,\ a_2,\ \ldots,\ a_m,\ b_1,\ b_2,\ \ldots,\ b_n\}. \end{aligned}$$

A operação de cruzamento ($\times$) é o produto cartesiano entre os conjuntos:

$$\begin{aligned}
P \times Q &= \{P\} \times \{Q\} \\
&= \{PQ\} \\
A \times P &= \{a_1,\ a_2,\ ...,\ a_m\} \times \{P\} \\
&= \{a_1P,\ a_2P,\ ...,\ a_mP\} \\
A \times B &= \{a_1,\ a_2,\ ...,\ a_m\} \times \{b_1,\ b_2,\ ...,\ b_n\} \\
&= \{a_1b_1,\ a_2b_1,\ ...,\ a_mb_1, \\
&\quad\ a_2b_1,\ a_2b_2,\ ...,\ a_mb_2, \\
&\quad\ \vdots \\
&\quad\ a_mb_1,\ a_mb_2,\ ...,\ a_mb_n\}.
\end{aligned}$$

E finalmente, a operação de aninhamento ($/$) resulta em um subconjunto do produto cartesiano, contendo apenas os elementos cujos atributos existem em ao menos um registro do conjunto de dados. Sejam A e B dois atributos, e $A(r)$ e $B(r)$ os valores respectivos desses atributos em um registro r do conjunto de dados R, a operação A/B é assim definida:

$$A/B = \{a_ib_j : \exists r \in R : A(r) = a_i \wedge B(r) = b_j\}.$$

Um exemplo dessa operação é o aninhamento entre um atributo mês $M = \{jan, fev, ..., dez\}$ e um atributo derivado trimestre $T = \{t_1, t_2, t_3, t_4\}$. Supondo que existem registros em todos os meses, o resultado de T/M seria:

$$\begin{aligned}
T/M =& \{t_1\} \times \{jan,\ fev,\ mar\} \\
&+\{t_2\} \times \{abr,\ mai,\ jun\} \\
&+\{t_3\} \times \{jul,\ ago,\ set\} \\
&+\{t_4\} \times \{out,\ nov,\ dez\} \\
T/M =& \{t_1jan,\ t_1fev,\ t_1mar,\ t_2abr,\ ...,\ t_4dez\}.
\end{aligned}$$

Onde cada trimestre é associado a cada um dos meses correspondentes. No entanto, é interessante aninhar atributos de domínios diferentes, criando agrupamentos hierarquicamente detalhados por esses atributos. Por exemplo, $Localidade/Trimestre$ agrupa itens por trimestre dentro de cada Localidade, como uma cláusula `GROUP BY (Localidade, Trimestre)` em uma consulta SQL. A partir desses operadores, o processo de seleção / transformação / agrupamento dos dados para a visualização ilustrada na Figura 2 pode ser definido com as expressões:

	x = State / Month
y = AVG(Profit)	... ...
+ AVG(Sales)	... ...

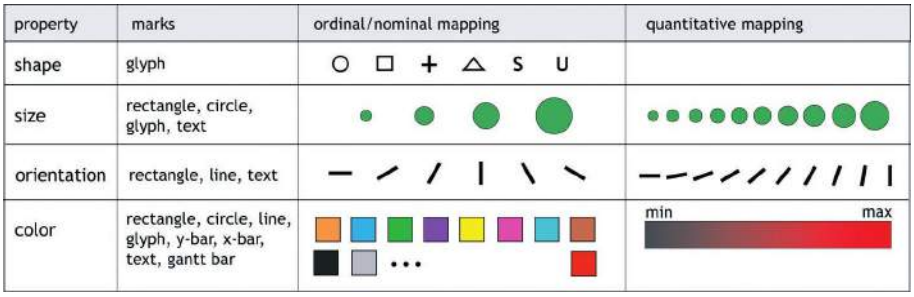


Figura 3 – Propriedades para representação de dados nas visualizações do sistema Polaris. Fonte: Stolte, Tang e Hanrahan (2002).

Seguindo com a escolha de uma visualização apropriada, esta expressão é renderizada em uma tabela com os dados das médias de vendas ($AVG(Sales)$) e lucros ($AVG(Profit)$) sobre produtos em cada partição por Estado e Mês ($State/Month$), trazendo o resultado na Figura 2. O sistema Polaris ainda apresenta várias opções de gráficos para as células dessa visualização, e permite associar os valores a diversas propriedades da visualização, como ilustrado na Figura 3.

Em suma, o sistema Polaris consegue agregar informações a partir de um conjunto de dados relacionais, permitindo sua análise sob diversas perspectivas, trazendo uma gama interessante de ferramentas para a análise.

2.2 Visualização hierárquica em cubos de dados

A seção anterior revisita os conceitos da modelagem em cubos de dados, exemplificados em uma aplicação extensa de seleção e visualização de informações, o Polaris. Partindo dessas abordagens baseadas em cubos de dados, esta seção expõe os estudos que tratam de explorar esses cubos com visualizações multi-escala: seguindo um processo de visão geral, zoom, filtragem e detalhamento sob demanda.

2.2.1 *Zoom Graph*

Prosseguindo com a pesquisa sobre o sistema Polaris, Stolte, Tang e Hanrahan (2003) estendem a visualização em cubos de dados e desenvolvem os *Zoom Graphs*, uma visualização multi-escala de dados hierarquicamente estruturados, que é projetada a partir da notação formal das especificações visuais existente no Polaris.

Um Zoom Graph pode ser entendido como um grafo de transições entre diferentes níveis de detalhes de uma visualização. Cada nó desse grafo é uma visualização definida previamente no sistema Polaris por meio de uma especificação visual, como discutido na seção 2.1. Assim, é possível construir um conjunto de especificações visuais das mesmas dimensões em diferentes escalas, e encadeá-las com transições de um Zoom Graph.

A Figura 4 ilustra esse conceito, onde a estrutura que abstrai os dados é o grafo de transição entre especificações visuais, e a abstração visual é a visualização fornecida pelo Polaris. Dessa

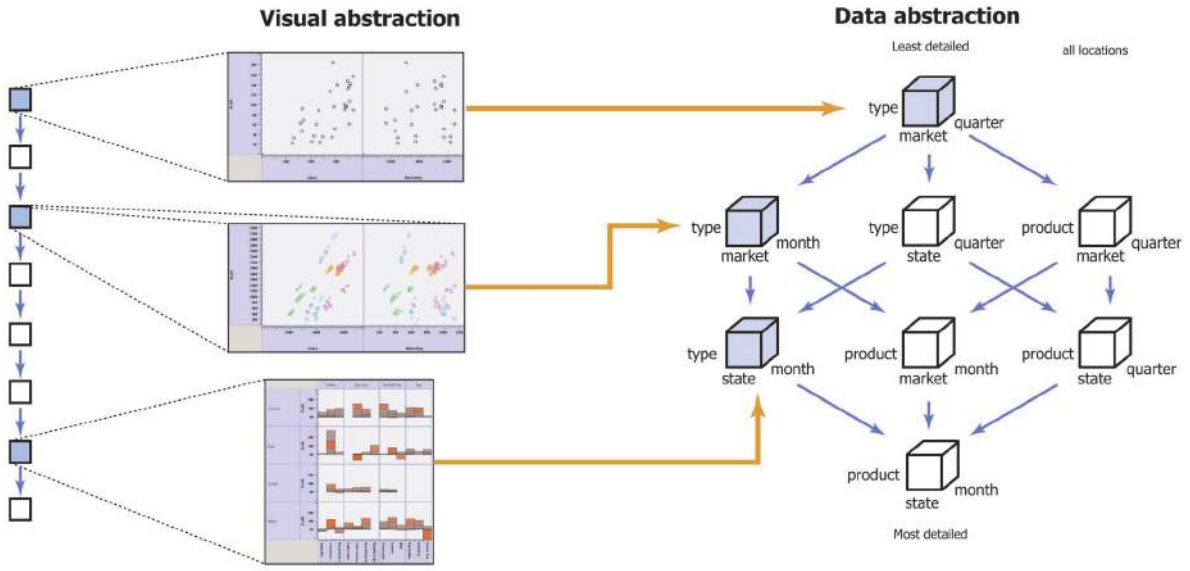


Figura 4 – Hierarquia de especificações visuais em um Zoom Graph, onde cada visualização corresponde à especificação visual de um cubo de dados do sistema Polaris. A Zoom Graph contém um grafo hierárquico com as transições de visualização possíveis a partir da visualização atual, permitindo alterar a resolução da escala de uma dimensão ou mudar a dimensão visualizada. Adaptado de Stolte, Tang e Hanrahan (2003).

maneira é possível obter um processo de visualização geral, com detalhamento sob demanda a partir da transição para especificações com mais atributos aninhados.

Durante o desenvolvimento desse sistema, Stolte, Tang e Hanrahan (2003) identificam quatro diferentes padrões de projeto de visualizações exploratórias, demonstrando a utilização da visualização multi-escala sobre diferentes representações gráficas:

- **Gráficos de área empilhados sobre dimensões ordinais**: que associam agregações de dados quantitativos à dimensões ordinais, como tráfego de rede por diferentes janelas de tempo (por hora, diário, semanal, ...);
- **Mapas temáticos**: que associam medidas de uma dimensão à localização geográfica correspondente, permitindo a comparação das regiões pela dimensão em questão;
- **Gráficos de dispersão sobre dimensões quantitativas**: esse tipo de gráfico é útil para analisar o comportamento dos dados ao longo de seções no espaço amostral, como o comportamento de uma função ao longo de um plano;
- **Matrizes**: que em *layout* tabular, facilitam a comparação direta de células de dados próximas. Mudando-se a escala, células podem ser detalhadas em sub-matrizes menores.

O sistema Polaris estendido pela visualização multi-escala tem a capacidade de modelar esquemas complexos, com a grande vantagem de poder se trabalhar com múltiplas hierarquias que podem ser comparativamente exploradas, inclusive em níveis de detalhes diferentes. Contudo, essa técnica acaba por restringir a interação com o usuário, porque é preciso seguir um caminho de exploração previamente definido na Zoom Graph.

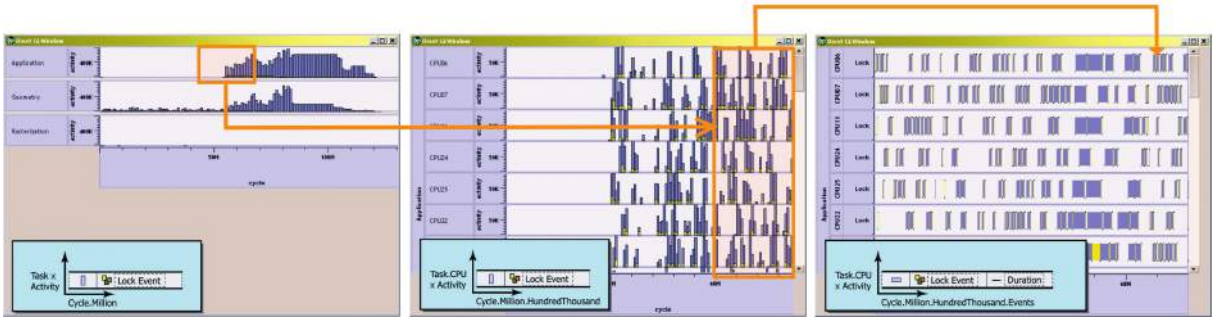


Figura 5 – Exemplo de um dos padrões de visualização multi-escala, com a visualização dos tempos de CPU de vários processos, utilizando gráficos de área empilhados. Cada tela apresenta uma visualização, e as setas indicam as regiões que são detalhadas nas visualizações subsequentes. Adaptado de Stolte, Tang e Hanrahan (2003).

2.2.2 *Cube Presentation Model*

Em uma linha diferente, Maniatis et al. (2003a) desenvolve uma nova abstração visual de cubos de dados, propondo a disposição dos componentes visuais em uma camada de apresentação separada da camada lógica dos dados. Esse modelo é denominado *Cube Presentation Model* (CPM), e provê a recuperação de dados provenientes de uma tabela de fatos agregada em um cubo de dados, de acordo com os valores desejados para os atributos que serão apresentados.

A primeira parte desse modelo é a camada *lógica*, que define a formulação do cubo de dados sob análise. Nessa camada, são definidos quatro principais elementos:

- Uma **dimensão** é definida como um látice de *níveis*, como por exemplo $\{L_0 = cidade,\ L_1 = regiao,\ L_2 = pais\}$;
- **Funções ancestrais**, que mapeiam os valores entre níveis diferentes de uma mesma dimensão. Por exemplo, uma função $anc^{regiao}_{cidade} : L_0 \rightarrow L_1$ mapeia valores entre os níveis cidade e região, ou seja, mapeia cada uma das cidades à região que a contém;
- **Conjuntos de dados detalhados**, que são um modelo das tabelas de fatos, trazendo os dados com a menor granularidade disponível;
- **Cubos**, que representam as agregações sobre os dados detalhados, e portanto contém os elementos citados acima.

A segunda camada do modelo CPM é a camada de apresentação, que manipula os cubos e seus elementos definidos da camada lógica para a apresentação em uma tela 2D. Portanto, as entidades definidas nessa camada são projetadas para serem independentes da camada lógica, sendo elas:

- **Ponto**: lembra o conceito matemático de ponto sobre um eixo, mas que nesse caso representa uma seleção de igualdade sobre um nível, por exemplo $cidade = RioBranco$;

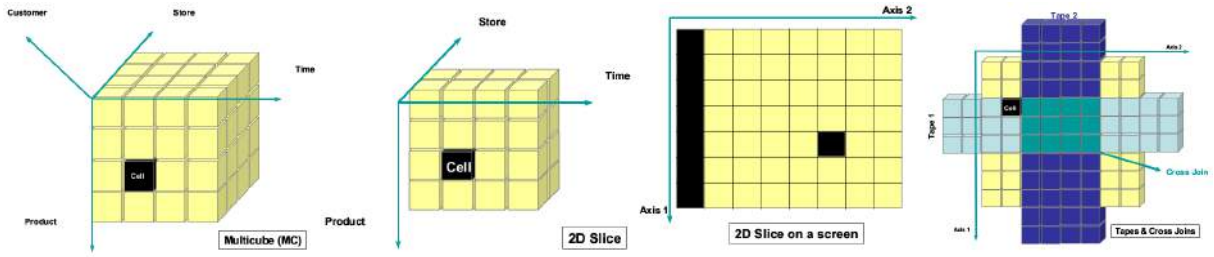


Figura 6 – Elementos do *Cube Presentation Model*: (a) Multicubo, (b) e (c) Fatia 2D extraída do cubo para a tela e (d) Fitas (azul escuro) com cross-join (azul claro) entre si. Fonte: Maniatis et al. (2003b).

- **Eixo**: que pode ser definido como um conjunto de pontos, nesse caso associado a uma dimensão;

- **Multicubo**: que é definido sobre (a) um espaço multidimensional formado por um conjunto de eixos; (b) o conjunto de dados a ser trabalhado com agregações, seleções e ordenações, dentre outros processamentos; e (c) o mapeamento entre esse conjunto de dados e os eixos do espaço multidimensional, definindo as computações do seu conteúdo;

- **Fatia 2D**: uma camada bidimensional dos dados, que será apresentada na tela. Tomando um multicubo com k eixos, uma fatia 2D sobre esse multicubo pode ser definida por um conjunto de $k-2$ pontos, de eixos distintos. Cada ponto delimita o espaço de forma que apenas 2 eixos sem pontos associados serão visualizados na tela;

- **Fita**: é uma coluna ou linha de uma fatia 2D. Pode ser definida por $k-1$ pontos de um multicubo, onde $k-2$ desses pontos correspondem à fatia 2D de origem, e 1 ponto marca um eixo dessa fatia, definindo uma reta paralela ao eixo oposto;

- **Cross-join**: é um cruzamento de duas fitas não-paralelas. Nesse caso, dois pontos de eixos distintos de uma fatia 2D podem ser escolhidos, delimitando duas fitas que atendem à restrição e definem o cruzamento;

- **Função de conteúdo**: são as funções que mapeiam valores computados sobre os dados detalhados ao multicubo, e também sobre combinações de valores presentes no multicubo, incluindo ordenações e demais restrições.

A Figura 6 ilustra de maneira intuitiva esses conceitos. Para mostrar como adaptar técnicas de visualização sobre o modelo CPM, Maniatis et al. (2003b) apresenta uma aplicação OLAP que emprega a técnica de *Table Lens* (Figura 7), desenvolvida por Rao e Card (1994). A ideia consiste em escolher fitas e cross-joins como foco para uma visualização em *Table Lens*, permitindo a comparação de várias janelas sob foco, como no exemplo da Figura 8.

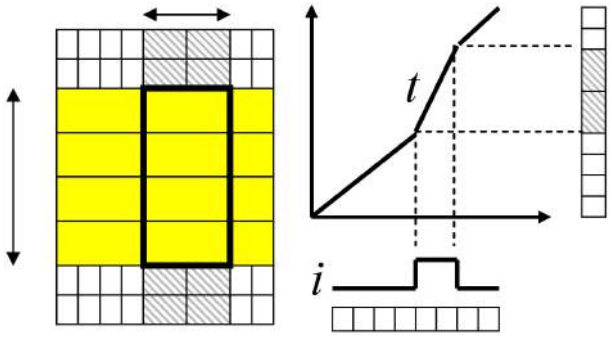


Figura 7 – Exemplo de *Table Lens*: o foco é definido em uma janela de 2×4 células do espaço original 8×8. Tomando por exemplo o eixo x, a função de interesse i atribui valores de interesse para cada célula, sendo um valor maior para as 2 colunas da janela em foco. Uma função de transferência t traduz os valores de interesse de cada célula para o tamanho da área a ser ocupada na tela. Adaptado de Maniatis et al. (2003b).

			C1		C2	C3	C4		C5	C6
			Venk				Netz			
			USA			Japan	USA			Japan
			USA_N		USA_S		USA_N		USA_S	
			Seattle	Boston			Seattle	Boston		
	QTR1	Jan	20	32	62	97	23	40	75	12
R1		Feb	25	40	74	121	18	32	51	20
		Mar	18	12	36	110	42	48	65	3
R2	QRT2		56	63	150	253	50	70	280	50
R3	QTR3		52	65	147	200	53	64	270	50
	QTR4	Oct	25	24	64	98	32	12	64	76
R4		Nov	28	28	76	102	40	21	83	69
		Dec	23	30	68	150	42	29	99	77

Figura 8 – Exemplo de *Table Lens* associado ao CPM: as cores ressaltam as fitas e cross-joins, as linhas mais grossas ressaltam as células com valores mais próximos da média, e dos extremos. Fonte: Maniatis et al. (2003b).

2.2.3 *Hierarchical Dynamic Dimensional Visualization*

Da mesma forma, Techapichetvanich e Datta (2005) introduzem o *Hierarchical Dynamic Dimensional Visualization* (HDDV) para explorar dados hierarquicamente estruturados a partir de cubos de dados. Essa técnica de visualização é incorporada em uma ferramenta de consultas OLAP, facilitando a interpretação e análise das informações extraídas do cubo de dados por operações de *slicing*, *drill-down* ou *roll-up*.

A HDDV opera mapeando dimensões do cubo para os níveis de uma árvore de exploração cuja visualização expõe uma série de operações de *slicing* e *drill-down* sobre o cubo de dados. Nessa árvore, cada nível é representado por uma barra (*barstick*) de comprimento fixo, dividida em seções para indicar a distribuição dos valores da dimensão correspondente.

Assim, barras de atributos quantitativos são divididas em faixas de valores, e atributos nominais são divididos em categorias com seus respectivos valores. Essa divisão dos atributos nominais usualmente leva a divisões desbalanceadas, mas que são úteis para se observar a representatividade (em termos de número de itens) de cada categoria, e assim permitir que o analista possa aferir intuitivamente a distribuição desses dados.

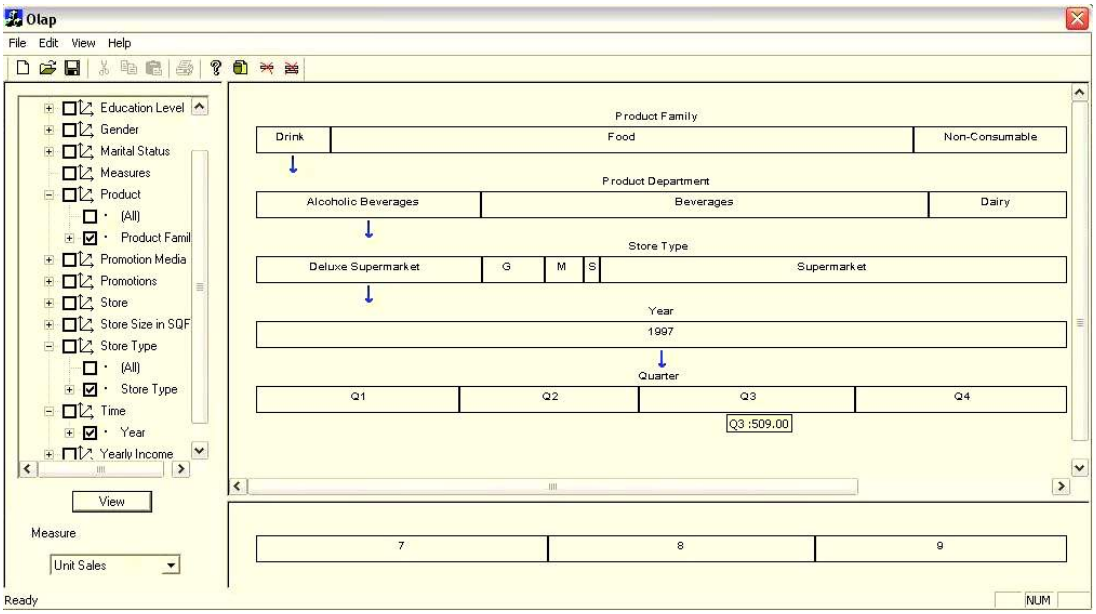


Figura 9 – *Barsticks* representando dimensões na HDDV. As dimensões selecionadas na hierarquia à esquerda são aninhadas no painel principal, selecionando os itens cuja família é "Drink", departamento de "Bebidas alcoólicas", loja tipo "Deluxe", no ano de 1997. A etiqueta de dica indica 509 vendas de itens com esses atributos no terceiro trimestre. Adaptado de Techapichetvanich e Datta (2005).

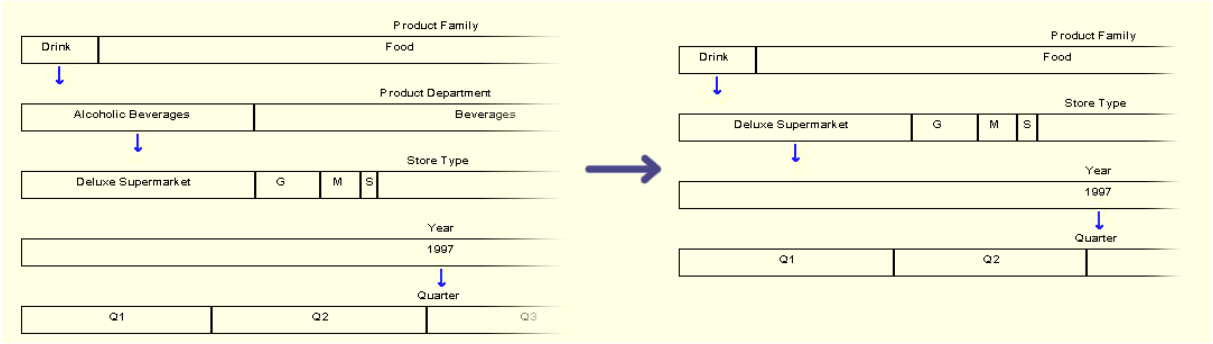


Figura 10 – Exemplo de *roll-up* na HDDV, onde o nível de "Product Department" é removido da árvore, reconstruindo a consulta com os níveis restantes. As barras subsequentes que dividiam apenas bebidas alcoólicas são recalculadas para refletir a seleção de todos os itens da família "Drink". Adaptado de Techapichetvanich e Datta (2005).

A Figura 9 ilustra essas características com um exemplo de uma aplicação OLAP com visualização HDDV, exibindo dados de um cubo das vendas de produtos de uma rede de supermercados. O usuário pode descer interativamente na hierarquia de dimensões, adicionando um novo nível aos já selecionados (o quê equivale a uma operação de *drill-down* no cubo de dados), ou remover níveis internos da árvore (equivalente a um *roll-up*), reaplicando as seleções subsequentes ao nível imediatamente superior (Figura 10). Nesse sistema, essa navegação constrói as consultas ao cubo de dados implicitamente.

Esse método permite ao usuário construir visualmente consultas sobre cubos de dados, e provê uma visualização hierárquica para a sequência de níveis que foram selecionados. Com essa visualização, é possível alternar facilmente entre diferentes ramos na hierarquia, e assim

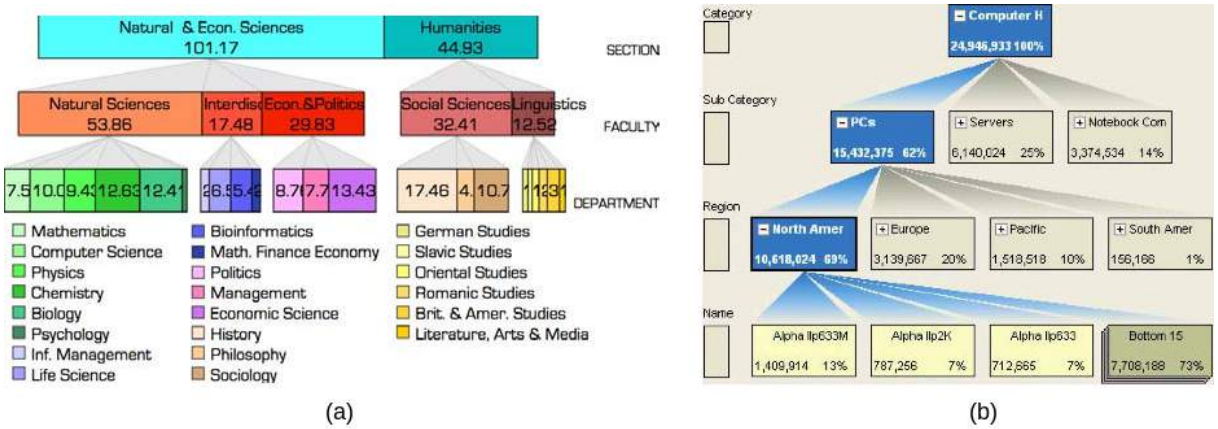


Figura 11 – Árvore de decomposição com nós representados em barras (a) e caixas (b). A visualização (a) mostra uma árvore onde cada nível é decomposto por inteiro, e expandido aos níveis subsequentes. Em (b) apenas um nó é decomposto por vez, trazendo uma ideia semelhante à do método HDDV. Adaptado de Mansmann e Scholl (2007).

executar observações comparativas sobre diferentes grupos de itens. Contudo, a exploração definida nesta técnica permite a visualização de somente um ramo da árvore por vez.

2.2.4 *Enhanced Decomposition Tree*

Com uma motivação similar à do método HDDV, Mansmann e Scholl (2007) propõem outro modelo de visualizações sobre consultas OLAP. Ao invés de definir uma única árvore de exploração como o HDDV, a metodologia desses autores utiliza de múltiplas hierarquias de exploração (com a *Enhanced Decomposition Tree* – EDT), que combina uma camada lógica baseada no *Cube Presentation Model* de Maniatis et al. (2003a) de maneira hierárquica, e uma camada de apresentação com diferentes técnicas de visualização de preenchimento de espaço (com barras, caixas, e gráficos).

Nesse esquema uma árvore de decomposição representa as desagregações que particionam os dados provenientes do cubo, formando o caminho de exploração para os dados em níveis mais detalhados. Essa árvore pode ter sub-árvores compostas por diferentes atributos, que podem ser desenhadas em paralelo em uma mesma visualização. Com isso, é resolvida a principal limitação existente no método HDDV, permitindo a comparação de diferentes ramos da árvore de exploração na mesma tela, minimizando o esforço cognitivo entre as transições de contexto como acontece com a aplicação do método HDDV. Um exemplo dessa diferença é ilustrado pela Figura 11.

Esse trabalho também traz uma importante consideração sobre a navegação do usuário sobre as dimensões do cubo: ela pode ser baseada na *extensão* das dimensões, isto é, nos grupos ou classes de valores que as dimensões assumem, e posteriormente sendo expandidas para exibir as próximas dimensões da hierarquia; ou baseada na *intenção* da dimensão, ou seja, evidenciando a estrutura do esquema da base de dados, com as entidades e atributos correspondentes à dimensão, os quais podem ser particionados sob demanda.

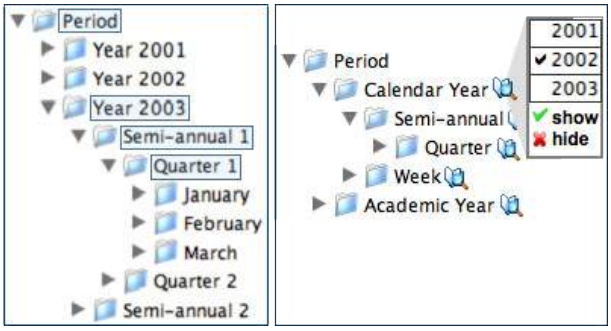


Figura 12 – Hierarquia de um cubo por extensão (esquerda) e intenção (direita). A hierarquia por extensão tem foco em detalhar os valores das dimensões do cubo de dados, sendo a abordagem usualmente utilizada nessa área. A hierarquia por intenção busca mostrar as informações de maneira mais próxima do esquema do banco de dados, focando em exibir os atributos disponíveis para a construção da árvore de decomposição. Fonte: Mansmann e Scholl (2007).

A abordagem por intenção é um ponto chave para suportar hierarquias heterogêneas, pois permite a construção de uma hierarquia cujos nós não precisam selecionar uma extensão específica, como acontece com a HDDV. A Figura 12 ilustra essas duas abordagens, onde é possível notar uma hierarquia mais compacta na navegação por intenção, sem a perda da capacidade de especificar extensões escolhidas para a visualização.

2.2.5 *Zoom Tree*

Ainda nessa linha de trabalho, Wang et al. (2011) apresentam um sistema de visualização cliente-servidor chamado *Zoom Tree*. Seu sistema permite um esquema de navegação com base em cubo de dados semelhante ao esquema de Mansmann e Scholl (2007): a *Zoom Tree* representa a árvore de exploração, cujas transições entre os nós são as operações de *drill-down* no cubo de dados correspondente, até o último nível de maior detalhe.

O sistema apresentado nesse trabalho é inspirado no Polaris (STOLTE; TANG; HANRAHAN, 2002), e pode ser entendido como uma evolução incremental com adição das várias ideias sobre visualização hierárquica em cubos de dados. O fluxo de interação principal se assemelha ao do Polaris, onde os dados são primeiramente selecionados e transformados para então seguirem à visualização em um *layout* tabular (Figura 13). Porém, nesse sistema a navegação do usuário (detalhamento das células da tabela a cada nível) é feita pela abordagem de intenção (Figura 14), e é armazenado na *Zoom Tree*, formando assim uma árvore como a EDT (MANSMANN; SCHOLL, 2007).

Duas características desse sistema o diferenciam da abordagem EDT. Primeiro, ao particionar os níveis da árvore, as dimensões com muitos valores são apresentadas de acordo com partições organizadas por faixas de valores, de maneira similar ao que ocorre com atributos quantitativos no sistema de Techapichetvanich e Datta (2005). A segunda característica marcante é que ao contrário da abordagem com EDT, o usuário não tem uma visão geral da *Zoom Tree* com a qual trabalha. O detalhamento (*drill-down*) é feito sob demanda para cada célula do *layout*,

Figura 13 – Visualização de *Zoom Trees* inspirada no sistema Polaris, com visualizações contidas em células de uma grade tabular. Fonte: Wang et al. (2011).

Figura 14 – Seleção de atributos na *Zoom Tree*. Fonte: Wang et al. (2011).

abrindo janelas sucessivas com os níveis de detalhe escolhidos, perfazendo uma visualização hierárquica (Figura 15).

2.3 Visualização hierárquica em grafos

De acordo com a literatura, a visualização hierárquica se mostra uma ideia poderosa para para consultar e estudar dados modelados em cubos de dados. No entanto, a ideia de visualização hierárquica abrange um escopo ainda maior, sendo aplicada aos mais diversos tipos de dados, como textos, séries temporais e grafos. Dentre esses, a visualização de grafos traz uma enorme gama de ferramentas, como descrito por Landesberger et al. (2011).

Dado que um banco de dados relacional pode ser facilmente modelado como um grafo onde as entidades são vértices e os relacionamentos são arestas, uma ideia interessante seria aplicar técnicas de visualização hierárquica ao grafo de um banco de dados relacional. Ao contrário do

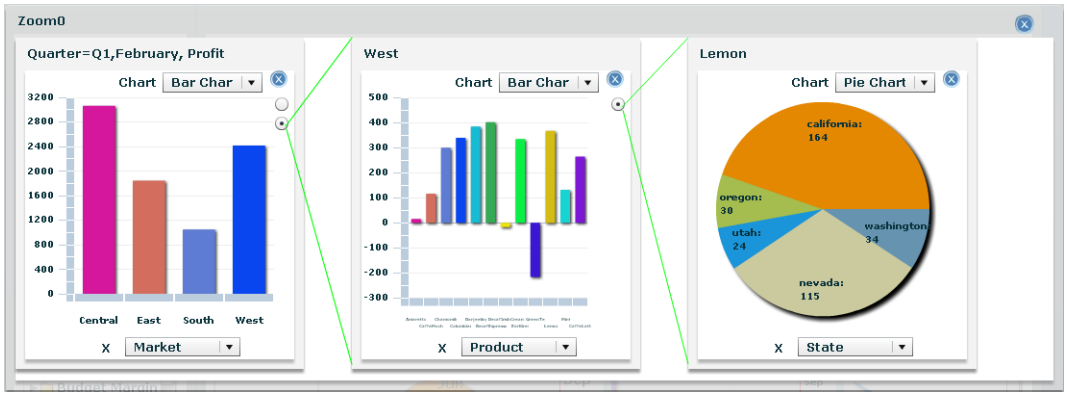


Figura 15 – Detalhamento de uma célula na *Zoom Tree*, ocorrendo em uma janela separada da visualização principal (grade tabular). Fonte: Wang et al. (2011).

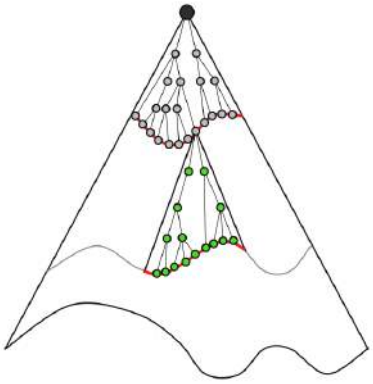


Figura 16 – A anticadeia superior (cinza) é antecessora da anticadeia inferior (verde), pois todos os nós da segunda são descendentes de nós da primeira, e nenhum nó da primeira é descendente de qualquer nó segunda. Adaptado de Abello, Ham e Krishnan (2006).

cubo de dados, essa abordagem preserva os relacionamentos entre as entidades, tendo potencial para disponibilizar novos tipos de informações e tarefas analíticas.

Um dos principais estudos na visualização hierárquica de grafos é apresentado com o sistema ASK-GraphView, proposto por Abello, Ham e Krishnan (2006). Nesse sistema os autores desenvolvem uma estrutura de dados hierárquica para o particionamento e armazenamento de grafos, e uma metodologia em pipeline para processamento, agrupamento, e visualização de grafos e sub-componentes.

A estrutura de dados nesse trabalho baseia-se em transformar o grafo em uma árvore hierárquica, formada por estruturas menores chamadas de *anticadeias*. Uma anticadeia é um conjunto de nós da árvore hierárquica de forma que quaisquer dois nós nesse conjunto nunca sejam antecessores um do outro, ou seja, uma anticadeia é uma sub-árvore da árvore hierárquica. Dada duas anticadeias disjuntas de uma mesma árvore hierárquica, A_1 e A_2, diz-se que A_1 é antecessor de A_2 se todos os nós de A_2 forem descendentes de pelo menos um nó de A_1, e nenhum nó de A_1 seja descendente de qualquer nó de A_2. A Figura 16 ilustra esse conceito.

Assim, é possível estabelecer que cada anticadeia armazena um grupo de nós do grafo original, e são organizadas hierarquicamente na árvore. Outro detalhe é que os níveis da

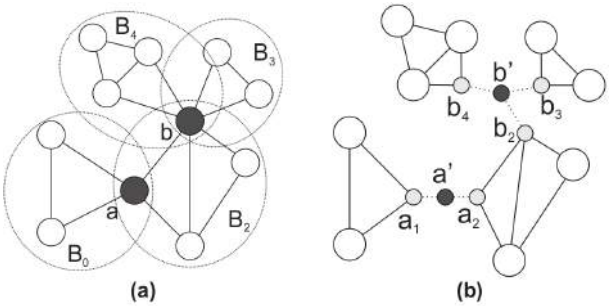


Figura 17 – Separação de componentes biconexos. Em (a) são identificados os componentes e os vértices que os articulam, e então (b) é criada uma cópia da articulação para cada componente que a contém, separando-os. Fonte: Abello, Ham e Krishnan (2006).

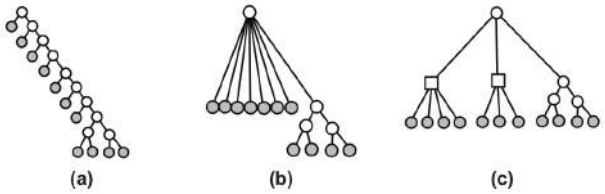


Figura 18 – Correção de ramos degenerados (a) por meio de uma retração (b), balanceando novamente a árvore (c). Os nós que balanceiam a árvore (quadrados) não contam como vértices do grafo. Fonte: Abello, Ham e Krishnan (2006).

hierarquia podem ser associados a níveis de detalhe do grafo, correspondendo a visões similares a agregações de dados, mas sem a perda das informações de relacionamento entre os nós. Com essa configuração, essa estrutura hierárquica permite a pronta aplicação de várias técnicas de visualização hierárquica.

No entanto, ainda é necessário escolher quais os grupos de nós que serão reunidos em cada nível. Como esse sistema almeja tratar grafos de qualquer natureza, o método utilizado para criar a hierarquia baseia-se apenas na estrutura topológica do grafo. Primeiro identificam-se os componentes biconexos do grafo, que são separados no vértice de articulação (Figura 17). Cada componente é então atribuído a uma anticadeia, na ordem em que são divididos. Esse processo pode gerar anticadeias com ramos degenerados, ou seja, longas cadeias de nós com apenas um filho, o que é corrigido no passo a seguir por meio da retração balanceada desses ramos (Figura 18).

A partir dessa hierarquia, o sistema melhora o agrupamento aplicando um método mais robusto e então envia o resultado para uma visualização hierárquica de grafo, concluindo o processo (Figura 19). Esse trabalho descreve uma estrutura em grafo muito semelhante ao conceito de SuperGrafos desenvolvido por Rodrigues et al. (2013), a ser explorado na metodologia deste trabalho. Embora os conceitos de particionamento hierárquico em anticadeias tenha uma formulação próxima à definição de SuperGrafos, diferentemente, a proposta do presente trabalho emprega a informação estrutural do banco de dados para particionar o grafo e construir a hierarquia, e não apenas a estrutura topológica do grafo em questão.

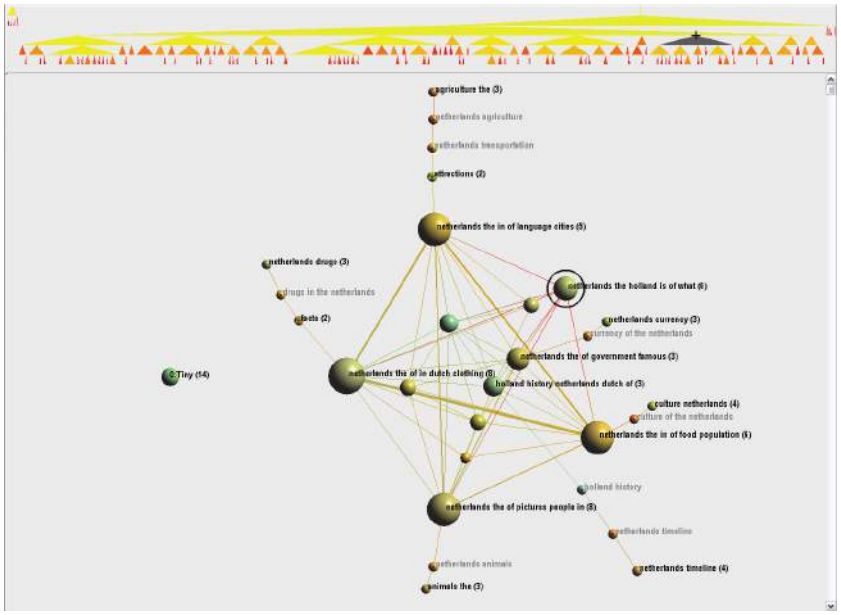


Figura 19 – Visualização de grafo no sistema ASK-GraphView. Adaptado de Abello, Ham e Krishnan (2006).

2.4 Conclusão do capítulo

Este capítulo trouxe uma revisão dos trabalhos mais representativos em análise de bancos de dados relacionais com técnicas de visualização hierárquica, e também em técnicas de visualização hierárquica de grafos sem a estrutura relacional. A primeira linha de trabalhos baseia-se na estrutura tabular do modelo relacional, modelando a base de dados como um cubo de dados.

Nessa linha, uma das grandes preocupações é a escalabilidade, tanto em termos do tempo de resposta causado pelo processamento, quanto da sobrecarga visual ao trabalhar com bases de dados grandes. Tal preocupação pede por sistemas visuais interativos, como os que foram desenvolvidos nesses estudos, e assim proporciona um ambiente bem estabelecido para a análise exploratória de bancos de dados relacionais.

No entanto, esses estudos estão ligados a análises quantitativas, exigindo do analista conhecer os atributos e operações certas a fim de compor as visualizações. E mais, tais sistemas oferecem pouco ou nenhum apoio para explorar os relacionamentos entre as entidades, pois descartam a informação relacional em favor da análise das agregações de transações, que é justamente o foco da metáfora de cubo de dados.

Em uma linha diferente, os trabalhos em visualizações hierárquicas sobre grafos têm a capacidade de trabalhar com esses relacionamentos, caso eles sejam embutidos no grafo em estudo. Entretanto, devido ao fato de que trabalhos analíticos sobre grafos focam em domínios de dados que definem redes, há uma lacuna no que se refere a este potencial no universo de dados relacionais.

De maneira inovadora, a metodologia descrita neste trabalho não se concentra em transações, mas sobre as múltiplas relações que surgem a partir da estrutura de bases de dados operacionais. Tais estruturas podem ser modeladas como múltiplos grafos derivados de múltiplas relações, o

quê, feito de maneira hierárquica, provê a análise detalhada do que ocorre entre e dentro das relações, como será descrito.

Capítulo

3 Visualização de grafos

Visualizar grandes grafos é uma tarefa que aplica os mesmos princípios da disciplina de visualização de informações (WARE, 2012), sendo que diversos trabalhos estudam formas de mais bem aproveitar o espaço visual e a interação com o usuário. No decorrer deste capítulo, será exposta uma breve retrospectiva das principais técnicas de visualização de árvores e grafos, prestando referências para o design do *layout* visual implementado neste trabalho.

3.1 Layout visual

O problema de desenhar um grafo pode parecer muito simples a princípio, já que é bem intuitivo projetar um algoritmo que desenhe os nós do grafo como algum tipo de símbolo visual, e as arestas entre cada dois nós como linhas interligando-os. Porém, algumas questões surgem quando se empenha nesta tarefa: *Quantos vértices e arestas podem ser facilmente visualizados? É possível desenhar qualquer grafo de modo que não ocorram cruzamentos de arestas? Qual a classe de complexidade dos algoritmos envolvidos? Quanta informação cabe no espaço de exibição?*

Esses questionamentos ilustram que representar certos grafos pode trazer problemas nada triviais, como por exemplo a planaridade, propriedade de um grafo poder ser desenhado em um plano sem cruzamento de arestas. Bases de dados podem gerar grafos muito grandes e com uma estrutura muito complexa, o que é difícil de ser analisado sem um *layout* adequado; mesmo assim, é importante estudar as características e comportamento de tais bases.

Levando em consideração estas características, diversas visualizações foram estudadas e desenvolvidas para contribuir com uma solução, e podem ser divididos nos três grupos de técnicas a seguir (HERMAN; MELANCON; MARSHALL, 2000; LANDESBERGER et al., 2011):

- **Desenho de nós interligados**: tenta desenhar nós e arestas utilizando primitivas gráficas, como a ilustração de uma rede;
- **Preenchimento de espaço**: aproveita ao máximo o espaço existente para a exibição das informações, trabalhando com subdivisão do espaço disponível;

- **Outras abordagens**: combinam múltiplas técnicas em diferentes seções ou níveis de detalhe de uma visualização, ou aplicam ideias totalmente novas, buscando solucionar dificuldades encontradas.

Aspectos estéticos são considerados nos trabalhos, sendo que os seguintes critérios são importantes para uma boa representação (LANDESBERGER et al., 2011):

- **Critérios gerais**, incluindo o melhor uso do espaço disponível, a contextualização dos dados apresentados, e a redução de artefatos ou distorções que atrapalhem a compreensão;
- Para grafos **dinâmicos**, espera-se uma certa estabilidade visual ao relacionar elementos em visões de dois momentos distintos, e uma menor ambiguidade temporal (nós diferentes ocupando a mesma posição em momentos distintos no tempo);
- **Escalabilidade estética** referindo-se a manter a qualidade da representação com o aumento seja em número de vértices, arestas ou mesmo grafos.

Vale ressaltar que tais técnicas e critérios podem contribuir com, mas não são suficientes para garantir uma boa representação visual. Certas características dos dados podem se tornar difíceis de se perceber com algumas técnicas, e cabe ao analista empregar mais de uma visualização para decidir qual melhor se encaixa na análise.

3.2 Desenho de árvores

Dado que uma árvore é um tipo de grafo, estudar as técnicas de visualização de árvores é um passo necessário ao se estudar a visualização de grafos. Embora essas técnicas não sejam aplicáveis a grafos em geral, existem meios interessantes de representar informações importantes do grafo por meio de árvores, criando assim uma importante ferramenta para explorar grandes grafos (HERMAN; MELANCON; MARSHALL, 2000).

3.2.1 Nós interligados

Este *layout* é uma maneira intuitiva e amplamente utilizada pela simplicidade de implementação, mas propõe diversos desafios ao desenhar grafos mais complexos. Entretanto, levando em consideração que árvores são grafos bem simples, esta é uma técnica muito prestativa, pois vários grafos estudados seguem topologias hierárquicas (LESKOVEC; KLEINBERG; FALOUTSOS, 2005), naturalmente representadas como árvores.

Na Figura 20, podem-se observar as abordagens mais simples: (a) H-Tree, onde as arestas são ortogonais; (b) *layout* radial, que desenha a raiz da árvore ao centro, e os nós de cada nível em órbitas concêntricas; e (c) *Ballon View*, organizando cada grupo de nós irmãos em balões, de forma que fiquem mais próximos de si que dos nós pais (HERMAN; MELANCON; MARSHALL, 2000).

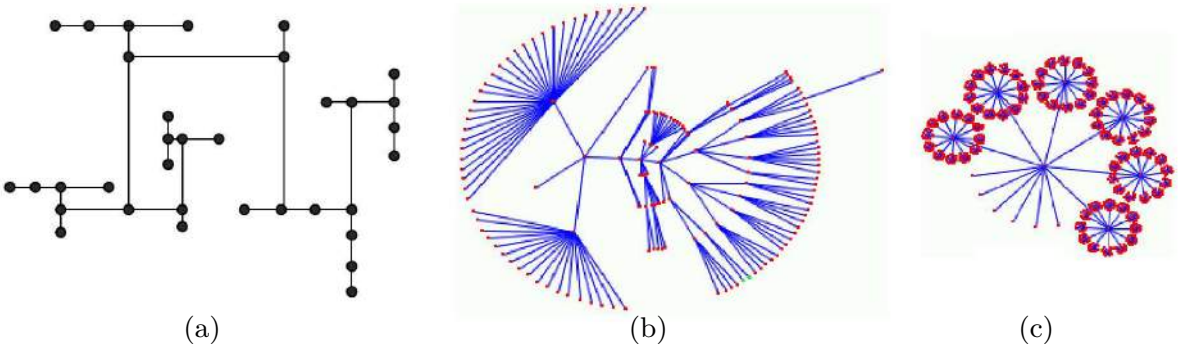


Figura 20 – Abordagens baseadas em nós interligados. (a) H-Tree, (b) Radial, (c) *Ballon View*. Fonte: Herman, Melancon e Marshall (2000).

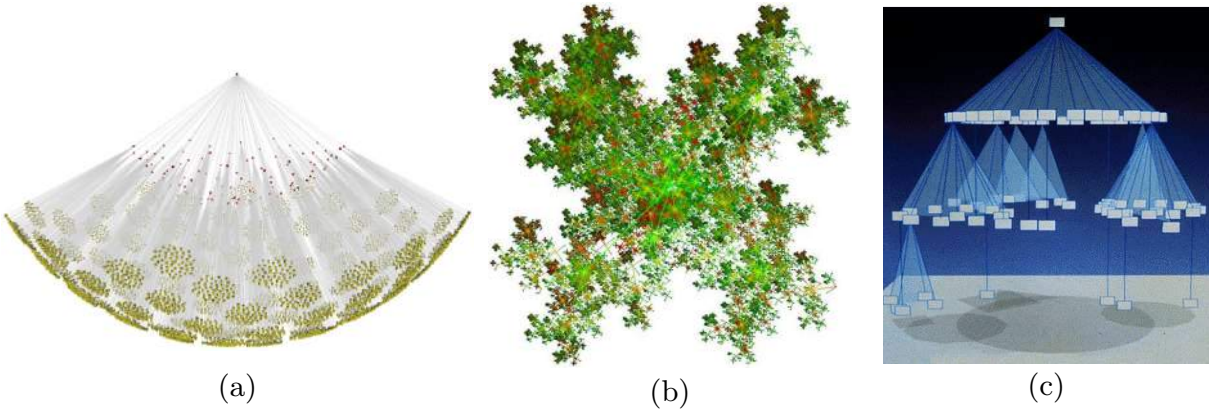


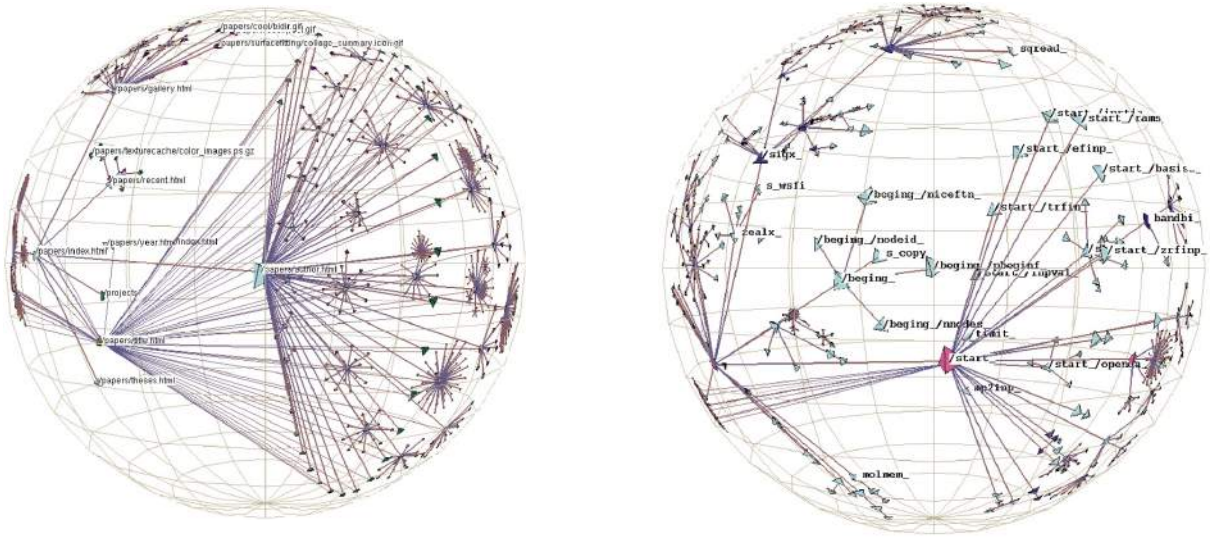
Figura 21 – Outras abordagens baseadas em nós interligados. (a) Phyllotree (NEUMANN; CARPENDALE; AGARAWALA, 2006), (b) Point-based tree (SCHULZ; HADLAK; SCHUMANN, 2009), (c) Cone tree (ROBERTSON; CARD; MACKINLAY, 1991).

Outros formatos são inspirados em padrões naturais, como a Phyllotree (Figura 21a) e a Point-based tree (Figura 21b). Algumas abordagens podem ser retrabalhadas em três dimensões, como a Cone Tree da Figura 21c, cuja projeção paralela às bases dos cones lembra uma*Ballon View*. O sistema H3 (MUNZNER, 1998) também contribui com estas abordagens, criando uma projeção da árvore em um espaço hiperbólico (Figura 22).

Com esta técnica, espera-se obter uma interação mais rica e com maior quantidade de informação. Entretanto, os problemas de oclusão e projeção trazidos por essa abordagem podem dificultar alguns aspectos da visualização, como a perda do contexto, ou dificuldade em visualizar e comparar vários nós em diferentes posições.

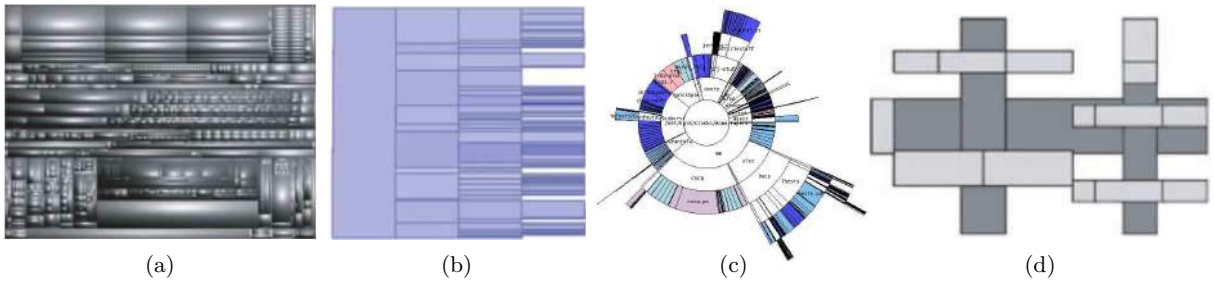
3.2.2 Preenchimento de espaço

Esta abordagem busca utilizar toda a área disponível para representar os elementos, podendo codificar dados por meio das áreas de seus componentes, dessa maneira conseguindo inserir mais informações em uma única visualização, ao custo de uma maior carga perceptual. Pode ser dividida em três categorias principais (LANDESBERGER et al., 2011):



(a) (b)

Figura 22 – Projeções em um espaço hiperbólico. (a) Sítio web do *Stanford graphics group*, (b) Estrutura do grafo de chamadas de funções para um *benchmark* científico em Fortran. Fonte: Munzner (1998).



(a) (b) (c) (d)

Figura 23 – Desenho de árvores com preenchimento do espaço. (a) Cushion TreeMap (WIJK; WETERING, 1999), (b) Icicle Plot (TEKUSOVA; SCHRECK, 2008), (c) Sunburst TreeMap (STASKO; ZHANG, 2000), e (d) BeamTree (HAM; WIJK, 2003).

- **Encapsulamento**: cada nó é representado dentro de uma área fixa, e os filhos são postos sobre esta área, dividindo o espaço de maneira recursiva;

- **Adjacência**: o espaço é subdividido de maneira que os nós fiquem próximos de seus pais, mas em áreas separadas, para que não se sobreponham;

- **Cruzamento**: os nós são arranjados como por na técnica por encapsulamento, mas com uma sobreposição apenas parcial, mantendo a noção de proximidade das técnicas de adjacência; com um aproveitamento maior do espaço.

Na Figura 23, podem ser observados alguns exemplos proeminentes destas abordagens: (a) Cushion TreeMap (WIJK; WETERING, 1999), que adiciona uma técnica de *shading* nos nós folha para melhorar a distinção de nós pequenos; (b) Icicle plot (TEKUSOVA; SCHRECK, 2008), com cada coluna representando um nível da árvore; (c) Sunburst TreeMap (STASKO; ZHANG,

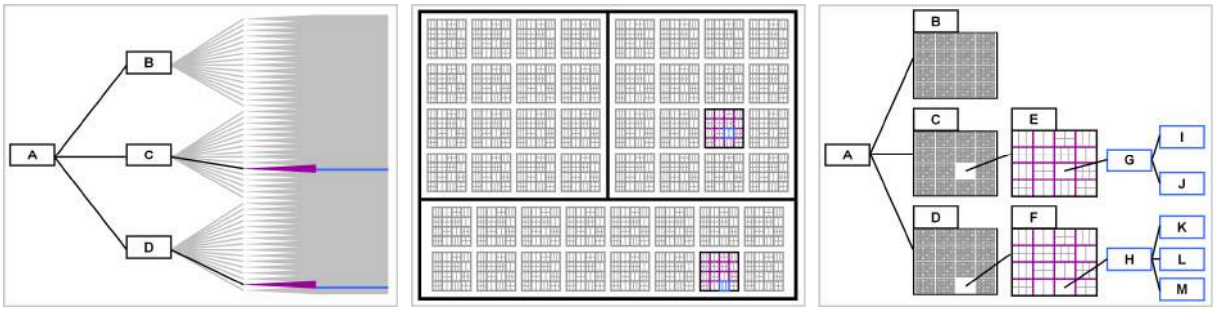


Figura 24 – Comparando as abordagens de desenho de árvores na visualização de elementos de interesse (marcados nas cores azul e magenta), temos as seguintes abordagens: (a) a árvore apresentada em um *layout* de nós e arestas usual; (b) a mesma árvore apresentada em TreeMap; e (c) arranjo híbrido utilizando as duas abordagens anteriores. Nessa abordagem híbrida, o uso de TreeMaps menores apenas nas áreas em foco aproveita melhor a área da tela para mostrar os elementos de interesse. Fonte: Zhao, McGuffin e Chignell (2005).

2000), que emprega um *layout* radial para separar mais os nós folha; e (d) BeamTree (HAM; WIJK, 2003), onde cada grupo de nós filhos é desenhado perpendicularmente sobre o nó pai.

3.2.3 Abordagens híbridas

Analisando as técnicas baseadas em nós interligados, é fácil perceber que a representação visual deixa muito espaço inutilizado, além de se tornar muito sobrecarregada e pouco legível ao se trabalhar com grafos muito grandes (Figura 24a). A utilização de uma abordagem orientada a preenchimento de espaço pode aumentar a quantidade de informações visualizadas, como ocorre com a técnica *TreeMap* apresentada na Figura 24b.

Na Figura 24 é possível perceber a dificuldade de analisar informações em grandes grafos. Em (a) é visível a grande frequência de oclusões, tornando difícil a distinção de várias arestas. Um arranjo em TreeMap (b) consegue uma densidade maior de informações na tela, e deixa facilmente perceptíveis as características codificadas como cor e área. No entanto, dificulta-se a leitura da hierarquia, a qual é um ponto forte da abordagem anterior.

Um desenvolvimento interessante dessas ideias é proposto por Zhao, McGuffin e Chignell (2005), cujo trabalho introduz uma mistura das duas abordagens, representando partes da estrutura com técnicas diferentes. Este método obtém uma maior densidade de informação que a abordagem baseada em nós e arestas, pois exibe em detalhes apenas a parte da topologia da árvore que está marcada para observação, mantendo outros ramos sob TreeMaps mais densas.

3.3 Desenho de grafos

A tarefa de desenhar grafos traz vários desafios e casos particulares devido à grande complexidade encontrada em certos grafos, mas ainda assim as técnicas desenvolvidas seguem as mesmas linhas gerais das abordagens utilizadas para desenhar árvores.

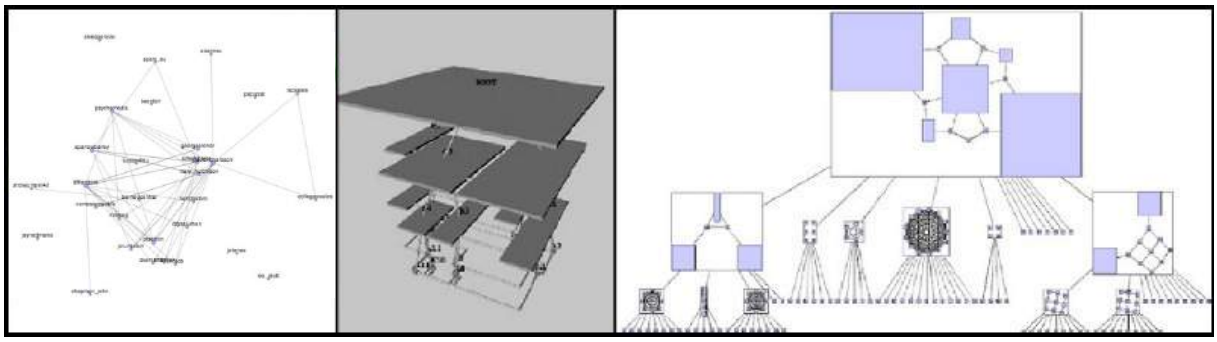
3.3.1 Vértices e arestas

A abordagem orientada a vértices e arestas remete ao mesmo formato aplicado no desenho de árvores, posicionando símbolos gráficos para cada vértice e linhas para as arestas. Algumas convenções comuns incluem:

- Evitar sobreposição de vértices;
- Evitar cruzamento de arestas;
- Manter homogêneo o comprimento das arestas;
- Manter proximidade espacial entre nós topologicamente próximos (vizinhança).

O posicionamento dos vértices, entretanto, raramente é definido, sendo notável exceção os grafos com informações geográficas. Nestes casos onde não existe um posicionamento pré-determinado, diferentes técnicas trabalham ideias para um posicionamento automático veloz e que exponha bem a estrutura do grafo em estudo. Dentre as categorias dessas técnicas, Herman, Melancon e Marshall (2000) e Landesberger et al. (2011) classificam-nas nas seguintes:

- **Dirigido por forças**: simula um sistema físico, com forças de repulsão e atração entre nós e arestas, até chegar a um equilíbrio estático, com o posicionamento dos nós com boas propriedades estéticas. Existem modelos empregando forças gravitacionais, elásticas ou elétricas;
- **Baseado em restrições**: um passo além do *layout* dirigido por forças, este esquema determina restrições de posicionamento entre os elementos, como alinhamento horizontal ou vertical, distância mínima entre elementos (por exemplo, por agrupamento, vizinhança ou similaridade), e não sobreposição de arestas;
- **Abordagem multi-escala**: por meio de uma decomposição hierárquica do grafo em vários sub-grafos menores, é feito o *layout* do subgrafo mais simples, e interativamente cada um dos subgrafos restantes é adicionado à visualização;
- **Layouts em camada**: também conhecido como *layout hierárquico*, esta abordagem divide várias partições do grafo em camadas, e a cada camada adicionada, mais nós e arestas são exibidos;
- **Não convencionais**: algumas técnicas vão de encontro às usuais citadas acima, mas utilizando ideias alternativas, como métodos de agregação, desenho de arestas e disposição baseada em informações semânticas, conseguindo acrescentar excelentes resultados às abordagens anteriores.



(a) (b) (c)

Figura 25 – Visualizações de grafos baseadas em nós e arestas. (a) Dirigido por forças, (b) Em camadas, (c) Multi-escala. Fonte: Landesberger et al. (2011).

Na Figura 25, é possível ver as três abordagens principais. Em (a), um exemplo da abordagem dirigida a força. Em (b), Eades (1999) explora o particionamento hierárquico organizado em árvore, desenhando os diferentes níveis dessa hierarquia em camadas. E em (c), Papadopoulos e Voglis (2006) propõem o desenho dos grafos com base na teoria de decomposição modular, utilizando o particionamento modular, e apresentando em árvore um desenho planar da estrutura hierárquica do grafo.

Comparando estas abordagens, um primeiro atributo preocupante é o desempenho desses métodos. Em geral os métodos com simulações, como o dirigido por força, têm um custo computacional elevado, tornando mais difícil empregá-los em sistemas interativos. Entretanto, estudos sobre a legibilidade dos *layouts* Huang (2007), Pohl, Schmitt e Diehl (2009) revelam que a aplicação de *layouts* dirigidos por força são mais intuitivos e preferidos pelos usuários.

3.3.2 Matrizes de adjacência

Esta abordagem constrói um desenho da matriz de adjacência de um grafo, codificando os atributos das arestas nos elementos da matriz. Esta representação consegue aproveitar bem o espaço da tela, com a grande vantagem de evitar cruzamento de arestas, gerando bons resultados em exibir grafos densos (Figura 26);

Entretanto, embora as matrizes tenham grandes vantagens ao se trabalhar com grandes grafos, pesquisas mostram que usuários não familiarizados podem ter dificuldade com esta abordagem (GHONIEM; FEKETE; CASTAGLIOLA, 2004). Enquanto a leitura de nós adjacentes a um nó pode ser feita percorrendo a coluna (ou linha) correspondente ao nó sob inspeção, encontrar a vizinhança desses nós adjacentes é mais trabalhoso, devido à própria disposição dos elementos.

Outra desvantagem é que a ordem das colunas e linhas pode influenciar bastante o *layout* final e, assim, a facilidade em encontrar grupos e padrões de vizinhança. Reordenar a matriz é um problema NP-Completo, e muito embora existam heurísticas e vários algoritmos para

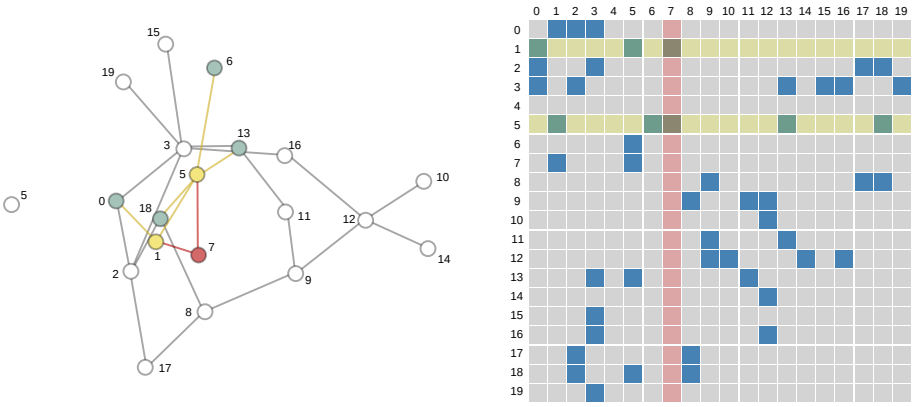
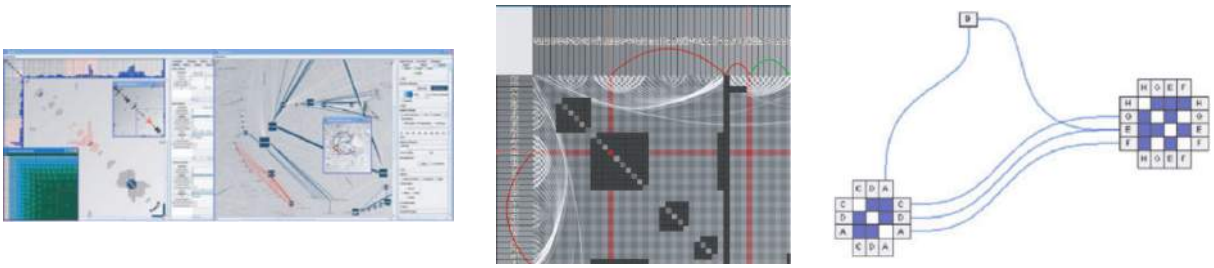


Figura 26 – Comparando as duas abordagens: o mesmo grafo sintético é visualizado com um *layout* de nós e arestas dirigido por forças e com um *layout* de matriz de adjacência. Nas figuras, é ressaltada a vizinhança do vértice de número 7 (vermelho), onde os vizinhos a um salto estão marcados em amarelo, e os vizinhos a dois saltos são marcados em verde. Pode-se notar que visualização a partir de nós e arestas é mais intuitiva na identificação da vizinhança, porém sofre de cruzamento de arestas. A visualização em matriz evita esse problema, mas é mais difícil encontrar elementos de acordo com sua adjacência.



(a) (b) (c)

Figura 27 – Abordagens híbridas para o desenho de grafos. Visões simultâneas (a), matriz com arestas (b) e NodeTrix (c). Fonte: Landesberger et al. (2011).

tal (FEIGE, 2000; DAVIS; HU, 2011), eles dependem bastante da estrutura do grafo em estudo (LANDESBERGER et al., 2011), não apresentando uma solução definitiva.

3.3.3 Abordagens híbridas

Buscando casar a naturalidade da representação de nós e arestas, e a eficiência do uso de matrizes, as abordagens híbridas surgem com três principais propostas: visões simultâneas, matrizes com arestas, e representações parciais, ilustradas na Figura 27.

Na Figura 27a, pode-se visualizar um exemplo com visões simultâneas do mesmo grafo, divididas em duas janelas. As operações de interação são atualizadas sincronizadamente em ambas (HENRY; FEKETE, 2006). A Matlink (HENRY; FEKETE, 2007) propõe a matriz de adjacência com desenho de arestas, exibindo ligações entre nós nas bordas da matriz, realçando linhas ou colunas envolvidas, e assim auxiliando o usuário a seguir caminhos (Figura 27b). Henry, Fekete e McGuffin (2007) mostra a abordagem NodeTrix de representações parciais à visualização de grafos, similar à mesma abordagem para árvores (Figura 27c), onde as áreas densas são representadas por matrizes, minimizando a ocorrência de cruzamentos entre arestas.

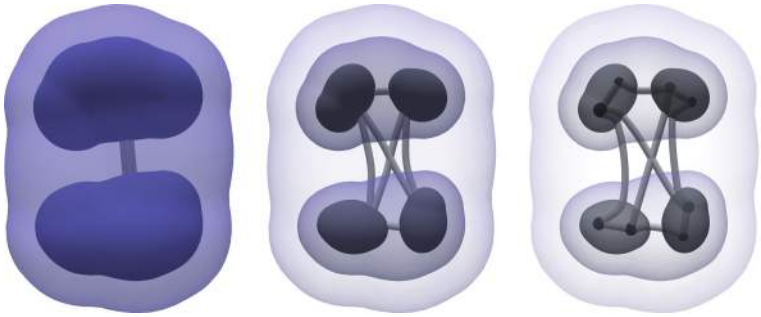


Figura 28 – Visualização em níveis de detalhe de grafos agrupados. Adaptado de Balzer e Deussen (2007).

3.3.4 Outras abordagens

Diferentes abordagens mostram resultados interessantes, porém com métodos menos usuais. Um exemplo, por Harel e Koren (2002), propõe uma projeção de um espaço de alta dimensionalidade para um espaço bi-dimensional (tela), muito eficiente, mas também muito dependente da estrutura do grafo para produzir uma boa visualização.

Como forma de gerenciar o custo associado ao processamento de grandes grafos, algumas aplicações seguem o fluxo de navegação por meio de um *layout* guia, e fornecendo a opção de aplicar outros *layouts* apenas em partes visíveis. Eades e Feng (1997) e Frishman e Tal (2004) mostram exemplos, com metodologias de posicionamento de entidades gráficas baseadas em múltiplos níveis hierárquicos, dinâmica ou estaticamente.

Munzner (1998) trata a sobrecarga visual com a extração de uma árvore geradora, visualizada por meio da técnica H3 (Figura 22). Balzer e Deussen (2007) propõem um sistema para exploração do grafo em um nível de detalhes ajustável, objetivando um arranjo estético favorável à interação (Figura 28). Archambault, Munzner e Auber (2011) propõem um sistema de particionamento hierárquico de grafos, desenvolvido a partir dos estudos anteriores de Abello, Ham e Krishnan (2006), que permite alterar dinamicamente a apresentação do grafo segundo um vértice de interesse. Porém, de acordo com ressalvas do próprio autor, há um custo proibitivo para uso em sistemas interativos.

3.4 Conclusão do capítulo

Neste capítulo foram revisadas classificações com várias abordagens para desenho de grafos e árvores, segundo as tendências da literatura. Percebeu-se que os pontos mais destacados pelos autores são a legibilidade da visualização e a escalabilidade das aplicações, principalmente ao se trabalhar com grandes grafos.

Porém, ao investigá-las um pouco mais, percebe-se que várias dessas abordagens trabalham com representações limitadas do grafo, em prol de obter maior legibilidade e desempenho. Tais representações não possuem as informações de conectividade do grafo em sua totalidade, e embora sejam eficientes para o propósito de visualização, elas limitam ou impedem a aplicação

de outros algoritmos para processamento de grafos, minando a oportunidade de um trabalho analítico mais profundo.

Com essas dificuldades em vista, o presente trabalho apresenta uma proposta promissora, baseando-se em uma estrutura de dados eficiente para atacar esse problema (capítulo 4), utilizando uma combinação de técnicas bem sucedidas para a exploração visual de grafos, e possuindo a possibilidade de ser estendido com técnicas adicionais. Essas características o tornam uma ferramenta com potencial de aplicabilidade para a análise de grafos oriundos de bancos de dados relacionais.

Capítulo

4 Metodologia

Estudando a literatura de análise de bancos de dados relacionais, observam-se vários estudos e aplicações baseadas em técnicas hierárquicas de visualização, notadamente combinadas com a analogia de cubos de dados sobre processamento transacional OLAP. Diferentemente, na metodologia proposta (LIMA; JR.; TRAINA, 2013a; LIMA; JR.; TRAINA, 2013b), inova-se ao se aplicar técnicas hierárquicas sobre dados não transacionais representados como grafos hierarquicamente particionados.

Esta metodologia é descrita nesta seção, incluindo o modelo usado para a representação de dados relacionais como grafos; a engenharia de uma estrutura de dados para armazenamento desse grafo; uma metodologia para o particionamento do grafo utilizando as informações estruturais e os atributos do esquema relacional; e uma metodologia para visualização hierárquica desse grafo particionado, dentro de um protótipo de análise visual.

4.1 SuperGrafos

O método proposto baseia-se em transformar e armazenar os dados em uma estrutura chamada Graph-Tree (RODRIGUES JR. et al., 2006). Esta estrutura pode ser entendida como uma implementação particular do conceito de SuperGrafos (RODRIGUES et al., 2013), uma formalização que abstrai um gráfico hierarquicamente particionado. O SuperGrafo é recursivamente composto de nós, SuperNós (grupos de nós), arestas e SuperArestas (grupos de arestas), formando uma estrutura definida como se segue:

Definição 1: [SuperGrafo] Dado um grafo finito não-direcionado $G = \{V, E\}$, sem laços nem arestas paralelas, um SuperGrafo é definido como $\overline{G} = \{\overline{V}, \overline{E}\}$, onde $\overline{V}$ é o conjunto de SuperNós $\overline{v}$, e $\overline{E}$ é o conjunto de SuperArestas $\overline{e}$.

Definição 2: [SuperNó] Um *SuperNó* $\overline{v}$ é recursivamente definido como um conjunto $\overline{V'}$ de SuperNós ou vértices no grafo (se for um SuperNó folha), mais um conjunto $\overline{E'}$ de SuperArestas

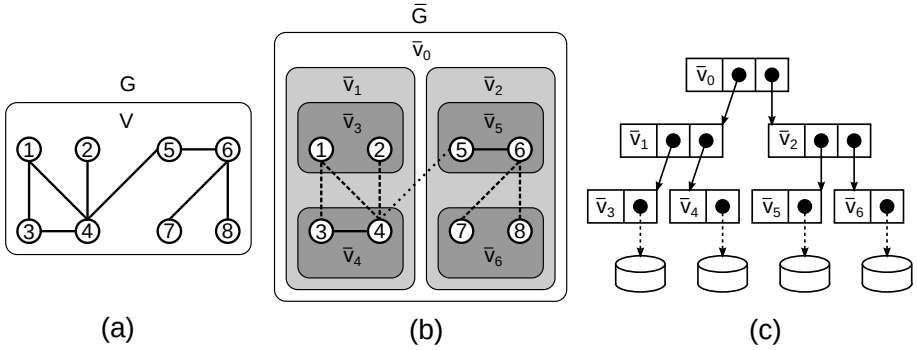


Figura 29 – SuperGrafo obtido a partir de um grafo particionado.

$\overline{e_{ij}}$. Como segue:

$$\begin{aligned}\overline{v} = \{\overline{V'} = &\{\overline{v_0}, \overline{v_1}, ..., \overline{v_{(|\overline{V'}|-1)}}\},\\ &\overline{E'} = \{\overline{e_{ij}} = \{(\overline{v_i}, \overline{v_j}) | \overline{v_i}, \overline{v_j} \subset \overline{V'}\}\}\}.\end{aligned} \tag{4.1}$$

Definição 3: [SuperAresta] Uma SuperAresta representa todas as arestas $(u,v) \in E$ que conectam vértices de um SuperNó $\overline{v_i}$ a vértices de outro SuperNó $\overline{v_j}$. Formalmente, a SuperAresta entre $\overline{v_i}$ e $\overline{v_j}$ é definida como:

$$\begin{aligned}SuperAresta(\overline{v_i}, \overline{v_j}) = \overline{e_{ij}} = \{e = (u,v) | (u,v) \in E,\\ u \in \text{fechamento de } v_i \text{ e } v \in \text{fechamento de } v_j\}.\end{aligned} \tag{4.2}$$

Definição 4: [Peso de uma SuperAresta] O peso de uma SuperAresta é a soma dos pesos de suas arestas.

A Figura 29 exemplifica a abstração SuperGrafo. Na Figura 29(a), pode-se ver o grafo G, definido como $G = \{V = \{1,2,3,4,5,6,7,8\}, E = \{(1,3),(1,4),(2,4),(3,4),(4,5),(5,6),(6,7),(6,8)\}\}$.

A partir do grafo G, é possível conceber a divisão hierárquica apresentada como o SuperGrafo $\overline{G}$ na Figura 29(b). Essa divisão é composta dos SuperNós $\overline{v_0}$ a $\overline{v_6}$ e SuperArestas correspondentes:

$$\begin{aligned}
&\overline{v_0} = \{\overline{V'} = \{\overline{v_1}, \overline{v_2}\}, \overline{E'} = \{\overline{e_{12}} = \{(4,5)\}\}\}\\
&\overline{v_1} = \{\overline{V'} = \{\overline{v_3}, \overline{v_4}\}, \overline{E'} = \{\overline{e_{34}} = \{(1,3),(1,4),(2,4)\}\}\}\\
&\overline{v_2} = \{\overline{V'} = \{\overline{v_5}, \overline{v_6}\}, \overline{E'} = \{\overline{e_{56}} = \{(6,7),(6,8)\}\}\}\\
&\overline{v_3} = \{\overline{V'} = \{1,2\}, \overline{E'} = \{\}\}\\
&\overline{v_4} = \{\overline{V'} = \{3,4\}, \overline{E'} = \{\overline{e_{44}} = \{(3,4)\}\}\}\\
&\overline{v_5} = \{\overline{V'} = \{5,6\}, \overline{E'} = \{\overline{e_{55}} = \{(5,6)\}\}\}\\
&\overline{v_6} = \{\overline{V'} = \{7,8\}, \overline{E'} = \{\}\}.
\end{aligned}$$

A Figura 29(c), por sua vez, apresenta a estrutura da Graph-Tree correspondente, que reflete o particionamento hierárquico do SuperGrafo. Na figura, pode-se ver que a Graph-Tree é

projetada para que os SuperNós folha sejam seletivamente carregados do disco. A principal característica da Graph-Tree é a sua capacidade para determinar dinamicamente as arestas que interligam nós ou SuperNós. Esta característica implica que:

- dado um nó, pode-se determinar todas as arestas que se conectam a este nó sem ter que verificar todas as partições e níveis da hierarquia do grafo;
- dados quaisquer dois SuperNós, pode-se determinar todas as arestas que conectam esses dois grupos de nós.

Estas duas funcionalidades da estrutura são a chave para este trabalho, pois permitem a inspeção dinâmica dos dados estruturais de um banco de dados relacional, permitindo a composição de visualizações interativas que disponham esses dados em paralelo com os dados quantitativos que são agregados às partições do grafo.

4.2 Graph-Tree orientada a disco

Uma das contribuições desta proposta foi o projeto e o desenvolvimento de uma versão aprimorada da estrutura de dados Graph-Tree. Na nova versão projetada, a estrutura utiliza o subsistema de armazenamento em disco da biblioteca Arboretum (http://gbdi.icmc.usp.br/arboretum). Este subsistema fornece uma camada de abstração sobre o sistema de arquivos do sistema operacional, permitindo a utilização de arquivos divididos em páginas (blocos sequenciais), atrelados a políticas de *caching* – é similar à paginação de memória, porém com características próprias (TANEMBAUM, 2007).

À primeira vista, reimplementar o acesso à disco em páginas com *caching* parece ser um esforço desnecessário, pois são serviços comuns fornecidos pelos sistemas operacionais atuais para melhorar a eficiência dos sistemas de arquivos (TANEMBAUM, 2007). No entanto, devido à necessidade de servir a diferentes perfis de aplicações, os sistemas operacionais utilizam configurações adequadas a um propósito geral (armazenar milhares de arquivos diversificados em conteúdo e tamanho). Estas configurações nem sempre são as melhores no contexto de bancos de dados, e por este motivo é importante utilizar uma camada de abstração para configurar estes parâmetros independentemente do sistema de arquivos utilizado.

De acordo com essas considerações, deve-se analisar o caso particular da Graph-Tree. Ao se imaginar uma representação direta dos conjuntos que formam o SuperGrafo da seção anterior (Figura 29), obtém-se uma intuição de como uma Graph-Tree pode ser armazenada em disco. Esta ideia é ilustrada na Figura 30. Para obter acesso eficiente às arestas de conectividade, informações dos vértices cobertos por cada SuperNó são inseridas na árvore (Figura 30b). Assumindo que os vértices sejam ordenáveis, essa informação de cobertura pode ser representada por um elemento com a chave de busca que divide os vértices dentro de cada SuperNó, de modo similar a uma árvore de busca binária.

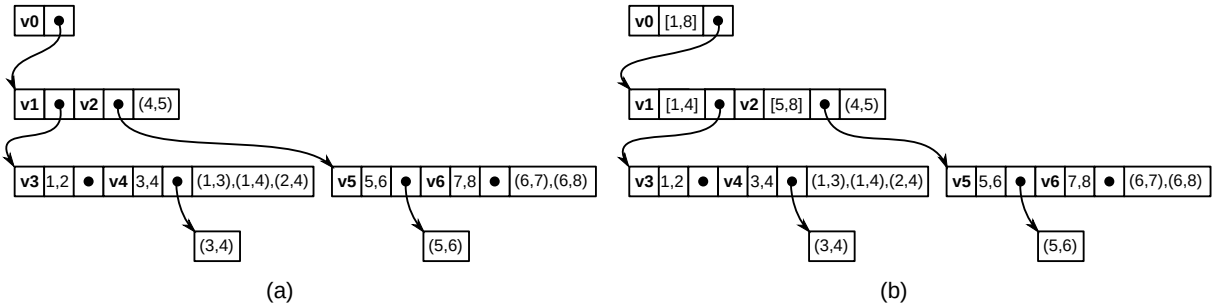


Figura 30 – Estrutura Graph-Tree em disco. (a) Representação dos elementos do SuperGrafo, onde cada nível contém as SuperArestas (como conjuntos de arestas) entre cada par de SuperNós; e os SuperNós folha contém uma lista adicional com os vértices e arestas internas ao subgrafo. (b) Adição de informação de cobertura, neste caso, quais vértices estão contidos no ramo da hierarquia delimitado por cada SuperNó. Por exemplo, o SuperNó $\overline{v}_0$ cobre todos os vértices no intervalo $[1,8]$, enquanto o SuperNó $\overline{v}_2$ cobre os vértices no intervalo $[5,8]$.

Com esta estrutura em mãos, é possível resolver consultas de conectividade entre quaisquer dois SuperNós (computando as arestas uma SuperAresta) ou vértices dessa hierarquia. Exemplificando, dados os dois SuperNós armazenados nos blocos $v4$ e $v5$, uma consulta de conectividade entre esses dois SuperNós é o conjunto de arestas entre os vértices contidos em ambos, neste caso, $\{3,4\} \cup \{5,6\}$. Estas arestas estão armazenadas nos níveis superiores na hierarquia, exatamente no SuperNó cujo conjunto de vértices contém $\{3,4\} \cup \{5,6\}$. Assim, pode-se buscar estas arestas com o algoritmo abaixo:

Algoritmo 1: Conectividade entre SuperNós.

Data: v_a : SuperNo, v_b : SuperNo

Result: $\{e_{ij}\}$: arestas entre vértices cobertos por v_a e v_b, i.e., SuperAresta entre v_a e v_b

$p \leftarrow v_a$ **while** $cobertura(v_b) \not\subset cobertura(p)$ **do**
  $p \leftarrow pai(v_a)$
end
return $\{e_{ij} \in arestas(p) : i \in cobertura(v_a) \wedge j \in cobertura(v_b)\}$

Neste algoritmo, *cobertura* é o intervalo armazenado em cada SuperNó, indicando os nós contidos na hierarquia abaixo do mesmo. Inicialmente, estão carregados em memória os blocos onde residem os SuperNós $v4$ e $v5$. A cada chamada do procedimento *pai*, são lidos os blocos em disco dos respectivos ancestrais – $v1$ e $v2$, que neste caso residem no mesmo bloco $[v1,[1,4],\downarrow,v2,[5,8],\downarrow,\{(4,5)\}]$. Na próxima chamada de *pai*, é lido o bloco de $v0$ (cuja *cobertura* contém os vértices de ambos $v4$ e $v5$), de onde são lidas as arestas correspondentes aos vértices de cada SuperNó, retornando esse conjunto como uma SuperAresta.

No entanto, dado que um procedimento exploratório multi-escala inicia-se em um nível de detalhes geral e progressivamente carrega novos sub-níveis (*drill-down*), esta estrutura pode ser utilizada para carregar e computar os SuperNós e SuperArestas do próximo nível lendo apenas o próximo bloco de disco apontado pelo ponteiro correspondente ao SuperNó inicial. Por exemplo, visualizando o SuperNó $v1$, o usuário deseja um *zoom* sobre os dados e relacionamentos contidos neste SuperNó. As SuperArestas já computadas entre este SuperNó e outros SuperNós também

são subdivididas entre seus filhos, detalhando assim os relacionamentos entre as entidades. Este procedimento é sumarizado como o algoritmo a seguir:

Algoritmo 2: Conectividade dos filhos de um SuperNó.

Data: v : SuperNó
Result: $\{\overline{e}_{ix}\}$: SuperArestas dos filhos v_i de v e quaisquer SuperNós v_x
foreach $v_i \in filhos(v)$ **do**
  foreach $v_j \in filhos(v) : i \neq j$ **do**
    $\overline{e}_{ij} \leftarrow \{e_{ij} \in arestas(v) : i \in cobertura(v_i) \wedge j \in cobertura(v_j)\}$
  end
  foreach $\overline{e}_{vw} \in SuperArestas(v)$ **do**
    $\overline{e}_{iw} \leftarrow \{e_{pq} \in \overline{e}_{vw} : p \in cobertura(v_i)\}$
  end
  $\overline{e}_{ix} \leftarrow \{\overline{e}_{ij}\} \cup \{\overline{e}_{iw}\}$
end
return $\{\overline{e}_{ix}\}$

Como apresentada, a estrutura em disco da Graph-Tree se assemelha muito mais a uma árvore de tuplas (contendo arestas) em um banco de dados do que a uma árvore de diretórios e arquivos. Observando o segundo exemplo de Graph-Tree na Figura 30, é possível notar a semelhança com uma *B-Tree* (CORMEN et al., 2009), onde cada nível da árvore contém uma sequência de nós ordenados divididos em páginas, com ponteiros para páginas de continuação e páginas dos próximos níveis. De igual modo, a Graph-Tree é projetada para ter escalabilidade no acesso a dados armazenados em disco, agrupando os dados necessários a cada consulta de conectividade em uma única página (ou série de páginas contíguas).

4.3 Sistema RMine

O sistema RMine, uma extensão do sistema GMine Rodrigues Jr. et al. (2006), é um protótipo criado para atender à análise de grafos em geral, comportando diversos algoritmos para processamento. Este sistema usa um ambiente visual seguindo um layout multi-nível minimalista baseado em nós e arestas, incluindo operações interativas de pan, zoom, e permitindo a disposição dos nós manual ou automaticamente (influence-based ou force-directed layout). Os nós nessa visualização correspondem a SuperNós nos primeiros níveis da hierarquiva, e podem ser expandidos em subgrafos, carregando e exibindo os nós filhos, em um diferente nível da hierarquia.

O RMine recebe como entrada dois formatos de grafos. Uma delas é um arquivo com uma lista de arestas (um par de nós por linha), utilizada para grafos não-particionados. Outro formato processa grafos particionados em formato METIS, contendo um conjunto de arquivos de arestas correspondentes a cada subgrafo da partição.

Os grafos particionados são carregados em uma instância da estrutura Graph-Tree, enquanto os grafos não-particionados e os subgrafos dos SuperNós folha da Graph-Tree são carregados utilizando-se a biblioteca GLib inclusa na biblioteca SNAP (http://snap.stanford.edu/).

Além destas características, o sistema RMine, protótipo desenvolvido nesta proposta, envolveu os seguintes desenvolvimentos:

- Reestruturação do sistema para suportar informações armazenadas em disco de acordo com o projeto da Graph-Tree orientada a disco, cujas partições são carregadas sob demanda durante as interações de expansão de SuperNós e SuperArestas;
- O *layout* visual foi remodelado com adição de *shading* e uma expansão multi-nível simplificada, diminuindo a quantidade de nós apresentados na visualização e consequentemente o esforço cognitivo necessário para a interação.

4.4 Layout visual

O layout nessa proposta baseia-se em uma visualização multi-nível. Esse design foca no modo de operação da navegação hierárquica, alterando as interações de expansão de SuperNós e zoom.

A operação de expansão do sistema RMine fornece uma opção inspirada no zoom olho de peixe, sendo visualmente semelhante aos diagramas apresentados na Figura 35 (Seção 4.6). Assim, o RMine adiciona opções de interação alternativas durante cada expansão de SuperNó: (a) desenhar o subgrafo sobre toda a área do SuperNó pai, ocultando os SuperNós irmãos, (b) calcular automaticamente as SuperArestas de conectividade para os nós no próximo nível da hierarquia, e (c) ocultar as SuperArestas para SuperNós não selecionados. Para indicar o nível na hierarquia, é atribuído um rótulo ao SuperNó, indicando os ancestrais previamente selecionados na hierarquia. A comparação dessa abordagem e a visualização original do GMine é ilustrada na Figura 31.

Essa figura também mostra outras decisões de design com o objetivo de melhorar a acuidade visual do usuário durante a interação:

- *Shading* no preenchimento de SuperNós, para facilitar a identificação visual dos nós e seu fechamento;
- *Cores* diferentes são usadas para os SuperNós no primeiro nível, com tonalidades progressivamente menos saturadas para os SuperNós descendentes;
- *Transparência* de arestas, podendo codificar o peso da aresta (maior peso, maior opacidade) ou o nível de profundidade na hierarquia (arestas entre SuperNós mais profundos são mais transparentes);

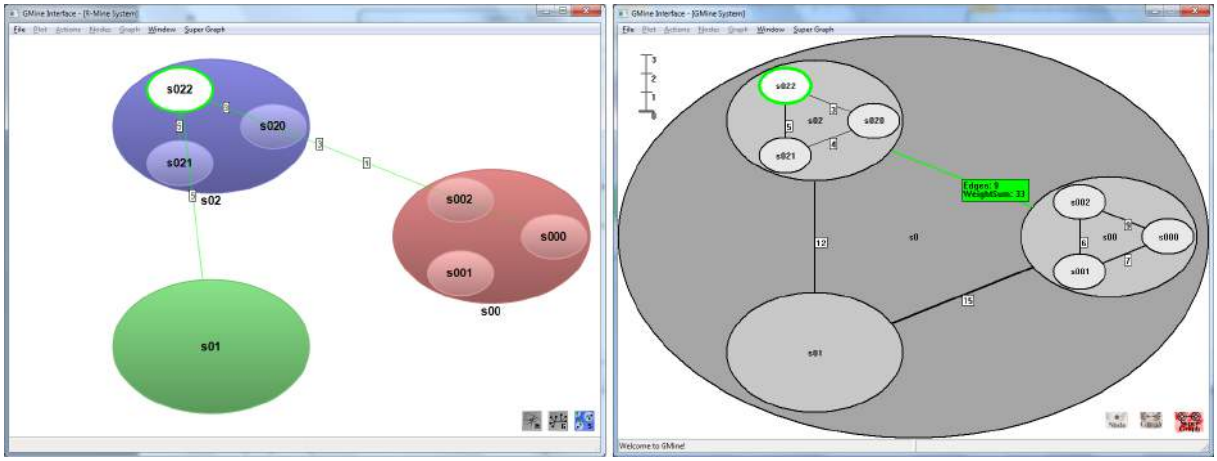


Figura 31 – Layout visual dos sistemas RMine e GMine, onde é possível ver mudanças estéticas no desenho dos elementos, e a apresentação da hierarquia com zoom no SuperNó mais produndo.

- *Espessura* das Super Arestas era originalmente $6\log(peso_{aresta})$, tornando as arestas discerníveis de acordo com a ordem de grandeza do respectivo peso. No RMine, a constante foi transformada em um parâmetro α, resultando na fórmula $\alpha\log(peso_{aresta})$.

Este projeto de layout adiciona ao particionamento relacional (desenvolvido na Seção 4.6) algumas características estéticas desejáveis, as quais facilitam a percepção visual dos relacionamentos entre os grupos de entidades de maneira condensada. Aliada ao esquema de visualização multi-escala, cujos detalhes da hierarquia são solicitados sob demanda, este protótipo traz um layout com uma menor sobrecarga visual em relação à apresentação original do GMine e trabalhos relacionados (BALZER; DEUSSEN, 2007; ARCHAMBAULT; MUNZNER; AUBER, 2011). A grande desvantagem das visualizações desses trabalhos é que à medida que são expandidos mais subgrafos da hierarquia, seus nós são adicionados à visualização; este processo aumenta gradativamente a quantidade de elementos visuais na tela, até o momento em que o trabalho cognitivo se torna demasiado.

4.5 Desenvolvimento

O sistema RMine é desenvolvido em C++ (ISO:98) e utiliza as bibliotecas: wxWidgets para interface gráfica e GLib (contida na SNAP) para estruturas de dados de grafos nos nós folha. O sistema é desenvolvido em Microsoft Visual Studio e compilado nativamente para Windows (XP ou superior, 32 e 64 bits). Durante o desenvolvimento do protótipo RMine, foram criados Makefiles próprios para a compilação em Linux com o compilador GNU GCC (64 bits). O programa é organizado em três grupos de componentes:

- Interface gráfica e desenho de primitivas, que trata de janelas, menus, eventos de entrada e desenho de formas geométricas simples;

- Estruturas de dados de grafos e SuperGrafos, incluindo persistência em disco;

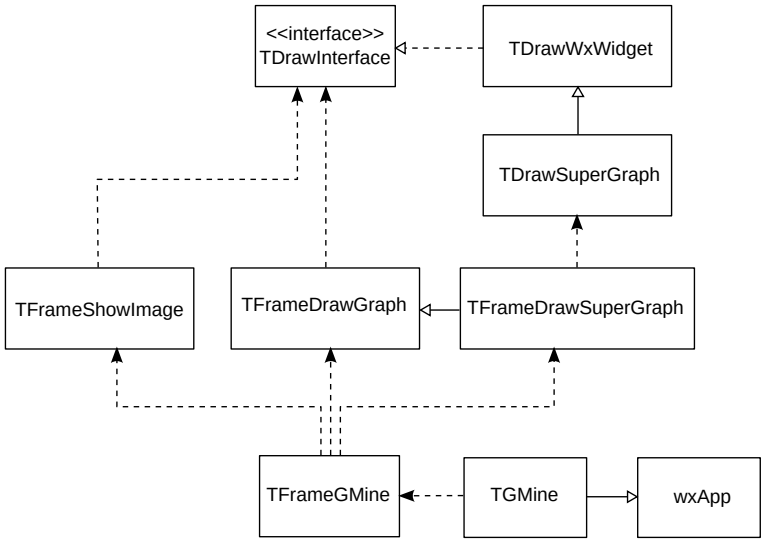


Figura 32 – Diagrama de classes da interface gráfica

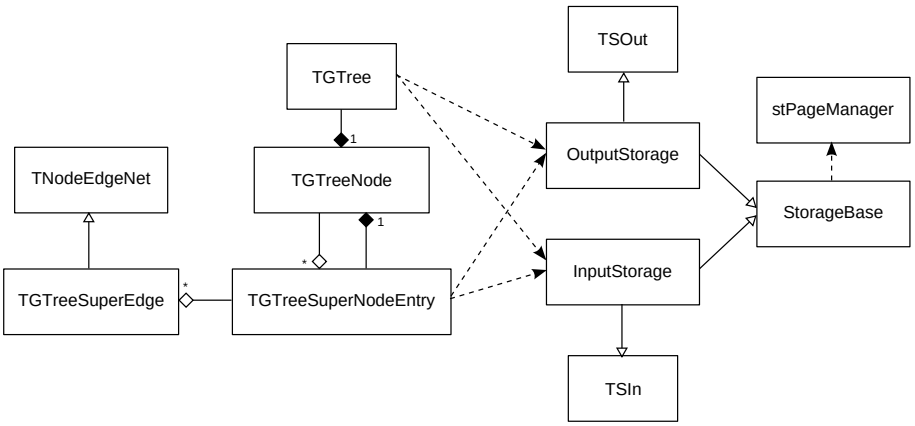


Figura 33 – Diagrama de classes das estruturas de dados

- Funções sobre grafos para particionamento, visualização (layouts), transformações e cálculo de métricas.

Esses componentes são organizados em classes, de acordo com os diagramas a seguir (apenas as classes principais são mostradas nos diagramas). O subsistema de interface gráfica é implementado através de janelas e diálogos do wxWidgets, e para o desenho de grafos utiliza uma interface genérica (TDrawInterface) implementada pela classe TDrawWxWidget (Figura 32). A estrutura de SuperGrafo (Figura 33) é implementada em uma árvore (TGTree) de TGTreeNodes, que são listas de SuperNós (TGTreeSuperNodeEntry), que por sua vez contém as SuperArestas (TGTreeSuperEdge). As SuperArestas contém os dados dos vértices e arestas do subgrafo (correspondente aos pares de SuperNós), e portanto modelados como subclasse da estrutura de grafo da GLib (TNodeEdgeNet).

O desenvolvimento realizado no decorrer do mestrado se concentrou nas classes correspondentes ao desenho de SuperGrafos (TDrawSuperGraph e TFrameDrawSuperGraph), e na estrutura de persistência em disco (StorageBase, InputStorage e OutputStorage). A classe

StorageBase provê uma interface às funcionalidades de acesso a disco da classe stPageManager, permitindo a alocação de páginas independentes para cada nível da hierarquia. As classes InputStorage e OutputStorage implementam uma interface similar à TSIn e TSOut da GLib (SNAP), para reuso dos métodos de persistência da classe TNodeEdgeNet da GLib (que utilizam TSIn e TSOut por padrão, sem paginação).

4.6 Particionamento Relacional

Em trabalhos anteriores, a estrutura de dados Graph-Tree foi usada para processar e interagir visualmente com grafos que foram automaticamente particionados. O problema desta aplicação é que o número h de níveis na hierarquia é determinante na interpretação do particionamento hierárquico do grafo, no entanto não existem algoritmos para automaticamente determinar estes valores de um dado grafo. Para resolver este problema, neste trabalho, define-se o número de níveis hierárquicos como o número de atributos de interesse em uma relação de banco de dados. Como será explicado mais adiante, isto define uma hierarquia semanticamente rica, organizada de acordo com os valores dos atributos encontrados nos dados.

Juntamente com esta abordagem, usa-se a informação dada pelas relações entre as diferentes entidades do banco de dados para se representar os dados de acordo com o modelo de grafo. Dessa maneira, foi possível se produzir grafos hierarquicamente particionados que incorporam as informações de bancos de dados inteiros considerando a semântica dada por seus atributos e a estrutura dada por seus relacionamentos – isto é, referências definidas por chaves estrangeiras do esquema de dados. Esta abordagem é totalmente diferente quando comparada com as técnicas anteriores de visualização de base de dados, as quais centram em valores quantitativos transacionais, desprezando a informação importante representada pela estrutura da base de dados.

Primeiro nível

Seguindo esta linha de pensamento, o primeiro nível da hierarquia é determinado por entidades e relacionamentos muitos-para-muitos. Por exemplo, considere o esquema de banco de dados mostrado na Figura 34 – a entidade Pessoa tem um relacionamento muitos-para-muitos para entidade Publicação. Seguindo a representação de SuperGrafo, o primeiro nível da hierarquia terá uma partição (ou SuperNó) para cada entidade correspondente – Pessoa e Publicação, no caso. Esses SuperNós são adicionados como filhos do nó raiz.

Ao utilizar este particionamento inicial, a visualização de dados hierárquica correspondente apresentará as entidades da base de dados no seu primeiro nível, conforme ilustrado na Figura 34(a). Neste primeiro nível, as arestas entre Pessoa e Publicação definem um SuperAresta composta por todas as arestas entre os nós dessas partições. O primeiro nível da visualização, portanto, fornece uma visão geral de como os dados são estruturados e como as diferentes entidades se manifestam no banco de dados.

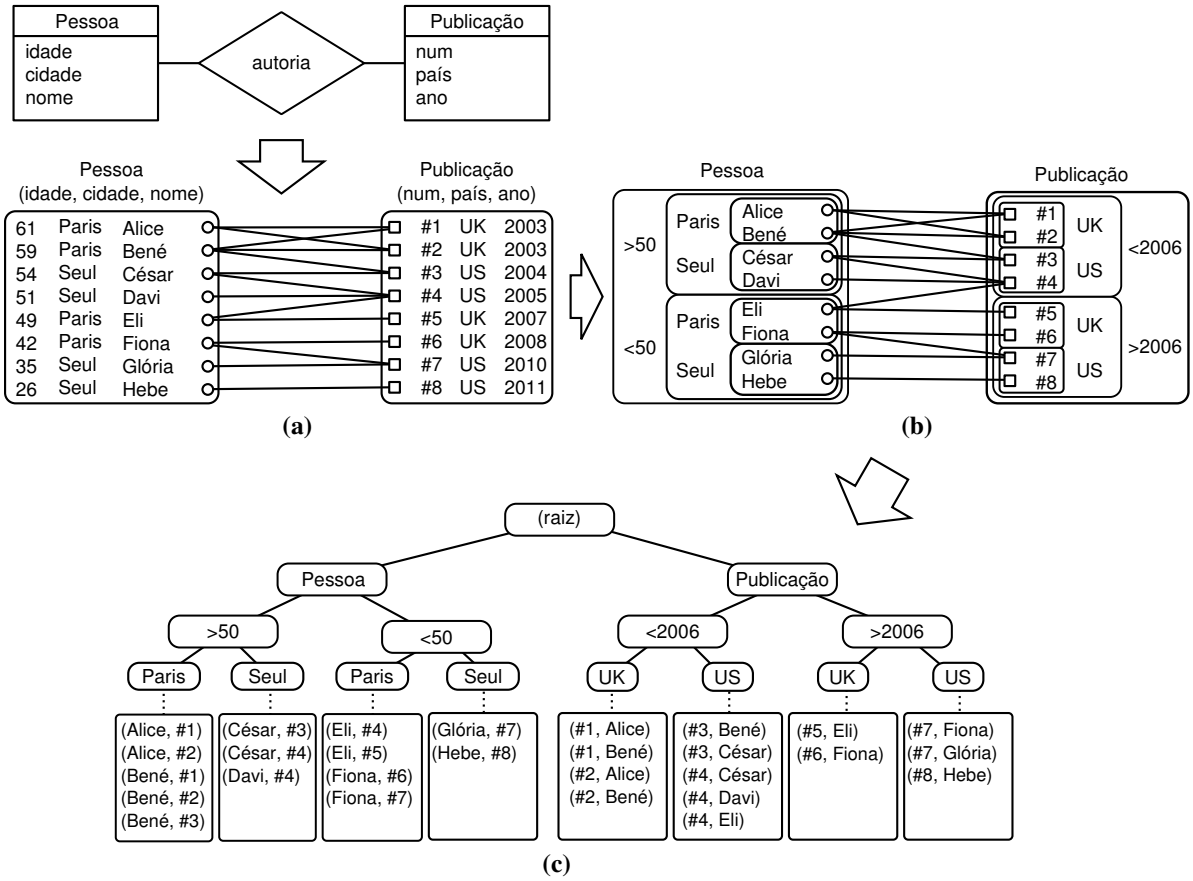


Figura 34 – Construção de um SuperGrafo.

No sistema RMine, esta visualização é interativa. Além de visualizar os SuperNós, é possível recuperar mais detalhes inspecionando as SuperArestas da visualização. Assim, um duplo clique sobre uma SuperAresta de interesse faz com que a Graph-Tree carregue e apresente quais nós interagem uns com os outros nesta SuperAresta, além de prover informações agregadas de contagem.

Níveis seguintes

A seguir, a ideia é ter a possibilidade de refinar as informações de cada SuperNó de entidade no primeiro nível. Isto é, cada SuperNó deve ser particionado em outro conjunto de SuperNós, formando um nível inferior na hierarquia. O problema aqui é como determinar esse particionamento em cada nível, e que número de níveis utilizar. Para responder a estas duas perguntas, este método considera os atributos de cada entidade como as informações para orientar os níveis mais baixos do particionamento. Prossegue-se considerando:

a) o número e o significado dos níveis são dados pelos atributos de cada entidade, um nível para cada atributo;

b) o número de particionamentos em um dado nível é dado pela distribuição dos valores do atributo como se observa na instancia da base de dados.

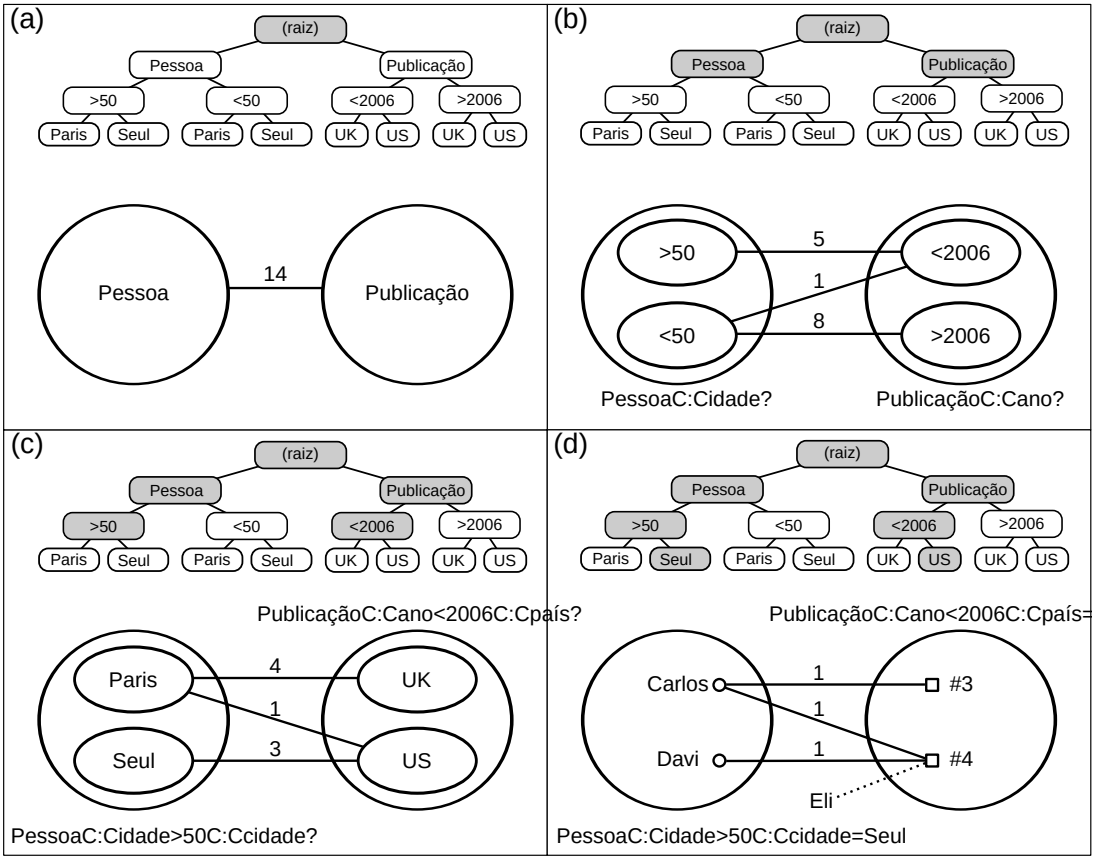


Figura 35 – Visualização hierárquica de um SuperGrafo.

A ação a) implica que os atributos mais representativos de uma determinada entidade devem ser considerados. No exemplo, a entidade Pessoa pode ser representada por idade e cidade, determinando dois níveis abaixo do primeiro nível, e a entidade Publicação pode ser representada por país e ano, novamente dois níveis abaixo do primeiro nível – ilustrados na Figura 34. A ação b) é um pouco mais complicada, pois ela requer a inspeção do banco de dados para se verificar a distribuição dos valores de cada atributo.

Existem diferentes tipos de atributos, os principais tipos são: qualitativos categóricos ou nominais, e quantitativos. Atributos categóricos com pequeno número de categorias, como sexo, por exemplo, irão determinar uma partição para cada categoria. Para um maior número de categorias, é interessante um particionamento uniforme de acordo com sua distribuição. Para atributos nominais e quantitativos, o número de partições pode ser determinado considerando a distribuição dos valores.

A Figura 34(b) mostra um exemplo com cada atributo dividido em dois grupos – o atributo idade divide o SuperNó Pessoa em dois SuperNós "$idade > 50$" e "$idade < 50$", e então cada SuperNó é novamente dividido em dois SuperNós, "$cidade = Paris$" e "$cidade = Seoul$". Este método resulta na estrutura Graph-Tree ilustrada na Figura 34(c), a qual pode ser usada em uma visualização hierárquica do SuperGrafo como ilustrado na Figura 35. De acordo com tal visualização, o usuário pode expandir níveis em ramos diferentes na árvore para inspecionar as SuperArestas entre os vários arranjos de SuperNós, de maneira comparativa.

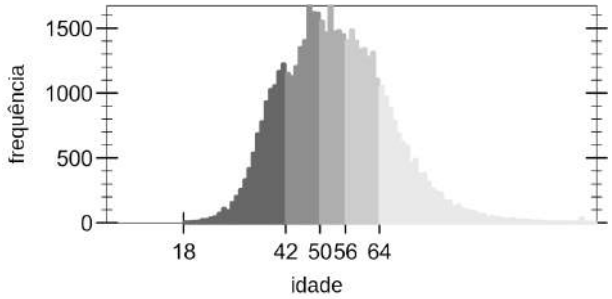


Figura 36 – Distribuição do atributo *Pessoa.idade* evidenciando o particionamento automático em quintis ($k = 5$).

Ainda para atributos nominais e quantitativos, pode-se considerar um número maior de partições, o que pode ser alcançado com análise estatística clássica baseada em percentis. Como exemplo, a Figura 36 mostra a distribuição das idades de um banco de dados realístico, utilizando particionamento em $k = 5$ partições. Por ser um atributo quantitativo, utiliza-se um particionamento por faixa de valores, com frequências aproximadamente homogêneas. Obtém-se assim os quintis $QU_1 = [18, 41]$ (0-20% das observações), $QU_2 = [42, 49]$ (20-40% das observações), $QU_3 = [50, 55]$ (40-60% das observações), $QU_4 = [56, 63]$ (60-80% das observações) e $QU_5 = [64, 162]$ (80-100% das observações).

Seguindo a abordagem estatística por percentis, as partições obtidas tendem a ter o mesmo tamanho em número de instâncias. Outras abordagens podem ser utilizadas, como por exemplo, indicando um particionamento pelo desvio padrão, assim obtendo partições de $[\bar{x}, \sigma]$, $]\sigma, 2\sigma]$, e assim em diante, o que tende a separar os extremos da distribuição e pode ser utilizado para se observar o comportamento de minorias de indivíduos.

Estes são procedimentos diretos destinados a uma análise prévia automática, mas a metodologia também pode se beneficiar a partir de parâmetros definidos pelo usuário para objetivos específicos de análise. Contudo, determinar esses parâmetros automaticamente (como número de partições, distribuição, ordenação das características, dentre outros) não é uma tarefa trivial. Uma vez que se lida com um conjunto visual interativo, o valor inicial de k é escolhido a partir da Lei de Miller (MILLER, 1956), que afirma que a memória de trabalho humana é limitada a 7 ± 2 elementos, sugerindo o uso de k não maior que esse valor. Esta restrição conduz a um número de partições por nível que não sobrecarrega o usuário.

Construção visual das consultas SQL

O particionamento relacional descrito concebe uma visualização do banco de dados desenvolvida diretamente do esquema relacional, e que remete visualmente a um diagrama entidade-relacionamento. Com essa estrutura, não é difícil conceber as operações relacionais e consultas SQL correspondentes para construir essas visualizações. Esta subseção esclarece como estas consultas são abstraídas, as desvantagens dessas consultas SQL, e como estas consultas podem ser resolvidas com auxílio da estrutura Graph-Tree.

No primeiro nível, cada SuperNó corresponde a uma relação de entidade do modelo entidade-relacionamento. Assim, cada SuperNó desse primeiro nível representa a seleção de todos os objetos da entidade correspondente; e as SuperArestas entre cada um desses SuperNós representa a seleção das arestas entre essas entidades. A consulta pelas entidades seria simplesmente uma seleção sem predicados, por exemplo, na Figura 35 o SuperNó no primeiro nível correspondente a Pessoa seria:

SuperNó **Pessoa** ← `SELECT nome FROM Pessoa;`

A cada nível de detalhe selecionado na hierarquia, é expandido um SuperNó filho correspondente aos objetos que atendem ao predicado determinado pelo SuperNó. No exemplo, os SuperNós filhos do SuperNó Pessoa determinam predicados envolvendo o atributo idade de acordo com o particionamento definido. Como são definidos dois SuperNós, "< 50" e "> 50" respectivamente, cada SuperNó é a concatenação das consultas do SuperNó pai e o predicado correspondente ao SuperNó:

SuperNó **Pessoa : idade<50**
← `SELECT nome FROM Pessoa WHERE idade < 50;`

SuperNó **Pessoa : idade>50**
← `SELECT nome FROM Pessoa WHERE idade > 50;`

E assim recursivamente para os demais níveis da hierarquia. Por exemplo, os SuperNós filhos de "Pessoa : idade < 50" seriam dados pelas consultas:

SuperNó **Pessoa : idade<50 : cidade=Paris**
← `SELECT nome FROM Pessoa WHERE idade < 50 AND cidade = 'Paris';`

SuperNó **Pessoa : idade>50 : cidade=Seoul**
← `SELECT nome FROM Pessoa WHERE idade > 50 AND cidade = 'Seoul';`

Entretanto, essas são consultas SQL simples que são rapidamente respondidas caso as tabelas tenham índices criados sobre os atributos que são referenciados nas consultas (nesse caso: nome, idade e cidade). Contudo, o interesse principal da visualização em grafos é expor os *relacionamentos* entre as entidades, que são as arestas representadas pelas tuplas das relações de relacionamentos. Por exemplo, a primeira SuperAresta visível na Figura 35 representa os relacionamentos entre os objetos das entidades Pessoa e Publicação. As arestas dessa SuperAresta podem ser selecionadas diretamente da relação que relaciona essas duas entidades:

SuperAresta **Pessoa** × **Publicação** ←
```
SELECT pes.nome, pub.titulo
FROM ParticipacaoPublicacao pp
JOIN Pessoa pes ON pp.cod_pessoa = pes.cod_pessoa
JOIN Publicacao pub ON pp.cod_publicacao = pub.cod_publicacao;
```

Relembrando que cada SuperNó abre um caminho na hierarquia contendo mais SuperNós subdivididos recursivamente, é possível calcular SuperArestas entre *cada par* desses SuperNós internos, mesmo entre dois níveis diferentes na hierarquia. Por exemplo, ao se perguntar sobre que objetos "Pessoa" têm "Publicações" após o ano 2006, selecionam-se as arestas entre esses dois SuperNós, as quais são extraídas do relacionamento entre essas entidades com a seguinte consulta:

SuperAresta **Pessoa** × **Publicação : ano>2006** ←
```
SELECT pes.nome, pub.titulo
FROM ParticipacaoPublicacao pp
JOIN Pessoa pes ON pp.cod_pessoa = pes.cod_pessoa
JOIN Publicacao pub ON pp.cod_publicacao = pub.cod_publicacao
WHERE pub.ano > 2006;
```

À medida que o usuário navega pelos diferentes níveis hierárquicos da visualização definida pelo particionamento relacional, e vai consultando os relacionamentos entre cada grupo (representado por um SuperNó) de entidades, várias dessas SuperArestas são consultadas. Quanto mais se aprofunda na hierarquia, mais predicados são adicionados à consulta, aumentando a seletividade das mesmas. Porém, a junção necessária para o cálculo da SuperAresta traz um custo computacional e cognitivo elevados, e estes problemas se tornam ainda mais complexos quando as entidades são relacionadas por mais de um relacionamento, exigindo várias junções aninhadas.

É neste ponto que é possível utilizar a estrutura Graph-Tree. Esta estrutura armazena o grafo com os relacionamentos entre os objetos das entidades do banco de dados. Além disso, a estrutura armazena as arestas correspondentes a estes relacionamentos de maneira distribuída pela hierarquia, permitindo que seja possível o cálculo das SuperArestas entre dois SuperNós visitados buscando apenas nos seus SuperNós ancestrais, como ilustrado na Figura 37.

Com a estrutura Graph-Tree, as consultas por SuperArestas são realizadas apenas na parte da hierarquia que já foi visualmente visitada, isto é, nos SuperNós que foram expandidos e são ancestrais dos SuperNós dos quais se deseja visualizar os relacionamentos. O desempenho da implementação dessa estrutura no protótipo RMine é avaliado no capítulo seguinte, Experimentos, onde se confirmam melhorias significativas sobre a execução dessas consultas com um SGBD Relacional.

4.7 Conclusão do capítulo

Neste capítulo foi descrito todo o processo de tratamento de um banco de dados relacional para a representação em grafos, estrutura de dados para comportar esse grafo, composição de um particionamento hierárquico a partir de informações do esquema, e projeto de uma visualização hierárquica desse grafo previamente construído. Adicionalmente, foram abordadas algumas

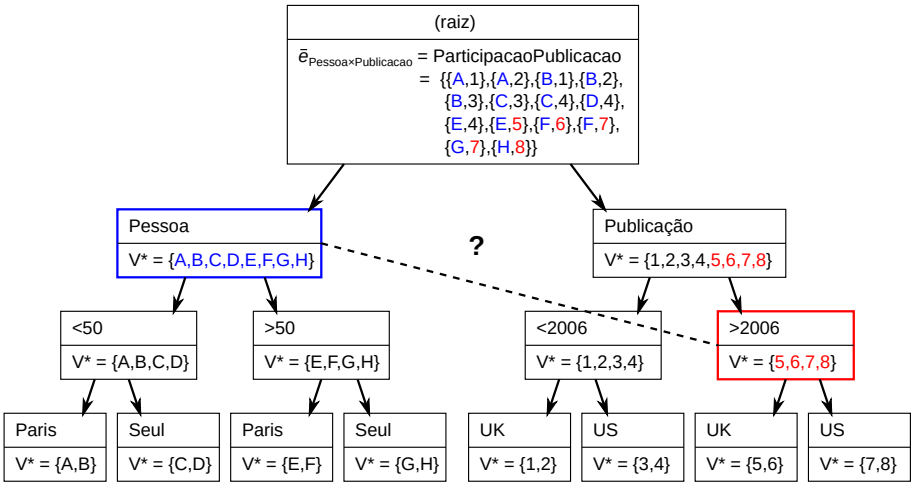


Figura 37 – Cálculo de SuperAresta na Graph-Tree da Figura 35. Dado que se busca uma SuperAresta entre [Pessoa] e [Publicação : ano>2006]; a SuperAresta com os nós relacionados é obtida pela busca nos SuperNós ancestrais pelas arestas cujos vértices são os nós do grafo descendentes desses SuperNós. As arestas que possuem vértices de ambos (azul e vermelho) formam a SuperAresta entre esses dois SuperNós. (Aqui V^* representa a cobertura dos vértices de cada SuperNó, isto é, seus descendentes.)

notas sobre a implementação dessas mudanças no protótipo desenvolvido, servindo de ponto de partida para futuras modificações.

Com a implementação do protótipo RMine, tornou-se possível o tratamento de bancos de dados relacionais em um ambiente visual interativo sobre a estrutura Graph-Tree orientada a disco. Utilizando-se o RMine, essa metodologia de particionamento e visualização de relações é posta à prova no próximo capítulo, por meio de experimentos que objetivam demonstrar o potencial deste trabalho em termos de desempenho e usabilidade durante a execução de diferentes tarefas analíticas.

Capítulo

5 Experimentos

Para demonstrar o potencial da abordagem proposta, esta seção cobre algumas experiências. Dado um banco de dados, são apresentadas algumas tarefas exploratórias que ilustram cada tarefa em conjunto com suas respectivas consultas SQL e tempos de processamento. A intenção é demonstrar que este método pode substituir complexas e dispendiosas operações SQL que, de outro modo, exigiriam tempo para serem escritas e processadas. O método proposto permite que as mesmas operações sejam executadas satisfazendo restrições de tempo de interação.

5.1 Dados e configurações

Os dados utilizados nas experiências seguintes provém da base de dados Tycho-USP (ilustrado na Figura 38). O sistema Tycho-USP é um sistema acadêmico da Universidade de São Paulo, que reúne dados sobre alunos, professores, e seus trabalhos acadêmicos. Os dados são coletados a partir de diversos sistemas de informação na universidade e são fundidos com dados externos de outras agências científicas do governo brasileiro.

Esses dados estão estruturados em um esquema relacional com cinco entidades principais: Eventos (352.400 nós), Bancas (382.890 nós), Publicações (691.083 nós), Supervisões (26.237 nós) e Pessoas (50.779 nós); com relacionamentos Pessoa-Banca (851.168 arestas), Pessoa-Evento (247.516 arestas), Pessoa-Publicação (691.083 arestas), Pessoa-Supervisão (52.439 arestas) e Publicação-Evento (26.237 arestas), perfazendo um total de 1.503.389 nós e 1.868.443 arestas.

Este banco de dados foi escolhido para ilustrar a capacidade da metodologia proposta para lidar com estruturas relacionais complexas, com bancos de dados que não cabem em memória, e com visualizações que extrapolam a capacidade dos dispositivos de exibição.

As quatro entidades da base Tycho-USP foram consideradas de acordo com seus atributos de interesse, da seguinte maneira: Pessoa (idade, localização, gênero), Publicação (país, ano, assunto), Evento (classificação, esfera, tipo), Supervisão (tipo, progresso, papel) e Banca (país, curso, instituição). A estrutura do banco de dados foi considerada de acordo com os relacionamentos que ligam Pessoa para todas as outras entidades, e Publicação a Evento. Os atributos

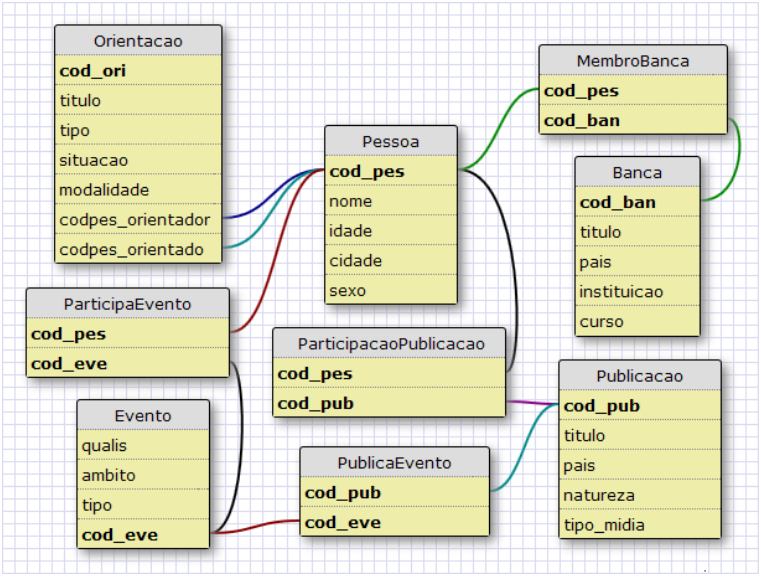


Figura 38 – Diagrama do modelo relacional do banco de dados Tycho-USP.

foram utilizados para determinar os níveis da hierarquia e as relações foram utilizadas para determinar as arestas do gráfico subjacente.

Para os experimentos utiliza-se o parâmetro $k = 5$. Os atributos nominais são divididos em classes correspondentes a cada possível valor do atributo. As k classes com maior cardinalidade são exibidas na partição, e mais uma classe "(Outros)" com a união das demais classes menos frequentes. Atributos numéricos são divididos em percentis com aproximadamente a mesma cardinalidade, como descrito na Seção 4.6.

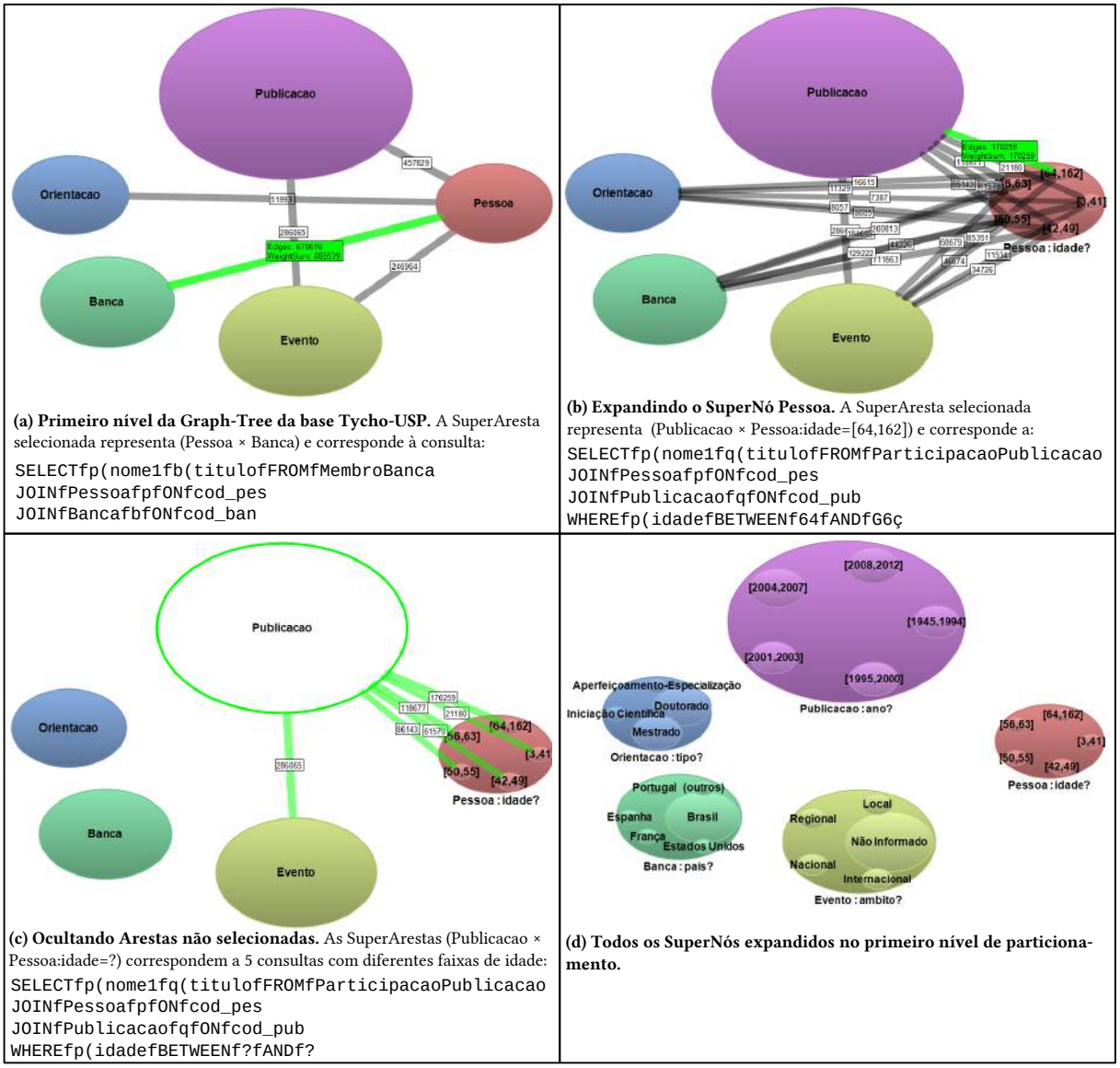
A primeira tarefa é construir a estrutura de dados, para este fim, o nosso método recebe um conjunto de configurações contendo as entidades e relacionamentos de interesse e constrói uma Graph-Tree vazia. Em seguida, o procedimento de construção escreve os nós e arestas nos SuperNós folha, e preenche os níveis superiores com SuperArestas de conectividade. Essa etapa inicial leva em torno de 7 minutos e cria uma Graph-Tree persistente no disco. A partir desta estrutura, a mesma base de dados pode ser carregada em menos de 10 segundos.

Todas as medidas de tempo são tempos de execução decorridos a partir do relógio do sistema (wall-clock), tomadas em um computador pessoal equipado com um processador AMD Phenom II X4 850, 4 GB de memória principal DDR3, um único disco rígido SATA de 500 GB e sistema operacional Microsoft Windows 7 Professional 64-bit.

5.2 Análise Visual

Nesta seção, demonstra-se como o método proposto pode ser usado para inspecionar visualmente um banco de dados relacional. Executam-se as seguintes tarefas:

1. Visualizar a distribuição das entidades e de seus relacionamentos: quantas Publicações, Pessoas, e Eventos existem e como eles estão organizados?



(a) Primeiro nível da Graph-Tree da base Tycho-USP. A SuperAresta selecionada representa (Pessoa × Banca) e corresponde à consulta:

```
SELECTfp(nome1fb(titulofFROMfMembroBanca
JOINfPessoafpfONfcod_pes
JOINfBancafbfONfcod_ban
```

(b) Expandindo o SuperNó Pessoa. A SuperAresta selecionada representa (Publicacao × Pessoa:idade=[64,162]) e corresponde a:

```
SELECTfp(nome1fq(titulofFROMfParticipacaoPublicacao
JOINfPessoafpfONfcod_pes
JOINfPublicacaofqfONfcod_pub
WHEREfp(idadefBETWEENf64fANDfG6ç
```

(c) Ocultando Arestas não selecionadas. As SuperArestas (Publicacao × Pessoa:idade=?) correspondem a 5 consultas com diferentes faixas de idade:

```
SELECTfp(nome1fq(titulofFROMfParticipacaoPublicacao
JOINfPessoafpfONfcod_pes
JOINfPublicacaofqfONfcod_pub
WHEREfp(idadefBETWEENf?fANDf?
```

(d) Todos os SuperNós expandidos no primeiro nível de particionamento.

Figura 39 – Sequência de interações ilustrando a expansão de SuperNós.

2. Visualizar o relacionamento entre Pessoa e Publicação: que grupos têm o maior número de publicações?

3. Quem são os autores mais ativos nos últimos anos?

Com a Graph-Tree construída, o sistema RMine carrega e apresenta o primeiro nível abaixo da raiz, com um SuperNó para cada entidade do banco de dados. Como observado na Figura 39(a), por seleção de um dos SuperNós, o sistema RMine calcula e apresenta as SuperArestas que ligam este SuperNó para as outras entidades. Os tamanhos dos SuperNós e a espessura das SuperArestas são proporcionais ao número de nós e o número de arestas (relacionamentos) que representam, respectivamente. Desta maneira, pode-se dizer intuitivamente quais são as maiores relações (Publicação e Evento) e quais são as relações mais intensas (Pessoa-Evento e Publicação-Pessoa), assim como abordadas pela pergunta 1).

O passo seguinte da interação é expandir um SuperNó de interesse. Essa ação desencadeia o particionamento do próximo nível, de acordo com o primeiro atributo. Cada expansão de um SuperNó desencadeia uma série de cálculos que atribuem SuperArestas de conectividade entre os SuperNós filhos recém-expostos e os SuperNós restantes na visualização (Figura 39(b)). Para o exemplo de Publicação-Pessoa, pode-se responder à pergunta 2) pela simples leitura dos pesos das SuperArestas de conectividade – que, naturalmente, apontam para as partições de pessoas mais antigas, de acordo com atributo idade. Após o primeiro nível, mostrar todas as arestas iria sobrecarregar a visualização, assim, escondem-se as SuperArestas que ligam SuperNós não selecionados (Figura 39(c)).

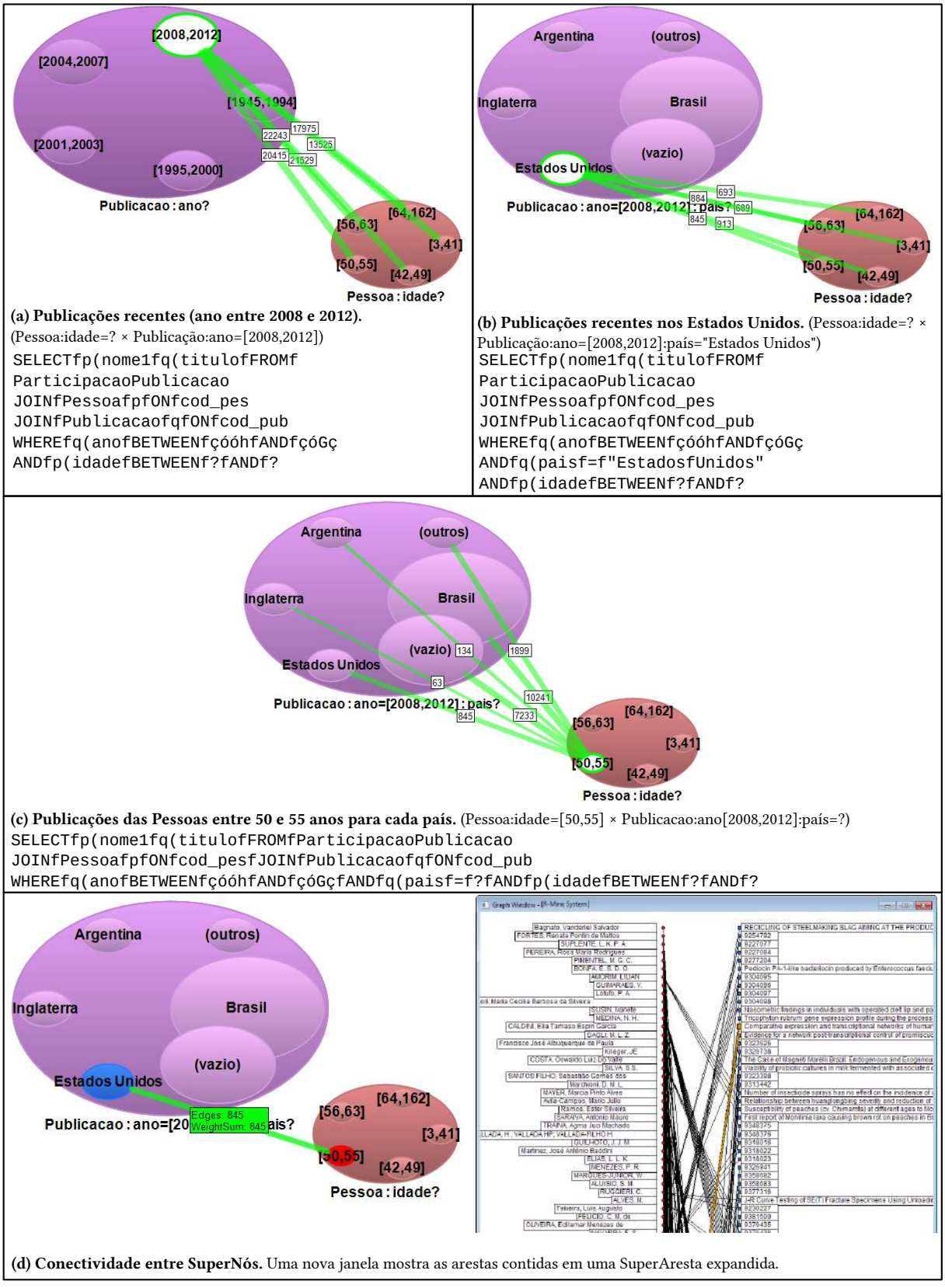
Depois de expandir mais entidades, a visualização será semelhante à Figura 39(d), em que um subconjunto de partições é apresentado em níveis mais profundos da hierarquia. Olhando em detalhes o SuperNó Pessoa, agora dividido por idade, observa-se que o particionamento automático faz com que cada um dos SuperNós Pessoa por idade corresponda a uma faixa com um número aproximadamente igual de objetos. A figura mostra que cerca de 20% das pessoas neste banco de dados têm menos de 42 anos de idade. Em outro aspecto da partição Publicação por ano, pode-se ver que os intervalos das partições tendem a encurtar para os períodos mais recentes; já que a divisão seguiu a abordagem percentual, isso significa um aumento no número de publicações por ano.

Neste ponto, expandindo a partição de Publicação e seu nível de particionamento por ano, pode-se responder a questão 3). Para esta tarefa, seleciona-se o SuperNó mais recente (2008-2012), para que seja possível ver as SuperArestas de conectividade para cada um dos outros SuperNós (Figura 40(a)). A visualização mostra que o grupo de pessoas entre 50 e 55 anos é o que tem o maior número de publicações. Ao expandir este SuperNó, e olhar para as publicações 2008-2012 por país (Figura 40(b)), pode-se confirmar que esta partição é realmente a líder no que diz respeito a publicações, no período mais recente.

Também é possível selecionar um SuperNó Pessoa por idade, e ver suas relações com publicações recentes particionadas por países (Figura 40(c)). Nesta visualização, clica-se duas vezes em uma dessas SuperArestas de conectividade, e obtém-se a tela mostrada na Figura 40(d). É uma visualização do grafo exibindo nós (tuplas) e arestas (relacionamentos) englobados pelos SuperNós em cada extremidade da SuperAresta em destaque na figura.

Esta tela em particular emprega um layout bipartido, com nós decrescentemente ordenados por número de arestas, o quê nos permite identificar os nós mais inter-relacionados no que diz respeito ao número de publicações. Além disso, para reduzir a quantidade de elementos nesta tela, o sistema RMine permite ocultar nós com poucos relacionamentos (aqueles cujo grau é menor que um parâmetro dado).

Esta seção demonstrou que o método proposto permite que consultas com agregações complexas sejam realizadas de forma intuitiva, e dentro de um tempo de interação estrito – da ordem de 0,1 segundo. Mostra-se que, não somente o usuário é poupado da necessidade de escrever código SQL complexo, mas também que os requisitos de processamento são significa-



(a) Publicações recentes (ano entre 2008 e 2012). (Pessoa:idade=? × Publicação:ano=[2008,2012])

```
SELECTfp(nome1fq(titulofFROMf
ParticipacaoPublicacao
JOINfPessoafpfONfcod_pes
JOINfPublicacaofqfONfcod_pub
WHEREfq(anofBETWEENfçóóhfANDfçóGç
ANDfp(idadefBETWEENf?fANDf?
```

(b) Publicações recentes nos Estados Unidos. (Pessoa:idade=? × Publicação:ano=[2008,2012]:país="Estados Unidos")

```
SELECTfp(nome1fq(titulofFROMf
ParticipacaoPublicacao
JOINfPessoafpfONfcod_pes
JOINfPublicacaofqfONfcod_pub
WHEREfq(anofBETWEENfçóóhfANDfçóGç
ANDfq(paisf=f"EstadosfUnidos"
ANDfp(idadefBETWEENf?fANDf?
```

(c) Publicações das Pessoas entre 50 e 55 anos para cada país. (Pessoa:idade=[50,55] × Publicacao:ano[2008,2012]:país=?)

```
SELECTfp(nome1fq(titulofFROMfParticipacaoPublicacao
JOINfPessoafpfONfcod_pesfJOINfPublicacaofqfONfcod_pub
WHEREfq(anofBETWEENfçóóhfANDfçóGçfANDfq(paisf=f?fANDfp(idadefBETWEENf?fANDf?
```

(d) Conectividade entre SuperNós. Uma nova janela mostra as arestas contidas em uma SuperAresta expandida.

Figura 40 – Sequência de interações explorando os relacionamentos entre SuperNós.

tivamente reduzidos. O custo desses benefícios é o tempo de pré-processamento que, também, está dentro de um limite de tempo aceitável, especialmente ao se considerar o fato de que a estrutura Graph-Tree é persistente no disco.

5.3 Desempenho do protótipo

Na seção anterior, foi demonstrado o apelo interativo visual do presente trabalho, que se presta à análise de dados exploratória, e também como o método possibilita que os usuários não precisem escrever consultas SQL complexas por meio das funcionalidades da Graph-Tree. Além destas contribuições, este método permite que custosas agregações relacionais sejam executadas em uma fração do tempo que um sistema de banco de dados relacional levaria para a mesma tarefa.

Nesta seção, demonstra-se esta funcionalidade por meio da comparação do tempo para a execução de consultas em um banco de dados relacional, e do tempo de processamento no sistema RMine. Para isso, utilizam-se 374 cálculos de conectividade diferentes, cada cálculo correspondendo a uma SuperAresta calculada no sistema RMine, e também à agregação SQL equivalente. Para estas medidas de tempo, utilizou-se o sistema PostgreSQL 9.2.1 amd64 e um esquema com todos os índices necessários para responder às consultas com bom desempenho.

Por exemplo, há SuperArestas de conectividade entre Publicação por ano e país e Pessoa por idade. Esta informação pode ser obtida de forma interativa no sistema RMine ou usando uma consulta SQL em um SGBDR. Por exemplo, a conectividade entre o SuperNó (Publicação : ano=[2008-2012] : país=Estados Unidos) e o SuperNó (Pessoa : idade=[18-41]) pode ser selecionado a partir do esquema na Figura 5 pela consulta SQL a seguir:

```
SELECT Pessoa.nome, Publicacao.titulo
FROM ParticipacaoPublicacao
JOIN Pessoa ON cod_pes
AND Pessoa.idade BETWEEN 18 AND 41
JOIN Publicacao ON cod_pub
AND (Publicacao.ano BETWEEN 2008 AND 2012)
AND (Publicacao.pais = 'Estados Unidos');
```

O mesmo vale para os 374 cálculos de conectividade ilustrados nesta seção. Os cálculos variam, tanto para o SGBDR quanto para o método proposto, dependendo da intensidade da relação entre os SuperNós que participam na análise. Por isso, mostra-se que o tempo necessário pelo sistema RMine é significativamente menor que o tempo que um SGBDR levaria; não se considera o tempo que um usuário levaria para escrever o SQL desejado, pois este tempo é suplantado pelo aspecto interativo apresentado nas seções anteriores.

A Figura 41 mostra o tempo para cada uma das consultas consideradas no experimento. A figura mostra que, em todos os casos, exceto para as primeiras cinco consultas, o método proposto consegue a resposta mais rapidamente do que o SGBDR. A figura também mostra que o tempo varia dependendo

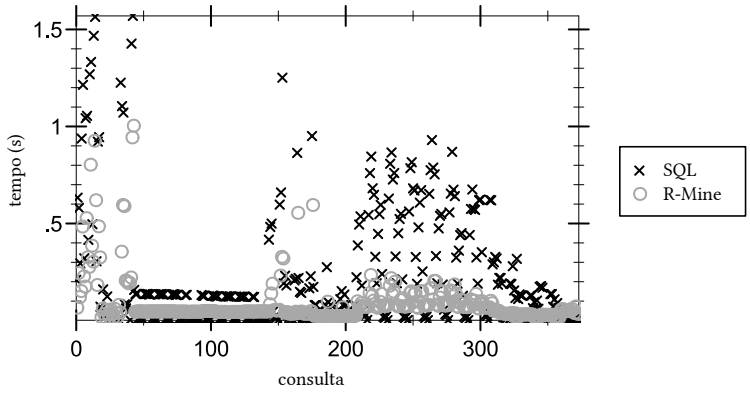


Figura 41 – Tempo de execução de cada cálculo de conectividade no RMine e consulta SQL correspondente no PostgreSQL.

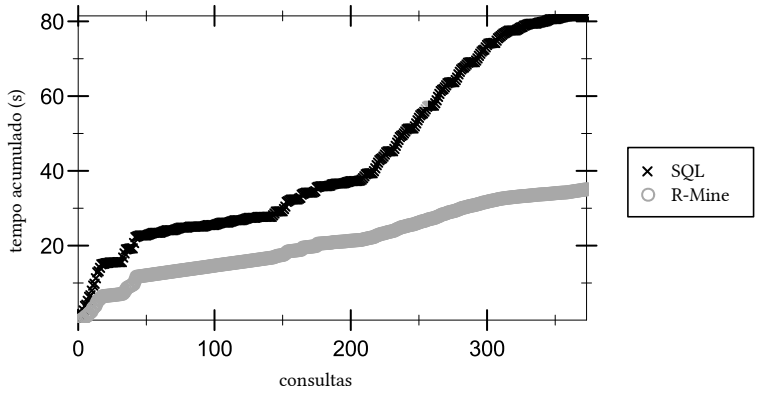


Figura 42 – Tempo de execução acumulado dos cálculos de conectividade no RMine e consultas SQL correspondente no PostgreSQL.

Tabela 1 – Tempo acumulado medido para a expansão de cada SuperNó.

SuperNó expandido	*Carga* (s)	*Conectividade* (s)	*SQL* (s)
(carga inicial)	6,03	-	-
Pessoa	0,06	5,85	7,35
Evento	0,27	5,28	26,72
Publicação	0,16	4,48	27,68
Total	6,52	15,61	61,74

do atributo envolvido na consulta; alguns atributos nominais determinam partições desbalanceadas, e, portanto, conjuntos de resultados desbalanceados. Em outras palavras, algumas consultas retornam mais dados do que outras. A Figura 42 apresenta o tempo acumulado para as consultas. A figura mostra que, para sequências de consultas, o método proposto progride aritmeticamente melhor do que o SGBDR. Esta foi uma necessidade na concepção da Graph-Tree orientada a disco, porque a interação exploratória pede sequências longas de tentativa e erro.

Na Tabela 1 estão listados os tempos tomados pelo RMine e pelo SGBDR em passos chave da interação, que são as expansões de SuperNós. A coluna Carga corresponde ao tempo que o RMine leva para carregar dados do disco; a coluna Conectividade corresponde ao tempo que uma operação de expansão leva no RMine – uma expansão desencadeia um conjunto de cálculos de conectividade do

SuperNó expandido para outros SuperNós no contexto, e a coluna SQL corresponde ao tempo que o SGBDR leva para executar as consultas equivalentes. Ainda na Tabela 1, considera-se uma linha com o tempo para carga inicial da Graph-Tree pré-processada do esquema Tycho-USP, e linhas para cada uma das operações de expansão, considerando as entidades Pessoa, Evento e Publicação. Todos os tempos estão em segundos. A totalização da tabela demonstra como uma seção interativa executando sobre um SGBDR seria proibitiva. Na verdade, SGBDRs não são projetados para inspeção exploratória de dados, e este é um dos pontos atacados neste trabalho.

5.4 Testes de usabilidade

Para validação do protótipo interativo são empregados testes empíricos com usuários; os testes seguem aos métodos previstos da disciplina de engenharia de usabilidade. Nos testes, os usuários executam uma tarefa interativa sobre a base de dados mostrada anteriormente neste capítulo (Tycho-USP). O grupo de usuários é formado por estudantes e profissionais de computação, jovens (20 a 30 anos), com domínio em projeto de bancos de dados relacionais e linguagem SQL.

Esta tarefa é composta do planejamento e execução de quatro consultas em sequência, onde os usuários utilizam tanto o protótipo RMine quanto o cliente do SGBD (pgAdmin). O tempo total levado por cada usuário é medido para cada método de execução (x_S para SQL e x_R para RMine). Após a execução das tarefas, esses dados são avaliadas por testes de significância estatística. As consultas contidas na tarefa são apresentadas na forma de perguntas a serem respondidas pelos usuários, sendo as que seguem:

- Que estados possuem maior número de pessoas jovens? Considerando as pessoas de um desses estados, a quais categorias de orientação elas estão mais relacionadas?
- Considerando apenas as publicações de tipo “completo”, em que países são publicados mais trabalhos desse tipo? Que faixa etária de pessoas é responsável pelo maior número dessas publicações nos últimos anos (2008-2012)?
- Quais publicações do final dos anos 90 (1995-2000) foram publicadas em eventos de âmbito local na Argentina? Qual a natureza dessas publicações?
- Considerando as defesas de bancas realizadas no exterior, qual o país preferido pelas pessoas, qual o curso nesse país com mais bancas defendidas e qual é a faixa etária das pessoas que participaram dessas bancas?

Várias recomendações são sugeridas para a realização e apresentação dos testes aos usuários. Contudo, dentre os frequentes problemas encontrados na execução desse tipo de avaliação, os mais críticos são os relacionados à seleção e distribuição de tarefas adequadas aos grupos de usuários, que devem ter perfis pessoais e profissionais próximos do público-alvo esperado; e ao aprendizado implícito à realização da tarefa, o que favorece os resultados provenientes das tarefas mais simples e também das mais tardiamente executadas.

Para atenuar esses fenômenos, o grupo de usuários é escolhido dentro de uma faixa mais limitada, formada por estudantes e profissionais de bancos de dados. Cada um dos usuários realiza todas as tarefas,

Tabela 2 – Tempos de execução das tarefas de usuário com o cliente SQL pgAdmin e o sistema RMine, em segundos.

1		2		3		4	
x_S	x_R	x_S	x_R	x_S	x_R	x_S	x_R
140	91	211	119	299	100	255	142
414	107	541	184	476	173	466	205
447	93	715	122	547	96	429	120
383	125	473	126	262	86	319	122
380	117	484	177	230	189	369	111

e a ordem de execução das tarefas é sorteada aleatoriamente para cada usuário. Na Tabela 2 seguem os tempos de execução das tarefas por cada usuário.

A partir destes dados, em primeiro lugar as amostras são caracterizadas pelas medidas amostrais de média e desvio padrão para se obter uma intuição da distribuição dos valores. A seguir é aplicado um teste de normalidade de Kolmogorov-Smirnov (Lilliefors) para confirmar se a distribuição dos dados é normal, o que é uma condição necessária para vários testes de significância, como o teste F e o teste T de Student usados neste trabalho.

Tomando $\bar{x}_S$ e $\bar{x}_R$ os tempos médios levados pelos usuários para executar as tarefas respectivamente com o cliente SQL e o RMine, deseja-se comprovar a hipótese de que o tempo médio usando-se o RMine é menor do que tempo médio usando-se um cliente SQL. Para tal, assume-se a hipótese de que $\bar{x}_R$ é maior ou indiferente de $\bar{x}_S$, e rejeita-se essa hipótese apenas se o resultado do teste for menor do que o valor crítico da cauda esquerda, sendo assim:

$$\begin{aligned} H_0 &: \bar{x}_S \leq \bar{x}_R \\ H_1 &: \bar{x}_S > \bar{x}_R \end{aligned} \tag{5.1}$$

Na Tabela 3 a seguir seguem os resultados do teste de normalidade (Kolmogorov-Smirnov) para as amostras. E na Tabela 4 seguem os testes de significância (unicaudais) pareados sobre x_S e x_R: teste F, teste T de Student, e teste de Wilcoxon. Utiliza-se um valor crítico $\alpha = 5\%$, normalmente sugerido para esses testes estatísticos e bem aceito pela comunidade científica.

Tabela 3 – Teste de normalidade Kolmogorov-Smirnov sobre as amostras de tempo.

Teste Kolmogorov-Smirnov	x_S	x_R
α	0,05	0,05
KS_{crit}	0	0
KS	0,100727	0,247055
valor-p	0,858380	0,002386
(distribuição é normal se $p > \alpha$)		
conclusão:	possivelmente normal	não normal

Como pode ser notado na Tabela 3, o teste de Kolmogorov-Smirnov atesta que a distribuição do conjunto de dados x_S é possivelmente normal, mas que a distribuição de x_R não é normal. Isto significa que os testes F e T de Student – que assumem distribuição normal – podem não trazer resultados significativos se aplicados sobre os amostras de x_R. Como precaução neste caso, também foi aplicado o

teste T de Wilcoxon, para fornecer evidência mesmo que ocorra um erro com os testes Kolmogorov-Smirnov, F ou T de Student.

Tabela 4 – Testes de significância unicaudais sobre pares $\langle x_S, x_R \rangle$ de amostras de tempo.

Teste F		Teste T de Student		Teste T de Wilcoxon	
α	0,05	α	0,05	α	0,05
F_{crit}	2,168252	T_{crit}	1,729133	$S+$	210
F	14,494731	T	8,765205	$S-$	0
valor-p	0,000000140	valor-p	0,000000021	valor-p	0,000047845
(variâncias são iguais se $p > \alpha$)		(rejeitar H_0 se $p < \alpha$)		(rejeitar H_0 se $p < \alpha$)	
conclusão: variâncias desiguais			$\bar{x}_S > \bar{x}_R$		$\bar{x}_S > \bar{x}_R$

A Tabela 4 contém os resultados dos testes de significância. Como a variável x_R parece não seguir uma distribuição normal, o teste T de Student não é confiável, mas está em concordância com o teste Wilcoxon (que é não-paramétrico e portanto confiável neste caso). De acordo com estes resultados, existe uma forte possibilidade de que os tempos medidos para as atividades executadas com o RMine (variável x_R) sejam consistentemente menores que o das atividades executadas com consultas SQL (x_S). Assim sendo, rejeita-se H_0 e acredita-se estatisticamente que a hipótese H_1 é verdadeira: a execução de tarefas exploratórias com RMine é mais simples e eficiente que a execução das mesmas tarefas com consultas SQL sobre um SGBD relacional.

5.5 Conclusão do capítulo

Nesta seção foi demonstrado o potencial e eficiência da metodologia proposta e implementada no sistema RMine sob diferentes perspectivas. No primeiro momento deste capítulo, foi apresentado o banco de dados Tycho-USP – uma base de dados acadêmica consolidada a partir dos diversos sistemas da USP; as configurações de teste, e o ambiente visual do sistema RMine. A partir destes elementos, utilizou-se de um estudo de caso com várias tarefas exploratórias sobre os dados da base Tycho-USP para apresentar o potencial da análise visual exploratória no sistema RMine.

Logo a seguir, foram utilizados os dados coletados durante as várias sessões de exploração visual para compor um comparativo entre a abordagem implementada no sistema RMine e o uso de consultas SQL sobre um SGDB relacional previamente configurado. Este comparativo demonstrou a eficiência da representação do banco de dados como um grafo hieraquicamente particionado, implementado sobre uma estrutura de dados orientada a disco, a Graph-Tree. O protótipo implementado teve um desempenho superior ao SGBD relacional, podendo servir de base para estudos aprofundados sobre os conceitos de SuperGrafos e também de futuras implementações deste conceito.

E finalizando o capítulo, aplicou-se um estudo de usabilidade guiado por tarefas exploratórias, visando comparar o caráter interativo do sistema RMine com o processo usual de compor consultas SQL. Os resultados deste estudo mostram que o sistema RMine facilita a execução desse tipo de tarefas exploratórias, conseguindo com que os usuários finalizassem essas tarefas em intervalos de tempo consistentemente menores.

Capítulo

6 Conclusão

Neste trabalho foi definida e experimentada uma nova abordagem para analisar a estrutura, os dados e os relacionamentos definidos em bancos de dados relacionais. A presente solução é baseada na estrutura Graph-Tree orientada a disco e em algoritmos baseados no conceito de SuperGrafos (RODRIGUES et al., 2013), o que proporcionou um meio eficiente de armazenar, recuperar e calcular os relacionamentos entre os objetos contidos em uma base de dados relacional, funcionalidades chave para a metodologia apresentada.

Sobre a Graph-Tree orientada a disco foi definido um procedimento para ler e organizar as informações do banco de dados de acordo com um particionamento hierárquico de grafo baseado nas informações estruturais do esquema relacional do banco de dados, obtendo-se assim uma estruturação semântica nesse particionamento. A Graph-Tree, em seguida, foi utilizada para armazenar esse particionamento, agregando-se ao sistema RMine e formando assim um protótipo operacional para análise visual relacional.

Nos experimentos analíticos, estudos de caso demonstraram que a metodologia proposta tem potencial para responder perguntas de caráter agregativo sobre os relacionamentos das relações do banco, mantendo disponíveis todos os objetos envolvidos nesta consulta. Nos experimentos de desempenho, observou-se que a Graph-Tree orientada a disco responde mais rapidamente a consultas complexas sobre os dados, este fato se deve ao fato de que bancos de dados relacionais não são projetados para análise exploratória de dados, o que traz vários entraves quando são utilizados para esta finalidade. Finalmente, os experimentos de usabilidade demonstraram significativos ganhos no que se refere à execução de tarefas de análise de dados utilizando consultas SQL sobre um banco de dados relacional, e sobre o sistema RMine utilizando a exploração visual sobre um particionamento hierárquico estruturado.

Portanto, a abordagem visual proposta demonstrou ser intuitiva no que diz respeito à exploração visual, e se mostrou eficiente em termos de custo computacional. A exploração visual poupa o analista da necessidade de escrever consultas SQL complexas, enquanto o custo computacional é beneficiado pela eficiência da Graph-Tree orientada a disco para responder as consultas agregativas sobre os relacionamentos entre objetos. Estes resultados embasam a contribuição do trabalho afirmando a hipótese geral traçada no capítulo introdutório: de que é possível explorar e compreender bancos de dados relacionais bem mais rapidamente por meio de representações visuais interativas, que são baseadas em grafos hierarquicamente particionados sobre uma estrutura eficiente de armazenamento e recuperação de dados em disco.

Referências

ABELLO, J.; HAM, F. van; KRISHNAN, N. ASK-GraphView: A large scale graph visualization system. *IEEE TVCG*, v. 12, n. 5, p. 669–676, set. 2006. ISSN 1077-2626.

AGRAWAL, R. et al. *The Claremont Report on Database Research*. University of California at Berkeley, 2008. University of California at Berkeley. Disponível em: <http://db.cs.berkeley.edu/claremont-/claremontreport08.pdf>.

ANTHES, G. Happy birthday, RDBMS! *Comm. ACM*, ACM, New York, NY, USA, v. 53, n. 5, p. 16–17, maio 2010. ISSN 0001-0782.

ARCHAMBAULT, D.; MUNZNER, T.; AUBER, D. Tugging graphs faster: Efficiently modifying path-preserving hierarchies for browsing paths. *IEEE TVCG*, v. 17, n. 3, p. 276–289, mar. 2011. ISSN 1077-2626.

AUBER, D. et al. Multiscale visualization of small world networks. In: *Proceedings of the Ninth annual IEEE conference on Information visualization*. Washington, DC, USA: IEEE Computer Society, 2003. (INFOVIS'03), p. 75–81. ISBN 0-7803-8154-8. Disponível em: <http://dl.acm.org/citation.cfm?id=1947368-.1947385>.

BALZER, M.; DEUSSEN, O. Level-of-detail visualization of clustered graph layouts. *IEEE Transactions on Visualization and Computer Graphics*, p. 133–140, 2007.

CHEN, P. P.-S. The entity-relationship model – toward a unified view of data. *ACM Trans. Database Syst.*, ACM, New York, NY, USA, v. 1, n. 1, p. 9–36, mar. 1976. ISSN 0362-5915. Disponível em: <http://doi.acm.org/10.1145/320434.320440>.

CORMEN, T. H. et al. *Introduction to Algorithms*. 3rd. ed. Cambridge, Massachusetts; London, England: The MIT Press, 2009. ISBN 978-0-262-03384-8.

DAVIS, T. A.; HU, Y. The university of florida sparse matrix collection. *ACM Trans. Math. Softw.*, ACM, New York, NY, USA, v. 38, n. 1, p. 1:1–1:25, dez. 2011. ISSN 0098-3500. Disponível em: <http://doi.acm.org/10.1145/2049662.2049663>.

DŽEROSKI, S. Multi-relational data mining: an introduction. *SIGKDD Explor. Newsl.*, ACM, New York, NY, USA, v. 5, p. 1–16, July 2003. ISSN 1931-0145. Disponível em: <http://doi.acm.org/10.1145/959242-.959245>.

EADES, P. Drawing clustered graphs on an orthogonal grid. In: *Journal of Graph Algorithms and Applications*. [S.l.: s.n.], 1999. v. 3, n. 4, p. 3–29.

EADES, P.; FENG, Q.-W. Multilevel visualization of clustered graphs. In: NORTH, S. (Ed.). *Graph Drawing*. Springer Berlin / Heidelberg, 1997, (Lecture Notes in Computer Science, v. 1190). p. 101–112.

ISBN 978-3-540-62495-0. 10.1007/3-540-62495-3_41. Disponível em: <http://dx.doi.org/10.1007/3-540-62495-3_41>.

FEIGE, U. Coping with the np-hardness of the graph bandwidth problem. In: *Algorithm Theory - SWAT 2000*. Springer Berlin Heidelberg, 2000, (Lecture Notes in Computer Science, v. 1851). p. 10–19. ISBN 978-3-540-67690-4. Disponível em: <http://dx.doi.org/10.1007/3-540-44985-X_2>.

FORTUNATO, S. Community detection in graphs. *Physics Reports*, v. 486, n. 3–5, p. 75 – 174, 2010. ISSN 0370-1573. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0370157309002841>.

FRISHMAN, Y.; TAL, A. Dynamic drawing of clustered graphs. In: *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*. [S.l.: s.n.], 2004. p. 191 –198. ISSN 1522-404X.

GHONIEM, M.; FEKETE, J.-D.; CASTAGLIOLA, P. A comparison of the readability of graphs using node-link and matrix-based representations. In: *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*. [S.l.: s.n.], 2004. p. 17 –24. ISSN 1522-404X.

GRAY, J. et al. Data cube: a relational aggregation operator generalizing group-by, cross-tab, and sub-totals. In: *Data Engineering, 1996. Proceedings of the Twelfth International Conference on*. [S.l.: s.n.], 1996. p. 152 –159.

HAM, F. van; WIJK, J. J. van. Beamtrees: Compact visualization of large hierarchies. *Information Visualization*, v. 2, n. 1, p. 31–39, 2003. Disponível em: <http://ivi.sagepub.com/content/2/1/31.abstract>.

HAREL, D.; KOREN, Y. Graph drawing by high-dimensional embedding. In: GOODRICH, M.; KOBOUROV, S. (Ed.). *Graph Drawing*. Springer Berlin Heidelberg, 2002, (Lecture Notes in Computer Science, v. 2528). p. 207–219. ISBN 978-3-540-00158-4. Disponível em:<http://dx.doi.org/10.1007/3-540-36151-0_20>.

HENRY, N.; FEKETE, J. Matrixexplorer: a dual-representation system to explore social networks. *Visualization and Computer Graphics, IEEE Transactions on*, v. 12, n. 5, p. 677–684, 2006. ISSN 1077-2626.

HENRY, N.; FEKETE, J.; MCGUFFIN, M. Nodetrix: a hybrid visualization of social networks. *Visualization and Computer Graphics, IEEE Transactions on*, v. 13, n. 6, p. 1302–1309, 2007. ISSN 1077-2626.

HENRY, N.; FEKETE, J.-D. Matlink: Enhanced matrix visualization for analyzing social networks. In: BARANAUSKAS, C. et al. (Ed.). *Human-Computer Interaction – INTERACT 2007*. Springer Berlin Heidelberg, 2007, (Lecture Notes in Computer Science, v. 4663). p. 288–302. ISBN 978-3-540-74799-4. Disponível em: <http://dx.doi.org/10.1007/978-3-540-74800-7_24>.

HERMAN, I.; MELANCON, G.; MARSHALL, M. Graph visualization and navigation in information visualization: A survey. *Visualization and Computer Graphics, IEEE Transactions on*, v. 6, n. 1, p. 24 –43, jan-mar 2000. ISSN 1077-2626.

HUANG, W. Using eye tracking to investigate graph layout effects. In: *Visualization, 2007. APVIS '07. 2007 6th International Asia-Pacific Symposium on*. [S.l.: s.n.], 2007. p. 97–100.

KNOBBE, A. *Multi-Relational Data Mining*. Tese (Ph.D. Dissertation) — Faculteit Wiskunde en Informatica, Universiteit Utrecht, the Netherlands, 2004.

LANDESBERGER, T. von et al. Visual analysis of large graphs: State-of-the-art and future research challenges. *Computer Graphics Forum*, Blackwell Publishing Ltd, v. 30, n. 6, p. 1719–1749, 2011. ISSN 1467-8659. Disponível em: <http://dx.doi.org/10.1111/j.1467-8659.2011.01898.x>.

LESKOVEC, J.; KLEINBERG, J.; FALOUTSOS, C. Graphs over time: densification laws, shrinking diameters and possible explanations. In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. New York, NY, USA: ACM, 2005. (KDD '05), p. 177–187. ISBN 1-59593-135-X. Disponível em: <http://doi.acm.org/10.1145/1081870.1081893>.

LIMA, D. M.; JR., J. F. R.; TRAINA, A. J. M. Graph-based relational data visualization.*Proceedings of the 17th International Conference on Information Visualisation*, CPS, London, UK, jul. 2013.

LIMA, D. M.; JR., J. F. R.; TRAINA, A. J. M. Hierarchical graph techniques applied to database visualization. *Proceedings of the 10th CONTECSI International Conference on Information Systems and Technology Management*, TECSI, São Paulo, BR, jun. 2013.

MANIATIS, A. et al. Cpm: A cube presentation model for olap. In: KAMBAYASHI, Y.; MOHANIA, M.; Wöß, W. (Ed.). *Data Warehousing and Knowledge Discovery*. Springer Berlin Heidelberg, 2003, (Lecture Notes in Computer Science, v. 2737). p. 4–13. ISBN 978-3-540-40807-9. Disponível em: <http://dx.doi.org/10.1007/978-3-540-45228-7_2>.

MANIATIS, A. S. et al. Advanced visualization for olap. In: *ACM DOLAP '03*. New York, NY, USA: ACM, 2003. p. 9–16. ISBN 1-58113-727-3. Disponível em: <http://doi.acm.org/10.1145/956060.956063>.

MANSMANN, S.; SCHOLL, M. H. Exploring olap aggregates with hierarchical visualization techniques. In: *ACM SAC '07*. New York, NY, USA: ACM, 2007. p. 1067–1073. ISBN 1-59593-480-4. Disponível em: <http://doi.acm.org/10.1145/1244002.1244235>.

MARTINS, R. et al. Multidimensional projections for visual analysis of social networks. *Journal of Computer Science and Technology*, Springer Boston, v. 27, p. 791–810, 2012. ISSN 1000-9000. 10.1007/s11390-012-1265-5. Disponível em: <http://dx.doi.org/10.1007/s11390-012-1265-5>.

MILLER, G. A. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, American Psychological Association, Washington, DC, USA, v. 63, n. 2, p. 81–97, 1956. Disponível em: <http://dx.doi.org/10.1037/h0043158>.

MUNZNER, T. Exploring large graphs in 3d hyperbolic space. *Computer Graphics and Applications, IEEE*, v. 18, n. 4, p. 18 –23, jul/aug 1998. ISSN 0272-1716.

NEUMANN, P.; CARPENDALE, S.; AGARAWALA, A. Phyllotrees: phyllotactic patterns for tree layout. In: *Proceedings of the Eighth Joint Eurographics / IEEE VGTC conference on Visualization*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2006. (EUROVIS'06), p. 59–66. ISBN 3-905673-31-2. Disponível em: <http://dx.doi.org/10.2312/VisSym/EuroVis06/059-066>.

PAPADOPOULOS, C.; VOGLIS, C. Drawing graphs using modular decomposition. In: HEALY, P.; NIKOLOV, N. (Ed.). *Graph Drawing*. Springer Berlin / Heidelberg, 2006, (Lecture Notes in Computer Science, v. 3843). p. 343–354. ISBN 978-3-540-31425-7. 10.1007/11618058_31. Disponível em: <http://dx.doi.org/10.1007/11618058_31>.

POHL, M.; SCHMITT, M.; DIEHL, S. Comparing the readability of graph layouts using eyetracking and task-oriented analysis. In: *Proceedings of the Fifth Eurographics conference on Computational Aesthetics in Graphics, Visualization and Imaging* Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2009. (Computational Aesthetics'09), p. 49–56. ISBN 978-3-905674-17-0. Disponível em: <http://dx.doi.org/10.2312/COMPAESTH/COMPAESTH09/049-056>.

RAO, R.; CARD, S. K. The table lens: merging graphical and symbolic representations in an interactive focus + context visualization for tabular information. In: *ACM SIGCHI '94*. New York, NY, USA: ACM, 1994. p. 318–322. ISBN 0-89791-650-6. Disponível em: <http://doi.acm.org/10.1145/191666.191776>.

ROBERTSON, G.; CARD, S.; MACKINLAY, J. Cone-trees: Animated 3d visualization of hierarquical information. In: *ACM SIGCHI Intl. Conf. on Human Factors in Computing*. [S.l.]: ACM Press, 1991. p. 189–194.

RODRIGUES, J. et al. Large graph analysis in the GMine system. *IEEE TKDE*, v. 25, n. 1, p. 106 –118, jan. 2013. ISSN 1041-4347.

RODRIGUES JR., J. F. et al. GMine: a system for scalable, interactive graph visualization and mining. In: *Proceedings of the 32nd international conference on Very large data bases*. VLDB Endowment, 2006. (VLDB '06), p. 1195–1198. Disponível em: <http://dl.acm.org/citation.cfm?id=1182635.1164242>.

SATO, S.; MISUE, K.; TANAKA, J. Readable representations for large-scale bipartite graphs. In: LOVREK, I.; HOWLETT, R.; JAIN, L. (Ed.). *Knowledge-Based Intelligent Information and Engineering Systems*. Springer Berlin / Heidelberg, 2008, (Lecture Notes in Computer Science, v. 5178). p. 831–838. ISBN 978-3-540-85564-4. 10.1007/978-3-540-85565-1_103. Disponível em: <http://dx.doi.org/10.1007/978-3-540-85565-1_103>.

SCHULZ, H.-J.; HADLAK, S.; SCHUMANN, H. Point-based tree representation: A new approach for large hierarchies. In: *Visualization Symposium, 2009. PacificVis '09. IEEE Pacific*. [S.l.: s.n.], 2009. p. 81–88.

STASKO, J.; ZHANG, E. Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. In: *Information Visualization, 2000. InfoVis 2000. IEEE Symposium on*. [S.l.: s.n.], 2000. p. 57–65. ISSN 1522-404X.

STOLTE, C.; TANG, D.; HANRAHAN, P. Polaris: a system for query, analysis, and visualization of multidimensional relational databases. *IEEE TVCG*, v. 8, n. 1, p. 52–65, jan. 2002. ISSN 1077-2626.

STOLTE, C.; TANG, D.; HANRAHAN, P. Multiscale visualization using data cubes. *IEEE TVCG*, v. 9, n. 2, p. 176–187, abr. 2003. ISSN 1077-2626.

TANEMBAUM, A. S. *Modern Operating Systems*. 2. ed. Upper Saddle River, NJ, USA: Prentice Hall, 2007. ISBN 0-13-600663-9.

TECHAPICHETVANICH, K.; DATTA, A. Interactive visualization for OLAP. In: GERVASI, O. et al. (Ed.). *ICCSA 2005*. [S.l.]: Springer Berlin / Heidelberg, 2005, (Lecture Notes in Computer Science, v. 3482). p. 293–304. ISBN 978-3-540-25862-9.

TEKUSOVA, T.; SCHRECK, T. Visualizing time-dependent data in multivariate hierarchic plots - design and evaluation of an economic application. In: *Information Visualisation, 2008. IV '08. 12th International Conference*. [S.l.: s.n.], 2008. p. 143–150. ISSN 1550-6037.

THOMSEN, E. *Olap Solutions: Building Multidimensional Information Systems*. 2nd. ed. New York, NY, USA: John Wiley & Sons, Inc., 2002. ISBN 0471400300.

TOMINSKI, C.; ABELLO, J.; SCHUMANN, H. CGV - an interactive graph visualization system. *Computers & Graphics*, v. 33, n. 6, p. 660–678, 2009. ISSN 0097-8493. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0097849309000879>.

TSOURAKAKIS, C. E. et al. Doulion: counting triangles in massive graphs with a coin. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM, 2009. (KDD '09), p. 837–846. ISBN 978-1-60558-495-9. Disponível em: <http://doi.acm.org/10.1145/1557019.1557111>.

WANG, B. et al. ZoomTree: Unrestricted zoom paths in multiscale visual analysis of relational databases. In: RICHARD, P.; BRAZ, J. (Ed.). *Computer Vision, Imaging and Computer Graphics. Theory and Applications*. [S.l.]: Springer Berlin Heidelberg, 2011, (Communications in Computer and Information Science, v. 229). p. 299–317. ISBN 978-3-642-25382-9.

WARE, C. *Information Visualization, Third Edition: Perception for Design*. 3rd. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2012. ISBN 0123814642, 9780123814647.

WIJK, J. van; WETERING, H. Van de. Cushion treemaps: visualization of hierarchical information. In: *Information Visualization, 1999. (Info Vis '99) Proceedings. 1999 IEEE Symposium on*. [S.l.: s.n.], 1999. p. 73–78, 147. ISSN 1522-404X.

ZHAO, S.; MCGUFFIN, M.; CHIGNELL, M. Elastic hierarchies: combining treemaps and node-link diagrams. In: *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on.* [S.l.: s.n.], 2005. p. 57–64.