



UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"

Campus de Presidente Prudente

FCT - Faculdade de Ciências e Tecnologia
DMC - Departamento de Matemática e Computação
Bacharelado em Ciência da Computação

Guilherme Henrique Lourenço dos Santos

*Visualização de dados pluviométricos com
utilização de Banco de Dados não convencionais*

Revisão Bibliográfica

Orientador: Prof. Dr. Milton Hirokazu Shimabukuro

2016

Visualização de dados pluviométricos com utilização de Banco de Dados não convencionais

Autor: *Guilherme Henrique Lourenço dos Santos*

Orientador: *Prof. Dr. Milton Hirokazu Shimabukuro*

Responsável: *Prof. Dr. Celso Olivete Junior*

Disciplina: Trabalho de Conclusão de Curso I

Curso: Bacharelado em Ciência da Computação

Instituição: Universidade Estadual Paulista
Faculdade de Ciências e Tecnologia
Departamento de Matemática e Computação

Presidente Prudente, 10 de outubro de 2016.

Guilherme Henrique Lourenço dos Santos (aluno)

Prof. Dr. Milton Hirokazu Shimabukuro (orientador)

Prof. Dr. Celso Olivete Junior (responsável)

Resumo

Este trabalho apresenta a fundamentação teórica das áreas de estudo de Bases de Dados não convencionais e de Visualização de dados, com enfoque no uso de banco de dados NoSQL orientados a documento com aplicação no SGBD PostgreSQL atrelado ao uso de técnicas de visualização para dados espaciais e temporais.

palavras-chave: PostgreNOSQL. NOSQL. Visualização.

Sumário

1 Bancos de Dados Não Relacionais - NoSQL	p. 9
1.1 Definição	p. 9
1.2 Aspectos Comuns ao NoSQL	p. 10
1.2.1 Recapitulando SGBDRs	p. 10
1.2.2 ACID e BASE	p. 11
1.2.3 Teorema CAP	p. 12
1.3 Formas de Armazenamento	p. 12
1.3.1 Armazenamento baseado em Chave-Valor	p. 12
1.3.2 Armazenamento orientado a Documento	p. 13
1.3.3 Banco de Dados Orientado a Colunas	p. 14
1.3.4 Banco de Daods Orientado a Grafos	p. 15
2 PostgreSQL e os tipos de dados não convencionais	p. 16
2.1 PostgreSQL	p. 16
2.2 JSON	p. 17
2.3 BSON	p. 17
2.4 Json vs JsonB	p. 17
2.5 Uso no PostgreSQL	p. 20
2.5.1 Criação de Tabelas	p. 20
2.5.2 Realizando Inserções	p. 21

2.5.3	Execução de Queries	p. 21
3	Visualização de Dados	p. 22
3.1	Introdução	p. 22
3.2	Mineração e Visualização de Dados	p. 22
3.3	Técnicas de Visualização	p. 24
3.3.1	Técnicas Orientadas a Pixel	p. 24
3.3.2	Técnicas de Projeção Geométrica	p. 26
3.3.3	Técnicas Iconográficas	p. 27
3.3.4	Técnicas Hierárquicas	p. 27
3.3.5	Variação Temporal Uni-escala	p. 28
3.3.6	Variação Temporal Multi-escala	p. 29
	Referências	p. 30

Lista de Figuras

1	Exemplo de Chave e Valor em Redis (NÄSHOLM, 2012).	p. 13
2	Exemplo de um documento JSON em CouchDB (WEBER, 2010). . . .	p. 14
3	Exemplo de um pequeno banco de dados do cliente. Pode ser visto que as linhas têm chaves únicas que os identifica e que cada linha pode definir seu próprio conjunto de colunas. Note-se que esta é uma imagem simplificada; timestamps são omitidos, e todos os campos são realmente uma sequência de bits. (NÄSHOLM, 2012).	p. 14
4	Exemplo de um grafo Neo4j armazenando os dados de 3 clientes. Todos os clientes são alcançáveis a partir do nó vazio. A imagem é um recorte da tela da ferramenta de visualização Neoclipse. (NÄSHOLM, 2012). .	p. 15
5	Performance relativa de BSON e JSON para aumento de tamanho do documento sobre número constante de tuplas (WHITTAKER, 2013). .	p. 18
6	Performance relativa de BSON e JSON para aumento do número de tuplas sobre documento de tamanho estático(WHITTAKER, 2013). . .	p. 19
7	Comparação do fator do aumento de velocidade para chaves perto do começo do documento e perto do fim do documento (WHITTAKER, 2013).	p. 19
8	Comparação do fator do aumento de velocidade para documentos contendo exclusivamente um tipo de dado (WHITTAKER, 2013).	p. 20
9	Comparação do total de tempo do carregamento entre JSON e BSON para 1.000, 10.000, 100.000 inserções (WHITTAKER, 2013).	p. 20
10	Passos do processo KDD (SHIMABUKURO, 2004) adaptado de (FAYYAD et al., 1996).	p. 23

11	Múltiplas janelas para um caso de 6 atributos em técnicas orientadas a pixel. (SHIMABUKURO, 2004) adaptado de (KEIM, 2000).	p. 25
12	Possibilidades de disposição dos pixels nas técnicas orientadas a pixel. (KEIM; KRIEGEL, 1996 apud SHIMABUKURO, 2004).	p. 25
13	Formato circular para apresentação dos pixels em Segmentos de Círculo. (SHIMABUKURO, 2004) adaptado de (KEIM, 2000).	p. 25
14	Esquema de arranjo dos pixels em Gráfico de Barras de Pixels. (SHIMABUKURO, 2004) adaptado de (KEIM et al., 2001).	p. 26
15	Correlação e distribuição de valores em Coordenadas Paralelas. (WARD, 1994 apud SHIMABUKURO, 2004).	p. 26
16	Sobreposições de linhas com apresentação de 473 itens. (INSELBERG, 1997 apud SHIMABUKURO, 2004).	p. 27
17	Configuração para Figura de Arestas usando cinco arestas (KEIM; KRIEGEL, 1996 apud SHIMABUKURO, 2004).	p. 27
18	Diferentes texturas representadas por Figura de Arestas. (GRINSTEIN; TRUTSCHL; CVEK, 2001 apud SHIMABUKURO, 2004).	p. 28
19	Resultado do empilhamento dimensional. Quatro atributos são aninhados aos pares (HOFFMAN, 1977 apud SHIMABUKURO, 2004).	p. 28
20	Construção da representação visual ‘Variação Temporal Uni-escala’.(SHIMABUKURO, 2004).	p. 29
21	Construção da representação visual ‘Variação Temporal Multi-escala’.(SHIMABUKURO, 2004).	p. 29

Lista de Tabelas

1	Operadores de json e jsonb (POSTGRESQL, 1996)	p. 21
---	---	-------

Bancos de Dados Não Relacionais - NoSQL

Em um trabalho acadêmico intitulado "Selected Topics on Software-Technology Ultra-Large Scale Sites" (Tópicos Seleccionados na Tecnologia de Softwares de Sites de Escala Ultra Grande) (STRAUCH; SITES; KRIHA, 2011) na Stuttgart Media University (Universidade de Mídia de Stuttgart) Christof Strauch reúne várias referências e tópicos sobre todo o assunto de base do NoSQL, conforme seu trabalho, os bancos de dados relacionais são os tipos predominantes para armazenamento de dados estruturados, desde 1970 este tipo de base de dados se suporta em uma aritmética relacional com o proliferado uso do SQL (Secure Query Language) para inserir, alterar, recuperar e remover os dados. Bases de dados não convencionais eram utilizadas apenas para aplicações de algum nicho específico como projetos utilizando SOAP que utiliza o padrão XML, porém o pensamento por cima desse uso disseminado "um tamanho para caber todos" tem sido questionado tanto pelas empresas como pelos acadêmicos devido ao aumento de usuários e volume de dados no meio.

Conforme segue na argumentação de Christof Strauch, para suprir essa demanda algumas alternativas começam a ser propostas, vários outros modelos são criados para abranger mais nichos antes dominados pelos SGBDRs, esses modelos adotam caminhos não convencionais nem sempre garantindo a ACID (Atomicidade, Consistência, Isolamento e Durabilidade) dos dados, porém conseguem otimizar o espaço de armazenamento e a performance das bases de dados, não necessariamente se desfazendo do uso do SQL, por isso a sigla NoSQL significa "*not only secure query language*".

1.1 Definição

Existem dois pontos de vista não equivalentes sobre o princípio do NoSQL, o NoSQL Archive (ARCHIVE, 2009) apresenta uma visão seguindo o termo "NonSQL" que significa

uma base de dados sem SQL em tudo, no artigo "NoSQL Databases" (Base de Dados NoSQL) (WEBER, 2010) Silvan Weber define outro ponto de vista mais aceito sendo "Not only SQL" (Não apenas SQL), ou seja, base de dados que não é baseada em SQL porém, pode ter ou não funcionalidades SQL. Além do conceito presente no nome, Weber levanta outras características fundamentais:

- **Armazenamento Distribuído:** Um dos maiores intuítos e motivador para o desenvolvimento de bases NoSQL é a necessidade de distribuir a carga de armazenamento e acessos das bases a vários servidores porém, mesmo dividido em vários servidores, a base continua interagindo com os outros sistemas como uma.
- **Fonte aberto:** Esse princípio assegura que o código fonte de bases NoSQL estejam disponíveis para que possam ser alterados para melhor se adaptar a necessidade de cada aplicação, diferentemente de SGBDRs comerciais que, além de serem muito onerosos, nem sempre atendem aos requisitos das aplicações
- **Escalável horizontalmente:** Uma das necessidades de ter um armazenamento distribuído é para que quanto mais servidores ligados à base maior será a performance da base por completo, seguindo uma linha de proporção próxima do linear.

1.2 Aspectos Comuns ao NoSQL

Esta seção destina-se a criar uma base sólida sobre os principais aspectos e conceitos que envolvem o NoSQL, além de apontar diferenças e temas relacionados com SGBDRs também serão abordados teoremas e tópicos específicos sobre o NoSQL

1.2.1 Recapitulando SGBDRs

Em uma tese de mestrado intitulada "Extracting Data from NoSQL Databases. A Step towards Interactive Visual Analysis of NoSQL Data" (Extração de dados de bases NoSQL. Um passo avante para análise de visualização interativa de NoSQL) (NÄSHOLM, 2012) da Chalmers University of Technology (Chalmers, Universidade de Tecnologia) Petter Näsholm apresenta uma recapitulação rápida sobre as bases de dados convencionais. SGBDRs se baseiam no conceito de tabelas com conjunto de atributos podendo ser chamados de colunas da tabela, cada atributo é associado a um tipo de dado (como inteiro, real ou string). Cada linha desta tabela é considerada uma tupla, ao passo que os atributos dentro de cada tupla podem ser preenchidos com componentes referente ao seu tipo. As tabelas também podem ter colunas com informações sobre a relação entre elas mesmas.

1.2.2 ACID e BASE

Petter Näsholm também aborda as garantias que a maioria dos SGBDRs adotam sendo elas a sigla ACID(Atomicidade, Consistência, Isolação, Durabilidade) onde Atomicidade é a garantia de que ou uma transação é executada de forma completa ou não é executada de forma alguma, a Consistência garante que toda transação saia de um estado da base válido para outro, Isolação é a garantia de que não interfere outra no tempo em que estão ocorrendo e a Durabilidade garante que o efeito de uma transação deve persistir e, portanto, nunca ser perdido. No trabalho de Christof Strauch o autor apresenta o BASE que são as garantias dos NoSQL:

Basic available(Basicamente disponível): Uma aplicação funciona basicamente todo o tempo.

Soft-state(Estado "leve, macio", simples): Não precisa ser consistente o tempo todo.

Eventual consistency(Eventual consistência): Em algum momento conhecido haverá consistência.

O aspecto com maior diferença entre os modelos é a parte da consistência, no modelo ACID existe uma consistência estrita enquanto no base há uma consistência eventual. A consistência estrita implica que toda operação de leitura deve retornar resultados a partir da última escrita, devido a isso as operações de leitura e escrita devem ser feitas no mesmo servidor ou obedecer a um protocolo de transação distribuída, porém, como veremos abaixo pelo teorema CAP, essa forma de operar implica ou na perda da disponibilidade dos dados ou na tolerância do particionamento dos mesmos. A consistência eventual permite que leitores acessem a base conforme os dados vão sendo escritos, não importando necessariamente se aquele registro foi o último a ser escrito, eventualmente em um estado estável será retornado o último registro escrito, até isso ocorrer o sistema pode estar em um estado de inconsistência.

Além das garantias colocadas acima Christof Strauch lista outras que também podem ser fornecidas:

RYOW - Read Your Own Writes(leia seus próprios escritos): O leitor deve visualizar imediatamente aquilo que ele próprio escreveu, independentemente se o leitor acessa pelo próprio servidor que escreveu ou por outro.

Consistência da Sessão: Acompanha a garantia do RYOW porém apenas para a sessão que o leitor estiver conectado, ou seja, caso escreva os dados por um servidor e tentar ler por outro a consistência não será garantida.

Consistência ocasional: se foi escrito y após ser lido x, qualquer leitor que ver y também verá x.

1.2.3 Teorema CAP

O Teorema CAP que, conforme Christof Strauch, é muito adotado pela comunidade NoSQL, apresenta um conceito referente as limitações estritamente associadas as capacidades que oferecem.

A sigla CAP se refere à Consistência(Consistency), Disponibilidade(Availability), Tolerância ao particionamento(Partition Tolerance), sendo respectivamente a habilidade de manter os dados compartilhados disponíveis a todos os leitores após um escritor executar um update, a habilidade de continuar disponível mesmo com a falta de algum cluster ou parte do sistema ficar indisponível e a habilidade de continuar operando mesmo que um servidor não consiga se conectar a outro servidor da mesma aplicação também entendido como a habilidade de poder adicionar ou remover nós de uma rede particionada. Este Teorema propõe que só é possível para qualquer aplicação combinar duas dentre as três características, ou seja ela pode ter muita consistencia e disponibilidade porém com pouca tolerância ao particionamento, muita consistencia e tolerância ao particionamento porém com pouca disponibilidade ou muita disponibilidade e tolerância ao particionamento e pouca consistência.

Observando pelo teorema CAP os NoSQL buscam ofertar alternativas para aplicações que nem sempre dependem de consistência ou disponibilidade ou tolerância ao particionamento.

Também existem outros aspectos mais profundos mas que não serem abordados neste documento como o versionamento do conjunto dos dados, os cenários distribuídos e o particionamento. Os aspectos vistos acima nos dão aparato suficiente para progredir o trabalho focando nas questões principais sobre as necessidades da aplicação e o uso de diferentes tipos de SGBDs para qual se encaixam melhor.

1.3 Formas de Armazenamento

Nesta seção serem abordados alguns tipos de bases NoSQL, o modelo orientado à documento utilizando JSON será aprofundado pois foi escolhido para dar suporte a este trabalho devido a sua fácil integração com a ferramenta de visualização de dados escolhida.

1.3.1 Armazenamento baseado em Chave-Valor

Bases de dados Chave-Valor são extremamente simples, existe uma chave única, geralmente gerada por algum algoritmo de Hash, associada a um valor que pode ser de diferentes tipos como: string, inteiros, reais, cadeias de bytes. Algumas bases de dados

não implementam nem a verificação de tipo.

Tem a vantagem de ter um modelo de dados simples, feito com foco na rápida e eficiente manipulação dos dados para armazenamento distribuído sendo totalmente escalável. Alguns de seus exemplos: Google's BigTable¹, Memcached², Redis³.

```
(customerIds,           [12, 67])
(customerId:12:name,    Smith)
(customerId:12:location, Seattle)
(customerId:67:name,    Andersson)
(customerId:67:location, Stockholm)
(customerId:67:clubCard, 82029)
```

Figura 1: Exemplo de Chave e Valor em Redis (NÄSHOLM, 2012).

Algumas bases são construídas para funcionarem na memória RAM, como é o caso do Memcached, isso permite uma performance muito superior porém implica em um limite de espaço reduzido e o risco de perda de dados no caso de quedas de energia.

1.3.2 Armazenamento orientado a Documento

Este tipo de armazenamento pode ser entendido como um caso particular de uma base Chave-Valor, a diferença é que a base deve ler e escrever em algum tipo conhecido de arquivo e(ou) estrutura. Utiliza documentos com diferentes tipos de conjuntos de dados, alguns tipos de documento são de fácil interpretação para humanos, dentre eles: XML, JSON, XAML. Também é possível acessar e alterar diretamente tais documentos pelo servidor.

O esquema que existe nas bases é a estrutura utilizada no próprio arquivo, sem relação nenhuma dentre os documentos da base, cada documento funciona independente do outro, no caso de bases que os arquivos dependem são chamadas de semi-estruturadas.

Como a estrutura dos arquivos é algo arbitrário campos sem dados não precisam ser preenchidos e nenhum espaço de armazenamento precisa ser reservado. Dentre os tipos existentes estão: CouchDB⁴ e MongoDB⁵

¹<https://cloud.google.com/bigtable/>, acessado em 16 de Maio de 2016

²<https://memcached.org/>, acessado em 22 de Maio de 2016

³<http://redis.io/>, acessado em 25 de Maio de 2016

⁴<http://couchdb.apache.org/>, acessado em 13 de Abril de 2016

⁵<https://www.mongodb.com/>, acessado em 13 de Abril de 2016

```

{
  "_id": "a26ab886a7d200a93a32ae192200416d",
  "_rev": "4-c97d88dfa8c79468bcbe03fcf0e5921d",
  "another_attribute": 17,
  "yet_another_attribute": "a_string",
  "_attachments": {
    "Screenshot.png": {
      "content_type": "image/png",
      "revpos": 3,
      "length": 22666,
      "stub": true
    },
    "trace.nam": {
      "content_type": "application/octet-stream",
      "revpos": 2,
      "length": 7830,
      "stub": true
    }
  }
}

```

Figura 2: Exemplo de um documento JSON em CouchDB (WEBER, 2010).

1.3.3 Banco de Dados Orientado a Colunas

Diferentemente de bases SQL onde o conjunto de dados é armazenado em uma unidade, este modelo armazena um atributo do conjunto de dados em uma unidade, exemplo: Supondo termos 3 atributos nome, idade, telefone, com o conjunto de dados "Julio", 18, 33821522; "Lucas", 20, 82331248; "Marina", 25, 39031169. Organizando estes dados no formato orientado a colunas eles ficam nesta disposição: "Julio", "Marina", "Lucas"; 18, 20, 25; 33821522, 82331248, 39031169.

O uso dos atributos separados por coluna garante uma rápida análise dos dados mesmo em grande quantidade, pois estão descritos sucessivamente. Então análise como obter a média retornam rapidamente sem percorrer outros atributos desnecessários. As bases mais conhecidas são: HBase¹ e Cassandra²

Row keys	Columns			
0	name	location	clubCard	points
	Müller	Zürich	48 551	7,946
12	name	location		
	Smith	Seattle		
67	name	location	clubCard	
	Andersson	Stockholm	82 029	

Figura 3: Exemplo de um pequeno banco de dados do cliente. Pode ser visto que as linhas têm chaves únicas que os identifica e que cada linha pode definir seu próprio conjunto de colunas. Note-se que esta é uma imagem simplificada; timestamps são omitidos, e todos os campos são realmente uma sequência de bits. (NÄSHOLM, 2012).

¹<https://hbase.apache.org/>, acessado em 23 de Junho de 2016

²<http://cassandra.apache.org/>, acessado em 24 de Junho de 2016

1.3.4 Banco de Daods Orientado a Grafos

A teoria dos grafos serve como suporte para a elaboração dessa base onde, cada vértice representa uma entidade com dados, e as arestas são as relações entre os dados. Pode-se também utilizar o fator da direcionalidade das arestas do grafo como parte do conjunto de dados, no caso do Twitter¹, por exemplo, poderia ser utilizado esse recurso, supondo existirem "João" e "Maria", "João" segue maria no Twitter, ou seja, tem uma aresta na direção de "João" para "Maria" simbolizando essa relação, Caso "Maria" também seguisse "João" haveria outra aresta na direção de "Maria" para "João".

Sua aplicação é muito utilizada em redes sociais e serviços de transito, aproveitando recursos de encontrar caminhos mais curtos entre nós utilizando algoritmos da teoria dos grafos como o de Dijkstra. Além disto é altamente escalável horizontalmente replicando vértices para aumentar a performance e novamente utilizando a teoria dos grafos para particionálo tornando os caminhos das buscas ainda mais curtos. Exemplos de bases: Neo4j² e FlockDB³

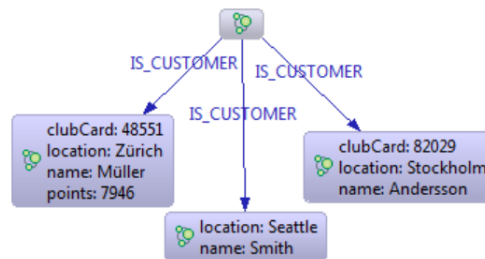


Figura 4: Exemplo de um grafo Neo4j armazenando os dados de 3 clientes. Todos os clientes são alcançáveis a partir do nó vazio. A imagem é um recorte da tela da ferramenta de visualização Neoclipse. (NÄSHOLM, 2012).

¹<https://twitter.com/>, acessado em 30 de Junho de 2016

²<https://neo4j.com/>, acessado em 30 de Junho de 2016

³<https://blog.twitter.com/2010/introducing-flockdb>, acessado em 30 de Junho de 2016

Capítulo 2

PostgreSQL e os tipos de dados não convencionais

2.1 PostgreSQL

PostgreSQL(POSTGRESQL, 1996) é um sistema de banco de dados relacional poderoso e open source. Com mais de 15 anos de desenvolvimento tem uma reputação forte sobre sua confiabilidade, integridade dos dados e correção. Ele é executado em todos os principais sistemas operacionais, incluindo Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) e Windows. É totalmente compatível com ACID, tem suporte completo para chaves estrangeiras, joins, views, triggers e procedures. Inclui a maioria dos tipos de dados SQL:2008: INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVALO e TIMESTAMP. Também suporta o armazenamento de grandes objetos binários, incluindo imagens, sons e vídeo. Tem interfaces de programação nativas para C / C ++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, entre outros.

Recentemente também tem adotado algumas funcionalidades NoSQL buscando fornecer mais alternativas reunidas em uma base só, evitando que o serviço tenha que usar uma base SQL para parte de sua funcionalidade e outra base NoSQL para outro grupo de funcionalidades. Dentre elas o suporte orientado a documento(json e xml) e do tipo Chave-Valor(hstore).

Sobre os dados sem schema que utiliza, o Xml é estruturado de maneira hierarquica sendo o padrão W3C standard 1998 o padrão é base para o protocolo de troca de mensagens SOAP, porém seu uso é um pouco complexo e seu desempenho de velocidade é o pior dos dados em armazenamento em documentos postgresql, sem indexação.

Outro tipo é o Hstore, este tipo tem um desempenho bem expressivo pois baseia-se

apenas em chave para valores em strings ou outros valores hstore, embora seja eficiente a falta de aninhamento e diferentes tipos dificulta sua utilização para situações mais complexas, não é muito flexível.

Por último temos o tipo JSON e JSONB, que é o tipo Orientado a Documento baseado na notação JavaScript, é utilizado como forma para armazenamento e troca de dados entre várias plataformas web e mobile. Abaixo temos um aprofundamento sobre este tipo por ser base ao trabalho.

2.2 JSON

JSON significa "JavaScript Object Notation" (INTERNATIONAL, 2013), é um formato de dados de fácil leitura e escrita para os humanos e de fácil geração e análise para as máquinas. Um objeto em JSON é um conjunto de pares de chave e valor que podem ser organizados de forma aninhada e(ou) listados em arrays, a chave é uma string e o valor pode ser uma string, um numero, um booleano, um array e até mesmo outro objeto aninhado. O array pode ter valores dos mesmos tipos citados. Pode haver também uma lista de objetos no mesmo nível, esta lista pode obedecer ou não a sua ordem.

2.3 BSON

O termo BSON é uma junção de Binary JSON, ele é a serialização de documentos JSON para binário, foi desenvolvido para melhorar o desempenho na busca das chaves. Além de armazenar os dados em arquivos binários a maior diferença entre ele e o JSON é o uso de tipos de dados com definição de tamanho e tipo pré-determinadas, para armazenar um número inteiro o JSON utiliza uma cadeia de caracteres arbitrária enquanto o BSON utiliza 32 e 64 bits para inteiros e 64 bits para double e floats, tipos booleanos em JSON são representados pela cadeia de caracteres "True" ou "False", em BSON é representado por apenas um bit 0 ou 1. Para valores como cadeias de strings são armazenadas meta-informações como o comprimento dela e, como os outros tipos já tem comprimentos definidos, um parsing na busca por uma chave pode facilmente saltar para os pontos em que interessa no documento.

2.4 Json vs JsonB

Antes do JSONB fazer parte do PostgreSQL Geoffrey Litt desenvolveu em seu projeto final "Improving performance of schemaless document storage in PostgreSQL using

BSON” (Aprimorando a performance de armazenamento orientado a documento em PostgreSQL utilizando BSON) (WHITTAKER, 2013) da Yale University (Universidade de Yale), através do fonte do PostgreSQL, bibliotecas de parsing JSON e bibliotecas de parsing BSON do MongoDB, um campo similar chamado BSON, em sua tese pode testar as diferenças de performance destes campos. Posteriormente o PostgreSQL adotou este campo.

Para o primeiro teste utilizou o mesmo documento e variando seu tamanho de 1 à 90 vezes, nele executava uma query simples para encontrar um valor no meio do documento, o teste demonstrou que quanto maior o documento maior era a velocidade do BSON sobre o JSON, com o tamanho máximo o BSON foi 8.28 vezes mais rápido para o JSON.

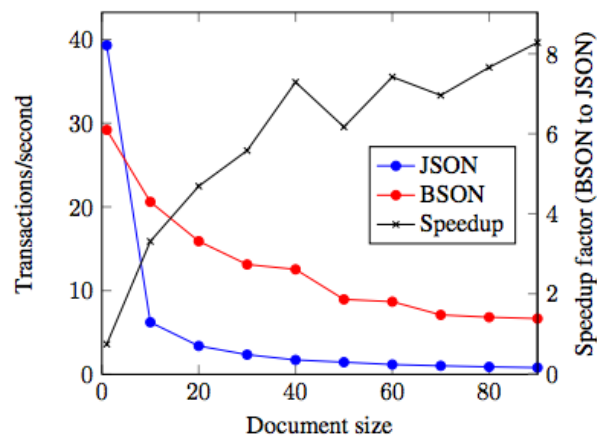


Figura 5: Performance relativa de BSON e JSON para aumento de tamanho do documento sobre número constante de tuplas (WHITTAKER, 2013).

No segundo teste utilizou o mesmo tamanho de documentos mas aumentou o número de tuplas, nele pode constatar que a velocidade de BSON sobre o JSON aumentava com o número de tuplas porém após o número de 100 tuplas essa velocidade começa a convergir para um valor constante de 6 vezes em relação ao JSON.

Outro teste foi realizar queries por uma chave variando seu lugar do começo ao fim do documento e variando o tamanho do documento, a velocidade do bson foi aproximadamente 4 vezes maior em relação ao json para todos os tamanhos de documentos com chaves próximas do começo do documento, para chaves próximas do fim do documento a velocidade do bson foi aproximadamente 6 vezes maior em relação ao json para todos os tamanhos de documentos, isso deve-se à capacidade que o bson tem de atravessar rapidamente as chaves por ter meta-informações sobre seu comprimento enquanto o json tem que analisar caracter a caracter.

Foi realizado também um teste sobre a composição do documento, ou seja, os tipos de dados dentro do documento, executando queries em 3 documentos, o primeiro com apenas strings de 10 caracteres, o segundo apenas com inteiros e o terceiro com apenas booleanos, para strings a velocidade do bson foi 3 vezes superior sobre o json, para inteiros e booleanos

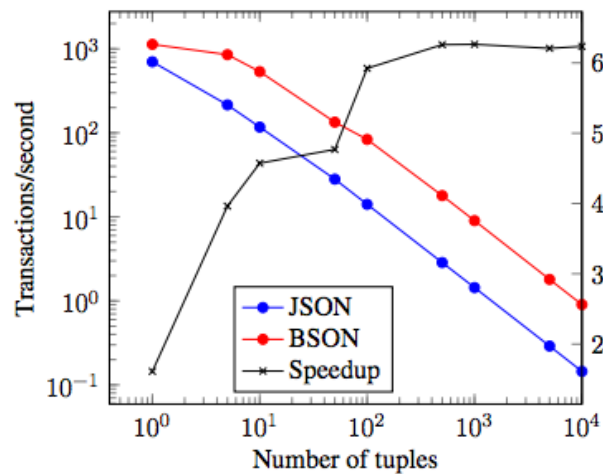


Figura 6: Performance relativa de BSON e JSON para aumento do número de tuplas sobre documento de tamanho estático(WHITTAKER, 2013).

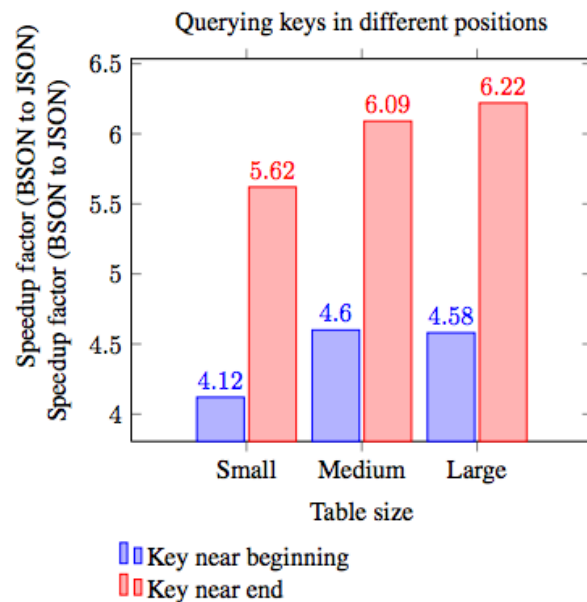


Figura 7: Comparação do fator do aumento de velocidade para chaves perto do começo do documento e perto do fim do documento (WHITTAKER, 2013).

a velocidade do bson foi aproximadamente 5,5 vezes superior ao json, o resultado superior de inteiros e booleanos deve-se ao fato do bson utilizar inteiros de 32bits e booleanos de 1 bit, como foi comentado acima, e o json utilizar cadeias de caracteres para representá-los.

Por último foi realizado um teste para verificar o tempo de carregamento dos arquivos em disco, como o bson precisa de um passo a mais no parsing transformando os valores de strings para os tipos de inteiros, booleanos, etc, criando os metadados sobre strings e comprimento das chaves, seu tempo de carregamento é pouco mais demorado em relação ao json, como pode-se verificar na figura 9, entretanto para esse teste foi utilizado o parsing de validação do json e posteriormente foi salvo em disco, caso tivesse sido usado um parsing

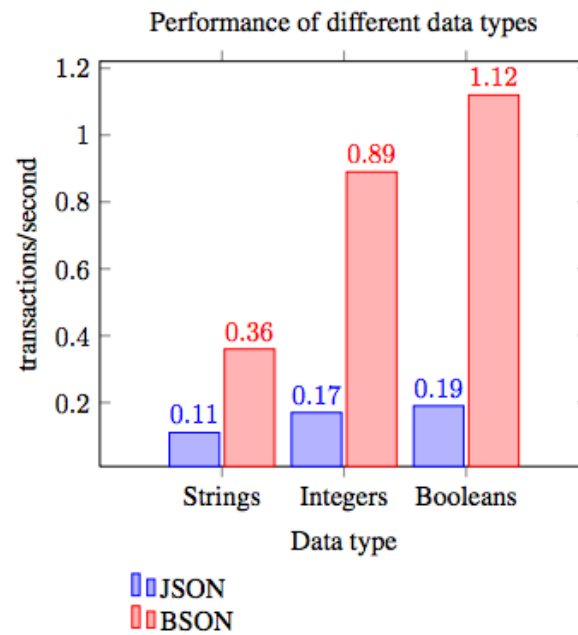


Figura 8: Comparação do fator do aumento de velocidade para documentos contendo exclusivamente um tipo de dado (WHITTAKER, 2013).

de validação já em bson essa diferença poderia se igualar ou até mesmo inverter-se.

	1,000 tuples	10,000 tuples	100,000 tuples
JSON	6.100s	62.09s	630000s
BSON	6.139s	62.95s	635600s

Figura 9: Comparação do total de tempo do carregamento entre JSON e BSON para 1.000, 10.000, 100.000 inserções (WHITTAKER, 2013).

2.5 Uso no PostgreSQL

Nesta seção serão abordadas as definições técnicas para utilizar a sintaxe de operadores para executar as tarefas das seções abaixo em campos Json e Jsonb do PostgreSQL, funções que encapsulam funcionalidades não serem abordadas.

2.5.1 Criação de Tabelas

Os tipos Json e Jsonb podem ser criados como um atributo qualquer de qualquer tabela, basta identificar o tipo após o nome do campo:

```
CREATE TABLE eventos (
  nome varchar(200),
  attrJson json,
```

```
attrBinario jsonb
```

```
);
```

2.5.2 Realizando Inserções

A inserção também ocorre de maneira natural como em outros atributos da tabela, a diferença nesse caso é o conteúdo da coluna:

```
INSERT INTO eventos values('Show de Sábado', '{"lista de convidados": ["João",
"Maria", "Roberto"]}', '{"lista de intrusos": []}');
```

2.5.3 Execução de Queries

Para execução de queries foram criados operadores adicionais que podem ser utilizados juntos do comando SELECT. Operadores de json e jsonb na tabela

Operador	Operando à Direita	Descrição	Exemplo	Resultado do Exemplo
->	int	Retorna o elemento do array JSON	'["a","b","c"]::json->2	c
->	texto	Retorna o objeto da chave	'{"a": [1,2,3]}::json->'a'	[1,2,3]
->>	int	Retorna o elemento do array JSON como texto	'["a","b","c"]::json->>1	"b"
->>	texto	Retorna o objeto da chave como texto	'{"a": [1,2,3]}::json->>'a'	"[1,2,3]"
#>	texto[]	Retorna o objeto JSON no caminho	'{"a": {"b":2}}::json#>'a,b'	2
#>>	texto[]	Retorna o objeto JSON no caminho como texto	'{"a": {"b":2}}::json#>>'a,b'	'2'

Tabela 1: Operadores de json e jsonb (POSTGRESQL, 1996)

Capítulo 3

Visualização de Dados

3.1 Introdução

O Trabalho de Conclusão de Curso de Genghini (GENGHINI, 2011) lista uma coleção de técnicas de visualização que serviram de base para este, principalmente as de Visualização de Múltiplas Coordenadas (SHIMABUKURO, 2004). Ele começa introduzindo a quantidade enorme e crescente de dados sendo armazenados e como é importante definirmos técnicas para que possamos acessar as informações que tais dados possam apresentar pois "Tais dados são potencialmente inúteis, um simples depósito volumoso, se não houver formas adequadas de processamento dos mesmos" (SHIMABUKURO, 2004).

Informações como dados espaciais de coordenadas geográficas, observando a distância entre dados, a densidade dos mesmos e o comportamento ao longo do tempo ajudam a compreender fenômenos de vários domínios como turismo, negócios, saúde pública e saneamento, meio ambiente e clima.

Então, para podermos analisar tais dados a ponto de obtermos informações relevantes que permitam o traçar de estratégias são necessárias técnicas tanto para organizar tais dados e para apresentá-los que serão vistas nas seções que seguem.

3.2 Mineração e Visualização de Dados

Técnicas que dão suporte para extrair de forma automática ou semi-automática informações relevantes de enorme volume de dados são atribuídas a 'Mineração de Dados' (DM, Data Mining) (KEIM; KRIEGEL, 1995 apud SHIMABUKURO, 2004) (FAYYAD, 1998 apud SHIMABUKURO, 2004) área de estudo que, para alguns é sinônimo de e para outros faz parte do processo de 'Descoberta de Conhecimento em Base de Da-

dos'(KDD; Knowledge Discovery in Databases)(FAYYAD, 1998 apud SHIMABUKURO, 2004)(FAYYAD et al., 1996 apud SHIMABUKURO, 2004).

O processo de descoberta de conhecimento é descrito pelas fases, em sequência de: Seleção(é selecionado um conjunto de dados, destes são selecionados itens dos dados e o foco em um subconjunto de atributos ou retira-se uma amostra), Pré-processamento(remoção de dados estranhos ou corrompidos, remoção de informações inexistentes na base e ordenação seguindo algum atributo), Transformação(refinamento diminuindo o número de variáveis a serem consideradas, ou a divisão dos mesmos em partes menores), Mineração(aplicação de operações a fim de reconhecer padrões entre os dados refinados), Interpretação/avaliação(os padrões extraídos são avaliados podendo levar a uma repetição do processo. Caso aceita, as informações estão prontas para serem utilizadas).

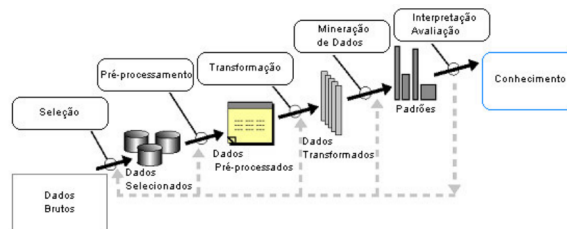


Figura 10: Passos do processo KDD (SHIMABUKURO, 2004) adaptado de (FAYYAD et al., 1996).

Descrevendo melhor a etapa da Mineração de Dados podemos tomá-la como sendo o mapeamento dos dados referentes a alguma informação, sendo a validade das mesmas atreladas ao domínio da aplicação e dos objetivos do usuário segundo (FAYYAD et al., 1996 apud SHIMABUKURO, 2004). Pode ser utilizada para prever novas entradas de dados baseado nos dados já existentes como parâmetro, extrair informações por critérios pré-estabelecidos informando o tipo da informação ou de dado entre outros modelos como os estatísticos e de aprendizagem de máquina. Chen(CHEN; HAN; YU, 1996 apud SHIMABUKURO, 2004) apresenta categorias baseadas no tipo de informação tomada dos dados:

- Regras de Associação: Procura desvendar o relacionamento entre os atributos se existe uma implicância da presença de um posto a presença do outro, associando-os. A procura utiliza os parâmetros de confiança c e suporte s , onde uma regra de associação “se existe X então existe Y” tem uma confiança c e um suporte s , a porcentagem de c significando o quanto ocorre esta relação e a porcentagem de s o quanto X ou Y aparecem nos dados.
- Classificação de dados: Os dados são separados em classes definidas pelos conteúdos de alguns atributos, criadas de um subconjunto de treinamento

- Agrupamento (Clustering) de Dados: Os dados são agrupados sem uma pré-separação por classes, denominada classificação não supervisionada. Os agrupamentos são feitos dando preferência a similaridades de atributos dentro da classe do que entre as classes identificando, de acordo com medidas de distância, regiões mais povoadas sendo estas os *clusters*.
- Pesquisa por similaridade baseada em padrões: Pesquisa por atributos similares espaciais e temporais identificando risco, causalidade e tendências associadas ao padrão. Sendo estas informações importantes para diversas áreas financeira, meteorológica e médica. Tais atributos não precisam ser exatamente idênticos mas ter algum nível de similaridade.

Além da necessidade de extrair informações relevantes de bases de dados é importante que exista um meio para apresentá-las, para isso são atreladas técnicas de Visualização de Dados, a qual “reúne métodos que permitem interagir com representações visuais de dados, exibidas como imagens, para ganhar entendimento e introspecção sobre eles” (RHYNE, 2003 apud SHIMABUKURO, 2004). com o uso das técnicas conjuntas surge o termo ‘Mineração Visual de Dados’(VDM, Visual Data Mining), técnicas que buscam unir o refinamento da Mineração de Dados a Técnicas de Visualização com o objetivo de extrair um número ainda maior de informações, aumentando a “descoberta de conhecimento”(WONG, 1999 apud SHIMABUKURO, 2004), adicionando a cognição humana a este processo.

3.3 Técnicas de Visualização

Em sua tese de doutorado Shimabukuro levanta técnicas de visualização multidimensional propostas por Keim e Kriegel (KEIM; KRIEGL, 1996 apud SHIMABUKURO, 2004), são chamados multidimensionais dados que tem mais de 3 atributos como 3 coordenadas x , y , z espaciais e um atributo t de tempo. Tais técnicas são descritas nas seções que seguem, além destas também serão apresentadas as técnicas de ‘Variação Temporal Uni-escala’ e ‘Multi-escala’ propostas por Shimabukuro (SHIMABUKURO, 2004).

3.3.1 Técnicas Orientadas a Pixel

Esta Técnica utiliza uma janela com pixels de coloração variável podendo dividir a tela em quadrantes representando cada atributo dos dados, como na Figura 14

É possível utilizar algumas alternativas referentes aos aspectos desta técnica que são:

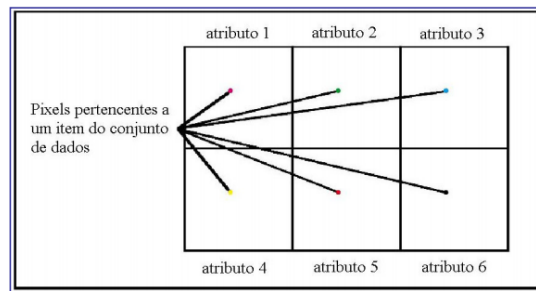


Figura 11: Múltiplas janelas para um caso de 6 atributos em técnicas orientadas a pixel. (SHIMABUKURO, 2004) adaptado de (KEIM, 2000).

- Arranjo dos pixels nas janelas: A disposição dos pixels pode ficar de difícil leitura devido ao distanciamento dos atributos, da esquerda para a direita, de cima para baixo. Assim, são feitas consultas com um valor de requisito para os atributos, a janela é dividida em 4 quadrantes de forma espiral como na Figura 12, os dados com mais proximidade do valor vão estar mais próximos do centro dos eixos. Esta alternativa tem a precariedade de representar no máximo 4 atributos.

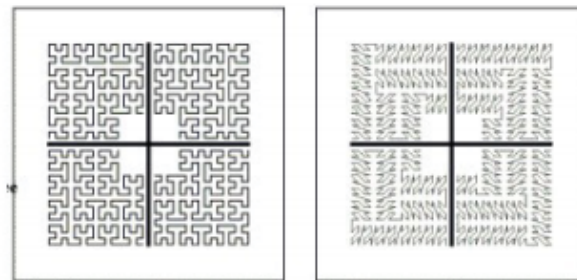


Figura 12: Possibilidades de disposição dos pixels nas técnicas orientadas a pixel. (KEIM; KRIEGEL, 1996 apud SHIMABUKURO, 2004).

- Formato das Janelas: Ainda em relação ao problema da janela temos o aspecto da retangularidade que dificulta a visualização pelo distanciamento dos atributos, podendo utilizar um formato circular como na Figura 13 ou gráfico em barras na Figura ?? para aproximar esta distância.

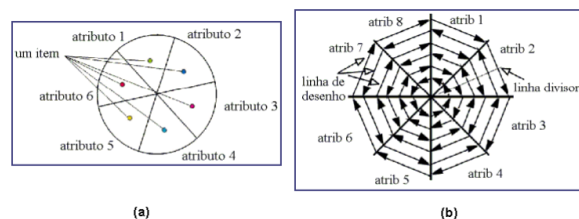


Figura 13: Formato circular para apresentação dos pixels em Segmentos de Círculo. (SHIMABUKURO, 2004) adaptado de (KEIM, 2000).

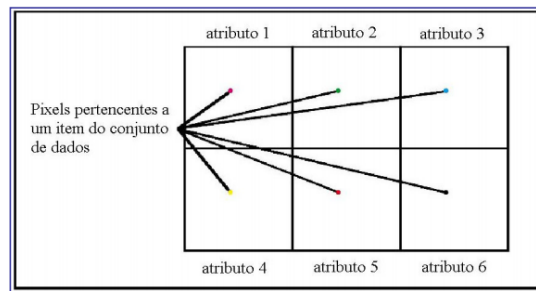


Figura 14: Esquema de arranjo dos pixels em Gráfico de Barras de Pixels. (SHIMABUKURO, 2004) adaptado de (KEIM et al., 2001).

- Mapeamento de cor: Um dos problemas é a escolha de uma escala de cores que permita a maior extensão de representação e também seja natural para o usuário e o domínio da aplicação. Uma saída interessante para a variação é o uso de diferentes escalas de brilho para distribuição de um único parâmetro.

3.3.2 Técnicas de Projeção Geométrica

Utiliza um gráfico paralelo aos eixos x e y onde cada atributo é um eixo paralelo e uma linha traçada perpendicularmente corta os eixos denotando o valor como na Figura 15. Este tipo visa demonstrar bi ou tridimensionalmente visualizações de n dimensões

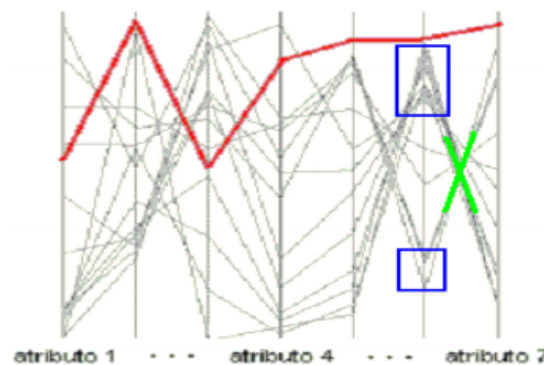


Figura 15: Correlação e distribuição de valores em Coordenadas Paralelas. (WARD, 1994 apud SHIMABUKURO, 2004).

O problema desta técnica é que a imagem pode ficar saturada com o número de retas inseridas como na Figura 16

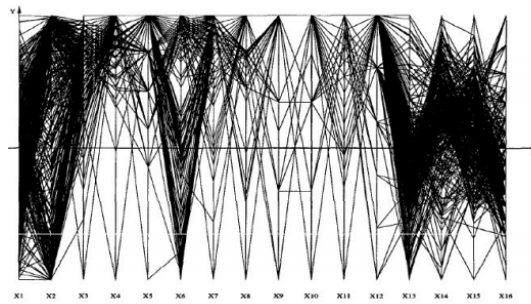


Figura 16: Sobreposições de linhas com apresentação de 473 itens. (INSELBERG, 1997 apud SHIMABUKURO, 2004).

3.3.3 Técnicas Iconográficas

Nesta técnica cada item dos dados é transposto a um ícone, como na técnica de ‘Figura de Arestas’(Stick Figure) em que cada dimensão da tela é um atributo dos dados, no caso 2, e os outros atributos podem ser o ângulo de cada aresta e(ou) o comprimento da mesma, como na Figura 17. Outras características podem ser disponibilizadas para os atributos como a cor e a espessura das arestas, O conjunto de vários itens de dados também pode gerar um padrão de textura de rápida percepção visual como na Figura 18. A desvantagem desta técnica é poder apresentar um número máximo de dimensões perto de 10.

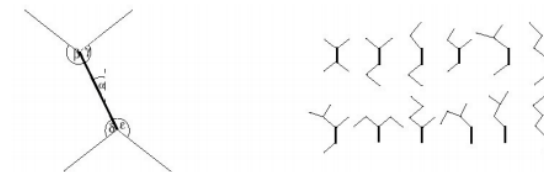


Figura 17: Configuração para Figura de Arestas usando cinco arestas (KEIM; KRIEDEL, 1996 apud SHIMABUKURO, 2004).

3.3.4 Técnicas Hierárquicas

Nas técnicas de visualização hierárquicas, a janela é dividida em um conjunto de espaços e cada espaço pode ter uma subdivisão dentro de si em mais espaços representando atributos para cada divisão como na Figura 19. essa técnica também é conhecida como ‘Empilhamento Dimensional’(Dimensional Stacking). A dificuldade é que não é possível levantar muitos atributos por causa das subdivisões no espaço da tela, a não ser que exista alguma funcionalidade interativa para ampliar e reduzir o tamanho da imagem.

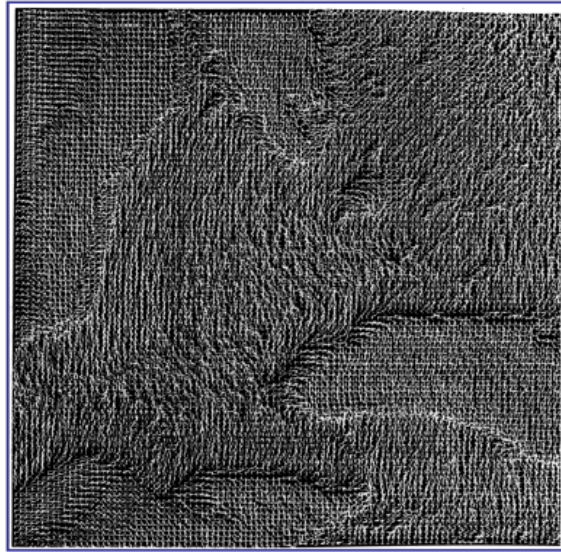


Figura 18: Diferentes texturas representadas por Figura de Arestas. (GRINSTEIN; TRUTSCHL; CVEK, 2001 apud SHIMABUKURO, 2004).

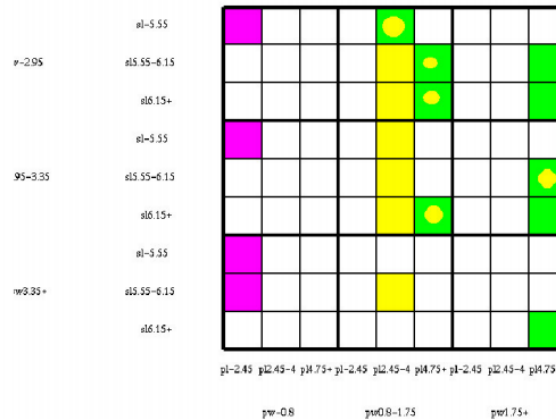


Figura 19: Resultado do empilhamento dimensional. Quatro atributos são aninhados aos pares (HOFFMAN, 1977 apud SHIMABUKURO, 2004).

3.3.5 Variação Temporal Uni-escala

Shimabukuro (SHIMABUKURO, 2004) propõe um novo modelo que se encaixa muito bem para atributos que apresentam uma variação temporal em alguma escala definida (dia, semana, ano, ...) utilizando uma tabela de cores, mapeando os intervalos de valores em interesse nas mesmas. Tais valores podem ser representados por sequências de pixels ou pontos organizados como quadrados como na Figura 20. O autor descreve que cada sequência de pixels é correspondente a um objeto espacial, colocados verticalmente ou horizontalmente. Esta técnica permite uma visão simultânea de vários objetos referentes ao seu valor durante o intervalo de tempo.

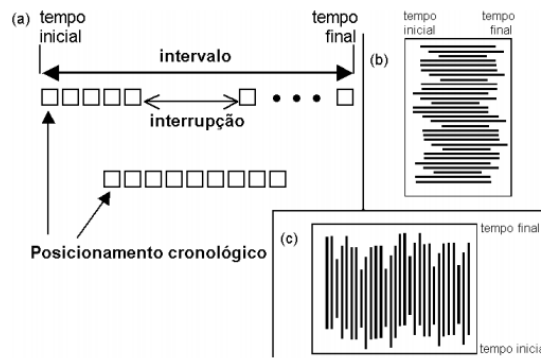


Figura 20: Construção da representação visual ‘Variação Temporal Uni-escala’.(SHIMABUKURO, 2004).

3.3.6 Variação Temporal Multi-escala

A técnica multi-escala é similar a uni-escala porém, ao invés de ter o foco em observar vários objetos ao longo do tempo, ela tem o foco de observar apenas um objeto com um detalhamento temporal mais específico. Neste caso são agrupados os valores do objeto em diferentes escalas de tempo na mesma janela, tomando um caso qualquer seria possível observar os valores diários, mensais, anuais e no decorrer dos anos de tal objeto como na Figura 21.

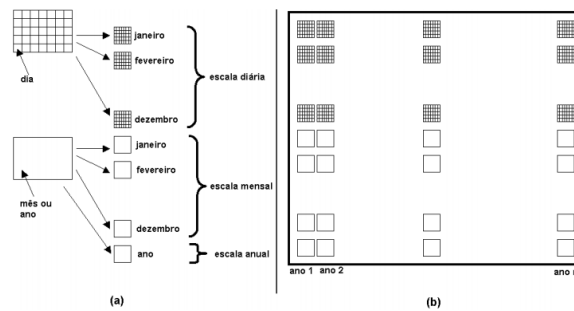


Figura 21: Construção da representação visual ‘Variação Temporal Multi-escala’.(SHIMABUKURO, 2004).

Referências

- ARCHIVE, N. *NoSQL Databases*. 2009. Disponível em: <http://nosql-database.org/>.
- CHEN, M.-S.; HAN, J.; YU, P. S. Data mining: an overview from a database perspective. *IEEE Transactions on Knowledge and data Engineering*, IEEE, v. 8, n. 6, p. 866–883, 1996.
- FAYYAD, U. M. Mining databases: Towards algorithms for knowledge discovery. *IEEE Data Eng. Bull.*, v. 21, n. 1, p. 39–48, 1998.
- FAYYAD, U. M. et al. Knowledge discovery and data mining: towards a unifying framework. In: *KDD*. [S.l.: s.n.], 1996. v. 96, p. 82–88.
- GENGHINI, F. Extensão da plataforma de visualização infovis incorporando recursos de interatividade. Universidade Estadual Paulista (UNESP), 2011.
- GRINSTEIN, G.; TRUTSCHL, M.; CVEK, U. High-dimensional visualizations. In: CITESEER. *Proceedings of the Visual Data Mining Workshop, KDD*. [S.l.], 2001.
- HOFFMAN, P. E. *Table visualizations: a formal model and its applications*. Tese (Doutorado) — University of Massachusetts Lowell, 1977.
- INSELBERG, A. Multidimensional detective. In: IEEE. *Information Visualization, 1997. Proceedings., IEEE Symposium on*. [S.l.], 1997. p. 100–107.
- INTERNATIONAL, E. *Introducing JSON*. 2013. Disponível em: <http://www.json.org/>.
- KEIM, D. A. Designing pixel-oriented visualization techniques: Theory and applications. *IEEE Transactions on visualization and computer graphics*, IEEE, v. 6, n. 1, p. 59–78, 2000.
- KEIM, D. A. et al. Pixel bar charts: A new technique for visualizing large multi-attribute data sets without aggregation. In: *IEEE Symposium on Information Visualization, 2001. INFOVIS 2001*. [S.l.: s.n.], 2001. p. 113–120.
- KEIM, D. A.; KRIEGEL, H.-P. Issues in visualizing large databases. In: *Visual Database Systems 3*. [S.l.]: Springer, 1995. p. 203–214.

KEIM, D. A.; KRIEGEL, H.-P. Visualization techniques for mining large databases: A comparison. *IEEE Transactions on knowledge and data engineering*, IEEE, v. 8, n. 6, p. 923–938, 1996.

NÄSHOLM, P. Extracting data from nosql databases-a step towards interactive visual analysis of nosql data. Chalmers University of Technology, 2012.

POSTGRESQL. *JSON Functions and Operators*. 1996. Disponível em: <https://www.postgresql.org/docs/9.6/static/functions-json.html>.

RHYNE, T.-M. Does the difference between information and scientific visualization really matter? *IEEE Computer Graphics and Applications*, IEEE Computer Society Press, v. 23, n. 3, p. 6–8, 2003.

SHIMABUKURO, M. H. *Visualizações temporais em uma plataforma de software extensível e adaptável*. Tese (Doutorado) — Instituto de Ciências Matemáticas e de Computação, 2004.

STRAUCH, C.; SITES, U.-L. S.; KRIHA, W. Nosql databases. *Lecture Notes, Stuttgart Media University*, 2011.

WARD, M. O. Xmdvtool: Integrating multiple methods for visualizing multivariate data. In: IEEE COMPUTER SOCIETY PRESS. *Proceedings of the Conference on Visualization'94*. [S.l.], 1994. p. 326–333.

WEBER, S. Nosql databases. *University of Applied Sciences HTW Chur, Switzerland*, 2010.

WHITTAKER, G. L. S. T. J. Improving performance of schemaless document storage in postgresql using bson. 2013.

WONG, P. C. Visual data mining. *IEEE Computer Graphics and Applications*, IEEE, v. 19, n. 5, p. 20–21, 1999.