Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales

Tesis de Licenciatura en
Ciencias de la Computación

SQLVis: Herramienta para la visualización de logs de sentencias SQL lentas con patrones similares

Julian R Berlin
LU: 678/00
julian.r.berlin@gmail.com

Facundo Pippia
LU: 520/00
Facundomensajes@gmail.com

Directores:
Lic. Fernando Asteasuain
Lic. Diego Ariel Aizemberg

Buenos Aires, 22 Enero 2013

Resumen

Los motores de base de datos relacionales de hoy en día cuentan con mecanismos para almacenar las consultas que por alguna razón demoran más tiempo de lo usual. Es muy común que el volumen de información almacenado en tales archivos sea de tamaño considerable, por lo que el proceso de detectar cuál o cuáles son las consultas que generan más carga en el servidor es una tarea muy difícil y las herramientas existentes no proveen una interfaz gráfica atractiva para atacar este problema eficientemente.

En este trabajo se propone SQLVis, una herramienta de visualización de logs de consultas lentas agrupadas por similaridad, basada en *Squarified Treemaps*. Se estudiaron las tecnologías actuales para el desarrollo de sistemas que requieran visualizar datos complejos o fuera de lo común para luego seleccionar la que mejor se adaptara para desarrollar la herramienta propuesta. El objetivo es que con esta manera de visualizar la información contenida en el log, una persona pueda rápidamente comprender y evaluar cuál o cuáles son los grupos de consultas que más están afectando el rendimiento de la base de datos para un esquema de datos dado. Con SQLVis se obtuvieron buenos resultados en términos de tiempo en la identificación de los grupos de consultas con tiempos de ejecución por encima del umbral definido para logs de tamaño considerable.

Abstract

Nowadays, relational database engines have mechanisms to log those queries that for some reason take longer time than usual to execute. It is very common that the volume of such information is of considerable size, so the process of detecting which of these queries generate more load in the server is a very difficult task since existing tools do not provide an attractive graphic interface to solve the problem efficiently.

In this work we propose SQLVis, a visualization software based on *Squarified Treemaps* to visualize slow query logs grouped by similarity. We first studied the current available technologies for the development of visualization systems requiring visualize complex or unusual data and then we selected the best suited for the development of the proposed tool, the goal is that visualizing the information stored in the slow query log by using this tool, a person can understand which are the worst query patterns in terms of execution time that are impacting the performance of the database system for a given schema. With SQLVis we obtained good results in terms of time to identify groups of queries with similar patterns that takes longer execution times in logs of considerable size.

Keywords: Infovis, Treemaps, Information Visualization, SQL Slow Query Logs, Query optimization.

Indice general

1 INTRODUCCIÓN

1.1 MOTIVACIÓN

El origen del proyecto fue nuestro interés en el área de visualización de la información. Decidimos contactarnos con Ariel Aizemberg especialista en la representación de información, profesor de la materia: "Visualización de Información". Él estaba trabajando en unas optimizaciones sobre consultas SQL en un entorno MySQL y nos propuso la idea de hacer una herramienta que permita ver de manera rápida cuál o cuáles son las consultas que más problemas de performance traen, ya que es muy común que logs de este tipo tengan miles de registros y en general con muchas consultas similares entre si es difícil ver dónde está el problema, como veremos no existen demasiadas herramientas de visualización orientadas a este dominio, existen lo que se denominan herramientas de optimización que en general son herramientas de consola.

Por lo tanto el objetivo planteado fue construir un visualizador orientado a este dominio seleccionando una metáfora visual apropiada que explote al máximo la información contenida en el log y que sea sencilla de usar y entender.

1.2 ESTRUCTURA DEL DOCUMENTO

Para facilitar la lectura y compresión el documento se organizó de la siguiente manera:

- En el **Capítulo 2** se introducen los siguientes tópicos: bases teóricas y fundamentos de la visualización de la información, los aportes de otras disciplinas como la Comunicación Gráfica, la Interacción y la Psicología Cognitiva; y los conceptos teóricos propios discutiendo los Modelos de Referencia y las Taxonomías de sistemas de Visualización de Información propuestas. Así como también se mencionan buenas prácticas de diseño de manera de obtener visualizaciones de alta calidad.

- En el **Capítulo 3** se describen las tecnologías actuales para desarrollar proyectos de visualización, en esta sección hacemos foco en las ventajas y desventajas de cada una mencionando las principales características, las metáforas visuales que se pueden implementar en dichas tecnologías y luego decidimos que tecnología utilizar en base a nuestro criterio.

- En el **Capítulo 4** abordamos el desarrollo de la propuesta, partiendo de las motivaciones y orígenes del mismo y luego tomando como punto de partida el pipeline de Ben Fry [2], comentamos el desarrollo del prototipo y describimos cada una de las etapas del pipeline que utilizamos para desarrollar la herramienta desde la obtención de los datos en crudo hasta el refinamiento de la visualización.

- En el **Capítulo 5** realizamos las pruebas sobre el prototipo, focalizando en responder las preguntas planteadas para el dominio de datos, como cuál es el grupo de consultas que más tiempo demora en ejecutar, como son esas consultas así como también mostramos las capacidades visuales del prototipo desarrollado.

- Finalmente en el **Capítulo 6** ponemos las conclusiones del trabajo realizado, así como también que mejoras futuras se podrían realizar de manera de mejorar el prototipo original, dotarlo de más funcionalidad y más capacidades para hacerlo más atractivo al usuario.

1.3 TERMINOS Y ABREVIACIONES

En la siguiente tabla se definen los términos y abreviaciones utilizados en el documento.

Término o Abreviación	Descripción
D3	*Data Driven Documents*
FLEX	Tecnología de Macromedia para desarrollar aplicaciones de internet de alto contenido multimedia
CANVAS	*Canvas* es un elemento que permite la creación de objetos gráficos componiendo otros objetos gráficos. Su traducción del inglés es *tapiz*, también es un elemento de html5.
DATA MINING	En Español minería de datos, es la disciplina de las ciencias de la computación que estudia el proceso de descubrimiento de patrones en grandes volúmenes de datos.
SVG	Gráficos Vectoriales Escalables
RENDERIZADO	Es la acción de graficar un objeto en base a un conjunto de propiedades definidas
WPF	*Windows Presentation Foundation*
DOM	*Document Object Model.*
CSS3	*Cascading Style Sheet*
JIS	*Java Infovis Toolkit*
J2EE	*Java 2 Platform Enterprise Edition*
Framebuffer	El *framebuffer* es una memoria de acceso rápido utilizado para la graficación por las placas gráficas.

2 BASE TEÓRICA – VISUALIZACIÓN DE LA INFORMACIÓN

2.1 INTRODUCCIÓN

En la actualidad, los volúmenes de información vienen creciendo de manera exponencial en gran parte por la explosión y la difusión de los sistemas de información en todos los ambientes de la vida cotidiana, dado semejante volumen, se hace extremadamente difícil para un ser humano poder concluir tendencias, patrones y todo otro conocimiento que se desprendan de dicha información en bruto sin la asistencia de herramientas específicas. Es aquí en donde las herramientas de visualización de la información entran en juego para proveer mecanismos que permitan abstraer la información de manera tal que sea más clara y entendible para un observador dado y por otro lado brindar mecanismos que permitan al observador manipular la información representada.

La Real Academia Española de la Lengua (RAE), define el concepto de visualización como "***representar mediante imágenes ópticas fenómenos de otro carácter"***, es decir formar en la mente una representación visual de un concepto abstracto o imaginar con rasgos visibles algo que no se puede ver. Utilizando la misma fuente, la imaginación se define *como* ***la facultad del alma que representa las imágenes de las cosas reales o ideales***. ***Imagen formada por la fantasía***. Por tanto, al definir visualización de la información estaremos ante un proceso de pensamiento que involucra el concepto de aprendizaje y los sentidos del ser humano.

Un claro ejemplo de aplicación: El gráfico de la fiebre de un enfermo respecto al tiempo permite obtener información rápidamente de la tendencia y evolución de la enfermedad. La fiebre (temperatura) y el tiempo (las horas) son las variables. La enfermedad en si es el concepto subyacente. El objetivo es hacer que un concepto difícil de comprender sea más claro e inmediato para un observador dado, utilizando las herramientas apropiadas para tal propósito.

Variable *x* (hora)	10	12	14	16	18	20
Variable *y* (temperatura)	37 °C	39 °C	38 °C	38 °C	36 °C	38 °C

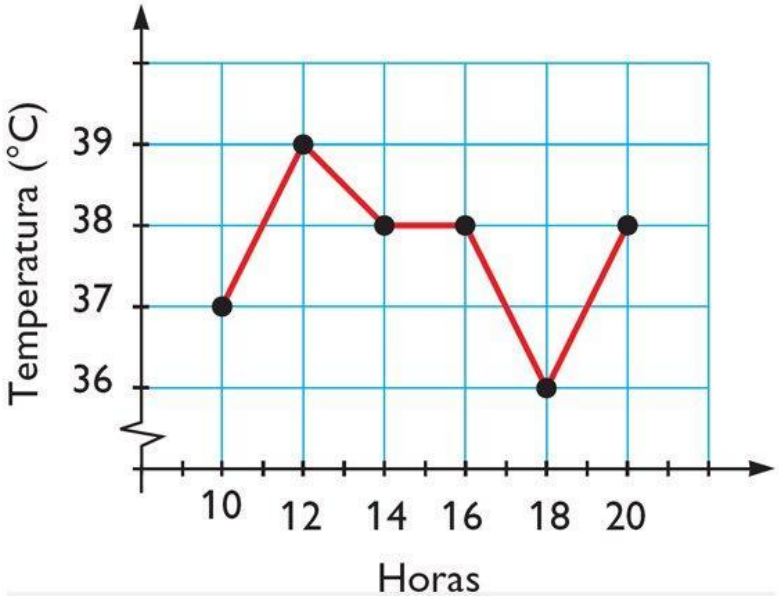


Figura 1- Grafico que muestra la temperatura en función del tiempo.

El objetivo más importante de la visualización de la información es la representación adecuada tanto de la información con parámetros múltiples como de las tendencias y las relaciones subyacentes que existen en dicha información. Su propósito no es la concepción de las imágenes en sí mismas sino el *insight*, es decir, la absorción rápida de conocimiento. En la actualidad, Tecnologías de visualización altamente complejas son accesibles al público en general gracias a la evolución de la computación y los procesadores. Esta evolución ha posibilitado su difusión a lo ancho y a lo largo en todos los ámbitos científicos y particulares.

2.2 DEFINICIONES

En esta sección se mencionan las definiciones del área más importantes a tener en cuenta para poder comprender y entender claramente las siguientes secciones. Según el glosario de términos definido en [4] tenemos lo siguiente:

Visualización

"Es la formación en la mente de una imagen de un concepto. En este contexto, se entiende visualización como la representación gráfica de variables asociadas al concepto que se quiere visualizar."

Visualización de la Información

"Es el proceso de interiorización del conocimiento mediante la percepción de la información. La visualización de la Información se beneficia básicamente de que:

- *Los seres humanos reciben información de forma eminentemente visual ya que es el sentido con mayor ancho de banda, es decir, que proporciona mayor cantidad de información;*

- *La capacidad simbólica del cerebro humano.*

De esta manera la disciplina trata la problemática de la generación de herramientas y derivados que permitan visualizar información en diferentes dominios de aplicación."

Visualización científica

"La diferencia con la definición anterior es que la visualización científica trata y se centra en problemas de visualización de datos generados por las disciplinas científicas que le dieron origen.

El denominador más común en este tipo de visualizaciones es la dependencias de atributos espaciales en los datos y el interés está en generar y focalizarse en representaciones realistas que respeten reglas físicas de elementos naturales como fluidos y otros tipos de fenómenos que tienen leyes formales bien definidas en el dominio en cuestión (por ejemplo fenómenos meteorológicos, etc.)."

Metáfora visual

Una Metáfora visual consiste en la representación de un sistema mediante atributos visuales propios de otro diferente que ya es familiar al usuario y que se comporta de una manera análoga.
Un caso arquetípico es la metáfora del escritorio. En ella se sustituye el esquema tradicional del árbol jerárquico de directorios y subdirectorios por el interfaz gráfico de los archivadores y carpetas.

El árbol genealógico es otro ejemplo típico de metáfora visual

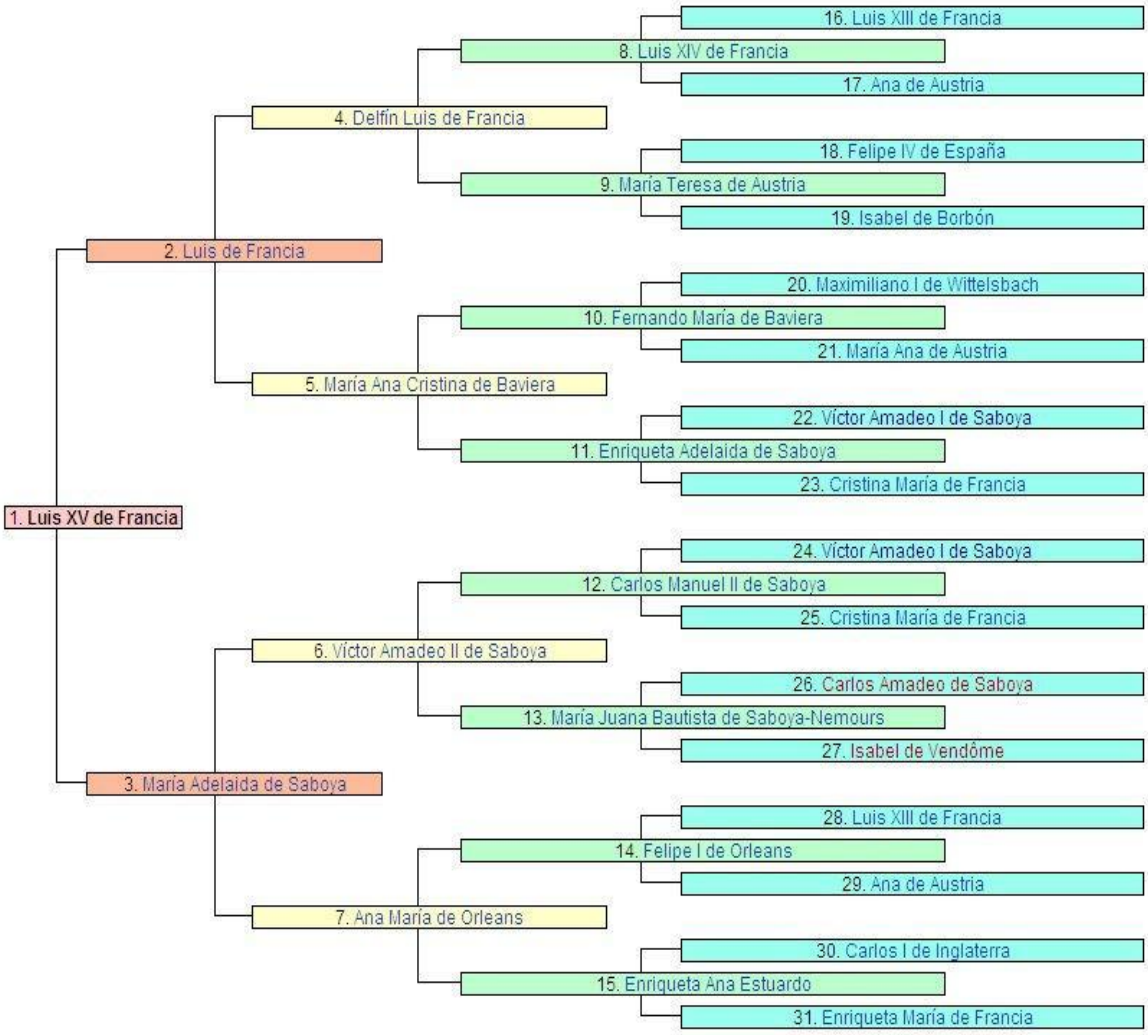


Figura 2 - Árbol genealógico de Luis XV de Francia.

Otra definición que es interesante citar es la definida en [14]:

"Una metáfora es una figura del lenguaje donde se realiza una comparación entre dos objetos que no tienen ninguna relación aparente.
Un ejemplo conocido es la frase "El tiempo es oro" donde se transfieren las características de la palabra "oro" (valioso, escaso, preciado) a la palabra "tiempo" para dar a entender su valor.

Originalmente metáfora es una palabra griega que significaba transferir . Su etimología viene de meta que significa "cambio" y pherein que significa "llevar o trasladar". Así, la palabra Metáfora tiene a su vez un significado metafórico: "llevar o transferir un significado de una cosa a otra"

Las metáforas visuales funcionan de una manera similar, se sustituye un mensaje complejo mediante una imagen más simple pero evocativa que le transfiere a éste su significado. Por ejemplo cuando una empresa que produce jugos de frutas enlatados le pone a su producto una marca con un pictograma representando a un árbol está pretendiendo sustituir en la mente del receptor las características del producto (enlatado, industrializado, producido en serie etc.) por las del árbol (naturaleza, frescura, saludable)."

2.3 RESEÑA HISTÓRICA

La visualización, en cualquiera de sus numerosas formas, no es un concepto nuevo [21]. Desde hace mucho tiempo el hombre creó representaciones gráficas de los datos que disponía para diferentes propósitos, Un claro ejemplo son los mapas, quizás los mapas sean la expresión más antigua de la visualización, proviniendo algunos de la época de antes de Cristo (AC), como el mapa de "Mesopotamia Norte" hecho en una tabilla de barro en el 3800 AC.

Figura 3 - Mapa de "Mesopotamia Norte" que data del 3800 antes de Cristo.

El propósito de los mapas era para la navegación y la exploración tanto por mar y tierra [21]. También fueron usados imágenes para documentar fenómenos como la posición del sol durante las diferentes épocas del año, las mareas, las lunas llenas, etc.

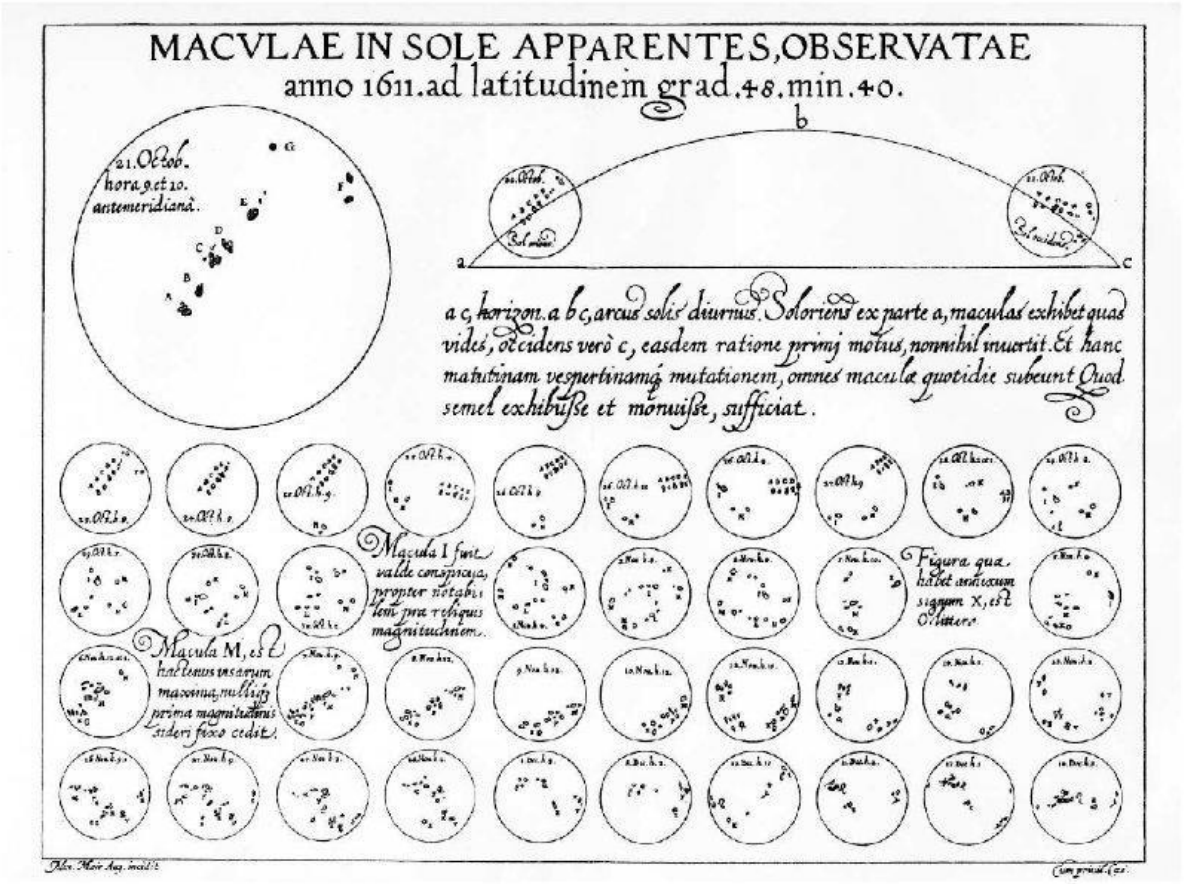


Figura 4 - Grafica de Scheiner realizada en 1626 que describe las posiciones del sol durante a lo largo del tiempo (del 23 Octubre hasta el 19 de Diciembre del año 1611).

Ya para el siglo 16, las técnicas e instrumentos para la medición precisa de las magnitudes físicas ya estaban bien desarrolladas. En el siglo 17 se vio un gran crecimiento nuevo en la teoría y la práctica de lo que hoy llamamos visualización de la información, el surgimiento de la geometría analítica y la estadística así como también los métodos para medición y control de errores de dichas mediciones, usadas principalmente con fines demográficos.

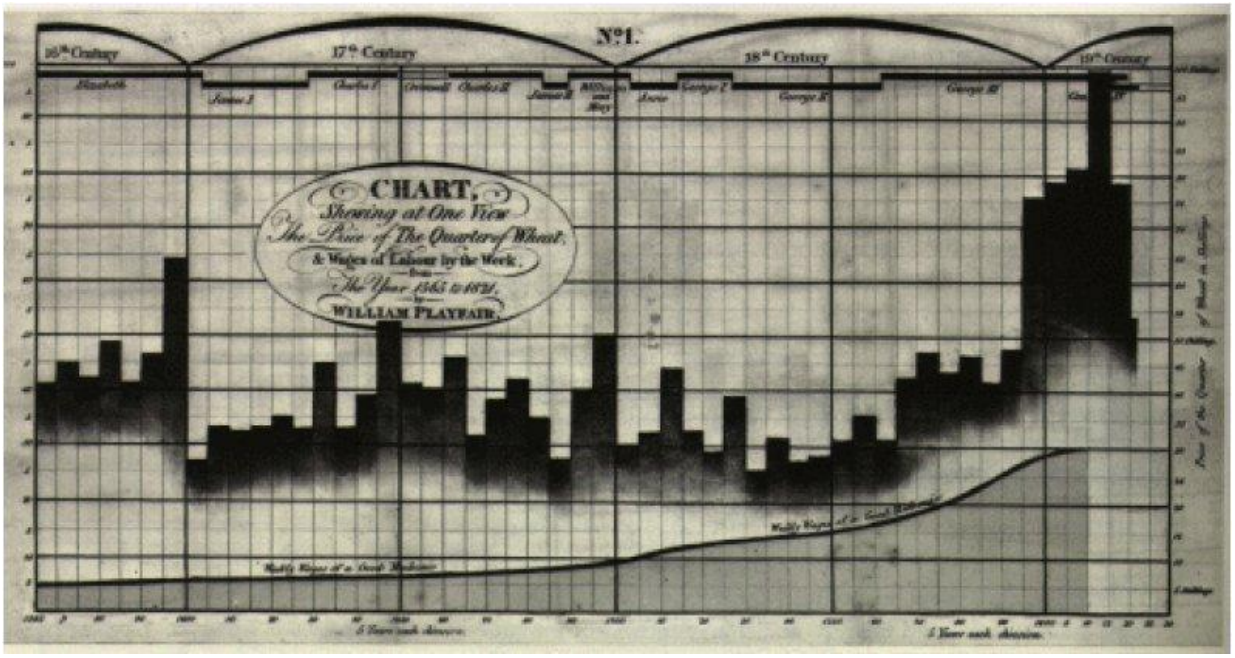


Figura 5 - William Playfair's 1821 grafico de tiempo que muestra los precios, salarios y el monarca ingles de turno en cada periodo. Imagen tomada del libro de Tufte (1983, p. 34).

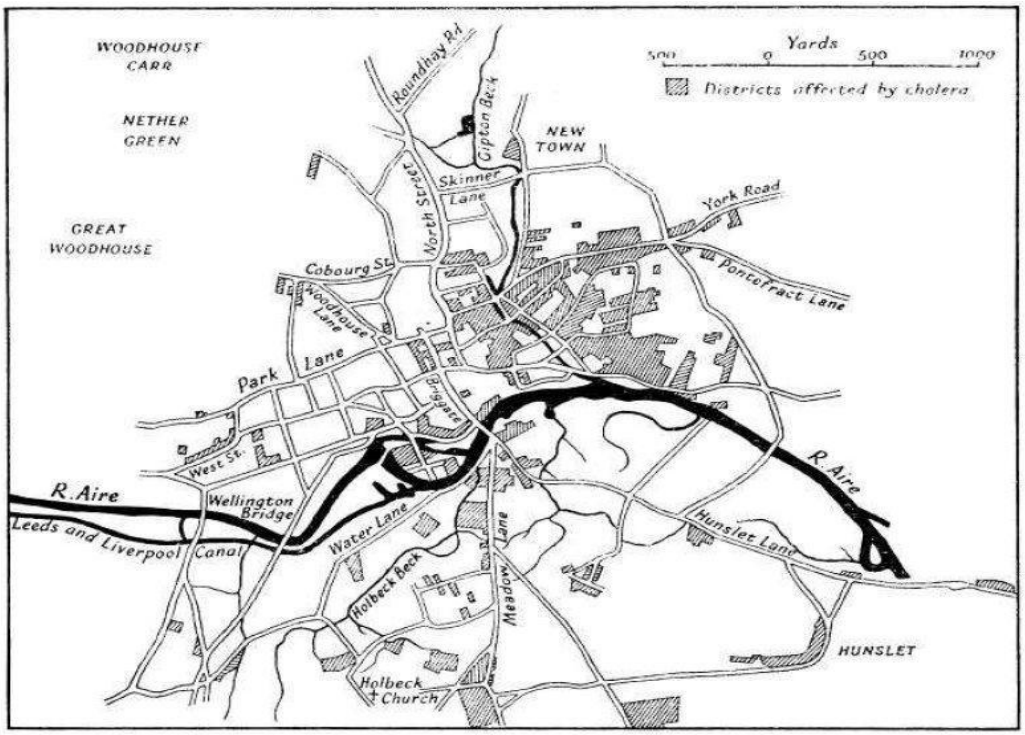


Figura 6 - Segmento del mapa del cólera de la ciudad de Leeds, que data de 1833 y muestra los diferentes distritos afectados por el cólera, realizado por Robert Baker's.

A través de los siglos 18 y 19, Se comienzan a recolectar de manera masiva los números de las estadísticas sociales, médicas y económicas, en grandes series y de manera periódica y, además, dichos conjuntos de datos fueron utilizados para la planificación y respuesta gubernamental ante ciertos eventos como plagas, crecimiento de la población, planificación urbana y distribución del ingreso entre otras cosas. La disciplina tomó forma y comenzó a desarrollarse como propio objeto de estudio y también comenzó a ser reconocida.

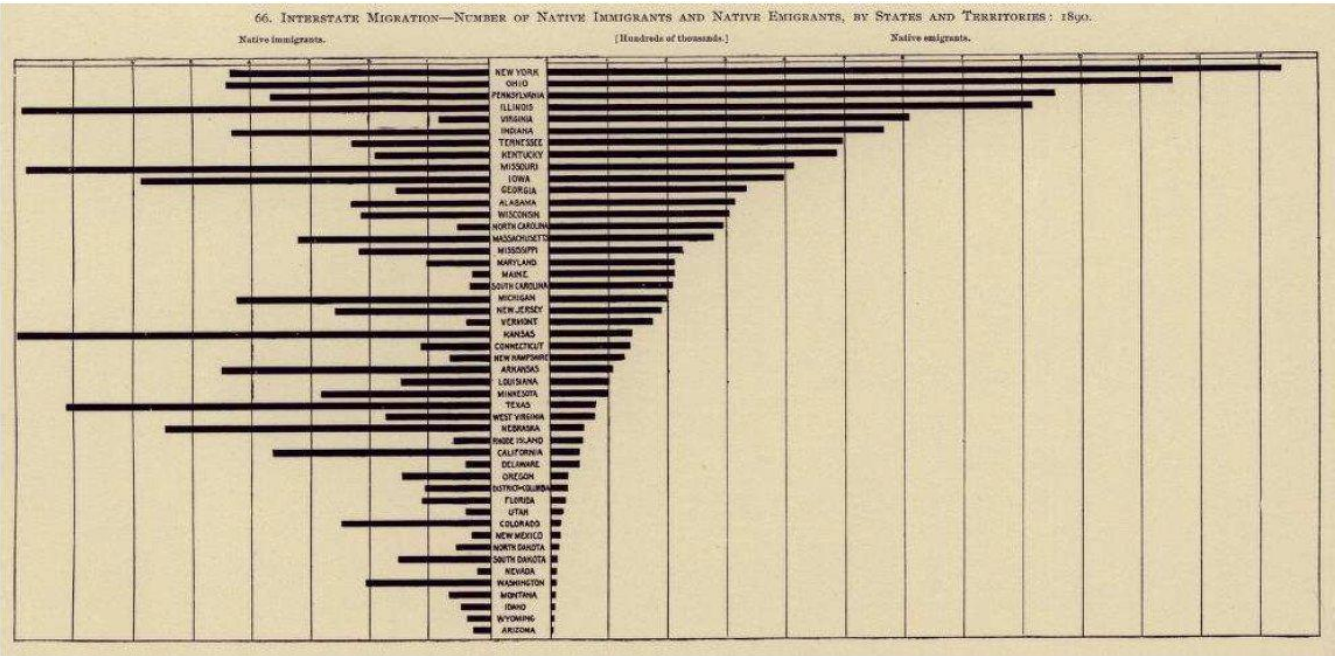


Figura 7 - Gráfica que muestra la migración entre estados en USA - Fuente: Statistical Atlas of the Eleventh Census, 1890.

Los gráficos y modelos en dos dimensiones fueron los primeros pasos en materia de visualización de la información. Dichas graficas fueron progresivamente evolucionando a modelos de 3, 4 y más dimensiones. Inicialmente los modelos de 3 dimensiones se concibieron como modelos de líneas o alambre y luego más tarde se les brindo volumen mediante el *rendering*. Que básicamente es la construcción de una superficie solida mediante la interpolación de los puntos existentes.

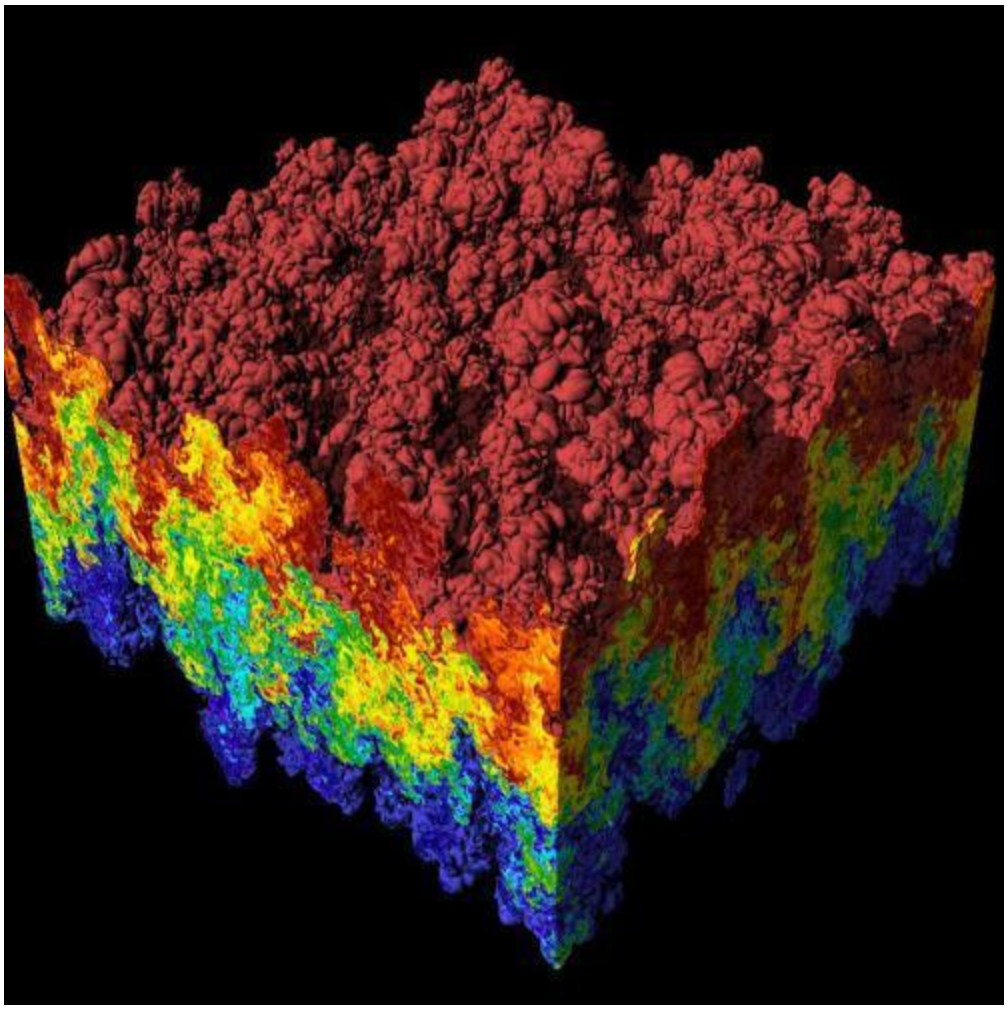


Figura 8 - Visualización científica en 3D de una simulación del
Problema de inestabilidad Rayleigh-Taylor.

Luego al pasar del tiempo fueron surgiendo métodos para interactuar, modificar y brindar animaciones a estos modelos, Algunos de estos métodos gráficos brindan la posibilidad de representar 2 o más variables en un plano dado mediante la manipulación del plano, un ejemplo claro de este tipo de técnicas son las cartas de Smith (Figura 9) utilizadas para el diseño y análisis de las impedancias de antenas. Conforme va aumentando el número de variables que se deben tomar en cuenta, Los métodos de relación de datos van aumentando en complejidad. Se vuelve muy común usar computadoras de gran poder para procesar grandes volúmenes de información y luego presentar los resultados gráficamente en otra estación de trabajo de manera de distribuir el procesamiento en ambos computadores. Se vuelve cotidiano grabar las imágenes generadas en medios de almacenamiento de manera de facilitar su distribución.

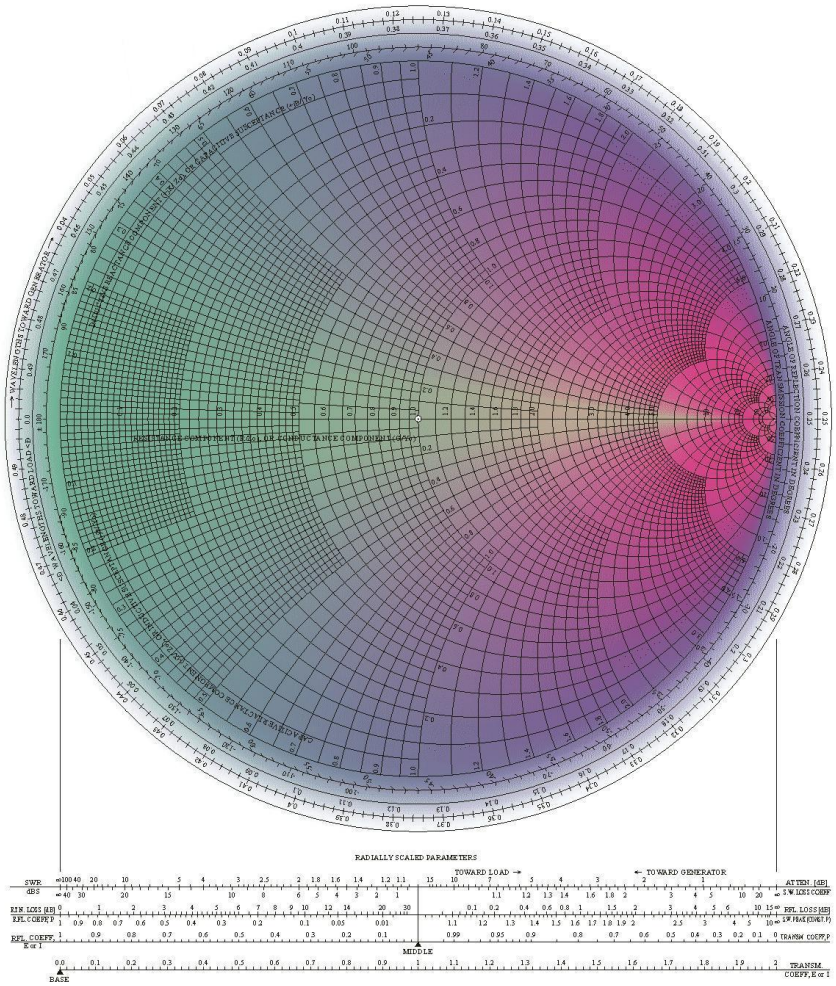


Figura 9 - Carta de Smith [19], que muestra cómo varía la impedancia compleja de una línea de transmisión a lo largo de su longitud, para analizar redes de transmisión.

Uno de los avances más recientes en visualización de la información es la utilización de la realidad virtual, Mediante esta técnica es posible representar objetos o lugares en 3 dimensiones que no se podrían representar usando las tecnologías convencionales. Usos comunes de la realidad virtual son la exploración y estudio del funcionamiento de los órganos de un ser humano. También es posible representar objetos de 4 dimensiones en 3 dimensiones sin tener que reducirlos a 2 dimensiones al visualizarlos en una pantalla normal.

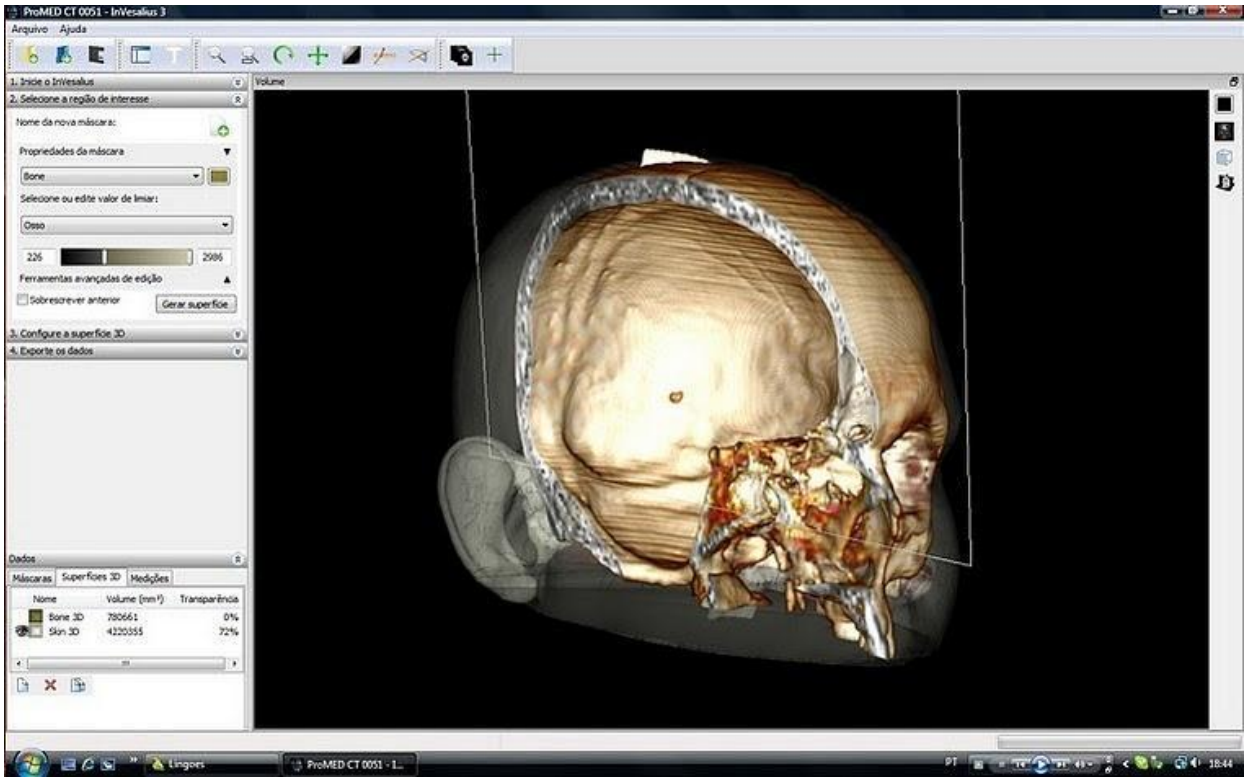


Figura 10 - In Vesalius software de visualización que permite recrear los órganos del cuerpo humano.

El análisis estadístico, la simulación de fenómenos físicos y naturales y la educación son las aplicaciones más comunes de la visualización en lo que respecta al análisis exploratorio y confirmatorio. En el Área de análisis existen herramientas para el control de calidad, análisis de esfuerzos y estimaciones financieras entre otras. Las aplicaciones de análisis de esfuerzo junto a las de modelos naturales y atmosféricos podrían considerarse como parte de la categoría de herramientas de simulaciones. Para lo que respecta educación existen aplicaciones para realizar y contrastar demostraciones matemáticas y modelos de física cuántica entre otras cosas.

La visualización parecería ser que sólo existe en ámbitos exclusivamente científicos, pero al día de hoy cada vez más gente utiliza computadores para generar imágenes o gráficos con los cuales se expresan y representan información que ellos utilizan en el día a día en ámbitos como el trabajo y la vida cotidiana. Es por eso que progresivamente las tecnologías de visualización de la información se van generalizando, en resumidas cuentas podemos decir que la sociedad está en una etapa de transición en la manera en que las personas se comunican y transmiten información. Pasando de la afluencia de los textos a medios de comunicación gráficos.

2.4 CONTEXTO Y APLICACIONES

Hoy en día la cantidad y variedad de aplicaciones para visualizar información es inmensa y es un área de estudio que está en constante evolución dada la explosión de los volúmenes de información de los sistemas actuales, hay sistemas de visualización que deben manejar millones de gigabytes en una diminuta pantalla y dichos sistemas deben brindar respuesta en tiempos razonables es decir en tiempos interactivos. Esto genera que haya líneas de investigación activas que se focalizan en el estudio de la performance y en el desarrollo de nuevas técnicas de visualización que permitan obtener visualizaciones más efectivas y performantes desde el punto de vista del usuario.

Los métodos utilizados tienen asociados, en su gran mayoría, un gran costo computacional y es por ello que para lograr una representación de la información con tiempos de respuesta interactivos es importante contar con métodos apropiados para los distintos tipos de datos de manera de explotar al máximo el poder de procesamiento de una computadora y que los tiempos de respuesta al usuario sean realmente buenos.

En la actualidad hay una inmensa cantidad de aplicaciones visualización de la información para visualizar información de todo tipo orientados a dominio y de carácter general. Un buen ejemplo es *Narrative 2.0* que permite visualizar música de una manera original, la canción se segmenta en canales y los canales se muestran en forma de abanico y las líneas se mueven desde el centro de distancia con el tiempo. El ángulo de la línea cambia de acuerdo con la frecuencia del canal, mientras que la frecuencia alcanza un nivel alto, el canal queda resaltado en naranja.

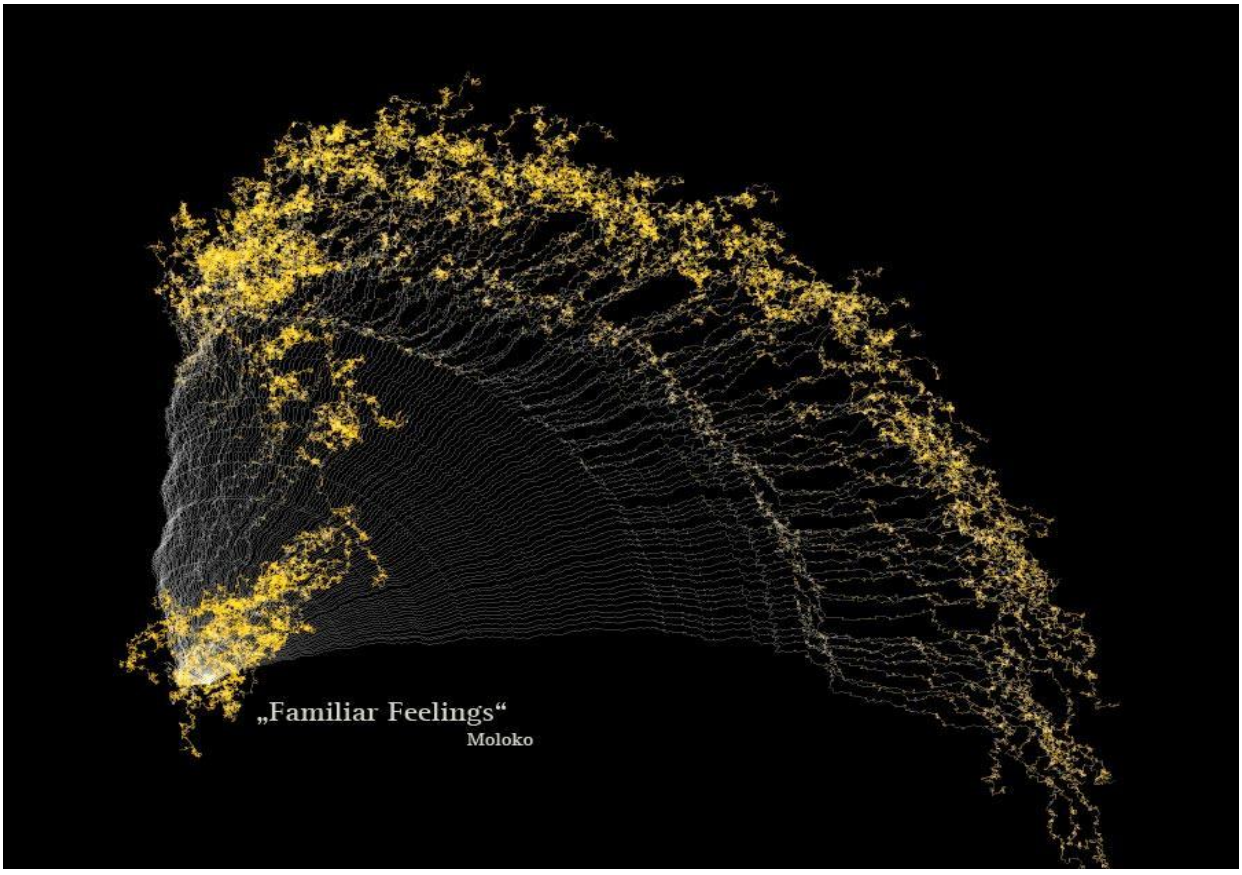


Figura 11 - Visualización de la música de la canción Familiar feelings de Moloko.

Todos los años se realiza la *VisWeek* (http://visweek.org) en el cual se presentan los últimos avances en visualización científica e visualización de la información en todos los campos de aplicación, avances en cuanto a técnicas, herramientas y tecnologías, En la edición 2012 uno de los puntos de discusión más importante fue el referido a cómo definir métricas para medir la calidad de las visualizaciones (QoV) para contribuir a la madurez de la disciplina.

2.5 PSICOLOGÍA COGNITIVA

En esta sección introducimos algunos conceptos y teoría de psicología cognitiva que están fuertemente relacionados con la visualización de la información, pues la visualización es un amplificador de la capacidad de aprendizaje y comprensión del conocimiento. El objetivo de los sistemas de visualización es ayudar a resolver problemas como detectar tendencias y comportamientos. Es por eso que la psicología cognitiva es de suma relevancia en la visualización de la información pues para lograr visualizaciones efectivas, es importante entender como los seres humanos aprendemos y absorbemos conocimiento.

2.5.1 INTRODUCCIÓN

El concepto de aprendizaje está fuertemente relacionado con la visualización de la información. La visualización de la información define un esquema en el cual los datos se transforman en información y dicha información se representa de manera visual, entonces es cuando la información se transmite a nuestro cerebro a base de estimular nuestra percepción sensorial creando en función del contexto, la cultura y la experiencia previa del observador una experiencia netamente cognitiva.

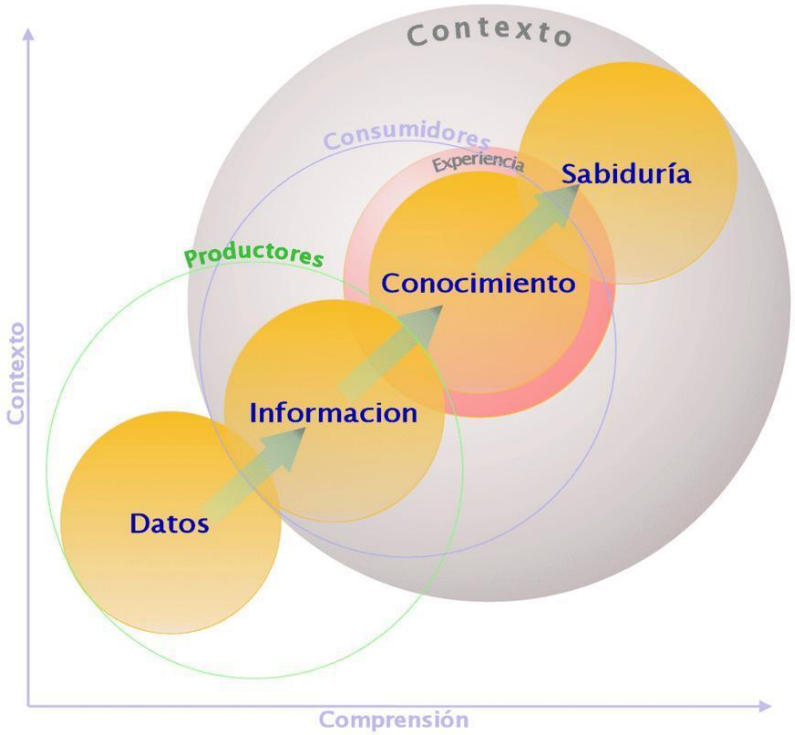


Figura 12 - Diagrama de Shedroff: Esquema elemental de conversión de datos en sabiduría.

Según Shedroff [12] y como vemos en el diagrama de la figura 12, existen cuatro elementos conceptuales principales, que son los siguientes, según el artículo de *Juan C. Dürsteler, [15]* podemos citar las siguientes definiciones :

- **Datos: "***Los datos son simples hechos, carentes de contexto. Si no nos informan, no son información, o lo que es lo mismo desprovistos de contexto son simplemente la materia en bruto del que partimos para la comprensión. 07012007 es un dato, que puede tener muchos significados, una fecha, el nombre de un lote de fabricación, un cumpleaños, etc.*"

- **Información**: "*La información son los datos puestos en contexto. Es un concepto ligado al de metadato, un dato que hace referencia al significado de otro dato. Por ejemplo, si en una tabla de datos una de las columnas reza número de lote, 07012007 una tira de caracteres en esa columna cobra un significado particular. La información es la destilación de los datos o los datos con su significado, pero aun no son conocimiento."*

- **Conocimiento**: "*Lo que diferencia el conocimiento de la información es la complejidad de las experiencias que se necesitan para llegar a él. Para que un conjunto de informaciones se conviertan en conocimiento hay que estar expuesto a él de diferentes maneras y hay que elaborar una experiencia propia respecto al mismo. El conocimiento se puede expresar como un patrón cuya medida de interés para el usuario supera un cierto umbral. Esto es, si no nos interesa es difícil que la información se convierta en conocimiento. El conocimiento no es transferible, se lo fabrica uno mismo experimentando la información.* "

Shedroff recomienda el "diseño de experiencias" como la manera de generar las experiencias que construyen conocimiento de forma más óptima.

- **Sabiduría**: "*Es el nivel último del entendimiento. Cuando entendemos un abanico suficientemente amplio de patrones y meta-patrones de forma que los podemos utilizar y combinar de formas y situaciones nuevas totalmente diferentes a las que nos sirvieron para aprender. La sabiduría es, como el conocimiento, algo personal que se elabora íntimamente y que va con las personas y se pierde con ellas, a diferencia de los datos y la información."*

Dichos elementos los hemos representado como una evolución en el tiempo, sobre dos ejes, por un lado tenemos el incremento paulatino de la comprensión y por otro el aumento de la importancia que tiene el contexto entendido como la cultura, la experiencia y el conjunto de patrones adquiridos.

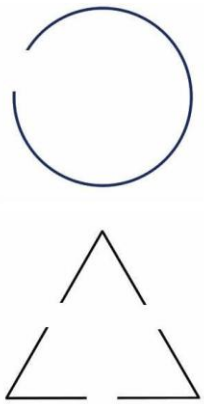
En el gráfico de la figura 12, como se puede observar, existen dos círculos, el primero denota el perímetro de las personas que generan la información a partir de los datos (los productores) y el segundo denota el perímetro de los que consumen dicha información y la procesan en conocimiento (los consumidores). Un círculo de contexto engloba la transición de información a Conocimiento y finalmente de éste a Sabiduría. El conocimiento está englobado en un círculo de experiencia.

El diagrama realizado por Shedroff no hace referencias a otros artefactos o gráficos, ni a la manera en que se realizan las transformaciones que dan lugar a la conversión de unas entidades en otras. Dicho diagrama es esencialmente conceptual.

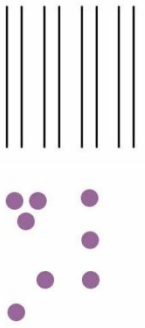
2.5.2 ESCUELA DE LA GESTALT

La Gestalt fue un movimiento surgido a principios del siglo veinte que se dedicó principalmente al estudio de la percepción del ser humano. El movimiento de la Gestalt postulaba que las imágenes son percibidas en su totalidad, como forma o configuración (del alemán, Gestalt), y no como la suma de sus partes constitutivas. Por otro lado el contexto dentro de esta teoría juega un papel muy importante en el proceso perceptivo, el movimiento formulo una serie de leyes para denotar los procesos perceptivos, dichas definiciones las citamos del artículo de *Milko A. García Torres* [16] sobre la teoría de la Gestalt*:*

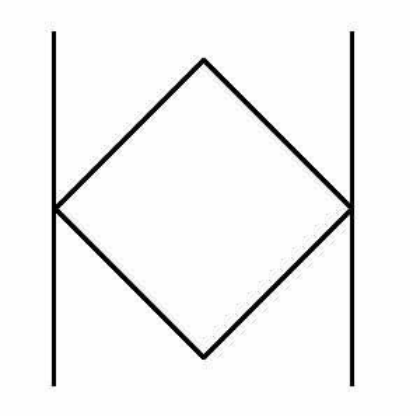
- **Ley de cierre**: *"Nuestra mente añade los elementos faltantes para completar una figura. Existe una tendencia innata a concluir las formas y los objetos que no percibimos completos. Nuestra mente ve tanto el círculo como el triángulo, aunque no sean elementos completos."*



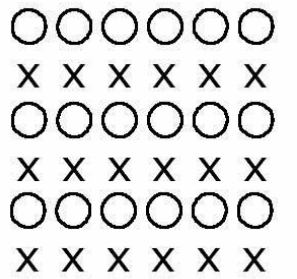
- **Ley de proximidad: "***Los elementos tienen a agruparse con los que se encuentran a menor distancia. Influyen también la semejanza de la forma, el tamaño, el color y otros aspectos visuales de los elementos. Así en el primer grupo, percibimos más prontamente caminos estrechos, y no anchos; y en el segundo caso, percibimos antes el pequeño grupo de tres, antes que la forma de cinco de la derecha."*



- **Ley de simplicidad:** *"Cuando miramos una figura la percibimos de la manera más simple posible. Se percibe un diamante o rombo, pero nadie aprecia las dos letras "K" una frente a la otra."*



- **Ley de similitud: "***Tendemos a percibir agrupados los objetos iguales, miramos las filas de círculos y cuadrados, pero no apreciamos las columnas."*



- **Figura y fondo:** *"Cualquier campo perceptual puede dividirse en figura contra un fondo. La figura se distingue del fondo por características como: tamaño, forma, color, posición, etc."*

2.5.3 ASPECTOS DE LA PERCEPCIÓN

El ser humano crea una percepción tridimensional del mundo a través de imágenes en dos dimensiones proyectadas en la retina, Es por ello que podemos decir que la percepción es un proceso esencialmente creativo en sí mismo. Las personas son capaces de reconocer objetos aunque la imagen proyectada en la retina varié según la luz. El brillo de una imagen proyectada sobre la retina va cambiando a medida que una persona se va moviendo o la luz ambiental se va cambiando.

A partir de los detalles de una imagen visual y un contexto dado, el cerebro desarrolla preceptos complejos, combinando las partes que más se correspondan al mundo real, y luego se basa en experiencias posteriores y en la organización de las conexiones, dicho proceso es continuo y dinámico, En principio una persona selecciona una porción de la imagen como foco principal mientras el resto queda en segundo plano. El centro de atención de una imagen queda delimitado por sus bordes y límites, en base a esto las personas reconocen imágenes.

La manera en la cual el cerebro organiza la percepción, distorsión, selección y llenado de omisiones queda determinada en un conjunto de leyes referidas al proceso perceptivo mencionadas en el apartado anterior de la escuela de la Gestalt, que están relacionadas con la similitud y la proximidad en las alusiones

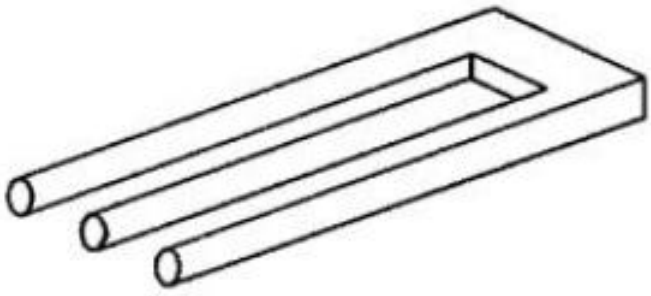


Figura 13 - Imagen que provoca tendencia a formar objetos irreales.

Tomar en cuenta estos aspectos es de suma importancia a la hora de desarrollar y diseñar visualizaciones para que estas sean efectivas, Pues permite al diseñador conocer anticipadamente como determinadas características visuales serán percibidas más rápidamente que otras, cuáles variables visuales serán buenas para discriminar determinadas características de los datos y cuáles podrán generar confusión en el observador entre otras cosas. Se deben tomar en cuenta los problemas de interpretación subjetiva asociados con la percepción de imágenes. Los elementos que denotan el contexto en la vista deben reforzar la metáfora visual y ser consistentes de principio a fin.

2.6 COMUNICACIÓN VISUAL

Según el diccionario enciclopédico Salvat, Una imagen es, desde un punto de vista filosófico, la conciencia de un objeto ausente o inexistente y en psicología es la representación construida al margen de los correspondientes estímulos sensoriales.

En las siguientes secciones describiremos como es que a través de una imagen es posible obtener conocimiento, Como a partir de otras imágenes podemos construir otras más complejas de manera de transmitir mas información o hacer que la transmisión sea más efectiva y luego por ultimo veremos cómo es el proceso general a partir del cual se diseña una visualización y las buenas prácticas asociadas al diseño.

2.6.1 LA CODIFICACIÓN GRÁFICA

La visualización es una concepción de la mente que va mas allá de la percepción sensorial y que como tal construcción se acerca al conocimiento, que es la captura intelectual de las cosas, comprender una porción de información quiere decir abordar e interiorizar dicha información.

La definición más común y aceptada del concepto de información y la que figura en la mayoría de los diccionarios es la que dice *"conocimiento adquirido a través de la experiencia o el estudio*", mientras que para otras personas es la "*comunicación o adquisición de conocimientos*", básicamente la información se basa en la elaboración de los datos crudos que se pueden recoger de los objetos o los fenómenos, para construir el conocimiento.

Los gráficos tienen la habilidad de mostrar grandes cantidades de información y permiten que dichos datos se puedan organizar de manera coherente con el objetivo de facilitar la compresión, permitir la comparación y beneficiar la comunicación y la interpretación. Los gráficos son probablemente el mecanismo visual más eficiente para representar información de manera efectiva y resumida, Pues los gráficos tienen un gran poder expresivo y aprovechan al máximo las habilidades perceptivas y cognitivas del ser humano, de esta manera es posible transmitir conceptos complejos de manera clara y eficiente.

2.6.2 SEGMENTACIÓN VISUAL

Podemos decir que una imagen puede construirse a partir de elementos más pequeños, es decir que una imagen se puede segmentar en elementos más pequeños los cuales podemos denominar atributos o elementos visuales. Entre ellos podemos considerar los polígonos, las líneas, los puntos, colores, textos y otras imágenes.

La combinación de estos elementos gráficos y el mapeo de los distintos atributos de los datos con los elementos visuales, conforman las bases para desarrollar metáforas visuales que permiten visualizar información para un dominio dado.

Figura 14 - Ejemplo de segmentación, mediante atributos, formas y imágenes más pequeñas formamos una visualización con el objetivo de transmitir información al observador.

Es muy importante saber aplicar y combinar estos elementos de manera adecuada con el objetivo de obtener una visualización efectiva para un contexto dado, Para ello en general se requiere un conocimiento previo del dominio del problema en cuestión y capacidad suficientemente creativa como para plasmar una metáfora visual de buena calidad, además de ser necesario conocer mínimamente las características de los datos como por ejemplo de que tipo son(unidimensionales, bidimensionales, temporales, etc.), qué unidades tienen (km, mts, kg, etc.), las escalas, entre otras propiedades así como también saber seleccionar la paleta de colores adecuada.

2.6.3 EL DISEÑO DE LAS VISUALIZACIONES

Como vimos en la reseña histórica sobre la evolución de la visualización a lo largo del tiempo, En un principio, antes de la era de la computación, los medios para desarrollar visualizaciones eran unidireccionales, el diseñador era el responsable de transmitir una idea o concepto previamente existente, él era el responsable de organizar los datos brutos y seleccionar una técnica para representar la información contenida en ellos de manera apropiada, estas visualizaciones tenían la característica de ser estáticas y no permitían al observador interactuar con dicha información, es decir que el observador en si mismo tenía un rol pasivo. Adicionalmente también era la persona que debía poseer un conocimiento extenso del dominio del problema de manera de poder destacar tendencias y patrones relevantes en la información representada.

Hoy en día con el avance de la tecnología que ha alcanzado a todas las disciplinas científicas y todos los ámbitos cotidianos, esto ha cambiado drásticamente. Ahora el diseñador es el responsable de concebir sistemas y herramientas visuales capaces de brindar métodos interactivos para manipular la información. El observador ahora tiene un rol activo en el cual a partir de las herramientas visualizadas puede detectar tendencias y patrones en la información representada así como generar conocimiento a partir de dicha información.

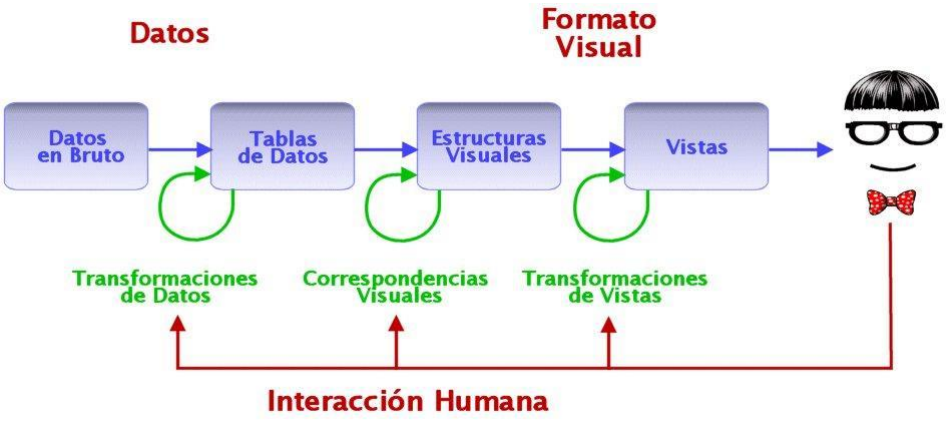


Figura 15: Modelo de diseño de una visualización en los tiempos de la computación.

Bajo esta nueva modalidad de diseño (Figura 15), El diseñador solo es responsable de proveer una herramienta capaz de permitir una mirada inicial de los datos y los métodos interactivos para poder manipular la información representada. De esta manera el rol del observador pasa a hacer activo. Participa activamente en la búsqueda del *insight* de los datos utilizando y usando apropiadamente las capacidades interactivas de la herramienta provista por el diseñador.

2.6.4 BUENAS PRÁCTICAS DE DISEÑO

En esta sección se van a describir dos conjuntos de buenas prácticas para el diseño de visualizaciones, El primero propuesto por Ben Shneiderman y el segundo por Edward Tufte, es muy importante que a la hora de abordar un proyecto de visualización se tengan en cuenta buenas prácticas y consejos de diseño para obtener buenos resultados.

Ben Shneiderman [6] propone 7 reglas de oro para el diseño efectivo de visualizaciones clasificadas por el tipo de dato que caracteriza a la información que debe ser representada:

1. **Overview**: La idea es poder ver un panorama general de toda la información, es decir tener una vista general en donde esté representada toda la colección de datos. Y luego disponer una vista secundaria que permita ver el detalle al hacer zoom en una porción particular de la visualización, esto se puede lograr dotando de controles para manejar la escala y también un control del estilo ojo de pez con poder de magnificar una determinada porción de datos. Este tipo de estrategias son muy utilizados por los sistemas geográficos para visualizar mapas y focalizar sobre una determinada porción sin perder la vista principal de foco.

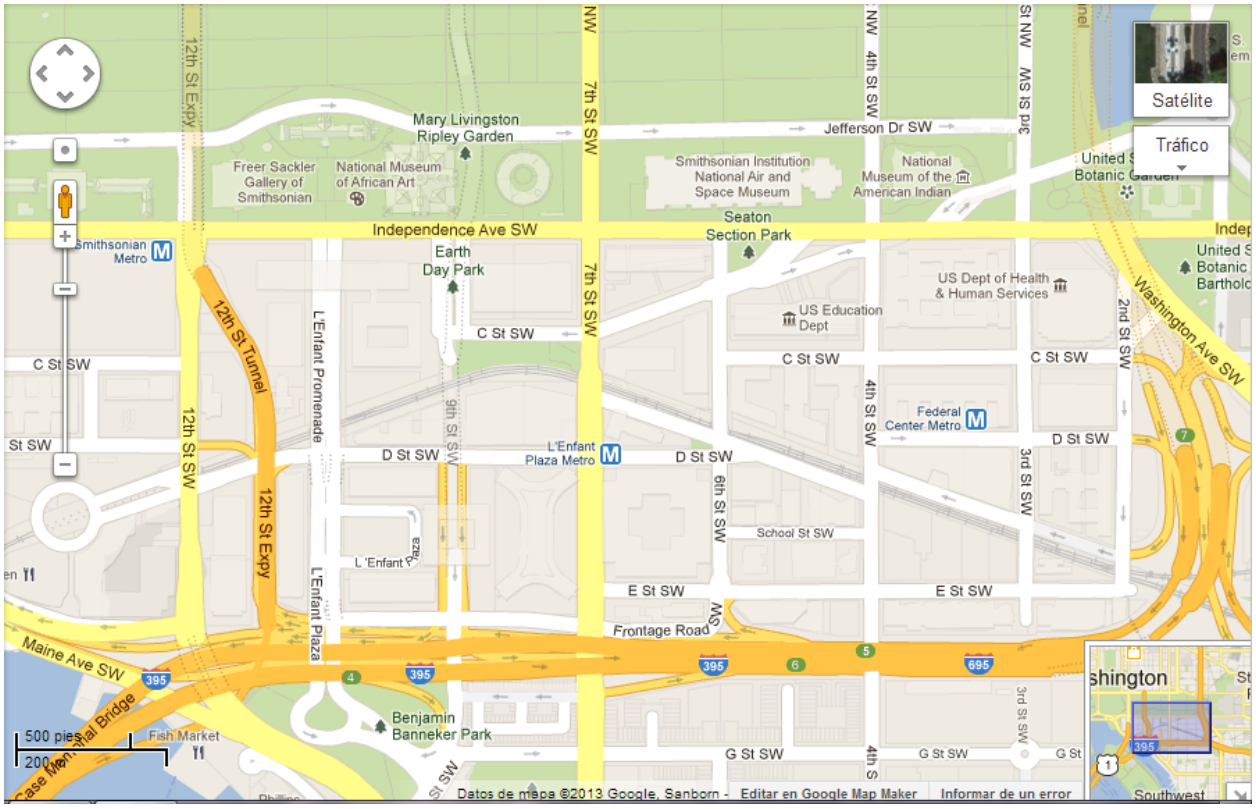


Figura 16 - *Google Maps* con control de *overview* en la parte inferior derecha.

2. **Zoom**: Se debe permitir hacer zoom sobre elementos de interés, En general el observador tiene interés en alguna porción en particular de la colección de datos y es necesario dotar de mecanismos que permitan controlar el zoom y el foco sobre una determinada porción de dicha información, que para el usuario puede resultar de interés. Es importante que esta transformación de escala se realice suavemente de manera de preservar el sentido de posición y contexto. *Google Maps* (Figura 16) cuenta con mecanismos de zoom con transiciones suaves.

3. **Filtros**: El observador tiene que tener la posibilidad de poder filtrar la información que no le resulte de su interés, eliminando la información representada de la visualización, Esto se puede lograr mediante mecanismos que controlen el contenido de la visualización, para de esta manera hacer foco sobre la colección de datos que realmente interesa. se recomienda usar controles que permitan un filtrado rápido incluso para muestras de miles de registros, de esta manera el observador puede focalizarse en la información que realmente puede llegar a ser importante para obtener tendencias y patrones.

4. **Detalles bajo demanda**: Permite al observador seleccionar un item determinado cuando sea necesario y permitir navegar la información contenida en él de manera detallada. Esto evita tener que mostrar toda la información de golpe, que lleva a generar confusión y produce una sobrecarga en la vista de manera innecesaria afectando la performance global y la efectividad de la visualización.

5. **Relacionar**: La idea es poder ver relaciones entre los items, Esto en la realidad es muy difícil de lograr de acuerdo al contexto y al dominio de los datos. Un ejemplo donde se ve aplicada esta buena práctica es el sistema de visualización orientado al dominio de la filmografía llamado *FilmFinder* [9]

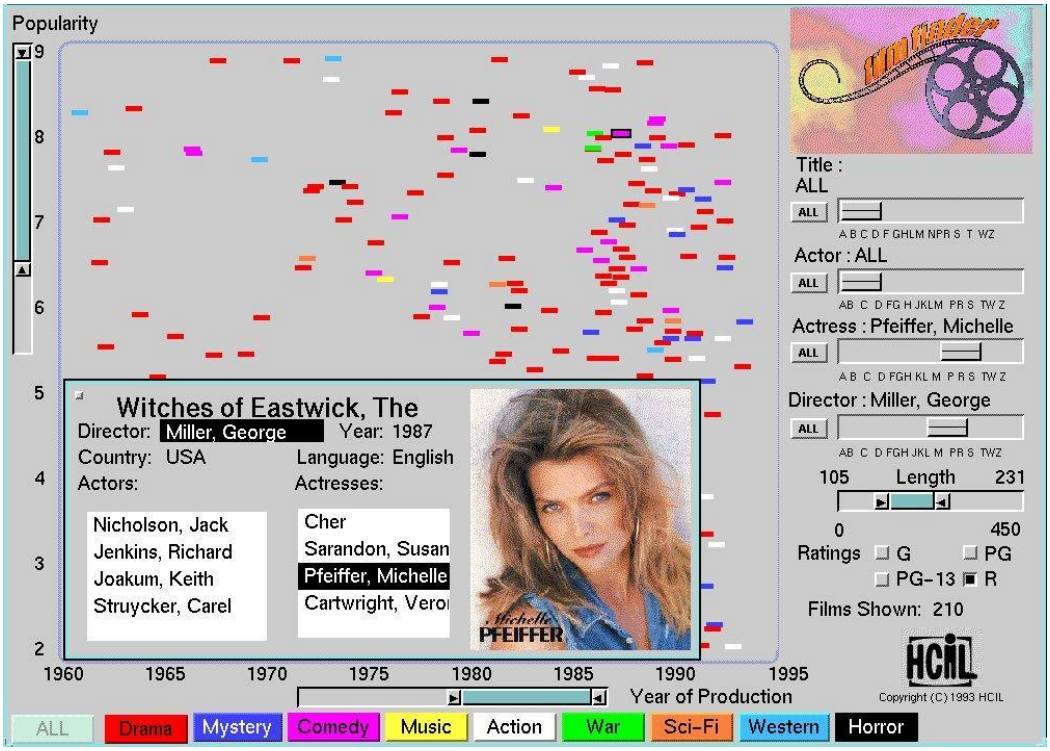


Figura 17 - *FilmFinder*, al seleccionar el Director del objeto que representa una película, se iluminan el resto de las películas dirigidas por el mismo director.

6. **Historial**: Mantener un historial de acciones, de manera que el usuario pueda ir para atrás y revertir sus acciones o realizarlas nuevamente con el objetivo de obtener un refinamiento progresivo. Es poco común que una sola acción sobre la visualización lleve al resultado esperado.

7. **Extracción**: Permitir al usuario extraer subconjuntos de información de la visualización de manera que pueda exportarla a un medio conveniente como un archivo de texto con un determinado formato que facilite otros usos como enviar por correo, imprimir, graficarlo, etc.

Luego tenemos la premisas de un buen diseño de Tufte que en [1] describe un conjunto de buenas prácticas para el diseño de visualizaciones con el objetivo de lograr la excelencia gráfica cuya principal premisa es la de obtener una presentación bien diseñada de información interesante. Según este autor la excelencia gráfica consiste en:

- transmitir ideas complejas con claridad y precisión;
- proveer al observador un gran número de ideas en el menor tiempo que sea posible usando la menor cantidad de tinta;
- contar la verdad acerca de los datos, es decir permitir sacar conclusiones y tendencias.

Para ello se propone un set de buenas prácticas que se detallan a continuación:

- **Mostrar los datos**

Es lo básico de cualquier representación gráfica de un conjunto de datos, es lo mínimo que se puede esperar de un sistema de visualización.

- **Maximizar la tinta para Datos**

La idea es tratar de mostrar sólo lo que es realmente es relevante al observador evitando desperdiciar tinta para datos cuando realmente no hace falta. Todo lo que sea redundante y que no aporte nada debe ser eliminado. Para este concepto, Tufte posee una particular métrica tomando como variables a la cantidad de datos y a la tinta que se utiliza en el gráfico:

$$La\ razón\ Datos\text{-}Tinta = \frac{Datos\text{-}Tinta}{Total\ de\ tinta\ usada\ en\ imprimir\ el\ gráfico}$$

- **Uso apropiado de los colores**

El objetivo del color es resaltar ciertas características de la información que sobresalen por sobre las demás, siempre es recomendable usar esquemas de colores agradables a la vista y evitar usar combinaciones que produzcan mucho contraste.

En [5] Edward Tufte expresa lo siguiente sobre el problema de seleccionar los colores para la visualización de la información:

"Evitar catástrofes se convierte en el primer principio al brindar color a la información: Por encima de todo, no provocar daño"

El color usado apropiadamente puede mejorar y clarificar una presentación; usado erróneamente puede obstruir y generar confusión. El uso más importante del color en la representación de la información es distinguir un elemento de otro, resaltar las características más relevantes. Tufte recomienda utilizar colores provenientes de paisajes naturales.

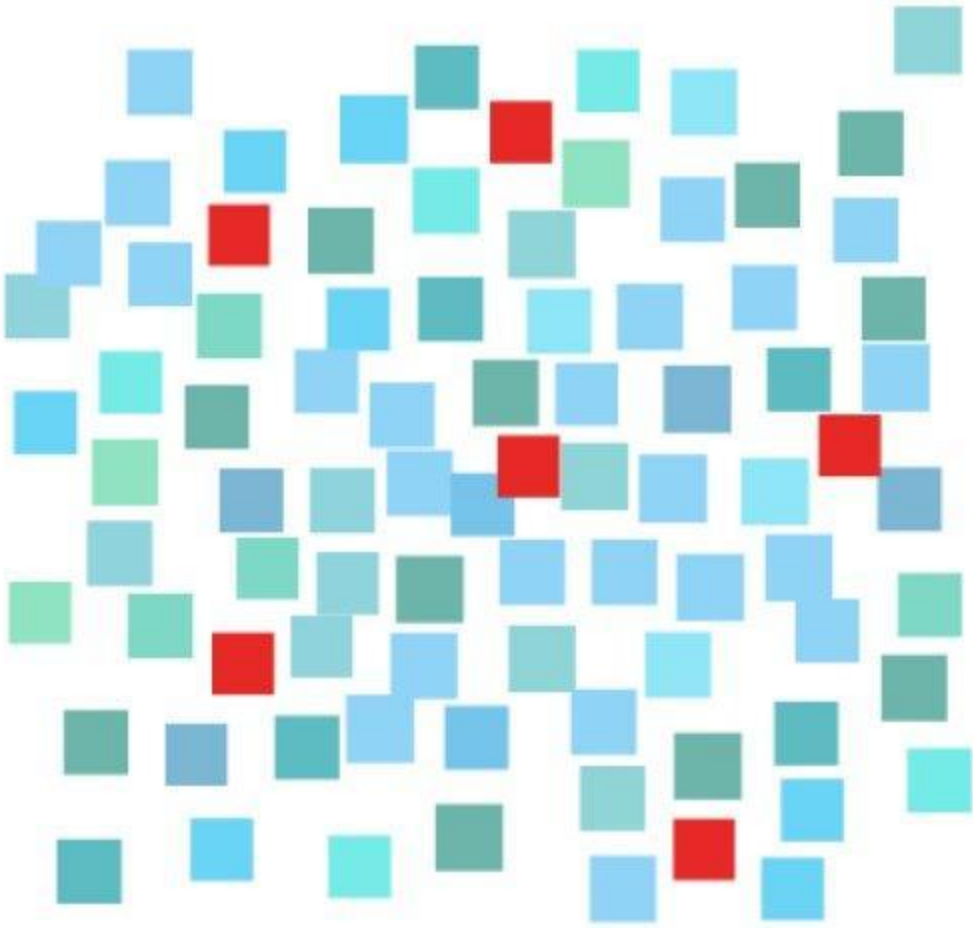


Figura 18 - Contraste y analogía. Los cuadrados rojos contrastan con los análogos azules y verdes y se destacan por encima del resto.

- **Small Multiples**

Un "*Small Multiple*" (también llamado *Trellis chart, Lattice chart, grid chart*) es una serie o matriz de gráficos o charts similares, cuyo objetivo es permitir la comparación de cada uno de ellos fácilmente. Para ello Tufte [5] recomienda utilizar pequeños *Small múltiples* con la misma escala y asignación de colores.

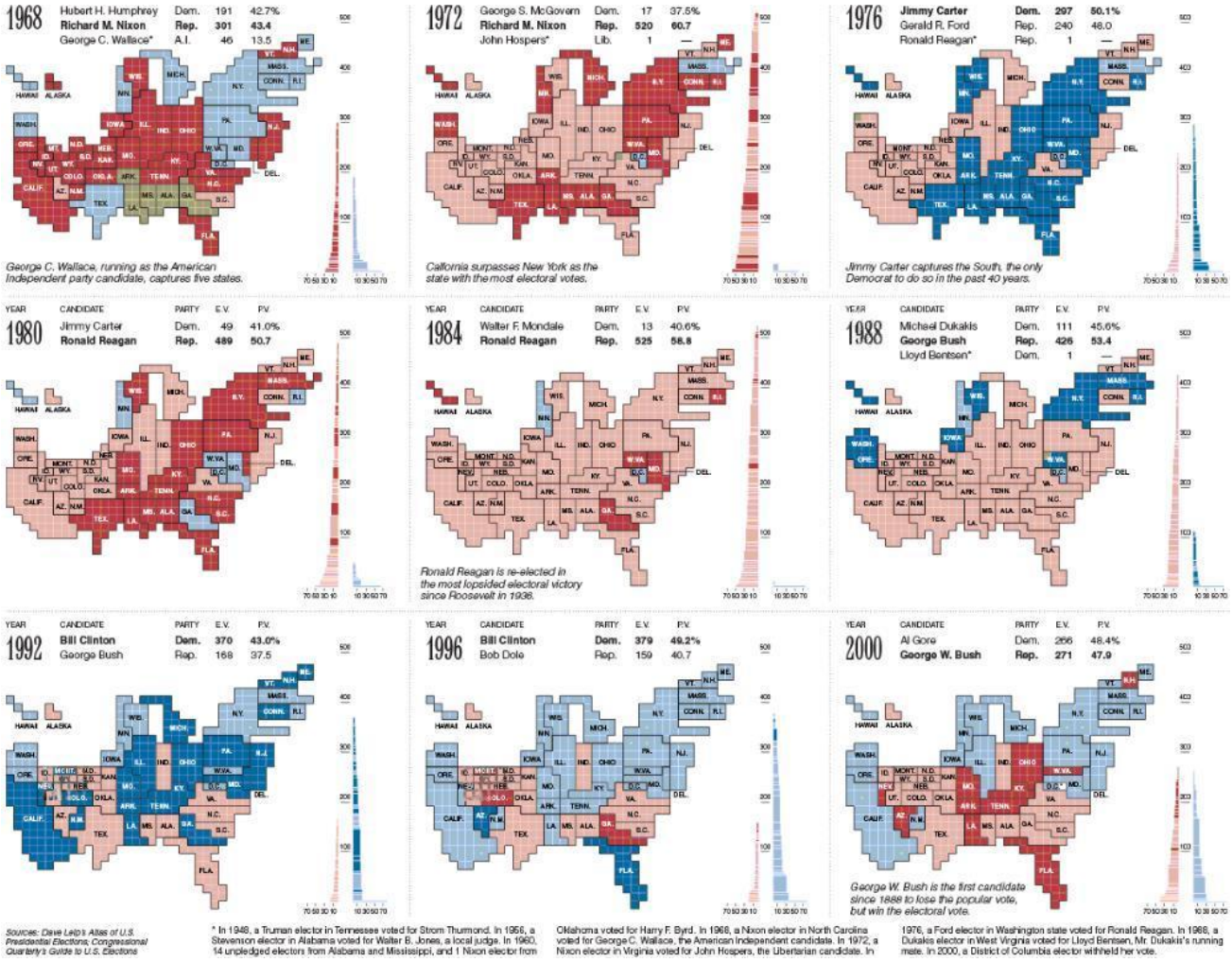


Figura 19 - Ejemplo de S*mall múltiple* que muestra la distribución de votos a lo largo de los años en Estados Unidos

- **Evitar el chart junk**

El concepto de "*Chartjunk*" definido por Tufte en [1], se refiere a todos aquellos elementos gráficos en las visualizaciones que no son necesarios para comprender la información representada en el gráfico o que distraen al observador de la información allí representada generando confusión.

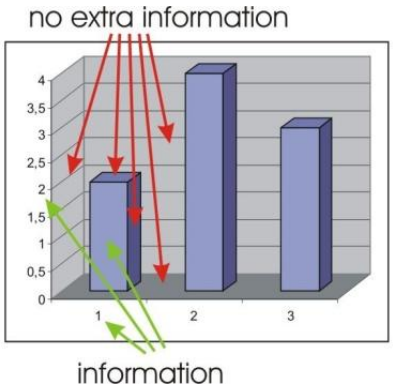


Figura 20 - Ejemplo de *Chartjunk* en un gráfico de barras, la tercera dimensión no aporta nada en este contexto

- **Usar elementos Multifuncionales**

Para minimizar la basura en los gráficos (*Chartjunk*) y maximizar la tinta destinada para representar la información, Tufte propone utilizar elementos multifuncionales en los gráficos, un ejemplo concreto es la utilización de leyendas en los datos para remarcar características destacables en los datos.

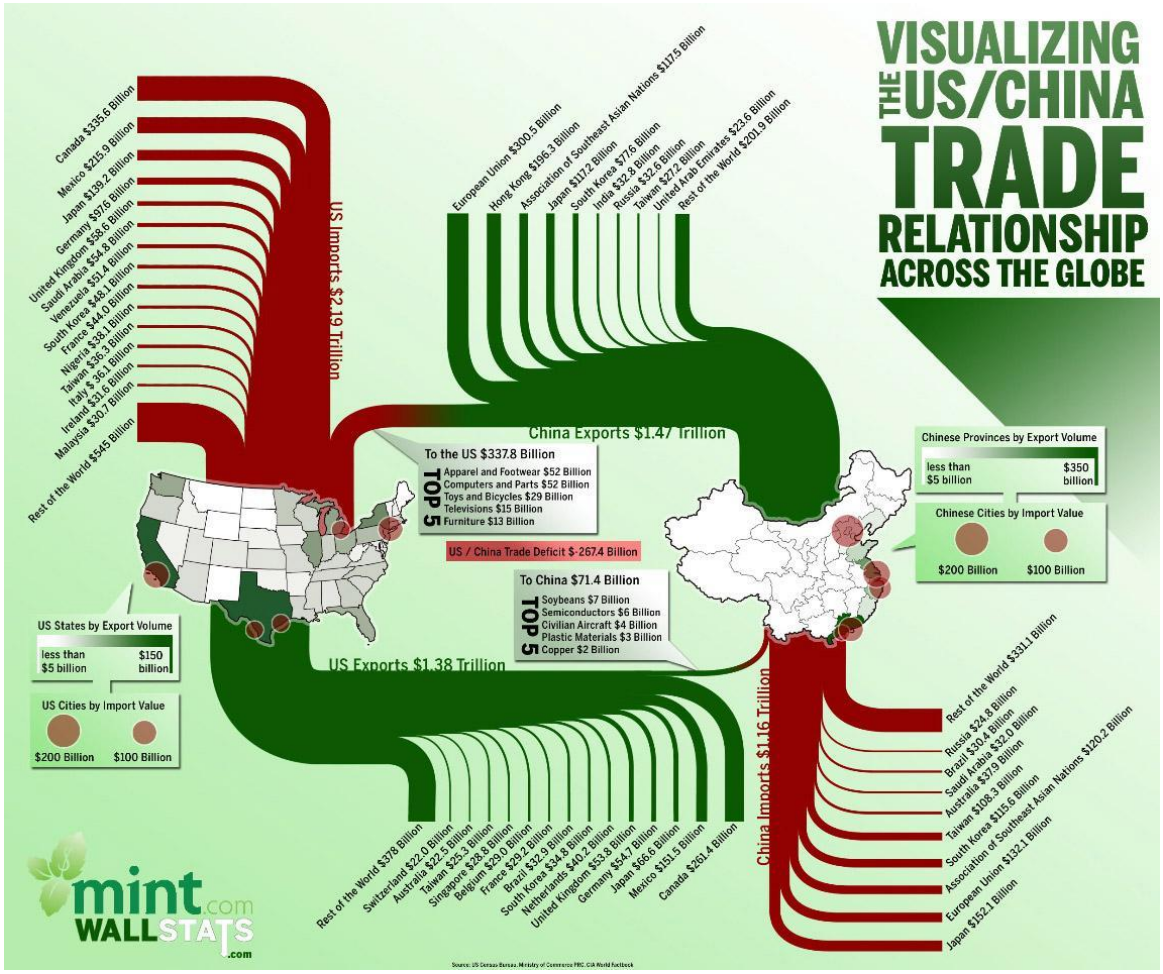
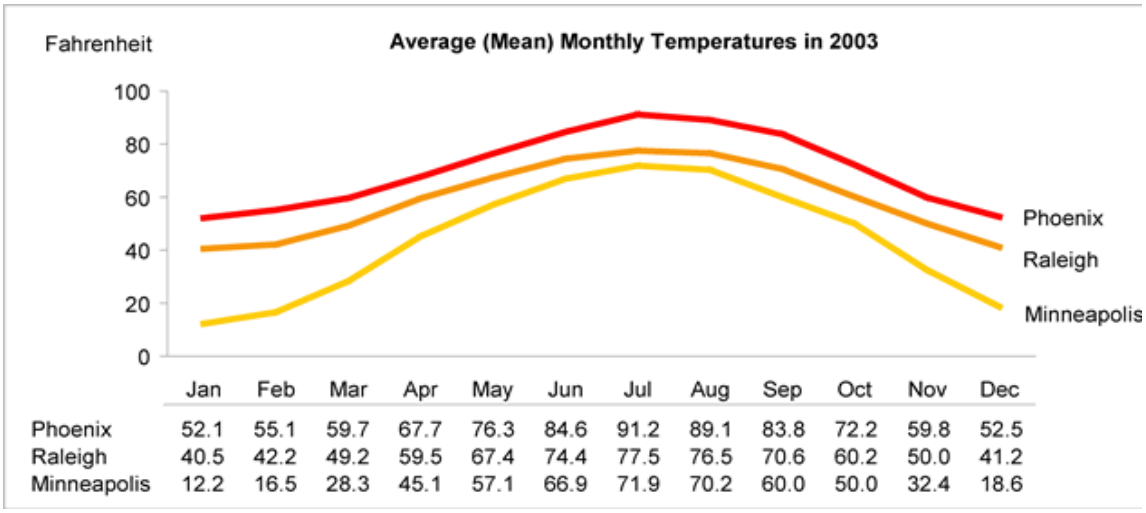


Figura 21 - Ejemplo de gráficos multifuncionales en donde los colores, la saturación, tamaño y posición pueden representar múltiples valores simultáneamente.

- **Textos en los gráficos**

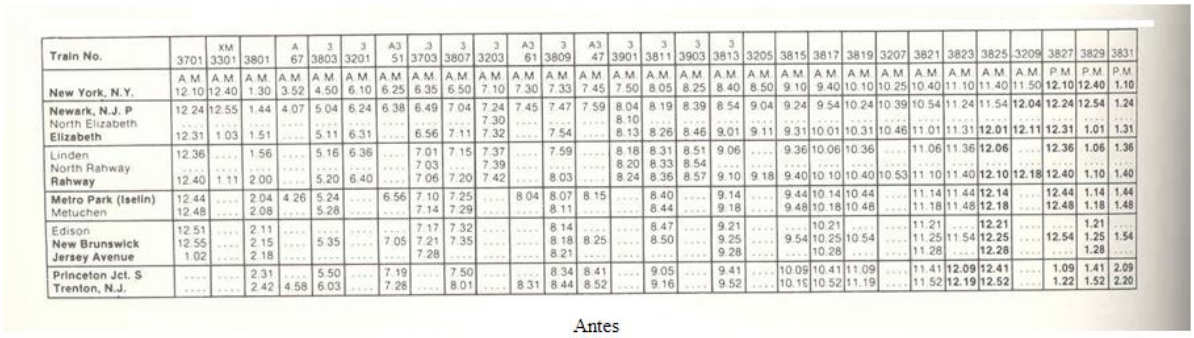
Utilizar textos en los gráficos apropiadamente, siempre hay que tratar de evitar poner leyendas en un espacio aparte y lo que dice Tufte es que el texto tiene que ir dentro del grafico como se observa en la figura 22.



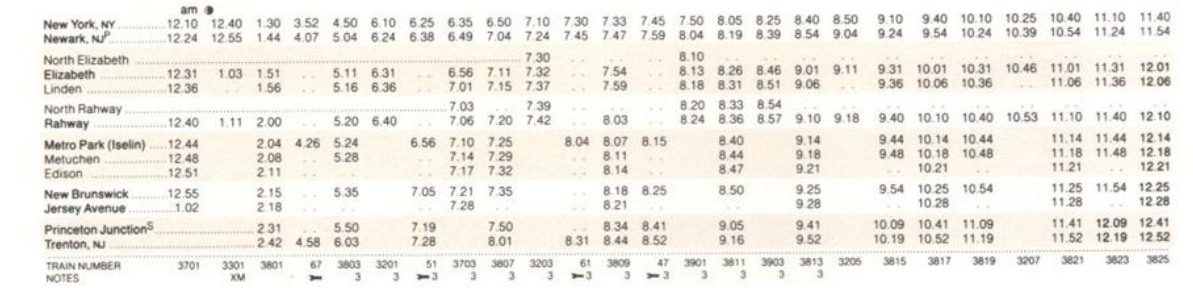
	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Phoenix	52.1	55.1	59.7	67.7	76.3	84.6	91.2	89.1	83.8	72.2	59.8	52.5
Raleigh	40.5	42.2	49.2	59.5	67.4	74.4	77.5	76.5	70.6	60.2	50.0	41.2
Minneapolis	12.2	16.5	28.3	45.1	57.1	66.9	71.9	70.2	60.0	50.0	32.4	18.6

Figura 22 - Grafico de barras que muestra la temperatura promedio en 3 ciudades con las leyendas dentro del gráfico.

Básicamente lo que se pretende es lograr una armonía entre todos los elementos gráficos de manera que el contenido del mismo sea lo más importante. Otras cosas que sugiere Tufte es usar grillas y etiquetas con bordes finos de manera que no queden demasiado resaltados en el gráfico, como por ejemplo se puede observar en la siguiente figura:



Antes



Despues

Figura 23 – Optimización de los bordes de la grilla según las recomendaciones de Tufte.

2.7 VISUALIZACIÓN DE DATOS CUALITATIVOS

Los datos cualitativos son naturalmente extremadamente variados, virtualmente pueden incluir cualquier tipo de información que puede ser numérica o no. A lo largo de la historia de la de visualización de información los métodos utilizados fueron poco intuitivos para los observadores. Para realizar una correcta visualización de la información es necesario diferenciar el tipo de datos que se van a utilizar. Ben Shneiderman [6] estableció una taxonomía de tipos de sistemas de visualización en base a los diferentes tipos de datos.

La clasificación de Ben Shneiderman [6] tiene como objetivo hacer un mapeo entre los problemas que típicamente se buscarán resolver en una estructura de determinado tipo de dato y como se visualizaría dicha estructura para facilitar la resolución de dichos problemas. Por ejemplo, si tuviéramos información bidimensional correspondiente a un mapa los usuarios desearían poder calcular caminos, rutas y distancias entre puntos determinados, si tuviéramos una estructura de árbol los usuarios desearían ver la relación entre padres y hijos. En base a esto, Shneiderman propone una taxonomía de sistemas para cada uno de los siguientes tipos de datos:

- Unidimensional
- Bidimensional
- Tridimensional
- Multidimensional
- Temporal
- Jerárquica
- Red

Dependiendo de cada tipo de dato se van a ir empleando diferentes tipos de técnicas para su visualización.

- **Unidimensional**

Significa una variable (un tipo de datos)

Algunos problemas de interfaz de diseño que incluyen son las fuentes, los colores, el tamaño y como desplazar o métodos de selección que pueden ser utilizados. Problemas de los usuarios pueden ser encontrar el número de elementos, consultar los artículos que tienen ciertos atributos (mostrar sólo las líneas de un documento que son títulos de las secciones, las líneas de un programa que se han cambiado desde la versión anterior, o en una lista de personas que son mayores de 21 años) o ver un artículo con todos sus atributos.

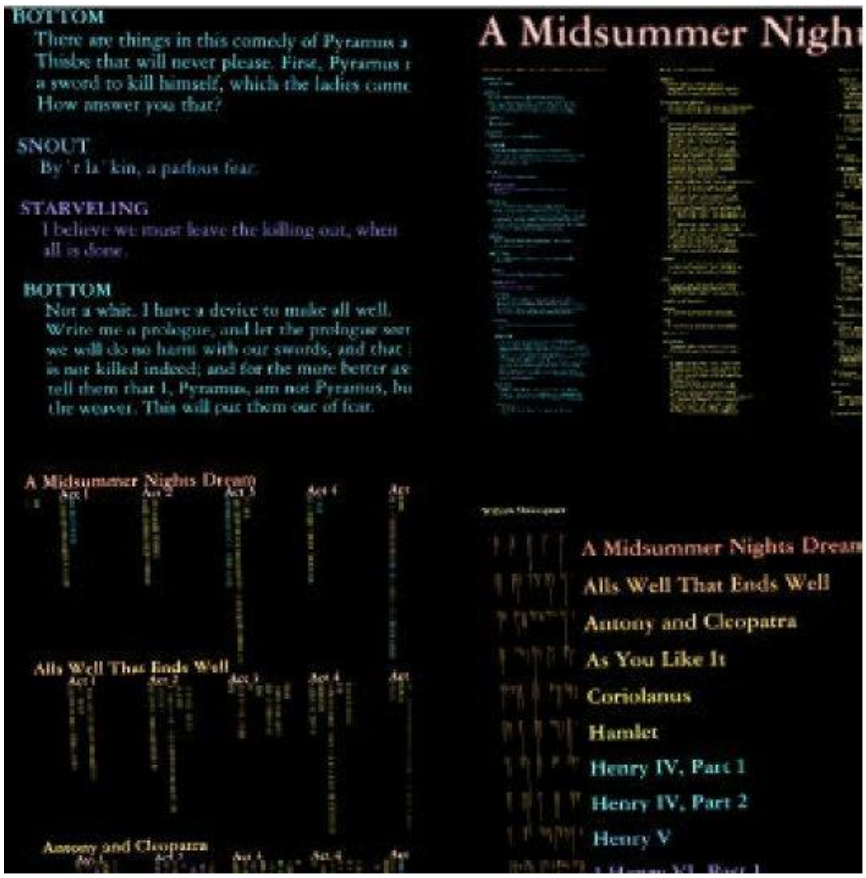


Figura 24 -Proyecto Shakespeare, que permite navegar las obras de dicho autor.

Lo que se pretende al mejorar visualizar texto es mejorar la visualización de grandes volúmenes de texto.

Un buen ejemplo de un sistema para datos unidimensionales es el *Shakespeare project* [10]. El proyecto permite ver de una manera original y eficiente la totalidad de los textos escritos por Shakespeare con un tipo de letra determinado y con diferentes colores así como también comparar los diálogos de los diferentes personajes.

- **Bidimensional**

Sistemas de dos variables, es decir que hay dos tipos de datos. Sistemas típicos que utilizan tipos de datos de 2 dimensiones son los llamados sistemas GIS (Sistemas de Información Geográfica). Los sistemas GIS se usan desde hace bastante tiempo para logística de medios de transporte ya sean terrestres, marítimos y aéreos. Pero su uso se ha popularizado recientemente a través del internet, en donde existen sistemas o sitios que ofrecen servicios de localización y logística básicos, ejemplos de este tipo de sistemas son *Compumap* y otro muy popular es el *Google Maps*.

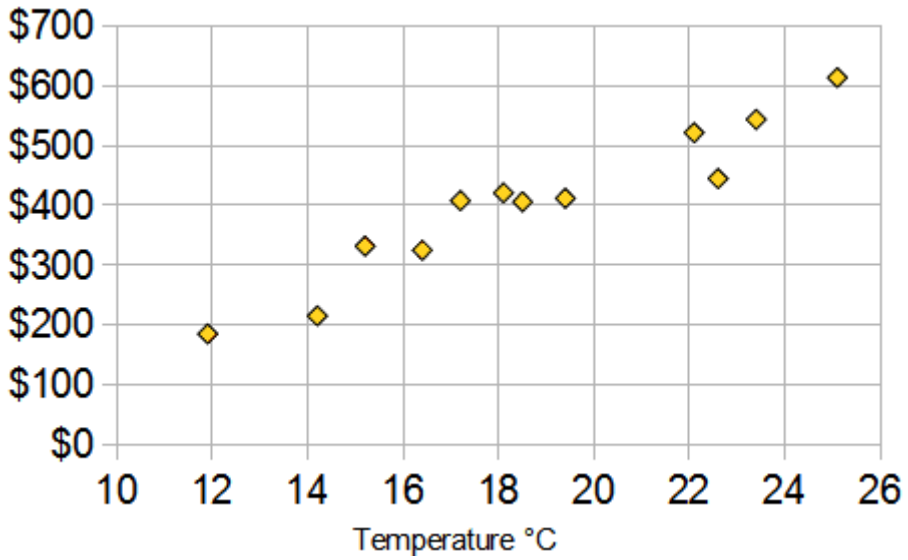


Figura 25 – Gráfico que muestra como un clima más cálido conduce a más ventas.

- **Tridimensional**

Objetos del mundo real como el cuerpo humano, las moléculas y las construcciones hechas por el hombre tienen elementos con volumen y estos objetos tienen relaciones potenciales con otros elementos, Es por ello que es necesario sistemas especiales que sean capaces de manejar relaciones complejas de 3 dimensiones, La visualización de datos tridimensionales es principalmente utilizada en entornos de investigación científica aunque cada vez más muchas herramientas comienzan a popularizarse en otros entornos. La visualización científica tiene como objetivo representar fenómenos físicos y naturales y objetos complejos en un modelo computarizado para luego tomar este modelo para poder realizar investigaciones y pruebas que no sería posibles realizarlas en el objeto o tendría un costo muy alto.

Un claro ejemplo de un sistema de este tipo es el 3D Studio, usado para el diseño de estructuras arquitectónicas, vehículos y aviones entre otras cosas. Este tipo de representaciones tridimensionales otorgan mayor realismo y facilidades para visualizar aspectos que no se podrían observar con una visualización en dos dimensiones tales como mapas, fotos, etc.

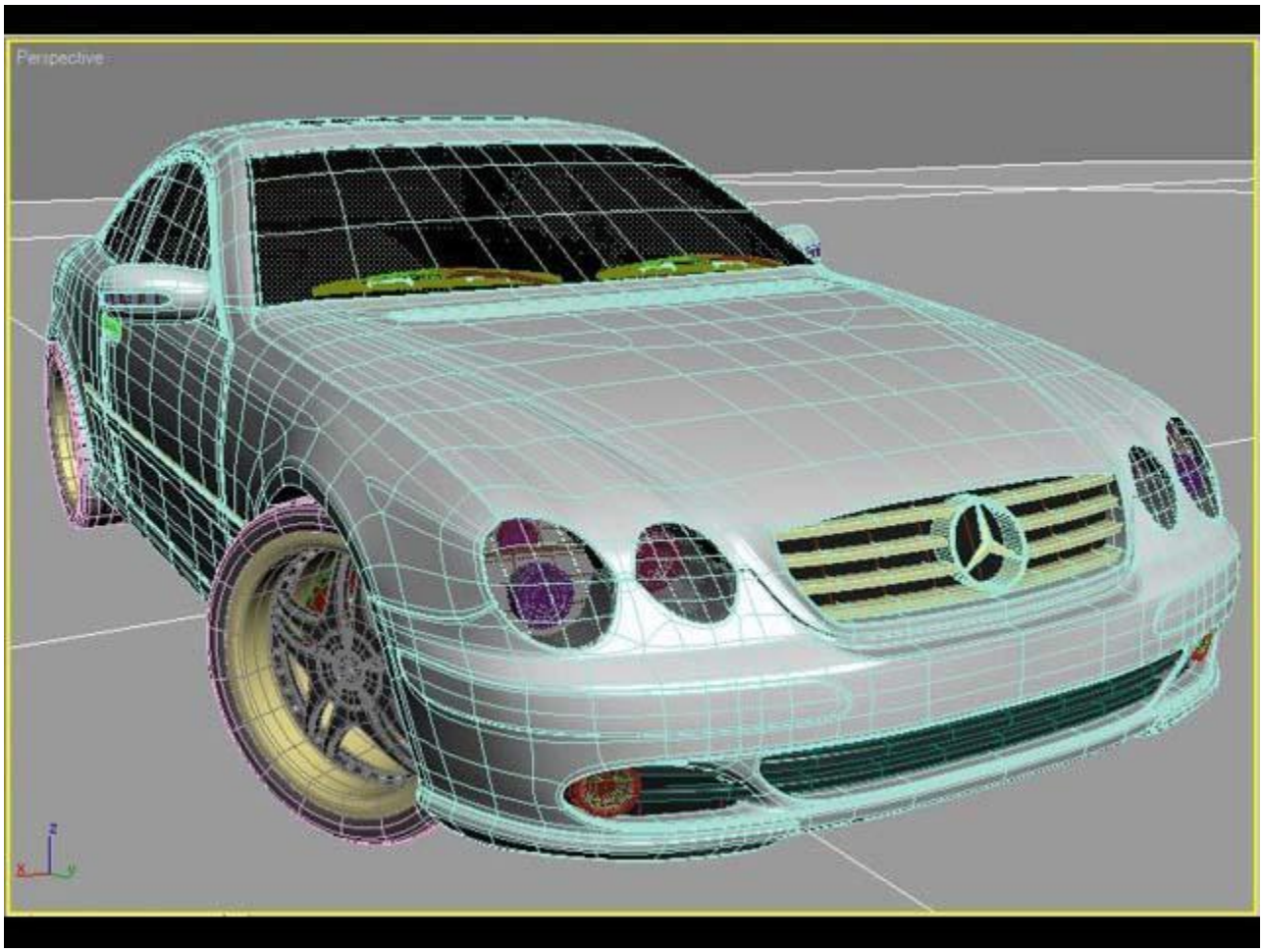


Figura 26 - Figura de un auto modelado enteramente en 3D en la herramienta 3D Studio.

- **Multidimensional**

Cuando un dato tiene más de tres dimensiones se dice que dicho tipo de dato es multidimensional. La mayoría de las bases de datos relacionales son manipuladas como base de datos multidimensionales en donde los elementos de n atributos se convierten en puntos en un espacio de n dimensiones. En general, los sistemas de visualización de datos multidimensionales, muestran información en donde el aspecto espacial no es el principal.

Por ejemplo si quisiéramos representar y visualizar información referente a una casa, se tendrían atributos del estilo: Nro. de habitaciones, Metros cuadrados, Nro. de baños, Distancia al subte, etc. Estos datos se convierten realmente en multidimensionales cuando el usuario o sistema reacciona ante dichos valores.

Un técnica típica para representar datos multidimensionales es la denominada "*Coordenadas paralelas*" [11] inventadas por Maurice d'Ocagne y popularizadas por Al Inselberg. Algunas aplicaciones importantes son sistemas de detección de colisiones, control de tráfico aéreo y estudios estadísticos.

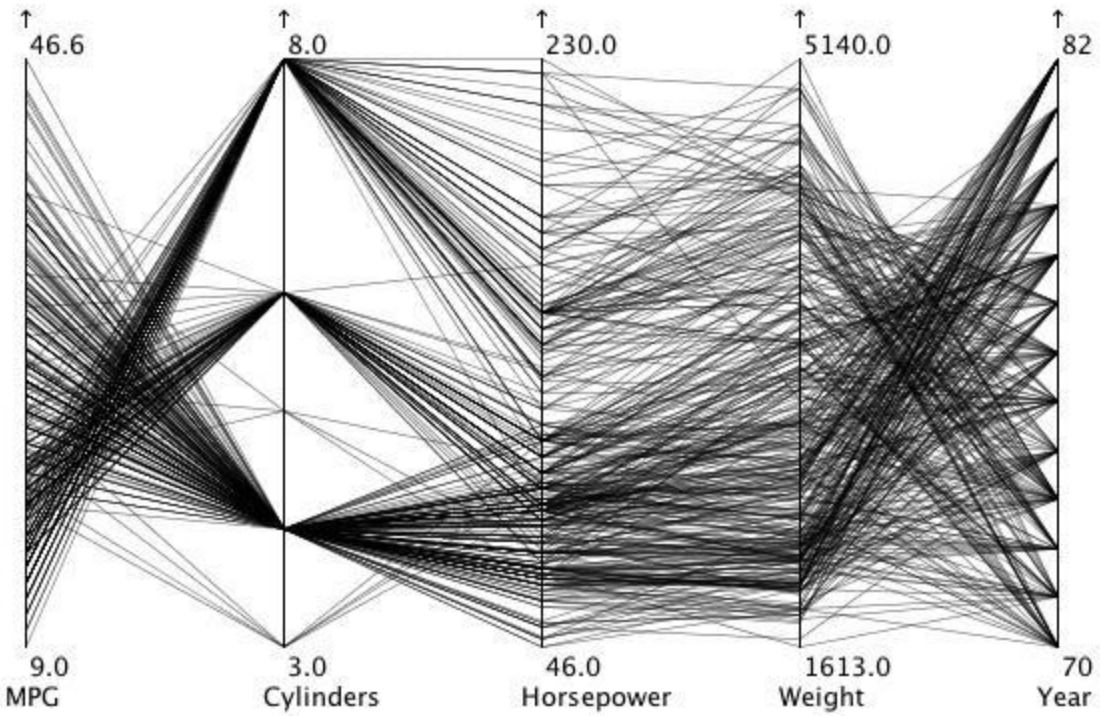


Figura 27 - Diagrama de coordinadas paralelas que describe los modelos de autos fabricados entre 1970 y 1982 y contiene sus kilometraje, cilindradas, caballos de fuerza, peso y año en el que fueron introducidos.

- **Datos temporales**

Las líneas de tiempo son ampliamente utilizadas para registros médicos y gestión de proyectos de toda índole, Dichas graficas proporcionan información para poder inferir que pasó, que puede ocurrir y que es lo que va ocurrir. De esta manera es posible tomar cursos de acción de acuerdo a los objetivos que se quieran lograr en esos tiempos representados. Existen muchas herramientas que representan, como por ejemplo Microsoft Project (figura 28) es una de las más populares y luego tenemos *perspective wall de* (Robertson et al., 1993*)* y *LifeLines* (Plaisant et al., 1996). *Lifelines* grafica la historia de los diferentes eventos de los jóvenes ocurridos a lo largo del tiempo con fines estadísticos, utilizado por el departamento de justicia juvenil del estado de Maryland, Estados Unidos.

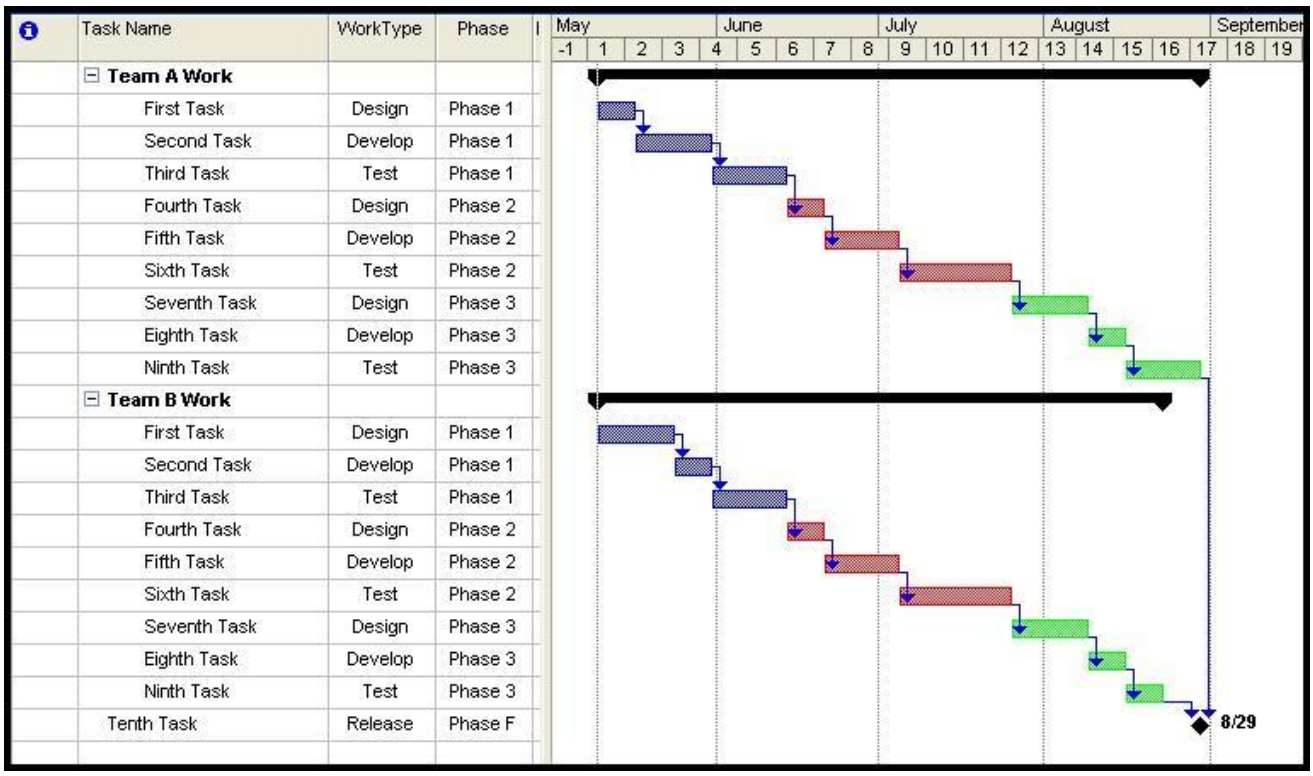


Figura 28 - Imagen de un proyecto de prueba en MS Project, esta aplicación utiliza líneas de tiempo para representar las tareas, el cumplimiento de metas y desvíos entre otras cosas.

- **De red**

Las redes son básicamente una estructura de nodos y ejes entre dichos nodos con atributos asociados tanto a los ejes como a los nodos. Podemos verlos como arboles con múltiples ejes y ciclos, un árbol tiene la característica de no tener ciclos. También llamados grafos, dichas estructuras son ampliamente utilizadas para representar diferentes tipos de redes, como por ejemplo las diferentes ciudades o localidades (nodos) y los caminos entre ellas (ejes). La red metro (Figura 29) entre otros usos típicos. En la actualidad la representación de redes es bastante compleja pues al igual que los arboles al tener redes de gran tamaño está el problema de que se pierde claridad y es complejo brindar interacciones efectivas.

Figura 29 - Grafo que representa la red de metro de Londres.

- **Jerárquica**

Las jerarquías o árboles son colecciones de elementos en donde cada elemento tiene una conexión a un elemento padre a excepción de la raíz. Tanto los ítems como dichos enlaces pueden tener múltiples atributos definidos. Existen múltiples usos para los datos jerárquicos, como por ejemplo representar árboles genealógicos, representar la estructura de carpetas de almacenamiento en una computadora, como por ejemplo el *Windows Explorer* (figura 30).

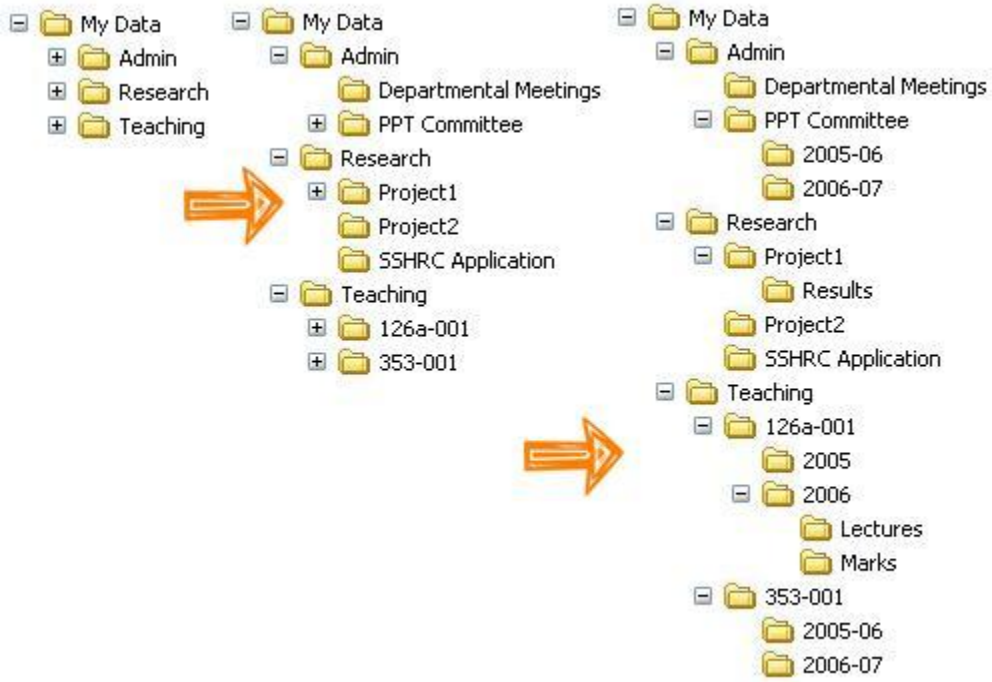


Figura 30 - Explorador de windows, permite visualizar la estructura de directorios.

Un problema típico es que a medida que la jerarquía se hace más grande es casi imposible visualizar el árbol completamente en su totalidad. Para atacar este problema la estrategia más utilizada es que aquellos nodos que no nos interesen mantenerlos colapsados, es decir ocultar las ramas que no contengan la información deseada. Si bien esta es una buena estrategia a la larga el problema de fondo volverá a aparecer, por ello una tendencia es la utilizar arboles 3D que se pueden rotar y manipular de manera de poder ver todo el árbol, "*Cone and Cam Tree*" de Xerox es un sistema de visualización que utiliza un árbol 3D (Figura 31).

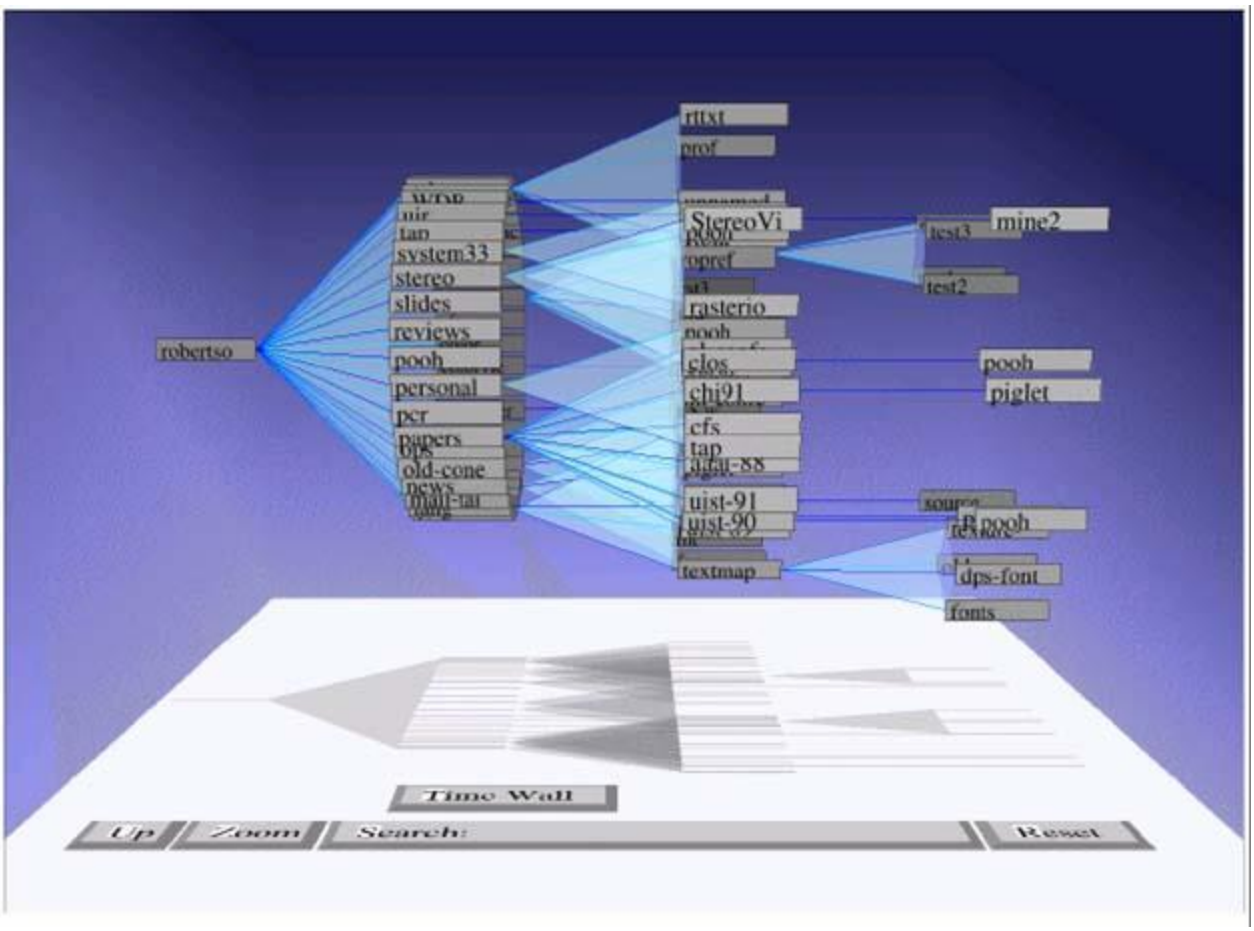


Figura 31 - Cone and Cam Tree de Xerox.

2.8 TREEMAPS

La técnica denominada *Treemaps* fue concebida por Brian Johnson y Ben Shneiderman [17], para representar información jerárquica como un conjunto de rectángulos anidados en donde cada cuadrado representa un nodo del árbol y los cuadrados interiores representan sus hijos.

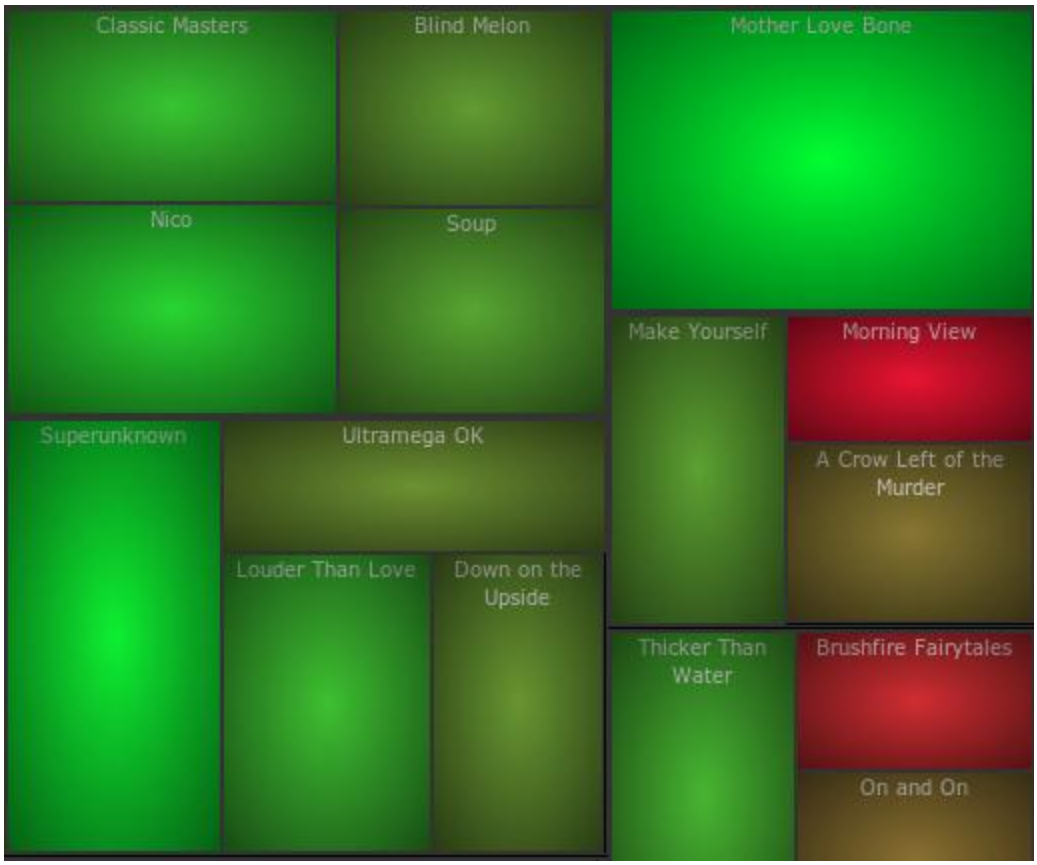


Figura 32 –*Treemap* en donde se puede observar toda la jerarquía de manera eficiente para un árbol dado.

La motivación principal para desarrollar tal técnica fue la dificultad para visualizar estructuras jerárquicas de gran tamaño como por ejemplo información referente a estructura de directorios y archivos en una computadora. Visualizar arboles de más de 1000 nodos como un árbol como vimos en la taxonomía de Shneiderman [6] es una tarea compleja y es casi imposible poder visualizar toda la jerarquía en una pantalla sin perder claridad en la visualización.

El objetivo principal del *Treemap* es visualizar grandes jerarquías sin perder claridad en la representación y explotando al máximo el espacio de la pantalla haciendo uso de cada pixel, otra particularidad de esta técnica es que es posible mapear atributos de los nodos de la jerarquía a atributos del *Treemap*, como el tamaño del rectángulo, el color, el texto que contiene entre otras cosas. De esta manera es posible que una persona pueda observar patrones en la información contenida que no sería posible si la información estuviera representada de otra manera.

Figura 33 – Newsmap , un *Treemap* que visualiza las noticias más relevantes agrupadas por tópico, y el tamaño representa el nivel de importancia de la noticia y cada color es un tópico en particular.

Existen algunas variaciones en la manera de representar *Treemaps*, una de ellas es la que usa círculos en lugar de cuadrados, estos tienen la desventaja con respecto a la técnica original de que desperdician bastante espacio de representación en la pantalla (Figura 34).

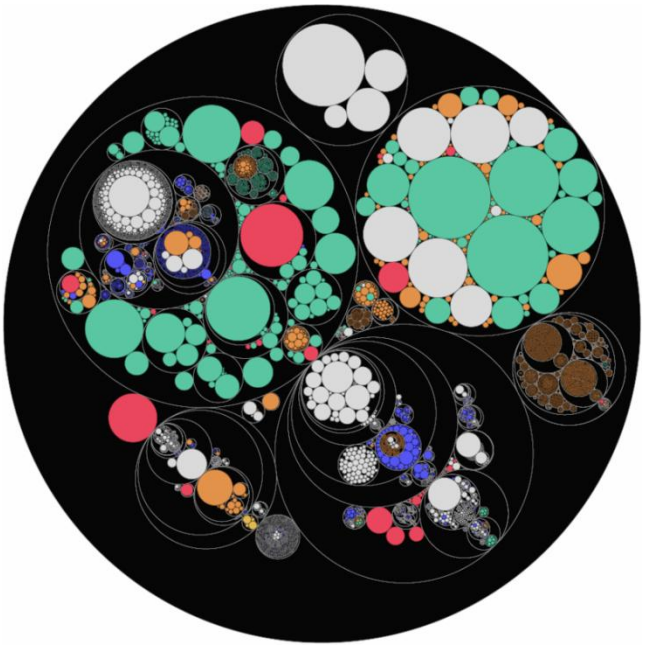


Figura 34 – *Treemap* circular.

Otra variación son los denominados *Voronoi Treemaps* [18], que en lugar de cuadrados utiliza polígonos para representar la información de una jerarquía dada y que utiliza la técnica de triangulación de Vonoroi para generar los polígonos, dicha técnica es utilizada para métricas de software (Figura 35).

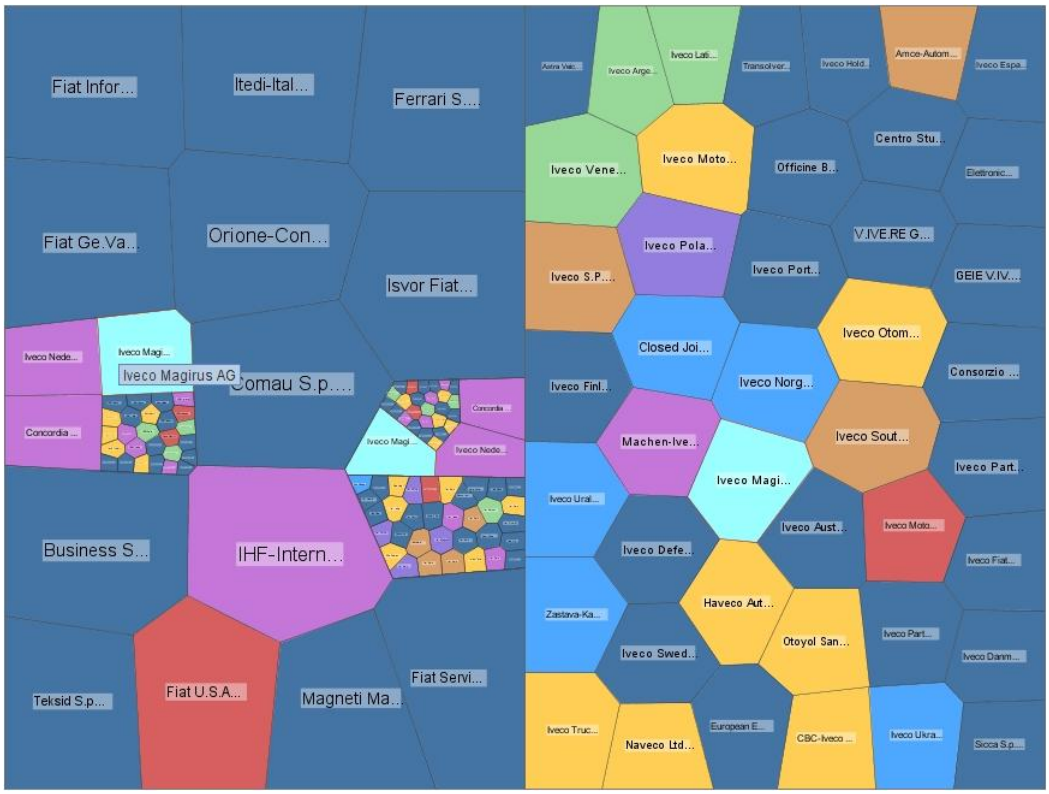


Figura 35 – Treemaps de Vonoroi.

Para crear un *Treemap*, es necesario definir un algoritmo que defina la manera en la cual un rectángulo será dividido en sub-rectángulos con áreas y atributos específicos, El método para dibujar los cuadrados del *Treemap* original sigue el esquema *preorder*, dibujando primero un cuadrado para un nodo dado y luego dibuja recursivamente los hijos de dicho nodo. El método asume que las propiedades del cuadrado como el tamaño, el color ya han sido previamente calculadas.

El objetivo ideal de un algoritmo de división es mantener la proporción de tamaño entre los cuadrados de manera que la información representada sea clara y también exista un orden entre ellos. Dado que no hay algoritmo exacto que proporcione una manera óptima de partir un cuadrado y asignar un orden a los mismos, Existen diferentes heurísticas para dividir los cuadrados que de determinan un tipo de *Treemap* particular, para el desarrollo de SQLVis se utilizó la técnica denominada *Squarified Treemaps* que proporciona buenos resultados ya que la heurística se focaliza en que la proporción del tamaño de cada cuadrado se aproxime a 1 al generar las divisiones. Esta técnica es explicada más adelante en la sección de desarrollo de la propuesta en la metáfora basada en *Treemaps*.

3 TECNOLOGÍAS ACTUALES

3.1 INTRODUCCIÓN

En esta sección se desarrollara revisión de las tecnologías existentes para la creación de la aplicación que implemente la metáfora visual, basándonos en las necesidades existentes en cuanto al acceso y manejo de los datos, renderización, interacción con el usuario, etc.

Los principales factores a tener en cuenta serán:

1. flexibilidad o su especificidad de acuerdo a nuestros objetivos;
2. velocidad;
3. usabilidad;
4. tipo de licencia;
5. facilidad de desarrollo (la curva de aprendizaje para el programador);
6. mantenimiento y documentación disponible.

Las tecnologías que se van a analizar son las siguientes:

- Processing
- Html5
- SVG + Canvas
- Protovis
- Flex & Flash
- Java infovis toolkit
- D3
- WPF

3.2 PROCESSING

Processing es un lenguaje de programación orientado a la creación de imágenes, animaciones e interacciones. Su entorno es de código abierto. Simplifica la generación de aplicaciones online. Viene con varios ejemplos para comenzar a usar. Está muy bien documentado y además cuenta con muchas librerías de imágenes, sonido y video.

El entorno de *Processing* es aún más sencillo que un compilador de Java. Incluye un pequeño entorno de desarrollo llamado "*sketchbook*".

Se pueden importar medios dentro de una librería. En *Processing* (y en Java) esta importación de medios está realizada en código, de manera similar cómo trabaja *HTML*.

En *Processing* es posible responder a un evento que se lanza con cada iteración en la rutina de sucesión de fotogramas. El usuario genera su rutina de dibujo propia para responder a dicho evento.
Processing fue escrito en Java y se puede escribir en Java los programas creados por el usuario. La popularidad de Java facilita reutilizar código programas en este lenguaje. También hay muchas librerías especialmente adaptadas, algunas de ellas están incorporadas en *Processing*.

Ventajas

- es de código abierto;
- es sencillo de usar, tiene muchas librerías y documentación y tiene tanto potencial como Java, que es uno de los lenguajes más populares actualmente;
- cuenta con librerías para realizar visualizaciones con Grafos, Treemaps y Timelines.

Desventajas

- es complicado de integrar en otras tecnologías como Microsoft .NET, y es un poco lento en la renderización.

Visualizaciones

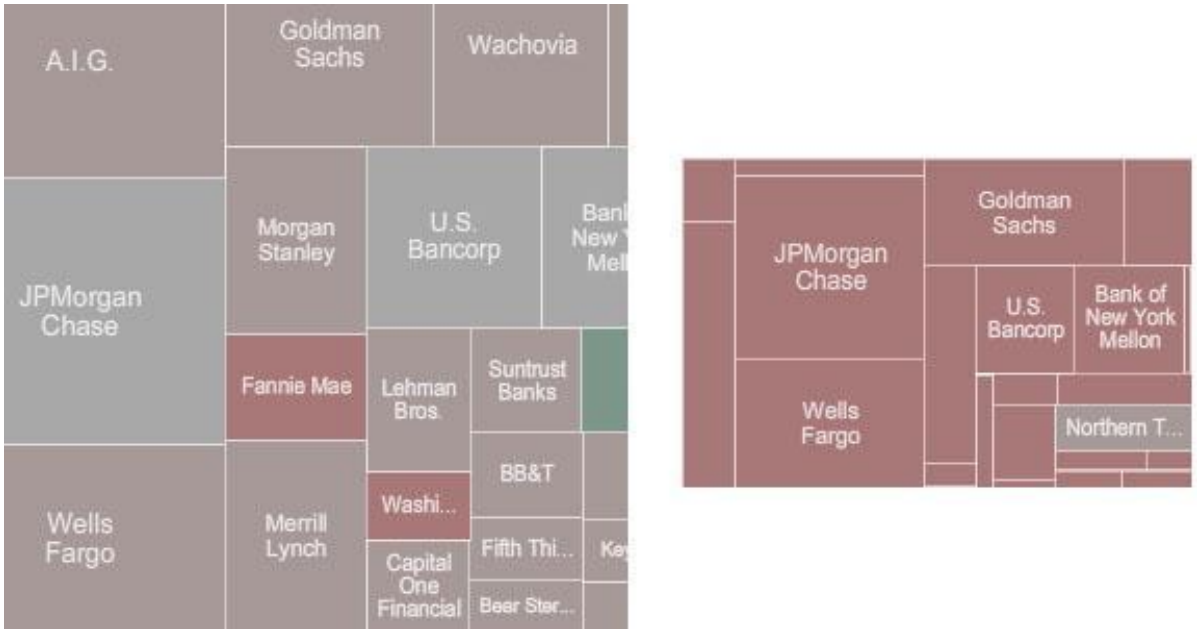


Figura 36 - Ejemplo de visualización *Treemap* en Processing.

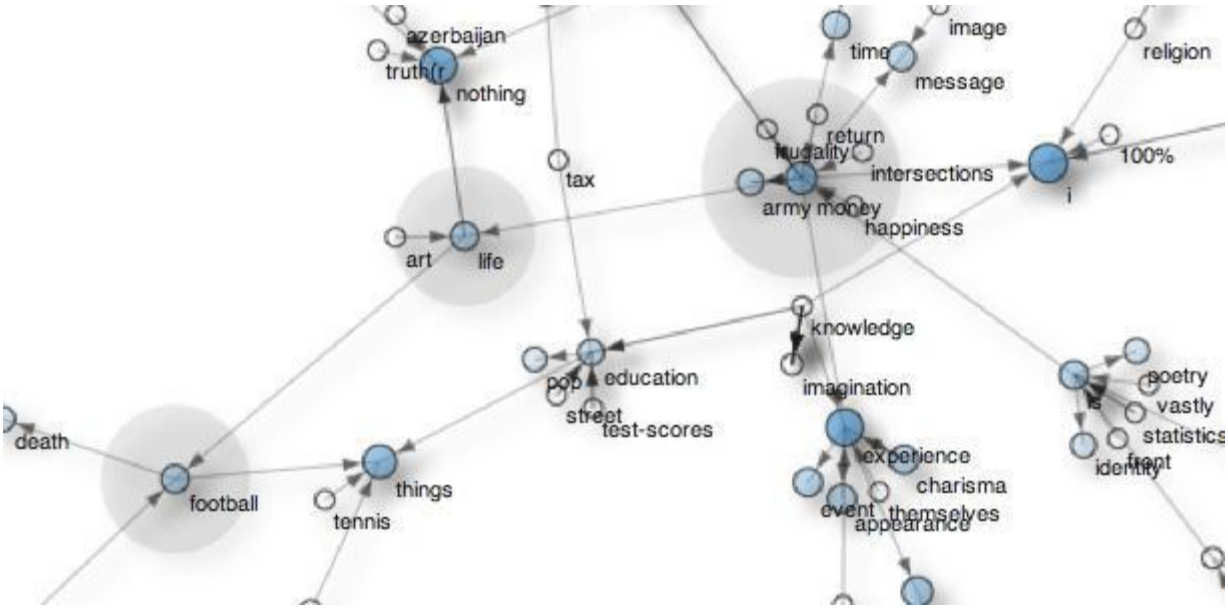


Figura 37 - Ejemplo de visualización de Grafos en Processing.

3.3 HTML5

HTML5 define una serie de nuevos elementos y atributos para proporcionar nuevas funcionalidades, como por ejemplo los elementos <audio> y <video> que permiten embeber directamente archivos multimedia como sonidos, y video así como también hacer render de elementos 3D. Su interfaz esta estandarizada de manera de hacerla compatible con la mayoría de los navegadores actualmente en uso.

El *Canvas* provee un *framebuffer* y un número de funciones de dibujado gráfico y de texto. El *canvas* sólo almacena píxeles, no objetos, esa es la diferencia principal. Cada interacción se tiene que hacer separada, pero se renderiza mas rápido, no modifica el DOM. Al ser de bajo nivel muchos browsers soportan el *canvas* con renderizado basado en hardware, haciendo más eficiente aun.

También hay nuevas etiquetas para manejar grandes conjuntos de datos, algunas de ellas son *Datagrid*, *Details*, *Menu* y *Command*. Con ellas es posible generar tablas dinámicas que se pueden ordenar, filtrar y ocultar contenido en el lado del cliente, también incorpora otras características interesantes como Geoposicionamiento, nuevas librerías de *Javascript, drag & drop* y elementos para almacenamiento.

Ventajas

- usando html5 se pueden ver visualizaciones interactivas en el navegador, escritas en *JavaScript* lo cual también lo hace rápido;
- la visualización es parte de la misma página y esta puede interactuar con otras partes sin necesidad de puentes entre ella y el contenido de la página. Aumentando así la integración;
- tiene aceleración de hardware. Corren rápido y en cualquier navegador con soporte (no hace falta instalar plugins);

- es multiplataforma, Anda bien en celulares, Aparatos móviles y es libre y gratuito, sin licencia que restrinja el uso.

Desventajas

- todavía se encuentra en modo experimental;
- no es reconocido en viejas versiones de navegadores por sus nuevas etiquetas.

Visualizaciones

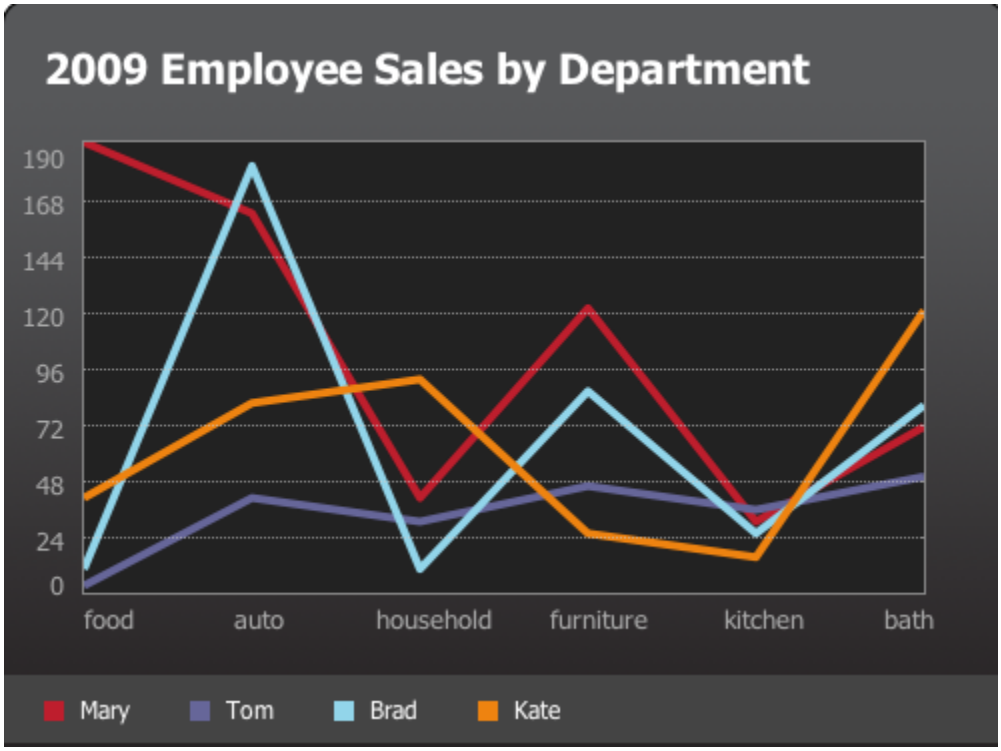


Figura 38 – Ejemplo de grafico de lineas en html5.

Figura 39 – mapa con html5.

3.4 SVG

SVG es una tecnología que permite crear objetos gráficos en el navegador que forman parte del modelo DOM. Por lo tanto pueden ser asociados a acciones cuando el usuario coloca el puntero del mouse encima de ellos o realiza una acción al presionar sobre un enlace. Al utilizar DOM puede traer algunas desventajas al hacer una actualización en los gráficos, se requiere más trabajo y tiempo, por eso puede hacer lenta una actualización grande.

Ventajas

- Los beneficios de usar gráficos vectoriales escalables más manejables que los pixeles son:
    - alta performance en dibujado 2D;
    - más rápida edición en gráficos y operaciones sobre pixeles.

Desventajas

- el mapeo de acciones a click, u otras acciones de usuario, resulta difícil;
- también hay que redibujar todo cada vez que alguien mueve o cambia algo para actualizar la visualización, Todos los elementos menos el background deberán ser redibujados.

Visualizaciones

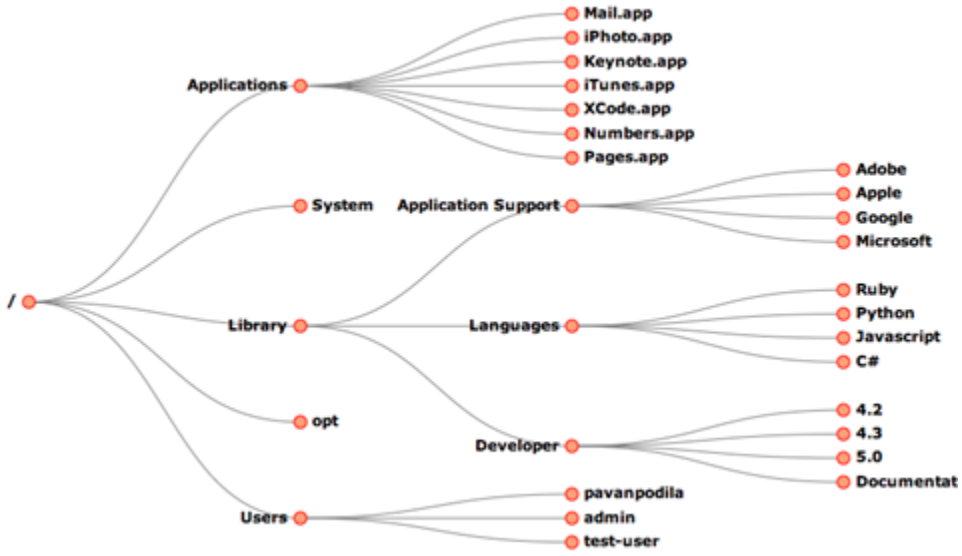


Figura 40 – Grafo jerárquico en SVG.

3.5 PROTOVIS

Protovis es un lenguaje de código abierto para desarrollar visualizaciones. Se distribuye bajo licencia BSD, internamente utiliza *javascript* y SVG para visualizaciones para la web. No es necesario tener un *plugin* en la maquina cliente para correr las visualizaciones desde una página web y es muy sencillo de aprender.

Ventajas

- posee gran versatilidad, se pueden realizar muy variadas visualizaciones. Como configurar un gráfico es relativamente sencillo;
- como funciona por capas conceptuales de la gramática, permite componer gráficos y subgráficos, aparte de facilitar la creación de nuevos elementos en cada nivel de definición del gráfico. Está muy enfocada a los gráficos estadísticos, permite pensar en su uso para visualizaciones estructuradas y basadas en los datos;
- incorpora algunas funciones estadísticas como preparación de los datos para la visualización;
- licencia BSD, prácticamente la que proporciona más libertad para su uso (tanto si el desarrollo final es también libre o propietario).

Desventajas

- ocupa bastante memoria, más bien pensada para Intranets o conexiones rápidas;
- por lo menos la primera vez la carga va a ser lenta, dependiendo del uso de cache y demás recursos;
- es recomendable para el uso específico, personalizado. Si el objetivo son gráficos sencillos es mejor optar por otras librerías más adecuadas;
- ya no posee soporte. Actualmente no está más en desarrollo

Visualizaciones

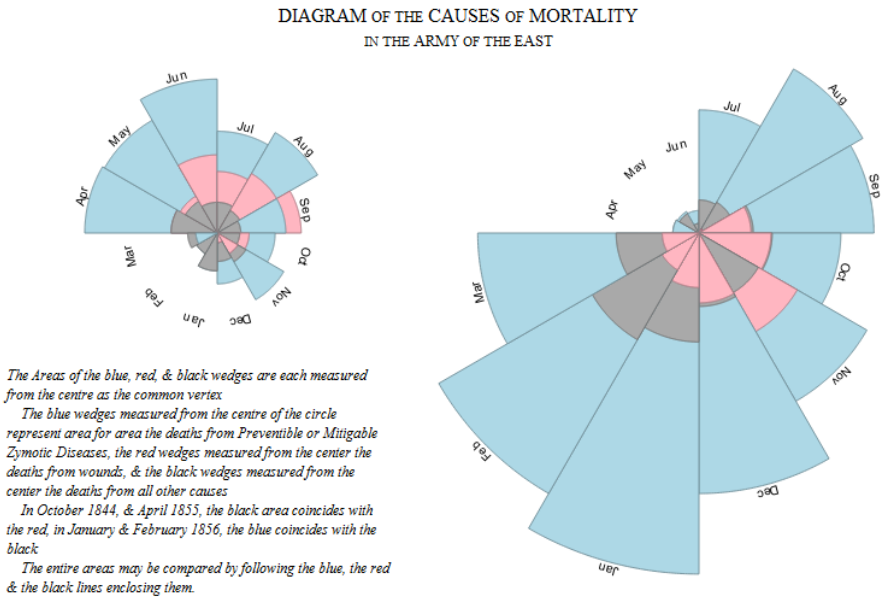


Figura 41 – Visualización usando Protovis sobre las diferentes causas de mortalidad en la armada del este entre 1844 y 1856.

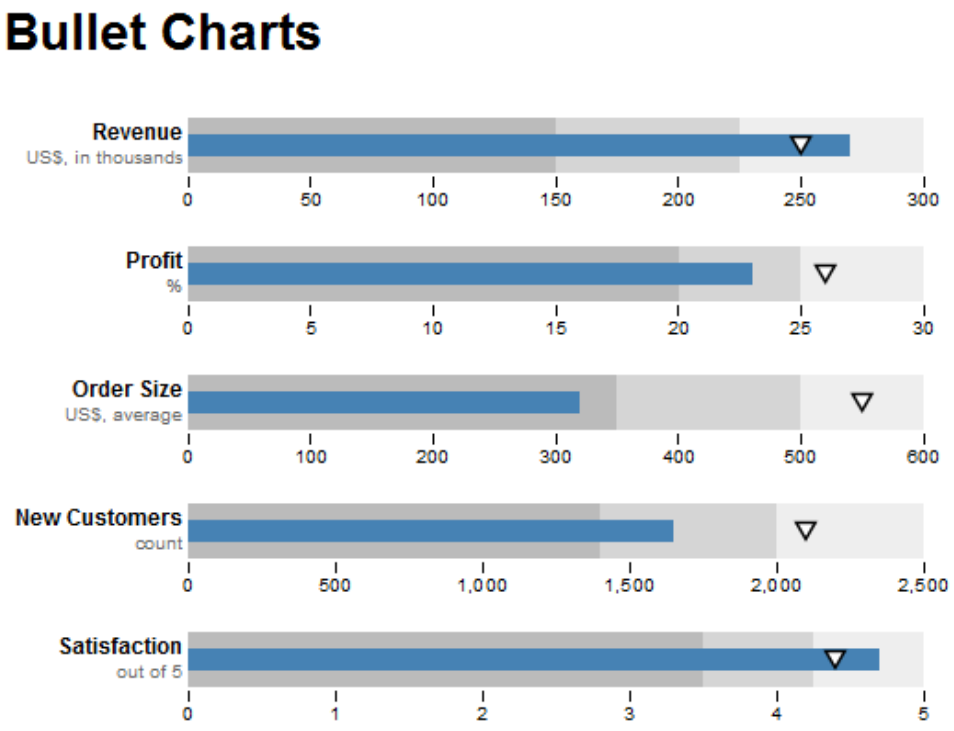


Figura 42 – Visualización con bullets charts en Protovis.

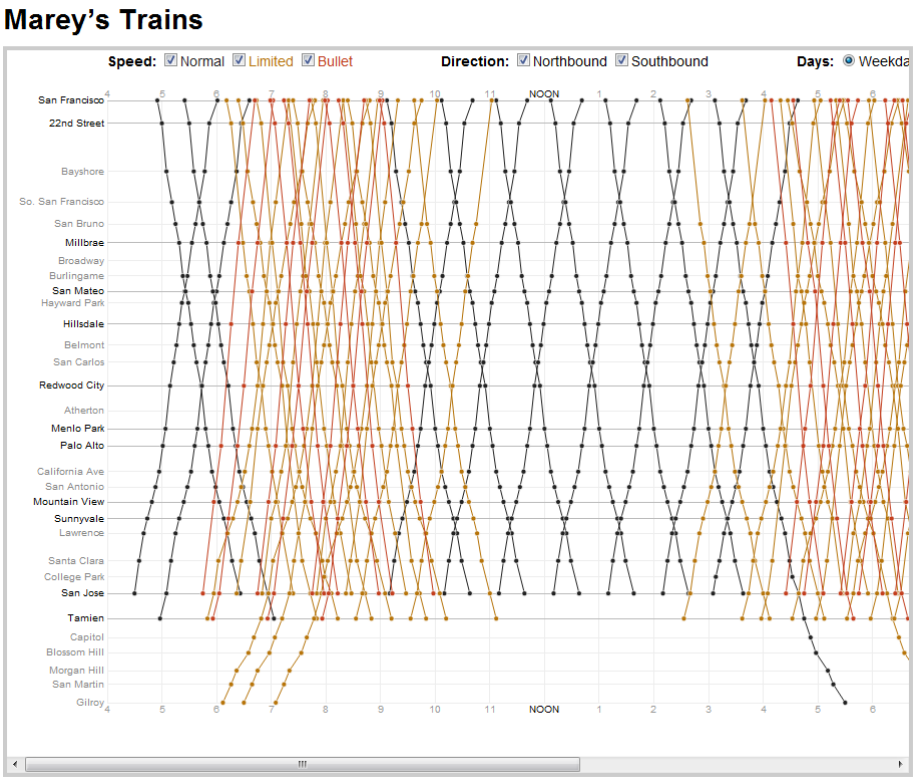


Figura 43 – Visualización de tabla de tiempos usando Protovis.

3.6 FLEX & FLASH

Adobe Flex, que hasta el 2005 se llamaba Macromedia Flex. Es una tecnología especialmente concebida para aplicaciones de internet de alto contenido multimedia. Flex en un principio fue una biblioteca de etiquetas JSP o una aplicación J2EE que compilaba el lenguaje de marcas Flex y ejecutaba mediante *ActionScript* aplicaciones Flash.

Ventajas

- posee buen soporte. Existen muchos desarrolladores familiarizados con la tecnología;
- el plugin para el navegador es comúnmente instalado;
- *ActionScript* 3 es un lenguaje maduro orientado a objetos;
- posee herramientas de soporte para editar, medir performance, testear y depurar código.

Desventajas

- la falta de performance;
- es lento para cargar y empezar a funcionar la visualización;
- hay que lidiar con temas de seguridad;
- el manejo de grandes conjuntos de datos está restringido a las limitaciones del *data handling* de Flash;
- es necesario un plugin en el navegador;
- hay que pagar la licencia para desarrollar en Flash, no es gratuita.

Visualizaciones

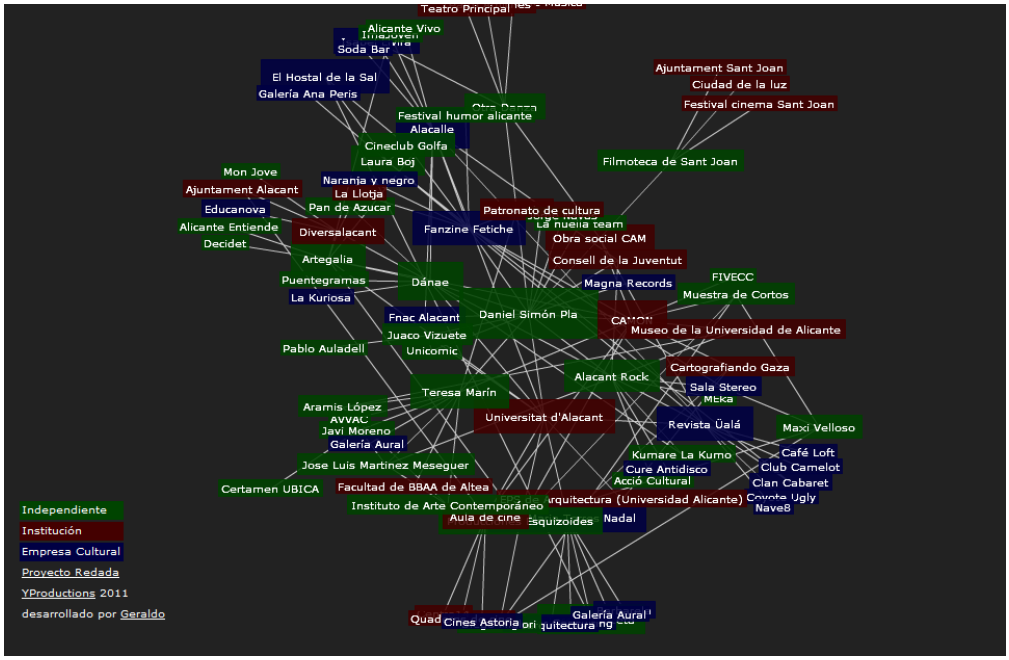


Figura 44 – Representación de un grafo en Flash.

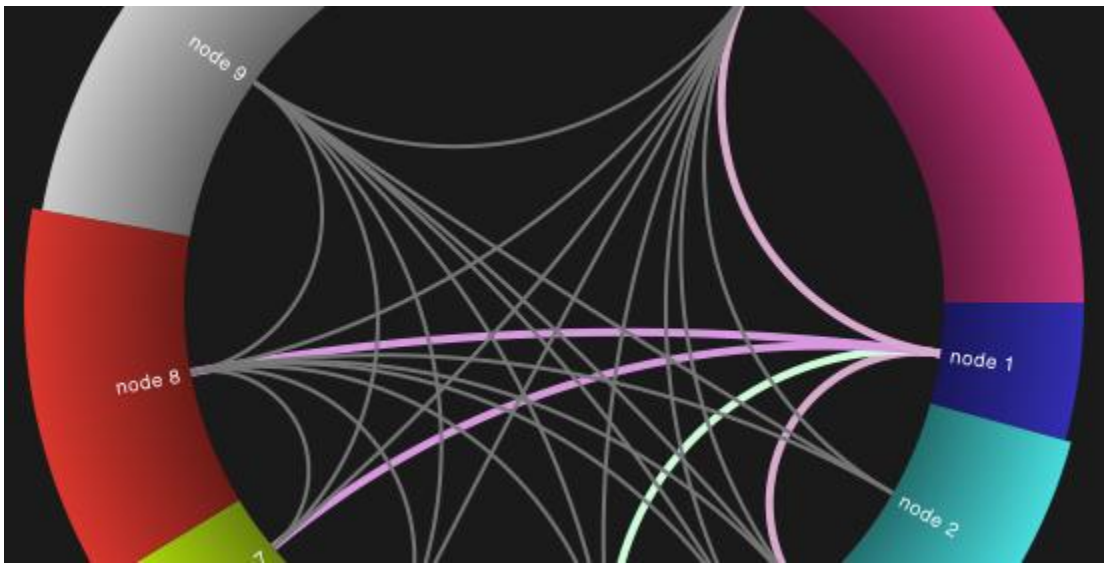


Figura 45 –visualización en Flash.

3.7 JAVA INFOVIS TOOLKIT

Java Infovis Toolkit es un set de librerías, que provee herramientas para la creación de visualizaciones interactivas, Posee una gran cantidad de gráficos, e introduce algunas animaciones, también se pueden aplicar filtros a los gráficos haciendo click en el. Toma datos entrada en JSon y salida en CANVAS. Está escrito en Java para facilitar el desarrollo de aplicaciones de visualización de la información y componentes derivados.

Ventajas

- una ventaja es también el peso de la biblioteca. JIT está mucho más pensada para el entorno web y sus archivos de javascript ocupan poco espacio;
- velocidad de procesamiento JIT, especialmente en los casos con estructuras complejas;
- incluyen animaciones e interacciones en la mayoría de gráficos de muestra.

Desventajas

- aparentemente, el punto de partida de JIT parece partir del objetivo de visualizar estructuras complejas, ya sean árboles o grafos;
- la curva de aprendizaje es grande requiere una buena experiencia en JavaScript para aprovechar el potencial de la herramienta.

Visualizaciones

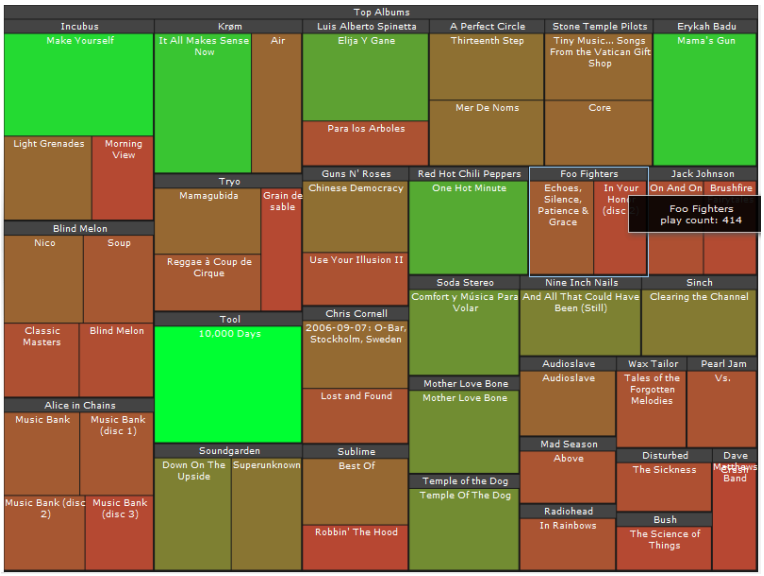


Figura 46 – Visualización de un *Treemap* en Java Infovis toolkit.

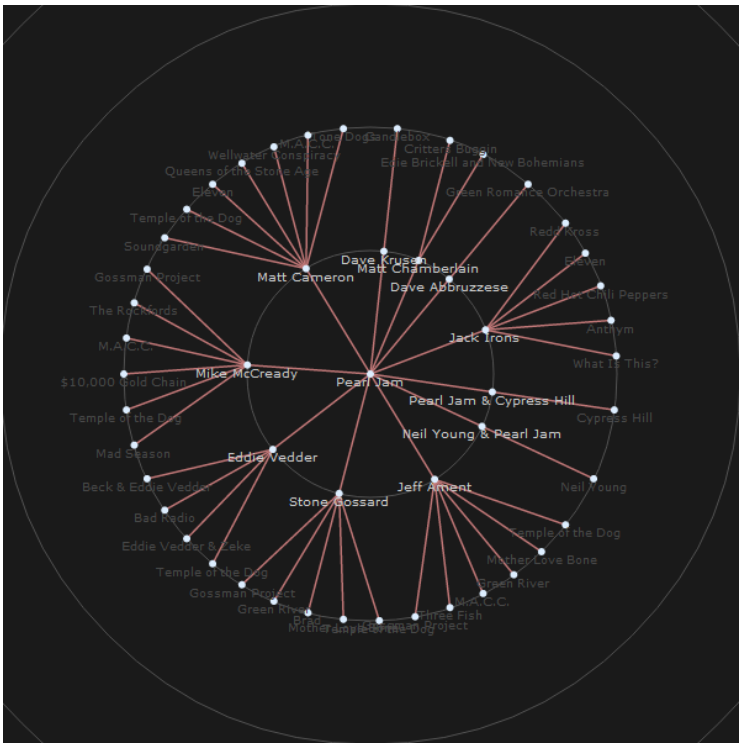


Figura 47 – Representación de un grafo en Java Infovis toolkit.

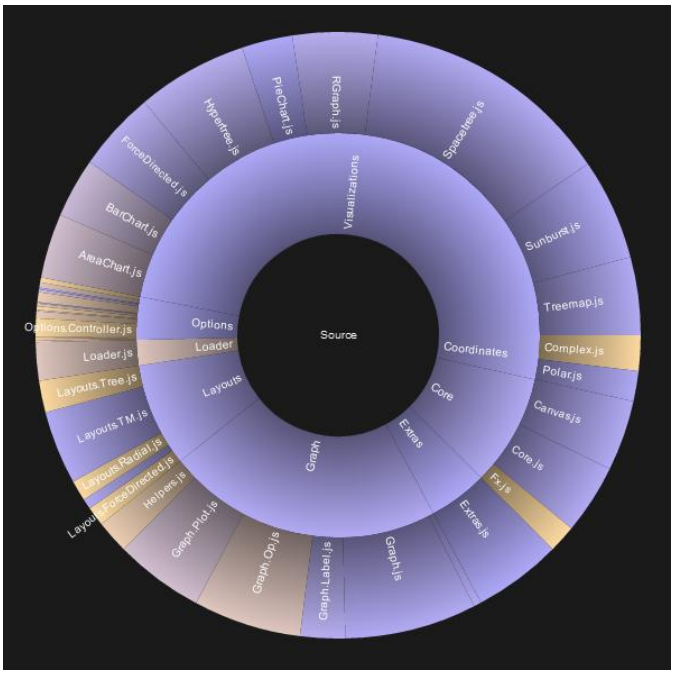


Figura 48 – Representación del sistema de archivos de una computadora en Java Infovis toolkit.

3.8 D3

D3 es una librería hace uso de estándares CSS3, HTML5 y SVG no implementa nuevas formas de representación gráfica. Es una librería de visualizaciones en *Javascript* relativamente nueva. Se puede usar directamente la consola del navegador para depurar comandos D3.

Es más bien un *framework* para el desarrollo de visualizaciones basadas en datos que un *toolkit* de gráficos,
D3 describe las transformaciones de las escenas mientras que por ejemplo Protovis describe las representaciones de las mismas, por otro lado proporciona mayor control sobre que queremos cambiar y también genera menos costo al no ser necesario cambiar todo.

Es una excelente herramienta para las personas que crean nuevas visualizaciones de datos y visualizaciones construidas con conjuntos de datos de gran tamaño.

Ventajas

- es muy liviana, flexible y *open source*.
- genera unos gráficos muy buenos y sofisticados.
- permite enlazar datos al DOM (Document Object Model) y manipular el documento como se requiera para generar la visualización.
- tiene mejor manejo y performance para grandes conjuntos de datos.
- permite obtener datos de un documento o mediante JSON y representarlos en una forma grafica.

Desventajas

- la curva de aprendizaje es alta se debe tener conocimientos de *Javascript*, *JSON* y matemáticos. Puede ser más sencillo si se está familiarizado con *JQuery* o *Prototype*;
- sólo sirve para aplicaciones web, no se puede utilizar en aplicaciones móviles y de escritorio.

Visualizaciones

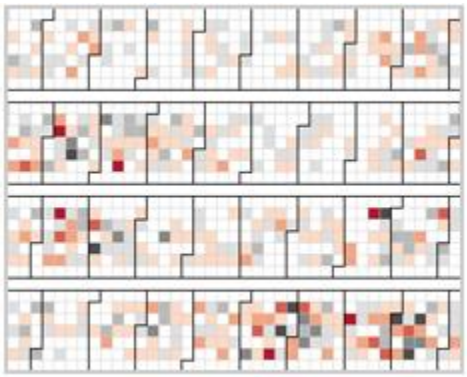


Figura 49 – Visualización de un calendario usando D3.

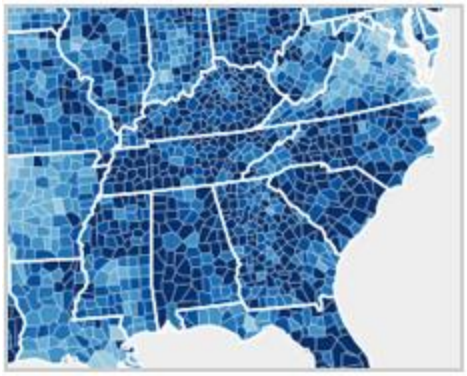


Figura 50 – Visualización usando D3 correspondiente a un Mapa de EEUU.

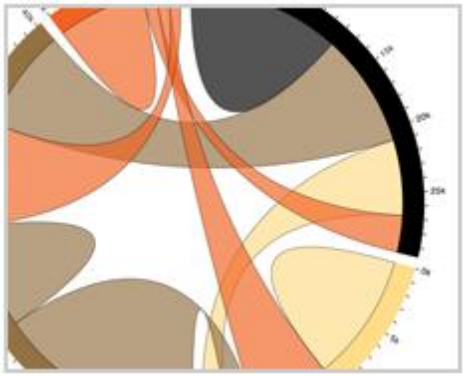


Figura 51 – Visualización usando D3.

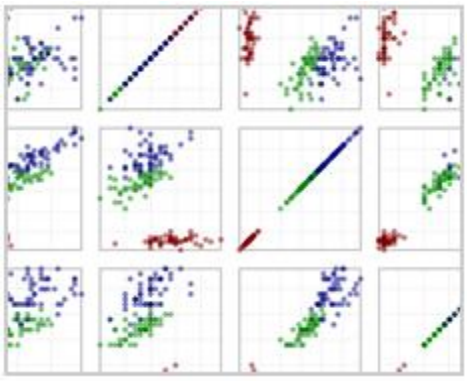


Figura 52– Small múltiples en D3.

3.9 WPF

WPF es un subsistema de presentación unificado para Windows. Permite crear aplicaciones cliente de Windows que proporcionen una experiencia impactante para el usuario desde el punto de vista visual, WPF da soporte a diversos tipos de aplicaciones. Separa la parte de interfaz con la de código de la aplicación.

Permite crear interfaces eficientes y sumamente atractivas que incorporen componentes multimedia, documentos, gráficos bidimensionales y tridimensionales, características tipo web y animaciones.

Ventajas

- permite a diseñadores y programadores juntos crear experiencias de usuario únicas y visualmente sorprendentes;
- estilo potente y estructurado;
- facilidad para crear estilos y aspectos;
- tiene capacidad de reutilización del código existente;
- permite enlazar datos con cualquier control;
- programación declarativa;
- hacer uso del potencial de las tecnologías .net.

Desventajas

- para aprovechar el potencial de WPF hace falta el trabajo de diseñadores gráficos también;
- modificar código en XAML resulta complejo en muchas ocasiones;
- se requiere tener instalado el .NET *framework*;
- se requiere de pc con potencial gráfico, deben soportar DirectX y tener una tarjeta gráfica con suficiente capacidad, aunque esto es común hoy en día;
- la curva de aprendizaje es bastante alta.

Visualizaciones

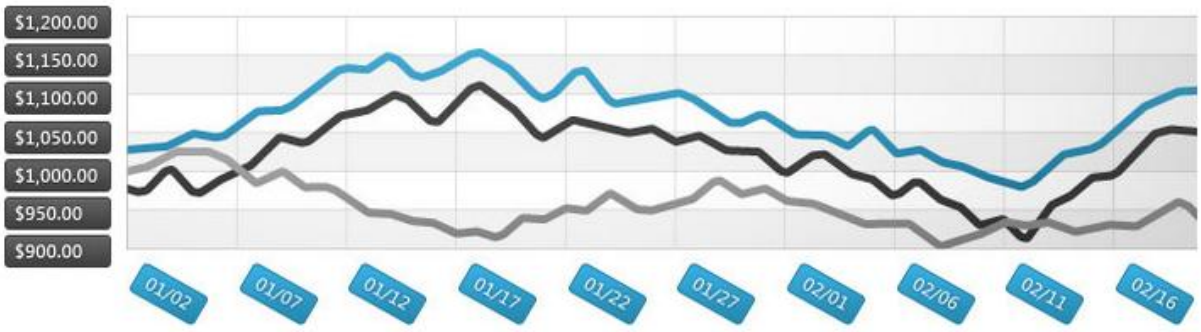


Figura 53 – Modularidad de ejes y series de gráfico, usando WPF.

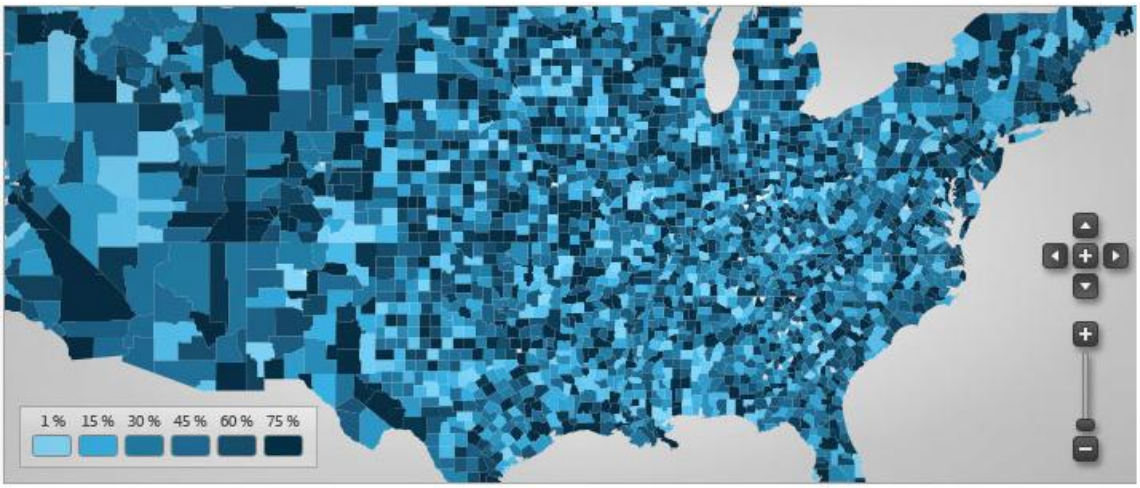


Figura 54 – Mapa de EEUU con saturación de colores usando WPF.

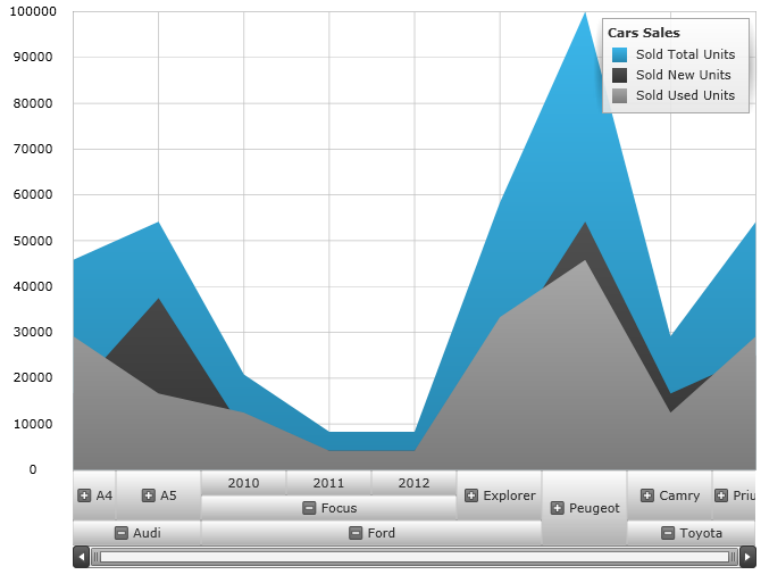


Figura 55– Gráfico de áreas usando WPF.

3.10 CONCLUSIONES

Como etapa final del estudio de las tecnologías existente más populares, haremos un análisis y comparación de cada una ellas tomando en cuenta los factores que mencionamos al principio que son los siguientes:

- **Flexibilidad**: Que tan fácil es modificar o adaptar la tecnología a nuestras necesidades.
- **Velocidad**: Que tan rápida es para renderizar y cargar grandes volúmenes de información.
- **Usabilidad**: Que tan intuitivas son las visualizaciones para un usuario, que herramientas de interacción cuentan las visualizaciones.
- **Facilidad**: Cuan fácil es aprender a usar dicha tecnología, configurar el entorno de desarrollo, adaptar las visualizaciones a nuestras necesidades puntuales.
- **Documentación:** Cuanta documentación disponible hay, foros, ejemplos, guías, soporte disponible de la tecnología.

Usando la siguiente escala de valores:

PESIMO	0,1,2,3,4,5,6,7,8,9,10	EXCELENTE

Definimos la siguiente tabla para ponderar las características más destacables de las tecnologías.

Tecnología	Flexibilidad	Velocidad	Usabilidad	Facilidad	Documentación
Processing	6	4	7	5	7
Html5	4	5	5	3	7
SVG + Canvas	4	7	6	5	6
Protovis	4	5	7	5	8
Flex & Flash	8	2	7	5	7
Java Infovis Toolkit	4	5	8	4	9
D3	6	6	7	4	4
WPF	7	7	8	10	9

Estos resultados son a modo resumen, Podemos concluir que en lo que es Flexibilidad, usabilidad y documentación disponible la mayoría de las tecnologías resulto buena. A la hora de elegir una tecnología pusimos mucho peso en la facilidad de aprendizaje y flexibilidad para lograr desarrollar la metáfora visual planteada y que el tiempo de desarrollo no se diluya demasiado en aprender y configurar las herramientas, Es por eso que la opción más óptima resultó ser WPF, pues aparte de ser flexible ya teníamos experiencia en dicha tecnología por lo que la curva de aprendizaje y configuración del entorno seria mucho menor en comparación con todas las otras, sumado a que es posible obtener muy buenos resultados gráficos.

Esta misma información la podemos ver visualizar mediante un *gráfico de estrella* [20] de la siguiente manera:

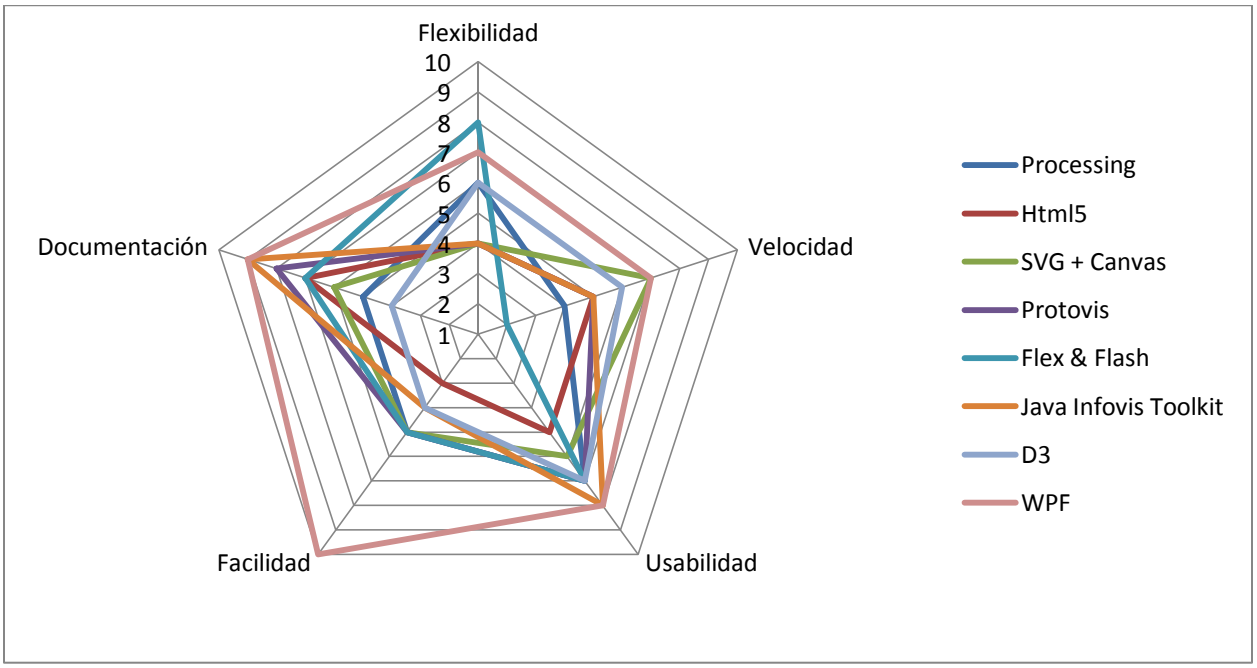


Figura 56 – Representación gráfica de la tabla anterior usando un gráfico de estrella.

Del grafico podemos observar que WPF se destaca por encima de las demás tecnologías, en parte porque como mencionamos antes, nuestro criterio fue buscar una tecnología que cuya curva de aprendizaje sea lo más baja posible y por otro lado provea buenos niveles de usabilidad y documentación.

4 DESARROLLO DE LA PROPUESTA

4.1 INTRODUCCIÓN

Un sistema de gestión de base de datos posee distintos módulos funcionales. Estos pueden categorizarse a grandes rasgos en módulos de gestión de almacenamiento y de procesamiento de consultas.

Las bases de datos corporativas pueden alcanzar tamaños de cientos de gigabytes. Esas cantidades de datos no entrarían en la memoria principal de la máquina, por eso se almacenan en disco. El acceso a disco es lento por eso hace falta estructurar bien los datos para minimizar la transferencia entre disco y memoria. El procesador de consultas abstrae de detalles físicos permitiendo acceder a los datos a través de un lenguaje de más alto nivel. En bases de datos relacionales el lenguaje de consultas más usado es el SQL para obtener la información desde la base de datos. Las consultas pueden llegar a ser muy complejas y su diseño tomar un tiempo considerable. Puede ser que la respuesta obtenida no siempre sea óptima.

El gestor de almacenamiento también se encarga de reservar el espacio en disco y el almacenamiento en memoria principal y que datos tratar en cache. También el manejo de índices que proporcionan más rápido acceso a elementos particulares. El procesador de consultas posee un intérprete para las instrucciones, un compilador que traduce en instrucciones de más bajo nivel y un plan de evaluación y el motor de evaluación que ejecuta esas instrucciones generadas por el compilador.

La optimización de consultas es un proceso de refinamiento de las mismas en el cual se espera mejorar los tiempos de respuesta del sistema. Mejorando su eficiencia y el uso de recursos disponibles.

El proceso de optimizar consultas puede llegar a ser costoso en esfuerzo. Por eso un buen punto de partida es saber cuáles son las consultas que más penalizan la performance de mi sistema. Una funcionalidad que brindan algunos sistemas de bases de datos es la posibilidad de generar un log con las consultas que sobrepasan un tiempo determinado en su ejecución. El problema que surge es que muchas veces estos logs poseen mucha información pero mal presentada o escondida entre los datos en crudo. Y aquí se nos presenta la posibilidad de refinar estos datos, y presentarlos en una visualización adecuada que nos permita deducir los puntos conflictivos y así introducir las mejoras necesarias.

4.2 DESAFÍOS ACTUALES

En la actualidad hay pocas herramientas o visualizadores orientados a este dominio que permitan visualizar los logs propiamente dicho con el objetivo de inferir y responder preguntas como las siguientes:

- ¿Qué tipo de consultas son las que penalizan la performance de mi sistema?
- ¿Cuál es el conjunto que más penaliza la performance?
- ¿Hay muchas similares?

Las herramientas existentes para el caso de MySQL (hay tanto *open source* como comerciales) en general son herramientas orientadas al ajuste de bases de datos. Si bien muchas cuentan con la opción de analizar el log de consultas lentas, pocas permiten visualizar dicho log de tal manera que se pueda ver la información de forma clara.

Una de las herramientas actuales de análisis de log más populares es *mk-query-digest* que básicamente permite detectar consultas lentas y proponer mejoras. Dicha herramienta utiliza las API de MySql para obtener la información y analizar el log. No es una herramienta visual y sólo se puede usar a través de un *prompt* de linux o DOS. Otra similar es *MyProfi* cuya imagen se muestra a continuación:

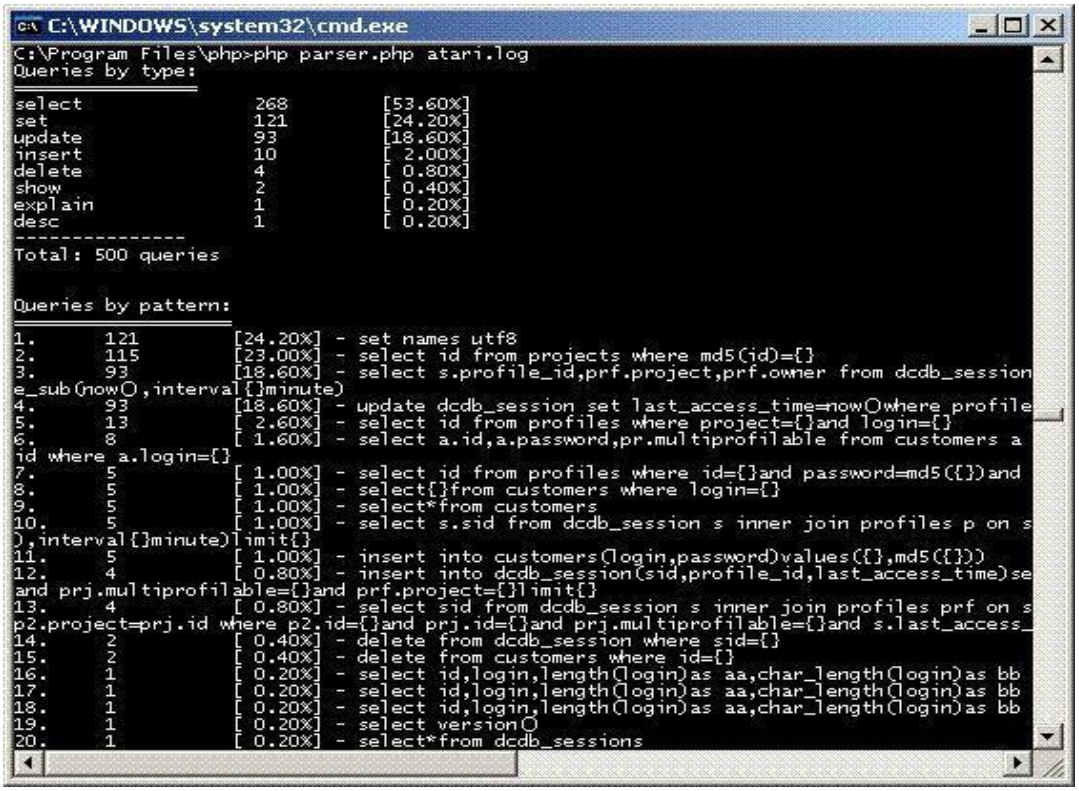


Figura 57 - Consola con el output de la herramienta MyProfi que permite analizar el log y agrupar por patrones.

MyProfi permite agrupar las consultas por patrón, es decir agrupar todas aquellas que son similares estructuralmente, Luego también tenemos *mysql-slow-query-log-visualizer*, esta herramienta es visual y permite ver en un gráfico la cantidad de consultas lentas que son logueadas en función del tiempo y también permite listarlas y permitir interacción como filtrado y ordenamiento.

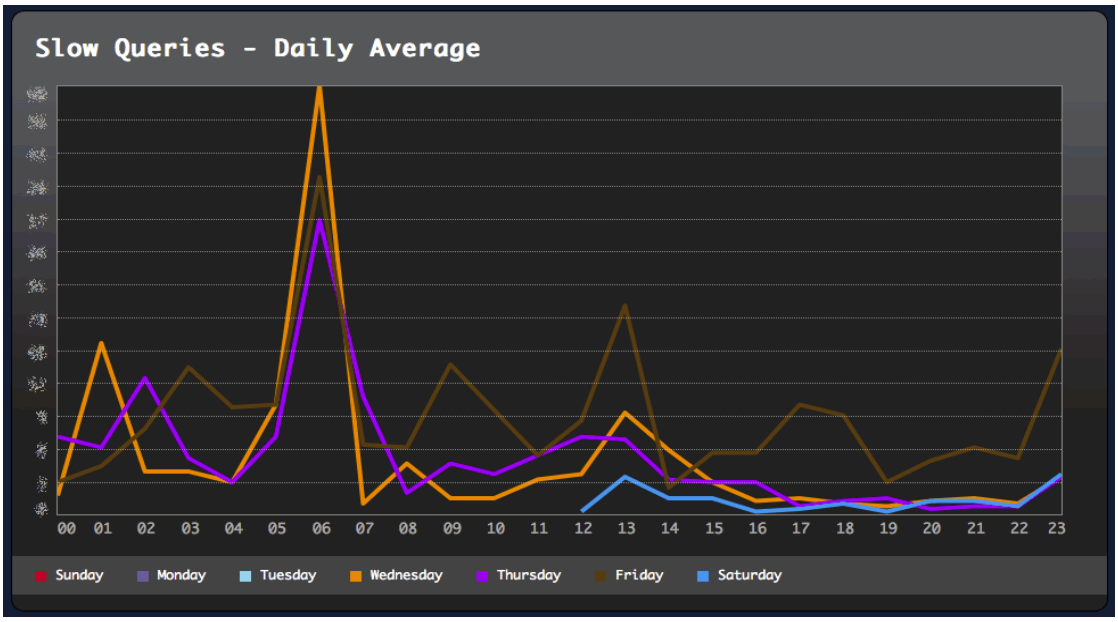


Figura 58 - Gráfico de consultas lentas de mysql-slow-query-log-visualizer, que permite ver en función del tiempo las ocurrencias.

4.3 PROCESO DE VISUALIZACIÓN

En esta sección describiremos cómo abordamos el desarrollo del proyecto cubriendo desde la obtención de los datos brutos que van a ser input para nuestra herramienta hasta el refinamiento para brindar más interacción al usuario en la visualización. A la hora de utilizar una metodología para desarrollo optamos por utilizar el pipeline de visualización definido por Ben Fry en [2].

4.4 INTRODUCCIÓN

Cada conjunto de datos para un propósito en particular tiene una necesidad de visualizarse de una determinada manera. Existen innumerables herramientas para visualizar información pero no es nada trivial realizar visualizaciones complejas orientadas a un cierto dominio.

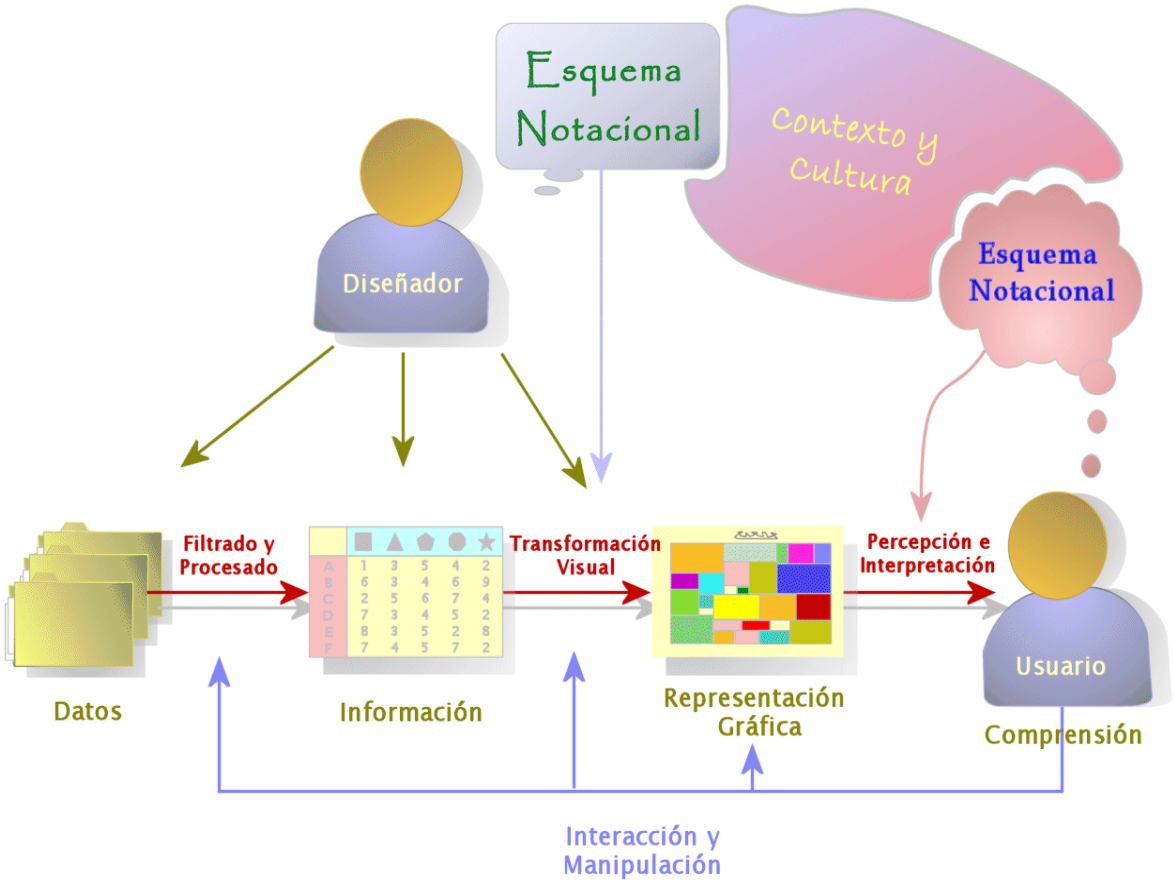


Figura 59 - Esquema general del proceso de desarrollo de una visualización.

Por ejemplo hacer una correspondencia de datos a una forma visual no resuelve el problema de cómo manejar fuentes de información de considerable tamaño. La disciplina de la minería de datos tiene como objetivo el manejo de grandes volúmenes de información, pero nada dice sobre cómo se debe interactuar con dichos datos.

Sin embargo es posible unir a todas ellas para generar un proceso a partir del cual lograr un marco por el cual podamos generar una visualización partiendo desde los datos extraídos en bruto hasta la interacción con el usuario en la visualización. En otras palabras, reconciliar dichas disciplinas en un solo proceso. Tal proceso une todas las disciplinas antes mencionadas poniendo el foco y consideración en cómo dicha información es interpretada. El proceso está dado por los pasos descritos en la figura 59, que nos conducirán a la respuesta o nos proveerán los mecanismos suficientes para inferir lo que esperamos.

Al comienzo de un proyecto de visualización es muy común enfocarse en toda la información en bruto que debe ser recolectada la cual es una cantidad significativa. Generalmente se recolecta información para inferir algo sobre la misma o bien responder preguntas concretas sobre un determinado dominio. Cuanto más específicas son estas preguntas que queremos obtener respuesta, más específica y clara será la visualización resultante.

Ben Fry propuso un pipeline de visualización [2] cuyo principal propósito es organizar la manera en la cual se debería implementar una visualización dada, dividiendo el proceso en etapas, partiendo de los datos crudos hasta el

refinamiento de la visualización agregando interacción a la misma. Cada una de estas disciplinas que conforman el pipeline ha evolucionado por separado por lo que es de esperar que tengan poco y nada en común.

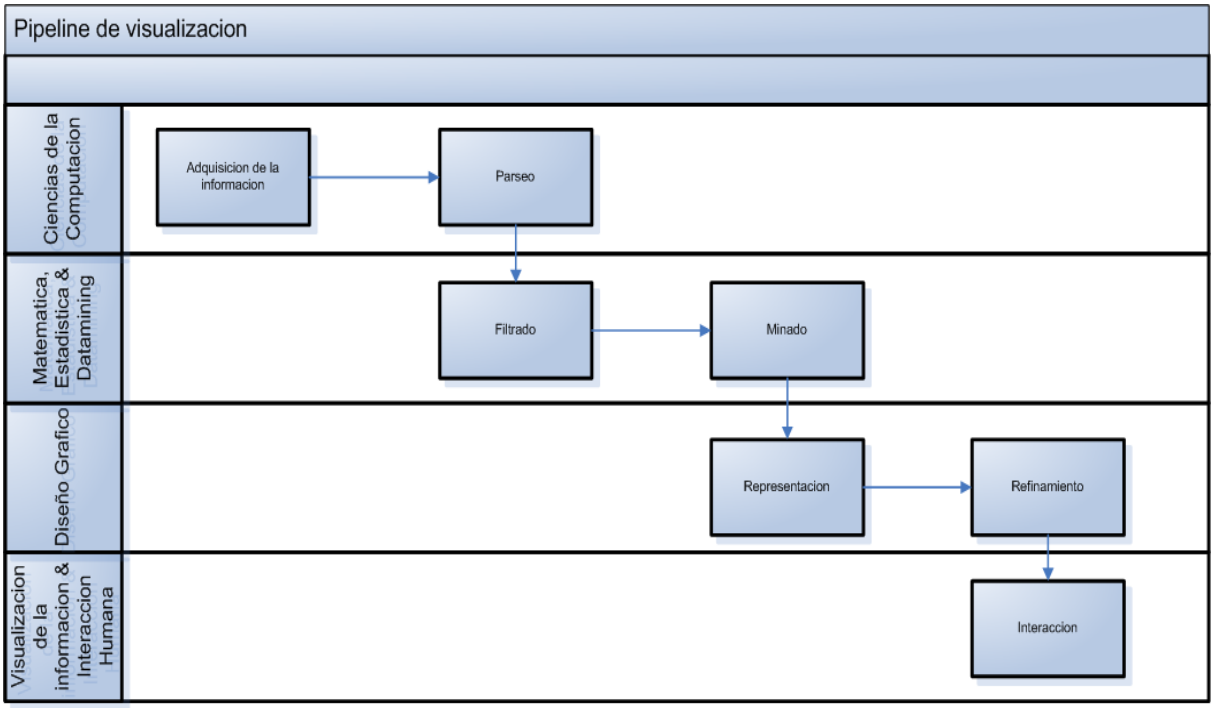


Figura 60 - Pipeline de visualización de Ben Fry.

La descripción de cada una de las etapas es la siguiente:

Adquisición de la información: Obtener la información que es necesaria, desde cualquier ubicación ya sea en disco, archivo o Internet.

Parseo: en esta etapa la idea es dar una estructura para la información obtenida en el paso anterior de manera tal que sea más sencillo manipularla y categorizarla.

Filtrado: remover toda información que sea inútil o no relevante al dominio del problema en cuestión para así optimizar el tamaño de la muestra.

Minado: aplicar métodos de estadística o minería de datos para poner la información en un contexto matemático.

Representación – Metáfora visual: la idea es elegir un modelo visual acorde para representar la información. Ejemplos pueden ser un gráfico de barras, una lista, un *treemap*, etc.

Refinamiento: refinar la visualización para hacerla más clara y concisa, hacer que sea visualmente atrayente e interesante al usuario.

Interacción: agregar métodos para manipular la información o controlar que atributos son visibles.

Esta más que claro que a la hora de encarar un proyecto, no hace falta implementar todos las etapas definidas del pipeline, solo debemos tomar las que consideremos necesarias y útiles para nuestro proyecto.

En las siguientes secciones describimos cómo implementamos las etapas del pipeline de Ben Fry, en nuestro caso no era necesario hacer minado pues la información que vamos a procesar no necesita de un contexto matemático y tampoco realizamos ningún filtro pues necesitamos procesar toda la muestra de datos indistintamente.

4.5 FUENTE DE INFORMACIÓN

La primera etapa del pipeline de Ben Fry es conseguir la fuente de datos que será el input de todo el proceso de transformación hasta llegar a la visualización. El input propiamente dicho es el archivo de Slow Query log que básicamente es un archivo de texto plano que contiene entradas como las siguientes:

UPDATE jforum_topics_watch SET is_read = 1 WHERE topic_id = 2577 AND user_id = 8181;
Time: 050718 19:45:49
User@Host: db[db] @ localhost [127.0.0.1]
 # Query_time: 25 Lock_time: 0 Rows_sent: 1673 Rows_examined: 3076476

SELECT post_id FROM jforum_search_wordmatch wm, jforum_search_words w WHERE wm.word_id = w.word_id AND LOWER(w.word) = LOWER('after');
Time: 050718 19:45:58
User@Host: db[db] @ localhost [127.0.0.1]
 # Query_time: 3 Lock_time: 0 Rows_sent: 43 Rows_examined: 10310

SELECT COUNT(pm.privmsgs_to_userid) AS private_messages, u.* FROM jforum_users u LEFT JOIN jforum_privmsgs pm ON pm.privmsgs_type = 1 AND pm.privmsgs_to_userid = u.user_id WHERE u.user_id = 7421 GROUP BY pm.privmsgs_to_userid;
Time: 050718 19:46:03
User@Host: db[db] @ localhost [127.0.0.1]
 # Query_time: 25 Lock_time: 0 Rows_sent: 0 Rows_examined: 3076542

SELECT post_id FROM jforum_search_wordmatch wm, jforum_search_words w WHERE wm.word_id = **w.word_id AND LOWER(w.word) = LOWER('gurl/guy');**
Time: 050718 19:46:04
User@Host: db[db] @ localhost [127.0.0.1]
 # Query_time: 23 Lock_time: 0 Rows_sent: 1673 Rows_examined: 3076476

SELECT post_id FROM jforum_search_wordmatch wm, jforum_search_words w WHERE wm.word_id = w.word_id AND LOWER(w.word) = LOWER('after');
Time: 050718 19:46:12
User@Host: db[db] @ localhost [127.0.0.1]
 # Query_time: 23 Lock_time: 0 Rows_sent: 0 Rows_examined: 3076476

SELECT post_id FROM jforum_search_wordmatch wm, jforum_search_words w WHERE wm.word_id = w.word_id AND LOWER(w.word) = LOWER('a');
Time: 050718 19:46:15
User@Host: db[db] @ localhost [127.0.0.1]
 # Query_time: 3 Lock_time: 0 Rows_sent: 0 Rows_examined: 0

Este archivo de log se genera cuando se habilita en los settings de MySQL. Cuando se habilita, se establece un umbral mínimo de tiempo que es considerado el límite entre lo lento y lo normal, es decir que cuando el tiempo de ejecución de una consulta sea mayor o igual a dicho valor, MySQL creará un registro en el LOG con la consulta junto con otra información de contexto como el host que ejecuto la consulta, el usuario, hora y fecha.

Esta etapa fue complicada, ya que conseguir buenas fuentes de información con datos ricos y de considerable tamaño no fue fácil. Buscamos por Internet y no obtuvimos buenas fuentes, y también pedimos ayuda al departamento de computación para obtener logs reales, ya que conseguir de empresas privadas era muy complicado, por temas de seguridad y confidencialidad de la información.

Ante este dilema optamos en un principio por construir un generador de Log para tener más flexibilidad con las pruebas y verificar que efectivamente el funcionamiento de la herramienta era correcto. Sobre el final del desarrollo pudimos conseguir muestras reales provistas por los administradores del centro de cómputos del DC para poder experimentar con una muestra de log real a la que tuvimos que ofuscar cierta información considerada confidencial.

4.6 PARSING

En esta sección vamos a describir cómo realizamos el proceso de parseo y luego como aplicamos el patrón de similaridad para agrupar consultas similares.

4.6.1.1 Introducción

Para poder manipular la información extraída fue necesario desarrollar una estructura de datos para dar soporte a la información y así poder visualizar dicha información agrupada de alguna manera conveniente. Como agrupamos consultas por similaridad agregamos campos especiales para poder agrupar las consultas y también agregamos otros campos para calcular el tiempo promedio de cada grupo de consultas. Luego implementamos un parser para poder cargar los datos en la estructura definida. Para parsear el código SQL contenido en cada registro utilizamos una librería llamada *gudusoft.gsqlparser* que permite partir la consulta en sus diferentes partes (Select, Where, From,etc) Esto resulta útil para desarrollar el criterio de agrupamiento que se detalla a continuación.

Como parte del proceso del parseo de datos es necesario realizar un procedimiento que determine qué consultas son similares entre unas y otras para poder agruparlas. Esto permite tener una mejor visión del problema ya que es muy frecuente que una misma consulta aparezca una y otra vez en el slow query log y también es común que haya leves diferencias entre una consulta y otra.

Esto también es muy útil para determinar el grupo de consultas que más penaliza la performance del motor de base de datos, ya que el conjunto de mayor número cardinal va ser aquel que tenga más consultas en dicho grupo por ende dichas consultas podrán ser las más frecuentemente ejecutadas por el DBMS.

4.6.1.2 Patron de agrupamiento

Para agrupar las consultas definimos una noción de similaridad entre 2 consultas dadas. En un principio pensamos utilizar la noción de similaridad entre cadenas de texto denominada distancia de Levenshtein [13], pero por temas de performance preferimos simplificar un poco el proceso de agrupar y definir nuestra propia noción de similaridad que es como sigue:

2 consultas dadas son similares si y sólo si ocurre lo siguiente:

- son exactamente iguales o
- difieren en lo sumo n campos/expresiones internamente pero su estructura es igual.

El análisis de similaridad es estructural, es decir dadas 2 consultas, y la primera es como la siguiente:

SELECT (Campo_1,Campo_2,.....,Campo_n) **FROM** (Table_1,.....,Table_n)
WHERE (Exp)
ORDER BY (Exp)

Y la segunda es como sigue:

SELECT (Campo_1,Campo_2,.....,Campo_n-3) **FROM** (Table_1,.....,Table_n)
WHERE (Exp)
ORDER BY (Exp)

Podemos concluir que entran dentro del mismo grupo, porque por más que difieran en a lo sumo k campos con k < n, tienen la misma estructura exterior (SELECT,FROM,ORDER BY).

En cambio sí tenemos lo siguiente:

SELECT (Campo_1,Campo_2,.....,Campo_n) **FROM** (Table_1,.....,Table_n)
WHERE (Exp)
ORDER BY (Exp)

Y la segunda es como sigue:

SELECT (Campo_1,Campo_2,.....,Campo_n-3) **FROM** (Table_1,.....,Table_n)
WHERE (Exp)
ORDER BY (Exp)
HAVING (Exp)

Son de estructura diferentes por lo tanto van a pertenecer a grupos diferentes. Esto va determinar un conjunto de grupos. Puede ocurrir que 2 consultas diferentes estructuralmente devuelvan el mismo resultado aun así dichas consultas estarán en distintos grupos.

4.6.1.3 Estructura de datos

La estructura de datos resultante para representar los grupos de consultas es de tipo jerárquico, es decir un árbol de 3 niveles en el cual tenemos la raíz que no representa nada y luego el primer nivel son los grupos de consultas y luego de cada grupo cuelgan los registros del log correspondientes a dicho grupo, gráficamente:

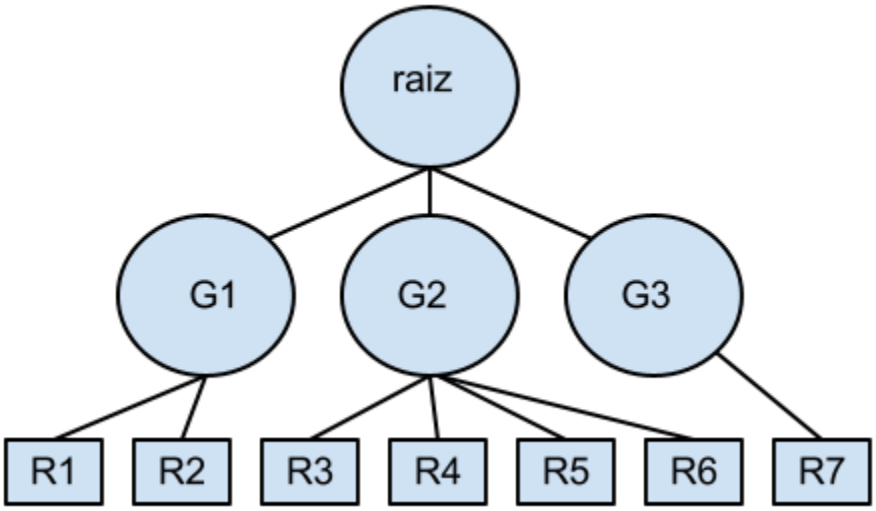


Figura 61 – Representación jerárquica de los datos del log cuando agrupamos por similaridad.

Donde G1...G3 son los grupos determinados en la etapa de parseo y R1...R7 son los registros del log, los hijos de cada grupo son los registros cuya consulta son similares entre sí. La raíz en este caso no cumple ninguna función en particular pero para nuestro caso para poder computar el tiempo promedio general y otra información nos pareció conveniente.

4.7 REPRESENTACIÓN

Es en esta etapa en la cual plasmamos la metáfora visual que mejor represente la información y que permita inferir y responder las preguntas que pueda tener un observador para el dominio de aplicación dado.

4.7.1.1 Introducción

El objetivo de la representación es identificar rápidamente aquellos grupos de consultas que más penalizan el rendimiento de una manera intuitiva y a partir de ahí poder navegar la información contenida en cada registro del log como el tiempo de ejecución de la consulta, host de donde se hizo la invocación así como también poder ver el código SQL original que fue ejecutado.

Primero analizamos la estructura de la información a visualizar y a continuación un extracto de la muestra de datos extraída de un log:

UPDATE jforum_topics_watch SET is_read = 1 WHERE topic_id = 2577 AND user_id = 8181;
Time: 050718 19:45:49
User@Host: db[db] @ localhost [127.0.0.1]
 # Query_time: 25 Lock_time: 0 Rows_sent: 1673 Rows_examined: 3076476

Los atributos que vemos interesantes para tener en cuenta para generar la metáfora visual son los siguientes:

- **Query:** En este campo viene la consulta cuyo tiempo supero el umbral definido en el servidor de base de datos. Una query puede ser cualquier tipo de consulta (SELECT, UPDATE, etc) que haya superado el umbral.
- **Query_Time:** Es el tiempo que ha tardado en ejecutar la consulta propiamente dicha.
- **#Rows_examined:** la cantidad de registros examinados por la consulta, uptate o delete.
- **User:** Es el usuario con el cual se ejecutó la consulta.
- **Host:** La máquina desde donde se originó la llamada.
- **Lock_time:** Tiempo de bloqueo de todos los registros scaneados por la consulta.
- **Rows_sent:** Los registros devueltos como resultado de la consulta.
- **Rows_examined:** La cantidad de registros examinados para producir el resultado.

La razón por la cual son interesantes es que proporcionan información contextual muy útil que ayudaran a responder las preguntas como cuáles son las consultas que tardan más tiempo, cómo son dichas consultas, qué usuarios las han ejecutado, desde qué máquina, entre otras cosas.

Para la representación, en un principio ideamos dos metáforas visuales de manera de tratar de explotar lo más posible el concepto de claridad cognitiva y facilidad de interpretación. A continuación se describen dichas metáforas.

4.7.1.2 Metáfora visual basada en grafos

Pensamos en grafos pues nos pareció una buena idea para representar la información ya que tienen un gran poder expresivo y en algunos aspectos se adaptan bien a nuestro dominio. Por otro lado ya estábamos familiarizados con el uso de grafos ya que es un tema ampliamente visto a lo largo de la carrera.

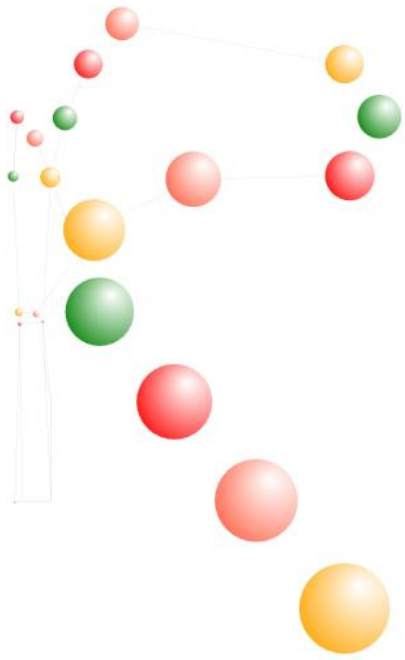


Figura 62 - Metáfora visual basada en grafos para la representación de los grupos que consultas.

La idea de esta representación es la siguiente:

Vértices: Los círculos representan a cada uno de los grupos de consultas definidos durante la etapa de parseo del slow query log, el diámetro de un nodo dado, dependerá de la cantidad de consultas que tenga ese grupo en particular.

Color Vértices: El color del círculo va variar entre verde y rojo, determinando la tonalidad de acuerdo al tiempo promedio de todas las consultas que componen el grupo, tomando el color rojo si está cerca del máximo y el color verde si está cerca del mínimo de la muestra (tiempo promedio).

Ejes: 2 vértices del grafo van a estar conectados si comparten tablas entre si. Esto lo consideramos útil para detectar conjunto de patrones de consultas altamente relacionados y tener una estadística de cuáles son las tablas más consultadas entre los grupos.

Luego de algunas pruebas el problema que vimos con este enfoque es que si bien tiene un muy buen poder expresivo no explota al máximo el espacio de la pantalla, es decir hay bastante *pixels* desperdiciados, considerando desperdicio todo aquello que no es visual fuera de la vista en sí. Otro problema es que para grandes volúmenes de datos la performance decrece pues generar el grafo implica recorrer todos los nodos para ver las tablas en común y agregar los ejes y luego cargarlo en la pantalla.

Por otro lado está el problema de las proporciones. Si por ejemplo tenemos grupos con miles de consultas y otros con menos de 10, muy probablemente dichos grupos no se verán a simple vista y sería necesario hacer zoom para poder apreciarlos. una posible solución es escalar de manera que haya una cota mínima y máxima para el tamaño de los nodos pero ya este también aumenta la complejidad de la generación del grafo en sí.

Por estos motivos se decidió descartar esta metáfora y tomar otro enfoque, esta vez utilizando otra técnica con *treemaps* que explote mejor el espacio de la pantalla.

4.7.1.3 Metáfora visual basada en treemaps

Luego del primer enfoque, pensamos en utilizar una metáfora visual que explote al máximo el espacio de representación y que al mismo tiempo tenga un gran poder expresivo de manera que sea rápido inferir y responder las preguntas que tenemos acerca de cuáles son las consultas que más penalizan la performance, ¿Cómo son? ¿Cuántas son? ¿Hay muchas similares? entre otras cosas.

Para esta metáfora visual utilizamos *Squarified Treemaps* [7]. Esta técnica de visualización tiene muy buenos resultados sobre conjuntos jerárquicos, fue ideada a principios de los 90 por Ben Shneiderman, que permite representar jerarquías de forma que se optimiza el llenado del espacio y, además, ver los atributos de la misma e identificar patrones anómalos o propiedades de la jerarquía usando colores.

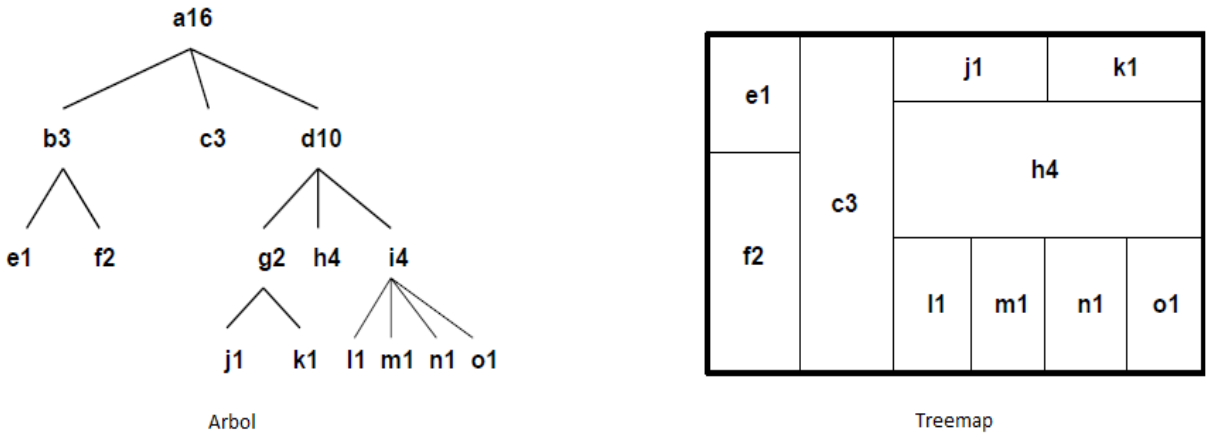


Figura 63 - Estructura jerárquica de ejemplo y el *treemap* resultante.

La diferencia entre los *Treemaps* convencionales y los S*quarified Treemaps* es que los primeros tienen el problema de que en ciertos casos la proporción del tamaño de los cuadrados no es bien estimada y se generan rectángulos largos muy finos afectando la calidad de la visualización.

Supongamos que tenemos un rectángulo de 4 de alto y 6 de ancho, y supongamos que el rectángulo debe ser dividido en 7 rectángulos con áreas 6, 6, 4, 3, 2, 2 y 1 respectivamente. El algoritmo estándar utiliza un enfoque muy simple, el rectángulo es dividido horizontalmente o verticalmente y es entonces cuando los cuadrados ultra finos emergen (Figura 64).

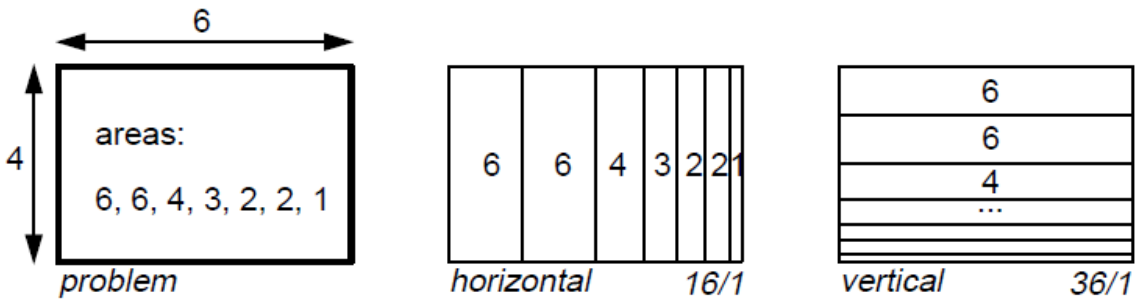


Figure 64 - Problema de la subdivisión en los *treemaps* convencionales.

Los *Squarified Treemaps* atacan este problema y optimizan la proporción de las áreas de los cuadrados de manera tal que no se generen cuadrados finos y la calidad de la visualización sea mejor, esto se logra calculando la relación de aspecto mediante un algoritmo de subdivisión que tiene como el objetivo tratar de mantener el *aspect ratio*[1] cercano a uno.

El primer paso del algoritmo es partir el cuadrado inicial. La heurística elige hacer una subdivisión horizontal porque el cuadrado original es más ancho que alto y luego a continuación se llena la mitad izquierda. Como vemos en la figura 65, primero se agrega un cuadrado, el aspect ratio es 8/3, a continuación agregamos otro rectángulo arriba del primero, los aspect ratio mejoran a 3/2 sin embargo si agregamos otro cuadrado arriba del último que agregamos el aspect ratio resultante sería 4/1, entonces es cuando el algoritmo determina que ha alcanzado el óptimo para la mitad izquierda y comienza a procesar la mitad derecha aplicando el mismo criterio en cada paso recursivamente.

[1] Aspect Ratio se define como max(altura/Largo; Largo/Altura).

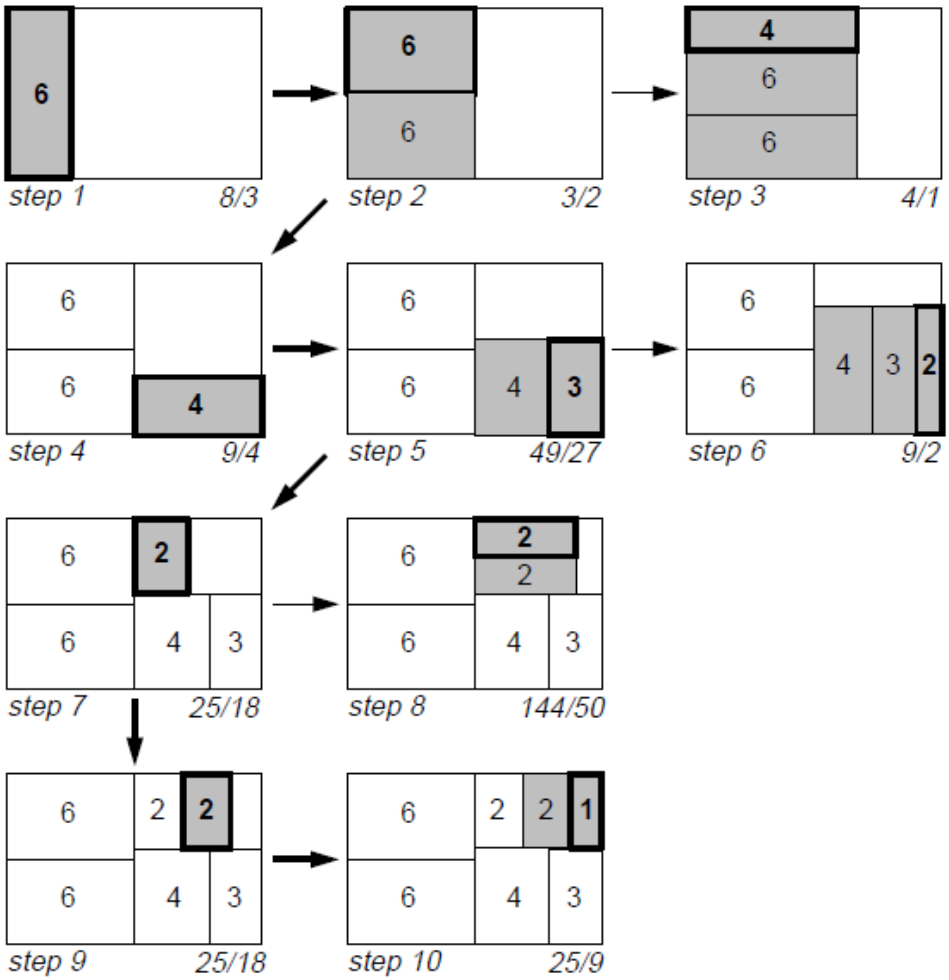


Figura 65 - Algoritmo de subdivisión para generar S*quarified Treemaps*.

A continuación describimos cómo hacemos el mapeo de atributos con los atributos del *Treemap* que básicamente son el tamaño, jerarquía, color y texto a mostrar.

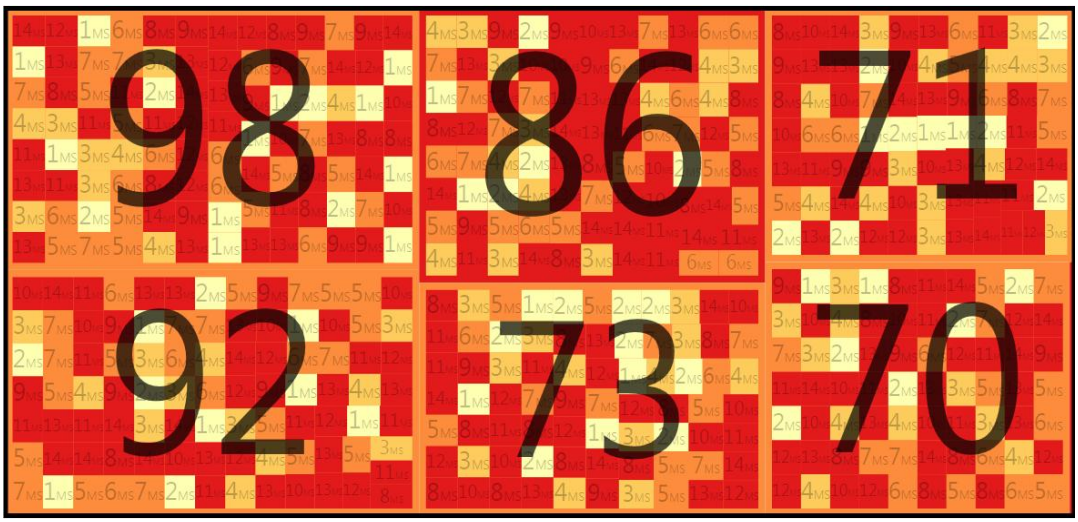
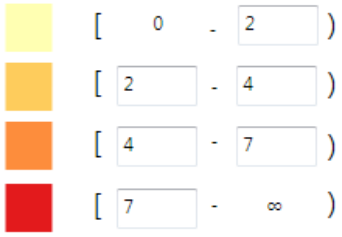


Figura 66 - Metáfora visual basada en *treemaps* para la representación de los grupos que consultas.

La figura 66 muestra el *Treemap* instanciado para un log, para el primer nivel tenemos un cuadrado/rectángulo para cada grupo. Recordemos que cada grupo representa un conjunto de consultas, para el ejemplo de la figura 66 se tienen varios grupos de consultas. El número contenido en cada cuadrado es el cardinal de dicho grupo y el color está dado por el tiempo promedio de acuerdo a la escala definida por el usuario. El usuario puede definir los intervalos de tiempo, actualmente hasta 4 intervalos es posible definir:



Cada intervalo tiene definido un color determinado.

De esta manera es fácil determinar cuál es el grupo de consultas que más penaliza el rendimiento del motor de base de datos para el esquema de base de datos dado. El tamaño del cuadrado/rectángulo va estar dado por la cantidad de consultas que componen el grupo en cuestión, a más consultas en un grupo que en otro, más grande será el cuadrado/rectángulo que la represente. Uno de los puntos más positivos de usar esta técnica es que los *Squarified Treemaps* [7] producen una visualización clara pudiendo observar desde los grupos más grandes a los grupos más pequeños todos al mismo tiempo, esto es porque la técnica toma en cuenta las proporciones al generar la división de los cuadrados generando proporciones escaladas de manera de poder explotar el tamaño de toda la pantalla de manera apropiada.

La principal desventaja al igual que la otra técnica es el problema de la performance, a grandes volúmenes de datos tenemos tiempos de respuestas por debajo de los esperados.

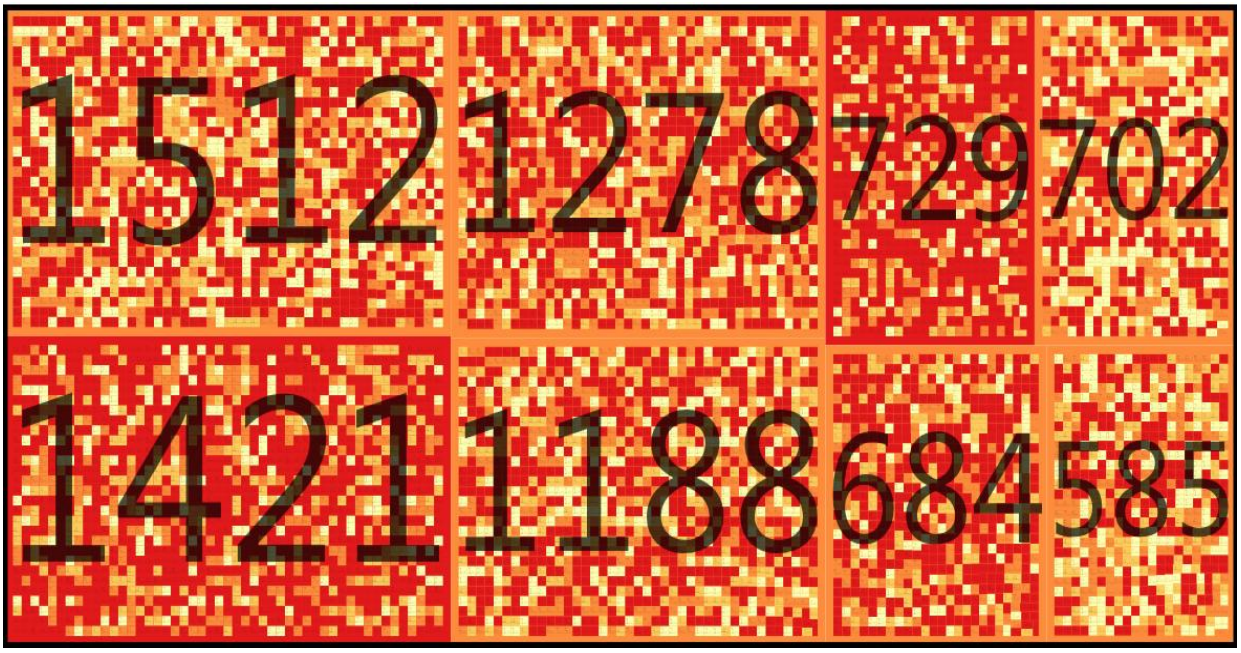


Figura 67 - Metáfora visual basada en *Treemaps* para la representación de un slow query log de tamaño considerable.

La imagen muestra el primer nivel, en el que para cada grupo tenemos un cuadrado/rectángulo. Para el ejemplo de la figura 67 se tienen varios grupos de consultas, el número contenido en cada cuadrado es el cardinal de dicho grupo y el color está dado por el tiempo promedio respetando los intervalos definidos por el usuario.

Luego para determinar los colores que cada escala debía llevar, es decir como colorear los cuadrados, utilizamos una paleta de colores propuesta por Cynthia Brewer [8]. Ella propone esquemas de coloración para diferentes contextos. En nuestro caso utilizamos el color para representar la variación del tiempo promedio de ejecución de los grupos de consultas y las consultas propiamente dichas. La paleta utilizada fue una secuencial.

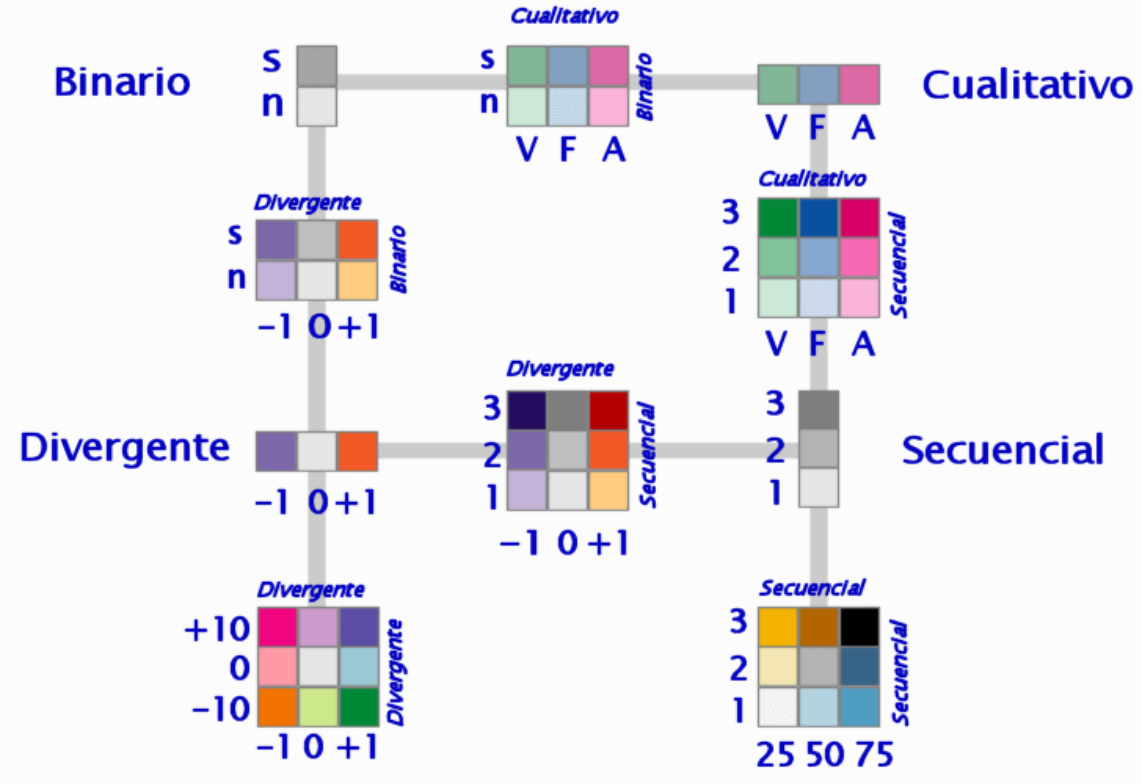


Figura 68 - Esquema de colores de Cynthia Brewer.

Esto es porque para representar datos ordenados en forma secuencial la recomendación más habitual es utilizar variaciones en la luminosidad o saturación de un color determinado. Por ejemplo podemos usar una gradación de rojo para representar datos de temperatura creciente. Usualmente se usan colores más subidos de tono (más oscuros) para valores más altos mientras que los colores más claros representan los valores más bajos, pero esto no es imprescindible, lo importante es asociar los cambios de luminosidad de forma que perceptivamente den cuenta de los cambios en los valores.

4.8 REFINAMIENTO Y INTERACCIÓN

En esta etapa mencionamos qué mejoras realizamos a la visualización de manera de hacerla más atractiva.

En un principio para la metáfora basada *treemaps*, la visualización solamente comprendía los grupos de las consultas. Luego pensamos que sería interesante poder refinar para que se puedan navegar de alguna manera las consultas que componen dicho grupo y de esta manera determinar cuáles consultas dentro de un grupo son las que más penalizan la performance para un esquema dado. Es por eso que agregamos la posibilidad de visualizar el registro del log propiamente dicho al llegar a una hoja del árbol.

Figura 69 - Rectángulo correspondiente a una hoja del árbol que muestra el registro del log.

También se agregaron tooltips para que en la vista de overview (arriba de todo) cuando el puntero del mouse se encuentra sobre un cuadrado dado, se muestre un tooltip conteniendo información referente al patrón del grupo y luego en el segundo nivel donde están las consultas el registro del log.

A medida que el usuario va navegando también se va actualizando el tiempo promedio total y el cardinal global de manera de mostrar la información del grupo actual de manera resumida.

Figura 70 – Tooltip que permite ver el patrón del grupo o bien el registro del log.

Los filtros que se definieron fueron los siguientes:

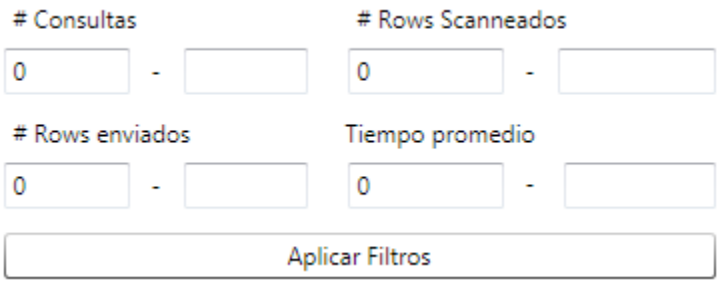


Figura 71 – Pantalla de filtros de SQLVis.

La idea es poder dotar al usuario la capacidad de filtrar por cantidad de consultas, la cantidad de registros examinados, cardinal del conjunto de datos devuelto y el tiempo promedio. Dichos filtros se pueden combinar, por ejemplo si quisiéramos obtener todas las consultas que demoran entre 1 a 6 segundos en promedio y cuyo conjunto resultados sea de cardinal entre 1 a 23 registros, es decir:

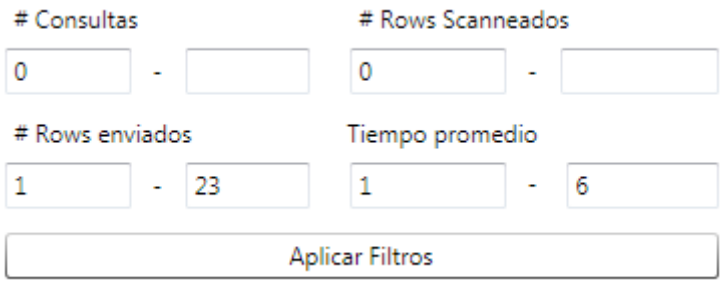


Figura 72 – Ejemplo de aplicación de filtros en SQLVis.

Aplicando el filtro obtenemos el siguiente resultado:

Figura 73 – Visualización resultante obtenida luego de aplicar un filtro a la visualización de la figura 70.

De esta manera la herramienta brinda flexibilidad para navegar y filtrar la información de manera de descartar la información no deseada y lograr focalizar en determinados aspectos de la visualización.

4.9 DETALLES DE IMPLEMENTACIÓN

A continuación mencionamos cómo implementamos la herramienta desde el punto de vista técnico, la tecnología utilizada fue C# y el lenguaje de la capa de presentación fue WPF.

Planteamos una arquitectura modular orientada a objetos separando la funcionalidad en diferentes módulos como se detalla en el siguiente diagrama:

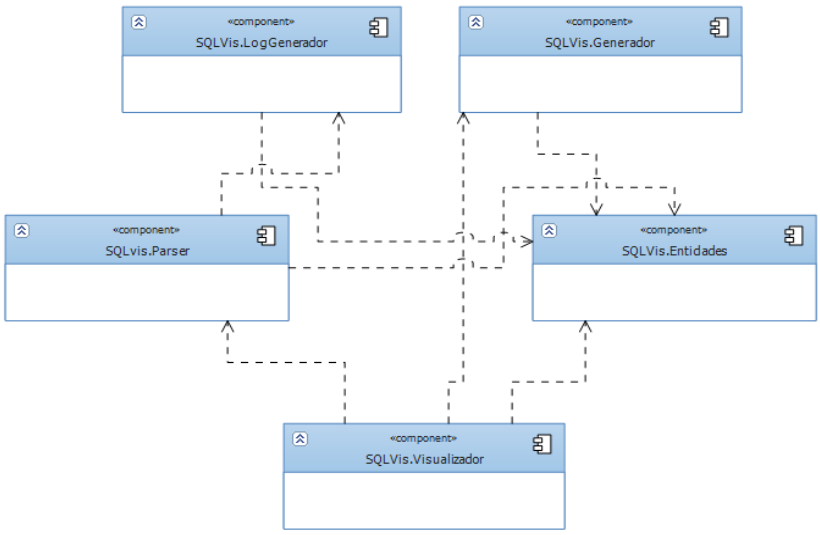


Figura 74 – Vista de módulos de SQLVis.

Para implementar los treemaps utilizamos una librería llamada *CLCV* a partir de la cual creamos un control propio para satisfacer las necesidades de la visualización como el *hover* sobre el mouse y la navegación.

Utilizamos la librería *General SQL Parser .NET* para parsear las consultas contenidas en los logs de manera de poder descomponerlas y poder analizar la similaridad entre ellas. La pantalla principal de la aplicación tiene el siguiente aspecto:

Figura 75 – Pantalla principal de SQLVis.

En la parte inferior tenemos el panel de visualización el cual se llena completamente al cargar un log:

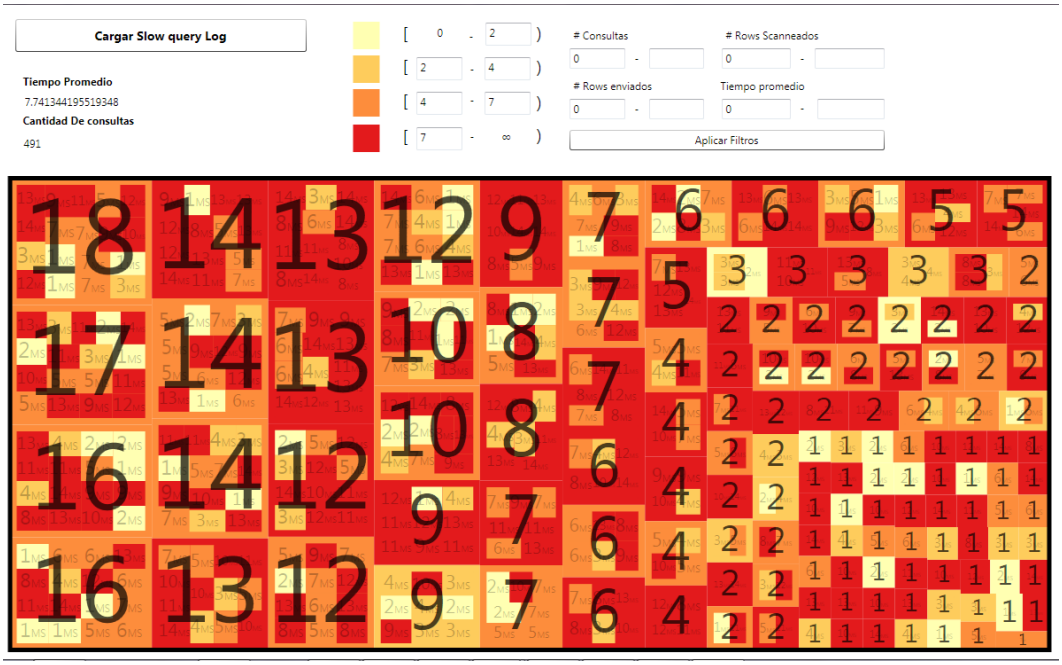


Figura 76 – Pantalla principal de SQLVis con un log cargado.

Abajo del botón de carga, tenemos 2 etiquetas que se van actualizando a medida que el usuario va navegando la información representada, la primera es el tiempo promedio total del grupo o muestra que está en foco en ese momento. Al cargar el log tendremos el tiempo promedio global a toda la muestra y luego al hacer zoom obtendremos el tiempo promedio del grupo. Al llegar al último nivel el tiempo promedio de la consulta misma. La misma idea aplicamos para la cantidad de consultas.

Luego a la derecha del botón de carga tenemos la parte de definición de rangos y los filtros que el usuario puede aplicar:

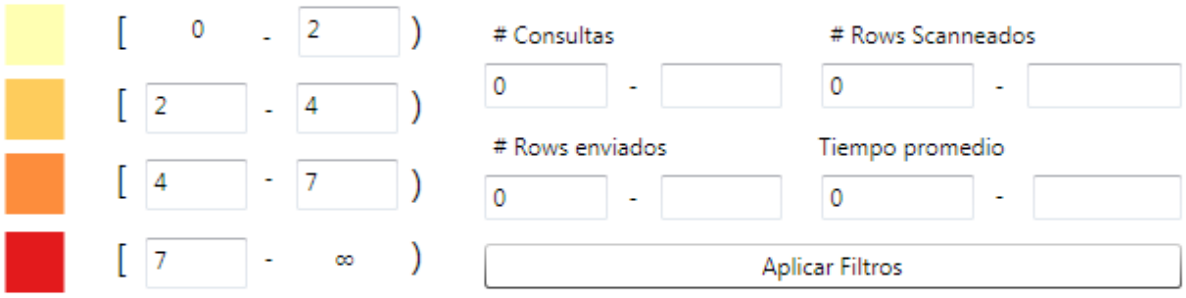


Figura 77 – Pantalla de definición de los intervalos y los filtros.

WPF permite crear interfaces graficas con gran versatilidad permitiendo vincular gráficos con datos de manera dinámica que se puedan actualizar en tiempo real sin necesidad de agregar complicadas rutinas y eventos para tal propósito. La aplicación desarrollada es una aplicación de escritorio para Windows.

Por otro lado también tomamos en cuenta la aplicación de varias de las buenas prácticas de diseño mencionadas por Tufte [1] y Ben Shneiderman [6] al momento de implementar la herramienta, las cuales se detallan a continuación:

- **Zoom**: La herramienta permite hacer zoom sobre grupos de consulta específicos de manera de poder navegar la información contenida en detalle, para ello la herramienta provee atajos para hacer zoom hacia dentro y hacia fuera.

- **Filtros**: Se proveen filtros para descartar la información que no aporta nada desde el punto de vista del usuario permitiendo que solo quede la información que realmente le interesa al usuario.

- **Detalles bajo demanda:** En cada vista se proveen tooltips que proveen detalle bajo demanda, a medida que el usuario va desplazando el puntero del mouse, la información desplegada se va actualizando con información referente al grupo o consulta, para el nivel superior se muestra el patrón de similaridad del grupo mientras que para el grupo inferior el detalle del registro completo que figura en el log.

Luego de Tufte implementamos las siguientes buenas prácticas:

- **Maximizar la tinta para Datos**: Al usar *Treemaps* para representar la información contenida en el log, nos garantiza que se utiliza el 100% de la pantalla para visualizar los datos.

- **Uso apropiado de los colores:** Para seleccionar los colores de acuerdo a lo que recomienda Edward Tufte seleccionamos una paleta de colores con transiciones suaves y tonos naturales usando una herramienta desarrollada por Cynthia Brewer [8] para obtener los tonos correctos.

- **Evitar el *Chartjunk*:** Tuvimos cuidado de no incluir ningún elemento que no aportara nada a lo que respecta a la visualización como leyendas o textos innecesarias entre otras cosas.

5 PRUEBAS

Las pruebas en sí tuvieron dos objetivos generales, el primero verificar que la herramienta desarrollada funcionara correctamente y fuera capaz de responder las preguntas como qué grupo de consultas son las que más penalizan el rendimiento para el esquema de datos dado y cómo son esas consultas y por otro lado hacer un estudio sobre una muestra real en donde poder sacar conclusiones sobre la misma. Para esta etapa del proyecto quedó fuera de alcance proceder a la optimización de las consultas o el esquema de base de datos.

Las primeras pruebas que hicimos sobre el prototipo consistieron sobre instancias simuladas que nos permitieron corroborar que efectivamente la herramienta funcionara bien y corregir errores en la visualización así como también realizar refinamientos, el tamaño de las instancias en un principio fue pequeño y a medida que fuimos refinando la visualización comenzamos a realizar pruebas con instancias más grandes.

Para la siguiente muestra pequeña de log:

 # Time: 4/27/2012 11:25:33 AM
 # User@Host: db[db] @ localhost [127.0.0.1]
 # Query_time: 23 Lock_time: 0 Rows_sent: 12349 Rows_examined: 9927
Select SupplierID
From Suppliers

 # Time: 4/27/2012 11:25:33 AM
 # User@Host: db[db] @ localhost [127.0.0.1]
 # Query_time: 1 Lock_time: 0 Rows_sent: 1 Rows_examined: 15669
Select CompanyName,
ContactTitle,
Country,
Address,
ShipPostalCode,
EntityID,
OrderID,
Country,
Shipped,
Amount,
Qty
From Customers,Orders
Where OrderID = 234532

 # Time: 4/27/2012 11:25:33 AM
 # User@Host: db[db] @ localhost [127.0.0.1]
 # Query_time: 12 Lock_time: 0 Rows_sent: 12349 Rows_examined: 12349
Select ContactTitle,
CompanyName,
Phone,
Region,
SupplierID,
PostalCode,

Address,
amount
From Suppliers

 # Time: 4/27/2012 11:25:33 AM
 # User@Host: db[db] @ localhost [127.0.0.1]
 # Query_time: 1 Lock_time: 0 Rows_sent: 123 Rows_examined: 15669
*Select **
From Customers,Orders
Where Amount > 2000

 # Time: 4/27/2012 11:25:33 AM
 # User@Host: db[db] @ localhost [127.0.0.1]
 # Query_time: 1 Lock_time: 0 Rows_sent: 23 Rows_examined: 15669
Select CompanyName,
ContactTitle,
Country,
Address,
ShipPostalCode,
InternalStructure
From Customers,Orders
Where Amount > 3000

 # Time: 4/27/2012 11:25:33 AM
 # User@Host: db[db] @ localhost [127.0.0.1]
 # Query_time: 24 Lock_time: 0 Rows_sent: 12349 Rows_examined: 12349
*Select **
From Suppliers

 # Time: 4/27/2012 11:25:33 AM
 # User@Host: db[db] @ localhost [127.0.0.1]
 # Query_time: 23 Lock_time: 0 Rows_sent: 12349 Rows_examined: 9927
*Select **
From Suppliers

 # Time: 4/27/2012 11:25:33 AM
 # User@Host: db[db] @ localhost [127.0.0.1]
 # Query_time: 9 Lock_time: 0 Rows_sent: 344 Rows_examined: 11239927
Select ContactTitle,
PostalCode,
Address
From Cities,Contacts
Where PostalCode = 34
group by ContactTitle

 # Time: 4/27/2012 11:25:33 AM
 # User@Host: db[db] @ localhost [127.0.0.1]
 # Query_time: 5 Lock_time: 0 Rows_sent: 234 Rows_examined: 11239927
Select ContactTitle,

PostalCode,
Address
From Suppliers,Customers
Where PostalCode = 23
group by ContactTitle

 # Time: 4/27/2012 11:25:33 AM
 # User@Host: db[db] @ localhost [127.0.0.1]
 # Query_time: 6 Lock_time: 0 Rows_sent: 23 Rows_examined: 11239927
Select ContactTitle,
PostalCode,
Address
From Suppliers,Customers
Where PostalCode = 01
group by ContactTitle

 # Time: 4/27/2012 11:25:33 AM
 # User@Host: db[db] @ localhost [127.0.0.1]
 # Query_time: 6 Lock_time: 0 Rows_sent: 17173 Rows_examined: 217173
*Select **
From EmployeeTerritories,Suppliers,Customers,Cities,Provinces
where city = 'BA'
group by City,province

 # Time: 4/27/2012 11:25:33 AM
 # User@Host: db[db] @ localhost [127.0.0.1]
 # Query_time: 15 Lock_time: 0 Rows_sent: 23 Rows_examined: 11239927
Select ContactTitle,
PostalCode,
Address
From Suppliers,Customers
Where PostalCode = 01
group by ContactTitle

La visualización resultante es la siguiente:

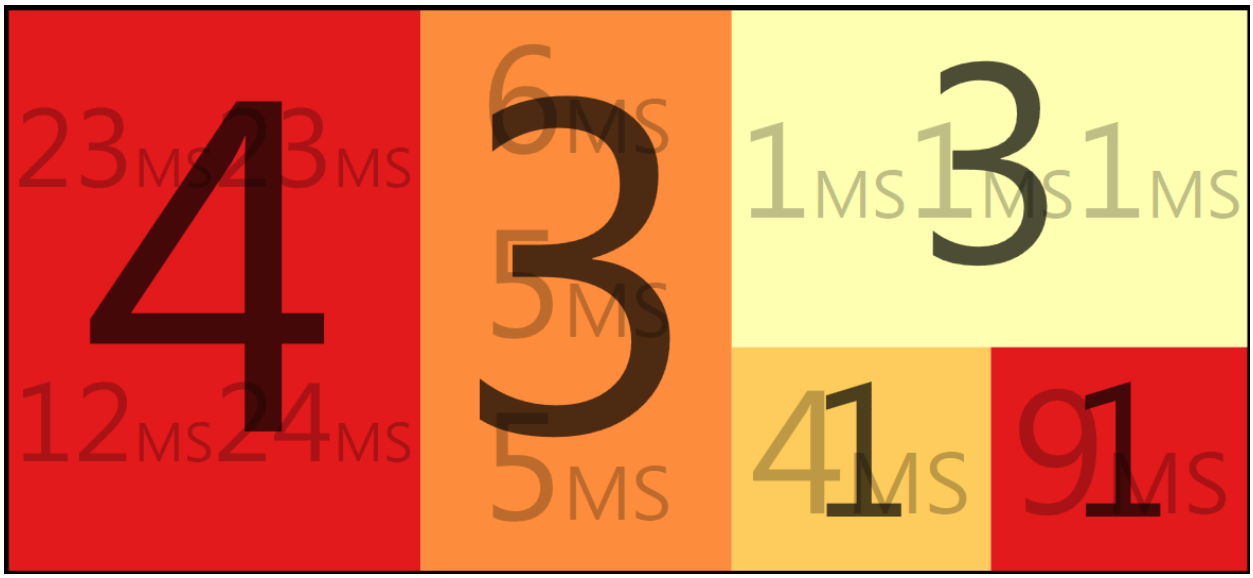


Figura 78 – Visualización obtenida a partir de la muestra de log de ejemplo.

Para esta pequeña muestra podemos verificar que los colores son correctos para los tiempos de ejecución de cada grupo de consultas de acuerdo a los rangos definidos y que el patrón implementado para agruparlas está funcionando bien.

La escala es customizable, es decir que el usuario puede ingresar a su criterio cual son los valores por los cuales se definen los colores:

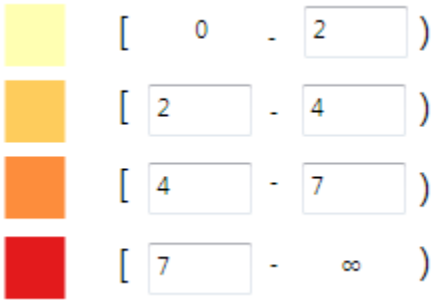


Figura 79 – Rangos de tiempo definidos.

Podemos observar que es fácil distinguir cual es el grupo con más problemas de *timing*, entonces la idea es que un observador pueda navegar los elementos de ese grupo y ver cómo son. El *tooltip* en cada nivel ayuda en cierta forma a comprender la información.

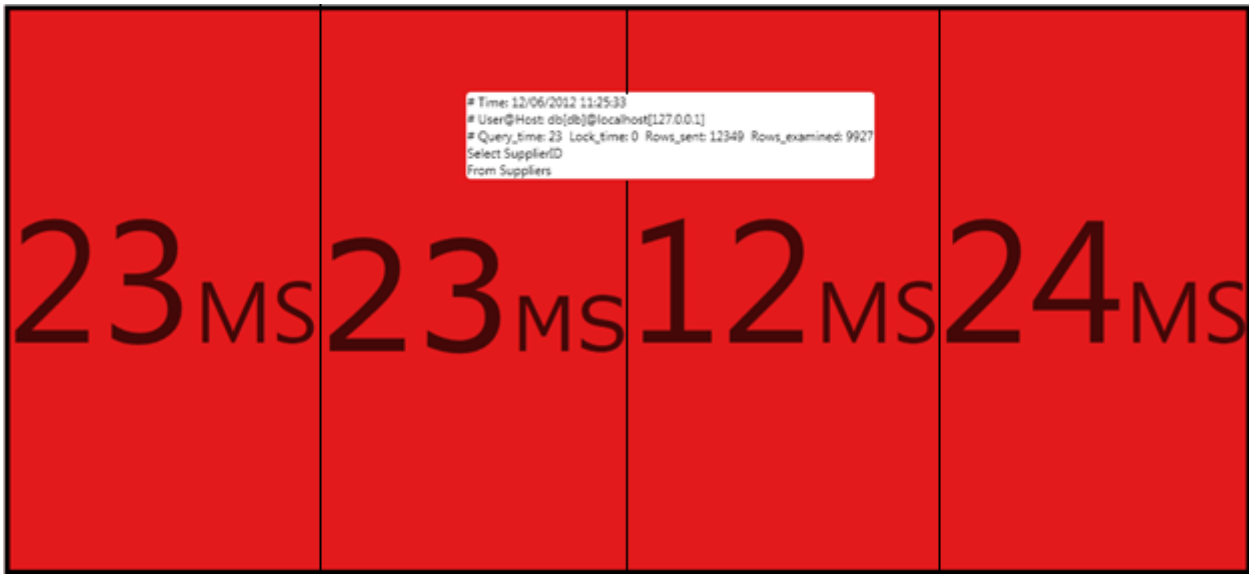


Figura 80 – Zoom del grupo con peores tiempos de ejecución.

Navegando el grupo en donde tenemos consultas con los peores tiempos de ejecución, tenemos que el tiempo promedio total de este grupo es de 20.5. Como son similares las consultas, posiblemente el problema sea común a todas ellas y de solucionarlo se mejoraría el tiempo notablemente. Para ello veamos cómo es una consulta de dicho grupo:

```
# Time: 09/24/2012 11:25:33
# User@Host: db[db]@localhost[127.0.0.1]
# Query_time: 12
 Lock_time: 0
 Rows_sent: 12349
 Rows_examined: 12349
Select ContactTitle, CompanyName, Phone, Region, SupplierID,
PostalCode, Address, amount From Suppliers
```

Figura 81 – Detalle de un registro del log en SQLVis.

Para las pruebas finales utilizamos una muestra de datos real provista por el Departamento de Computación de la FCEyN de la UBA. Básicamente consistió en varios archivos de logs que luego unimos en uno solo para usar en la herramienta. Hay que aclarar que fue bastante complicado conseguir muestras reales para experimentar, ya que es muy común que estos tipos de logs almacenen información sensible que no debe ser expuesta a ojos de terceros por lo tanto no es de extrañar que no se consiga un log completo buscando por internet. Por esta razón, al ver esta problemática en un principio optamos por simularlos generándolos nosotros a través de una herramienta desarrollada por nosotros. Luego pudimos conseguir una muestra real de tamaño considerable y basamos nuestras pruebas sobre esta muestra.

Como también hay información sensible, debimos ofuscar determinados campos como usuarios, nombres de servidores y otra información que pueda ser vulnerable.

Al cargar el Log obtenemos la siguiente visualización:

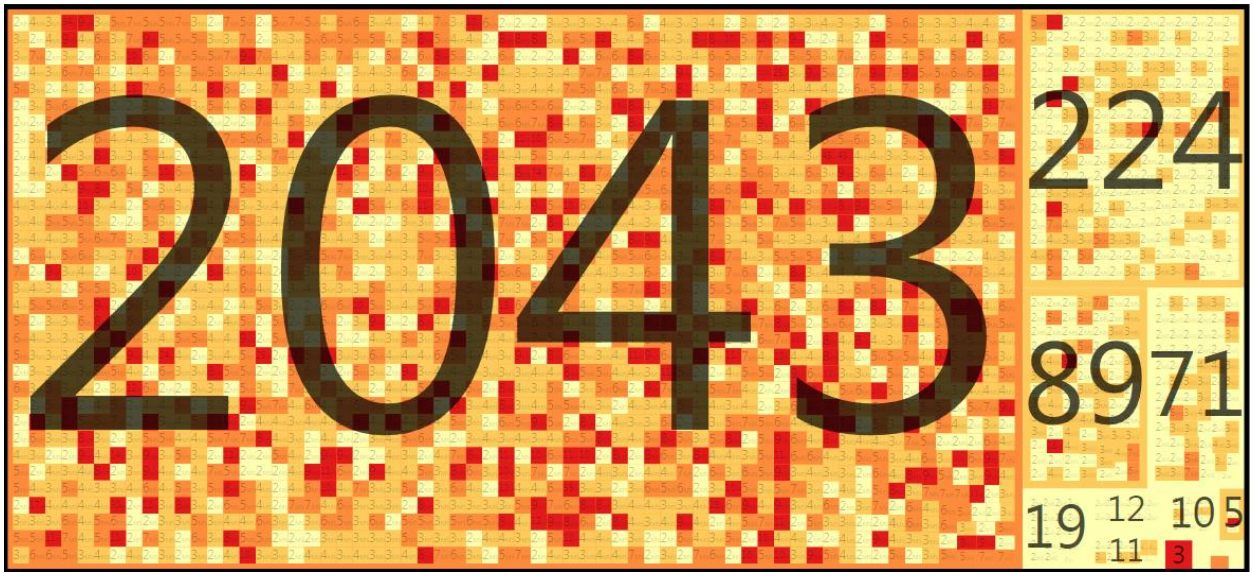


Figura 82 – Overview de cómo queda la visualización luego de cargar el log.

Podemos ver que al cargar el log quedan determinados 14 grupos, siendo el más grande de más de 2000 consultas y luego grupos de una sola consulta, Ya a estas alturas podemos inferir en esta primera instancia de exploración de la información que el grupo más grande es el que tiene las consultas con tiempos más largos de ejecución, esto es fácilmente visible al observador pues la cantidad de cuadrados rojos es muy superior en comparación a los otros grupos, Los mencionados grupos pueden verse en la siguiente tabla:

Patrón	Tiempo promedio	Cantidad de consultas	ejemplo
SELECT (a1,..an) FROM bayer_token WHERE (Exp)	5.066	2043	Select count(*) from Bayes_token Where id = ‘sth’ and (‘num’ - atime) > numcota
UPDATE bayes_token SET (A1=a1,..,An=an) WHERE (Exp)	3.357	224	UPDATE bayes_token SET spam_count = spam_count + '1', atime = 'num' WHERE id = 'num' AND token = 'usuario' AND spam_count + '1' >= 0;
INSERT INTO bayes_token Values (a1,..an)	3.044	89	INSERT INTO bayes_token (id, token, spam_count, ham_count, atime) VALUES ('num','chars','1','0','num');
SET (Exp)	2.577	71	Set autocommit = 1
SELECT (a1,..an) FROM awl WHERE (Exp)	2	19	SELECT count, totscore FROM awl WHERE username = 'nobody' AND email = 'mailOfuscado@serv.com' AND ip = 'LaIp'
SELECT (a1,..an) FROM bayes_seen WHERE (Exp)	2	12	SELECT flag FROM bayes_seenWHERE id = 'id' AND msgid = string';
INSERT INTO awl Values (a1,..an)	2.2	10	INSERT INTO awl (username,email,ip,count,totscore) VALUES ('nobody','ircnwbtdwsr@yahoo.co.jp','200.175','1','36.259');
SELECT (a1,..an) FROM bayes_expire WHERE (Exp)	4	5	SELECT max(runtime) from bayes_expire WHERE id = '5177';

DELETE FROM TABLE bayes_token Where (Exp)	14	3	DELETE from bayes_token WHERE id = '3739' AND atime < '1352543272';
SELECT (a1,..an) FROM bayes_vars WHERE (Exp)	2	2	SELECT spam_count, ham_count, token_count, last_expire, last_atime_delta, last_expire_reduce, oldest_token_age, newest_token_age FROM bayes_vars WHERE id = '5177';
SELECT (a1,..an) FROM docs, publidocs WHERE (Exp)	2	2	use basilic; SELECT type,source,sizeX,sizeY FROM docs,publidocs where docs.id=publidocs.idDoc AND publidocs.idPubli=475 AND (type='IMG' OR type='PDF' OR type='PS' OR type='PPT') ORDER BY type DESC,source ASC;
SELECT (a1,..an) FROM public, publiauthors, authors WHERE (Exp)	2	1	use basilic; SELECT DISTINCT publis.* FROM publis, publiauthors, authors WHERE 1 AND publis.id=publiauthors.idPubli AND authors.id=publiauthors.idAuthor AND authors.id=169 AND entry!='PhdThesis' AND entry!='TechReport' AND entry != 'MastersThesis' AND entry != 'Manual' AND entry != 'Misc' AND entry != 'Proceedings' AND entry != 'Unpublished' ORDER BY year DESC, title ASC;
UPDATE bayes_vars SET (A1=a1,..,An=an) WHERE (Exp)	2	1	UPDATE bayes_vars SET last_expire = ‘1352812351’ WHERE id = ‘4071’;
INSERT INTO bayes_expire Values (a1,..an)	5	1	INSERT INTO bayes_expire (id,runtime) VALUES ('3850','1352433651');

Identificados los grupos, ahora lo ideal es identificar aquellas consultas que cuyo tiempo de ejecución es por arriba del umbral del peor caso, para ello aplicamos un filtro por rango de tiempo:

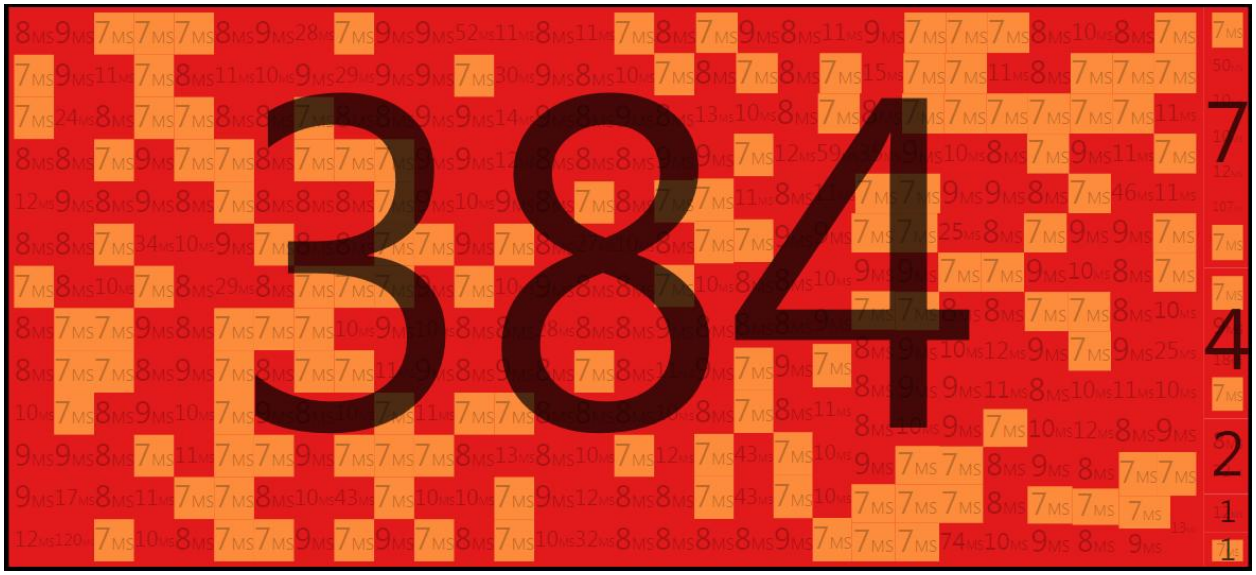


Figura 83 – Resultado obtenido luego de aplicar el filtro para detectar las consultas más lentas.

Observando la figura de arriba podemos decir que hay 6 grupos en los cuales se tienen consultas que tardan más de 7 segundos en ejecutar inclusive.

Patrón	Tiempo promedio	Cantidad de consultas	ejemplo
SELECT (a1,..an) FROM bayes_token WHERE (Exp)	10.08	384	Select count(*) from Bayes_token Where id = ‘sth’ and (‘num’ - atime) > numcota
UPDATE bayes_token SET (A1=a1,..,An=an) WHERE (Exp)	29	7	UPDATE bayes_token SET spam_count = spam_count + '1', atime = 'num' WHERE id = 'num' AND token = 'usuario' AND spam_count + '1' >= 0;

INSERT INTO bayes_token Values (a1,..an)	10.25	4	INSERT INTO bayes_token (id, token, spam_count, ham_count, atime) VALUES ('num','chars','1','0','num');
DELETE FROM TABLE bayes_token Where (Exp)	20	2	DELETE from bayes_token WHERE id = '3739' AND atime < '1352543272';
SELECT (a1,..an) FROM bayes_expire WHERE (Exp)	12	1	SELECT max(runtime) from bayes_expire WHERE id = '5177';
SET (Exp)	7	1	Set autocommit = 1

De este conjunto reducido, ya podemos ver a simple vista cual es la consulta más lenta de todo el log:

Figura 84 – Imagen de SQLVis con el detalle de la consulta más lenta de todo el log.

5.1 CONCLUSIONES PRUEBAS REALIZADAS

Una de los aspectos más positivos de las pruebas realizadas es que obtuvimos respuesta a las preguntas planteadas inicialmente. En este sentido la herramienta desarrollada cumplió satisfactoriamente con las expectativas y objetivos propuestos. Así como encontramos la consulta más lenta, podríamos haber encontrado la que devuelve más registros. También, si tuviéramos muchos grupos por filtrar existe la posibilidad de explorar sólo aquellos que tengan una determinada cardinalidad. Esto le da la facultad al usuario para hacer todo tipo de análisis sobre la información representada.

La idea es que lo que se pueda detectar en la herramienta sirva de entrada para elaborar una estrategia de optimización, es decir detectar las fallas más importantes y proceder a mejorar los tiempos de una manera organizada enfocándose primero en los grupos o consultas que tienen los tiempos de ejecución más largos. Dicha tarea se podría realizar de manera iterativa corriendo la herramienta nuevamente luego de aplicada las optimizaciones de manera de buscar nuevas mejoras.

Luego del proceso de optimización, es muy posible que grupos de consultas similares desaparezcan completamente del log o al menos su tiempo de ejecución disminuya considerablemente en el caso que sea factible una mejora tanto en la consulta como en el esquema de datos.

6 CONCLUSIONES

El objetivo de las herramientas de visualización es brindar una respuesta atractiva para la comprensión de grandes y diversas muestras de datos. Durante este proyecto hemos desarrollado un sistema de visualización para la representación de registros de consultas lentas de un log, para esta primer instancia específicamente para MySQL.

La idea del proyecto desde el principio fue realizar una visualización novedosa sobre un dominio poco explorado en lo que respecta a visualizaciones. Como hemos visto hay pocas herramientas visuales que puedan visualizar este tipo de datos y consideramos que la herramienta desarrollada puede ser de gran utilidad.

Estudiar las diferentes tecnologías de visualización existentes nos permitió tener un panorama amplio de cuales podíamos usar para desarrollar el proyecto, haber elegido WPF fue una buena decisión pues la flexibilidad que tiene para hacer cosas graficas sumado a que ya teníamos experiencia en tal tecnología facilito el desarrollo del proyecto.

La herramienta permite obtener un panorama general del estado de las queries de manera rápida y permite navegar la información de las mismas. El hecho de que las consultas estén agrupadas por similaridad da una idea de primero la cantidad de consultas similares que van a parar al log y luego la severidad. Es decir que ayuda a determinar por donde comenzar a optimizar y establecer una prioridad a la hora de encarar un proceso de mejora en las consultas.

6.1 TRABAJO FUTURO

El primer punto a desarrollar en un futuro cercano será la mejora del prototipo para dotarlo de aquellas características que quedaron fuera del mismo como ser que soporte varios tipos de slow query logs como por ejemplo el de Postgres entre otros. El desarrollo del prototipo estuvo focalizado en MySQL como motor de base de datos.

Otra gran mejora es poder aparte de visualizar, brindar funcionalidad para qué, dada una consulta se pueda explicar el plan de ejecución de manera de encontrar anomalías y mejorar los tiempos de las consultas de un grupo dado. Existen comandos específicos en MySQL como *Explain* que permite ver el plan de ejecución de una sentencia dada de manera textual, Un paso próximo seria hacerlo de manera grafica, Por ejemplo si tuviéramos la siguiente consulta:

```
SELECT City.Name, Country.Name
FROM City
JOIN Country on (City.CountryCode=Country.Code)
```

La idea sería mostrar en un frame aparte el plan de ejecución que tuvo esa consulta similar a la imagen que se muestra a continuación:

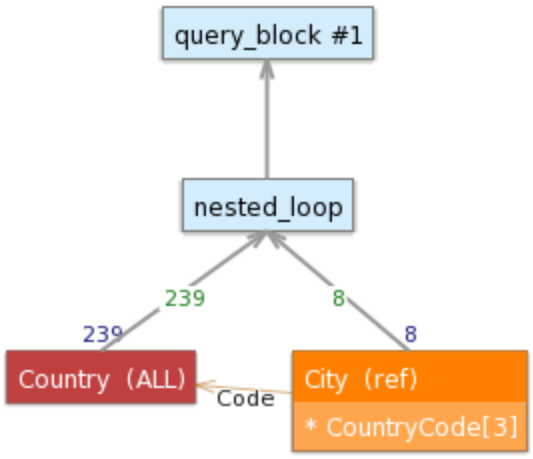


Figura 85 – Visualización del plan de ejecución para una consulta en Visual Explain.

Esto sería útil para entender las razones de la mala performance para una consulta SQL dada, como por ejemplo detectar si en las juntas se están utilizando índices sobre los campos correspondientes. La imagen que vemos es generada por la herramienta denominada *Visual Explain* desarrollada para MySQL.

Básicamente lo que representa ese plan de ejecución es lo siguiente, el cuadrado rojo en la parte inferior izquierda indica que va haber un full scan de la tabla *Country*, el Cuadrado naranja en la parte inferior derecha indica que la tabla ciudad tiene un índice cuyo nombre es *CountryCode* con una clave de longitud de 3 bytes. La flecha apuntando del cuadrado naranja al rojo significa que la columna *Code* será usada para hacer la junta entre las 2 tablas y el rectángulo azul en el medio está confirmando que el optimizador estará corriendo una junta anidada para obtener los datos.

Otra buena idea, seria complementar la vista de treemaps con una visualización secundaria cuyo propósito seria más que nada estadísticos sobre la información de cada grupo de consultas, para tal fin se podrían usar Box-Plots.

Un Box-Plot Es un gráfico que suministra información sobre los valores mínimo y máximo, los cuartiles Q1, Q2 o mediana y Q3, y sobre la existencia de valores atípicos[2] y la simetría de la distribución. Primero es necesario encontrar la mediana para luego encontrar los 2 cuartiles restantes:

[2] También llamados Outliers en estadística

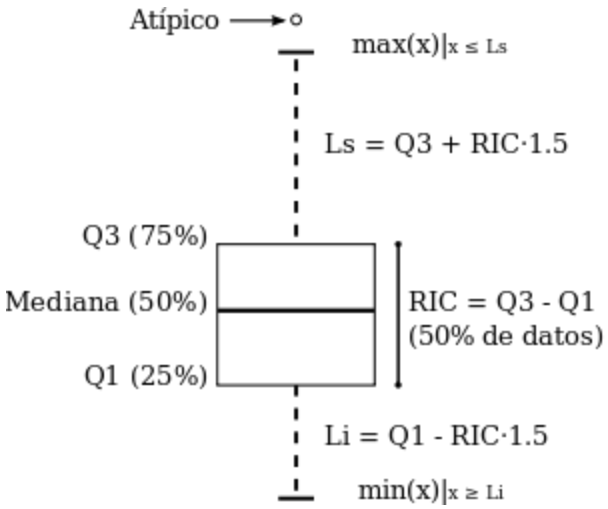


Figura 86 – Diagrama explicativo de un Box-Plot (diagrama de caja).

Entonces la idea sería que por cada grupo de consultas similares tengamos un box-plot para ver cómo se distribuye el tiempo de esas consultas en ese grupo.

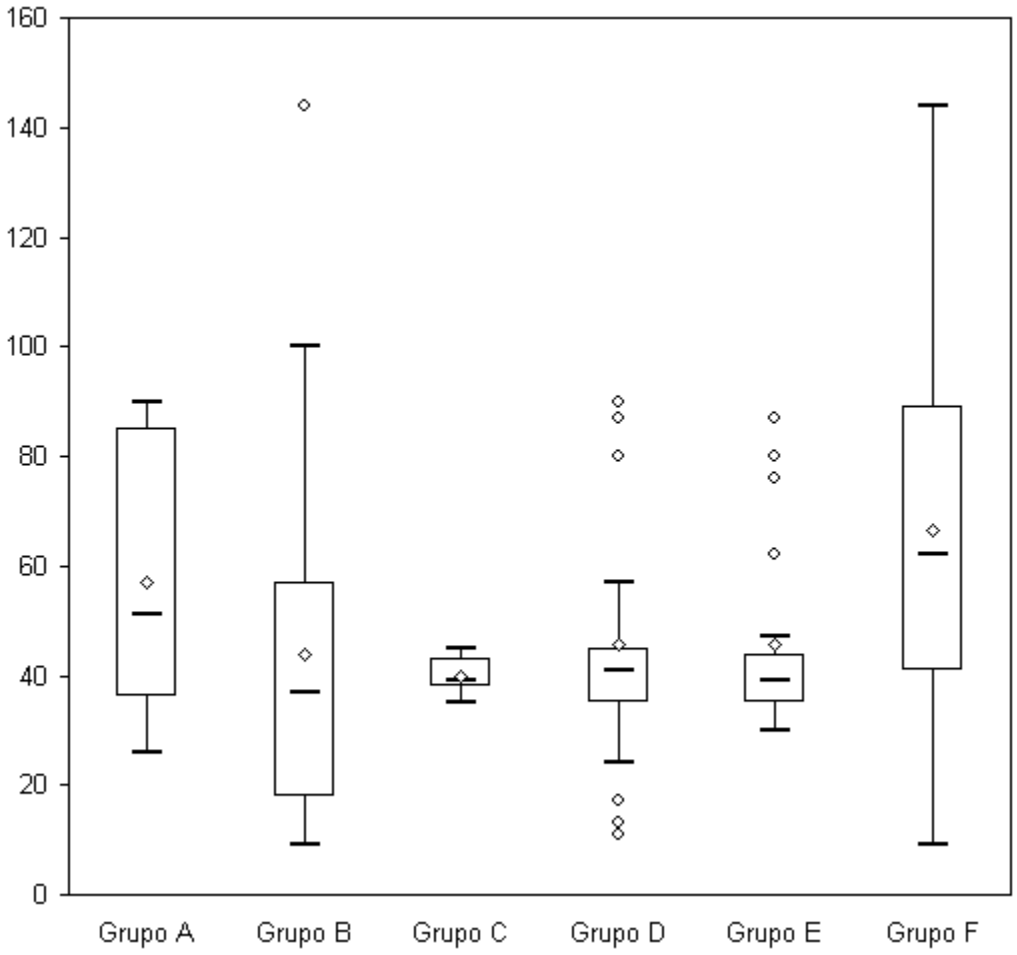


Figura 87 – Box-Plot esquematizado que representa cada uno de los grupos de consultas y los tiempos de ejecución.

7 BIBLIOGRAFIA

[1] E. R. Tufte. *"The Visual Display of Quantitative Information",* Second edition, Graphics Press, 2001.

[2] Ben Fry. "*Visualizing Data: Exploring and Explaining Data with the Processing Environment*", First edition, 2008.

[3] Jessica Hullman, Eytan Adar, and Priti Shah. "*Benefitting InfoVis with Visual Difficulties*", IEEE Transactions on Visualization and Computer Graphics, Volume 17 Issue 12, 2011.

[4] http://www.infovis.net , Es un proyecto dedicado a la divulgación de la Visualización de la Información. 2012.

[5] E. R. Tufte. "*Envisioning Information*", Graphics Press, 1990.

[6] Ben Shneiderman. "*The eyes have it: A task by data type taxonomy for information visualizations*". In IEEE Visual Languages, College Park, Maryland, U.S.A, Páginas 336–343, 1996.

[7] Mark Bruls , Kees Huizing and Jarke van Wijk. "*Squarified treemaps*", Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization. 1999.

[8] Cynthia Brewer. "*Color Use Guidelines for Mapping and Visualization*" in Visualization in modern cartography, ed. A. M. MacEachren and D. R. F. Taylor, paginas 123-147, Oxford-Pergamon Press, 1994.

[9] Christopher Ahlberg and Staffan Truvé. "*Tight Coupling: Guiding User Actions in a Direct Manipulation Retrieval System*", Chalmers University of Technology, Artikel 1995.

[10] Stephan Thiel. "*Understanding Shakespeare*" project, University of Postdam. 2010.

[11] D'Ocagne, Maurice. "*Coordonnées parallèles et axiales: Méthode de transformation géométrique et procédé nouveau de calcul graphique déduits de la considération des coordonnées parallèles*", Paris, Gauthier-Villars. 1885.

[12] Shedroff, N. "*An overview of understanding*". In R.S. Wurman, Information anxiety 2, paginas. 27-29. Indianapolis. 2001.

[13] Vladimir I. Levenshtein. "*Binary codes capable of correcting deletions, insertions and reversals"*, Soviet Physics Doklady, Paginas 707–710, 1966.

[14] Xabier, Berenguer. "*Escribir programas interactivos"*, Publicado en FORMATS, Universidad Pompeu Fabra, 1997.

[15] Juan C. Dürsteler. "*Acerca de la psicología cognitiva y la visualización"*, Infovis.net, 2007.

[16] Milko A. García Torres. "*Las leyes de la Gestalt*", Image & Art. 2012.

[17] B. Johnson and B. Shneiderman. "*Treemaps: a space-filling approach to the visualization of hierarchical information structures"*. In Proc. of the 2nd International IEEE Visualization Conference, páginas 284–291, 1991.

[18] Balzer, Michael; Deussen, Oliver. "*Voronoi Treemaps*", IEEE Symposium on Information Visualization (InfoVis 2005) Minneapolis, MN, USA, IEEE Computer Society, 2005.

[19] Phillip H. Smith. "*Electronic Applications of the Smith Chart in Waveguide, Circuit, and Component Analysis*", McGraw-Hill Book Co, first edition, 1969.

[20] Nancy R. Tague. "*The quality toolbox*", página 437, ASQ Quality Press, second edition, 2005.

[21] Michael Friendly, "*A Brief History of Data Visualization",* Springer-Verlag, 2006.