UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Máster Universitario en Ingeniería del Software, Métodos Formales y Sistemas de Información

Universitat Politècnica de València

Departamento de Sistemas Informáticos y Computación

Análisis de los Sistemas de Gestión de Bases de Datos actuales como soporte para las tecnologías de Internet de las Cosas.

Autor:
Solano Martes, Loanny Francisca

Tutor:
Celma Giménez, Matilde

Curso Académico 2012/2013

ABSTRACT

The Internet of the things refers to a network of traditional objects interconnected through Internet. The principal challenge that face off IoT in order to be deployed in a large scale is the needing of a Data Base Management System (DBMS) with the appropriate features to give solutions to one of the main challenges: The management of massive volume of information that is generated by IoT. In this master thesis, the current state of the DBMS technologies and the proposed solutions to these problems are analysed.

RESUMEN

Por Internet de las Cosas (IoT) se hace referencia a una red de objetos cotidianos interconectados a través de internet. El principal obstáculo al que se enfrenta IoT para que pueda ser implementado a gran escala es la necesidad de disponer de Sistemas de Gestión de Bases de Datos con características adecuadas que den solución a uno de los principales retos que se presentan: la gestión del gran volumen de información que se genera en IoT. En esta tesis de master, se analiza la situación actual de la tecnología de bases de datos y las soluciones propuestas a estos problemas.

A Boro, porque llegaste a mi vida como una luz que ilumina la oscuridad y porque me haces inmensamente feliz.

AGRADECIMIENTOS

A Dios: Ese ser supremo que me ha guiado por el buen camino, que me ha ayudado a seguir adelante a pesar de las luchas y dificultades, y que me ha llevado en sus brazos cuando me han fallado las fuerzas para continuar, gracias mi Dios, no tengo palabras para agradecerte por todas las bondades que haz hecho conmigo, aun sin merecerlo, por el aire que respiro, el sol que sale cada día y las hojas que se mueven en los árboles con la suave brisa que nos regala la lluvia en un atardecer, que demuestra tu existencia y la forma en que cuidas de mí, gracias mi Dios.

A mi madre : Leocadia Martes, cariñosamente "carlita", por ser una persona emprendedora, por demostrarme con hechos que si luchas por tus sueños lo puedes lograr, venciendo todos los obstáculos que encuentres en el camino, no importando las veces que tropieces, nadie dijo que sería fácil, pero no imposible, tienes que volver a levantarte y seguir adelante hasta poder alcanzar tus metas más anheladas, gracias mami, por estar ahí para mi cada vez que me sentí triste o desanimada, por darme los mejores consejos que me han ayudado en este arduo camino, le doy gracias a Dios por darme una madre como tú, este logro es para ti y simple y llanamente "Gracias por existir".

A mi padre : Florentino Solano, cariñosamente "Monchín", ejemplo de tenacidad y persistencia, quien me ha enseñado que los logros se consiguen si luchas sin desmayar, dándolo todo por el todo, demostrando que puedes lograrlo, no importando las dificultades, quien me ha enseñado a ver los obstáculos como retos que al principio son difíciles pero de alguna forma te ayudan a forjar tu carácter, a seguir adelante a pesar de las adver-

sidades, a dar lo mejor de ti, a demostrarle al mundo que eres un vencedor en esta batalla en donde las armas que utilizas son el amor al estudio y la perseverancia, y la meta a alcanzar, la más valiosa que puede existir, el conocimiento; gracias papi por demostrarme que si quieres, puedes lograrlo.

A mis hermanos : Nurkis, Charo, Annell y Nicole, y a mis dos hermosas sobrinas Sheryl y Mádison, quienes siempre estuvieron allí en momentos de tristeza para extenderme una mano amiga y demostrarme que no estaba sola, que siempre podía contar con ellos de manera desinteresada, o simple y llanamente para sacar una sonrisa de mis labios, y en momentos de alegría para demostrarme que mis logros eran sus logros y que disfrutaban mis éxitos como si fueran los suyos propios y por ser un rayito de luz en medio de la oscuridad, gracias hermanos míos, son una parte especia, muy especial en mí vida, los quiero inmensamente.

A Boro: Por apoyarme, por ayudarme, por siempre estar ahí para mí, por tenerme paciencia, pero sobre todo, gracias por quererme, eres muy especial para mí, "My hearth belong to you"

A esta hermosa ciudad, Valencia, destino hermoso, tranquilo y lleno de recuerdos bonitos.

A mis profesores de la UPV, en especial mi tutora de tesis Matilde Célma Gimémez, por ofrecerme cada día el pan de la enseñanza, por entregarme de una forma desinteresada la clave del éxito que es el estudio, por preocuparse cada día de mi aprendizaje, por guiarme por el camino correcto, por dame un ejemplo de vida, por animarme a continuar, estas cortas línea son para ustedes, padres de la enseñanza. Alcanzar el éxito sería difícil sin su ayuda.

Y a todos aquellos que de una u otra manera me extendieron la mano, muchas gracias.

Loanny Solano

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE CUADROS

CAPÍTULO 1

INTRODUCCIÓN

Desde los años prehistóricos el ser humano ha tenido la necesidad de mejorar su estilo de vida y facilitar sus tareas cotidianas; esa búsqueda constante de mejoría fue lo que lo impulsó a inventar, descubrir y crear nuevas técnicas y herramientas para garantizar su supervivencia ante los obstáculos e inclemencias que enfrentaban en su día a día.

Hoy en día esa necesidad sigue siendo de elevado interés para la humanidad, es una tendencia que en vez de disminuir va en total crecimiento, con usuarios mucho más exigentes aumenta la necesidad de nuevas tecnologás que cumplan con los requisitos emergentes de dichos usuarios.

Un ejemplo de esto es la domótica, lo que hace que cada vez tengamos mas sensores y actuadores en nuestros hogares, permitiendo al usuario controlar todos los servicios de su vivienda. Sin embargo el siguiente paso en estos sistemas es lo que se ha llamado "Internet de las Cosas" ("*Internet of Things (IoT)*"), que propone un paradigma distinto en el que todos objetos estarán interconectados entre si.

Figura 1.1: Esquema de un hogar inteligente

El número de sensores conectados a la red aumenta continuamente en todos los ámbitos: en el hogar, en el trabajo, en los lugares públicos, e incluso en nosotros mismos.

Los sensores son una tecnología habilitadora: si añadimos a un sistema pasivo la capacidad de ser consciente de su entorno, lo transformamos en un sistema activo, capaz de reaccionar a los eventos que ocurran a su alrededor. En la actualidad existen sensores para cualquier parámetro que queramos registrar (aceleración, posición, temperatura, humedad, consumo de energía, elementos químicos en el ambiente...)

Si construimos un sistema donde todos los dispositivos están generando una elevada cantidad de información, podría desbordar a un usuario al recibir toda esta información. Por ejemplo: que el sistema le comunique constantemente cada vez que se haya abierto o cerrado una puerta, o que el sistema de alumbrado le notifique que las luces fueron dejadas encendidas, o que el sensor que mide la temperatura le envíe un mensaje informándole cada vez que ocurra un cambios de la temperatura. A pesar de ser una información valiosa, acabaría por sobrecargar al usuario, al cual le gustaría estar informado de todo lo que ocurre, pero no está interesado en recibir a cada momento toda la información que se genera.

Uno de los factores más importantes de los que depende el éxito de IoT radica en la habilidad que tengan estos sistemas en hacer uso del flujo de datos que generarán estos sensores, que servirán de soporte para la gestión de recursos y generación de alertas sobre eventos significativos. Estos sistemas necesitan coleccionar información acerca de los usuarios y su entorno para hacer un apropiado análisis y filtrado de los datos de manera tal que puedan

ser usados para la toma de decisiones inteligentes. Para realizar este proceso, debemos utilizar técnicas de minería de datos.

1.1. Motivación y Justificación:

Por IoT nos referimos a una red de objetos cotidianos (desde objetos simples a los mas sofisticados) interconectados mediante internet. Se trata de una interconexión de multitud de objetos distintos relativamente sencillos y con cierta inteligencia, en forma de una red compleja, en la que el objetivo final es sensorizar, monitorizar, registrar y almacenar toda la información posible. Se estima que dentro de 10 años, por cada persona habrá de media 1000 sensores.[1]

Hoy en día hablamos de una tecnología que aún no ha alcanzado su periodo de madurez, pero que tiene un futuro muy prometedor, tan solo imaginemos un escenario en el que todos los objetos que utilizamos a diario formen parte de IoT, los libros que leemos, las mascotas, la comida, la ropa que utilizamos e incluso hasta nosotros mismos, todos estos objetos generando información de su estado, lo que supondrá una gran cantidad de información. Las bases de datos juegan un papel muy importante en esta área, ya que la información que produce cada uno de estos objetos debe ser almacenada y controlada para que a partir de ella se pueda hacer minería de datos y tomar decisiones.

La prioridad en este momento es la de encontrar métodos lo suficientemente eficientes y capaces de almacenar, indexar y acceder a toda esta información que será parte de IoT.[2]

1.2. Objetivos

La pregunta que nos hacemos es: ¿Cómo los sistemas de almacenamiento de datos se enfrentan al desafío de gestionar y almacenar toda la información que se genera en IoT?, ¿Estarán preparados los sistemas de bases de datos existentes para gestionar la información de un mundo conectado? Es necesario realizar estudios de las tecnologías existentes para el almacenamiento de datos para determinar si están preparados para la administración y los requerimientos de IoT.

Por todo esto, el objetivo de este trabajo de tesis es hacer un análisis de las tecnologías de Bases de Datos actuales como soporte los sistemas de Internet de las Cosas.

1.3. Plan de la Tesis

En el capítulo II, vamos a hacer una exposición del estado actual de las tecnologías involucradas en el desarrollo de esta tesis: IoT, domótica y sistemas de gestión de BBDD.

En el capítulo III estudiaremos los retos a los que se enfrentas los SGBBDD actuales a la hora de dar soporte a IoT y en que puntos se encuentran las debilidades.

Luego presentaremos en el capítulo IV soluciones que se han adoptado actualmente, las compararemos entre ellas y veremos en que puntos presentan problemas.

Por último presentaremos las conclusiones alcanzadas.

CAPÍTULO 2

ESTADO DEL ARTE

2.1. Internet de las cosas(IoT)

Cuando hablamos de IoT, nos referimos a un entorno en donde todos los dispositivos y objetos que utilizamos en nuestra vida cotidiana están interconectados entre sí.

El término IoT fue acuñado por Kevin Ashton y utilizado por primera vez en el año 1999, en el Instituto Tecnológico de Masschusetts. [3]

IoT se define como una red autónoma que por si sola interactúa y organiza objetos. La idea es que los objetos o "Cosas" sean capaces de procesar información, comunicarse entre ellos y con el medio ambiente y tomar decisiones de manera autónoma. Es la representación en el mundo virtual de objetos físicos, estos objetos a su vez deben transmitir en tiempo real datos acerca de su estado.

Para hacer posible que un objeto transmita sus datos es necesario el uso de tecnologías habilitadoras como son las etiquetas RFID (Radio Frecuency Identification Technology), dispositivos de sensores ubicuos, etc., el concepto de computación ubicua está relacionado con IoT, en el sentido de que los objetos físicos son habilitados por una larga escala de sensores embebidos.

La computación ubicua se define como la integración de la infor-

mática en el entorno de las personas de forma tal que los ordenadores no se perciban como objetos diferenciados. [1]

El término de computación ubicua fue acuñado e introducido por primera vez por Mark Weiser en el año 1988, cuando trabajaba para Xerox, en los laboratorios de Palo Alto(PARC), sin embargo fue en el año 1991 cuando adquirió reconocimiento mundial con la publicación del artículo *"The computer for 21st century"*, en el cual sienta las bases de la computación ubicua, cuya principal característica es la transparencia: *"Las tecnologías más profundas son las que desaparecen, se tejen ellas mismas sobre el tejido de la vida cotidiana hasta que no se distinguen de esta"*. [4]

Weisel percibía un nuevo nivel de integración con ciertas características como el que la tecnología no requiere atención y está disponible para su uso. Su visión era de una tecnología natural y totalmente transparente al usuario, él decía que cuando las cosas desparecen es que somos capaces de usarlas sin pensar y de esa forma podemos enfocarnos en otros objetivos, es como una señal de tránsito, cuando la vemos absorbemos la información inconscientemente sin la necesidad de leerla.

A esto se refería Weisel cuando hablaba de computación ubicua, la filosofía de un ordenador un usuario lo veía como un primer paso en la lucha por alcanzar el verdadero potencial, en sentido de que la interacción entre el usuario y la máquina no se realiza de forma integrada y transparente, su visión iba más allá de ese enfoque, se centraba en una tecnología que a pesar de formar parte de la vida cotidiana, se desvaneciera, y que en vez de servir de obstrucción, facilitara la vida del usuario.

Otra característica de los sistemas ubicuos es la autonomía, como expone W. Keith: *«para lograr el efecto de invisibilidad los sistemas deben ser tan independientes como sea posible de la administración humana»*[5]. Esto quiere decir que los sistemas deben tener cierta inteligencia a fin de poder tomar decisiones por si mismos.

A pesar de que los avances científicos no han ido tan rápido como vaticinaba Weiser, en los últimos años se han producido importantes avances en esa dirección.

Según estudios realizados[6], en el año 2011, el número de dispositivos interconectados en el planeta superaba el número de personas. Actualmente hay unos 9 billones de dispositivos interconectados y se espera que para el año 2020 se supere esa cifra a unos 24 billones de dispositivos. Esto

[1] https://es.wikipedia.org/wiki/Computación_ubicua

significa que un usuario tendrá al menos 6 dispositivos conectados a internet.

Ya predijo Bill Gates en el año 2003 la evolución que iba a experimentar la tecnología informática. Pronosticó que habría nuevos ordenadores optimizados para que cumplan tareas cotidianas de los usuarios; computadoras que tomen notas en una reunión, lean los correos electrónicos en el sofá, asistan al usuario en un entrenamiento, toquen música y películas en el televisor de la sala, y computadoras de bolsillo que mantengan a las personas conectadas e informadas donde quiera que estén [7].

La conclusión a la que llega Bill Gates es que podemos esperar que ese será solo el principio de una nueva generación, llegaremos al punto en el que la computadora desaparecerá, ya que los próximos ordenadores serán tan intuitivos y transparentes que difícilmente los notemos, corroborando con la visión de Weiser en años anteriores.

Para llegar a este ambiente de invisibilidad, se necesitan tecnologías más avanzadas y sistemas de descubrimiento que permitan al usuario conocer de la disponibilidad de algún servicio.Como es evidente Internet de las Cosas se puede ver como una evolución o profundización de la computación ubicua y sus términos paralelos [8].

2.2. Estructura de una red IoT

Figura 2.1: Estructura de una red IoT [9]

Un escenario IoT esta formado por tres niveles diferentes, cada uno con sus respectivas funciones; un ejemplo de un escenario IoT se muestra en la figura 2.1.

El primer nivel se conoce como "capa de reconocimiento" esta capa es la encargada de interconectar los objetos inteligentes con el internet(a través de tag RFD, scanners laser), con el objetivo de intercambiar información y servicios de comunicación.

El segundo nivel recibe el nombre de "capa de red", o "capa de sensores inalámbricos". Esta capa es responsable de trasmitir, procesar y clasificar la información recibida de la capa de reconocimiento.

En el último nivel esta la "capa de aplicaciones", en ella se localizan los servidores de las diferentes industrias.

De las tres capas, la principal es la capa de red, porque es la que provee los servicios en IoT y sirve como enlace entre la capa de reconocimiento y la capa de aplicaciones.

2.2.1. Elementos de IoT

Para que IoT pueda ser una realidad, existen tres componentes principales que no deben faltar: El primero de todos es el componente hardware formado por sensores, actuadores y sistemas de comunicación embebidos, el segundo es el middleware o software conformado por herramientas de computación para análisis de datos y sistemas de almacenamiento y el tercero es la presentación: herramientas de visualización, y diseño que puedan ser compatibles con diferentes plataformas.

- **Tecnología Radio Frequency Identification** Una de las tecnologías utilizadas para IoT es la (Auto-ID) o Identificación Automática; Un ejemplo de esta tecnología es la Radio Frecuency Identifications (RFID), utilizada para almacenar y recuperar datos de forma remota a través de dispositivos llamados etiquetas o tags RFID, Los tags RFID contienen una antena que es la que transmite y recibir las señales mediante ondas de radio y un circuito integrado que se encarga de almacenar la información. Los tags pueden ser adheridos a los objetos e incluso implantados en una persona o animal.

  Los tags RFID se ha popularizado enormemente y son utilizados en carreteras de peajes, gestión de pasaporte, bibliotecas, atención sanitaria, entre otras, aunque tiene algunos inconvenientes como la seguridad, ya que las etiquetas pueden ser leídas a distancia por cualquier usuario, aunque éste no sea el propietario del elemento.

- **Redes de sensores inalámbricos(WSN)**

  Una red de sensor inalámbrico esta formada por una colección de dispositivos autónomos, distribuidos físicamente, y que poseen la habilidad y capacidad de almacenar y comunicar datos en una red de forma inalámbrica.

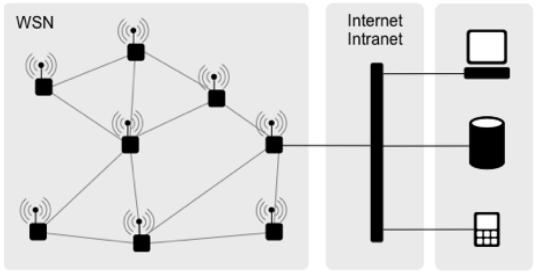

Figura 2.2: Red de sensores inalámbrica

Los sensores inalámbricos son colocados en el área de interés y se utilizan para el monitoreo de eventos, procesos, condiciones ambientales, etc; cada sensor actúa como enrutador ya que transmiten datos por múltiples nodos de entrada conectados a otras redes o a internet.

En la actualidad la tecnología de sensores inalámbricos ha mejorado notablemente, ya existen sensores inteligentes capaces de procesar, analizar y diseminar la información recopilada del entorno permitiendo que esta tecnología sea más viable, los factores que han influido en su éxito radican en las mejoras que han habido con relación a la eficiencia, el bajo costo y la disponibilidad de servicios etc.

Componentes requeridos en WSN:

a) **Hardware WSN:** los nodos están formados por unidades de procesamiento, suplidores de energía, interfaces de sensores, etc.

b) **Pila de comunicación WSN:** los nodos deben comunicarse con otras aplicaciones en la red, por lo que se requiere el diseño de una topología apropiada y el uso de un adecuando sistema de enrutamiento para mejorar la escalabilidad de la red y garantizar la transmisión de datos de forma segura. Cuando un nodo se encuentra inactivo, la pila de comunicación debe servir como interfaz entre los objetos de mundo real, la red de sensores inalámbricos y el internet.

c) **WSN Middleware:** el objetivo principal de middleware para redes de sensores es apoyar el desarrollo, mantenimiento, implemen-

tación y ejecución de aplicaciones de sensores. Apropiados mecanismos de abstracción son requeridos para manejar problemas de heterogeneidad de nodos. estos mecanismos deben respetar los principios de diseño y características especiales de WSN.

d) **Seguridad de agregación de datos:** para alargar el ciclo de vida de la red, un método seguro de agregación de datos es recomendable para asegurar la fiabilidad de los datos recolectados de los sensores.

Uno de los problemas más comunes en las WSNs está relacionado con el fallo de los nodos, seria recomendable que la topología que se utilice sea capaz de autorepararse por si misma. Otro aspecto crítico es la seguridad, tomando en cuenta que los sistema se comunican de forma automática con otros sistemas, es necesario que los mecanismos de seguridad a utilizar estén preparados para detectar anomalías (Detección de intrusos, ataques de negación de servicios, etc.) y garantizar la seguridad del sistema.

- **Esquema de direccionamiento:**

  El éxito de IoT dependerá primordialmente en contar con un sistema de direccionamiento efectivo para poder identificar y controlar los billones de dispositivos conectados a internet.

- **Almacenamiento de datos y análisis:**

  Nuestro mundo físico contiene datos de cientos y miles de objetos físicos manipulados por personas; la red solo posee una parte de estos, cuando se introduzca la infraestructura de IoT en la red esto causará una explosión de datos en grandes proporciones.

  Los datos necesitarán ser almacenados y manipulados de manera inteligente, por lo que se necesita desarrollar algoritmos inteligentes basados en arquitecturas centralizadas o distribuidas para facilitar la tarea de coleccionar y análisis de datos.

2.2.2. Tipos de datos en IoT

Los datos en IoT pueden ser de distintos tipos, existen datos que se generan de forma automática, mientras que otros son introducidos por los usuarios, pero en general lo podemos clasificar de la siguiente manera: [2]

- **Identificación por radiofrecuencia:** son los datos recopilados por los tags RFID y provienen de diferentes fuentes, la mayoría de estos datos son generados en tiempo real.

- **Direcciones/ Identificadores Únicos:** cada objeto que forma parte de IoT posee su identificador único, que es lo que lo distingue de los demás, tomando en cuenta la enorme cantidad de objetos que formarán parte de IoT en un futuro, se hace necesario un sistema de direccionamiento que pueda suplir estas necesidades.

  Uno de los avances notables que ha habido en cuestión de direccionamiento IP, ha sido el surgimiento del protocolo de internet de la versión (IPv6), que se espera pueda reemplazar la versión (IPv4) que está sufriendo el problema de agotamiento de direcciones, sin embargo IPv6 todavía no está extendido.

- **Datos descriptivos sobre objetos, proceso y sistemas:** en IoT, gran parte de los datos provienen de los registros de los objetos y los metadatos (son datos sobre datos, útiles para que el usuario pueda encontrar y acceder a la información apropiada).

- **Datos posicionales:** su objetivo es el de proporciona la ubicación de un objeto en particular, ya sea con el uso de un Sistema de Posicionamiento Global (GPS), o dentro de un Sistema de Posicionamiento Local (GPL). Este sistema es implementado a través de satélites que envían las señales a una unidad de control, con ella el objeto puede identificar cual es su ubicación.

  Estos sistemas pueden ser utilizados en diversos lugares como en teléfonos móviles, puntos de acceso Wi-Fi, en el interior de edificaciones, entre otros. Los datos de la ubicación pueden provenir de sensores o transmisores.

  Para determinar la ubicación de un objeto determinado, el objeto en cuestión recibe señales provenientes de diversos sensores, y estos sensores son los responsables de calcular la ubicación.

- **Datos de sensores:** las redes de sensores inalámbricos son las vías por la que los datos entran a IoT. Por medio de estos sensores se pueden monitorizar el clima, la temperatura y el ruido.

  Los sensores inalámbricos y las tecnologías de redes han hecho posible la captura de una elevada cantidad de datos de manera rápida.

- **Datos históricos:** los datos que son almacenados a través de sensores pueden ser requeridos en un futuro para realizar algún análisis o minería de datos y extraer conocimiento de ellos. Las aplicaciones de diseño deberían ser orientadas para que de forma automática analicen qué datos deberían conservarse y cómo hacerlo.

- **Modelos físicos:** los modelos físicas necesitan ser representados para poder acceder a ellos con el uso de algoritmos, cada vez que una aplicación de IoT lo requiera. Los modelos físicos son instancias de la realidad, como la gravitación, la luz, la fuerza, el sonido, el magnetismo, etc. al representar estos modelos nos permitirá modelar y simular escenarios físicos.

- **Actuadores y comandos de control de datos:** Utilizados básicamente para el control de dispositivos remotos.

2.2.3. Domótica

La domótica surge como respuesta a la necesidad que tiene el ser humano de mejorar y acomodar su estilo de vida y la tendencia de crear un espacio confortable con el más mínimo esfuerzo.

Es así como desde hace siglos, el hombre consigue descubrir máquinas que le simplifiquen su trabajo, hagan más fácil y cómodo su día a día, e incluso le permitan sacar un mayor rendimiento a su esfuerzo[10]. De esa forma llega la domótica a nuestros hogares.

No existe una fecha exacta para precisar el nacimiento de la domótica, ya que la misma fue evolucionando con el paso del tiempo, esto se lo debemos al crecimiento y desarrollo de sus ramas principales que son la informática y la electrónica.

El término domótica proviene de los años 60, en Francia, cuando se hace referencia a la palabra Domotique, "domo", del latín "domus"(casa), y el sufijo "tica"(que funciona por sí solo)[2] . y así en el año 1989 aparece por primera vez en el diccionario francés petit Larousse su definición:

Domotique: *(Del latin "domus", vivienda)"*, conceptos de habitad que integra todos los automatismos en materia de seguridad, gestión de energía y comunicación, etc.

[2]http://es.wikipedia.org/wiki/Domótica

Dicho en otras palabras y por otro autor: una casa Inteligente es aquella cuyos elementos y dispositivos están integrados y automatizados a través de una red y que por medio de dispositivos externos o internos, sus estados se pueden modificar, o los mismos dispositivos están capacitados para responder a cambios producidos en su entorno [11].

Las casas inteligentes a su vez deben estar dotadas de sistemas de inteligencia artificial, de manera tal que puedan asumir cualquier tipo de modificación de forma económica y factible y ser capaces de:

- Auto repararse en caso de fallos
- Tomar decisiones en caso de emergencias.
- Monitorizar y controlar las actividades y el funcionamiento de las instalaciones de la casa.

2.2.3.1. Servicios de la vivienda domótica

Los sistemas domóticos aportan inteligencia propia en el hogar, Algunos de los servicios más importantes que ofrece un sistema domótico se muestra en la figura 2.3 .

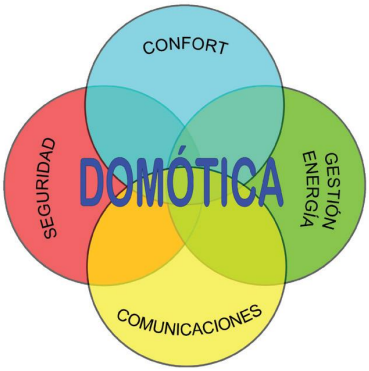

Figura 2.3: Servicios de una vivienda domótica

- **Seguridad:** Uno de los aspectos más atractivos de la domótica es el tema de la seguridad, poder estar al tanto de lo que sucede en el hogar es una tarea importante para el usuario, una vez el usuario se marcha de su vivienda se activa el simulador de presencia, y las luces se apagan

y encienden de forma automática en horarios diferentes para simular la presencia de alguien en el hogar.

Otro tipo de servicios que proporciona la domótica es la seguridad técnica: Detectores de inundación, humo y gas que en caso de activarse avisan sonoramente o mediante luces de las anomalías ocurridas, además envían un e-mail de forma automática a los teléfonos de emergencias.

- **Confort:** El confort es garantizado a través del uso de funciones que hacen posible combinar diferentes ambientes, como el control de iluminación, sistema de climatización, elementos motorizados, y sistemas de audio y video, todo esto es posible a través del uso de sensores y reguladores.

- **Ahorro energético:** La racionalización de consumo energético que se genera en las viviendas es posible a través de los dispositivos de medición que permiten controlar el nivel de consumo de energía en el hogar. Otros servicios que podemos nombrar son los sistemas de regulación de la luminosidad o sistemas de activación temporizada o detección de movimiento.

- **Monitorización:** La monitorización a distancia es otros de los servicios que ofrece la domótica, con el uso de pantallas táctiles en el interior de la vivienda, es posible detectar si existen luces encendidas, ventanas abiertas o sistemas de climatización activos. Además es posible consultar remotamente el estado de señales específicas con el uso de teléfonos móviles, internet, PDA y otros dispositivos electrónicos.

2.2.3.2. Características de un hogar inteligente

- **Integración:** Los equipos son autónomos y por ende los usuarios no necesitan estar pendiente de ellos.

- **Interrelación:** Una de las principales características que debe ofrecer un sistema domótico es la capacidad de relacionar diferentes elementos, y que estos elementos puedan comunicarse entre sí y tomar decisiones de manera autónoma, esto aportaría una gran versatilidad al sistema.

- **Facilidad de uso:** Esta característica consiste en la facilidad de uso del sistema domótico para poder maniobrarlo remotamente de forma tal que el usuario pueda acceder a él a través del ordenador y modificar cualquier componente si es necesario.

- **Control remoto:** El sistema domótico debe permitirle al usuario poder maniobrarlo a distancia, de manera tal que pueda realizar tareas como la de cambiar la temperatura del hogar, controlar el sistema de regar el jardín y monitorizar la vivienda a través del sistema de seguridad.
- **Fiabilidad:** La fiabilidad es uno de los aspectos de gran importancia, contar con equipos potentes y ágiles sería un valor añadido para un sistema domótico.
- **Actualización:** Disponer de actualizaciones que puedan ser instaladas de forma automática.

2.2.3.3. Componentes de un sistema domótico

La extensión de una solución domótica puede variar desde un único dispositivo que realiza una sola acción, hasta amplios sistemas que controlan prácticamente todas las instalaciones dentro de la vivienda.

A continuación se detallan los componentes principales que un sistema domótico debe tener:

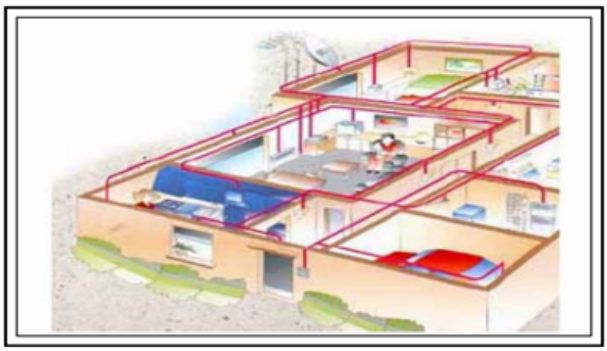

Figura 2.4: Vias de conexión para un hogar domótico [11]

Un sistema domótico puede constar de los siguientes elementos:

- **Elementos de campo:** Es el responsable de transmitir la señal a una unidad central inteligente que gestionará la información recibida.
- **Sensores:** Son los dispositivo que monitoriza el entorno, capturando información que le transmite el sistema (sensores de agua, gas, humo, temperatura, viento, humedad, lluvia, iluminación, etc).

- **Actuador:** es un dispositivo capaz de ejecutar y/o recibir una orden del controlador y realizar una acción sobre un aparato o sistema concreto (encendido/apagado, subida/bajada, apertura/cierre, etc).
- **Interfaz:** se refiere a los dispositivos (interruptores, conectores, pantallas, etc.) en los que se muestra la información del sistema para los usuarios a fin de que pueden interactuar con el sistema.
- **Controlador:** gestiona el sistema según la programación y la información que recibe, como si de un microprocesador se tratara.
- **Unidad central inteligente:** supervisa y gestiona el correcto funcionamiento de todas las partes eléctricas de la instalación, y de todos los elementos asociados a ella. Su función principal es la de recibir la señal transmitida por los sensores, detectores, captadores y actuadores y guardar la información recibida que luego utilizará para actuar sobre determinados circuitos relacionados con esas señales.

2.2.3.4. Tipos de arquitectura

Existen tres arquitecturas básicas: arquitectura centralizada, descentralizada y distribuida.

- **Arquitectura centralizada:**

  Es aquella en la que a los elementos a controlar y supervisar (sensores, luces, válvulas, etc.) están conectados a una unidad de control única, de manera tal que si este deja de funcionar afectaría todo el sistema como tal.

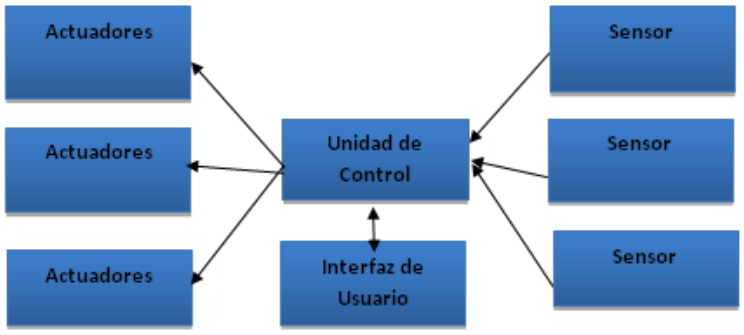

Figura 2.5: Características de una arquitectura centralizada

- **Arquitectura descentralizada:**

  Es aquella en la que el elemento de control se sitúa próximo al elemento a controlar.

  Cada elemento del sistema tiene su propia capacidad de proceso y puede ser ubicado en cualquier parte de la vivienda[11].

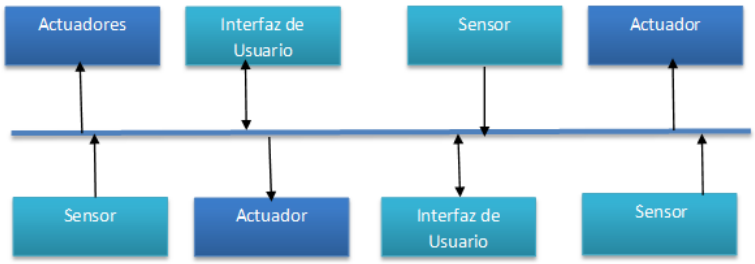

Figura 2.6: Características de una arquitectura descentralizada

- **Arquitectura distribuida:** Es una arquitectura intermedia entre el control centralizado y descentralizado. Está compuesto por un conjunto de nodos inteligentes conectados en red mediante un bus de datos. Cada nodo tiene acceso directo a una serie limitada de dispositivos de otros nodos.

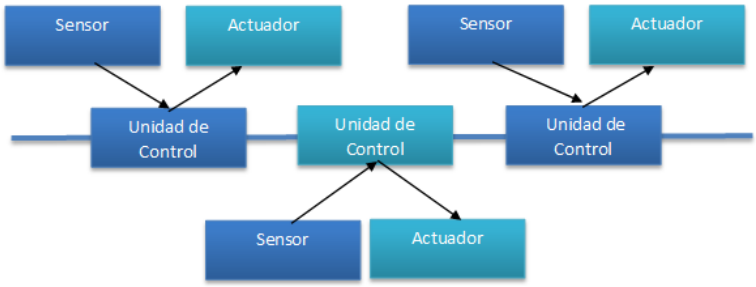

Figura 2.7: Características de una arquitectura distribuida

2.3. Evolución de las bases de datos

Antes de hablar de los orígenes de las bases de datos, primero vamos a dar una breve definición: Una base de datos es un conjunto de datos almacenados en memoria externa que están organizados mediante una estructura de datos cuyo objetivo principal es el de satisfacer los requisitos de información de una institución u organización.[12].

El término bases de datos fue escuchado por primera vez en un simposio celebrado en California en 1963 [13].

Cabe destacar que antes de que entrara en funcionamiento el uso de bases de datos en las empresas, se utilizaban sistemas de ficheros que consiste en un sistema descentralizado en el que cada departamento gestiona tanto los ficheros como los programas que lo componen de forma independiente, uno de los inconvenientes que supone el uso de ficheros es el de la duplicidad de datos comunes entre dos departamentos y el difícil acceso a los datos al encontrarse almacenados en lugares distinto.

Se dice que el origen de las bases de datos se remonta a los años 60, cuando surgen las ideas de enviar al hombre a la luna, en un proyecto estadounidense llamado APOLO; para esa época no existían sistemas lo suficientemente capaces de almacenar la elevada cantidad de información que el proyecto requería, y es cuando surge la necesidad de diseñar un sistema que pudiera suplir estos requerimientos.

NAA (North American Aviation) fue la primera empresa designada para el proyecto, sus ideas la llevaron a la implementación y el desarrollo de un software denominado GUAM (General Update Access Method) que consistía básicamente en que varias piezas pequeñas se unen para formar una pieza más grande, y así sucesivamente hasta que el producto final está ensamblado.

Esta estructura que tiene la forma de un árbol, es lo que se denomina una estructura jerárquica [12].

Posteriormente IBM (International Business Machines) se unió a NAA, para colaborar con el desarrollo del software GUAM, que luego es renombrado como IMS (Information Management System).

Década de 1950 y principios de 1960:

IBM restringe IMS al manejo de jerarquías de registro para permitir el uso de dispositivos de almacenamiento, de ahí surgen las cintas magnéticas para el almacenamiento de datos.

Con el surgimiento de las cintas magnéticas, las tareas de procesamiento de datos y administración de nóminas fueron automatizadas, sin embargo, la memoria principal de la cinta era demasiado pequeña para almacenar la cantidad de información que se generaba y solo se podía acceder a ellas de manera secuencial.

A mitad de los sesentas, General Electric desarrolló IDS (Integrate Data Store), un sistema de bases de dado dirigido por Charles Bachmann, que fue concebido con el objetivo de definir un estandard de base de datos, a esta iniciativa se unió el grupo CODASYL (Conference on Data Systems Languages) y se crea el Data Base Task Group (DBTG), grupo encargado de regir las mejores prácticas para la creación y manejo de BD.

En el año 1971, DBTG presentó su propuesta final, sin embargo, esta no fue aceptada por ANSI (American National Standards Insitute), aún así muchos sistemas se basaron en ella, a esos sistemas se conocen hoy en día como CODASYL o DBTG.

Los sistemas jerárquicos y de red constituyen la primera generación de los SGBD. sin embargo, presentan algunos inconvenientes como son:

- La complejidad con la que se debían escribir los programas de aplicaciones para responder a una sencilla consulta de datos.
- La independencia de datos es mínima.
- No posee un fundamento teórico.

Principo de los años 1970

En el año 1970, Edgar Frank Codd, miembro de los laboratorios de investigación IBM, escribió el artículo *«Un modelo relacional de datos para grandes bancos de datos compartidos»*.

Codd describe las deficiencias de los sistemas jerárquicos y de red ya existentes y propone que los sistemas de bases de datos deberían presentarse a los usuarios con una visión de los datos organizados en estructuras llamadas relaciones.[14]

Codd describe que los usuarios de grandes bases de datos no deberían preocuparse por la representación interna de los datos, de estas ideas surge el modelo relacional.

Sin embargo, a pesar de lo académicamente interesante, las ideas de Codd no fueron bien recibidas en IBM; no fue sinó a partir del año 1974, cuando un grupo de la Universidad de Berkeley en California, liderado por Michael Stonebreaker, prestó interés en ella.

Stonebreaker utilizó la ideas del modelo relacional y junto a su equipo, desarrolló un sistema al que puso como nombre *"el Ingres"*, cuya primera versión se presentó en ese mismo año, dando como resultado la creación del primer manejador relacional de base de datos funcional. Sin embargo, a pesar de lo innovadora de la propuesta, ésta también poseía sus debilidades, una de ellas era la limitada capacidad de modelar datos, esto dio como origen a un sin número de investigaciones en búsqueda de solucionar este problema.

En los años 1976, Peter Chen, presentó el modelo Entidad Relación, que permite representar el esquema de una base de datos de manera gráfica; esta propuesta fue muy exitosa, tanto así que hoy en día es una de las técnicas más utilizadas en el diseño de base de datos.

Década de 1980

A principios de 1980 las base de datos relacionales competían con los sistemas de base de datos jerárquicas y de red, tanto así que incluso, hasta llegaron a reemplazarlos.

La razón principal por la que los sistemas relacionales escalaron mucho más que los sistemas jerárquicos y de red radicaba no solo en su sencillez y buen rendimiento, sino en el hecho de que en los sistemas relacionales las tareas de bajo nivel se realizaban de manera automática, mientras que

en los sistemas jerárquicos y de red los programadores tenían que crear sus propias consultas y realizar manualmente las tareas de bajo nivel, lo que implicaba un gran esfuerzo.

La creación de *"el Ingres"* terminó despertando el interés de IBM que a su vez reaccionó poniendo en marcha en el año 1980 otro sistema basado en las especificaciones del modelo relacional, el *"System R"*.

Del System R surgen dos propuestas de desarrollo muy interesantes, como lo es el nacimiento de un lenguaje de consulta estructurado, el SEQUEL, que posteriormente pasaría a llamarse SQL (estándar actual de los sistemas relacionales) y la producción de varios SGDB relacionales (Sistemas de Gestión de Bases de Datos), como es el caso de DB2, SQl/DS. Para entonces Larry Ellison, un empresario del Valle del Silicón, a partir de las investigaciones de Edgar F.Codd crea un nuevo producto y una nueva empresa que hasta la fecha se conoce como Oracle; [14] [15] a esto se le denominó la segunda generación de los SGBD relacionales.

La tercera generación los SGBD, viene marcada por los avances de las base de datos orientadas a objetos, la extensión de las base de datos relacionales y el procesamiento distribuido.

Finales de la década de 1990

El acontecimiento de mayor envergadura que caracterizó la década de los años 90 fue sin lugar a duda el nacimiento y crecimiento explosivo del World Wide Web. Fue una época en donde las base de datos se empezaron a usar de forma masiva, en está década la implementación y el desarrollo de base de datos fue mucho más extensiva que nunca antes.

Actualidad

Hoy en día, existen cientos de SGBD relacionales, tanto para microordenadores como para sistemas multiusuario, aunque muchos no son completamente fieles al modelo relacional.

Otros sistemas relacionales multiusuario son INGRES de Computer Associates, Informix de Informix Software Inc. y Sybase de Sybase Inc. Ejemplos de sistemas relacionales de microordenadores son Paradox y Base IV de Borland, Access de Microsoft, FoxPro y R: base de Microrim.

2.3.1. Modelo Relacional

Tal como se ha tratado en el tema anterior, el modelo relacional surgió de las ideas de Edgar F. Codd, en el año 1970; en sus investigaciones perseguía diseñar un modelo mucho más flexible y sencillo que los existentes en esa época.

Codd pretendía utilizar sus conocimientos matemáticos y científicos de computación para representar de una forma intuitiva la información del mundo real, introduciendo conceptos fáciles de entender por cualquier usuario.

El modelo relacional define los metadatos que son los que mantienen la información de las características principales de las base de datos, así como también incorpora mecanismos de consultas muy eficientes y sobre todo totalmente independientes del SGBD (Sistema de Gestión de Base de Datos).

2.3.1.1. Evolución del Modelo Relacional

La figura 2.8, muestra un esquema de como ha ido evolucionando el modelo relacional y cada una de las fechas trascendentales y de mayor importancia que marcaron su crecimiento.

PRERELACIONAL		1968 - 1970	↔	Surge el modelo
		1970 . . .	↔	Desarrollos teóricos
		1973 - 1978	↔	Prototipos (Ingres, sistema R, etc. . .)
	RELACIONAL	1978	↔	QBE
		1979	↔	Oracle
		1980	↔	Ingres
		1981	↔	SQL
		1982	↔	DB2
POSTRELACIONAL		1986	↔	SQL/ ANS
		1987	↔	SQL ISO (9075)
		1989	↔	SQL Addendum
		1989	↔	Manifiesto de los SGBO
		1990	↔	Modelo Relacional Versión 2
		1990	↔	Manifiesto de los SGBO- 3G
		1992	↔	SQL 92
		1995	↔	3er Manifiesto
		1999	↔	SQL 3

Figura 2.8: Evolución del modelo relacional [16]

Los aspectos más importantes que define el modelo relacional son los siguientes:

1. **La estructura**, a través de ella se puede representar la información que nos interesa del mundo real.
2. **La manipulación**, a la que da apoyo mediante las operaciones de actualización y consulta de los datos.
3. **La integridad** que se establece mediante reglas y condiciones que deben ser cumplidas por los datos [17].

El modelo relacional lo que pretende es que el usuario perciba la base de datos como una estructura lógica formada por un conjunto de relaciones y no como una estructura física de implementación. La ventaja que ofrece este modelo es la de que los datos consiguen un alto nivel de independencia. [17]

2.3.2. Estructura del Modelo Relacional

Una base de datos relacional es una colección de relaciones [18] El modelo relacional está formado por relaciones, tuplas y atributos.

id_trabajador	nombre	tarifa_hr	tipo_de_oficio	id_supv
1235	F. Aguilera	12,50	Electricista	1311
1412	A. Calvo	13,75	Fontanero	1540
2920	N. Marín	10,00	Carpintero	null
3231	O. Pons	17,40	Albañil	null
1540	J.M. Medina	11,75	Fontanero	null
1311	J.C. Cubero	15,50	Electricista	null
3001	D. Sánchez	8,20	Albañil	3231

Figura 2.9: Tuplas, Atributos y Dominios [18]

Las filas se denominan tuplas, una cabecera de columna es un atributo y la tabla es una relación. El tipo de datos que describe los tipos de valores que pueden aparecer en cada columna se llama dominio.[19]

Ahora se puede ver con más detalle:

Un dominio D, es un conjunto de valores atómicos, en el que cada valor es indivisible con respecto al modelo relacional. Ejemplos de dominio serían los siguientes:

- Población: El conjunto total de los habitantes de un país
- Apellidos: El conjunto de apellidos de personas.
- Universidades: El conjunto total de universidades.

Cabe destacar que un dominio debe poseer un nombre, tipo de datos, y un formato establecido.

Un esquema relacional R, denotado por: $R(A_1, A_2, ...A_n)$, se compone de un nombre de relación R, y una lista de atributos, $A_1, A_2, ...A_n$. En donde cada atributo A_i es el nombre de un papel desempeñado por algún dominio D en el esquema R. R es la relación, el grado de una relación es el número de atributos, n, de su esquema de relación. [19]:

Atributo $(A_i -)$: Elemento susceptible de tomar valores.

Tupla: La estructura tupla coincide con el tipo de datos y/o registros presentes en todos los lenguajes de programación, así como en los de datos anteriores. La estructura relación es especifica del Modelo Relacional, y es la que lo caracteriza [20].

Relación:

El nombre de modelo *relacional* surge de la relación que existe entre el elemento principal del modelo y el concepto matemático de relación.

Una tabla o relación es una matríz rectangular que almacena líneas con una estructura concreta:

Un tipo de relación se define como un conjunto de pares de la forma: $\{(A_1, D_1), (A_2, D_2)....,(A_n, D_n)\}$, denominado esquema de relación [20].

Un tipo de relación de esquema:$\{(A_1, D_1), (A_2, D_2)....,(A_n, D_n)\}$,es un conjunto de tuplas de dicho esquema [20].

En la figura 1, aparece una tabla de platos de un restaurant, la primera linea identifica el nombre de la columna, el grado de la tabla viene dado por el número de campos que posee, y la cardinialidad es el número de tuplas concretas que almacena, en este caso es de 7.

Código	Nombre	Precio	Menú
15	Salomillo a la pimienta	1000	No
23	Fondue Neuchatel	1500	No
17	Migas de Chocolate	850	Si
34	Ajo blanco con uvas	850	Si
12	Paella Valenciana	1100	Si
21	Mono a la canfonesa	7500	No

Cuadro 2.1: Ejemplo de tabla

Visión informal de una relación

A continuación se muestra la representación tabular de una relación que contiene datos de empleados.

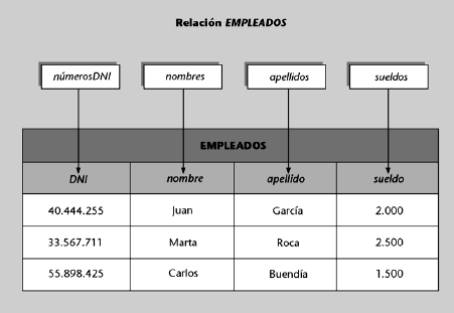

Figura 2.10: Visualización tabular de una relación [17]

La estructura de los datos del modelo relacional resulta facil y sencilla para el usuario.

Visión formal de una relación

Un Sistema de Gestión de Bases de Datos Relacional (SGBDR), tiene como objetivo apoyar las definiciones de datos, utilizando para ello la estructura de datos del modelo relacional, también es el responsable de manipular dichos datos con las operaciones del modelo, de manera tal que satisfaga las reglas de integridad establecidas por el modelo relacional.

2.3.2.1. Normalización

La normalización de base de datos es el proceso de descomponer grandes tablas en otra más reducida, con el objetivo de eliminar duplicidad y redundancia de datos. [21] En otras palabras, normalización es el proceso de transformación de datos complejos en un conjunto de estructuras de datos más pequeñas, con la finalidad de facilitar su manejo. [22].

La normalización surge para solucionar los errores que se generaban al colocar todos los datos en un solo lugar, ya que este era un proceso ineficiente y conducía a errores de lógica cuando se trabajaba con los datos.

La ventaja que ofrece la normalización es el consumo de espacio. Una base de datos normalizada ocupa menos espacio que una que no lo esté.

Grados de Normalización: La forma normal son las reglas empleadas para asegurar el correcto funcionamiento de las tablas y verificar la estructura de la misma. Existen tres niveles de normailzación y cada una a su vez posee sus propias reglas que serán empleadas par probar los errores que podrian producirse.[22][21]

- **Primera Forma Normal (1NF):** la primera Forma normal establece que las columnas que se repiten deben ser eliminadas y reubicada en una tabla diferente.

  Se dice que una relación R se encuentra en su primera forma normal, si todos los dominios subyacentes contienen solo valores atómicos.

- **Segunda Forma Normal (2NF):** la segunda forma normal contempla que se deben eliminar todas las dependencias parciales y además deben ser separadas en sus propias tablas. Entiéndase como dependencia parcial a aquellos datos que no dependen de la llave primaria de la tabla para poder identificarlos.

  Para identificar que una relación R se encuentra en su segunda forma normal primero debemos asegurarnos que ya está en primera forma normal y que además cada atributo que participa en la llave primaria depende plenamente de ella.

- **Tercera Forma Normal (3NF):** decimos que una tabla se encuentra en su tercera forma normal cuando todas las columnas que no son llaves dependen completamente de la llave primaria y no hay dependencia transitivas, una dependencia transitiva es aquella en la que existen columnas que no son llaves que a su vez pertenecen a otras columnas que tampoco son llaves.

  Una relación R, se encuentra en su tercera forma normal, si cada tupla de la relación consiste de una llave primaria que identifica a alguna entidad con un grupo de atributos independientes y que no tienen dependencia funcional.

2.3.2.2. Ventajas del Modelo Relacional

Entre las ventajas que se pueden señalar en el uso del modelo relacional podemos mencionar las siguientes:

- La independencia de datos, tanto a nivel físico como lógico.
- Establece relaciones entre diferentes tablas mediante el uso de llaves foráneas.
- Se evitan los datos duplicados, a través de la normalización con el uso de llaves primarias.

- La fácil recuperación de los datos almacenados con el uso de llaves primarias y foráneas que agilizan esta operación.
- Visualización de los datos mediante varias consultas a las Bases de Datos[21].

2.3.3. Almacenes de Datos

En actualidad, con el desarrollo de nuevas tecnologías y el elevado volumen de datos históricos que manejan las compañías, es de vital importancia el desarrollo de tecnologías de base de datos que satisfagan las necesidades de las empresas.

Una de las razones primordiales por las que se precisa la evolución de dicha tecnología, redica en la necesidad que tienen las organizaciones de extraer conocimientos a partir de la información histórica que poseen, de manera tal que puedan tomar decisiones convenientes a partir de los conocimientos extraídos, eso es básicamente lo que persiguen las tecnologías de almacenamiento de datos (ADs).

Un almacenamiento de datos se define como una colección de datos orientada a un ámbito determinado, es integrada, no volatil y variable en el tiempo, además ayuda al proceso de toma de decisiones.[23]

Los almacenes de datos manejan grandes cantidades de información que se subdividen en unidades lógicas más pequeñas dependiendo la entidad que las utilice.

En la siguiente figura se muestra como está estructurado un sistema de almacenamiento, toda la información que integran el almacén de datos viene de diferentes fuentes, tanto de origen interno como externo.

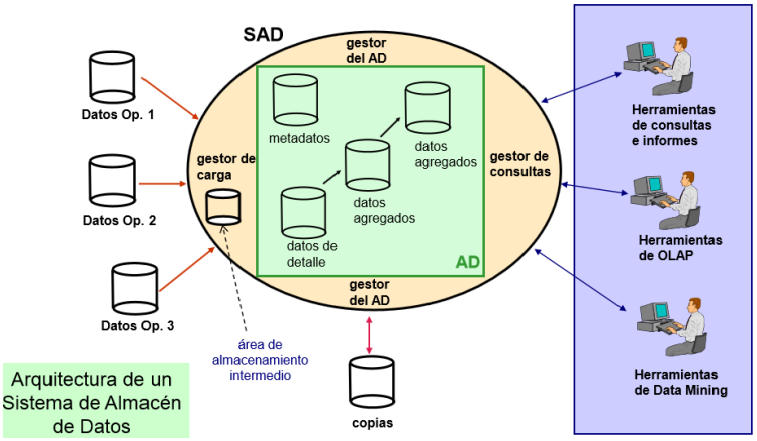

Figura 2.11: Exploración de un almacenamiento de datos [24]

Posterior al almacenamiento de datos, los datos son preparados de la forma más adecuada con el objetivo de utilizarlos para el análisis. Dicho análisis es realizado por los usuarios utilizando herramientas OLAP (On Line Analitycal Procesing) y de Minería de Datos (DM)

2.4. Minería de Datos (DM)

En la actualidad, con los avances tecnológicos que han surgido, se cuenta con base de datos que pueden almacenar una gran catidad de información, esta información a su vez necesita ser procesada para poder extraer conocimiento de ella, de esta necesidad de análisis es que surge la tecnología de Minería de Datos o Data Mining (DM).

Sin embargo, no podemos hablar de DM sin antes tratar el tema de inteligencia de Negocios o Business Intelligence.

La Inteligencia de Negocios se define como el proceso de analizar los bienes y datos acumulados en la empresa con el objetivo de extraer conocimientos de ellos[25], es una disciplina que ayuda al proceso de toma de decisiones en la empresa; BI resulta ser la solución más adecuada cuando contamos con una gran cantidad de información y a través de ella se pueden generar escenarios, pronósticos y reportes que funcionan como auxiliar para buscar las soluciones más acertadas en una institución, lo que al final se traduce en ventajas competitivas.

En la actualidad existen muchas soluciones orientados a BI, y

estas a su vez pueden ser utilizados en diferentes áreas de las empresas, tanto en mercadeo y ventas como en las finanzas. Contar con una de estas soluciones es muy beneficioso y permite que las empresas escalen notablemente.

DM es un componente esencial de BI, abarca una serie de técnicas, algoritmos y métodos cuyo fin es la explotación de grandes volúmenes de datos con vistas al descubrimiento de información previamente desconocida y que puede servir de ayuda en el proceso de toma de decisiones[26], de ahí la estrecha relación que existe entre estas dos disciplinas.

DM también se define como el proceso de descubrir patrones en los datos según citan Frank y Witter [27]

Otra definición sería la elaborada por Sumathi y Sivandam:[28]"DM es el proceso eficiente, no trivial, de extraer información valiosa(patrones y tendencias) de una gran colección de datos."

En resumen, todos las definiciones coinciden en el punto de que el objetivo primordial de la minería de datos es la extracción de conocimiento valioso a partir de los datos.

Las principales razones que impulsaron el crecimiento de DM se describen a continuación:

- Las nuevas necesidades de análisis de grandes volúmenes de datos.
- Gran parte de la información es histórica.
- El notable crecimiento de la web en estos últimos tiempos.
- La competitividad entre las organizaciones.
- El surgimiento de una gran variedad de software y herramientas para llevar a cabo DM.
- La necesidad de predecir las información futura.
- La toma de decisiones en las empresas se basan en datos históricos.

DM se nutre de la siguientes disciplinas: aprendizaje automático, base de datos, estadística, gestión de organizaciones, teoría de la decisión, visualización.

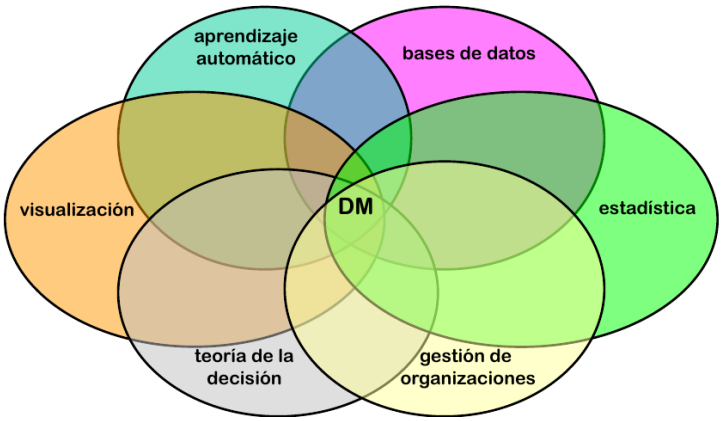

Figura 2.12: Asociación de las diversas disciplinas relacionadas a MD[29]

2.4.1. Aplicaciones de Minería de Datos

A continuación se detallan algunas de las áreas en las que DM tiene mucha utilidad:

- **Aspectos climatológicos**: para predecir cambios climáticos, como tormentas, huracanes, etc.
- **Medicina:** encontrar la probabilidad de una respuesta satisfactoria a un tratamiento médico.
- **Mercadotecnia:** identificar el tipo de clientes que responder a ofertas de productos y servicios recibidos vía correo electrónico, fidelidad de clientes, afinidad de productos, etc.
- **Inversión en casas de bolsa y banca:** analizar si a un cliente se le puede conceder un préstamo y a que determinado monto de crédito.
- **Detección de fraudes y comportamientos inusuales:** telefónicos, seguros, en tarjetas de crédito, evasión fiscal, electricidad, etc.
- **Industria:** estimaciones de composiciones óptimas de una mezcla, detección de piezas dañadas, etc.
- **Análisis de canastas de mercado** para mejorar la organización de tiendas, segmentación de mercado.

  Cabe destacar que estas son solo algunas de las tantas disciplinas con las que DM se relaciona.

2.4.2. Proceso de Extracción de Conocimiento a partir de bases de datos

La minería de datos es una etapa dentro de un proceso mayor llamado extracción del conocimiento de Bases de Datos (Knowledge Discovery from Data Bases o KDD), KDD se define como el descubrimiento de conocimiento a partir de las Bases de Datos.

Usama Fayyad lo define como el proceso no trivial de identificar patrones válidos, nuevos, potencialmente útiles y en ultima instancia comprensible a partir de los datos.[30]

Proceso de KDD: El proceso de descubrimiento de conocimiento en base de datos consta de diferentes fases:

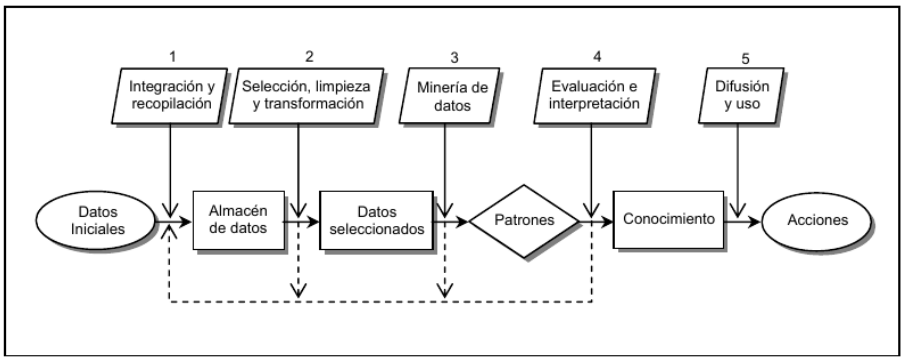

Figura 2.13: Proceso de Minería de Datos [26]

1. **Integración y recopilación de datos:** Esta es una de las fases más difíciles y que más tiempo requiere en KDD, en ella se determina que fuentes de investigación deben utilizarse para la investigación.Una de las dificultades que surgen en este paso es sin lugar a duda la diversidad de datos que pueden ser recopilados y el hecho de que los datos provienen de diferentes orígenes como: electrónica, académicos, clínicos, científicos, demográficos, etc.

2. **Selección, limpieza y transformación:** Es el proceso de seleccionar y filtrar los datos que sean relevantes para DM, además se eliminan datos redundantes y ruidosos.

3. **Minería de Datos:** En este paso se elige cuales técnicas son las más adecuadas para ser utilizadas y que estén acorde con los objetivos del

proyecto. Estas técnicas seleccionadas son aplicadas a los datos previamente preparados, que luego se validan con su respectivo dominio.

4. **Evaluación e interpretación:** Los patrones resultantes del proceso anterior son interpretados y evaluados para observar e identificar patrones significativos. Se pueden utilizar técnicas como la *validación cruzada:* que divide los datos en dos conjuntos (*datos de prueba y datos de entrenamiento*); *matrices de confusión*: para identificar las clasificaciones que se han realizado de forma correcta, entre otras.

5. **Difusión y uso del conocimiento:** En esta etapa se procede a incorporar el conocimiento extraído en un sistema o procedimiento para su difusión y uso de los usuarios finales Los resultados deben ser evaluados para evitar errores en la presentación del conocimiento.

2.4.3. Principales características y objetivos de la Minería de Datos

- Explorar los datos que se encuentran almacenados en base de datos o almacenes de datos que contiene información histórica de la empresa.
- La forma en la que se almacenan los datos puede variar, puede ser que se encuentren consolidados en un almacén de datos o puede ser que permanezcan en servidores de internet o intranet.
- El término de minería se refiere a que los software y herramientas de extracción, ayudan a extraer el mineral de la información guardada en archivos privados o registros públicos.
- El entorno de DM en la mayoría de los casos se basa en una arquitectura Cliente-Servidor.
- Hurgar y sacudir los datos ayuda al descubrimiento de resultados valiosos e inesperados.
- Las herramientas de DM se combinan con facilidad y pueden ser analizadas y procesadas con rapidez.

2.4.4. Modelos y tareas de Minería de Datos

En DM los datos son analizados para extraer conocimiento valioso, para ello recurre a modelos que le sirvan como soporte para encontrar reglas o patrones y relaciones inferidas [29].

Los modelos de DM pueden ser *descriptivo* y el *predictivo*.

2.4.4.1. Modelos Descriptivos

Con el uso del modelo descriptivo se identifican patrones que explican o resumen un conjunto de datos, proporciona información sobre las relaciones entre los datos y sus características. Mediante este modelo se identifican patrones que explican o resumen el conjunto de datos, siendo estos útiles para explorar las propiedades de los datos examinados. Los modelos descriptivos siguen un tipo de aprendizaje *no supervisado*, es decir, adquiere conocimiento desde los datos disponibles, sin requerir influencia externa que indique un comportamiento deseado al sistema [28]

2.4.4.2. Modelos Predictivos

El modelo predictivo responde a preguntas sobre datos que desconocemos y es utilizado para hacer una estimación de los valores futuros de variables de interés basándose en la información histórica de los datos, para predecir el comportamiento de los datos, ya sea mediante categorización o regresión o de clasificación. Este modelo sigue una arquitectura *supervisada* este tipo de arquitectura aprende por medio del control de un supervisor que determina la respuesta que desea generar el sistema.

2.4.4.3. Tareas de Minería de Datos

De los modelos descriptivos y predictivos detallados en el apartado anterior, surgen tareas diversas como son: reglas de regresión, clasificación, asociación, etc. Cada una de ellas está orientada a un problema en específico en DM.

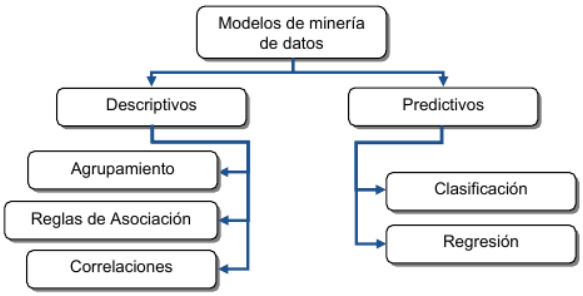

Figura 2.14: Representación general de los modelos y tareas de minería de datos

- **Asociación y dependencia:** La relación entre dos atributos surge cuando la frecuencia de que se de dos valores determinados de cada uno conjuntamente es relativamente alta.
- **Correlaciones:** Analizan las relaciones bivariantes o multivariantes entre atributos numéricos.
- **Agrupamiento:** también conocido como clustering, sirve para detectar grupos o conjuntos de individuos, en él no se conocen ni las clases ni sus números, y su objetivo primordial es determinar grupos o cluster diferenciados del resto.

- **Clasificación:** la clasificación se define como el esclarecimiento de una dependencia, en la que el atributo dependiente puede tomar un valor entre varias clases ya conocidas.
- **Regresión:** El objetivo primordial de la regresión es predecir los valores de una variable a partir de la evolución sobre variables continuas generalmente el tiempo.

CAPÍTULO 3

ARQUITECTURAS DE SGBD

Por sus características, una base de datos es uno de los elementos primordiales en todo sistema de información; a diferencia de otros medios de almacenamiento, como los archivos, las bases de datos poseen estructuras y componentes internos que organizan los datos de manera eficaz, el acceso a estos datos es permitido a través de un lenguaje de consultas que puede ser de dos tipos: SQL (Structure Query Languaje), o NoSQL(Not only Structure Query Languaje). Las bases de datos son administradas por un motor de base de datos o un sistema de administración de bases de datos.

En el año 1997, Michael Lesk escribío un artículo titulado *"¿Cuánta información hay en el mundo?"*[31], de acuerdo con su análisis, la cantidad de información que existe en el mundo tomaría varios millones de petabytes de almacenamiento. Los diferentes SGBD serían los responsables de almacenar y gestionar toda esta información de madera adecuada.

Hoy en día, la importancia que tiene el uso de BD es cada vez más incuestionable, las mismas son utilizadas en todo tipo de organizaciones: tanto con propósitos académicos, como en entidades comerciales, y almacenan toda clase de información. Los SGBD deben afrontar ciertos desafíos para gestionar de manera eficiente este volumen masivo de información y extraer conocimiento de estos repositorios, existen arquitecturas de diferentes tipos, diseñadas con características específicas para facilitar las tareas de almacenamiento y gestión de los datos.

3.1. Centralizada

Una base de datos centralizada es aquella que almacena grandes cantidades de datos en un solo lugar físico o nodo central. En una arquitectura Cliente-Servidor, los datos son accedidos por los usuarios a través de terminales que solo muestran resultados.

Las principales características de una base de datos centralizada se detallan a continuación:

- **Control Centralizado:** en una base de datos centralizada existe un nodo central que es el responsable de procesar todas las operaciones. Sus principales componentes son: los datos, los dispositivos de almacenamiento secundario asociados a este y el software de gestión de bases de datos.
- **Redundancia:** la redundancia en base de datos consiste en que un mismo dato se encuentra repetido en más de un sitio a la vez, en los sistemas centralizados la redundancia es prácticamente nula, ya que existe un único almacenamiento de datos y aunque haya una redundancia temporal mientras se trabaja con el dato en memoria, en el momento en que éstos son guardados al disco duro la redundancia desaparece.
- **Estructuras físicas compleja y acceso eficiente:** las bases de datos centralizadas permiten utilizar estructuras complejas como índices secundarios, cadenas de archivos internos, etc. La finalidad de estas estructuras complejas es optimizar el acceso a los datos.
- **Integridad, recuperación y control de concurrencia:** aunque estas tres características se refieren a problemas distintos, están fuertemente relacionadas. El control de concurrencia es lo que permite que un sistema trabaje de manera eficiente, en sentido de que controlan ejecuciones de transacción que operan concurrentemente y que acceden a la misma información. Los sistemas que gestionan de forma correcta el problema de la concurrencia, consiguen que los usuarios no se vean afectados por la existencia de otros usuarios.

  Si el control de concurrencias no está bien gestionado, se producirían problemas de integridad y recuperación, ya que las acciones de un usuario podrían afectar a la sesión de otro usuario.

- **Privacidad y Seguridad:** en la base de datos tradicionales, el administrador de base de datos es el responsable de definir los permisos y roles que gestionan el acceso a los datos, por esa razón la seguridad está determinada por el propio SGBD.

Los SGBD centralizados, tienen las siguientes desventajas:

- Si el sistema de base de datos falla, se perdería la disponibilidad y procesamiento de la información que posee el sistema.
- Difícil sincronización para su recuperación.
- No es posible distribuir la carga de trabajo entre varios nodos.

3.2. Distribuida

Bases de datos distribuidas: este tipo de base de datos tiene una particularidad que la diferencia de las demás, y es que los datos están almacenados en distintas máquinas interconectadas entre sí, formando parte del sistema.

Cada uno de los procesadores que integran dicho sistema reciben el nombre de localidad o nodo, por lo que toda la información se encuentra distribuida en diferentes espacios lógicos, a diferencia de una base de datos centralizada cuya característica principal es que la información se almacena en un solo nodo; hay que destacar que a pesar de que los datos se encuentran distribuidos, el conjunto de nodos que forman parte del sistema representan una única base de datos.

Las principales características de una base de datos distribuidas son las siguientes:

- **Control distribuido:** en una base de datos distribuida existen varios nodos que trabajan en paralelo, de forma tal que cada uno es responsable de procesar una parte de las operaciones.
- **Reducción de redundancia:** la redundancia es inherente a un sistema distribuido, ya que para aumentar la fiabilidad del sistema, se duplica la información entre varios nodos, de manera tal que si uno de los nodos falla, otro nodo puede ocupar su posición sin que se pierdan datos.

- **Estructuras físicas complejas y eficiencia del acceso:** las mismas estructuras que se utilizan en los sistemas centralizados para optimizar las búsquedas, se implementan en los sistemas distribuidos, sin embargo, requieren un grado más de complejidad, ya que deben asumir el hecho de que la información se encuentra distribuida entre varios nodos, por lo tanto, el acceso a la información es un poco menos eficiente, ya que requiere un paso más en el proceso de recuperación.
- **Integridad, Recuperación y Control de Concurrencia:** la concurrencia presenta los mismos problemas de gestión en los sistemas distribuidos que en los centralizados, sin embargo, asegurar la integridad y recuperación es más complejo, ya que al haber información redundante, habría que asegurarse que esté actualizada toda la información, y en caso de que no sea así, evitar que dos usuarios accedan a dos versiones diferentes del mismo dato.
- **Privacidad y Seguridad:** en las bases de datos distribuidas, la gestión de seguridad, se hace de la misma forma que la centralizada, mediante roles, permisos y usuarios, pero hay un punto más para controlar, ya que los distintos nodos se comunican entre si, y hay que cerciorarse de que esta comunicación sea segura.

3.3. Federada

La arquitectura federa fue introducida por primera vez por Denis Heimberge en el año 1982 [32], está basada en el intercambio coordinado de información organizada por modelos que describen conceptos y comportamientos comunes. Es una colección de diversos sistemas de base de datos cooperativos y autónomos en la que existe una interfaz común, que es utilizada por los usuarios para acceder al sistema.

Las Base de Datos Federadas (BDF) están formadas por varios esquemas unificados; cada uno describe una porción de la base de datos que aparenta ser una sola, sin embargo, es una colección de sistemas independientes, heterogéneos y autónomos. Una BDF aparenta ser una BD normal y corriente, pero no tiene existencia física, es una vista lógica.

Debido a su naturaleza distribuida y heterogénea, esta arquitectura es adecuada para ser utilizada en escenarios con problemas de complejidad causados por ambientes heterogéneos, como en IoT, donde hay una enorme

cantidad de dispositivos u objetos, que ofrecen servicios muy diversos y localizados en diferentes lugares.

Hay que destacar que una base de datos federada, es diferente de una distribuida, por que a pesar de que ambas arquitecturas comparten ciertas semejanzas,(como por ejemplo el hecho de que los datos está repartidos en diferentes ubicaciones, y que los nodos están interconectados a un nodo central), sin embargo su diseño lógico es diferente, un SGBD distribuido es de tipo Top-down, es decir, existe una sola base de datos distribuida en diferente localidades y que se rigen por las mismas reglas, por lo tanto, carecen de independencia, además los usuarios la perciben como una única base de datos, aunque la misma esté distribuida; mientras que un SGBD federado es de tipo Botton-up, es decir un conjunto de nodos autónomos e independientes interconectados a un nodo central con estructura.

3.4. Peer to Peer

Peer to Peer (P2P) es es un tipo de arquitectura descentralizada y distribuida con la característica principal de que no cuenta con un servidor central, por ende los nodos se pueden comunicar entre si, intercambiando información sin la necesidad de una unidad central. La ventaja principal de utilizar P2P radica en que al no existir una autoridad central única, (como es el caso de una red centralizada), el sistema no tiene que lidiar con los problemas que este acarrea, (como por ejemplo que la red no esté disponible porque el nodos central haya colapsado, etc.,) además existe un mejor aprovechamiento de recursos (ancho de banda, capacidad de almacenamiento, etc.)[33]

Características de una arquitectura Peer to Peer

- **Escalabilidad** una arquitectura P2P tiene mayor escalabilidad que una arquitectura centralizada, porque al nó poseer un nodo central, elimina los inconvenientes de congestionamiento de red y cuellos de botellas, permitiendo que los recursos del sistema sean mejor aprovechados, Además agregar nuevos nodos no representa ninguna dificultad.
- **Robustez** La naturaleza distribuida hace que un sistema Peer-to-Peer sea robusto, ya que las peticiones no han de pasar por un nodo central y si un nodo falla, otros nodos pueden ocupar su lugar, por lo que los fallos del sistema son muy bajos.

- **Descentralización** estas redes por definición son descentralizadas y todos los nodos son iguales. No existen nodos con funciones especiales, y por tanto ningún nodo es imprescindible para el funcionamiento de la red.
- **Seguridad** en una arquitectura P2P, se utilizan mecanismos de seguridad para las comunicaciones entre los nodos.

CAPÍTULO 4

MODELOS DE DATOS PARA SGBD

Un modelo de datos es el responsable tanto de determinar el tipo de estructura lógica que será utilizada en una base de datos, como de establecer el modo en el que los datos serán organizados, almacenados y estructurados. Para la definición de dicha estructura el modelo de datos utiliza lenguajes para hacer consultas que pueden ser de dos tipos:

4.1. SQL

El lenguaje SQL (Structured Query Language) surge en la década de los años 60, momentos en que Edgar Frank Codd presentaba su propuesta del modelo relacional, su primera versión fue publicada por el organismo ANSI en la década de los 80s, y a partir de ella han surgido muchas otras versiones. [20]

Desde su concesión, el lenguaje SQL se ha extendido bastante por ser un lenguaje declarativo muy similar al lenguaje natural y además ha sido impulsado por dos compañías importantes como lo son IBM y Oracle.

Concepto: SQL es un lenguaje para hacer consulta utilizado como estándar predeterminado para los Sistemas de Gestión de Bases de Datos Relacionales (SGBDR), cuya función principal es la de gestionar de manera adecuada las operaciones de actualización, creación de esquemas y modificaciones en el sistema relacional.

El lenguaje SQL está compuesto por varios elementos:

- **Cláusulas:** elementos incluyentes en las consultas y declaraciones, en ocasiones opcionales.
- **Predicados:** son los que especificas las condiciones u/o expresiones lógicas que pueden ser evaluadas, como es el caso de los valores booleanos y de verdad utilizados para limitar los efectos de preguntas y declaraciones o para cambiar el flujo del programa.
- **Expresión:** formado por filas y columnas de datos.
- **Consultas:** permite la recuperación de los datos tomando en cuenta criterios definidos.
- **Declaraciones:** sirven para controlar las transiciones, el flujo de programas, las conexiones y sesiones, [23]

Conceptos fundamentales SQL al ser un estándar relacional, muchos de sus conceptos pertenecen al modelo relacional. Sus principales conceptos son los siguientes: [20]

- **Instrucciones SQL:** es el mecanismo que utilizan los usuarios para poder interactuar con la base de datos, dichas instrucciones pueden ser tanto interactivas, como embebidas
- **Tablas:** en el lenguaje SQL, las tablas se definen como una colección de filas en donde la presencia de filas idénticas es posible.
- **Vista:** una vista es una representación virtual de una tabla, el nombre de virtual viene dado porque no se almacena físicamente. La declaración de una vista es permitida realizando una consulta que contiene su estructura y la forma en que sus filas son derivadas, tomando en cuenta las tablas de la base de datos.
- **Esquema y Catálogo:** en SQL, los objetos que se crean pertenecen a un esquema, y el esquema a su vez agrupa una colección de objetos. El nombre completo de un objeto viene dado por el nombre calificado por el esquema.

- **Conexión, sesión y transición:** se dice que una conexión es la asociación entre un cliente SQL y un servidor de base de datos.

  Una Sesión es el contexto en el cual un usuario ejecuta una secuencia de instrucciones SQL, utilizando para ello una conexión.

  Una transición se define como una unidad lógica de procesamiento, formada por una secuencia de instrucciones SQL.

4.2. NoSQL

Debido al notable crecimiento que han tenido los sistemas informáticos en las últimas décadas atribuido al desarrollo de nuevas aplicaciones y recursos web, el volumen de datos asociados a estas aplicaciones también ha crecido. Un ejemplo de ello, es el motor de búsqueda google, cuyo volumen de datos alcanza el nivel de petabyes, 10^{15} bytes para ser más exactos [34]. En escenarios como estos, en donde se manejaba una gran cantidad de información, los SGBD relacionales se mostraban poco escalables e ineficientes; para afrontar estos problemas del modelo relacional fueron buscadas soluciones alternativas, una de ellas fue la de aumentar el número de servidores, sin embargo, esta solución no resultaba tan sencilla de aplicar por lo estructurado que es un SGBDR.

Las bases de datos NoSQL surgieron como alternativa para solucionar los problemas del modelo relacional; hay que destacar que el propósito de NoSQL no es el de sustituir el modelo relacional, más bien, surge como alternativa para escenarios en donde se requería una arquitectura mucho más flexible y menos estricta que la del modelo relacional.

El término NoSQL fue utilizado por primera vez en el año 1998, expresión alusiva a las base de datos que omiten el uso de SQL [35]

La función principal de los sistemas NoSQL es la gestión y el almacenamiento de datos semiestructurados [1], cada uno de estos procesos se hace de forma independiente, por lo que los datos no necesitan tener un modelo predefinido ni encajar en tablas relacionales, lo que hace que el sistema sea más flexible en escenarios con ambientes heterogéneos.

Un poco de historia:

[1] Un dato semiestructurado es un tipo de datos que no poseen definición de tipos, ni conceptos de variables o atributos, y no están organizados mediante un patrón determinados.

En el año 2004, Google, en búsqueda de diseñar un sistema que le proporcionara mayor disponibilidad y escalabilidad, lanza el primer SGBD basado en NoSQL, a este sistema se le conoce con el nombre de "BigTable"; posteriormente, ya para los años 2007 y 2008, surgieron dos propuestas más en esa dirección, una de ellas sería la del SGBD Dynamo que es una base de datos no relacional para servicios web de Amazon, y CASSANDRA, un sistema distribuido desarrollado por Facebook para manejar elevados volúmenes de datos. Más adelante, CASSANDRA fue también implementado en Twitter. Entre los años 2009 y 2010 fueron diseñados dos SGBD orientadas a documento: Apache CouchDB y MongoDB, estos sistemas tenían la característica de que almacenaban los datos sin la necesidad de seguir un esquema predefinido, y es así como se han extendido el uso de SGBD de tipo No SQL. [34]

Tabla comparativa de las bases de datos relacionales y NoSQL:

	BD Relacional	BD NoSQL
Escalabilidad	Más complejo debido a su naturaleza estructurada	La principal ventaja de las BD NoSQL es que poseen una estructura más flexible.
Consistencia	Se podría decir que la consistencia es uno de los puntos más fuertes del modelo relacional. Como las reglas son muy estrictas, la información es consistente.	No garantiza consistencia de información, en caso de que la información no sea actualizada, retornará el mismo valor.
Disponibilidad	Este modelo tiene problemas de disponibilidad dado su dificultad para distribuir los datos.	Otro punto fuerte de BD NoSQL, es que posee un alto grado de distribución de datos y garantiza un mayor número de solicitudes.

Figura 4.1: Comparación de SQL y NoSQL

Para analizar la eficiencia de las base de datos SQL y NoSQL, en el artículo [36] se realizó una comparativa de varios sistemas de base de datos SQL y NoSQL.

Sistemas basados en SQL:

- **PostgreSQL:** es una base de datos tradicional de archivo abierto, de tipo SQL, que existe desde el año 1995, y es una representación del modelo SQL.

Sistemas basados en NoSQL

- **CASSANDRA:**es un sistema de archivo abierto, de tipo SQL, diseñado para trabajar con un elevado volumen de datos distribuida en diferentes servidores, provee una estructura de tipo Clave-Valor, utilizando el mecanismo de consistencia eventual.
- **MONGODB:** en un SGBD de tipo SQl, y provee una estructura de tipo Clave-Valor, para gestionar documentos basados en BSON(JSON binario)

La prueba consistían en analizar el rendimiento de cada uno de estos sistemas al gestionar datos provenientes de sensores, para ello se utilizó un servidor físico y varias máquinas virtuales que se ejecutaban en el mismo servidor. Cada una de las pruebas medía el periodos de tiempo que tardaban cada una de los sistemas para escribir los datos en la BD o para leer los dados almacenados en ella. Y estos fueron los resultados obtenidos:

	Multi Write		Multi Read	
	Single Client	Multi Client	Single Client	Multi Client
Cassandra				
- physical	120,000	95,000	13,000	69,000
- virtual	53,000	79,000	11,000	560,000
MongoDB				
- physical	47,000	41,000	99,000	98,000
- virtual	41,000	26,000	64,000	83,000
PostgreSQL				
- physical	27,000	73,000	130,000	410,000
- virtual	24,000	55,000	95,000	360,000

Figura 4.2: Número de operaciones por segundo

En las operaciones basadas en escritura, los SGBD de tipo NoSQL(MongoDB y CASSANDRA), presentaron mejor rendimiento que el SGBD de tipo SQL. esto se debe a que el primer grupo, poseen una estructura mucho más flexible que los de tipo SQL, tal como hemos mencionado en el apartado anterior. Sin embargo podemos notar que en las operaciones de lectura, los sistemas de tipo SQL tienen un mejor rendimiento, esto se debe a que su estructura de datos es estricta y por ende permite la localización de los datos de manera más rápida que los de tipo NoSQL.

Ventajas de NoSQL frente a SQL:

Evitar complejidad innecesaria: las base de datos relacionales proporcionan una serie de funcionalidades relacionadas con la consistencia de

datos (el llamado ACID) que las base de datos NoSQL no implementan, lo que hace que el rendimiento sea mayor al reducir el numero de comprobaciones que tiene que hacer el sistema.

Alta Escalabilidad: añadir y eliminar servidores a una base de datos NoSQL es mas fácil porque no se requiere ninguna configuración adicional en el resto de servidores.

Evitar el mapeo a objetos: muchas aplicaciones trabajan con una estructura de datos muy simple (únicamente una lista de pares clave/valor), por lo que no se benefician del modelo relacional que proporcionan las base de datos tradicionales.

Sacrificar fiabilidad frente a rendimiento: en algunas situaciones puede ser interesante sacrificar la fiabilidad para ganar rendimiento.

Bases de datos + capa de cache versus sistemas creados desde cero pensando en alta escalabilidad: cuando empezaron a crecer los requerimientos de las base de de datos se creó lo que se llama una capa de cache, que permite aumentar la capacidad de carga de las base de datos. El problema es que este sistema es un parche que está llegando ya a su límite de capacidad, por eso se plantea utilizar un sistema que se haya creado desde cero y que desde su planteamiento sea diseñado para proporcionar un alto rendimiento/escalabilidad.

CAPÍTULO 5

RETOS A LOS QUE SE ENFRENTAN LOS SGBD TRADICIONALES PARA SU USO EN IOT

En el año 1997, Michael Lesk, publicó un artículo titulado: *"How much information is there in the world?"*[31], *("¿Cúanta información hay en el mundo?")*, Lesk intentaba ofrecer varias estimaciones sobre la cantidad de información existente en el mundo real, para ello, tomó en consideración las cifras y datos sobre la web, capacidad de almacenamiento de discos, contenido multimedia (Música, video, fotos. etc...), librerías, etc.

Luego de una exhaustiva investigación Lesk había llegado a la conclusión de que existían unos miles de petabytes de información en el mundo y que para los años futuros estas cifras aumentarían, ya que seríamos capaces de almacenar todo tipo de información.

Si para el año 1997, momentos en que el internet aún no estaba al alcance de todo el público y cuando aún no existían las tecnologías de IoT la cantidad de información existente llegaba a miles de petabytes, en la actualidad y con el amplio crecimiento que ha habido a nivel tecnológico esta cifra ha sido superada notablemente.

Estudios realizados en el artículo *"IoT, En un mundo conectado de objetos inteligentes"* [6] indican que en el año 2010 existían cerca de 3,000 millones de etiquetas RFID, ya para el año 2020 se estima que el universo digital será cuarenta y cuatro veces más grande que en el año 2009. Esto indica que las estimaciones y conclusiones a las que Lesk había llegado no

podían ser más acertadas.

Si para la actualidad resulta complicado para los SGBD gestionar toda esta información, tan solo imaginemos en un futuro en donde todo tipo de objeto forme parte de internet y la cantidad de información se multiplique.

Empresas importantes como Google y Microsoft han mostrado preocupación por el tema y han decidido invertir miles y millones de dólares para aumentar el número de servidores en todo el mundo. La pregunta a la que se debe dar respuesta es: ¿Será suficiente la solución de aumentar servidores? o ¿Existe la necesidad de reestructurar todo el sistema y diseñar nuevas tecnologías de Bases de datos? .

En esta sección se tratará el tema de las limitaciones y los desafíos que deben afrontar las base de datos tradicionales para poder cumplir las exigencias de IoT.

5.1. Tamaño

En IoT se ha adoptado como un estándar de facto el uso de Tags RFID para la identificación de los objetos, de forma que cualquier dato que se genera en IoT viene asociado con el RFID del objeto al que pertenece.

Una red IoT consta de una gran cantidad de objetos, conectados a una serie de sensores que está generando información de forma continuada, todo este flujo de información debe ser tratado y gestionado para ser almacenado en una base de datos para usos futuros.

Si por ejemplo implementamos una red IoT, que mediante Tags RFID y sensores GPS se va registrando de forma continuada la posición de todos los objetos que conforman la red, seria necesario crear en la base de datos una tabla para almacenar la información que genera la red, esto es: RFID, Timestamp y posición. Los requisitos de almacenamiento de esta tabla serian:

Columna	**Tipo de dato**	**Bytes necesarios**
RFID	CHAR(128)	128 Bytes
Ts	TIMESTAMP	4 Bytes
Latitud	DECIMAL(9,6)	5 bytes
Longitud	DECIMA(9,6)	5 bytes
	Total	*142 Bytes*

Cuadro 5.1: Requisitos de capacidad

Hay que tener en cuenta, que aunque no parezca mucho 142 Bytes por registro, se están generando de forma continuada muchos registros. Si suponemos que los sensores están actualizando la posición cada segundo, cada objeto estaría generando cerca de 4 gigabytes de información cada año. Si bien este volumen de datos si que es asequible por la mayoría de SGBD actuales, algunos podrían empezar a tener problemas si estuviéramos tratando del orden de miles o decenas de miles de objetos.

Según el artículo *"Sensors empower the 'Internet of Things' "*[1], se estima que para los próximos diez años habrá una media de 1000 sensores por persona, por lo que para una pequeña población de 1000 habitantes se estarían generando 40 petabytes de información cada año, y con la velocidad actual de los sistemas de almacenamiento serian necesarios 50 discos duros trabajando en paralelo de forma continua durante las 24 horas del día para poder almacenar toda esta información, y cada año aumentará el numero de sensores por persona, por lo que el ratio de almacenamiento sería cada vez mayor.

Hay otros problemas relacionados con el tamaño, como lo es la limitación física de la capacidad de ancho de banda de la conexión, o las necesidades de procesamiento de datos antes de ser insertados en la base de datos.

También habría que tener en cuenta la indexación de los datos, con un volumen tan elevado de objetos y sensores es necesario diseñar métodos de indexación adecuados para facilitar la identificación de un objeto en concreto.

Un sistema centralizado tendría más dificultad para lidiar con estos problemas que uno distribuido, ya que un único nodo seria el responsable de gestionar toda la información, mientras que uno distribuido repartiría la carga de trabajo entre todos sus nodos. Sin embargo ambos sistemas llegarían a su limite de capacidad cuando se estuviera tratando con el volumen de datos que generaría una red IoT a gran escala[2].

5.2. Tipo de datos e integración

Características de los datos

- **Heterogéneos y de diferentes fuentes:** los datos provienen de diferentes dispositivos (Tag RFID, cámaras, lectores, sensores de temperatura, etc.), cada uno con sus propias características y estándares, de hecho no es posible ser conscientes de todos los tipos de datos que se van a gestionar en el sistema, ya que en un futuro pueden añadirse nuevos tipos de objetos y sensores que no estaban previstos en un principio.
- **Correlación Espacio-Temporal:** a diferencia de los datos tradicionales, en IoT todos los datos que se generan llevan asociada una ubicación y una marca de tiempo.
- **Interoperabilidad:** en su concepción original, todas las redes IoT deben estar interconectadas, para lograr esto las redes IoT no pueden ser independientes, y se necesitará que interoperen entre sí y compartan información para facilitar el trabajo colaborativo, de forma que un objeto de una red puede moverse libremente a cualquier otra.

En el ejemplo anterior, se ha creado una tabla para almacenar la información que generaban los objetos sobre su posición, pero ¿Qué pasaría si se desea almacenar la información de la temperatura?, la tabla que se utilizó en el ejemplo para la posición ya no sería útil, se tendría que crear una tabla nueva para almacenar temperatura; del mismo modo se tendría que actuar cada véz que se registraran otras propiedades. Más aún, la base de datos está esperando recibir la posición en un formato específico(grados en formato decimal), pero ¿Qué ocurre cuando se intenta añadir al sistema IoT un objeto nuevo que envía la información de su posición en otro formato?(por ejemplo en formato NMEA)

Una posible solución seria ampliar la tabla para incluir una columna para cada tipo de registro que pueda entrar en la base de datos, de esa forma el sistema podría admitir varios tipos de datos, pero seguiría estando limitado en el caso de que aparezca un nuevo tipo de dato que no se había previsto. Ademas se estaría desaprovechando mucho espacio, ya que en toda la fila de la base de datos solo se estaría utilizando uno de los campos, mientras que el resto estarían vacíos.

En resumen, este sistema, tal y como se está planteando no seria dinámico y no podría adaptarse a cambios, y este es uno de los desafíos principales a los que se enfrentan las BD, la heterogeneidad de los datos que producen los diferentes tipos de objetos en IoT.

Para solucionar estos problemas la mejor solución sería definir un estándar de forma tal que todos los objetos utilicen el mismo formato de datos a la hora de enviar información, esto solo resolvería el problema del formato en el que la base de datos recibe los datos, pero se seguiría teniendo el problema de rigidez a la hora de aceptar nuevos tipos de datos.

Pero se puede ir mas allá y definir un formato estándar que admita cualquier tipo de datos, presentes o futuros, incluso aquellos en lo que todavía no se ha pensado. La solución ideal se basaría en el uso de XML para almacenar la información en la base de datos.

El Sistema de Gestión de Bases de Datos Oracle incluye el tipo de datos XMLType desde la versión 9i y permite utilizar el lenguaje SQL para hacer consultas dentro de la estructura de los XML almacenados. Por eso se considera que la mejor opción es utilizar este tipo de datos para definir este estándar.

Si para el anterior ejemplo del sistema que almacena las posiciones de los objetos en vez de tener dos campos para almacenar latitud y longitud, hubiese un campo XMLType, la información que se almacena podría tener por ejemplo este formato:

```
<iotdata
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceShemaLocation=
        "http://localhost:8080/source/schemas/xsd/iotdata.xsd">
        <type>gps</type>
        <data format="LatitudeLongitude">
                <latitude>-0.361511</latitude>
                <longitude>39.407785</longitude>
        </data>
</iotdata>
```

De esta forma todos los objetos mandarían la información de su posición a la base de datos encapsulada dentro de un XML. Si ahora en el sistema entrara un nuevo objeto cuyo funcionamiento es distinto a todos los existentes y la

forma de publicar las coordenadas es con un formato distinto, este objeto debería seguir mandando su información a la base de datos con un XML y seria el XML el que indicaría que el formato utilizado es diferente, tal como se aprecia en el siguiente ejemplo:

```
<iotdata
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceShemaLocation=
        "http://localhost:8080/source/schemas/xsd/iotdata.xsd">
        <type>gps</type>
        <data format="NMEA">
                $--GLL,-0.36151,N,39.407785,E,203110.10,A,M,*20
        </data>
</iotdata>
```

Ahora, en el momento en que se decida empezar a registrar una nueva propiedad de los objetos, no será necesario crear una nueva tabla en la base de datos ni modificar la existente, ya que la información que se guarda seguirá siendo en formato XML:

```
<iotdata
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceShemaLocation=
        "http://localhost:8080/source/schemas/xsd/iotdata.xsd">
        <type>temperature</type>
        <data format="fahrenheit">
                20
        </data>
</iotdata>
```

Y ahora, si se quisiera hacer una consulta sobre la base de datos, se podría utilizar la función *"extractValue"*

```
SELECT
        RFID,
        ts,
        extractValue(data, '/iotdata/data/latitude') AS Latitude,
        extractValue(data, '/iotdata/data/longitude') AS Longitude
FROM
        RawData
WHERE
        extractValue(data, '/iotdata/type') = 'gps'
        AND extractValue(data, '/iotdata/data/@format') = 'LatitudeLongitude'
```

En la consulta se están seleccionando los valores de latitud y longitud que hay en el campo XML de los registros que tienen datos de tipo GPS, y en formato LatitudeLongitude.

Uno de los inconvenientes que tiene esta solución propuesta consiste en que se aumenta considerablemente el espacio requerido para almacenar la información.

En la propuesta inicial solo se necesitaba 10 bytes para registrar la latitud y la longitud, con el nuevo formato harían falta 275 bytes, esto es casi 30 veces más de espacio.

Otro inconveniente sería que se aumentarían los requerimientos de procesamiento a la hora de hacer consultas en la base de datos, ya que no se están haciendo consultas directamente sobre campos de la tabla, sino sobre valores que hay dentro del XML del campo.

El elegir XML como base para sugerir este estándar es por el hecho que existen SGBD que ofrecen soporte nativo a este tipo de datos (como Oracle), pero también seria factible utilizar otro sistema, como puede ser JSON o BSON si existiera alguna base de datos que diera soporte; de hecho, en SGBD de tipo NoSQL orientados a documentos, se basan precisamente en esta idea, ya que utilizan un campo para identificar el documento y otro campo para almacenarlo, estando este en formato XML, JSON o BSON, dependiendo del SGBD.

5.3. Lenguaje de consultas

Como se ha dicho en el punto anterior, uno de los problemas es la heterogeneidad de los datos, y se ha propuesto como solución utilizar documentos XML (o JSON, o BSON) para almacenar la información, sin embargo, esto ocasionaría un nuevo problema, y es que el lenguaje que existe actualmente para hacer consultas (SQL) no está preparado para gestionar este tipo de documentos.

Oracle si que da soluciones de forma nativa a este problema, ya que dispone de métodos (como por ejemplo ExtractValue) para realizar consultas sobre documentos en formato XML, sin embargo, esto es una solución particular adoptada por un único SGBD, por lo que esta extensión del lenguaje para hacer consultas, es exclusivo de Oracle y no funciona en el resto de SGBD.

El problema aquí es que se necesita definir un estándar de lenguaje para hacer consultas sobre datos semi-estructurados. Por las propias características que tienen los datos en IoT, se pueden identifican tres tipos de consultas:

- **Consultas Históricas:** para buscar datos sobre un objeto en un instante de tiempo. Este tipo de consulta permite recuperar el estado en el que se encontraba un objeto en un momento dado.
- **Consultas de Trazas:** permiten obtener los valores que a lo largo del tiempo ha tomado un objeto. En este tipo de consultas se obtendría un listado de valores ordenados por tiempo que corresponderían a la evolución que ha sufrido el objeto.
- **Consultas Espaciales:** son consultas relacionadas con la posición de los objetos. de ella se establecen dos sub-tipos: El primero consiste en dada una posición se obtiene la información de todos los objetos ordenados por distancia. El segundo consiste en, dados dos objetos, se obtiene la evolución de la distancia entre ellos a lo largo de un periodo de tiempo determinado.

Es importante identificar estos tres tipos, ya que se deben tener en cuenta a la hora de definir el estándar para las consultas.

El área en la que más avances se han realizado es en las consultas sobre XML, existiendo actualmente varias propuestas de lenguaje de consultas sobre documentos de tipo XML, entre ellos los que más éxito han tenido son:

- **XQL99:** se basa en la navegación jerárquica por los documentos, es el único que no permite especificar el resultado.
- **XMLQL:** es un lenguaje para consultas que utiliza sintaxis XML.
- **XSLT:** es un lenguaje de transformación de árboles XML.
- **Lorel:** se creó originalmente para hacer consultas sobre base de datos orientas a objetos.
- **XQuery:** en 2001, el W3C publicó el primer borrador y se pretende que este sea considerado el estándar para consultas sobre XML. Está basado en todos los anteriores, pero presenta algunas limitaciones que si que estaban resultas,

por ejemplo, no soporta todos los tipos de datos especificados en el esquema XML.

	XQL99	XMLQL	XSLT	Lorel	XQuery
Modelo de datos	Árbol	Grafo	Árbol	Grafo	Nodos
Respuesta múltiple	Sí	No	Sí	No	Sí
Consultas de manipulación	No	No	Sí	Sí	No
Operador Like	No	No	No	Sí	Sí
Búsqueda por similitud	Sí	Sí	No	No	No

Cuadro 5.2: Características de los lenguajes para hacer consultas sobre XML

5.4. Transacciones y mantenimiento del estado:

El sistema estándar en base de datos para procesamiento de transacciones es conocido como ACID que se define como un conjunto de propiedades necesarias para que un conjunto de instrucciones puedan ser consideradas como *"transacción"*.

ACID posee cuatro propiedades principales (Atomicity, Consistence, Isolation, Durability), las cuales de detallan a continuación:

- **Atonomicity:** define que el cambio producido por una transacción es atómico, es decir, todas las operaciones de una transacción se realizan o ninguna de ellas se lleva a cabo. Esto tiene sentido, cuando trabajamos con base de datos complejas, en las que para hacer una operación hay que realizar cambios en varias tablas simultáneamente. Sin embargo, en IoT, los procesos de almacenamiento consisten de una única instrucción.
- **Consistency:** establece que solo los valores o datos válidos serán escritos en la base de datos y que cada vez que se realice algún cambio de estado en el sistema, se registrará ese valor, mientras no se vuelva a cambiar ese dato.
- **Isolation:** esta propiedad asegura que no sean afectadas entre sí las transacciones, esto quiere decir que cuando dos transacciones se ejecutan concurrentemente e intentan actualizar/leer un mismo dato, el sistema debe evitar que se produzcan inconsistencias. Pero en IoT no se va a dar el caso de que dos transacciones modifiquen un mismo dato a la vez, porque para empezar, la modificación no esta contemplada, los sensores están continuamente insertando nuevos registros, y los que ya están no se modifican, por lo que para IoT no hace falta que la base de datos implemente "Isolation",

- **Durability:** define que una vez finalizada una ejecución, sus resultados son permanentes. La durabilidad es uno de los aspectos de mayor importancia en los SGBD, en escenarios donde son utilizados varios nodos, si uno de estos falla se perdería toda la información de ese nodo sino se cuenta con esta propiedad.

  Algunos sistemas tradicionales con durabilidad replican la información, de forma tal que si un nodo falla se tiene una copia de los datos en otros nodos y no se pierde la información. El problema es que la naturaleza de los objetos en IoT es dinámica y los sensores producen información de manera continua y en tiempo real, por lo que grabar todos estos datos en varios lugares harían que el sistema funcione muy lentamente.

  Lo que proponen algunas base de datos NoSQL(Not only SQL) como MongoDB es relajar la restricción de la durabilidad, haciendo una replicación de la información de los nodos en diferido, por ejemplo una vez al día se copia toda la información de los nodos a un sistema de backups, de manera tal que si falla un nodo, no se perderían todos los datos que este tenía, sólo la parte que se ha almacenado desde el ultimo backup, por lo que a pesar de sacrificar un poco la durabilidad, garantizaría un mayor rendimiento.

  Es una situación de riesgo/beneficio. ¿Cuál es la probabilidad de que un nodo falle? ¿Merecería la pena perder un tanto por ciento de los datos si consigo duplicar la velocidad?. Todo dependerá de cuáles sean nuestras necesidades y que sería más apropiado para nuestro sistema.

En el año 2002, se publicó el teorema CAP o Teorema de Brewer que afirma:

En cualquier sistema sólo podemos tener dos de las siguientes tres propiedades, pero no las tres a la vez:

- **Consistencia:** En el mismo sentido que se define la consistencia de ACID.
- **Disponibilidad:** El sistema está disponible y responde en un tiempo adecuado a los clientes.
- **Tolerancia a Fallos:** Un sistema tolerante a fallos sigue funcionando aunque alguno de su servidores se caiga.

Eso quiere decir, que si se aseguran dos de las propiedades, se tiene que sacrificar la tercera, por ejemplo: Una forma de lograr que un sistema sea tolerante a fallos, es teniendo muchos nodos, de forma que si uno falla, están el resto para sustituirlo, si ahora se desea que el sistema sea también consistente, cada vez que hagamos una escritura, se debe hacer en todos los nodos a la vez y no dar por finalizada la operación hasta que el proceso se termine, lo que por supuesto está

haciendo que el sistema vaya más lento y por tanto eliminando la disponibilidad. Si se quiere aumentar la disponibilidad, la operación debe terminar antes, es decir, antes de que termine de actualizarse la información en todos los nodos, lo que provocaría consistencia eventual(durante un período de tiempo los datos no serian consistentes)... y así con todas las propiedades, en el momento en que se logra satisfacer dos de las propiedades, se elimina la tercera.

Como en IoT, una de las principales características que requieren las Bases de Datos, es la disponibilidad, debido a la gran cantidad de información que deben registrar en tiempo real, y apoyándose en el teorema CAP, los desarrolladores de Bases de Datos han optado por no implementar ACID en las Bases de Datos NoSQL, para así lograr mayor rendimiento, sacrificando las propiedades que ofrece ACID.

En cotraposición a ACID, los sistemas NoSQL, suelen implementar BASE:

- **Basically Available:** el sistema garantiza la disponibilidad en el mismo sentido que lo define el teorema CAP.
- **Soft state:** el estado del sistema puede cambiar aun sin ninguna entrada (debido a la consistencia eventual).
- **Eventual consistency:** habrá instantes en los que el sistema no será consistente, pero, eventualmente llegará a un estado consistente.

ACID	BASE
Consistencia fuerte	Consistencia débil
Aislamiento entre transacciones	Datos obsoletos están "bien"
Se centra en el "Commit"	La disponibilidad es lo más importante
Disponibilidad restrigida	
Bloqueo pesimista	Bloqueo optimista
Evolución compleja	Evolución fácil
	Más simple
	Más rápido

Cuadro 5.3: ACID Vs BASE

5.5. Seguridad y protección de los datos

El tema de seguridad ha sido uno de los retos que han tenido que afrontar los SGBD desde sus inicios; para el año 2012, según un informe realizado por la

compañía Verizon [37], el 96 por ciento de los datos sustraídos provenían de bases de datos, de los cuales unos 242 millones resultaron potencialmente comprometidos, si en la actualidad las políticas y protocolo existentes resultan ser insuficientes y en muchos casos vulnerables, ¿que será en un futuro cuando millones de objetos con características ubicuas e inteligentes formen parte de Internet?, los requisitos de seguridad que deberán suplir los SGBD tendrán que ser más estrictos en un escenario de IoT.

Tan solo se debe imaginar el alto riesgo que representa el hecho de que grandes volúmenes de datos provenientes de sensores, y que contienen información personal llegara a mano inapropiadas, esto representaría un tesoro valioso que podría utilizarse para fines maliciosos, por ende, se requiere el uso de políticas de seguridad más severas para proteger el almacenamiento, procesamiento y trasmisión de los datos, y prevenir el acceso no autorizados y operaciones ilegales con la información almacenada, en otras palabras, los sistemas deben garantizar confidencialidad y proponer nuevos métodos de autenticación eficientes para facilitar la seguridad de los nodos que se comuniquen entre sí y proporcionarle al usuario la transparencia necesaria para saber quién tiene acceso a sus datos y para que fines los utiliza.

Sin embargo, hay que destacar que la misma política de gestión de seguridad de datos no se ajusta a todas las situaciones, por lo tanto debe haber políticas sobre el manejo de diversos tipos de datos generados de manera dinámica y heterogénea. El desarrollo de este tipo de políticas de gestión de datos no es sencilla, de hecho, requiere el uso de una serie de reglas alineadas con la legislación sobre protección de datos, que a su vez podría cambiar.

Las principales propiedades de seguridad que un SGBD debe suministrar para IoT son:

- **Confidencialidad:** que el sistema sea capaz de garantizar el almacenamiento seguro de datos provenientes de sensores, y prevenir acceso no autorizados.
- **Autenticación:** que los nodo se identifiquen antes de participar en el intercambio de información.
- **Integridad:** garantizar que la información no sea eliminada sin autorización.
- **Disponibilidad:** garantizar que todos los nodos estén trabajando y proveyendo mejores servicios.
- **Actualización:** que los datos que se envían así como también los datos que se reciban estén actualizados.

Ataques más comunes en bases de datos:[37]

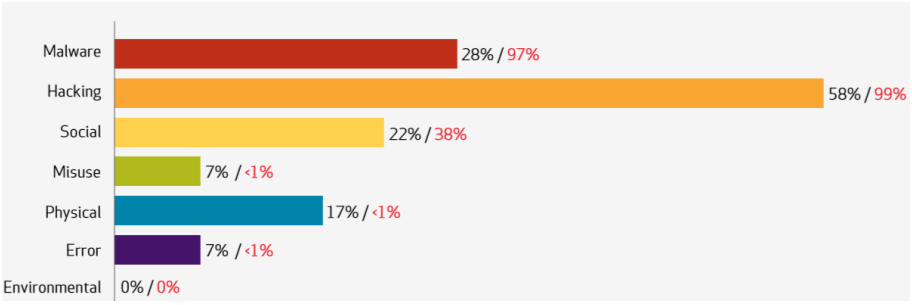

Figura 5.1: Porcentaje de ataques por categoría[37]

- **Malware:** es un tipo de software malicioso utilizado o diseñado con el propósito de poner en peligro cualquier archivo de información sin el consentimiento de su propietario.
- **Hacking:** es un tipo de ataque que persigue violar los mecanismos de seguridad, con el objetivo de obtener acceso no autorizado a la información almacenada en una base de datos. Este tipo de ataques por lo general, es realizado en un punto remoto, permitiéndole al atacante el beneficio de anonimato.
- **Inyección por SQL** un ataque de esta clase está diseñado para permitir al atacante el acceso no autorizado a una base de datos sin ningún tipo de restricción para que obtenga control total sobre la base de datos.
- **Denegación de servicio (DoS):** ataque diseñado para negar el acceso a las aplicaciones de red o datos a los usuarios previstos. Por lo general, este ataque consiste en enviara peticiones repetidas a las bases de datos, de modo que estas se sobrecarguen y no puedan responder a un usuario auténtico.
- **Social:** los atacantes utilizan tácticas sociales (engaño, manipulación, intimidación) con el propósito de robar los datos.
- **El mal uso:** El uso de recursos para los propósitos para los que no fueron diseñados.
- **Física:** Consiste en el robo de dispositivos Hardware para extraer información importante.

Tal como se mencionaba, la mayoría de los ataques son dirigidos a los servidores, por tal motivo la seguridad en un SGBD es imprescindible.

Con la cantidad de objetos que formaran parte de IoT, y la cantidad de usuarios que estarán en constante comunicación con la red, los SGBD deben

reforzar sus políticas de seguridad para evitar ser afectados por los ataques antes mencionados.

5.6. Procesamiento de datos

Como se ha expuesto en puntos anteriores, los sensores de los objetos en IoT generarán un gran volumen de información, si se toma en cuenta que un usuario tendrá al menos 10 objetos inteligentes conectados a Internet generando una gran cantidad de información que será enviada a los SGBD de manera constante, los SGBD deben estar preparados para poder asimilar de manera adecuada este gran volumen de información generado en tiempo real.

Para comprobar la habilidad que poseen las base de datos actuales para registrar y hacer consultas de los datos generados en tiempo real, el proyecto Digital Environment Home Management System (DEHEMS), realizó algunos experimentos en esta dirección [38].

El experimento consistía en el desarrollo de un sistema de rastreo de consumo energético en los hogares, se tomó como prueba unas 250 viviendas para monitorizar los niveles de energía consumidos con el propósito de hacer minería de datos y permitirle al usuario acceder a la información sobre sus patrones de consumo.

La información que se generaba de la lectura de los sensores era gestionada por un SGBD para determinar cual sería el más factible.

Entre los factores estudiados de los SGBD se encuentran los siguientes:

- La capacidad para comunicación cuando entra al sistema un gran volumen de datos.
- La capacidad de registrar grandes volúmenes de datos generados de forma constante.
- La capacidad de almacenaje a largo plazo.

Para analizar la capacidad de inserción, se creó un generador de registros aleatorios que trabajaba a un ritmo predefinido, de forma tal que se pudieran hacer distintas pruebas a distintos ratios de inserción.

Las base de datos evaluadas fueron las siguientes:

Bases de datos de tipo SQL:

- **Oracle BerkeleyDB**[1]**:** es un motor de base de datos para sistemas embebidos, el sistema mediante el cual organiza los datos es utilizando árboles-B, que son árboles balanceados en el que los nodos pueden tener más de dos hijos. El coste de inserciones, borrados y búsqueda es logarítmico, por lo que es una buena opción para solucionar los problemas de rendimiento ocasionados por un gran volumen de datos.
- **MySQL**[2] es un SGBD de código libre, es multiusuario y multihilo. Está pensado para entornos con alta carga y para sistemas embebidos.
- **PostgreSQL**[3] es una base de datos orientada a objetos, altamente escalable en el número de usuarios concurrentes y en la cantidad de datos que puede manejar, sus capacidad en cuanto a tamaño de base de datos, tabla, fila e índices son muy altos en relación con otros SGBD.
- **SQLiTe3:** [4] es un SGBD autocontenido que no requiere un servidor ni configuración.

Bases de datos de tipo NoSQL

- **IBM Informix**[5]**:** dispone de un módulo(TimeSeries DataBlade) que está pensada específicamente para trabajar con los datos provenientes de una red de sensores.
- **MonetDB**[6]**:** esta basado en tablas de asociación binaria y es una base de datos orientada a columnas, ofrece un alto rendimiento para tablas grandes.
- **Hypertable**[7] **:** es un sistema de almacenamiento distribuido, orientado a columna, con un alto rendimiento, diseñado para manejar y procesar la información de un gran cluster de servidores. Permite una alta escalabilidad y es muy resistente a fallos.

La prueba que se realizó consistía en hacer consultas de recuperación e inserción aleatorias con una frecuencia determinada. Se plantearon tres escenarios distintos: uno con 50,000 usuarios, otro con 100,000 y el último con 250,000, de forma que las inserciones por segundo eran de 83,340, 166,670 y 416,000.

Y estos fueron los resultados obtenidos:

[1] `http://www.oracle.com/technology/products/berkeley-db/index.html`
[2] `http://www.mysql.com/`
[3] `http://www.postgresql.org/about/`:
[4] `http://www.sqlite.org/about.html`:
[5] `http://www-01.ibm.com/software/data/informix/`
[6] `http://monetdb.cwi.nl/projects/-monetdb/MonetDB/index.html`
[7] `http://hypertable.org/about.html`

Table 2: Overall results for large-scale tests

Database system	Average inserts (thousands operations)			Deviation – inserts – (thousands operations)		
	50 000	100 000	250 000	50 000	100 000	250 000
BerkeleyDB	120.44	120.22	120.13	44.64	44.58	45.07
Hypertable	260.79	264.80	262.32	52.12	55.27	54.46
Informix	73.65	10.99	73.51	3.64	33.32	3.64
MonetDB	10.71	10.64	10.78	0.577	0.613	0.571
MySQL	4.67	4.29	4.49	7.30	7.02	7.11
PostgreSQL	44.90	45.66	45.89	26.19	26.27	26.47
SQLite3	18.61	18.43	18.40	8.75	8.71	8.65
	Average scans (thousands operations)			Deviation –scans – (thousands operations)		
	50 000	100 000	250 000	50 000	100 000	250 000
BerkeleyDB	1760	1760	1.730	24.78	23.80	19.47
Hypertable	22.37	22.58	22.56	1.60	1.67	1.66
Informix	136.79	117.30	133.16	0.776	0.526	0.743
MonetDB	135.89	135.84	135.99	6.35	5.91	5.94
MySQL	*273.94*	*279.58*	*277.19*	*6.17*	*5.20*	*6.40*
PostgreSQL	*62.22*	*61.43*	*60.72*	*17.06*	*17.29*	*17.37*
SQLite3	281.12	280.78	280.03	1.92	2.67	2.28

Figura 5.2: Análisis de SGBD

De esta tabla de resultados se puede concluir que ninguno de los sistemas probados da el mejor rendimiento tanto en la lectura como en la escritura, por ejemplo, Hypertable es el que mejor rendimiento tiene en inserciones(262,320 operaciones por segundo en el escenario con 250,000 usuarios), sin embargo, es de los que peores resultados da en las lecturas (22,560 operaciones por segundo para el mismo escenario).

El criterio para decidir que SGBD seleccionar, es saber cual es la finalidad que se le va a dar, ya que por ejemplo, para IoT lo que de pretende es priorizar las inserciones. De los SGBD comparados, los dos que se consideran una opción válida para utilizar en IoT, serían en primer lugar Hypertable, ya que pese a su bajo rendimiento en las consultas, ofrece muy buen rendimiento en inserciones, que es el punto a priorizar, mientras que las consultas, al ser un proceso que se ejecutará a un ratio muy por debajo del que se ha utilizado en las pruebas, no es preciso un rendimiento tan alto. La segunda opción seria IBM Informix, ya que, aunque tiene un ratio menor de inserciones (aunque sigue siendo más alto que el de la mayoría), está más equilibrado con las consultas, pero lo que de verdad lo hace una opción atractiva es el módulo que integra para trabajar con datos provenientes de una red de IoT.

Esta propuesta puede ser utilizada como punto de partida para otros estudios que tengan una aplicación directa a los retos a los que se enfrentan las base de datos a IoT. La principal conclusión que se extrae de este estudio es que las base de datos actuales no están preparadas para trabajar con el volumen de datos que genera IoT, por lo que es necesario estudiar soluciones alternativas.

Una de las posibles soluciones sería trabajar con un sistema distribuido de base de datos de forma que cada nodo del sistema sería capaz de recibir una parte del volumen de información que se genera.

CAPÍTULO 6

SOLUCIONES PROPUESTAS PARA IOT

6.1. Infraestructura orientada al dominio

Domain Name System (DNS) es un sistema jerárquico orientado a recursos conectados a Internet y otros servicios, cuyo objetivo primordial es el de traducir nombres de dominios adecuados para los equipos conectados a la red, proporcionándole una dirección IP de manera tal que se puedan localizar y direccionar cada dominio para facilitarle al usuario la tarea de navegación.

De la misma manera que el DNS es una parte fundamental en Internet, lo será el servicio de nombres para Internet de Objetos (IO), de hecho es la clave para que IoT sea posible. El servicio de nombres se puede utilizar para traducir nombres de "Cosa" que pertenezcan a espacios de nombres heterogéneos, en sus correspondientes direcciones IP, es decir serviría para designar un nombre único a cada objeto que forme parte de IoT.

A este tipo de servicios en IoT se le denomina Things Name Services (TNS). en TNS el objeto "cosas" de IoT (basado en una red TCP/IP o restringido), puede comunicarse con facilidad con otro objeto en la misma red o en cualquier otra por el nombre que tiene asignado, sin tomar en cuenta su dirección IP.

Para lograr que TNS pueda suplir los requisitos de IoT, se necesita analizar varios aspectos importantes, como lo es la eficiencia de la red restringida, la compatibilidad de los espacios de nombres heterogéneos y la protección de la privacidad de este tipo de servicios.

Una de las topología propuesta en esta dirección se basa en una es-

tructura jerárquica; en la mayoría de los casos, en el mundo de IoT, gestionar la información de ámbito local (ciudad, edificio, etc), requiere el diseño de una infraestructura que refleje el origen nativo de los objetos para su interacción con otros objetos.

Por lo tanto, este diseño de arquitectura propuesto presenta muchas similitudes con el DNS, que como se menciona en el inicio del capítulo, es un sistema jerárquico que utiliza base de datos distribuidas para computadoras, servicios o cualquier recurso conectado a Internet.

En general tenemos una jerarquía de dos niveles; el primero es el nivel de dominio o de red, y el segundo nivel es el que incorpora los recursos que interactúan con el nivel de dominio o de red.

En esta topología, el registro y mapeo de las entidades y los recursos se hace de manera distribuida; designando la autorización a un Servidor de Directorio(SD) en cada dominio en particular. Esta información es luego utilizada para saber a que SD se debe suscribir.

Los SD autorizados son los responsables de sus respectivos dominios, y a la vez pueden asignar otros SD para que sean responsables en sus sub-dominios.

Este mecanismo hace que la infraestructura sea distribuida y tolerante a fallos, evitando la necesidad de tener que realizar consultas y actualizaciones a un único registro centralizado.

El espacio de directorio de dominio consiste en un árbol formado por otros directorios del dominio. Cada nodo u hoja en el árbol tiene cero o más registros de recursos o entidades que contienen información relacionada con el directorio de dominio. El árbol se sub-divide en zonas, empezando por la zona raíz. Una zona debe tener un único dominio, o puede consistir en muchos dominios y sub-dominios.

Localización de un recurso:

Este sistema de directorio de dominio se mantiene por un sistema de base de datos distribuida, que utiliza el modelo cliente-servidor.

Los nodos de esta base de datos son los SD. Un SD puede ser un Servidor Maestro: que es el que almacena las copias originales de todos los registros de la zona o un Servidor Esclavo: usa un mecanismo de actualización automática en comunicación con su maestro para mantener una copia idéntica de los registros maestros.

En el nivel superior, hay un indexador maestro que mantiene una actualización de toda la información disponible de los directorios de menor nivel. El indexador físico maestro esta formado por varios servidores con la misma información, si hay que realizar modificaciones, son hechas únicamente por el indexador

maestro, mientras que el resto de indexadores esclavos replican la información, de manera tal que solo se permita el acceso a modo consulta para evitar congestionamiento e inconsistencia de información en el indexador maestro.

De esta forma mantiene la ubicación, información y posición de los recursos en cualquier momento. El indexador maestro se actualiza con los cambios que ocurren en el sistema global, para proveer la respuesta correcta y mantener la información más reciente del recurso en cada momento.

El directorio maestro sirve como soporte al SD, al intercambiar información necesaria para saber que, donde y como alcanzar los datos almacenados en el directorio deseado. Este es el propósito principal de indexador maestro. El descubrimiento se ejecuta cuando se conoce la información deseada, pero no su localización ni los recursos provistos por esa información. El indexador maestro resolverá esta tarea ofreciendo la localización del SD.

Para comprender mejor tenemos el siguiente ejemplo:

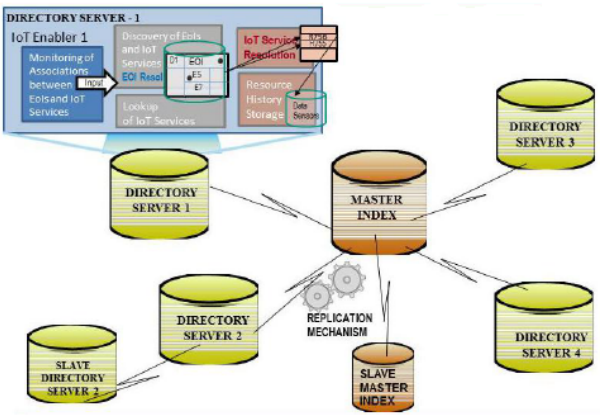

Figura 6.1: Catálogo maestro. Estructura de los servidores de directorio[39]

Características de una arquitectura orientada al dominio

- **Disponibilidad:** En una arquitectura orientada a dominio, los datos son distribuidos en forma de árbol, Esto significa que cada nodo es responsable de redireccionar la solicitud por todo el árbol, Por eso si un nodo falla, afectará todo el sub-árbol que dependa de él. Si el fallo ocurre en un nivel más alto, mayor sera el número de nodos que afectará, lo que significa que los nodos de más alto nivel son un cuello de botella.
- **Fiabilidad:** Una arquitectura orientada al dominio es tan fiable como su componente menos fiable y el índice de fiabilidad, tiempo transcurrido entre

fallos, etc., dependerá de la estabilidad de cada servidor de dominio. Como punto positivo, al ser el sistema subdividido y distribuido, un fallo en un servidor de dominio no afectará a los otros.

- **Escalabilidad:** En la infraestructura orientada al dominio, los objetos son enlazados a los servidores de directorios; Los SD son luego organizados en jerarquías en árbol y los nodos hojas contienen la información de cada objeto y sus servicios. Cada nuevo nivel representa una nueva división de los nodos padres, de esta forma la estructura no tiene limitaciones de capacidad ya que el árbol puede crecer todo lo que sea necesario para dar cabida a nuevos objetos. Sin embargo el indexador maestro contiene un índice con todos los objetos de la red, por lo que una vez más se convierte en el cuello de botella en el sistema, debido a la gran cantidad de datos que almacena.

6.2. Sistemas distribuidos

Las razones por las que un ambiente centralizado no se ajusta a IoT se menciona a continuación:

- En primer lugar, si se toma en cuenta que la información proviene de distintas fuentes, realizar minería de datos en un sistema centralizado y sobre datos distribuidos sería muy complejo.
- En segundo lugar, la cantidad de información que genera IoT es muy elevada y requiere ser procesada en tiempo real, por lo que haría falta un nodo central con una capacidad de cálculo demasiado grande.
- En tercer lugar, por temas de seguridad y privacidad no es recomendable almacenar toda la información en un único lugar
- Y por último, enviar toda la información de los sensores provocaría sobrecarga en la red.

Por todo esto una arquitectura distribuida no solo evitaría todos esos problemas, sino que también reduciría la complejidad global del sistema.

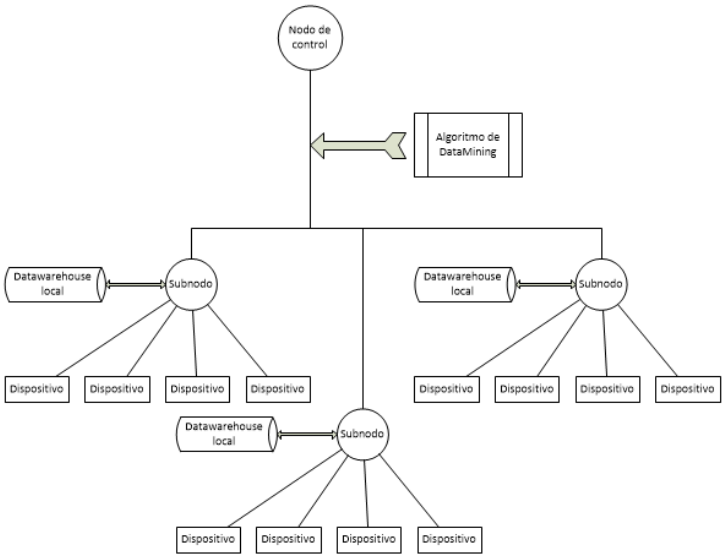

Figura 6.2: Arquitectura de un sistema distribuido para IoT

El funcionamiento de este sistema seria el siguiente: un nodo central se encarga de seleccionar el algoritmo de minería de datos a utilizar y se comunica con los sub-nodos que contengan la información que se necesita. Los sub-nodos recaban la información de los dispositivos y la pre-procesan, almacenando el resultado de este procesamiento en un datawarehouse local. Cada sub nodo utiliza el algoritmo proporcionado por el nodo central para consultar sobre su datawarehouse local y devuelve el resultado al nodo central que se encarga de integrar toda la información.

6.3. Tablas hash distribuidas

Una infraestructura Peer-to-Peer(P2P) es un tipo de organización distribuida siguiendo el patrón Cliente-Servidor, en la que el servidor ofrece como servicios sus funciones que son consumidas por el cliente.

Una de las ventajas de usar esta infraestructura es que los roles de Cliente-Servidor puedan ser asumidos por el mismo nodo en la red, eliminando así las barreras entre ambos.

El P2P mantiene una arquitectura no centralizada y el reto está en el diseño de mecanismos de localización y enrutamiento.

Actualmente existen tres enfoques:Peer to Peer puro, el método de Napster y el método basado en Tablas Hash Distribuidas. De todas ellas, la más

apropiada es la basada en Tablas Hash Distribuidas, ya que es la que da mejores resultados de rendimiento para IoT. Peer to Peer puro producirá demasiado tráfico en la red, ya que el método de localización se hace por difusión, y el método Napster utiliza un servidor central que indexa todos los objetos que se convierte en un punto crítico en el sistema.

El sistema basado en Tablas Hash Distribuidas, es similar al método Napster ya que dispone de un índice de búsqueda para localizar los objetos, la diferencia está en que este índice se haya distribuido entre todos los nodos de la red. De esta forma se elimina el problema que tenía el método Napster en el que si fallaba el servidor central dejaba de funcionar el sistema de búsqueda.

Cada objeto del sistema solo almacena una parte de la tabla Hash, para evitar que los requisitos de memoria y procesado sean demasiado elevados, la información que se almacena es de la forma de un par clave-valor, en donde la clave es el código Hash y el valor la dirección del nodo relacionado con la clave. Como cada registro de esa tabla se encuentra en varios objetos de la red, no hay problema en que alguno de estos desaparezca (Porque ha abandonado la red, o por un fallo).

El funcionamiento es el siguiente:

- Cada objeto es responsable de almacenar la Tabla Hash de un rango de código determinado.
- Cada objeto también almacena en su porción de Tabla Hash el enlace a los objetos próximos a él (Por próximos se hace referencia a los objetos que almacenan porciones de Tablas Hash de un rango próximo al objeto propio).
- Cuando llega una petición a un objeto, éste realiza la búsqueda en su porción de Tabla Hash.
- Si no lo encuentra, redirige la petición a alguno de sus objetos cercanos cuyo rango sea más próximo al criterio de búsqueda.
- La petición de búsqueda va saltando de objeto en objeto hasta que llega a su destino. El coste medio de las búsquedas es logarítmico.

Características de una Peer to Peer con tablas hash distribuidas

- **Disponibilidad y Fiabilidad:** Al ser un sistema altamente distribuido, es muy robusto. Como la información esta replicada en varios objetos, la probabilidad de que se pierda un dato disminuye de forma exponencial con el número de objetos que comparten esa información.

- **Escalabilidad:** Una red Peer to Peer no tiene problemas a la hora de añadir nuevos nodos, sin embargo, la única limitación que tiene es de consumo de ancho de banda a la hora de hacer las búsquedas, aunque como el coste de búsqueda es logarítmico, el número de objetos tiene que ser muy grande antes de que empiecen a notarse los efectos sobre rendimiento en las búsquedas.

6.4. Base de datos federadas

6.4.1. Funcionamiento de una arquitectura federada

Las base de datos federadas son un buen candidato para utilizarse en IoT porque permiten la integración de un conjunto de sistemas heterogéneos y autónomos.

La forma de definir la estructura del sistema federado es utilizando la representación estructurada de la relación contiene-contenido que existe en el mundo real: una ciudad contiene barrios, un barrio contiene edificios, un edificio contiene plantas... y en sentido inverso: una planta está contenida en un edifico, un edificio está contenido en un barro y un barrio está contenido en un ciudad.

De esta forma cada nodo de la estructura federada se encargaría de registrar los dispositivos de IoT que se encuentren en el área física que delimitan.

Hay que aclarar que un sistema federado no es un base de datos en si misma, y requiere que cada nodo disponga de su propio SGBD, y el tipo puede ser distinto de un nodo a otro. La interfaz externa que publica el nodo raíz es el de una base de datos lógica, que es el resultado de la unión de todas las base de datos, pero este nodo no dispone de ningún dato, por lo que todas las consultas se propagan por todos los notos de la estructura.

La estructura federada permite que cualquier dispositivo pueda enlazarse con cualquier nodo de la red, y este pasa a ser gestionado por este nodo. Cualquier nodo tendrá acceso a todos los dispositivos que controla de forma directa, pero, a los dispositivos que controla de forma indirecta (aquellos que están vinculados a nodos que están por debajo de él en la estructura federada), solo tendrá acceso a una parte de sus propiedades.

6.4.2. Ejemplo de una arquitectura federada para IoT

En este enfoque se propone una arquitectura federada sobre un ambiente distribuidos, para ello se tormará el esquema de la Universidad Autónoma de

Santo Domingo (UASD); El nodo principal contendrá el nodo universidad (UASD), la universidad facultades, las facultades contendrán los departamentos, los departamentos contendrán aulas y las aulas contendrán los objetos que son representandos por los alumnos y/o dispositivos de IoT directa o indirectamente conectados, tal como se muestra a continuación.

Esquema de una Red IoT en la Universidad Autónoma de Santo Domingo.

- UASD (Universidad Autónoma de Santo Domingo)

Los nodos serán los siguientes:

- La biblioteca
- El comedor
- La facultad
- Los Departamentos
- Las Aulas

La interconexión entre estos nodos se hace de la siguiente manera:

- El nodo UASD se interconecta con la biblioteca, el comedor y las facultades.
- Las facultades se interconectan con cada uno de sus respectivos departamentos
- Los departamento se interconecta con sus respectivas aulas.
- Las aulas se interconectan con los dispositivos IoT.

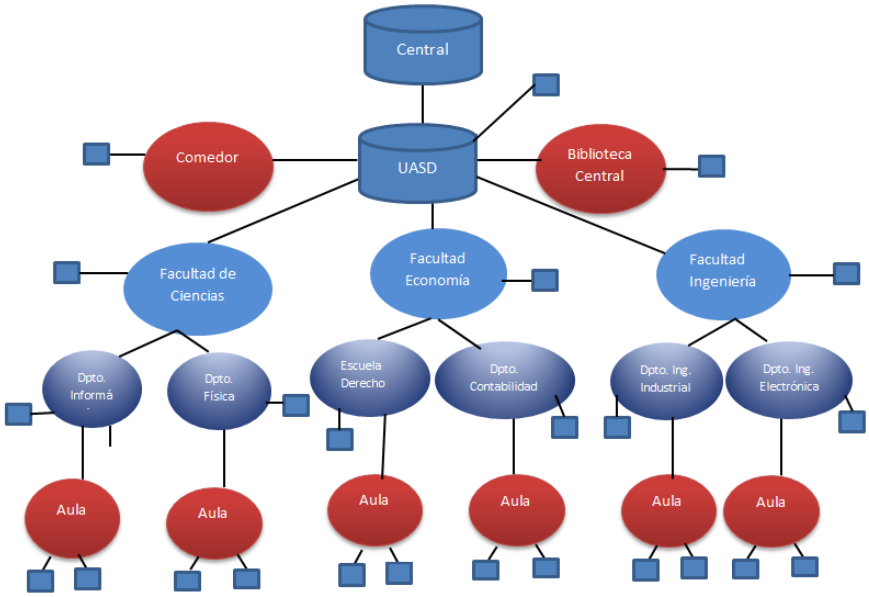

Figura 6.3: Ejemplo de una arquitectura federada

En esta arquitectura los objetos serán representados por los alumnos, cada alumno tendrá un Tag RFID, que servirá para identificarlo en el sistema, cada nodo tiene un acceso completo a toda la información de los objetos que cuelgan de el directamente (si un alumno esta paseando por el campus y entra en una zona que no pertenece a ninguna facultad, entonces se enlazará con el nodo de la universidad directamente y el nodo universidad tendrá pleno acceso al objeto alumno).

Cada nodo tendrá información parcial de los servicios que gestiona indirectamente, es decir, todos los objetos que están por debajo de él pero en la misma rama, y solo tendrá acceso a una parte de los datos que el objeto publica, (por ejemplo: el nodo facultad de ciencia solo controla de forma indirecta a un alumno que este en un aula del departamento de física, pero no sabrá nada de uno que esté en la facultada de economía o que cuelgue del nodo de la universidad directamente), pero por lo general un objeto puede ser gestionado por cualquier nodo, no solo por los que están más abajo de él.

Características de una Red Federada:

- **Fiabilidad y Disponibilidad:** Una red federada es tan sensible a fallos como lo sea el más débil de sus componentes, si un servidor de la red cae, dejarán de ser accesibles todos los objetos que hay por debajo de él.
- **Escalabilidad:** Los objetos se estructuran en un árbol, y si este se queda sin capacidad, es fácil añadir nuevos servidores al árbol para aumentar su

capacidad. Además, como un nodo se puede organizar en clusters, la escalabilidad de este también es buena. Sin embargo, como se había mencionado anteriormente, una red federada está diseñada para que cada nodo gestione los objetos de su respectiva área, si existe una movilización masiva de estos objetos a un nodo determinado, (como por ejemplo que los estudiantes se desplacen de sus respectivas áreas a la biblioteca o al comedor), la red se saturará al intentar responder a tantas peticiones a la vez, por lo que habría que sobre-dimensionar los nodos propensos a picos de uso, quedando desaprovechados los recursos de estos cuando los objetos regresen a sus lugares de procedencia, por todo lo antes dicho, una arquitectura federada sería poco escalable en ambientes dinámicos.

6.5. Ventajas del uso de SGBD de tipo NoSQL

Tal como se menciona en el capítulo de retos, uno de los problemas que tienen los SGBDR se derivan de aplicar el teorema CAP; en el siguiente gráfico se muestran varios SGBD y como estos se ven afectados por ese teorema:

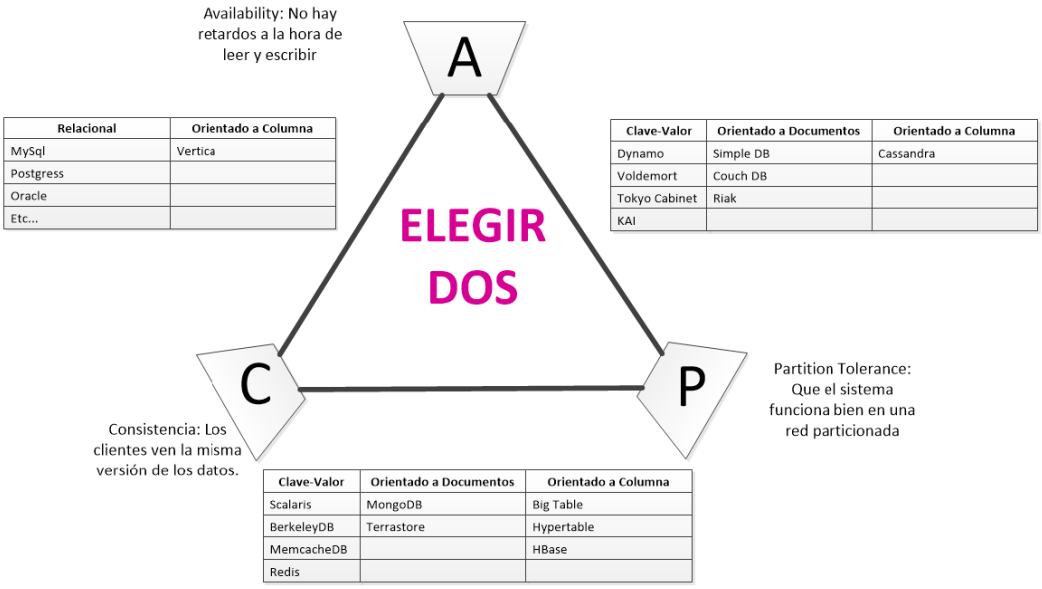

Figura 6.4: Diagrama CAP

Según se aprecia en la figura 6.4, se pueden hacer tres grandes grupos de SGBD, tomando en cuenta la manera en la que estos aplican el teorema CAP:

El primer grupo (CA) sería el de aquellas bases de datos que no tienen tolerancia a particionado, dentro de este grupo están incluidas los SGBDR, ya que no están preparados para tener nodos en redes distintas. El segundo grupo (CP) es el de aquellas que han sacrificado una disponibilidad alta, y el ultimo grupo(AP), son aquellas que no implementan ACID y por tanto han perdido la consistencia de datos.

Para IoT, de las tres características de Teorema CAP la más importante es la de disponibilidad, ya que en IoT los datos son masivos y en tiempo real, por lo que es más crítico que el sistema permita la escritura y lectura de datos con rapidez, por eso se puede descartar el grupo CP, ya que no tiene esta característica fundamental.

De las dos características restantes, la más importante seria la Tolerante a Particiones, en virtud de que esta permite que la red esté configurada en varios nodos que pertenezcan a redes distintas y cada nodo gestiona los objetos que estén en su zona, por lo que en un escenario IoT esto sería más factible.

Ahora solamente quedaría disponible AP, hay que tener en cuenta, que aunque estos SGBD no implementan ACID, y por tanto, no hay consistencia de datos, no quiere decir que los datos sean siempre inconsistentes, ya que la mayoría de estos sistemas utilizan técnicas alternativas (Como puede ser la actualización en segundo plano, la replicación, etc.) para implementar lo que se llama consistencia eventual que significa que los datos que no son consistentes "eventualmente" lo serán.

Una vez acotado cual sería el grupo más apropiado para ser utilizado en IoT, se destaca que dentro del grupo hay tres filosofías:

- **Clave-Valor:** es un sistema que no requiere ningún esquema y en el que los datos son almacenados en forma de un par clave/valor. Este par suele estar formado por una clave(generalmente una cadena de texto que identifica al valor), y el valor que puede ser de cualquier tipo (entero, string, double, etc...)
- **Orientado a columnas:** en una Base de datos de este tipo, los datos están almacenados en forma de columna, lo positivo del uso de esta clase de sistemas es que permite el acceso a elevados volúmenes de dados a gran velocidad, sin embargo, tiene la desventaja de que la lectura de los datos es muy lenta, por lo que no seria apropiada para un entorno de IoT, ya que los datos se generan de manera constante y este sistema podría dar problemas de rendimiento.
- **Orientado a documento:** es aquella diseñada para manejar información semi-estructurada, también llamada información orientada a documento. Generalmente, estos documentos suelen estar en la forma de código XML,

JSON, BSON, etc... Este tipo de SGBD esta recomendado para datos heterogéneos, y tiene un alto rendimiento en inserciones, por lo que es apropiado para IoT.

6.5.1. NoSQL tipo Clave-Valor

Para afrontar los problemas de almacenamiento que sufren los SGBD, muchos sistemas (de tipo NoSQL en su mayoría) han optado por usar base de datos en la nube.

La propuesta presentada a continuación, es un solución de base de datos NoSQL de tipo Clave-Valor, en la nube[40]. Esta propuesta consiste en una solución de almacenamiento para IoT, llamada "IoT Management Data Base"(IOTMDB).

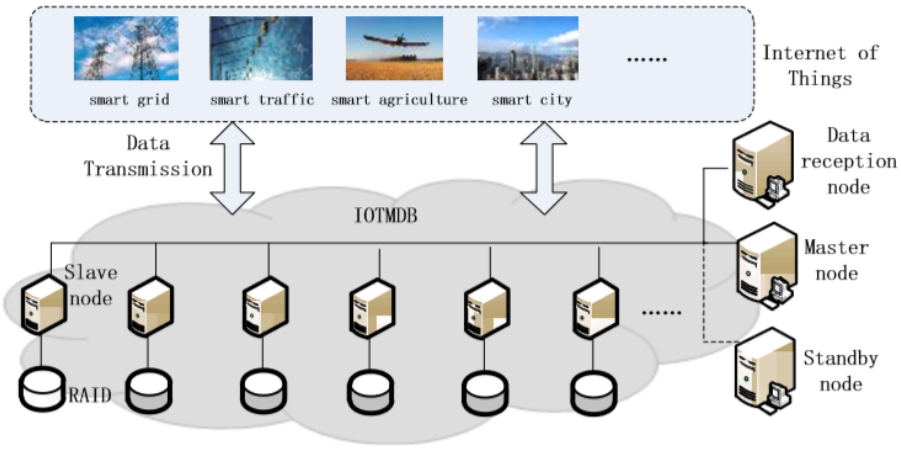

Figura 6.5: Arquitectura del Sistema IOTMDB

En esta arquitectura existen cuatro roles principales: El nodo maestro, nodo standby, nodo recepción y nodo esclavo.

El nodo maestro que es el encargado de gestionar los diferentes clusters, sus tareas incluyen: aceptar conexiones de los clientes y repartir la escritura a los distintos nodos esclavos y monitorizar la secuencia de escritura, el nodo standby es un duplicado del nodo maestro, en caso de que el nodo maestro falle, el nodo standby tomará su lugar. El nodo de recepción de datos es el que recibe los datos provenientes de los sensores y los pre-procesa.

Para afrontar los problemas de datos heterogéneos, el autor de este artículo propone el uso del estándar *"Sharing"* para definir que información debe ser almacenada y como se deben emitir los datos, para ello se utiliza la unidad

más pequeña del sistema IOTMDB nombrada *"SampleElement"* que representan la *"clave"* y el*"valor"*, tal como se muestra en el siguiente ejemplo:

"SampleElement" =<*key:value*>

key representa el nombre del objeto o magnitud, y el *value* representa el valor leido en ese instante, ejemplo:

<*temperatura:20*>, La temperatura es la magnitud y el número 20 representa el valor de la temperatura leido por el sensor en ese momento.

Para almacenar los datos se utiliza la unidad de *"SampleRecord"* formado por un conjunto de *"SampleElement"*. El *"SampleRecord"* contiene dos tipos de información: La información estática: que es la información no variable del objeto, como su número de identificación personal, el campo al que el objeto pertenece etc. y la información dinámica: que describe los cambios de estado del objeto como son: el tiempo, la temperatura, velocidad, etc. Los *"SampleRecord"* están organizados en colecciones y una colección puede contener diferentes objetos o aplicaciones.

NO.	SampleRecord
1	((objID:"a01", field:"agriculture", type:"shed"), (t:t1, loc: (40.1,20.5), (light:50, humidity:0.56, light:23.1))
2	((objID:"b123", field:"traffic", type:"car"), (t:t2, loc:(123,1024), speed:210, capturePhoto:(type:"image", format:"jpeg", plateNumber:"BJ0A435", digestValue:" *b38767a34dd2d764c0d597986010e5b1",* originalValue*:"*p_1")))

Figura 6.6: Ejemplo de un SampleRecord

Con el uso del *"SampleRecord"* los datos heterogéneos pueden ser representados de manera uniforme.

6.5.2. NoSQL orientado a documentos

En el articulo [41] se presenta una solución para utilizar una base de datos orientada a documentos para IoT:

Los objetivos que se deseaban alcanzar con este diseño eran los siguientes:

- **Dependencia de datos:** El modelo de datos debe ser lo suficientemente flexible para soportar contenido heterogéneo y multimedia y aceptar diferentes formatos de datos, como imágenes, videos, y audio.

- **El Estandard API:** La función provista por la infraestructura de almacenamiento debe exportarse como un estándar de servicios web para facilitar la integración e interconección con otros sistemas y el desarrollo de aplicaciones IoT.
- **Soporte para datos masivos:** La infraestructura debe soportar la replicación robusta y eficiente para proporcionar escalabilidad. La arquitectura del sistema debe proveer medios de balanceo de carga y procesamiento de consultas a modo distribuido.
- **Flexibilidad y Manejabilidad:** El almacenamiento de los datos y la recuperación de funciones debe soportar metadatos y cambios de notificaciones.
- **Seguridad** El sistema debe soportar autenticación a alta escala y comunicación encriptada.

Para que el modelo de datos alcanzara los objetivos antes expuestos, el diseño fue realizado de la siguiente manera:

En este esquema el documento es la unidad fundamental de almacenamiento, formado por una serie de campos y sus adjuntos, y es identificado por su correspondiente ID. Cada documento contiene metadatos para facilitar su organización y administración, los campos del documento contienen los datos textuales, un campo puede ser de tipo booleano, o un número o una lista.

Los datos son generados por un nodo (dispositivo o sensor de la red) y reportados a una colección para fines de almacenamiento. Los datos individuales están representados por un documento almacenado en la base de datos. Los datos escalares son mapeados a su determinado campo tomando el cuenta su tipo, mientras que los datos multimedia son mapeados a su respectivo adjunto. Esto permite flexibilidad de almacenamiento para una gran cantidad de datos heterogéneos en el sistema. Para facilitar la organización y recuperación, la estructura utilizada para los documentos es ligera lo que la hace muy adecuada para aplicaciones de IoT.

CID	Descripción	Formato
0	Temperatura	float(Grados celsius)
1	Humedad	float (Porcentaje)
2	Presión	float (kiloPascal)
3	Aceleración	(X, Y, Z) triple
4	Ubicación	(Latitud, Longitud, Elevación)
5	Datos multimedia	referente al adjunto

Cuadro 6.1: ID de los datos y su correspondiente formato

Los datos reportados son asociados a los siguientes campos:

- **Timestamp:** describe el tiempo en que los datos fueron colectados del medio ambiente.
- **ID del dispositivo:** Identificador único del dispositivo.
- **ID de la clase:** Identificador único que describe la naturaleza de los datos.
- **Datos:** los datos escalares o referencia a un adjunto del mismo documento
- **Tags:** Colección de metadatos de aplicaciones específicas.

La aplicación es lo suficientemente flexible para soportar campos con múltiples valores, de manera tal que si en un nodo se miden dos magnitudes distintas, los valores puedan ser reportados en el mismo documento, tal como se muestra en el cuadro 6.1.

Entre las operaciones soportadas por esta propuesta, las más importantes para IoT serian las siguientes:

- **Inserción y Eliminación:** Esta solución soporta tanto la inserción como la eliminación de documentos en la base de datos. La eliminación de documentos puede servir para manejar aquellos datos insertados de manera errónea (Errores de software) o para eliminar datos innecesarios.
- **Modificación y Versionamiento:** En algunos casos en los que se deba hacer cualquier modificación (ya sea por la reconfiguración de un sensor o por otro motivo) los datos pueden ser actualizados para que sean consistentes con la nueva versión del software o sensor. para ello esta solución soporta la edición de documentos existentes y para mantener constancia de algún objeto en específico los records del mismo son mantenidos para futuras recuperaciones si fuese necesario.

Ejemplo de un documento:

```
"timestamp": "2009-07-30T13:31:37.459Z"
"tags": ["POSTPROCESS"]
"divice_ide": "0014:4F01:0000:01AB"
"class_id": [1,2]
"data": [70, 15.3]
```

Estas dos soluciones resultarían ser muy apropiadas para un escenario IoT; la de tipo clave-valor, proporcionaría cierta estructura a los datos de modo

que sea mucho más sencilla la tarea de búsqueda, y la segunda propuesta orientada a documentos facilitaría la tarea de almacenamiento y provee una solución para trabajar con datos heterogéneos con el uso de un estándar. Estas propuestas aportarían soluciones a los problemas de IoT y servirían como punto de partida para otras soluciones emergentes.

6.6. Minería de Datos para IoT

Tal como se ha tratado en el trascurso de esta tesina, los datos tienen un gran valor, tanto para las empresas, como a nivel científico, educativo, cultural, económico, etc., de ahí radica la importancia de saber gestionarlos correctamente y protegerlos de posibles atacantes; las empresas invierten mucho dinero en la creación de datoawarehouse para guardar los datos de sus operaciones, pero ¿Qué hacer con los datos luego de almacenarlos? y ¿Porqué razón a las empresas les interesa guardar sus datos históricos?, lo importante no seria almacenarlos, sino extraer conocimiento valioso de ellos, esto resulta crucial en entornos en que la toma de decisiones se produce en cuestión de segundos, de ahí la importancia de la Minería de Datos; el objetivo de la Minería de Datos es el de extraer conocimientos valioso de grandes cantidades de información, por esa razón es incalculable la utilidad de MD en un escenario IoT.

6.7. Soluciones de Minería de Datos

En el capítulo V, se trató el tema de los grandes desafíos a los que se enfrentan los SGBD existentes en la actualidad, para gestionar IoT.

Suponiendo que se dispone de un dispositivo de rastreo de las actividades que hace un usuario durante todo el día. Al echarle un vistazo a los datos que recolecta, se detectan fácilmente unos patrones específicos, luego se realiza una correlación de los datos de esas actividades con el tiempo y posteriormente se hace una comparación de estos datos con otros datos previamente almacenados. Se puede clasificar cada actividad por niveles diferente (mediano, alto..etc) y registrar en que categoría se encuentra cada una. A este proceso se le llama Agrupamiento de datos (Data Clustering).

Es muy fácil hacer un seguimiento de patrones manualmente cuando hay pocos datos que analizar, sin embargo manejar la cantidad de información que generaría un sistema de IoT con únicamente un par de miles de sensores, será un gran problema. las ciencias de la computación han aplicado métodos estadísticos y construido herramientas que permiten hacer minería de datos y extraer información valiosa de un conjunto de datos tan grande.

Las principales técnicas que se han diseñado para DM, que tienen utilidad en IoT son: detección de datos anómalos, clasificación de datos, selección de características, agrupamiento, y predicción de series temporales.

- **Detección de datos anómalos:** es una técnica que se utiliza para identificar situaciones que no corresponden a los patrones previamente definidos.

La mayoría de las alarmas y eventos que lance el sistema, estarán asociados a estas situaciones, por eso es importantes detectarlos.

- **Clasificación de datos:** es un método usado para identificar o esclarecer la asociación o dependencia de un atributo con varias clases determinadas.
- **Selección de características:** se entiende por "característica o atributo," al tipo de dato usado para el reconocimiento de patrones, la correcta selección de estos servirá para determinar cuales factores influyen en la ocurrencia de situaciones.
- **Agrupamiento:** consiste en detectar grupo de datos con características similares con la finalidad de simplificar los datos para facilitar la identificación de patrones y la detección de valores anómalos.
- **Predicción de series temporales:** las técnicas de series temporales consisten en realizar una estimación sobre comportamiento futuros basados en datos históricos previamente coleccionados y analizados.

6.7.1. Modelos multilayer

Una de las soluciones propuestas es la presentada a continuación.

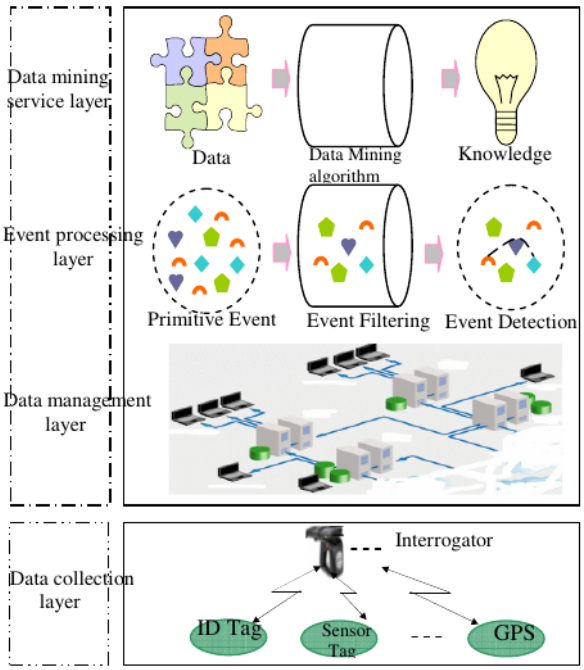

Figura 6.7: Capa de colección de datos [42]

El modelo consta de cuatro capas, cada una cumple una función: coleccionamiento de datos, administración de datos, procesamiento de eventos y servicios de minería de datos.

- **La capa de coleccionamiento de datos:** en ella se encuentran los dispositivos con lectores RFID y el sistema se encarga de coleccionar los datos provenientes de ellos.
- **La capa de administración de datos:** es aplicable tanto a arquitecturas centralizadas como distribuida, esta capa es la encargada de administrar los datos colectados de la capa anterior. luego de realizar un filtrado, los datos son almacenados en su datawarehouse correspondiente. Otra característica importante de esta capa es que en ella los objetos inteligentes se conectan entre si.
- **Procesador de eventos:** es la capa en donde se hace el procesamiento de datos, para ello se combinan los datos, el tiempo y otros factores. En esta capa los datos son analizados y posteriormente filtrados y luego son organizados tomando en cuenta los eventos.
- **Servicios de Minería de Datos:** Esta capa surge de la información obtenida de la capa de administración y procesamiento de eventos. En ella se utilizan patrones de minería de datos como clasificación, asociación, detección, clustering, etc. este capa esta orientada a los servicios.

6.7.2. Problemas de Minería de Datos

A) **Captura de datos de los dispositivos:** un dispositivo está continuamente capturando información, y si tiene que enviar toda esa información puede tener problemas de consumo energético, o de saturación de la red. Una solución es la agregación de los datos que permite reducir el volumen de información. Otra solución es la propuesta por Joydeep Ghosh [43] un framework probabilístico para la captura de los datos.

B) **Indexación y gestión de los datos:**. se precisan métodos de indexación que faciliten el rápido acceso a todos los objetos, sin embargo, hay que tener en cuenta que la información que se genera en IoT tienen unas características propias que no se suelen dar en otros sistemas, por lo que es necesario crear nuevos sistemas para manejarlos.

C) **Abstracción y compresión:** los objetos generan mucha información redundante y estos dos métodos ayudan a detectarlo para evitar el almacenamiento de información duplicada.

D) **Filtrado por eventos:** es una técnica que consiste en no almacenar toda la información que genera un objeto, sino solo los eventos asociados a cambios de estado; para explicar como funciona se presenta el siguiente ejemplo: se podría programar un sensor de temperatura para detectar la presencia de un individuo cuando entra en un lugar, de esa forma se controla que solo se almacene información de cuando una persona entra o sale de la habitación sin tomar en cuenta toda la información relacionada a la variación de temperatura, evitando así el uso innecesario de energía y espacio de almacenamiento.

E) **Preprocesado centralizado vs distribuido:** según el punto anterior la información debe ser procesada antes de almacenarla, lo ideal seria que los propios objetos realizaran este filtrado y solo enviarán al nodo central la información relevante, pero los nodos tienen limitaciones (procesador, energía, memoria) es probable que no se pueda realizar todo el procesado que es necesario en los nodos.

F) **Algoritmos de Minería de Datos:** hay que crear nuevos algoritmos para hacer Minería de Datos sobre la información de IoT, las propias características de IoT hace que los algoritmos actuales no sean del todo adecuados.

G) **La siguiente generación de internet:** Hay nuevas tecnologías en internet (computación ubicua, web semántica, etc...) que pueden aplicarse a IoT, pero requiere de futuros estudios, algunos de esos ya se están haciendo, por ejemplo adaptar la tecnología orientadas a la web semántica para crear modelos semánticos, entre otras, sin embargo aún falta muchos desafíos a vences tal como se explicaba en el capítulo V sobre retos.

CAPÍTULO 7

CONCLUSIONES

Los SGBD relacionales aparecieron en la década de los 70, han ido evolucionando para adaptarse a los nuevos requisitos emergentes, sin embargo, se está demostrando que este tipo de sistemas no pueden dar respuesta a las exigencias de IoT.

Si se tiene en cuenta la diversidad de dispositivos (objetos) que pueden formar parte de la red IoT, la única forma en la que los sistemas relacionales pueden gestionar estos datos heterogéneos es definiendo un estándar que establezca el formato en el que los objetos de la red deben transmitir sus datos. Pero esto es sólo una solución provisional, la verdadera solución reside en que el SGBD sea capaz de trabajar de forma nativa con datos heterogéneos.

Otra de las limitaciones que tienen los sistemas relacionales actuales es su capacidad de procesamiento. Si consideramos que los datos generados por los objetos en la red IoT son de gran volumen y que se generan de forma ininterrumpida, los SGBD deben mejorar su rendimiento para poder procesar de manera eficiente la cantidad de información que entrará de manera continua. Esta situación se soluciona actualmente utilizando una capa de caché, pero esta solución también es provisional, ya que solamente es funcional mientras se siga manteniendo un volumen de datos que todavía no ha alcanzado el que se estima que habrá cuando el uso de IoT se extienda.

En cuanto a las arquitecturas para SGBD existentes, las más apropiadas para una red IoT siguen teniendo debilidades: una arquitectura centralizada requerirá de una capacidad de almacenamiento muy elevada al utilizar un único nodo central; la estructura federada presenta problemas de dimensionado de los

nodos, debido al dinamismo y movilidad de los objetos, surgirían problemas de saturación en períodos de tiempo determinados, cuando los objetos se concentran en una área.

La mayoría de los problemas que presentan los SGBD relacionales se deben a lo estructurado del estándar ACID para la gestión de transacciones. Para un entorno dinámico como IoT sería más apropiado un sistema más flexible.

Todas estas limitaciones conducen a plantearse que la mejor opción sería crear un nuevo tipo de SGBD especializados para dar soporte a IoT, que sean capaces de dar respuestas a las características de un ambiente dinámico y heterogéneo.

Hoy en día, existen algunas propuestas orientadas en esta dirección, como las bases de datos NoSQL, que por su planteamiento están más orientadas a ser utilizadas en IoT, y actualmente son la solución más prometedora, sin embargo hay que tener en cuenta que todavía es una tecnología emergente, y no hay que apresurarse a afirmar que éste será el sistema más apropiado y definitivo para IoT, ya que solo ofrecen solución a una parte del problema que es el almacenamiento de la información, todavía queda pendiente el punto de consultar la información almacenada y hacer minería de datos sobre ella. Y por los mismos motivos se necesitan desarrollar nuevas tecnologías (como lenguajes para hacer consultas para datos semi-estructurados).

BIBLIOGRAFÍA

[1] Sensors empower the "internet of things". http://www.edn.com/design/sensors/4363366/Sensors-empower-the-quot-Internet-of-Things-quot-, Consultado el 14 de Abril del 2013.

[2] Anne James Joshua Cooper. *Challenges for Database Management in the Internet of Things.* Pg.320-329, Hildebrand, London, United Kingdom, 28-8-2009.

[3] That 'internet of things' thing. http://www.rfidjournal.com/articles/view?4986, Consultado el 15 de Abril del 2013.

[4] Mark Weiser. The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev.*, 3(3):3–11, July 1999.

[5] W. Keith Edwards. Discovery systems in ubiquitous computing, 2006. *Journal IEEE Pervasive Computing.*, 5:70–77, April 2006.

[6] Iot, en un mundo conectado de objetos inteligentes. http://www.fundacionbankinter.org/es/publications/the-internet-of-things Consultado el 26 de Junio del 2013.

[7] Bill Gates. *The dissapearing of computer*, 2003.

[8] Gustavo Adolfo Ramírez González. Tesis doctoral: Evaluación de introducción de internet de objetos en espacios de aprendizaje, july 2010.

[9] Gan Gang, Lu Zeyong, and Jiang Jun. Internet of things security analysis. In *Internet Technology and Applications (iTAP), 2011 International Conference on*, pages 1–4, 2011.

[10] Natalia Peccis Rubio. *Tesis de Master Rehabilitacion de viviendas eficientes e inteligentes.* Universidad Politećnica de Madrid, Madrid, España, 2011.

[11] Ernesto Serrano Carlos Álvares, Danilo Olguín. *Tesis de Grado: Diseño de una instalación Domótica en un condominio para el control de seguridad e iluminación mediante la tecnología LonWorks.* Escuela Superior Politénica del Litoral, Guayaguil, Ecuador, July 2007.

[12] Mercedes Marqués. *Bases de Datos.* Departamento de Ingenieria, Ciencias de la Computación, Universidad JAUME, 978-84-693-0146-3, Castellón de la Plana, España, 2011.

[13] Historia de las bases de datos. http://histinf.blogs.upv.es/2011/01/04/historia-de-las-bases-de-datos/ Consultado el 11 de Abril del 2013.

[14] El modelo relacional de bases de datos. http://umad.zapatolibrce.com/moodle files/ArticuloConsultado el 17 de Abril del 2013.

[15] Giovanni Ayala Zúñiga. *Historia y Evolución de las Bases de Datos.* Los Michis, Sinaloa, México, Enero 2009.

[16] Universidad de Sevilla. *Modelo Relacional de Codd.* Sevilla, España, 2004.

[17] El modelo relacional y el álgebra relacional. http://ocw.uoc.edu/computer-science-technology-and-multimedia/bases-de-datos/bases-de-datos/P06 M2109 02148.pdf, Consultado el 11 de Junio del 2013.

[18] Departamento de Ciencias de la Computación. *Fundamentos de diseño de bases de datos.* Granada, España.

[19] Shamkant B. Navathe Ramez Elmasri. *Sistemas de Base de Datos.* Addison-Wesley Iberoamerica, S.A., Wilmington, Delaware, Estados Unidos, 1997.

[20] Laura Mota Herranz Matilde Celma Giménez, Juan Carlos Casamayor Ródenas. *Base de Datos Relacionales.* Pearson, Educación, S.A., Madrid, España, 2003.

[21] Angie Aguilar Domínguez. *Base de datos con interfaz web para el departamento de multimedios y aplicaciones interactivas del museo Universum.* Universidad Nacional Autónoma de México, Distrito Federal, México, 2009.

[22] MySqlHispano. Normalización de base de datos. Mayo 2003.

[23] Nancy Patricia Macas Carrasco Ana Lucía Juntamay Tenezaca. *Estudio y Aplicación de procedimientos de análisis forense en servidores de Bases de Datos SQL Server y MySql, caso práctico.* Tesis de grado, Escuela Superior Politécnica de Chimborazo, Riobamba, Ecuador, 25 de enero del 2012.

[24] Matilde Celma Giménez. *Exploración de un almacen de datos, Herramientas OLAP.* Universidad Politécnica de Valencia, Valencia, España.

[25] Sofia J. Vallejos. *Minería de datos.* Universidad Nacional del Nordeste, Corrientes, Argentina, 2006.

[26] José Jesús García Julio Villena, Raquel Crespo. *Minería de Datos.* Universidad Carlos III de Madrid, Madrid, España.

[27] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems).* 2005.

[28] S. sumathi, S.N. sivanandam introduction to data mining and its applications. *Gene Expression.*

[29] José Hernández Orallo. *Introducción a la minería de datos).* Universidad Politécnica de Valencia, Valencia, España.

[30] Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors. *Advances in knowledge discovery and data mining.* American Association for Artificial Intelligence, Menlo Park, CA, USA, 1996.

[31] How much information is there in the world? http://www.lesk.com/mlesk/ksg97/ksg.html, Consultado el 25 de Junio del 2013.

[32] Dennis McLeod and Dennis Heimbigner. A federated architecture for database systems. In *Proceedings of the May 19-22, 1980, national computer conference*, pages 283–289. ACM, 1980.

[33] Alberto Brache Caballero. *Arquitectura Peer To Peer.* Universidad Autónoma del Caribe Barranquilla, Colombia.

[34] Ricardo W. Brito. *Bancos de Dados NoSQL x SGBDs Relacionais: Análise Comparativa.* Universidad de Fortaleza, Brasil, Noviember del 2010.

[35] The "nosql" discussion has nothing to do with sql. *http://cacm.acm.org/blogs/blog-cacm/50678-the-nosql-discussion-has-nothing-to-do-with-sql/fulltext*, Consultado el día 16 de Junio del 2013.

[36] Jan Sipke van der Veen, Bram van der Waaij, and Robert J Meijer. Sensor data storage performance: Sql or nosql, physical or virtual. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 431–438. IEEE, 2012.

[37] Verizon. *2012 Data Breach Investigations Report,*. the Australian Federal Police, Dutch National High Tech Crime Unit, Irish Reporting and Information Security Service, Police Central e-Crime Unit, and United States Secret Service, 2012.

[38] Ciprian. Pungila, Ovidiu. Aritoni, and Teodor-Florin. Fortis. Benchmarking Database Systems for the Requirements of Sensor Readings. *IETE Technical Review*, 26(5):342–349, 2009.

[39] Nuno Santos(NEC) Ricardo de la Heras(TID). *Concepts and Solutions for Identification and Lookup of IoT Resources.* Internet of Thing Architecture, Unión Europea, Dicember 2011.

[40] Tingli Li, Yang Liu, Ye Tian, Shuo Shen, and Wei Mao. A storage solution for massive iot data based on nosql. In *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*, pages 50–57. IEEE, 2012.

[41] Mario Di Francesco, Na Li, Mayank Raj, and Sajal K. Das. A storage infrastructure for heterogeneous and multimedia data in the internet of things. *2012 IEEE International Conference on Green Computing and Communications*, 0:26–33, 2012.

[42] Wang Xiaoyi Shen Bin, Liu Yuan. Research on data mining models for the internet of things. 9-11 April 2010.

[43] Joydeep Ghosh. A probabilistic framework for mining distributed sensory data under data sharing constraints. *Knowledge Discovery from Sensor Data*, 7:1, 2008.