UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 9

DISEÑO DE LA BASE DE DATOS PARA EL SISTEMA DE CAPTURA Y CATALOGACIÓN DE MEDIAS

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

AUTOR: Yusniel Yanes Perón

TUTOR: Ingeniero en Ciencias Informáticas Karlen Trimiño Pérez

Ciudad de la Habana, viernes 29 de mayo del 2009.

"Año 50 de la Revolución".

"La ciencia es la aproximación progresiva del hombre al mundo real"

Max Planck.

DEDICATORIA

Son muchas las personas a las que quisiera dedicarle esta tesis, porque son muchas las personas que me han ayudado a ser lo que hoy soy, pero por esta vez sólo voy a dedicar este trabajo a dos personas muy importantes para mí:

A Tania, mi bella, insustituible, cariñosa y adorada madre, por ser mi ángel de la guarda, y al mismo tiempo, mi mejor amiga.

A la memoria de José Yanes Pradera (Joseíto), por haberme enseñado tanto en tan poco tiempo.

AGRADECIMIENTOS

Agradezco en primer lugar a mis padres, que tanto han sacrificado y luchado por mí en la vida, y que de alguna forma quisiera retribuirles convirtiéndome en un profesional y un hombre de bien. A mi familia en general, que es una de las prioridades que debe tener todo ser humano. A mis hermanos, en especial a Yosbany y Daily, dos de los seres que más adoro. A mis primos, sobre todo a Lisván y Anay, que tan pendientes han estado de mí.

Agradezco a todos mis amigos, compañeros de aula, profesores. A Lily, mi preciosa novia que tanto apoyo me ha dado.

A mi tutor Karlen, mi tribunal, oponente y demás personas relacionadas con esta tesis.

Agradezco también a Fidel Castro y la Revolución cubana, sin los cuales esta carrera y este sueño que hoy vivo no hubieran sido más que una utopía.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor del presente trabajo, y autorizo a la Universidad de las Ciencias Informáticas, así como al Sistema de Captura y Catalogación de Medias del Polo de Video y Sonido Digital, a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de ____ del año _____.

Autor:

Yusniel Yanes Perón.

_____

Tutor:

Ing. Karlen Trimiño Pérez.

_____

RESUMEN

El presente trabajo surge como parte de un sistema de captura y catalogación de medias para el cual constituye el soporte, en lo que a base de datos respecta, para su correcto funcionamiento. Se trata de la propuesta de diseño lógico y físico de la base de datos para el proyecto Solución de Captura y Catalogación de Medias, del polo de Video y Sonido Digital de la Universidad de las Ciencias Informáticas, Cuba.

Se da una explicación minuciosa del diseño del modelo lógico y físico de la base de datos del sistema antes mencionado, como respuesta a la situación problémica planteada. Para lograr desarrollar el producto final fue necesario hacer un proceso investigativo en aras de brindar una solución lo más óptima posible, que cumpla con las necesidades planteadas de forma cabal y que se mantenga a la par de los sistemas actuales más modernos. Por otra parte se exponen una serie de conceptos, tecnologías, mecanismos, entre otros elementos que harán posible el entendimiento del presente trabajo.

Índice de contenidos

Índice de tablas

Índice de figuras

INTRODUCCIÓN

En la actualidad se le brinda una gran importancia a la gestión de la información a través de medios informáticos, pues estos optimizan en una gran medida la fluidez de los procesos, la toma de decisiones y la calidad de trabajo en las diferentes organizaciones. Los sistemas actuales tienen sus propios mecanismos de manejo de datos y almacenamiento de la información persistente. En este aspecto las bases de datos han demostrado su superioridad, y hoy es una necesidad el uso de las mismas sobre todo en los casos donde se almacenan grandes cantidades de información.

En la Universidad de las Ciencias Informáticas (UCI), Cuba, se lleva a cabo la realización del proyecto Sistema de Captura y Catalogación de Medias. Este será un sistema distribuido, y sus módulos estarán instalados en los ordenadores de forma independiente de manera que cada ordenador realice sólo las operaciones para las cuales está destinado. Dichos módulos tendrán un punto en común que será la base de datos, la cual estará en un servidor central de manera que todos los módulos se sirvan de ésta (ver Figura 1). Por tal razón, la necesidad principal consiste en la realización del modelo lógico y físico de una base de datos que sirva de soporte a los módulos del proyecto de manera que éstos puedan realizar funciones como: gestión de usuarios y roles, gestión de metadatos de las medias almacenadas, catalogación de videos, transcripción de audio, entre otras funciones englobadas en los requerimientos funcionales asociados a cada módulo del sistema, y de manera que cumplan con los requisitos no funcionales. Por tal razón se plantea la siguiente problemática:

¿Cómo realizar el modelo lógico y físico de la base de datos para el Sistema de Captura y Catalogación de Medias del polo Video y Sonido Digital de la Universidad de las Ciencias Informáticas, Cuba?

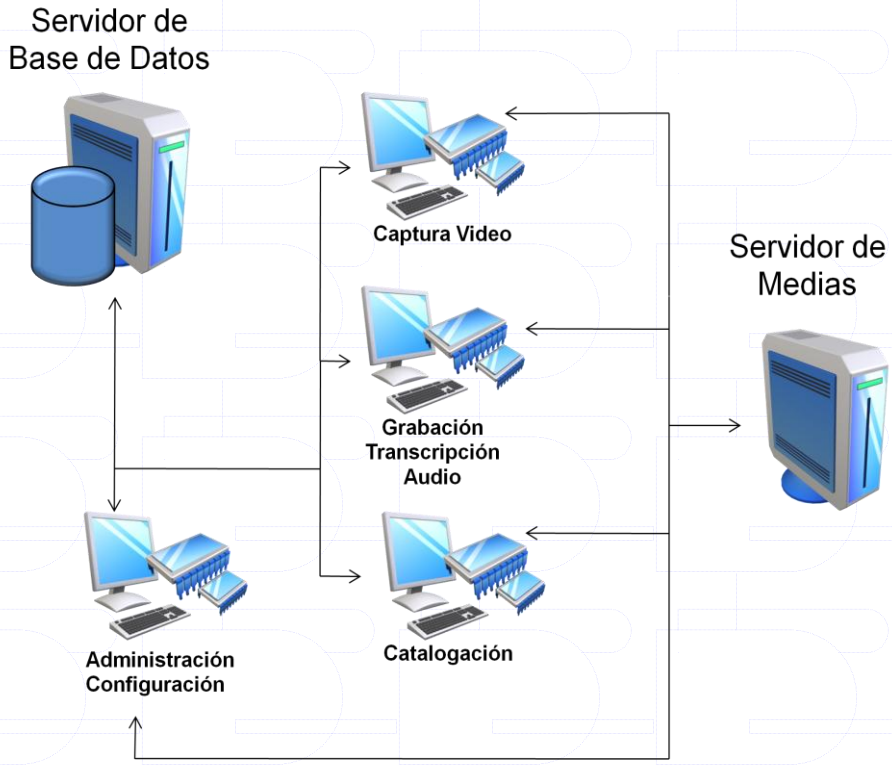


Figura 1: distribución del proyecto Sistema de Captura y Catalogación de Medias.

A continuación se muestran algunas siglas y conceptos asociados al dominio del problema.

BD: Base de Datos.

SGBD: Sistema Gestor de Base de Datos

SCCM: Sistema de Captura y Catalogación de Medias

Streaming: Sistema de transmisión de datos que permite, mediante un programa que maneje estos datos, ir recibiéndolos y a la vez procesándolos sin necesidad de esperar a que se descarguen del todo. Muy práctico para escuchar/ver programas de radio o televisión. (Universia, 2008).

Metadatos: Información que describe datos que incluyen el contenido, la forma y las características técnicas y editoriales de la información electrónica (Sánchez Calas, 2002).

Licencia BSD: Pertenece al grupo de licencias de software Libre. Utilizada por primera vez en 1980 por *Berkeley Source Distribution* (BSD). Esta licencia permite el uso de código BSD sobre software no libre aunque éste luego pase a ser liberado. (LINFO, 2004).

La realización de una base de datos para el Sistema de Captura y Catalogación de Medias del polo de Video y Sonido Digital de la UCI significa la integración de los módulos de dicho proyecto. La necesidad principal del trabajo consiste en que los módulos del sistema en cuestión necesitarán una base de datos que sirva de soporte para su funcionamiento de manera que se cumplan los requerimientos funcionales y no funcionales de manera correcta. La no concepción de la misma implicaría el fraccionamiento del producto final, y a su vez significaría la ausencia de un recurso indispensable para el sistema, traduciéndose esto en una pérdida considerable de tiempo y recursos de las entidades implicadas. En consecuencia de lo antes descrito se plantea el siguiente objetivo:

Realizar el diseño lógico y la implementación del modelo físico de la base de datos Sistema de Captura y Catalogación de Medias del Polo de Video y Sonido Digital de la Universidad de las Ciencias Informáticas, Cuba.

A continuación se listan una serie de tareas para dar cumplimiento a los objetivos planteados:

- Análisis de las tendencias y tecnologías actuales de bases de datos a nivel mundial y cuál sería la idónea para el proyecto.
- Valoración de la normalización y de-normalización de las bases de datos.
- Fundamentación de la definición de un sistema gestor de base de datos.
- Definición de la estrategia y la tecnología de acceso a datos a utilizar.
- Definición de los modelos de despliegue de la base de datos.
- Diseñar el modelo lógico de la base de datos.
- Análisis de la redundancia de la información en la base de datos.
- Implementación del modelo físico de la base de datos.
- Comprobación de la integridad de los datos en la base de datos.
- Comprobación del funcionamiento de la base de datos.

Se cuenta con la siguiente idea a defender: un diseño adecuado de la base de datos permitirá la obtención del Sistema de Captura y Catalogación de Medias del Polo de Video y Sonido Digital de la Universidad de las Ciencias Informáticas de manera que cumpla los requerimientos funcionales y no funcionales definidos para el sistema.

Los aportes prácticos esperados del trabajo consisten en el modelo lógico y físico de la base de datos del sistema Solución de Captura y Catalogación de Medias, así como la documentación de la investigación para el desarrollo de los mismos.

El objeto de estudio consistirá en los modelos lógicos y físicos de la base de datos en el proceso de desarrollo de software para captura y catalogación de medias, y el campo de acción estará enmarcado en las actividades de diseño de dichos modelos dentro del proyecto Sistema de Captura y Catalogación de Medias del Polo de Video y Sonido Digital de la UCI, Cuba.

Antecedentes principales de la presente solución:

Actualmente existen las Mediatecas, donde la información se guarda en cintas de formato analógico y pueden digitalizarse y almacenarse en un Servidor, catalogándose y relacionándose con otros videos o documentos, por lo que trabajan con bases de datos. Por otra parte existe un sinnúmero de aplicaciones destinadas a la captura de lo que acontece en la pantalla o en una parte de esta, por solo mencionar algunos: el *Super Screen Capture*, el *VirtualDub*, *XnView*, *Adobe Captivate*, entre muchos otros. Pero estas aplicaciones están destinadas al trabajo con ficheros y no con bases de datos, por lo que no constituyen antecedentes para la aplicación en propuesta. Habría que mencionar sin duda alguna otras aplicaciones destinadas al trabajo con *streaming* de audio y video que sí realizan trabajo sobre BD, como por ejemplo:

- MediaBox: gestor de Mediateca que facilita la catalogación y archivo multimedia (vídeo, audio, documentos electrónicos) desde diferentes tipos de fuentes y distintos formatos. El Sistema permite asociar documentos según criterio del usuario (por autor, materia, fechas, entre otros datos) digitalizando o transcodificando para guardarlos con la calidad deseada (XTREAM, 2008). MediaBox es un software propietario, por lo que es imposible ver su código fuente y diseño de la BD.
- Xtamp: es un producto que permite la captura, digitalización, compresión, almacenamiento y consulta de audio y vídeo. Xtamp está indicado para captura, archivo y edición de vídeo, así como captura de canales de audio. Responde a la necesidad de tratamiento de grandes volúmenes de información que ha de ser grabada y archivada para su posterior consulta (XTREAM, 2008), pero también es un software propietario.
- Servicio de Multidifusión o *Streaming* de Sarenet: permite difundir una señal de radio o televisión utilizando internet como una gran antena de emisión para llegar

a cualquier punto del mundo con acceso a la Red. También permite difundir piezas de audio o vídeo a muchos usuarios simultáneamente (SARENET, 2008). Este sistema está orientado a trabajar sobre WEB, por lo que difiere ante la actual propuesta.

La propuesta en cuestión está dirigida a ser desarrollada sobre software libre. Además de esto, la BD en cuestión no está destinada al almacenamiento y gestión de los archivos media en sí, sino que guardará las direcciones físicas de estos así como sus respectivos metadatos, para luego garantizar las actividades u operaciones de la aplicación final. Los archivos físicos estarán guardados en el servidor de medias, y el acceso a estos será a través de la BD.

Explicación del contenido del presente trabajo con una breve explicación de sus partes:

- Capítulo 1 "Fundamentación teórica": contiene lo correspondiente a las tendencias y tecnologías actuales a desarrollar, donde se ofrece una panorámica de las tendencias de las tecnologías relacionadas con las BD, así como su estado del arte a partir de modelos orientados a objetos; tipologías de bases de datos, exponiéndose la diversidad de las mismas, y al final, la selección de la opción que adoptará la presente propuesta; las tecnologías UML y RUP como lenguaje y metodología respectivamente adoptadas en el trabajo, así como la justificación del uso del Sistema Gestor de Base de Datos para el mismo.

- Capítulo 2 "Descripción y análisis de la solución propuesta":

En este capítulo se exponen los siguientes criterios:

  - Estrategia de integración de la solución con otros módulos o sistemas.
  - Selección y argumentación de los requisitos funcionales y no funcionales del sistema propuesto.
  - Modelo de objetos o diagrama de clases persistentes obtenido a partir del diagrama de clases del diseño.
  - Descripción de las clases.
  - Diseño de la BD.
  - Descripción de las tablas.

- Capítulo 3 "Validación del diseño realizado":

- En una primera parte se realizará la validación teórica del diseño: integridad de los datos, normalización de la BD, análisis de redundancia de información, seguridad, y trazabilidad de la acciones.
- Validación funcional será el segundo aspecto en ser abarcado:
  a) Generación de código de programas para un llenado voluminoso e inteligente de la base de datos.
  b) Búsqueda o diseño de herramientas para pruebas de carga intensiva.
- Conclusiones: se hará la valoración de los resultados y propuesta de iteración en el diseño.

CAPÍTULO #1: Fundamentación teórica.

1.1 Introducción

A partir de la década de 1990, las necesidades de almacenamiento en las base de datos se han vuelto más exigentes, y los nuevos sistemas son cada vez más específicos y exquisitos a la hora de utilizar las mismas, haciéndose necesario tratar textos, gráficos, voces y videos; por otra parte, las fronteras físicas para almacenar la información se han ido rompiendo dadas nuevas y más diversas necesidades contemporáneas. Actualmente las BD ofrecen un gran número de funcionalidades para el tratado de la información, sin embargo siguen aumentando las demandas de nuevas operaciones en el manejo de datos, lo cual viene acompañado del desarrollo de nuevas herramientas y técnicas que brindan nuevas alternativas para los usuarios. La intención es explotar la información de una forma más eficiente y cómoda para el usuario, de manera que se creen ventajas competitivas a las organizaciones e instituciones. Lo cierto es que no existe una alternativa universal ni óptima para el manejo de BD, por lo que los desarrolladores actuales simplemente toman, de las opciones que tienen a su alcance, la que más se ajusta a sus necesidades.

1.2 Tipología de Base de Datos

Las bases de datos pueden ser clasificadas desde diferentes puntos de vista teniendo en cuenta diversos criterios de clasificación:

❖ Teniendo en cuenta la variabilidad de los datos almacenados:

- **BD estáticas:** Estas bases de datos son sólo para consultas o lectura, su principal característica es que no se puede modificar la información que está guardada, es decir, sólo se puede consultar. Son utilizadas sobre todo para estudiar el comportamiento de datos históricos y realizar toma de decisiones.

- **BD dinámicas:** en estas bases de datos la información almacenada se modifica con el tiempo, por lo que en ellas están presentes todas las operaciones de gestión (agregar, modificar-actualizar, y eliminar) por lo que la información guardada en ellas sufre cambios constantemente.

❖ De acuerdo al modelo de administración de datos:

- **Modelo orientado a objetos:** Conjuga de forma centralizada los conceptos de abstracción, jerarquía, modularidad, persistencia, tipos y concurrencia. Un modelo

orientado a objetos tiene obligatoriamente que cumplir con lo siguiente: la estructura básica de trabajo son los objetos, no algoritmos; donde cada objeto no es más que una instancia de una clase ya definida y que dichas clases estarán relacionadas únicamente por relaciones de herencia.

- **Modelo relacional:** organiza la información en forma de tablas bidimensionales y de tal forma es que es percibida por los usuarios. Para interactuar con la información en dichas tablas este modelo tiene definido un grupo de operadores, y para construir las consultas a bases de datos relacionales el lenguaje más utilizado es *Structured Query Language* o Lenguaje Estructurado de Consultas (SQL), un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales. Durante el diseño de estas BD se pasa por un proceso conocido como normalización, con el fin de eliminar redundancias y evitar inconsistencias en las tablas.

- **Modelo jerárquico:** Estas bases de datos almacenan su información en una estructura jerárquica, los datos se organizan en una forma similar a un árbol (visto al revés), en donde un nodo padre de información puede tener varios hijos. El nodo que no tiene padres es llamado raíz, y a los nodos que no tienen hijos se los conoce como hojas. Las bases de datos jerárquicas son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos permitiendo crear estructuras estables y de gran rendimiento. Una de las principales limitaciones de este modelo es su incapacidad de representar eficientemente la redundancia de datos.

- **Modelo de red:** este modelo es muy semejante al jerárquico, su principal diferencia consiste en que en este modelo se permite que un hijo o nodo tenga varios padres, lo cual representa una mejora potencial en cuanto al manejo de redundancia de datos, pero el manejo de este modelo en sí es bastante complejo, por lo que generalmente es usado por los programadores en lugar de los usuarios finales.

1.2.1 Elección para la solución en propuesta:

Una base de datos para un sistema de captura y catalogación de medias sufrirá cambios constantemente, por lo que la misma será de tipo dinámica. En cuanto a la tipología, para la solución en propuesta la mejor opción la representa una BD de modelo relacional, ya que otras tipologías como la orientada a objetos imponen

restricciones como la estructura básica (los objetos) y la relación entre estos (sólo por herencia). Además, tipologías como la jerárquica o de modelo de red son de muy complejo entendimiento y sus mecanismos de reducción de redundancia son deficientes. Por otra parte, el modelo relacional brinda las siguientes ventajas:

- Fácil comprensión: al ignorar el almacenamiento físico de los datos, se centra en el modelo lógico de la BD, por lo que su interpretación humana resulta más sencilla.
- Garantiza la integridad referencial: al eliminar un registro elimina todos los registros relacionados dependientes.
- Posee mecanismos de reducción de redundancia de datos: estos mecanismos permiten optimizar la BD y evitar inconsistencias o duplicidades de datos.

1.2.2 Pasos para el diseño de Bases de Datos relacionales

A continuación se ofrece una descripción de algunos pasos a seguir para la confección de bases de datos relacionales.

- **Recolección y análisis de requerimientos:** Las necesidades de información persistente en la base de datos son recogidas y documentadas por los diseñadores de esta, así como cualquier otro requerimiento, regla o necesidad que el(los) usuario(s) de la base de datos plantee.
- **Diseño conceptual:** representa uno de los primeros pasos a realizar en el diseño de bases de datos relacionales. Consiste en crear un esquema conceptual mediante un modelo de datos conceptual de alto nivel.

  El esquema conceptual contiene una descripción detallada de los requerimientos de información de los usuarios, y contiene descripciones de los tipos de datos, relaciones entre ellos y restricciones. Es preciso utilizar, para el diseño de esquemas conceptuales, el modelo E-R (entidad - relación), que describe los datos como entidades, vínculos (relaciones) y atributos.
- **Diseño lógico de la base de datos:** es el siguiente paso en el proceso de diseño. Consiste en implementar la base de datos con un S.G.B.D., transformando el modelo conceptual al modelo de datos empleados por el S.G.B.D. (relacional).

- **Diseño físico de la BD:** En este paso se especifican las estructuras de almacenamiento internas y la organización de los archivos de la base de datos.

1.2.3 Elementos de una base de datos relacional

A continuación se ofrecen una serie de conceptos necesarios para el diseño bases de datos relacionales:

Modelo-Entidad-Relación (MER): también conocido como modelo de datos, es un tipo de diagrama para modelado de bases de datos. Su objetivo es representar relaciones que existen en la vida real entendiendo su semántica. Los cuatro elementos fundamentales de un modelo MER son: las entidades, atributos, interrelaciones y dominio.

Entidad: es el objeto básico representado en el modelo E-R. Representa elementos u objetos de la vida real con existencia necesaria en la base de datos. Una entidad puede ser un objeto con existencia física como: una persona, una máquina, una pieza; o bien puede ser un objeto con existencia conceptual: un curso académico, un departamento, un viaje, o cualquier otro elemento del cual se necesite almacenar información. Éstas deben utilizar nombres en singular preferentemente, y que sean descriptores de la entidad. Se representan por un rectángulo con el nombre de la entidad dentro.

Atributos: es la unidad menor de información sobre un objeto almacenado en la BD. Son propiedades que poseen las entidades, por ejemplo, un color, un nombre, identificador, o cualquier otro dato que ofrezca información sobre la entidad. Se representan mediante elipses, o pequeños círculos, conectadas a la entidad mediante una línea recta (ver Figura 2 y 3).

Dominio: conjunto de valores posibles que puede tomar un atributo determinado.

Interrelación: relación, vínculo o correspondencia entre entidades.

Tipo de interrelación: es la estructura genérica que describe un conjunto de interrelaciones de igual comportamiento (Castaño, y otros, 2000). Por ejemplo, Habita es el tipo de interrelación que relaciona a las entidades PERSONA y CASA. Se representan mediante un rombo con el nombre dentro (Ver Figura 2 y 3).

Llave: atributo (o conjunto de atributos) que identifica a una entidad como un objeto único e irrepetible dentro de la base de datos. Ejemplo: la llave de la entidad PERSONA puede ser el NoCI (número de carnet de identidad); en la entidad PIEZA la llave puede ser el número de serie. Se representa a través de una elipse con el nombre del atributo subrayado dentro, o a través de una pequeña esfera rellena (Figura 2 y 3 respectivamente).

Tipos de atributos:

- Simples o compuestos: los atributos *compuestos* están formados por un conjunto de atributos. Por ejemplo, el atributo "Dirección" está formado por "Calle" y "Número", por tanto es compuesto (Figura 2 y 3). Los *simples,* por el contrario, tienen una estructura modular.

- Multivaluados o monovaluados: los atributos *multivaluados* pueden tener más de un valor para una entidad determinada (por ejemplo: el atributo Color de la entidad UNIFORME es compuesto ya que un uniforme puede tener varios colores en su composición), mientras que los atributos *monovaluados* tienen sólo un valor. Los atributos multivaluados se representan mediante una elipse de trazo doble, mientras que el trazo de los monovaluados es simple (Figura 2). En la Figura 3 se aprecia otra variante para representar estos atributos.

- Derivados: estos son atributos cuyo valor para una entidad puede ser obtenido o calculado a través del valor de otros atributos (por ejemplo: a través del NoCI se puede obtener el atributo FechaNacimiento; a través de los atributos Ancho y Alto se puede calcular el Área de la entidad RECTÁNGULO). Se denotan mediante una elipse de trazo discontinuo (Figura 2). En la Figura 3 se aprecia otra variante para representar dichos atributos.

- Obligatorios y opcionales: los atributos *opcionales* son aquellos que pueden ser faltantes, es decir, son datos que pueden existir o no (Figura 3). Por otra parte, se puede obligar a un atributo a que tome un valor (al menos) del dominio para cada ejemplar de la entidad, es decir, el valor de ese atributo es *obligatorio* y no puede ser nulo (Castaño, y otros, 2000).

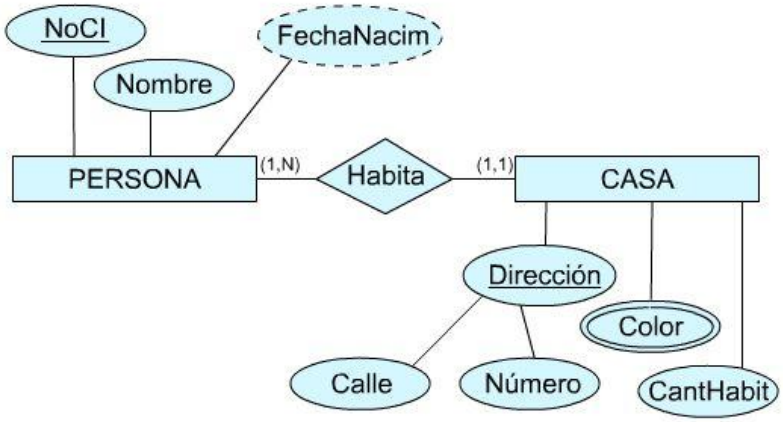


Figura 2: Modelo de Entidad-Relación.

Descripción: Atributo Llave (NoCI y Dirección), multivaluado (Color), compuesto (Dirección), tipo de interrelación (Habita), y derivado (FechaNacim).

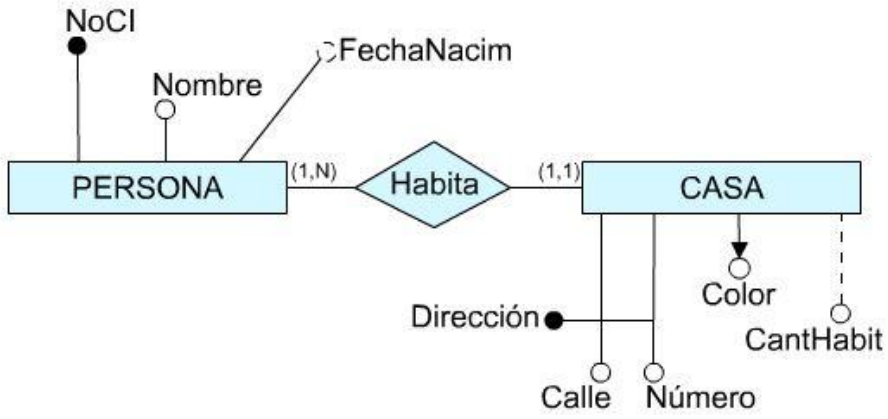


Figura 3: Modelo de Entidad-Relación 2.

Figura 3: Modelo Entidad-Relación 2.

Descripción: Atributo Llave (NoCI y Dirección), multivaluado (Color), compuesto (Dirección), tipo de interrelación (Habita), opcional (CantHabit), y derivado (FechaNacim).

1.2.4 Valoración de normalización y de-normalización de bases de datos

Una de las maneras más naturales y utilizadas de representar datos es la que se basa en las tablas bidimensionales, lo cual es la base del modelo relacional. La normalización es un proceso que permite obtener relaciones de la forma plana bidimensional, logrando que las tablas de datos tengan la forma adecuada, para evitar tanto la ocurrencia de anomalías de actualización como la falta de consistencia de los

datos. Las tablas normalizadas son matrices rectangulares que pueden ser descritas matemáticamente, y poseen las siguientes propiedades:

1. Cada entrada de la tabla representa un ítem de datos; no hay grupos repetitivos.
2. Son homogéneas por columna; es decir, todos los ítems de una columna son de la misma clase.
3. Cada columna tiene nombre propio.
4. Todas las filas son diferentes; no se admiten filas duplicadas.
5. Tanto las filas como las columnas pueden ser consideradas en cualquier secuencia y en cualquier momento, sin afectar por ello ni el contenido de información ni la semántica de cualquier función que utilice la tabla. (Martin, 1977).

Existen varias Formas Normales (FN) definidas: la 1ra FN, la 2da FN, la 3ra FN, la FN de Boyce-Codd, la 4ta y la 5ta FN (esta última se utiliza muy rara vez).

1ra FN: Esta forma normal se refiere a la apariencia de un tipo de registro. Bajo la primera forma normal todas las ocurrencias de un tipo de registro deben contener el mismo número de campos. Esta forma excluye campos y grupos de repetición variable. Las bases de datos relacionales no permiten registros de número variable de campos. Sólo permite valores atómicos: prohíbe que un atributo tenga como valor a un conjunto de relaciones, y prohíbe las "relaciones dentro de relaciones".

Ejemplo:

Producto (Nombre, ID, Precio, PProductor)

Nombre: Nombre del producto.

ID: Identificador del producto.

Precio: Precio del producto.

Productor: País productor del producto.

Tabla 1. 2: Violación de la 1ra FN.

Nombre	Marca	Precio	PProductor
Mouse	ABC	20	China, Japón
Teclado	FDE	100	Suecia
Monitor	NOC	250.50	China

Tabla 1.1: Solución de la 1ra FN.

Nombre	Marca	Precio	PProductor
Mouse	ABC	20	China
Mouse	ABC	20	Japón
Teclado	FDE	100	Suecia
Monitor	NOC	250.50	China

2da FN: Un esquema de Relación R está en 2da FN si todo atributo no primo A de R depende funcionalmente de manera total de la clave primaria de R. Es decir, ninguno depende parcialmente de cualquier clave de R. Cada campo no clave debe proveer un

hecho acerca de la clave entera, y nada sobre un campo no-clave. Esta forma normal es violada cuando un campo no-clave proporciona un hecho sobre un aparte de la clave. Ejemplo:

Tabla 1.3: Violación de la 2da FN.

Pieza	Almacén	Cantidad	Dirección

En este caso la clave está compuesta por Pieza y Almacén juntos, pero el campo dirección (del almacén) proporciona un dato solamente de Almacén. Los problemas que causa este tipo de diseño son los siguientes:

- La dirección del almacén es repetida cada vez que se guarde una pieza en el mismo.
- Al cambiar la dirección de un almacén habría que cambiar cada registro relacionado con una pieza en dicho almacén, lo cual puede crear inconsistencia mostrando un almacén con diferentes direcciones.
- Si en algún momento no hubieran piezas almacenadas en un almacén la dirección de este se pierde.

Para satisfacer la segunda forma normal, el registro antes mostrado debe ser descompuesto en dos registros:

Tabla 1.4: Solución para la 2ra FN.

Pieza	Almacén	Cantidad

Tabla 1.5: Solución para la 2ra FN.

Almacén	Dirección

3ra FN: Esta forma normal se basa en el concepto de dependencia transitiva, y se viola cuando un campo no-clave es un dato sobre otro campo no-clave.
Ejemplo:

Tabla 1.6: Violación de la 3ra FN.

Empleado	Empresa	Dirección-Empresa

Los problemas de este diseño son los mismos que los causados por la violación de la 2da FN. Para satisfacer la 3ra FN se descompone el registro en los dos registros siguientes:

Tabla 1.7: Solución para la 3ra FN

Empleado	Empresa	Empresa	Dirección-Empresa

FN de Boyce-Codd (FNBC): esta forma normal es una refinación de la 3ra FN un poco más estricta (por lo que para estar en la FNBC hay que estar en la 3ra FN), la cual agrega que para toda dependencia X->A (X determina funcionalmente a A) se tiene que X es súper-llave (es llave o contiene una llave). Una relación R está en FNBC si y sólo si cada determinante de las dependencias funcionales es una llave candidata.

4ta FN: trata los hechos con valores múltiples. Bajo la cuarta forma normal, un tipo de registro no debería contener dos o más hechos multi-valor independientes acerca de una entidad. En adición, el registro debe satisfacer la tercera forma normal. Considere el siguiente ejemplo:

Tabla 1. 8: Violación de la 4ta FN.

Empleado	Cargo	Idioma

Donde un Empleado puede tener varios cargos y hablar varios idiomas. En este caso existen dos relaciones muchos a muchos, las cuales deberían ser presentadas de la siguiente forma:

Tabla 1. 9: Solución para la 4ta FN.

Empleado	Cargo	Empleado	Idioma

Los problemas causados por la violación de esta FN son los mismos que presenta la violación de la 2da y 3ra FN, pero además, esto lleva a que las políticas de mantenimiento se hagan con incertidumbre.

5ta FN: esta forma normal se generaliza a casos no cubiertos por las otras formas normales. Trata situaciones donde la información puede ser construida a partir de datos más pequeños. Por ejemplo, si una fábrica hace piezas, y las compañías compran estas piezas, pudiera ser necesario tener un registro cobre cuál fábrica

produce cuál pieza para cuál compañía. Esta información podría ser guardada en un registro de 3 campos:

Tabla 1. 10: Violación de la 5ta FN.

Fábrica	Pieza	CompañíaCliente
Fábrica X	Fuselaje	Boeing
Fábrica X	Alas	ATR
Fábrica Y	Ruedas	Boeing

En este caso la Fábrica X produce fuselajes para la compañía Boeing y alas para ATR, pero esto no quiere decir que produzca alas para Boeing y Fuselajes para ATR, por tal motivo es necesario comprobar los tres campos para saber cuáles combinaciones son válidas y cuáles no lo son. Estos hechos pueden ser reconstruidos en una forma normalizada que mantenga los tres registros por separado:

Tabla 1. 11: solución para la 5ta FN.

Fábrica	Compañía-Cliente

Fábrica	Pieza

Pieza	Compañía-Cliente

Estos tres tipos de registro están en la 5ta FN. Como conclusión, se puede decir que un tipo de registro está en la quinta forma normal cuando su contenido de información no puede ser reconstruido a partir de varios tipos de registro más pequeños.

1.3 Fundamentación de un Sistema Gestor de Base de Datos

Cuando se trabaja con la gestión de la información se hace necesario trabajar de forma lo más sencilla y organizada posible. Es ahí donde entran a jugar los Gestores de Base de Datos. Estos sistemas están destinados a ofrecer mecanismos de gestión de BD con el fin de acelerar el trabajo, aligerarlo y ofrecer mecanismos de automatización de los procesos.

Debido a que el modelo decidido para la base de datos es el relacional no sería práctico entonces analizar algún SGBD para el trabajo con otro modelo de datos, es por eso que la siguiente sección estará dedicada a tecnologías para modelos relacionales solamente.

En la actualidad existe un gran número de SGBD con diferentes tipos de licencia, pero para la propuesta en cuestión se realizará un estudio solamente de las que sean libres o gratuitas dado a que esa es una de las restricciones planteadas para la base de datos a realizar.

SGBD Libres:

- PostgreeSQL: Es un SGBD relacional que incluye características como la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. Posee código abierto y licencia BSD. "El Grupo de Desarrollo Global de PostgreSQL ha liberado la versión 8.0 del Sistema de Gestión de Bases de Datos Objeto-Relacional PostgreSQL, reafirmando su posición como la base de datos de código abierto más avanzada del mundo. Esta versión incluye características antes disponibles solamente en los Sistemas de Gestión de Base de Datos propietarios más costosos." (EEM SYSTEMS, 2005).
- SQLite: es un SGBD relacional, que está contenida en una relativamente pequeña biblioteca en C. Posee código fuente de dominio público libre de: copiar, modificar, vender, publicar, usar, o distribuir el código original. Este sistema difiere de los SGBD cliente-servidor: la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo a través del cual realiza llamadas simples a subrutinas y funciones. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina host.
- Apache Derby: un sub-proyecto de Apache PP, es una base de datos relacional de código abierto aplicado en su totalidad en Java y disponible bajo la licencia Apache, Versión 2.0. Está basado en los estándares de Java, JDBC, y SQL. Derby es fácil de instalar, implementar y utilizar. (Apache Derby, 2007)
- Firebird: es una base de datos relacional que ofrece muchas características de SQL ANSI estándar y que funciona en Linux, Windows, MacOSX, y una variedad de plataformas UNIX. Ofrece una concurrencia excelente, alto rendimiento y un poderoso lenguaje de procedimientos almacenados y disparadores. Firebird es un proyecto independiente de programadores de C y C++, asesores técnicos y colaboradores que lo desarrollan. Basada en código fuente liberado por *Inprise Corp* (ahora conocida como *Borland Software Corp*) en el 2000. (Firebird, 2008)
- MySQL (versiones anteriores): SGBD relacional multiusuario, con más de 100 millones de copias y descargas en todo el mundo, lo cual la convierte en el SGBD más usado hasta el momento. En sus primeras versiones posee licencia GNU

GPL a partir de la versión 3.23.19 para cualquier uso compatible con esta licencia, pero la versión actual ha sido privatizada. Su principal objetivo de diseño fue la velocidad, por lo que en este aspecto es superior a los demás sistemas de su tipo incluyendo el PostgreeSQL. Consume pocos recursos tanto de CPU como de memoria, y posee muy buenas utilidades de administración (backup, recuperación de errores), y aunque se bloquee no suele perder información ni corromper los datos. (MySQL, 2008).

1.3.1 Elección del SGBD para la solución propuesta

Para la solución en propuesta la mejor opción la representa PostgreeSQL. ¿Por qué este? Existen diversos SGBD para soluciones de todo tipo, y sin lugar a duda una de las opciones más estables, utilizadas y robustas la representa PostgreeSQL. A la hora de la elección de una herramienta de trabajo los criterios a favor de una o de otra son variables, diversos y confusos ya que varias de ellas tienen comportamientos similares. La elección de la tecnología adecuada nunca debe decidirse a la suerte o al azar: tiene que haber un análisis previo de las opciones existentes, y partiendo de las necesidades del problema, elegir la(s) que más se ajuste(n) a las necesidades presentes. Por tal motivo, se hace necesario mencionar sus ventajas y desventajas.

Ventajas de PostgreeSQL:

- Posee licencia BSD.
- Instalación ilimitada, lo que se deriva además en**:** modelos de negocios más rentables con instalaciones a gran escala; no existe la posibilidad de ser auditado para verificar cumplimiento de licencia en ningún momento; flexibilidad para hacer investigación y desarrollo sin necesidad de incurrir en costos adicionales de licenciamiento.
- Mejor soporte que los proveedores comerciales.
- Considerables ahorros en costos de operación.
- Gran estabilidad y confiabilidad.
- Extensible gracias a que su código fuente está disponible para todos y sin costos.
- Es multiplataforma.
- Diseñado para ambientes de alto volumen.
- Posee varias herramientas gráficas de alto nivel para diseño y administración de BD.

(PostgreeSQL, 2008)

Desventajas:

Es un sistema robusto, por lo que su consumo de memoria es alto, pero esto se desagravia con las muchas ventajas que ofrece.

Elementos que justifican la elección de PostgreeSQL:

- Posee un diseño para ambientes de alto volumen.
- Instalación ilimitada y flexibilidad a los cambios.
- Sus considerables ahorros en operación, su licencia BSD, rentabilidad en los modelos de negocio, y su extensibilidad son características muy atractivas.
- Implementa el estándar SQL92/SQL99.
- Es multiplataforma (corre en Linux, Windows, y varias versiones de UNIX).
- Su dificultad ante el consumo de recursos es omisible dado a que a cambio PostgreeSQL ofrece una mayor seguridad y confiabilidad en los datos que cualquier otra herramienta de su tipo.
- Alta concurrencia, mediante un sistema denominado MVCC (Acceso Concurrente Multiversión, por sus siglas en inglés). Este permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Esta estrategia es superior al uso de bloqueos por tablas o por filas común en otros gestores. (García, y otros, 2008).
- Soporta diferentes tipos de datos.
- Soporta subconsultas, claves ajenas con chequeo de integridad referencial, *trigguers* y procedimientos del servidor.
- Estrategias preventivas para desastres: posee una estructura adelantada de registros que evita pérdida de datos en caso de falla eléctrica, de sistema operativo o de hardware.
- Posee productos software para el trabajo con el mismo: PgAdmin, PgAcces, Psql, PhpPgAdmin, PgCluster entre otros.
- Utiliza el PgCluster y el Slony-I para la replicación de datos, el primero en réplicas multi-maestro y el segundo en réplicas maestro esclavo.
- Posee un gran soporte brindado por la gran comunidad de usuarios que existe en el mundo que aportan experiencias y resultados obtenidos del uso del mismo.

1.4 El Lenguaje Unificado de Modelado (UML) como soporte de la modelación de la solución propuesta

Definición de UML: (UML por sus siglas en inglés *Unified Modeling Language*) es un lenguaje estándar diseñado para especificar, visualizar, construir y documentar

software orientado a objetos (GRADY, y otros, 2006). No es un proceso, método o metodología, es simplemente un lenguaje de modelado.

Características principales:

- Es un lenguaje de representación visual.
- Permite combinar diversos elementos gráficos y crear diagramas.
- Se usa sólo para modelar sistemas con tecnología orientada a objetos. El UML describe lo que hará un sistema pero no dice cómo implementarlo.

Composición de UML:

UML se compone por elementos se integran entre sí a través de relaciones para formar diagramas (ver Figura 1 Anexo 1).

Tipología de los diagramas UML:

❖Diagramas de estructura estática: Describen las propiedades estructurales del sistema. Por ejemplo: los diagramas de Casos de Uso, de clases, y de objetos.

❖Diagramas de comportamiento: estos son los diagramas de estado, los de interacción (que pueden ser de secuencia o de colaboración) y los diagramas de actividades.

❖Diagramas de implementación: diagramas de componentes y despliegue

Relaciones presentes en los diagramas UML:

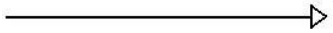
- ❖ Dependencia: esta es una relación entre dos elementos (uno dependiente y uno independiente), mediante la cual se expresa que un cambio en el elemento independiente afectará al elemento dependiente de este. Su estereotipo es una flecha discontinua como la que se muestra a continuación:

- ❖ Asociación: describe conexiones entre objetos. Un tipo de asociación muy común es la agregación. Su estereotipo es el siguiente:

- ❖ Generalización especialización: describe relaciones entre objetos especializados (hijos) que derivan de un objeto más general (padre), lo cual sirve para heredar comportamientos y estructuras con el fin de poder sustituirse entre sí o

especializarse en funciones más específicas (Vilas, 2001). Su estereotipo es el siguiente:



1.5 El Proceso Unificado de Desarrollo de Software (RUP) como metodología para el desarrollo de la solución propuesta

El Proceso Unificado de Desarrollo (RUP) es la metodología de desarrollo más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Es una *metodología robusta* que se combina con UML para la generación del producto final: artefactos, modelos, documentos anexos, ficheros ejecutables, o cualquier otro tipo de fichero que conforme el programa.

RUP define quién (trabajadores) debe hacer qué (artefactos) cómo (actividades) y cuándo (flujo de actividades).

Características de RUP:

- Iterativo e incremental.
- Centrado en la arquitectura.
- Guiado por casos de uso.

Fases de RUP:

RUP posee cuatro fases fundamentales: Inicio (o concepción), Elaboración, Construcción, y Transición, las cuales están divididas en diversos flujos de trabajo. (Ver Figura 2 Anexo 1).

Flujos de trabajo:

En RUP se han agrupado las actividades en 9 grupos lógicos de trabajo: Modelamiento del negocio, Requerimientos, Análisis y diseño, Implementación, Prueba (o testeo), Instalación (estos seis primeros son conocidos como “flujos de ingeniería”), Administración del proyecto, Administración de configuración y cambios, y Ambiente. Los tres últimos flujos con conocidos como “flujos de apoyo”. (Todos estos flujos de trabajo se describen en la Figura 2 del Anexo 1).

1.6 Herramientas de trabajo

Para la correcta elaboración de una base de datos es preciso utilizar los instrumentos apropiados que garanticen agilidad en el proceso de desarrollo, economía de tiempo, así como seguridad e integridad en la información que se trata. Las herramientas de trabajo para bases de datos hoy en día son disímiles y con

diferentes características. Entre estos se encuentran los SGBD mencionados en el capítulo 1.3 del presente trabajo, y otros como ERwin, EasyCASE, y Oracle. Pero un SGBD no es lo único que se emplea, también es necesario utilizar un lenguaje de modelado como el UML (ver capítulo 1.4 del presente trabajo) y alguna herramienta CASE para el trabajo con dicho lenguaje como Visual Paradigm, Rational Rose, BoUML, o CASE Studio. Por último será necesario definir una plataforma de desarrollo para el manejo y administración de la base de datos según el SGBD elegido. A continuación se hará un estudio y elección de las herramientas a utilizar.

1.6.1 Herramientas CASE para UML

- Herramienta CASE Studio:

CASE Studio es una herramienta profesional para el diseño de bases de datos, facilitando herramientas para la creación de diagramas de relación, modelado de datos y gestión de estructuras. Tiene soporte para trabajar con una amplia variedad de SGBD (Oracle, SQL, MySQL, PostgreSQL, Access, etc.) y permite la generación de scripts SQL. También permite realizar ingeniería inversa, usar plantillas de diseño personalizables y crear detallados informes en HTML y RTF. Posee algunas limitaciones, entre ellas: diagramas limitados, y algunas funciones deshabilitadas.

- Herramienta ArgoUML:

ArgoUML es una herramienta CASE escrita en Java y publicada bajo la Licencia BSD. Tiene soporte en varios idiomas, entre los cuales se pueden citar el inglés, alemán, francés, y español; también tiene soporte para el lenguaje de generación de Código: Java, PHP, Python, C++ y Csharp (C#). Permite generar ficheros en diferentes formatos de imagen, así como realizar ingeniería inversa, y soporte para todos los diagramas UML 1.4. Posee los inconvenientes de que no es conforme del todo a los estándares de UML, está incompleto desde la versión 0.2 y no soporta todos los tipos de diagramas del UML 2.0 (como el de secuencia, y colaboración).

- Herramienta BOUML:

Herramienta CASE gratuita (licencia GPL) que permite trabajar con UML 2.0. Entre sus características principales se tiene que: es rápida y su consumo de memoria es muy ligero; soporta una gran variedad de diagramas (incluyendo los que el ArgoUML no soporta); genera código para los lenguajes C++ y Java; genera documentación en varios formatos, y es multiplataforma. Entre los inconvenientes de este software están

que sus posibilidades de generación de código a lenguajes de programación son algo limitadas en comparación con otras herramientas de su tipo, y no permite exportar script de bases de datos.

- Herramienta Visual Paradigm:

Visual Paradigm para UML es una herramienta CASE profesional que soporta el ciclo de vida completo del desarrollo de software. Su licencia está paga por la Universidad de las Ciencias Informáticas. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. Soporta UML versión 2.1, generación de bases de datos (transforma diagramas entidad-relación en tablas) para SGBD como: MySQL, PostgreSQL, Oracle, Sybase, Infomix, BD2 y MS SQL Server. Permite realizar ingeniería inversa de base de datos. Soporta el uso de claves ajenas en el diseño del MER, y automatiza pasos como la creación de la tercera tabla que surge ante una relación de muchos a muchos.

Teniendo en cuenta lo antes descrito, la herramienta Visual Paradigm es la que más se ajusta a las necesidades del proyecto. Por sus características y ventajas resulta la alternativa más atractiva.

1.6.2 Herramientas para diseño y gestión de bases de datos

- Herramienta DBDesigner:

Es una herramienta libre para el diseño y administración de bases de datos, que combina características y funciones profesionales con un diseño totalmente visual. Permite realizar funciones como: crear y modificar tablas y relaciones; gestionar bases de datos para aplicaciones desarrolladas en Java, C++, Visual C#, Visual Basic, Perl o PHP; crear condiciones para garantizar la integridad relacional, así como crear y visualizar diagramas de entidad-relación.

- Herramienta pgAdmin:

Es una de las más utilizadas plataformas de desarrollo para PostgreeSQL. Está diseñada para responder a todo tipo de necesidades, tanto de consultas SQL simples hasta la elaboración de BD complejas. La interfaz gráfica soporta todas las características de PostgreeSQL y es de fácil administración. La aplicación también posee un editor de sintaxis SQL, y la conexión al servidor puede hacerse a través del protocolo TCP-IP o Unix Domain Sockets sin la necesidad de drivers especiales para

la conexión con la BD. Es Open Source, multiplataforma y su versión actual es la 1.8.4. (pgAdmin, 2008). Permite crear y visualizar diagramas de entidad relación. Se puede convinar con la herramienta Visual Paradigm para utilizarlas una en apoyo a la otra. Permite crear condiciones que deben prevalecer para mantener la integridad relacional, establecer y modificar relaciones, crear y generar el código que construye a la base de datos en el servidor, así como la ingeniería inversa.

1.7 Definición de la tecnología de acceso a datos a utilizar

Para la presente solución será necesario la utilización de una base de datos dinámica, ya que los diferentes módulos del sistema SCCM estarán interactuando en todo momento con la misma, por lo que sufrirá cambios constantemente y se realizarán las 3 operaciones básicas de gestión: agregación, actualización y eliminación de la información almacenada.

El hecho de tratarse de almacenamiento de media, de una aplicación distribuida en varias máquinas, y la variabilidad y cantidad de datos que necesitará conocerse acerca de cada media almacenada indican que se trata de una base de datos que alcanzará un gran volumen. Esto quiere decir que será necesario garantizar un buen tratamiento de reducción de redundancia y un modelo de datos capaz de mantener la integridad referencial y entendimiento de la información. Se decide entonces tomar el modelo relacional por todas las ventajas que presenta (descritas en el epígrafe 1.2.1).

Se empleará UML como lenguaje de modelado, ya que este brinda un soporte completo para RUP, así como para el modelado de bases de datos. La herramienta CASE a utilizar para el empleo del mismo será el Visual Paradigm, herramienta que permite modelar los diferentes diagramas y artefactos, puede transformar diagramas entidad-relación en tablas exportando el código y permite el uso de llaves foráneas (o claves ajenas).

El SGBD a utilizar será el PostgreeSQL dadas sus características de robustez, estabilidad, licencia BSD, diseño para ambientes de alto volumen y multiplataforma. Se utilizará el pgAdmin como plataforma de desarrollo ya que es una herramienta diseñada específicamente para el trabajo con PostgreeSQL que permite la administración y mantenimiento de bases de datos de este sistema gestor, además, permite la creación de roles o usuarios para el trabajo sobre la base de datos (es posible definir diferentes usuarios y permisos para establecer la conexión a la base de datos), y se puede utilizar simultáneamente junto con el Visual Paradigm (en este último se puede hacer el modelo-entidad-relación de una forma muy rápida, luego exportar las tablas y administrar la BD desde el pgAdmin).

Conclusiones

En este primer capítulo se ha visto el estado del arte de las tecnologías de datos existentes en el mundo, su tipología, características específicas, ventajas y desventajas de las mismas. El estudio de estos sistemas ha ayudado a la toma de decisiones de la estrategia y tecnología de acceso a datos a utilizar, y ha enriquecido la investigación hecha. Es válido aclarar que, al estudiar las herramientas y tecnologías a utilizar, la preferencia por alguna herramienta u otra en la presente solución no significa la “superioridad” de esa sobre las demás, sino que quiere decir que es la que más se ajusta a las necesidades particulares del trabajo, es decir, no ha sido un estudio comparativo para ver cuál es la mejor herramienta, sino cuál es la que satisface en mayor medida lo que se necesita. El conjunto de software decidido finalmente satisface de forma satisfactoria las necesidades del sistema SCCM, cuentan con las funcionalidades indispensables y ofrecen además un grupo de comodidades que agilizarán el trabajo que prosigue.

Capítulo 2: Descripción y análisis de la solución propuesta.

2.1 Introducción

El presente capítulo tiene una importancia vital para la construcción de la base de datos final, dado a que describe temas como: los requisitos a cumplir por la base de datos, el modelo de objetos o diagrama de clases persistentes obtenido a partir del diagrama de clases del diseño, la estrategia de integración con otros subsistemas, así como la descripción de la arquitectura propuesta.

Luego será desarrollado el diseño lógico y físico de la BD, para obtener a partir de estos las clases finales.

2.2 Estructura general y distribución de la base de datos

Partiendo de la investigación previamente realizada y la experiencia sacada de la misma se propone que la base de datos no radique en un solo sitio, ya que pudieran presentarse problemas como: ataques, inyecciones SQL, colapso de la red, fallo eléctrico, o cualquier otro imprevisto para el cual la base de datos deba prepararse. A continuación se propone la distribución de la base de datos y su estructura.

2.2.1 Servidor principal y reflejo de la base de datos

Con el fin de garantizar una mejor disponibilidad en las bases de datos es que se crean reflejos de la base de datos, que se puede usar conjuntamente con la réplica de la misma. La creación de un reflejo de la BD consiste en la creación de una copia de la base de datos (aparte de la original) que suele residir en un equipo diferente de forma tal que funcione como respaldo en caso de catástrofe, ataque o pérdida de la información. La base de datos principal será la única que estará en cada momento disponible a las máquinas clientes (ver Figura 3 Anexo 1). A la base de datos reflejada (la copia de la original) se le aplicarán las actualizaciones que se le hagan a la principal. Una copia de respaldo (o reflejo) servirá para corregir los datos en caso de irrupciones como: inyección SQL, modificación desautorizada, colapso de la base de datos principal o fallo de hardware en el servidor original.

El reflejo de la base de datos principal será fiel a la original en la misma medida en que los clientes le den importancia a la información almacenada. Si los clientes consideran que una pérdida de información obtenida en un período corto de tiempo es muy importante entonces el reflejo se actualizará con mayor frecuencia. Esta solución no resulta muy necesaria ni muy práctica para el sistema en cuestión ya que implica una mayor necesidad de recursos (se necesitaría un servidor con las mismas características de capacidad y procesamiento que el original), pero aún así representa

una opción a tener en cuenta tanto para este sistema como para otros sistemas futuros con características similares.

2.2.2 Base de datos principal con réplicas

Con el objetivo de que las estaciones clientes no interrumpan sus labores en caso de caída de la red, fallo eléctrico en el servidor central de base de datos, o imposibilidad de conexión con la base de datos, se propone el uso de réplicas locales ubicadas en dichas máquinas clientes (ver Figura 4 Anexo 1). Las réplicas en cada estación serán sólo de la información indispensable, sin la cual el(los) subsistema(s) instalado(s) se vea(n) afectado(s) en su funcionamiento, ya que no es conveniente tener información clasificada dispersa, pues muchas copias en diferentes lugares también representan un mayor número de puntos de posibles ataques. Cuando el sistema se encuentre en pleno funcionamiento trabajará con la BD principal ya que ésta es la que estará más actualizada en todo momento, las réplicas locales serán utilizadas solamente cuando exista alguna dificultad con la BD principal.

Por otra parte se propone el uso de una réplica de la BD principal (lo cual en este caso es una mejor opción que el uso de un reflejo de la misma). Esta réplica no tiene por qué mantener toda la información almacenada en la base de datos principal, sino que tendría sólo la información más relevante. Esto evitaría redundancias innecesarias y reduciría los costos ya que no se necesitaría de un servidor con las mismas características de capacidad y procesamiento que el servidor de base de datos central. Si existiera alguna violación de la seguridad que produjera corrupción de datos (una inyección SQL, un ataque a la integridad de la información, o cualquier otra violación) la información dañada podría recuperarse a partir de la réplica de la BD principal. Para esta solución la distribución quedaría de la siguiente forma: base de datos central con réplica, y pequeñas réplicas en las máquinas clientes que respondan a las necesidades de cada subsistema (Figura 4 del Anexo 1).

2.2.3 Integración con los demás módulos del sistema

Es preciso definir cómo se integrará la base de datos con los demás módulos del sistema, en otras palabras, es necesario concretar cómo será la interacción de los módulos con la base de datos, qué mecanismos de acceso a datos serán utilizados, quién manipulará la información y dónde situar el código específico del origen de datos subyacente.

La arquitectura definida por el arquitecto del sistema es una arquitectura de N capas (4 capas para el caso particular del proyecto en cuestión), donde cada capa

interactuaría únicamente con la capa inmediata a la misma. La capa de Acceso a Datos (AD) se crea con el objetivo de reunir todo el código específico del origen de datos subyacente, incluyendo el código que crea una conexión a la base de datos y que emite los comandos de consulta (*Select*, *Insert*, *Update* y *Delete)*. Esta capa contiene clases que implementan los métodos para tener acceso a los datos de la base de datos, por lo que cualquier otra capa no podría realmente trabajar directamente con los datos, sino que tendría que invocar clases y métodos en la capa AD para hacer las solicitudes. La capa de presentación no trabaja directamente con datos, en su lugar, invoca clases y métodos en la capa de acceso a datos para todas las solicitudes que haga.

Existen disímiles formas de llevar a cabo una capa de acceso a datos, una de ellas es crear una clase por cada tabla en la BD y mapearlas una a una (cada clase identificaría en la BD los datos que maneja y hace una manipulación directa de los mismos), lo cual es una solución fácil y factible. Otra forma es respetar un modelo orientado a objetos que represente los objetos del dominio del negocio, lo cual es una forma más correcta de trabajar, pero es más complejo mapear esas clases a sus tablas correspondientes. Este tipo de labor es llevada a cabo a través de herramientas conocidas como ORM (*Object to Relational Mapper*), en español: Mapeador de Objeto a Relacional. Una de las herramientas más populares de este tipo es el Hibernate, *framework* que se encarga de persistir objetos del negocio, tiene un lenguaje de consultas orientado a objetos propio y es totalmente independiente de la base de datos que se utilice (puede migrar de una base de datos a otra sin necesidad de alterar el código). Esta es la herramienta a utilizar definida por el arquitecto del sistema, y para el uso de la misma será necesario crear archivos de configuración XML que le van a indicar a Hibernate cómo tiene que mapear cada clase a sus respectivas tablas.

De esta forma queda creada una técnica de acceso a la información para los diferentes subsistemas, centrada en la idea de que cualquier interacción de la base de datos será a través de la capa de AD, única manejadora de la misma. Así los módulos del sistema podrán integrarse, y luego sufrir cambios que requieren una mínima alteración en su código o en la base de datos.

2.3 Descripción de los requisitos funcionales y no funcionales del sistema

En general, los requisitos son funcionalidades o condiciones que un sistema informático debe alcanzar o poseer para cumplir con un contrato o documento formal. Éstos pueden ser clasificados en dos tipos: requisitos funcionales y no funcionales. Los primeros son capacidades o condiciones que el producto debe cumplir, y los

requisitos no funcionales son propiedades que el producto debe tener. Aunque en la metodología RUP la base de datos de un sistema se diseña en el flujo de trabajo de Análisis y Diseño, los requerimientos se plantean en la fase de Inicio.

2.3.1 Requisitos funcionales del sistema propuesto

A continuación se describen en lenguaje natural los requisitos funcionales planteados por los analistas del sistema, para dar respuesta a las funcionalidades de los diferentes subsistemas que componen al Sistema de Captura y Catalogación de Medias.

Requisitos del sistema en general

R1 Autenticación de usuarios.

El sistema autenticará a todos los usuarios que hagan uso del mismo, por lo que la base de datos deberá contener la información necesaria de los mismos: usuarios creados, permisos predefinidos para dar acceso (o denegarlo) a los diferentes subsistemas.

R2 trazas de los usuarios.

El sistema gestionará las trazas que serán dejadas por los usuarios luego de utilizar las diferentes acciones sobre el sistema. Se almacenará en la base de datos toda esta información para ser visualizada por el personal pertinente mediante generación de reportes.

R3 Agrupación de usuarios por roles.

Los usuarios con permisos comunes serán agrupados en la base de datos mediante roles que definan sus permisos sobre la aplicación.

Requisitos para el módulo de Administración y Configuración:

R1 Gestionar Usuario.

- Insertar Usuario.
- Modificar Usuario.
- Eliminar Usuario.
- Buscar Usuario.

El subsistema deberá gestionar los usuarios mediante inserción, modificación, eliminación, o búsqueda. La BD debe brindar soporte a toda funcionalidad referente a esta funcionalidad del sistema en dependencia de las necesidades y especificaciones de la aplicación en el momento dado.

R2 Asignación de roles a usuarios.

El subsistema accederá a los datos de los usuarios y modificará (o definirá) los roles de los mismos.

R3 Planificación de Procesos.

En la BD se guardará toda la información correspondiente a la planificación de la captura de video, grabaciones de audio y transcripciones en cada uno de los subsistemas correspondientes, racionalizando los recursos y materiales disponibles para este fin.

R4 Gestionar configuración del servidor de medias.

El subsistema administrará la configuración del servidor de medias y guardará en la base de datos toda la información persistente de la misma.

R5 Gestionar reportes de trazabilidad de usuarios.

Las trazas de todas las operaciones realizadas por los usuarios serán guardadas en la BD, y luego consultadas por este subsistema.

R6 Gestionar las acciones y procesos del sistema.

Se guardará en la BD la información persistente de la gestión de acciones y procesos del sistema, funcionalidad necesaria para darle seguimiento a los procesos de la aplicación controlar las excepciones y errores de manera general, además de controlar también las acciones realizadas en la aplicación.

R7 Generar de reportes de acciones y procesos de la aplicación.

El subsistema obtendrá reportes de la base de datos sobre acciones y procesos de la aplicación a partir de diferentes criterios de búsqueda.

Requisitos del módulo de Captura y Transcripción de Audio

R1 Almacenar audio capturado.

El audio capturado por este subsistema será guardado en una dirección física, creando luego en la base de datos un archivo que conserve la información correspondiente al mismo: dirección física, metadatos, y cualquier otro dato necesario del audio.

R2 Prestación de *streaming* de audio.

El sistema consultará la BD para obtener un audio determinado con el fin de proveer *streaming* de audio desde un servidor central a distintas estaciones clientes,

por lo que la base de datos deberá tener control de todos los archivos almacenados para brindar soporte a esta funcionalidad.

R3 Reproducir audio.

El subsistema buscará en la base de datos los archivos de audio almacenados, seleccionará el deseado lo reproducirá.

R4 Guardar metadatos de ficheros capturados.

La base de datos contendrá los metadatos de todo el audio capturado, la cual será insertada a la misma por el sistema de forma automática.

R5 Guardar transcripciones de audio.

El sistema tendrá un editor de texto incluido que permita al usuario transcribir el audio almacenado, luego estas transcripciones deberán ser almacenadas en la BD.

R6 Gestión de archivos de audio.

El subsistema podrá agregar y eliminar ficheros en el servidor, así como obtener copias de los ficheros de audios que existen en el mismo. En cada acción la base de datos quedará actualizada.

R7 Visualizar reportes.

El usuario podrá visualizar varios reportes de la base de datos atendiendo a determinados criterios de búsqueda.

Requisitos del módulo de Catalogación

R1 Buscar Media

Se realizarán acciones de búsqueda de medias en la base de datos a partir de diferentes criterios de búsqueda.

R2 Reproducir Medias.

El subsistema deberá reproducir medias almacenadas: gestionará el material en la BD desde un servidor de *streaming* y lo visualizará sin necesidad de descargar el fichero en la máquina donde se esté trabajando.

R3 Gestionar ediciones de archivos de media.

La BD contendrá referencias de la media almacenada para su posible futura exportación. Estas ediciones serán gestionadas mediante funciones de: inserción, modificación y eliminación.

R4 Catalogación de Media.

El subsistema visualizará un archivo determinado desde un servidor de *streaming* y luego este archivo media será catalogado y se actualizarán los metadatos referentes al mismo en la base de datos.

R5 Visualizar Reportes a petición de los usuarios.

Se realizarán reportes de las catalogaciones realizadas de acuerdo a los criterios del usuario y se salvarán luego en un formato estándar para su portabilidad.

Requisitos del módulo de Captura e Indexación de Video

R1 Capturar Video

- Captura de Videos desde la planificación

Se capturará video según la planificación hecha previamente. Esta acción dejará un registro en la base de datos para la futura generación de reportes. El video capturado quedará almacenado en el servidor y los metadatos del mismo serán insertados de forma automática.

- Capturar desde PC de Captura

Introducir los datos de captura. Verificar si se puede realizar la misma y ejecutarla. Luego la BD deberá registrar la operación realizada.

R2 Ejecutar Indexación de medias.

- Extraer Fotogramas claves de la media.
- Captar datos del material.
- Fragmentar materiales.

Se extraerán cortes (o fotogramas claves) atendiendo a intervalos de tiempo de un material. Los datos de estos cortes deberán quedar almacenados en la BD además del registro de la acción realizada.

2.3.2 Requisitos no funcionales del sistema propuesto

R1 Apego a estándares internacionales

El sistema deberá estar bajo las normas establecidas en la norma ISO 9241-11 que establece los parámetros internacionales de usabilidad de cualquier software.

R2 Fiabilidad

- Disponibilidad

La base de datos deberá estar disponible siempre que el sistema esté en funcionamiento, pudiendo realizar operaciones sobre la BD en cualquier momento

que se necesite. En caso de una falla en el servidor (interrupción del servicio eléctrico, desconexión momentánea o caída de la red) se deberá contar con un respaldo para cada subsistema que le permita seguir con sus actividades.

- Confiabilidad

La base de datos deberá ser confiable, los datos no deben corromperse durante operaciones del sistema, y los mismos serán manejados de forma segura de modo que sean resistentes ante ataques y desastres.

- Confidencialidad

La información deberá ser manejada sólo por el personal apropiado y autorizado para ello, de modo que cada usuario sólo tenga los permisos indispensables para realizar sus funciones: nunca más que eso para que no puedan realizar acciones indebidas y nunca menos para que no se les afecten los servicios que deben tener.

R2 Rendimiento

La BD deberá resistir una alta concurrencia, así como un gran número de operaciones simultáneas sobre la misma.

R3 Capacidad

La base de datos deberá poder contener un alto volumen de información.

R4 Utilización de recursos

Se necesita un servidor de base de datos con un alto volumen de memoria capaz de soportar a la base de datos durante años.

R5 Herramientas de desarrollo

La base de datos se manejará en una plataforma de desarrollo diseñada para atender a todo tipo de necesidades y que permita la creación de BD complejas. Ésta además deberá ser de fácil administración y multiplataforma. Para la parte de modelación de la arquitectura se utilizará el Visual Paradigm.

Requisitos de Licencia

El SGBD a utilizar deberá poseer alguna licencia libre que permita el uso del software sin restricciones de uso o distribución, al igual que la plataforma de desarrollo para el mismo.

2.4 Definición de los modelos de despliegue

El Modelo de Despliegue se utiliza para representar los elementos de configuración del procesamiento y las conexiones entre esos elementos, y a su vez, para visualizar la distribución de los componentes de software en los nodos físicos. Los elementos que lo componen son los nodos, los dispositivos y los conectores.

El estereotipo para los nodos y para los dispositivos no es el mismo, difieren en que a los primeros se les sombrea el borde y a los segundos no. En el caso de los conectores hay que especificar qué tipo de conexión utilizan, tanto protocolo, vía de entrada, o cualquier otro mecanismo utilizado para conectar los elementos relacionados. A continuación se representa el modelo de despliegue definido para el sistema y la base de datos.

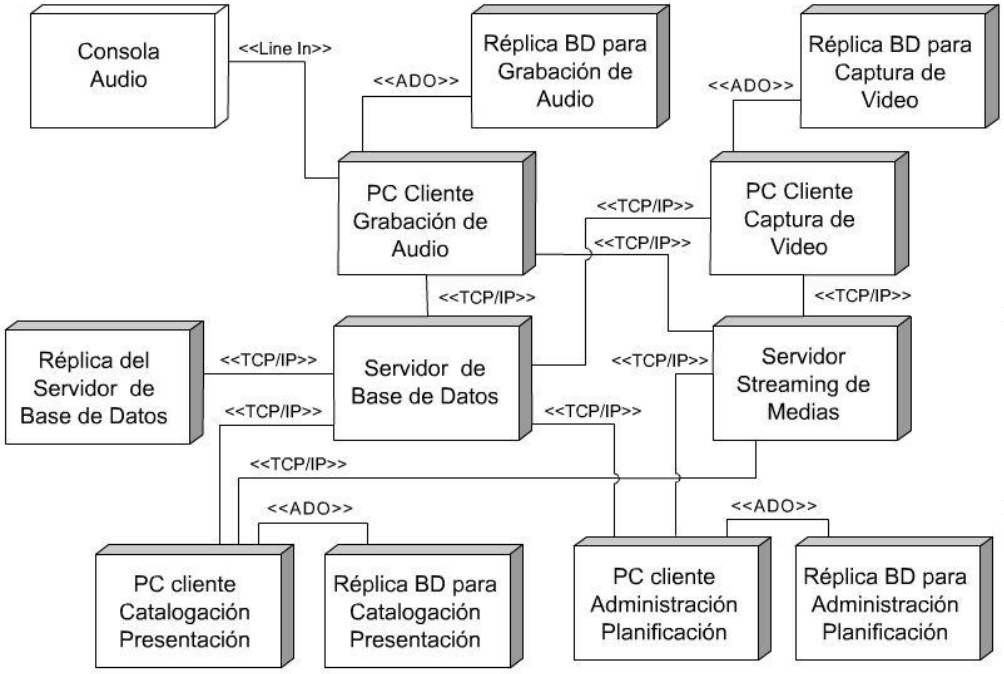


Figura 4: Modelo de despliegue.

Sobre este modelo vale señalar que todos los nodos que representan máquinas clientes tienen una conexión con el servidor de base de datos y otra con el servidor de medias. Por otra parte, cada uno de estos nodos posee su respectiva réplica de la base de datos a través de ADO (*ActiveX Data Object*). El servidor de base de datos y el de media no tendrán contacto alguno ya que a los mismos sólo se accederá a través de los módulos instalados en las máquinas clientes. El servidor de base de datos también tendrá su réplica, lo cual constituye la única conexión que tendrá la misma con algún otro elemento.

2.5 Diseño de la base de datos.

El presente epígrafe estará dedicado al diseño de la base de datos: patrones utilizados, diagramas y modelos así como las descripciones de las tablas obtenidas. A continuación se ofrece una descripción de los patrones de diseño utilizados para la confección del diagrama de clases persistentes y el diagrama de entidad relación de la base de datos.

- Nombre del patrón: **Representación de objetos como tablas.**

Descripción: Este patrón es muy utilizado cuando se quiere representar un objeto en un esquema de base de datos relacional. La solución consiste en definir una tabla para cada clase de objetos persistente. Los atributos de la clase pasan a ser las columnas de la tabla. Debe cumplirse que cada uno de estos atributos sea único para la entidad a la que pertenece, y cada tabla debe contar con un atributo (o grupo de estos) que funcione como identificador o clave de la misma.

- Nombre del patrón: **Representación de relaciones como tablas.**

Descripción: Al diseñar una base de datos relacional se debe tener en cuenta cómo representar las relaciones de la misma, el presente patrón de diseño propone soluciones para los distintos tipos de relaciones que existen en este modelo.

- Para las relaciones uno a uno o uno a muchos:
  1. Para representar la relación entre los objetos colocar una clave ajena en la tabla de cardinalidad uno.
  2. O, crear una tabla asociativa para registrar los identificadores de cada uno de los objetos de la relación.
- Para las relaciones muchos a muchos:
  1. Esta relación dará lugar a una tercera tabla asociativa que registrará los identificadores de cada uno de los objetos de la relación.

2.5.1 Diagrama de clases persistentes.

El diagrama de clases del diseño da lugar a la confección del diagrama de clases persistentes. La persistencia de una clase está definida por la propiedad de los objetos de trascender su estado en el tiempo y el espacio: una clase persistente existirá durante la ejecución de un programa (incluyendo sus versiones), y deberá sobrevivir incluso a la eliminación o colapso del mismo.

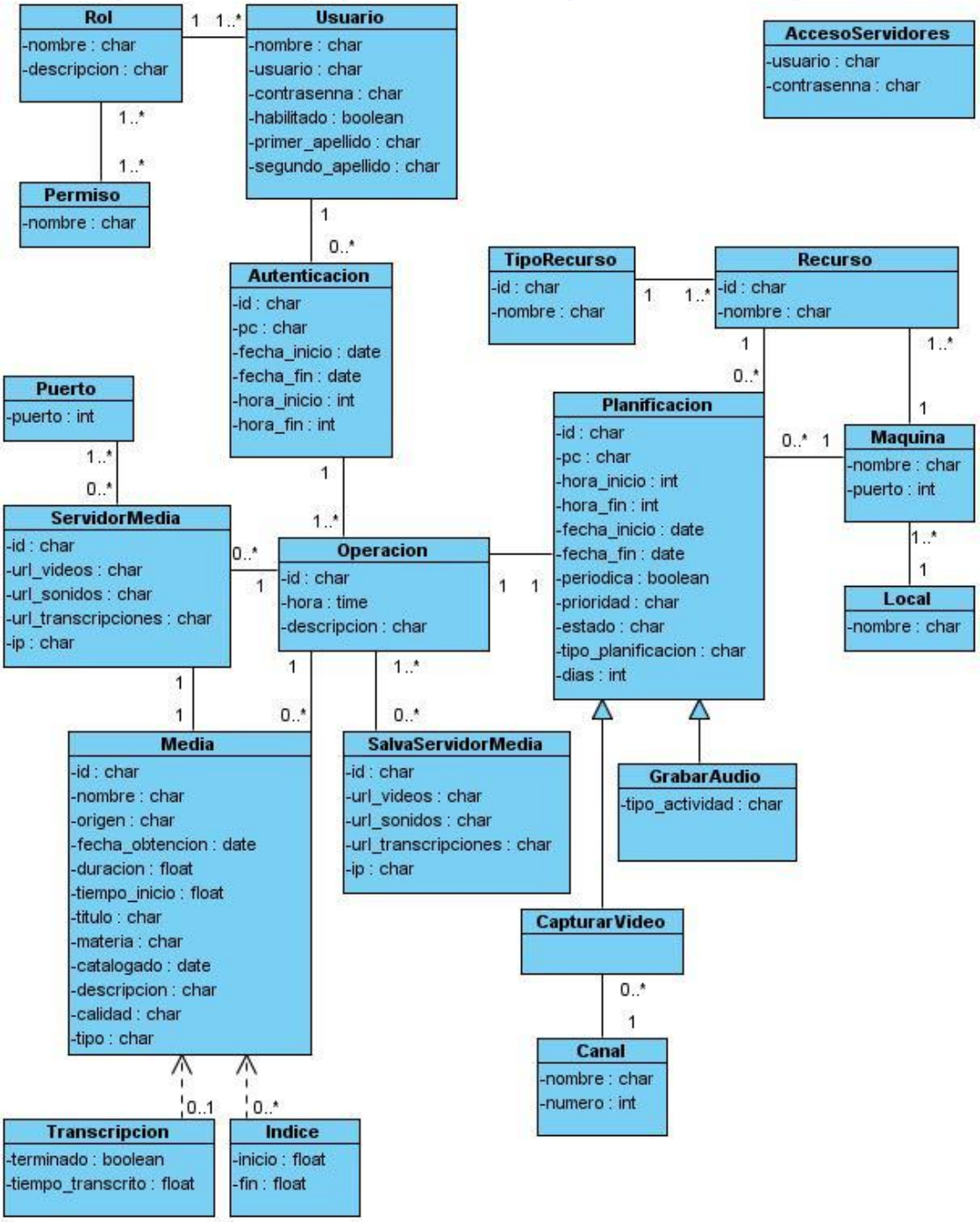


Figura 5: Diagrama de clases persistentes.

2.5.2 Descripción de las clases persistentes

A continuación se ofrece la descripción de las clases persistentes modeladas en el diagrama anterior. En cada descripción se especifica el nombre, atributos, tipo de cada uno de estos, y descripción tanto de la clase como de los atributos. Para la definición de estas clases se evitó el uso de caracteres extraños en sus nombres como vocales con tildes y la letra “ñ”, por tal motivo, en el presente documento aparecerán palabras

sin tilde o con una doble “n” sustituyendo la “ñ”, para respetar así la forma original en que aparecen en el diseño. (Por ejemplo, en lugar de decir “clase Autenticación” dirá “clase Autenticacion”).

Tabla 2. 1: Descripción de la clase Usuario.

Nombre: Usuario		
Descripción: representa los usuarios del sistema		
Atributos	Tipo	Descripción
usuario	Char	Nombre del usuario.
nombre	Char	Nombre del usuario.
contrasenna	Char	Contraseña del usuario.
habilitado	Boolean	Para habilitar-deshabilitar al usuario.
primer_apellido	Char	Primer apellido del usuario.
segundo_apellido	Char	Segundo apellido.

Tabla 2. 2: Descripción de la clase Rol.

Nombre: Rol		
Descripción: Representa el rol que tendrá el usuario.		
Atributos	Tipo	Descripción
nombre	Char	Nombre del rol
descripción	Char	Descripción del usuario

Tabla 2. 3: Descripción de la clase Permiso.

Nombre: Permiso		
Descripción: Representa a cada permiso que tendrán los roles.		
Atributos	Tipo	Descripción
nombre	Char	Nombre del permiso.

Tabla 2. 4: Descripción de la clase Autenticacion.

Nombre: Autenticacion		
Descripción: Registrará los datos de las autenticaciones hechas por los usuarios.		
Atributos	Tipo	Descripción
id	Char	Identificador de la autenticación hecha.
pc	Char	Máquina desde la cual se autentica.
fecha_inicio	Date	Fecha en la que se autentica el usuario.
fecha_fin	Date	Fecha en la que cierra su sesión.
hora_inicio	Int	Hora a la que se autenticó el usuario.
hora_fin	Int	Hora en la que cerró sesión.

Tabla 2. 5: Descripción de la clase Operacion.

Nombre: Operacion

Descripción: Representa las operaciones que se hacen en cada autenticación.		
Atributos	Tipo	Descripción
id	Char	Identificador de la operación.
hora	Date	Hora a la que se hizo la operación.
descripcion	Char	Descripción de la operación realizada.

Tabla 2. 6: Descripción de la clase ServidorMedia.

Nombre: ServidorMedia		
Descripción: Esta clase será la configuración del servidor de medias, contendrá las direcciones físicas en las cuales se guardarán las mismas.		
Atributos	Tipo	Descripción
id	Char	Identificador de la configuración del servidor de medias.
url_videos	Char	Dirección donde se guardarán los videos.
url_sonidos	Char	Dirección donde se guardarán los sonidos.
url_transcripciones	Char	Dirección donde se guardarán las transcripciones de los sonidos.
ip	Char	Dirección IP.

Tabla 2. 7: Descripción de la clase Puerto.

Nombre: Puerto		
Descripción: Cada configuración del servidor de medias tendrá una relación de los puertos por los que hará la transferencia de medias. Esta clase almacena dichos puertos.		
Atributos	Tipo	Descripción
puerto	Int	Puerto por el que se manejarán medias.

Tabla 2. 8: Descripción de la clase SalvaServidorMedia.

Nombre: SalvaServidorMedia		
Descripción: Esta clase surge con el fin de hacer salvas de las configuraciones del servidor de medias.		
Atributos	Tipo	Descripción
id	Char	Identificador de la configuración del servidor de medias.
url_videos	Char	Dirección donde se guardarán los videos.
url_sonidos	Char	Dirección donde se guardarán los sonidos.
url_transcripciones	Char	Dirección donde se guardarán las transcripciones de los sonidos.
ip	Char	Dirección IP.

Tabla 2. 9: Descripción de la clase Media.

Nombre: Media		
Descripción: Esta clase almacenará los datos generales de cada media.		
Atributos	Tipo	Descripción
id	Char	Identificador de la media.
nombre	Char	Nombre de la media.
origen	Char	Origen del que proviene la media.
fecha_obtencion	Date	Fecha en la que se obtuvo la media.
duracion	Float	Tiempo que dura la media.
tiempo_inicio	Float	Tiempo donde comienza la media (pues ésta puede ser un segmento de una media más grande).
titulo	Char	Título de la media.
materia	Char	Materia en la que fue catalogada.
catalogado	Date	Fecha en la que la media fue catalogada. Si este campo fuera nulo significa que no ha sido catalogado todavía.
calidad	Char	Calidad de la media.
descripción	Char	Descripción de la media.
tipo	Char	Tipo de media.

Tabla 2. 10: Descripción de la clase Indice.

Nombre: Indice		
Descripción: Índices que posee una media. La relación de esta clase con la clase "Media" es una dependencia, ya que "Indice" es una entidad débil (no puede existir por sí misma) y depende de que exista una clase "Media" para poder existir.		
Atributos	Tipo	Descripción
inicio	Float	Tiempo donde comienza el índice.
fin	Float	Tiempo donde termina el índice.

Tabla 2. 11: Descripción de la clase Transcripcion.

Nombre: Transcripcion		
Descripción: Representa la transcripción que tendrá una media. La relación de esta clase con la clase "Media" es una dependencia, ya que "Transcripcion" es una entidad débil (no puede existir por sí misma), y al igual que "Indice", depende de que exista una clase "Media" para poder existir.		
Atributos	Tipo	Descripción
terminado	Booleano	Para saber si el audio está o no transcrito.
tiempo_transcrito	Char	Para saber el tiempo en el que se quedó la transcripción (en caso de no estar terminado).

Tabla 2. 12: Descripción de la clase Planificacion.

Nombre: Planificacion

Descripción: Generalización de las planificaciones hechas para el sistema. Contiene los datos generales de las acciones a realizar.		
Atributos	Tipo	Descripción
id	Char	Identificador de la planificación.
pc	Char	Máquina en la cual se realizará la acción planificada.
hora_inicio	Int	Hora planificada para la acción.
hora_fin	Int	Hora en la que deberá terminar la acción.
fecha_inicio	Date	Fecha de inicio de la acción planificada.
fecha_fin	Date	Fecha de terminación de la acción planificada.
prioridad	Char	Define la prioridad que tendrá la acción planificada.
estado	Char	Estado en el que se encuentra la acción planificada (si está en ejecución, en espera, o cualquier otro estado).
tipo_planificacion	Char	Tipología de la acción a realizar.
dias	Char	Días en los que se hará la actividad planificada.

Tabla 2. 13: Descripción de la clase GrabarAudio.

Nombre: GrabarAudio		
Descripción: Es una especificación de planificación. Contiene los datos necesarios para la planificación de una grabación de audio en el sistema.		
Atributos	Tipo	Descripción
tipo_actividad	Char	Tipo de actividad.

Tabla 2. 14: Descripción de la clase CapturarVideo.

Nombre: CapturarVideo		
Descripción: Es una especificación de planificación. Contiene los datos necesarios para una captura de video en el sistema.		
Atributos	Tipo	Descripción

Tabla 2. 15: Descripción de la clase Canal.

Nombre: Canal		
Descripción: Canal por el cual un video puede ser capturado.		
Atributos	Tipo	Descripción
nombre	Char	Nombre del canal.
numero	Entero	Número del canal.

Tabla 2. 16: Descripción de la clase Maquina.

Nombre: Maquina		
Descripción: contendrá los datos de las máquinas disponibles para hacer las planificaciones.		
Atributos	Tipo	Descripción
nombre	Char	Nombre o identificador de la máquina.
puerto	Entero	Puerto a utilizar por la máquina para hacer transferencia de archivos.

Tabla 2. 17: Descripción de la clase Local.

Nombre: Local		
Descripción: clase que contendrá los datos de los locales en los que se encuentran las máquinas.		
Atributos	Tipo	Descripción
nombre	Char	Nombre del local.

Tabla 2. 18: Descripción de la clase Recurso.

Nombre: Recurso		
Descripción: representa los recursos que tendrán disponibles las máquinas dedicadas al sistema.		
Atributos	Tipo	Descripción
id	Char	Identificador del recurso de máquina.
nombre	Char	Nombre del recurso de máquina.
nombre_tipo_recurso	Char	Nombre del tipo de recurso.

Tabla 2. 19: Descripción de la clase TipoRecurso.

Nombre: TipoRecurso		
Descripción: guardará los datos de la tipología de cada recurso de máquina.		
Atributos	Tipo	Descripción
id	Char	Identificador del tipo de recurso.
nombre	Char	Nombre del tipo de recurso.

Tabla 2. 20: Descripción de la clase AccesoServidores.

Nombre: AccesoServidores		
Descripción: se define esta clase con el fin de establecer los datos de acceso de los subsistemas al servidor de medias.		
Atributos	Tipo	Descripción
usuario	Char	Usuario definido en el servidor de medias para la conexión al mismo.
contrasenna	Char	Contraseña establecida en el servidor de

		medias para la conexión al mismo.

2.5.3 Diagrama de entidad relación de la base de datos

A continuación se ofrece la descripción del diagrama de entidad-relación de la base de datos. Debido al tamaño del diagrama original (Figura 6 Anexo 1), el mismo ha sido simplificado en gran medida (eliminando los atributos no claves de las entidades, así como los tipos de variables y otros datos), sólo para mostrar las relaciones que existen entre las entidades con el fin de poder verlas en un solo gráfico. Posteriormente este diagrama ha sido dividido en partes más simples para explicar con más detalle cada una de las entidades y sus atributos.

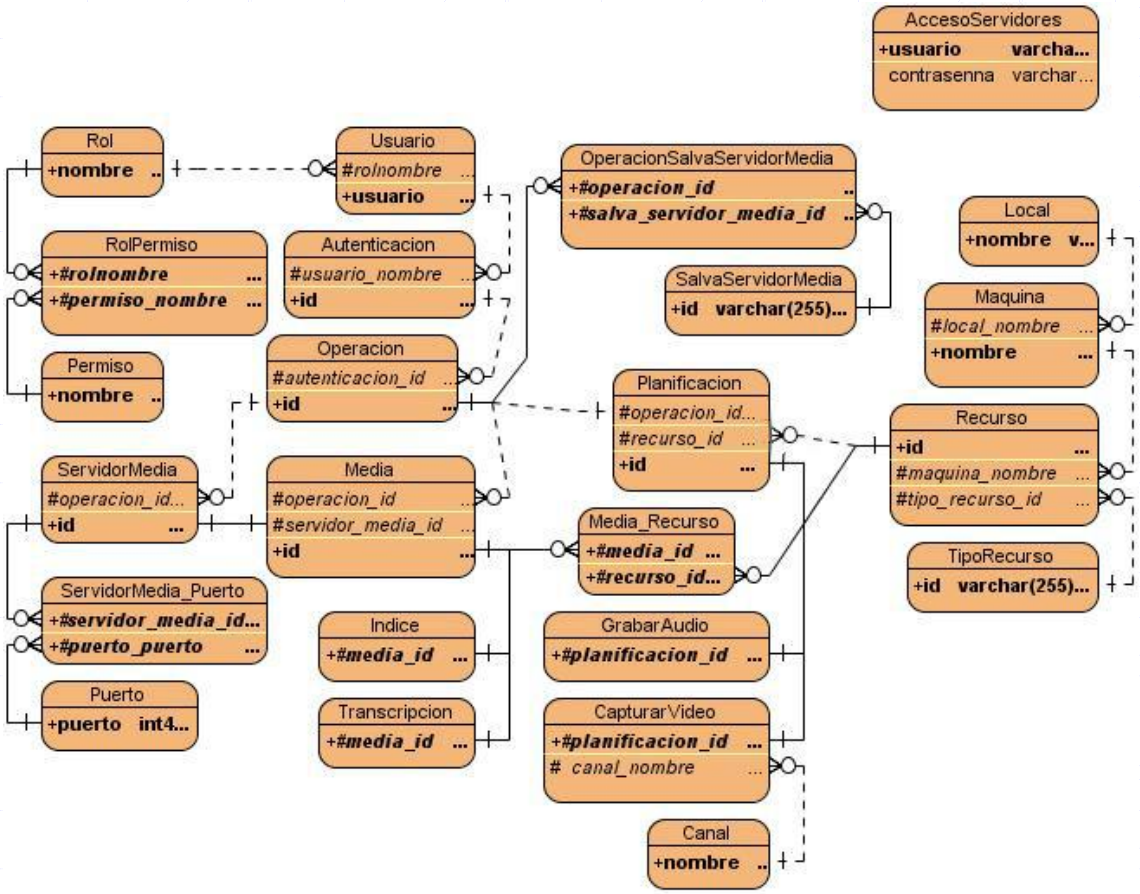


Figura 6: Diagrama de entidad-relación (resumido).

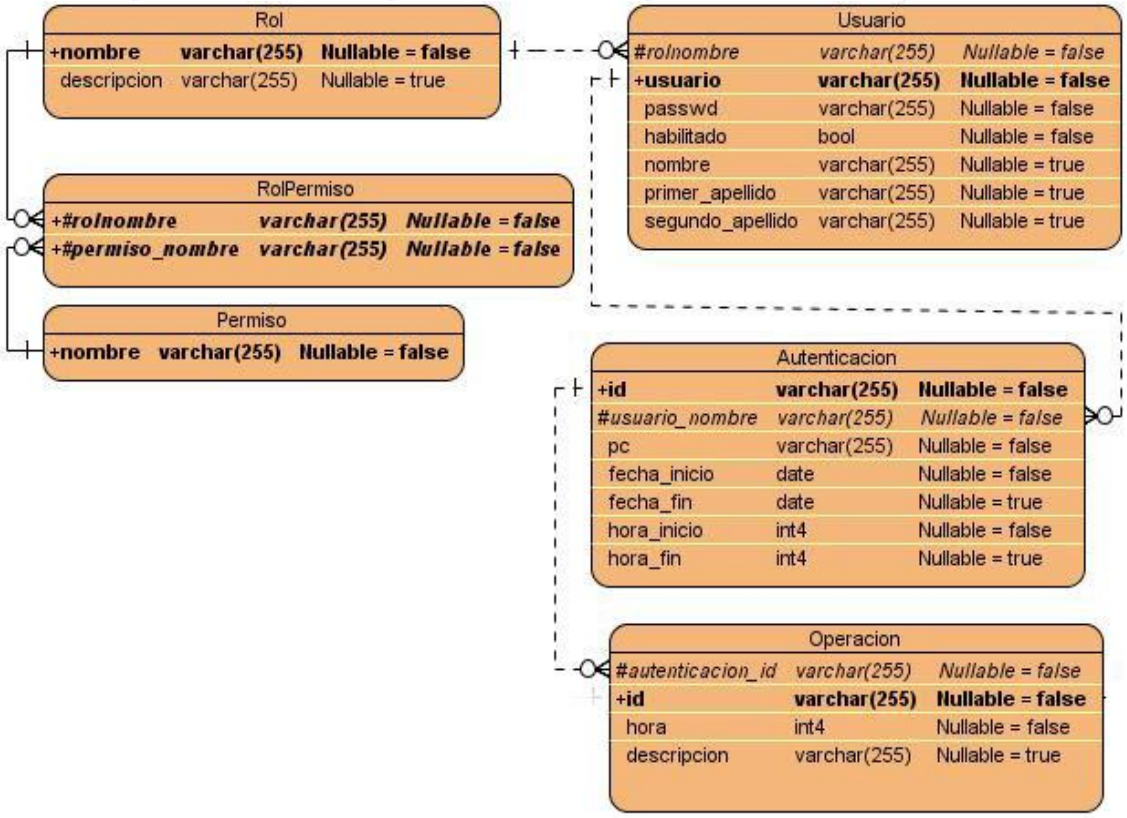


Figura 7: Diagrama de entidad-relación (parte 1).

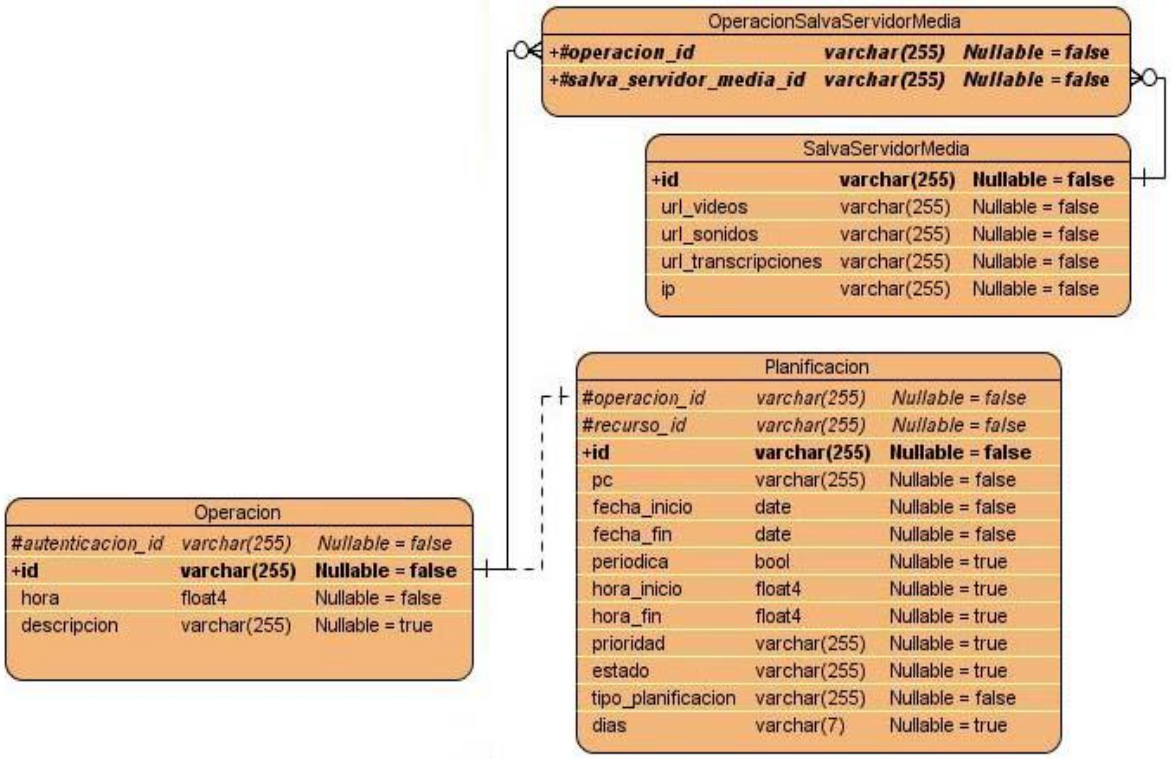


Figura 8: Diagrama de entidad-relación (parte 2).

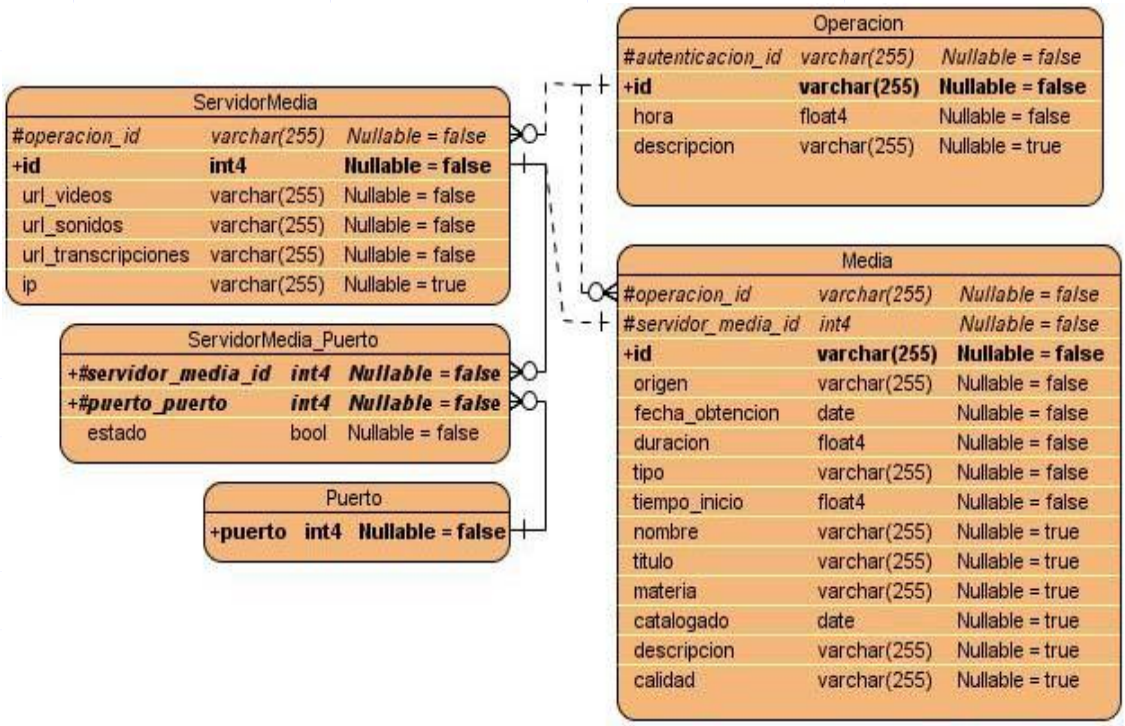


Figura 9: Diagrama de entidad-relación (parte 3).

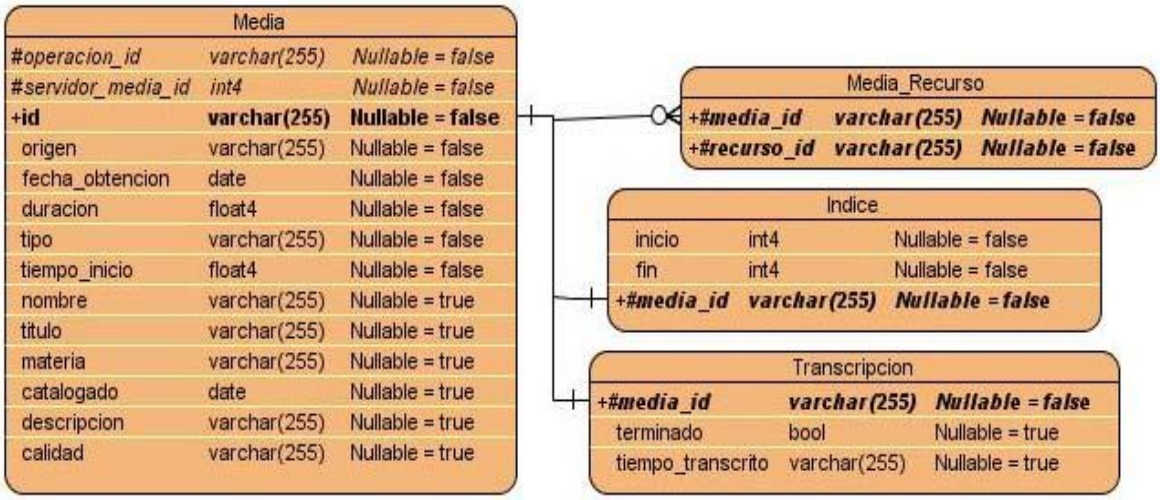


Figura 10: Diagrama de entidad-relación (parte 4).

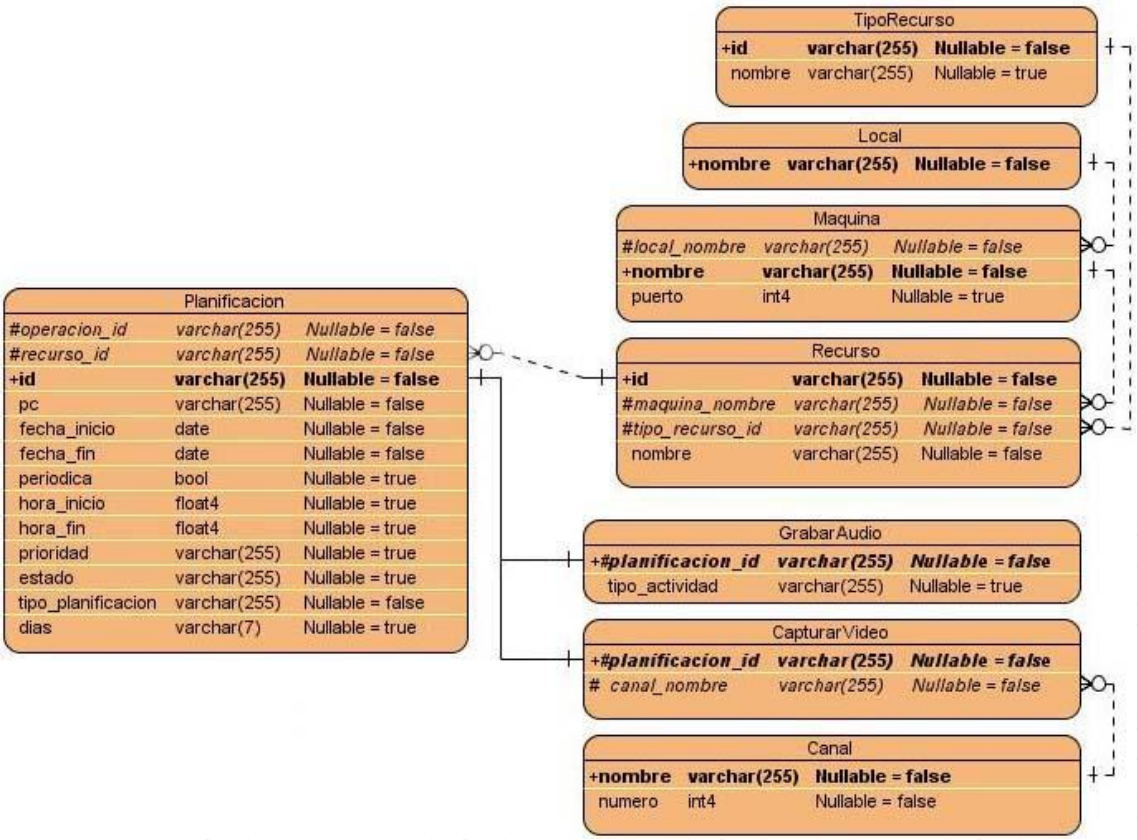


Figura 11: Diagrama de entidad-relación (parte 5).

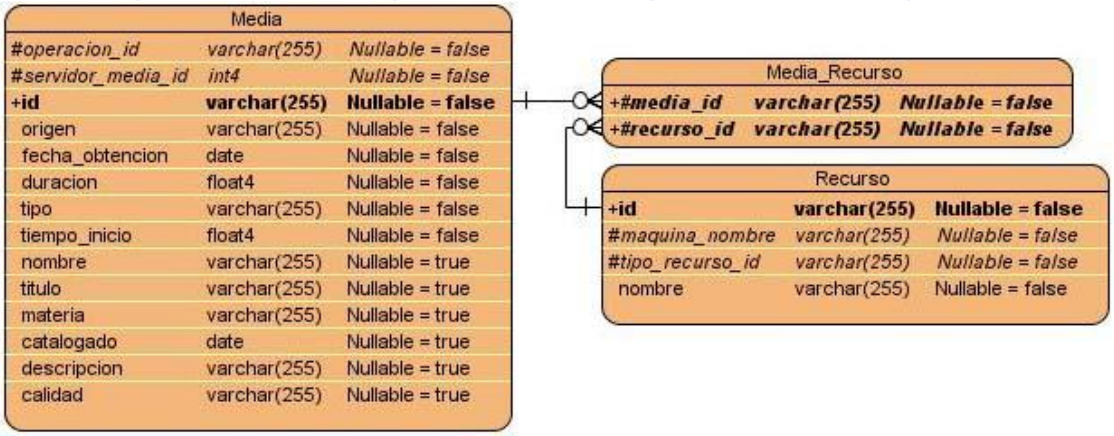


Figura 12: Diagrama de entidad-relación (parte 6).

Figura 13: Diagrama de entidad-relación (parte 7).

2.5.4 Descripción de las tablas del diagrama entidad-relación

Tabla 2. 21: Descripción de la tabla AccesoServidores.

Nombre: AccesoServidores		
Descripción: se define esta tabla con el fin de almacenar los datos de acceso al servidor de medias.		
Atributos	Tipo	Descripción
usuario	Char	Usuario definido en el servidor de medias para la conexión al mismo. Llave de la entidad.
contrasenna	Char	Contraseña establecida en el servidor de medias para la conexión al mismo.

Esta tabla está independiente porque no guarda relación alguna con ninguna de las demás entidades, su objetivo es asegurar que las políticas de seguridad del servidor de medias sean llevadas a cabo sin perjudicar el acceso de los subsistemas a ésta: que pueda cambiar el usuario o contraseña de acceso al servidor de medias sin tener que realizar cambios en los módulos.

Tabla 2. 22: Descripción de la tabla Rol.

Nombre: Rol		
Descripción: tabla que almacena los roles de los usuarios del sistema.		
Atributos	Tipo	Descripción
nombre	Char	Nombre del rol. Llave de la entidad.
descripcion	Char	Descripción del rol.

Tabla 2. 23: Descripción de la tabla Permiso.

Nombre: Permiso		
Descripción: tabla que almacena los permisos de los roles.		
Atributos	Tipo	Descripción
nombre	Char	Nombre del permiso. Llave de la entidad.

La relación que existe entre la entidad “Rol” y la entidad “Permiso” es de muchos a muchos (un rol puede tener varios permisos y un permiso puede estar asignado a varios roles), por lo que surge una nueva tabla: RolPermiso.

Tabla 2. 24: Descripción de la tabla RolPermiso.

Nombre: RolPermiso		
Descripción: relaciona a un rol con un permiso determinado.		
Atributos	Tipo	Descripción
rol_nombre	Char	Nombre del rol (llave primaria de la entidad Rol).
permiso_nombre	Char	Nombre del permiso (llave primaria de la entidad Permiso).

Tabla 2. 25: Descripción de la tabla Usuario.

Nombre: Usuario		
Descripción: Tabla que recoge los datos de los usuarios.		
Atributos	Tipo	Descripción
usuario	Char	Nombre del usuario. Llave de la entidad.
rol_nombre	Char	Clave ajena para conocer el rol del usuario.
nombre	Char	Nombre del usuario.
contrasenna	Char	Contraseña del usuario.
habilitado	Booleano	Para habilitar-deshabilitar al usuario.
primer_apellido	Char	Primer apellido del usuario.
segundo_apellido	Char	Segundo apellido

Tabla 2. 26: Descripción de la tabla Autenticacion.

Nombre: Autenticacion		
Descripción: Registrará los datos de las autenticaciones hechas por los usuarios en el sistema con el fin de dejar rastros de sus acciones en el mismo.		
Atributos	Tipo	Descripción
id	Char	Identificador de la autenticación hecha. Llave de la entidad.
usuario_nombre	Char	Llave ajena utilizada para conocer el usuario que hace la autenticación.
pc	Char	Máquina desde la cual se autentica.
fecha_inicio	Date	Fecha en la que se autentica el usuario.
fecha_fin	Date	Fecha en la que cierra su sesión.
hora_inicio	Int	Hora a la que se autenticó el usuario.
hora_fin	Int	Hora en la que cerró sesión.

La clave ajena usuario_nombre surge a partir de la relación de uno a muchos entre las entidades “Usuario” y “Autenticacion” respectivamente (un usuario puede hacer varias autenticaciones, y una autenticación solamente puede ser realizada por un usuario).

Tabla 2. 27: Descripción de la tabla Operacion.

Nombre: Operacion		
Descripción: Guarda las operaciones que se hacen en cada autenticación.		
Atributos	Tipo	Descripción
id	Char	Identificador de la operación. Llave de la entidad.
autenticacion_id	Char	Llave ajena que guarda la autenticación que realizó la operación.
hora	Int	Hora a la que se hizo la operación.
descripción	Char	Descripción de la operación realizada.

La clave ajena autenticacion_id surge a partir de la relación de uno a muchos entre las entidades “Autenticación” y “Operacion” respectivamente (una autenticación puede hacer varias operaciones, y una operación específica solamente puede ser realizada por una autenticación), de esta forma se puede hacer un seguimiento de todas las operaciones que realizan los usuarios durante una autenticación.

Tabla 2. 28: Descripción de la tabla Media.

Nombre: Media		
Descripción: Representa la generalización de las medias. Esta tabla almacenará los datos generales de cada media.		
Atributos	Tipo	Descripción
operacion_id	Char	Identificador de la media. Llave de la entidad.
servidor_media_id	Char	Llave ajena que guarda el identificador de la configuración del servidor de medias, la cual es utilizada para guardar la media nueva.
id	Char	Identificador de la media. Llave de la entidad.
nombre	Char	Nombre de la media.
origen	Char	Origen del que proviene la media.
fecha_obtencion	Date	Fecha en la que se obtuvo la media.
duracion	Float	Tiempo que dura la media.
tiempo_inicio	Float	Tiempo donde comienza la media (pues ésta puede ser un segmento de una media más grande).
titulo	Char	Título de la media.
materia	Char	Materia en la que fue catalogada.
catalogado	Date	Fecha en la que la media fue catalogada. Si este campo fuera nulo significa que no ha sido catalogado todavía.
calidad	Char	Calidad de la media.
descripción	Char	Descripción de la media.
tipo	Char	Tipo de media.

Tabla 2. 29: Descripción de la tabla ServidorMedia.

Nombre: ServidorMedia		
Descripción: Esta tabla contendrá la configuración del servidor de medias.		
Atributos	Tipo	Descripción
id	Char	Identificador de la configuración del servidor de medias. Llave de la entidad.
operacion_id	Char	Llave ajena que guarda la operación que configuró el servidor de medias.
url_videos	Char	Dirección donde se guardarán los videos.
url_sonidos	Char	Dirección donde se guardarán los sonidos.
url_transcripciones	Char	Dirección donde se guardarán las transcripciones de los sonidos.

ip	Char	Dirección IP del servidor de medias.

Tabla 2. 30: Descripción de la tabla SalvaServidorMedia.

Nombre: SalvaServidorMedia		
Descripción: Esta tabla recoge las salvas de las configuraciones del servidor de medias.		
Atributos	Tipo	Descripción
id	Char	Identificador de la configuración del servidor de medias.
url_videos	Char	Dirección donde se guardarán los videos.
url_sonidos	Char	Dirección donde se guardarán los sonidos.
url_transcripciones	Char	Dirección donde se guardarán las transcripciones de los sonidos.
ip	Char	Dirección IP del servidor de medias.

Como una operación puede hacer varias salvas, y a su vez, una salva puede ser modificada por varias operaciones, surge una nueva tabla: "OperacionSalvaServidorMedias" (la cual es el resultado de una relación de muchos a muchos).

Tabla 2. 31: Descripción de la tabla OperacionSalvaServidorMedias.

Nombre: OperacionSalvaServidorMedias		
Descripción: Esta tabla guarda la relación entre una operación y una salva específica.		
Atributos	Tipo	Descripción
operación_id	Char	Clave de la entidad Operación.
salvaservidormedia_id	Char	Clave de la entidad SalvaServidorMedia.

Tabla 2. 32: Descripción de la tabla Puerto.

Nombre: Puerto		
Descripción: Esta tabla guarda los puertos disponibles para realizar operaciones sobre el servidor de medias.		
Atributos	Tipo	Descripción
puerto	Int	Puerto por el que se hará la conexión al servidor de medias.

Tabla 2. 33: Descripción de la tabla ServidorMediaPuerto.

Nombre: SalvaServidorMedia		
Descripción: Esta tabla recoge las salvas de las configuraciones del servidor de medias.		
Atributos	Tipo	Descripción
servidor_media_id	Char	Identificador del servidor de medias. Pasa

		como llave por la relación de muchos a muchos entre las entidades “Puerto” y “ServidorMedia”.
puerto_puerto	Int	Puerto por el que se hará la conexión al servidor de medias. Pasa como llave por la relación de muchos a muchos entre las entidades “Puerto” y “ServidorMedia”.
estado	Char	Estado en que se encuentra el puerto de un servidor de medias específico.

Tabla 2. 34: Descripción de la tabla Indice.

Nombre: Indice		
Descripción: Recoge los datos de los índices extraídos a videos existentes.		
Atributos	Tipo	Descripción
media_id	Char	Identificador de un índice del video. Llave de la entidad.
inicio	Int	Marca el inicio del índice.
fin	Int	Marca el fin del índice.

Las los índices son entidades débiles (no pueden existir por sí mismos), dependen de que exista una “Media” para poder existir como entidad, ya que para que un índice exista realmente, tiene que tener una media a la cual pertenecer.

Tabla 2. 35: Descripción de la tabla Transcripcion.

Nombre: Transcripcion		
Descripción: Guarda los datos de la transcripción que tendrá cada sonido una vez que sea transcrito.		
Atributos	Tipo	Descripción
media_id	Char	Clave ajena. Es el identificador que identifica cuál transcripción le corresponde a un audio específico.
terminado	Booleano	Para saber si la transcripción está terminada.
tiempo_transcrito	Float	Indica hasta dónde ha sido trascrito en audio (en caso de no estar terminada la trascripción).

Las transcripciones, de la misma forma que los índices, son entidades débiles (no pueden existir por sí mismas), dependen de que exista una “Media” para poder existir como entidad, ya que para que una transcripción exista realmente, tiene que tener una media a la cual pertenecer.

Tabla 2. 36: Descripción de la tabla Planificacion.

Nombre: Planificacion
Descripción: Generalización de las planificaciones hechas para el sistema. Contiene los datos generales de las acciones a realizar por el sistema.

Atributos	Tipo	Descripción
id	Char	Identificador de la planificación. Llave de la entidad.
operacion_id	Char	Clave ajena. Se refiere a la operación que realizó la planificación.
recurso_id	Char	Recurso planificado.
maquina_nombre	Char	Máquina designada para realizar la acción planificada. Clave ajena.
pc	Char	Máquina en la cual se realizará la acción planificada.
hora_inicio	Int	Hora planificada para la acción.
hora_fin	Int	Hora en la que deberá terminar la acción.
fecha_inicio	Date	Fecha de inicio de la acción planificada.
fecha_fin	Date	Fecha de terminación de la acción planificada.
prioridad	Char	Define la prioridad que tendrá la acción planificada.
estado	Char	Estado en el que se encuentra la acción planificada (si está en ejecución, en espera, o cualquier otro estado).
tipo_planificacion	Char	Tipología de la acción a realizar.
dias	Char	Días en los que se hará la actividad planificada.

Una planificación se hace para una sola máquina, mientras que una máquina puede tener varias planificaciones, por eso es necesario tener la llave ajena "maquina_nombre" en la entidad "Planificacion".

Tabla 2. 37: Descripción de la tabla Maquina.

Nombre: Maquina		
Descripción: contendrá los datos de las máquinas disponibles para hacer las planificaciones.		
Atributos	Tipo	Descripción
nombre	Char	Nombre o identificador de la máquina.
local_nombre	Char	Local en que se encuentra la máquina. Llave ajena de la tabla "Local" por su relación de uno a muchos con esta entidad.
puerto	Entero	Puerto a utilizar para hacer transferencia de archivos.

Tabla 2. 38: Descripción de la tabla Local.

Nombre: Local		
Descripción: almacena los locales en los que se encontrarán las máquinas.		
Atributos	Tipo	Descripción
nombre	Char	Nombre e identificador del local.

Tabla 2. 39: Descripción de la tabla GrabarAudio.

Nombre: GrabarAudio		
Descripción: Es una especialización de planificación. Contiene los datos necesarios para la planificación de una grabación de audio en el sistema.		
Atributos	Tipo	Descripción
planificacion_id	Char	Identificador de la planificación de grabación de audio.
tipo_actividad	Char	Tipo de actividad.

Tabla 2. 40: Descripción de la tabla CapturarVideo.

Nombre: CapturarVideo		
Descripción: Es una especificación de planificación. Contiene los datos necesarios para la captura de un video en el sistema.		
Atributos	Tipo	Descripción
planificacion_id	Char	Identificador de la planificación de captura de video.
canal_nombre	Char	Canal por el cual se realizará la captura.

Tabla 2. 41: Descripción de la tabla Canal.

Nombre: Canal		
Descripción: Canal por el cual un video puede ser capturado.		
Atributos	Tipo	Descripción
nombre	Char	Nombre del canal.
numero	Int	Número del canal.

Tabla 2. 42: Descripción de la clase Recurso.

Nombre: Recurso		
Descripción: representa los recursos que tendrán disponibles las máquinas dedicadas al sistema.		
Atributos	Tipo	Descripción
id	Char	Identificador del recurso de máquina.
maquina_nombre	Char	Llave foránea. Especifica la máquina a la que pertenece el recurso.
tipo_recurso_id	Char	Llave ajena proveniente de la tabla “Recurso”.
nombre	Char	Nombre del recurso de máquina.

Tabla 2. 43: Descripción de la tabla TipoRecurso.

Nombre: TipoRecurso
Descripción: guardará los datos de la tipología de cada recurso de máquina.

Atributos	Tipo	Descripción
id	Char	Identificador del tipo de recurso.
nombre	Char	Nombre del tipo de recurso.

Tabla 2. 44: Descripción de la tabla MediaRecurso.

Nombre: MediaRecurso		
Descripción: tabla que relaciona a un "Recurso" con una "Media"		
Atributos	Tipo	Descripción
media_id	Char	Identificador de la Media.
recurso_id	Char	Identificador del Recurso.

Esta tabla surge de la relación de muchos a muchos entre "Recurso" y "Media".

2.5.5 Conclusiones

En el presente capítulo se ha planteado y descrito el diseño de la base de datos. Las entidades reflejadas en los diagramas se traducen en tablas bidimensionales (en correspondencia con lo que plantea el modelo relacional) y dichas tablas han sido descritas en detalle, así como sus respectivos atributos y el tipo de dato a almacenar en cada caso. El diseño realizado cubre todos los requisitos funcionales planteados para la base de datos por parte de cada uno de los módulos del sistema. En el capítulo siguiente se procederá a realizar la validación de dicho diseño.

Capítulo 3: Validación del diseño realizado.

3.1 Introducción

En el capítulo anterior se obtuvo una propuesta de solución para el diseño de la base de datos del sistema Solución de Captura y Catalogación de Medias, y las buenas prácticas de diseño de bases de datos proponen una validación teórica y funcional de dicha solución. Será necesario entonces, en el presente capítulo, realizar un análisis de la redundancia de la información, la integridad de los datos, normalización, análisis de la integridad de la información, así como pruebas de la base de datos que dejen constancia del buen funcionamiento de la misma.

3.2 Validación teórica del diseño realizado

3.2.1 Integridad de la base de datos

En este aspecto se pueden señalar varios tipos de parámetros a tener en cuenta para evaluar cualitativamente la integridad del diseño realizado como: integridad de los datos, de la clave, del dominio, referencial, e integridad semántica. Los análisis de estos parámetros antes mencionados son descritos a continuación.

- **Integridad de datos:**

Cuando las bases de datos con accedidas a través de sentencias de tipo *INSERT*, *DELETE* o *UPDATE*, la integridad de los datos queda vulnerable a sucesos no deseados como:

- Inserción de datos no válidos.
- Modificación de datos con valores incorrectos.
- Cambios aplicados de forma incompleta.
- Pérdida de datos debido a un fallo inesperado (fallo eléctrico, de hardware o de cualquier otro tipo).

En este aspecto PostgreSQL provee confiabilidad, ya que posee escritura adelantada de registros (WAL) para evitar pérdidas de datos en caso de falla de energía, sistema operativo o de hardware. Una de las funciones importantes de un modelo de datos relacional es preservar la integridad de sus datos almacenados en la mayor medida posible (García, y otros, 2008).

- **Integridad de la clave:**

Ningún atributo de una clave candidata toma valores nulos, además, sus valores son únicos para cada tupla. Se garantizan a través del uso de restricciones *PRIMARY KEY*.

- **Integridad del dominio**

Se refiere a la integridad de los datos para una entrada en una determinada columna, es decir, restringe los valores que puede tomar un atributo respecto a su dominio. Este tipo de integridad ha sido cubierta mediante las restricciones *CHECK*, *FOREIGN KEY*, definiciones *DEFAULT*, definiciones *NOT NULL*, y reglas. Algunos ejemplos se muestran a continuación:

Crear tabla “AccesoServidores”

```
CREATE TABLE AccesoServidores (
    usuario character varying(255) NOT NULL,
    contrasenna character varying(255) NOT NULL
);
```

Crear tabla “CapturarVideo”

```
CREATE TABLE CapturarVideo (
    planificacion_id varchar(255) NOT NULL,
    canal_nombre varchar(255) NOT NULL,
    PRIMARY KEY (planificacion_id),
    FOREIGN KEY (planificacion_id) REFERENCES Planificacion (id)
);
```

- **Integridad referencial**

A través de este tipo de integridad se garantiza que una entidad siempre se relacione con otras entidades válidas. Implica además que los datos sean correctos, sin duplicaciones, pérdida de datos o relaciones mal resueltas. Las bases de datos de tipo relacionales incluyen esta propiedad, pues el software gestor (en este caso el PostgreSQL) se encarga de su estricto cumplimiento. A continuación, un ejemplo:

Definición de una *FOREIGN KEY* (llave foránea) para la tabla “Operacion”

```
ALTER TABLE ONLY operacion
    FOREIGN KEY (autenticacion_id) REFERENCES autenticacion(id);
```

El atributo “autenticacion_id” de la tabla “Operación”, es una llave foránea que proviene de la llave primaria “id” de la tabla “Autenticacion”, lo cual garantiza la relación entre estas dos tablas de forma tal que, para que exista una operación, primero debe existir una autenticación que la realice. Por otra parte se cumple que ambos campos poseen el mismo dominio para ambas tablas, y el identificador “id” debe estar en su tabla origen de forma obligatoria.

- **Integridad semántica**

Sobre los dominios, o sobre los atributos, se pueden definir restricciones al diseñar una base de datos, pero estas restricciones pueden ser violadas por algunas operaciones. PostgreSQL permite restricciones de varios tipos como: de unicidad, no nulas, de actualización en cascada, entre otras restricciones, a través de una declaración adecuada y con una gran flexibilidad. El gestor puede fácilmente controlar las transacciones y detectar violaciones de integridad, ya que estas reglas de integridad se almacenan en el diccionario como parte adicional de la descripción de los datos (conocido como "control centralizado de la semántica"), lo cual las hace muy entendibles y manejables, facilitando su mantenimiento.

3.2.2 Normalización de la base de datos

En el epígrafe 1.2.4 se hizo un análisis del tema referente a normalización de bases de datos, describiendo su importancia, necesidad de la misma, y las diferentes formas normales que existen hasta el momento para bases de datos relacionales. La presente solución será llevada hasta la tercera forma normal.

- Análisis sobre la Primera Forma Normal:

La primera FN sólo admite valores atómicos para los campos de la BD. En ninguna de las tablas definidas existen atributos multivaluados, compuestos, o de cualquier otro tipo que pudieran anular esta condición. Los datos que se refieren a una fecha han sido llevados a tipo "Date", lo que trata a dicho valor como un elemento atómico y no como uno compuesto; los datos que se refieren a un tiempo de la media (tiempo de inicio, o de fin de la media) serán manejados por los subsistemas en milisegundos, por lo que dichas variables se manejan como datos *Float* (reales), que son capaces de soportar números grandes (hasta 500 horas en milisegundos); y los datos que se refieren a un tiempo (hora) determinado son manejados como valores enteros.

- Análisis sobre la Segunda Forma Normal:

Para satisfacer la 2da FN no deben existir dependencias parciales entre los atributos no claves y los claves, en otras palabras: un atributo no primo debe proveer información sobre la clave completa y no solamente sobre una parte de ésta. En la solución propuesta no existen dependencias parciales, ya que todas las claves determinan funcionalmente, de manera total, a todos los atributos no primos de la tabla.

- Análisis sobre la Tercera Forma Normal:

Para satisfacer la 3ra FN, primeramente se tiene que estar en 2da FN, y en adición a esto, no deben existir dependencias transitivas. Esta FN se viola cuando un atributo no clave representa un dato sobre otro atributo no clave. Anteriormente se había visto que la solución propuesta satisfacía la 2da FN, y por otra parte, la solución en propuesta no posee dependencias transitivas ya que estas han sido tratadas según las soluciones que plantea esta forma normal. Cumplidos estos dos parámetros se puede decir entonces que la solución propuesta está en 3ra FN.

- Análisis sobre la Forma Normal de Boyce-Codd

Esta FN es una variante de la 3ra FN un poco más explícita, ya que agrega una restricción (epígrafe 1.2.4). A pesar de que no era una intención explícita llevar a la presente solución hasta esta forma normal, cabe señalar que la misma es violada muy pocas veces (siempre que se está en 3ra FN), y valdría entonces analizar si el diseño realizado se encuentra en dicha FN. Se puede concluir que la presente solución está en la FNBC, ya que para cada dependencia funcional existente X->Y (X determina funcionalmente a Y), se tiene que X es súper-llave.

3.2.3 Análisis de la redundancia de la información

En una base de datos es muy común la aparición de información repetida, en muchos casos, innecesariamente, este tipo de hecho es conocido como “redundancia” de la información. Las bases de datos se pueden optimizar en dos aspectos principales: tamaño, y rapidez. Desafortunadamente es muy difícil optimizar estos dos aspectos a la vez ya que uno excluye al otro, por ejemplo, una forma de mejorar la velocidad en el trabajo sobre una BD es creando cierta redundancia que facilite la obtención de la información que se desea. La presente solución ha sido diseñada de forma tal que la redundancia de la información sea mínima. Algunos casos específicos son analizados en breve.

La entidad “Operacion” es de tipo abstracta, y surge ante la necesidad de eliminar la redundancia que significaría mantener, en cada entidad que se manipula, datos como la hora en que se manipuló, o qué tipo de acción se realizó sobre dicha entidad. Por otra parte, de una media es necesario conocer tres aspectos muy relacionados entre sí: el tiempo de inicio, el tiempo de fin y la duración (tener en cuenta que el tiempo de inicio de una media no tiene por qué ser cero, pues pudiera tratarse de un segmento de otra media más grande), pero con solamente dos de estos atributos se puede llegar al tercero a través de cálculos, lo cual hace que uno de estos tres atributos (en este caso, tiempo de fin) sea calculable (o derivado). De forma general, los atributos derivados son omitidos en la base de datos ya que pueden ser obtenidos

por el sistema sin necesidad de tenerlos almacenados, lo cual elimina redundancia en la base de datos.

La eliminación de redundancias en una base de datos crea complejidades en el diseño, pues muchas veces la solución está en derivar en más de una tabla y establecer relaciones entre éstas, pero por otra parte se optimiza en cuanto a la estabilidad, consistencia y tamaño en disco. De manera general, para la solución propuesta la redundancia ha sido eliminada en una gran medida, pues, al llevarse a la 3ra FN, se elimina este tipo de inconsistencia ya que es uno de los problemas que resuelve la normalización.

3.2.4 Análisis de la seguridad de la base de datos

El ataque a la información puede ser de diferentes tipos: eliminación o modificaciones malintencionadas de la información, descarga o lectura no permitida de datos, interrupción o impedimento de los servicios que necesitan los usuarios. Con la intención de contrarrestar lo antes mencionado es que se realiza la protección de los datos. Con el objetivo de garantizar la seguridad de la base de datos se han tenido algunos aspectos en cuenta.

- Autenticación de usuarios: el acceso a los subsistemas será controlado por la autenticación de los usuarios en el mismo. De esta forma se garantiza que la información sea accedida sólo por el personal pertinente, y que cada usuario que intente entrar al sistema sea realmente quien dice ser.
- Clasificación de usuarios por roles y establecimiento de permisos: los usuarios serán clasificados por roles, y estos a su vez serán previstos de permisos de forma predeterminada. El objetivo de esta medida es mantener el criterio de privilegios mínimos dentro del sistema: cada usuario tendrá sólo los permisos que les corresponde, aquellos sin los cuales no podría realizar sus acciones, nunca tendrá más de los que necesita.
- Rastros en las acciones realizadas: la base de datos está diseñada para que cada usuario deje una marca rastreable en las acciones que realiza en el sistema. Esto limitará la posible decisión de un usuario de realizar acciones malintencionadas en el sistema.
- Servidor de base de datos en lugar seguro: el servidor de base de datos deberá estar en un lugar protegido del clima, acceso de personal no autorizado, y de ataques indirectos como la interrupción malintencionada de servicios (muchas veces es más fácil interrumpir el servicio que atacar directamente al sistema). Este local deberá estar preferiblemente separado del resto de las máquinas donde trabajarán los usuarios.

- Gestión de clientes de PostgreSQL: éste gestiona el acceso de los clientes basados en máquinas (host). Permite determinar a cuáles bases de datos se pueden conectar cuáles usuarios, incluso, desde cuál host.
- Control de acceso de PostgreSQL: éste almacena datos de importancia para el acceso a la base de datos, entre estos están los datos de los usuarios, así como del grupo al que pertenece dentro del catálogo del sistema. Las conexiones al servidor deben ser realizadas por usuarios específicos, tratando siempre de evitar cuentas de administración para ello. Se puede definir qué usuarios pueden realizar cuáles consultas, actualizaciones, o inserciones de datos.
- Copias de seguridad: las copias de seguridad son medidas muy empleadas para la recuperación ante fallos y desastres. En el epígrafe 2.2.2 se describe la concepción de una base de datos principal con réplicas, lo cual representa la principal copia de seguridad de la base de datos.

3.2.5 Trazabilidad de las acciones

El término de trazabilidad (en términos particulares de la informática) se puede describir como la capacidad que poseen los sistemas informáticos de rastrear comportamientos de objetos monitoreados, expresando su resultado en alguna medida o estándar relacionado con referencias específicas, a través de una cadena de comparaciones con incertidumbres predefinidas. Partiendo de esta idea, se han definido algunos aspectos en el diseño de la solución propuesta que contribuirán a la trazabilidad de las acciones en el sistema. Las tablas “Autenticacion” y “Operacion” son los pilares principales de la trazabilidad de las acciones en el sistema, ya que fueron concebidas específicamente con ese objetivo, y son las encargadas de dejar rastros en las demás entidades de forma tal que pueda conocerse la relación entre los elementos almacenados en la BD. A partir de ese aspecto se pueden conocer datos muy específicos, por ejemplo: si se quisiera conocer qué usuario realizó una determinada planificación, se podría hacer un rastreo partiendo de la tabla “Planificacion”, obteniendo de esta el dato “operación_id”, que conduciría una tupla de la tabla “Operacion” que guarda los datos de la operación mediante la cual se hizo la planificación, obteniendo de esta el registro “autenticacion_id” para llegar a la autenticación desde la cual se hizo dicha operación, y de la misma forma llegar finalmente al nombre del usuario que hizo la planificación de la cual se partió inicialmente, y una vez obtenido su nombre se puede buscar cualquier otro dato del mismo. El hecho de que casi todas las tablas estén conectadas, o relacionadas entre sí, ofrece la posibilidad de hacer reportes y seguimientos exhaustivos de los datos almacenados.

3.3 Validación funcional del diseño realizado

El siguiente paso luego de realizar una validación teórica del diseño, es realizar una prueba que valide funcionalmente la solución propuesta. Para llevar a cado este procedimiento se ha configurado un servidor de prueba, utilizando en éste el gestor planteado por la presente solución: el PostgreSQL. Luego en dicho servidor se ha montado la base de datos, y se ha procedido luego a realizar una prueba de estrés, de integridad referencial y de velocidad de respuesta. ¿Cómo se ha llevado a cabo el procedimiento? La prueba ha consistido en generar decenas de miles de datos para cada tabla de la base de datos, para realizar un análisis posterior a los resultados de dicha prueba. Como manualmente demoraría demasiado tiempo generar datos para cada tabla, se ha utilizado el *EMS Data Generator for PostgreSQL*, capaz de generar un gran número de tuplas a una base de datos completa (tantas como el usuario desee), o a un conjunto de tablas específicas de la misma. Este programa, además, es capaz de generar los datos para cada atributo según el tipo de dato definido para el mismo.

La prueba se llevó a cabo por intervalos, generando los datos de forma interrumpida en aras de ampliar las probabilidades de ocurrencia de errores. En la primera iteración se generaron aproximadamente 51 000 tuplas como promedio para cada tabla, un total de 1 224 000 datos en total; luego, en una segunda iteración se generó un promedio de 50 000 tuplas más; y en una tercera iteración se generaron otras 50 000 tuplas, pero solamente a las tablas “Media”, “Autenticacion”, “Operacion” y “Planificacion”, acumulando un total de 2 644 000 datos en total para la base de datos.

Como todo el proceso fue realizado de forma interrumpida, resultaría poco exacto y un tanto especulativo decir el tiempo real de realización de todo el procedimiento, pero sí se puede decir que no excedió las dos horas. Esta prueba permitió probar la integridad referencial de los datos, que fue mantenida en un 100%; además, tras haber generado más de 2.6 millones de datos se a podido demostrar la robustez de la base de datos y el PostgreSQL en sí.

Luego de dicho proceso, fue necesario realizar consultas a la base de datos para comprobar el buen funcionamiento de la misma, así como su tiempo de respuesta. Antes de mostrar algunos ejemplos sería válido aclarar que el servidor sobre el cual se realizaron dichas pruebas, posee características técnicas de hardware (velocidad de procesamiento, capacidad de disco duro y RAM) inferiores a las que se piden en los requisitos no funcionales planteados por parte de los módulos del sistema, lo cual quiere decir que las velocidades de respuesta pueden ser aún mayores para el servidor real a utilizar para el sistema. Para las consultas se utilizó la herramienta

pgAdmin, a continuación se muestran algunos ejemplos de consultas realizadas y sus resultados.

La siguiente consulta devuelve los permisos de un rol determinado:

```
SELECT
  Rolpermiso.permiso_nombre
FROM
  RolPermiso
WHERE
  (Rolpermiso.rolnombre = ‘x’)
```

Esta consulta devolvió tres resultados en un total de 51 ms (milisegundos). Se escogió esta consulta porque es una de las más utilizadas por el sistema para validad a los usuarios y darle sus permisos en la aplicación.

La siguiente consulta devuelve si un usuario determinado (para este ejemplo, el usuario “A”) está habilitado:

```
SELECT
  Usuario.habilitado
FROM  Usuario
WHERE
  (Usuario.usuario = 'A')
```

Esta consulta devolvió el resultado en 16 ms, y al igual que el ejemplo anterior se trata de una consulta simple, pero importante para definir el acceso de un usuario en la aplicación.

La siguiente consulta devuelve la descripción de las operaciones que ha hecho el usuario “x”, además de la máquina y la hora en que realizó cada operación:

```
SELECT
  Operacion.descripcion, autenticacion.pc, operacion.hora
FROM  Operacion, Autenticacion
WHERE
  ((Operacion.autenticacion_id = Autenticacion.id) and Autenticacion.usuario_nombre
= 'x')
```

Esta consulta devolvió un total de 600 filas en 63 ms.

3.4 Resultados obtenidos

Con la obtención de una propuesta de diseño de la base de datos para el Sistema de Captura y Catalogación de Medias (SCCM), y las validaciones pertinentes de dicho diseño, se han obtenido un grupo de resultados:

- La base de datos para el SCCM cumple con el listado de requerimientos funcionales planteados para el mismo de forma total.
- La base de datos se encuentra normalizada hasta la FN de Boyce-Codd, por lo que está fortalecida contra redundancias e inconsistencias.
- Las propiedades a cumplir por la base de datos en cuanto a integridad referencial, integridad semántica, y de dominio, así como reducción de la redundancia y seguridad de los datos, han sido cumplidas correctamente.
- Existe trazabilidad en las acciones.
- La valoración funcional ha demostrado el buen funcionamiento de la base de datos.
- Se ha conseguido diseñar una base de datos que integra todos los módulos del SCCM.

3.5 Conclusiones

Luego de realizar el diseño de la base de datos, se ha procedido a analizar un conjunto de criterios que validarán el diseño realizado. Entre estos criterios de validación se encuentran: la normalización, integridad, trazabilidad de las acciones, redundancia de la información, y seguridad de los datos. El análisis de dichos criterios ha arrojado resultados positivos, lo cual ha validado el diseño realizado, por tanto se puede decir que dicho diseño satisface las necesidades planteadas para la base de datos del sistema. Los requisitos funcionales planteados para la BD son cubiertos en su totalidad. Luego de realizar las pruebas pertinentes se han obtenido tiempos de respuesta satisfactorios, dichas pruebas no han arrojado errores en el diseño, y las restricciones de dicho diseño han permanecido inviolables. Puede concluirse entonces que el diseño de la base de datos para el SCCM cumple con las necesidades de dicho sistema, a la misma vez que satisface las necesidades que debe cumplir como base de datos.

Conclusiones

Para el Sistema de Captura y Catalogación de Medias, que se encuentra actualmente en desarrollo, se ha propuesto el diseño de una base de datos que satisface las necesidades y requisitos planteados previamente para la misma. Para llevar a cabo todo el proceso de desarrollo de dicha propuesta ha sido necesario realizar primeramente un proceso de investigación de carácter científico, que ha sido traducido posteriormente en experiencias adquiridas, y que ha fundamentado la toma de decisiones a lo largo de todo el siclo de desarrollo del sistema.

El objetivo planteado para el presente trabajo ha sido realizar el diseño lógico e implementar el modelo físico de la base de datos para el Sistema de Captura y Catalogación de Medias del Polo de Video y Sonido Digital de la Universidad de las Ciencias Informáticas. Para el cumplimiento de dicho objetivo se seleccionaron las herramientas y tecnologías a utilizar, siendo éstas las que más se ajustaban a las necesidades existentes. Luego de realizar el diseño de la base de datos éste ha sido validado en diferentes aspectos como: integridad de los datos, normalización, redundancia de la información, seguridad, y funcionamiento, obteniendo resultados satisfactorios en cada una de dichas validaciones. Se puede concluir que los objetivos planteados para el presente trabajo han sido cumplidos en su totalidad, y que el SCCM cuenta con una base de datos que integra sus diferentes subsistemas y satisface sus necesidades.

Recomendaciones

Luego de haber cumplido los objetivos planteados, y a partir de los resultados obtenidos, se recomienda:

- Utilizar el diseño de base de datos realizado para el Sistema de Captura y Catalogación de Medias.
- Utilizar el Sistema Gestor de Base de Datos PostgreSQL para gestionar la información en otros sistemas.
- Utilizar las herramientas seleccionadas en el presente trabajo para el manejo de la base de datos realizada.
- Implementar el conjunto de réplicas explicadas en el presente documento.
- Definir los usuarios de autenticación, roles, y permisos en el servidor de PostgreSQL para la conexión de los diferentes módulos a la base de datos.

Bibliografía

Aguilar, Vicente y Suau, Pablo. 2000. MySQL vs. PostgreSQL. [En línea] 18 de agosto de 2000. [Citado el: 5 de diciembre de 2008.] http://www.bisente.com/documentos/mysql-postgres.html.

Berzal, Fernando. 2006. Relaciones entre clases: diagramas de clases UML. *ELVEX.* [En línea] 2006. [Citado el: 15 de diciembre de 2009.] http://elvex.ugr.es/decsai/java/pdf/3C-Relaciones.pdf. 35482.

CamStudio. 2007. CamStudio. [En línea] 2007. [Citado el: 25 de noviembre de 2008.] http://camstudio.es .

Castaño, Adoración de Miguel, Velthuis, Mario Piattini y Martínez, Esperanza Marcos. 2000. *Diseño de Bases de Datos Relacionales.* México : RA-MA, 2000. 84-7897-385-0.

García, Yissell María Castro y Villar, Roberto Llerena. 2008. Propuesta de diseño para la base de datos de Akademos 2.0. 2008.

JACOBSON, Ivar, BOOCH, Grady y RUMBAUGH, James. 2000. El Proceso Unificado de Desarrollo de Software. s.l. : Addison Wesley, 2000.

Leyva, Arnoldo Domínguez. 2008. Modelo Lógico y físico de la base de datos correspondiente a los módulos de investigación criminalística y estadística del proyecto CICPC. 2008.

Martin, J. 1977. *Organización de las bases de datos.* s.l. : Prentice-Hall, 1977. 544 p.

Martínez, Antonio. 2008. Probando PostgreeSQL. [En línea] 2008. [Citado el: 15 de diciembre de 2008.] http://www.antoniomtz.org/?q=postgresql.

PostgreeSQL. 2008. PostgreeSQL. [En línea] 2008. [Citado el: 10 de diciembre de 2008.] http://grupove.org.ve/?q=ventajas.

PRESSMAN, Roger. 2002. Ingeniería del Software. Un enfoque práctico. 2002.

Sánchez Calas, Juan Carlos. 2002. ¿Qué son los Metadatos? *SIEL.* [En línea] 2002. [Citado el: 25 de noviembre de 2008.] http://www.uportal.cl/siel/siel_docs/estandarizacion/metadatos_SIEL.pdf.

SQLite. 2007. *SQLite.* [En línea] 14 de noviembre de 2007. [Citado el: 10 de enero de 2009.] http://www.sqlite.org/copyright.html.

Vilas, Ana Fernández. 2001. Diagramas UML. [En línea] 20 de marzo de 2001. [Citado el: 25 de noviembre de 2008.] http://tvdi.det.uvigo.es/~avilas/UML/node18.html.

VivaLinux. 2009. VivaLinux. [En línea] 18 de enero de 2009. [Citado el: 19 de enero de 2009.] http://www.vivalinux.com.ar/.

Referencias bibliográficas

Apache Derby. 2007. *Apache Derby.* [En línea] 21 de mayo de 2007. [Citado el: 8 de enero de 2009.] http://db.apache.org/derby/.

Castaño, Adoración de Miguel, Velthuis, Mario Piattini y Martínez, Esperanza Marcos. 2000. *Diseño de Bases de Datos Relacionales.* México : RA-MA, 2000. 84-7897-385-0.

EEM SYSTEMS. 2005. PostgreeSQL. *EEM SYSTEMS.* [En línea] 19 de enero de 2005. [Citado el: 18 de enero de 2009.] http://eemsystems.com/web/psql.

Firebird. 2008. *Firebird.* [En línea] 2008. [Citado el: 17 de enero de 2009.] http://www.firebird.com.mx/modules/news/.

García, Yissell María Castro y Villar, Roberto Llerena. 2008. Propuesta de diseño para la base de datos de Akademos 2.0. 2008.

GRADY, BOOCH, IVAR, JACOBSON y JAMES, RUMBAUGH. 2006. *El lenguaje Unificado de Modelado.* s.l. : Pearson Prentice Hall, 2006. 8478290761.

LINFO. 2004. BSD Licence Definition. [En línea] 19 de abril de 2004. [Citado el: 30 de noviembre de 2008.] http://www.linfo.org/bsdlicense.html.

Martin, J. 1977. *Organización de las bases de datos.* s.l. : Prentice-Hall, 1977. 544 p.

MySQL. 2008. About MySQL. *MySQL.* [En línea] 2008. [Citado el: 18 de enero de 2009.] http://www.mysql.com/about/.

pgAdmin. 2008. *pgAdmin.* [En línea] 10 de febrero de 2008. [Citado el: 18 de enero de 2009.] http://www.pgadmin.org/.

PostgreeSQL. 2008. PostgreeSQL. [En línea] 2008. [Citado el: 10 de diciembre de 2008.] http://grupove.org.ve/?q=ventajas.

Sánchez Calas, Juan Carlos. 2002. ¿Qué son los Metadatos? *SIEL.* [En línea] 2002. [Citado el: 25 de noviembre de 2008.] http://www.uportal.cl/siel/siel_docs/estandarizacion/metadatos_SIEL.pdf.

SARENET. 2008. Multidifusión. *SARENET.* [En línea] 2008. [Citado el: 30 de noviembre de 2008.] http://www.sarenet.es/SARENET/Servicios/Multidifusion_(Streaming)/multidifusion.htm.

Universia. 2008. Diccionario tecnológico. [En línea] 2008. [Citado el: 14 de diciembre de 2008.] http://tecnologia.universia.es/diccionario/s.htm.

Vilas, Ana Fernández. 2001. Diagramas UML. [En línea] 20 de marzo de 2001. [Citado el: 25 de noviembre de 2008.] http://tvdi.det.uvigo.es/~avilas/UML/node18.html.

XTREAM. 2008. MediaBox. [En línea] 2008. [Citado el: 15 de diciembre de 2008.] http://www.xtreamsig.com/productosMediaBox.htm.

—. 2008. XTAMP. [En línea] 2008. [Citado el: 15 de diciembre de 2008.] http://www.xtreamsig.com/productosXtamp.htm.

Glosario de términos

ADO (*ActiveX Data Objects*): mecanismo utilizado por sistemas software para comunicarse con bases de datos, para gestionar (insertar, modificar y eliminar) información u obtenerla de las mismas.

Atributos: es la unidad menor de información sobre un objeto almacenado en la BD. Son propiedades que poseen las entidades, por ejemplo: el color, el nombre, identificador, o cualquier otro elemento que describa a la entidad que se necesite conocer. Se representan a través de elipses, o pequeños círculos, conectadas a la entidad mediante una línea recta.

Base de Datos (BD): Conjunto de datos con carácter persistente almacenados para su posterior uso en un sistema informático. Estos datos pueden estar localizados tanto en la misma máquina del sistema que los necesita (máquina cliente) como en una máquina remota (servidor de base de datos) conectados mediante una red local.

Base de Datos Relacional: base de datos cuya estructura está constituida por tablas bidimensionales estructuradas en registros (líneas) y campos (columnas), que se vinculan entre sí por un campo en común. Los campos de las tablas tienen las mismas características: nombre, tipo de dato, longitud. En cada tabla existe un campo (o conjunto de éstos) que identifica a cada registro como único, este campo es llamado identificador o llave.

Conectores: Uno de los elementos que componen al Modelo de Despliegue. Expresa el tipo de conector o protocolo utilizado entre el resto de los elementos del modelo.

Claves foráneas (*foreign key* en inglés): Utilizadas para relacionar tablas.

Disparadores: (también conocidos como Desencadenadores) son procedimientos especiales almacenados en la base de datos luego de ejecutarse una función INSERT, DELETE, o UPDATE. Se ejecutan automáticamente con el fin de realizar procedimientos o acciones predefinidas.

Dispositivos: Uno de los elementos que componen al Modelo de Despliegue. Representa a nodos estereotipados sin capacidad de procesamiento en el nivel de abstracción que se modela.

Dominio: conjunto de valores posibles que puede tomar un atributo determinado.

Entidad: es el objeto básico representado en el modelo E-R. Representa elementos u objetos de la vida real con existencia necesaria en la base de datos. Una entidad puede ser un objeto con existencia física como: una persona, una máquina, una pieza; o bien puede ser un objeto con existencia conceptual: un curso académico, un departamento, un viaje, o cualquier otro elemento del cual se necesite mantener información almacenada. Éstas deben utilizar nombres en singular preferentemente, y que sean descriptores de la entidad. Se representan por un rectángulo con el nombre de la entidad dentro.

Escalabilidad: es la capacidad que posee el sistema informático para adaptarse a nuevos cambios. Es su habilidad de adaptar su tamaño o configuración para ajustarse a las circunstancias cambiantes.

Framework: es una estructura de soporte que incluye bibliotecas de clases, un lenguaje interpretado, entre otros componentes encargados de unir diferentes partes de un proyecto. Es un software que permite que otro software pueda ser organizado y desarrollado.

Interrelación: relación, vínculo o correspondencia entre entidades.

Inyección SQL: es una vulnerabilidad en las validaciones que presentan los programas informáticos que utilizan bases de datos. Se trata de filtrar código SQL a través de los campos de entrada de datos para lograr que la consulta a la base de datos se realice de forma maliciosa. También puede hacerse “inyectando” o filtrando código SQL dentro de otro código SQL.

Licencia BSD: Pertenece al grupo de licencias de software Libre. Utilizada por primera vez en 1980 por *Berkeley Source Distribution* (BSD). Esta licencia permite el uso de código BSD sobre software no libre aunque éste luego pase a ser liberado. (LINFO, 2004).

Llave: atributo (o conjunto de atributos) que identifica a una entidad como un objeto único e irrepetible dentro de la base de datos. Ejemplo: la llave de la entidad PERSONA puede ser el NoCI (número de carnet de identidad); en la entidad PIEZA la llave puede ser el número de serie.

Metadatos: Información que describe datos que incluyen el contenido, la forma y las características técnicas y editoriales de la información electrónica.

Metodología robusta: Se basan en muchos años de experiencia en el uso de la tecnología orientada a objetos. Son muy útiles para desarrollar soluciones empresariales muy complejas. Guía a los equipos de proyecto en cómo administrar el desarrollo iterativo de un modo controlado mientras se balancean los requerimientos del negocio, y los riesgos del proyecto.

Modelo de Despliegue: Se utiliza para capturar los elementos de configuración del procesamiento y las conexiones entre estos elementos. También se utiliza para visualizar la distribución de los componentes de software en los nodos físicos.

Modelo-Entidad-Relación (MER): también conocido como modelo de datos, es un tipo de diagrama para modelado de bases de datos. Su objetivo es representar relaciones que existen en la vida real entendiendo su semántica.

Nodos: Uno de los elementos que componen al Modelo de Despliegue. Representa a elementos de procesamiento con al menos un procesador y memoria.

Normalización: La normalización es un proceso que permite obtener relaciones de la forma plana bidimensional, logrando que las tablas de datos tengan la forma adecuada, para evitar tanto la ocurrencia de anomalías de actualización como la falta de consistencia de los datos.

OO: Orientado a objetos (siglas OO). Es un paradigma de programación para la creación de aplicaciones y programas informáticos que emplea objetos (como estructura básica) y sus relaciones. Está basado técnicas como: herencia, modularidad, polimorfismo y encapsulamiento.

RAM: Memoria de Acceso Aleatorio (siglas *Random Access Memory*). Es un componente del ordenador que se encarga de almacenar las instrucciones que recibe el procesador, así como los resultados que arroja el mismo.

Requisito: son funcionalidades o condiciones que un sistema informático debe alcanzar o poseer para cumplir con un contrato o documento formal.

Requisitos funcionales (del sistema): capacidades o condiciones que el producto debe cumplir.

Requisitos no funcionales (del sistema): propiedades o cualidades que el producto debe tener.

SCCM: Solución de Captura y Catalogación de Medias

SGBD: Sistema Gestor de Base de Datos

Streaming: Sistema de transmisión de datos que permite, mediante un programa que maneje estos datos, ir recibiéndolos y a la vez procesándolos sin necesidad de esperar a que se descarguen del todo. Muy práctico para escuchar/ver programas de radio o televisión. (Universia, 2008).

Trigguers: ver Disparadores.

UCI: Universidad de las Ciencias Informáticas

XML: metalenguaje extensible de etiquetas llamado así por sus siglas en inglés de *Extensible Markup Language*. Desarrollado por el *World Wide Web Consortium*. Permite definir la gramática de lenguajes específicos y se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier tipo de tecnología de sistema informático.

Anexo 1: Figuras

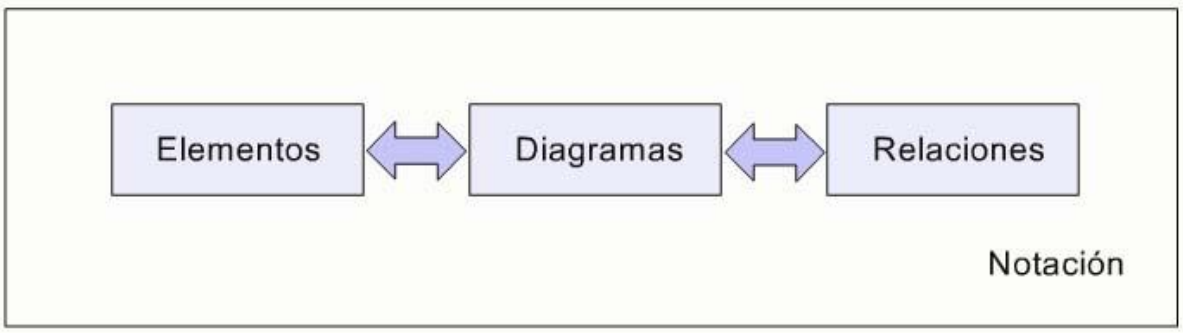


Figura 1: Composición de UML.

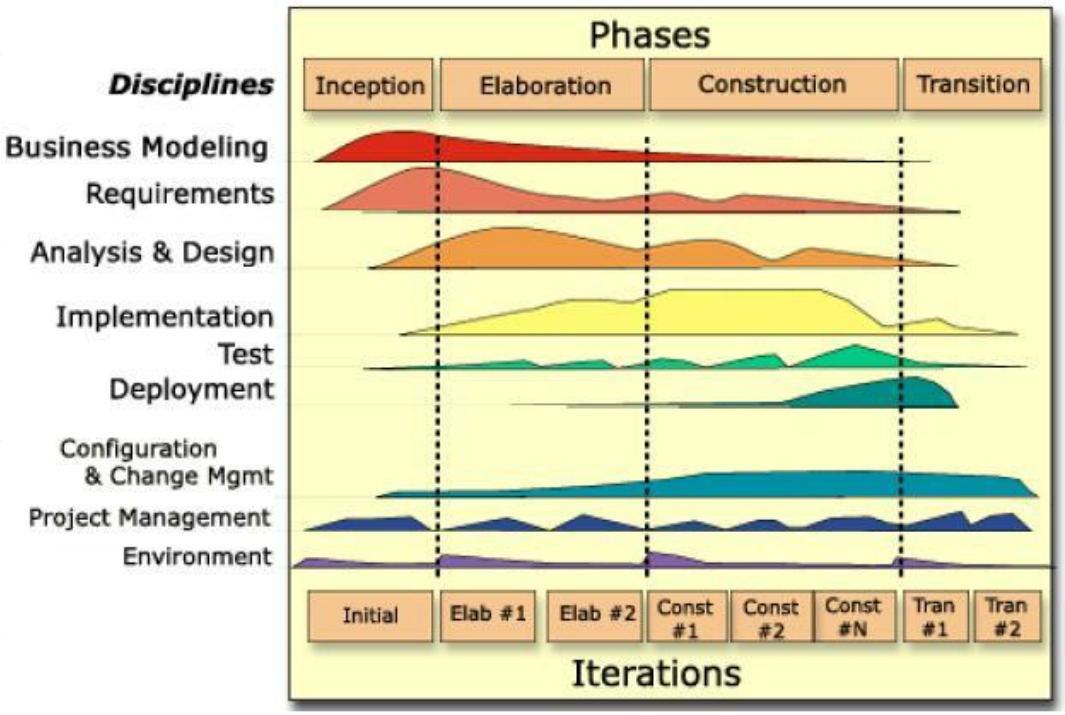


Figura 2: El ciclo de vida de RUP.

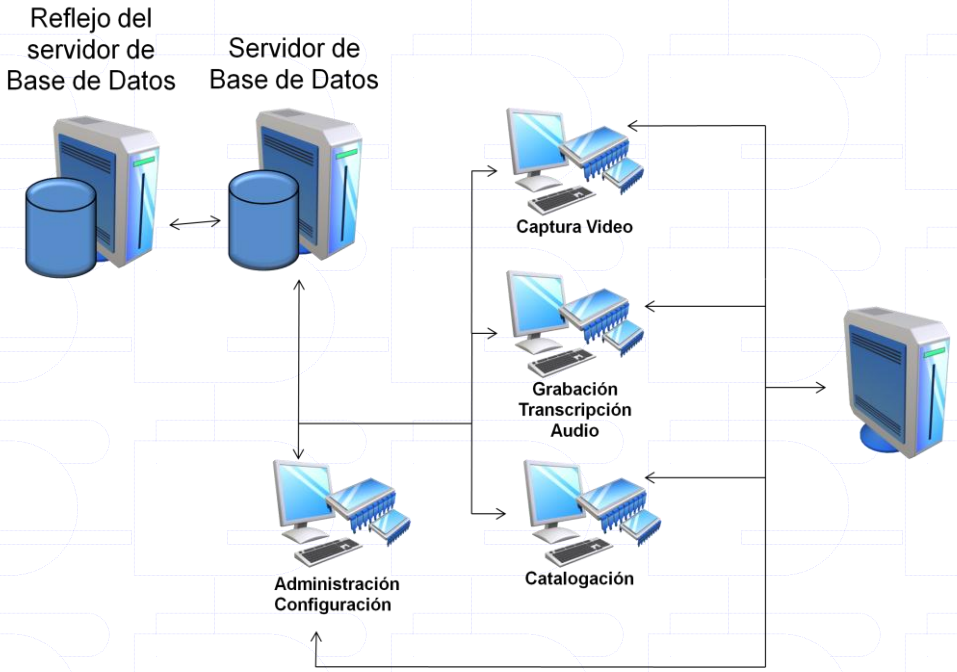


Figura 3: Reflejo de la Base de Datos.

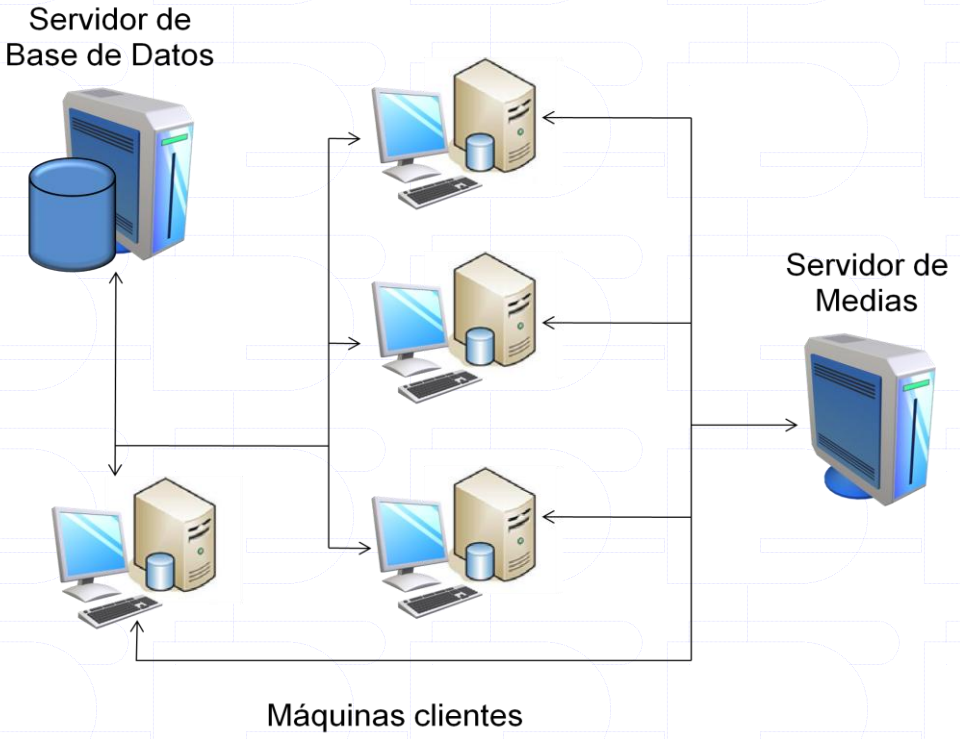


Figura 4: Réplicas locales de la base de datos ubicadas en las máquinas clientes.

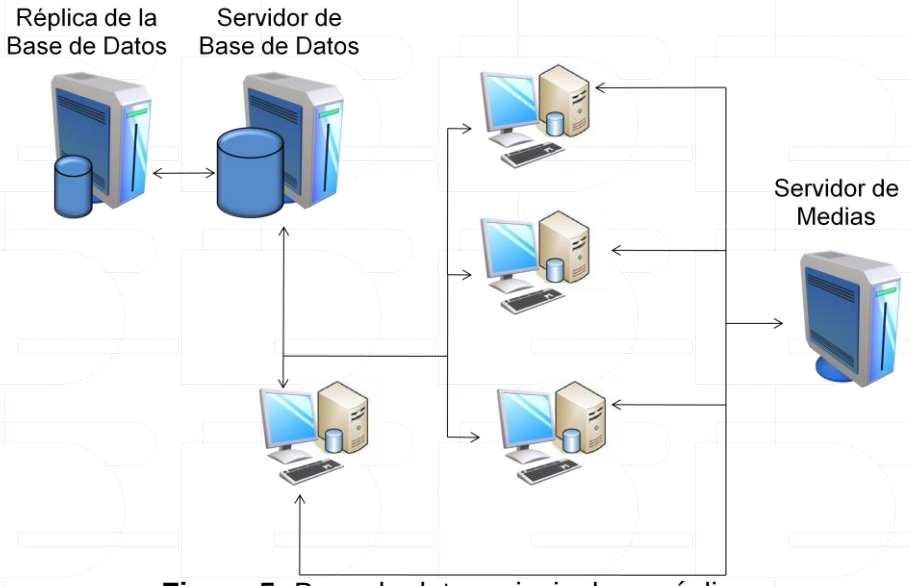


Figura 5: Base de datos principal con réplica.

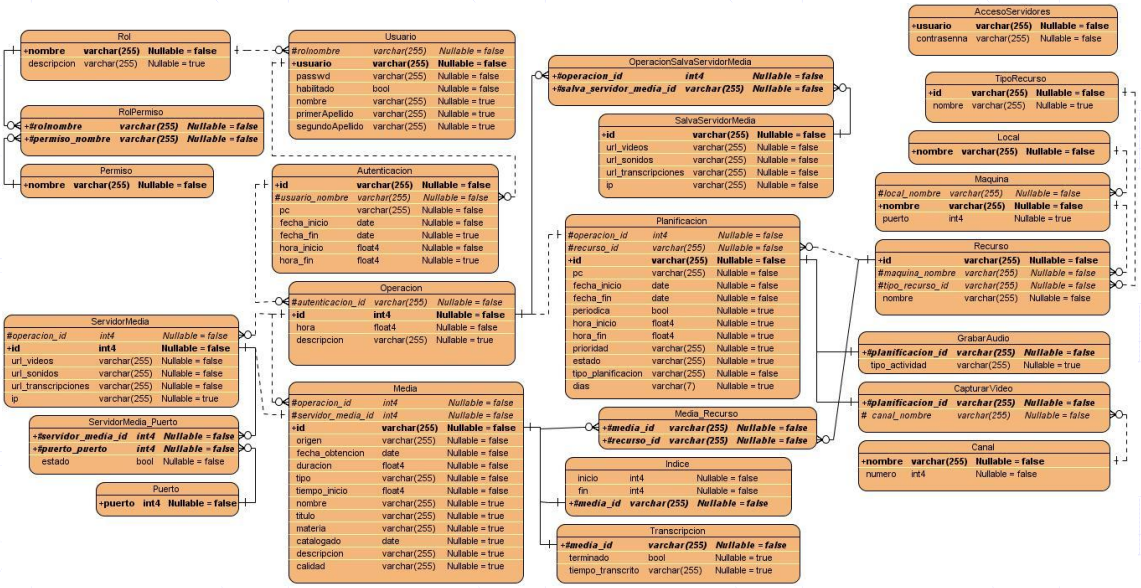


Figura 6: Diagrama de Entidad-Relación extendido.