Universidad de las Ciencias Informáticas

Facultad 9

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

Título: *Diseño de una base de datos para el control de los RRHH en los polos productivos de la facultad 9.*

Autor
Lidier González Ballester

Tutor
Yoandrys González González

Co-tutor
Arnaldo Gandol Álvarez

Ciudad de La Habana, Mayo 2009.

"Año 50 de la Revolución"

"Nuestra recompensa se encuentra en el esfuerzo y no en el resultado. Un esfuerzo total es una victoria completa."

Mahatma Gandhi

A mis padres, ustedes son mi razón de ser, son mi vida, gracias por su amor, su sinceridad, sus consejos, gracias por su crianza.

A mis tías Adelba y Margarita por ser mis otras madres, gracias por sus atenciones.

A mis primos Arián, Orestes, Marielkis, Ángeles y Felipín gracias por ser mis hermanos.

A mis tíos Moisés, Pipe, Gerardo y Orestes, gracias por sus enseñanzas.

A mis abuelitas Ignacia y Ramona, a mis abuelitos Felipe y Santiago gracias por brindarme su apoyo y su cariño.

A Isabelita y Rosa, gracias por todo, siempre van a estar presentes.

A Arletis, gracias por soportarme todo este tiempo, la verdad aunque no se vea, te debo mucho, gracias.

A mi tutor Yoandrys, gracias por tu apoyo, por enseñarme, gracias por todo.

A mis amigos gracias por su apoyo, si por algo no quisiera irme de la universidad es por ustedes, gracias por su amistad.

A Toledo, Yusdenys y a Kenny gracias por toda su ayuda en el desarrollo de la tesis.

A mis padres que son mi vida, a mi familia que siempre me dio su apoyo, a todos los que me ayudaron a llegar a este momento.

Lidier

Síntesis del tutor Ing. Yoandrys González González

Sexo: Masculino

Institución: Universidad de las Ciencias Informáticas

Dirección de la institución: Carretera a San Antonio de los Baños, Km. 2 ½, Reparto: Torrens, Municipio: Boyeros, Provincia: Ciudad de La Habana.

Correo electrónico: yggonzalez@uci.cu

Teléfono del trabajo: 835-8907

Categoría docente: Adiestrado

Cargo del trabajador: Profesor

Título de la especialidad de graduado: Ingeniero en Ciencias Informáticas

Año de graduación: 2007

Institución donde se graduó: Universidad de las Ciencias Informáticas

Resumen

La Universidad de las Ciencias Informáticas tiene como organización base para el desarrollo de software los Polos Productivos, los cuáles se organizan y planifican en pos de lograr un correcto proceso de desarrollo de software. La gestión de los RRHH en toda organización es un tema fundamental debido a que es el activo más importante con que cuentan.

En la actualidad el país está inmerso en un proceso de informatización de la sociedad en la que los Polos Productivos son fiel exponente evidenciado por la preparación que tiene el personal involucrado y la disposición de tecnologías para su desarrollo, no obstante el proceso del control y gestión de los RRHH se lleva a cabo de forma artesanal en el sentido de que tiene como base para procesar y persistir la información las hojas de cálculo, con sus respectivas desventajas: información redundante, inconsistente y frágil, poco útil para brindar reportes de calidad; por lo que la presente investigación tiene como objetivo principal desarrollar una base de datos que almacene y controle toda esta información para dar soporte a una aplicación web de gestión de los RRHH de los Polos Productivos.

Para cumplimentar el objetivo general propuesto se realizó un estudio del arte acerca de las bases de datos de forma general, enfocados fundamentalmente en el modelo relacional para su diseño, así como, una descripción de las herramientas utilizadas para el desarrollo de la base de datos.

En la presente investigación se describen los modelos lógico y físico de la base de datos, desarrollada con el objetivo de darle solución a la situación problemática planteada, encargada de almacenar los datos relacionados con los RRHH de los Polos Productivos de la facultad 9, además de validar teórica y funcional la base de datos.

Palabras Claves: gestión de RRHH, base de datos, modelo relacional

INDICE TABLAS.

INDICE FIGURAS.

ÍNDICE

Introducción**:**

Las personas no son recursos que la organización consume y utiliza, los que producen costos. Al contrario los RRHH constituyen un poderoso activo que impulsa la creatividad organizacional, de la misma manera que lo hacen el mercado o la tecnología .

La administración y gestión de los recursos humanos surge como una necesidad mediadora entre las entidades y las personas. En sus inicios se veía al ser humano como un simple elemento dentro de la organización, como lo podía ser la materia prima o procesos de trabajo, sin tomar en cuenta sus necesidades como individuo. En sus inicios la administración de los RRHH se encargaba solamente de vigilar que los intereses de las personas no se alejaran de los intereses de la organización.

La idea de Planificar los Recursos Humanos no es nueva; se viene aplicando, de forma más intuitiva que lógica, desde que las personas comenzaron a colaborar en grupos como forma básica de actuación. Sin embargo, es de obligado cumplimiento el aceptar que las empresas se han preocupado más por la planificación de los recursos financieros y productivos, dotándolos de herramientas informáticas adecuadas como los ERP, que de los Recursos Humanos, a pesar de que su coste, en función de la actividad de las empresas, representa entre un 60 y un 90% de los costes totales. En contraposición, todos parecen estar de acuerdo en el hecho de que los Recursos Humanos representan el principal generador de ventajas competitivas sostenibles dentro de las empresas y además que, por ahora, son absolutamente indispensables.**(1)**

En cualquier entidad en conjunto con las personas y las tareas que se desarrollan, es necesario tener en cuenta dos características más: las tecnologías necesarias para obtener sus metas (métodos y herramientas para realizar las tareas) y la estructura organizacional, que puede definirse como: el conjunto de todas las formas en que se divide el trabajo en tareas distintas, consiguiendo la coordinación de las mismas, con el fin de alcanzar los objetivos.

El éxito de toda organización está dado por una serie de factores en su mayoría, referidos a la actividad de los recursos humanos, por lo que se hace necesaria una correcta gestión de este vital recurso; de ahí la necesidad por parte de las organizaciones de poseer sistemas que automaticen esta engorroso proceso.

En el mundo existen software que son enfocados a esta área, sólo que para acceder a estos es necesario pagarles un considerable monto a sus propietarios. En nuestro país muchas organizaciones poseen sistemas automatizados con estos fines los que en algunos casos automatizan parte del proceso y en los menos todo lo vinculado con la gestión y administración de recursos humanos. Se encuentran principalmente en empresas capaces de costearse las inversiones que provocan el desarrollo de estos sistemas automatizados, la mayoría utilizando software propietario para la construcción de los mismos.

La Universidad de las Ciencias Informáticas surgida al calor de la Batalla de Ideas se crea como una universidad de excelencia en la que se reúnen a estudiantes de todo el país. Los cuales reciben un cúmulo de estudios intensivos relacionados con las Ciencias Informáticas y en cuyo principal propósito se tiene la investigación, desarrollo y creación de software. Para lo cual la universidad se encuentra dividida en 10 facultades y cada facultad presenta un número de Polos Productivos los cuales tienen perfiles asociados para a la asignación de proyectos de software relacionados con los mismos.

La gestión de los RRHH en el caso de la Universidad de las Ciencias Informáticas es atípico debido a que sí se controlan los mismos, pero las características de estos distan mucho de lo que una empresa normal, debido a que el grueso del capital humano lo componen los estudiantes.

En el caso de la facultad 9 presenta los polos Productivos de: Video y Sonido Digital, Simulación de Procesos Químicos, Petrosoft, PICG y Calidad de Software. Y con ellos una serie de proyectos asociados en desarrollo. El almacenamiento y gestión de la información referente a los recursos humanos de los polos productivos en la universidad de forma general y en esta facultad específicamente se hacen manual y fundamentalmente por una sola persona, lo cual está dado por la inexistencia de un sistema que automatice estas tareas. La forma de lograr la persistencia de dichos datos y su manipulación se hace mediante hojas de cálculo lo que puede provocar: eliminación del fichero con todos sus datos, modificación de la información, sustracción de información confidencial.

De ahí que para los administrativos de cada polo se le hace muy difícil el trabajo con tantos datos referentes a los recursos humanos por lo que provoca:

- Búsqueda ineficiente de la información.
- Incorrecto flujo de información.
- Lenta toma de decisiones.
- Mala comunicación entre los integrantes del polo (administrativos e integrantes).
- Fragilidad de la información.
- Subutilización de los RRHH.
- Mala Evaluación de los RRHH.

De aquí surge el **Problema a resolver:**

Inexistencia de una base de datos para la gestión de los RRHH en los polos productivos de la facultad 9.

El **objeto de estudio:**

El proceso de diseño de bases de datos relacionales.

Teniendo como **campo de acción:**

Diseño lógico y físico de la base de datos para la gestión de RRHH.

Por lo que el **objetivo general** consiste en**:**

Diseñar una base de datos que permita almacenar y controlar la información de los RRHH de los polos productivos de la facultad 9.

La **idea a defender** de la presente investigación es:

Si se logra diseñar la base de datos podrá dar servicio a la aplicación web de gestión de los RRHH de los Polos Productivos en la Facultad 9.

Para darle cumplimiento al objetivo general de la investigación las principales **tareas a desarrollar** son las siguientes:

1. Elaboración del diseño teórico y la fundamentación teórica de la investigación.
2. Desarrollo del modelo lógico de datos.
    - Identificar entidades persistentes.

- Identificar los atributos de las entidades persistentes.
- Identificar relaciones entre tablas.
- Normalizar el modelo de la base de datos.

3. Desarrollo del modelo físico de datos.
    - Definir dominios de las tablas.
    - Definir reglas de aplicación de integridad referencial y de datos.
    - Optimizar el acceso a los datos.
    - Definir la seguridad de la base de datos.
4. Validación del diseño realizado.

Con el desarrollo de la presente investigación se espera dotar ha los Polos Productivos de un componente (base de datos) para gestionar información de los recursos humanos de cada polo con lo cual se logre una mejor comunicación entre sus integrantes (administrativos y trabajadores) y una mejor planificación y control de las tareas asignadas a cada trabajador.

Los **Métodos Científicos** propuestos son:

Métodos Teóricos

- **Analítico-Sintético**:

Debido a que permite hacer un análisis del estado del arte del tema que se investiga, permite ver hasta donde se ha avanzado en ese tema, y cuáles son sus perspectivas, por lo que se analizaron las herramientas para el modelado de las bases de datos así como los SGBD para utilizarlos en el diseño de la base de datos.

- **Análisis Histórico Lógico:**

Permitirá realizar un estudio de tendencias de la trayectoria de las herramientas en el mundo para el modelado de base de datos, así como, los SGBD utilizados. Además se analizaron diferentes versiones de las herramientas para destacar las principales ventajas y así lograr un aprovechamiento óptimo de las mismas.

- **La Modelación:**

La modelación es el método científico mediante el cual se crean abstracciones con el objetivo de expresar la realidad. La condición principal de la modelación es la relación entre el modelo y el objeto que se modela. **(2)**

Se propone la modelación para abstraerse del mundo real y lograr la modelación del mismo primero mediante el modelo de datos, después mediante el diseño lógico de la base de datos, y por último llevarlo al modelo físico de la base de datos.

Métodos Empíricos.

- **La observación**:

Se realizará la observación para familiarizarse con los procesos de negocios existentes en el campo de acción para lograr obtener todas las entidades necesarias que pudieran modelar la situación real presente. Es el instrumento universal del científico, se realiza de forma consciente y orientada a un objetivo determinado.

Los **posibles resultados esperados** son los siguientes:

Diseño del modelo lógico de la base de datos de los RRHH de los Polos Productivos de la Facultad 9.

Diseño de la base de datos física de los RRHH de los Polos Productivos de la Facultad 9.

Capítulo1
Introducción:

En el presente capítulo se realiza un análisis histórico de las bases de datos y sus antecedentes además de una descripción de la metodología por la que se guía el proceso de desarrollo del presente componente, el sistema gestor de base de datos para implementar las BD y la herramienta de modelado de la BD a utilizar.

1.1 Importancia de las bases de datos en una organización.

La informatización de una base de datos tiene importantes ventajas para la gestión de los RRHH en una organización. Una base de datos es mucho más rápida, que los métodos manuales para archivar, procesar y conseguir un volumen importante de datos; una base de datos es más compacta que los archivos manuales convencionales reduciendo considerablemente la redundancia de los datos, pudiendo almacenar miles de datos en poco espacio de un dispositivo de almacenamiento; una base de datos en el ordenador es más flexible, al permitir examinar la misma información bajo distintas condiciones. Finalmente, una base de datos, con los debidos controles, es más segura frente a posibles imprevistos como el robo, los accesos no autorizados o el simple paso del tiempo. **(3)**

La parte administrativa dentro de la organización necesita saber en cada momento un número importante de datos y reportes acerca de la disponibilidad de los RRHH mediante una serie de reportes imprescindibles para un correcto funcionamiento de la organización, debido a que gracias a ellos es perfectamente posibles mediante una base de datos las decisiones por parte de los dirigentes, las que pueden ser mucho más rápidas y basadas por datos perfectamente reales y consistentes.

1.2 Surgimiento y desarrollo de las bases de datos.

Los sistemas de ficheros, precursores de las bases de datos, *surgieron al tratar de informatizar el manejo de los archivadores manuales con objeto de proporcionar un acceso más eficiente a los datos. Un sistema de ficheros es un conjunto de programas que prestan servicio a los usuarios finales. Cada programa define y maneja sus propios datos.* **(4)**

Los sistemas de ficheros presentan una serie de inconvenientes que a continuación se mencionan:

- *Separación y aislamiento de los datos:* Cuando los datos se separan en distintos ficheros, es más complicado acceder a ellos, ya que el programador de aplicaciones debe sincronizar el procesamiento de los distintos ficheros implicados para asegurar que se extraen los datos correctos.

- *Duplicación de datos:* La redundancia de datos existente en los sistemas de ficheros hace que se desperdicie espacio de almacenamiento y lo que es más importante: puede llevar a que se pierda la consistencia de los datos. Se produce una inconsistencia cuando copias de los mismos datos no coinciden.

- *Dependencia de datos:* Ya que la estructura física de los datos (la definición de los ficheros y de los registros) se encuentra codificada en los programas de aplicación, cualquier cambio en dicha estructura es difícil de realizar. El programador debe identificar todos los programas afectados por este cambio, modificarlos y volverlos a probar, lo que cuesta mucho tiempo y está sujeto a que se produzcan errores. A este problema, tan característico de los sistemas de ficheros, se le denomina también *falta de independencia de datos lógica-física*.

- *Formatos de ficheros incompatibles:* Ya que la estructura de los ficheros se define en los programas de aplicación, es completamente dependiente del lenguaje de programación. La incompatibilidad entre ficheros generados por distintos lenguajes hace que los ficheros sean difíciles de procesar de modo conjunto.

A partir de los inconvenientes de los sistemas de ficheros, los cuales se pueden atribuir a dos factores:

- La definición de los datos se encuentra codificada dentro de los programas de aplicación, en lugar de estar almacenada aparte y de forma independiente.
- No hay control sobre el acceso y la manipulación de los datos más allá de lo impuesto por los programas de aplicación.

La gestión de archivos físicos, por lo general, es insuficiente para gestionar la información de una entidad: ocupan demasiado espacio físico, su búsqueda es lenta, su actualización y optimización es complicada (a veces se deben modificar varios archivos únicamente para actualizar uno posterior), los archivos pueden estar dispersos por los distintos departamentos (esto puede ocasionar repeticiones o carencias de archivos), los métodos para controlar la seguridad son muy complejos y

con resultados que pueden ser precarios. Las bases de datos informáticas pretenden solucionar todos estos problemas.

El modelo seguido con los sistemas de bases de datos, en donde se separa la definición de los datos de los programas de aplicación, donde se da una definición interna de un objeto y una definición externa separada. Los usuarios del objeto sólo ven la definición externa y no se deben preocupar de cómo se define internamente el objeto y cómo funciona. Una ventaja de este modelo, conocido como abstracción de datos, es que se puede cambiar la definición interna de un objeto sin afectar a sus usuarios ya que la definición externa no se ve alterada. Del mismo modo, los sistemas de bases de datos separan la definición de la estructura de los datos, de los programas de aplicación y almacenan esta definición en la base de datos. Si se añaden nuevas estructuras de datos o se modifican las ya existentes, los programas de aplicación no se ven afectados ya que no dependen directamente de aquello que se ha modificado.

1.3 Conceptos y características de Bases de Datos:

El término base de datos es ampliamente utilizado por los desarrolladores de software y especialmente por los diseñadores de bases de datos, por lo que surge la necesidad de conceptualizar el mismo por varios autores.

Se entiende por base de datos:
Conjunto de datos interrelacionados entre sí, almacenados con carácter más o menos permanente en la computadora. O sea, una BD puede considerarse una **colección de datos variables en el tiempo**. **(5)**

Una *base de datos* es un conjunto de datos almacenados entre los que existen relaciones lógicas y ha sido diseñada para satisfacer los requerimientos de información de una empresa u organización. En una base de datos, además de los datos, también se almacena su descripción. **(6)**

Según Date un sistema de base de datos es:
Básicamente un sistema computarizado para llevar registros. Se considera a la propia base de datos como una especie de armario electrónico para archivar, es decir, es un depósito o contenedor de una colección de archivos de datos computarizados.

Los usuarios del sistema pueden realizar una variedad de operaciones sobre dichos archivos:

- Agregar nuevos archivos vacíos a la base de datos.
- Insertar datos dentro de los archivos existentes
- Recuperar datos de los archivos existentes.
- Modificar datos en archivos existentes.
- Eliminar datos de archivos existentes.
- Eliminar archivos existentes de la base de datos.

Características de las bases de datos:
Las características que ha de presentar una base de datos de forma general son las siguientes:

- Control centralizado de los datos.
- Integridad de los datos.
- Minimización de las repeticiones.
- Independencia de los datos y las aplicaciones.
- Acceso concurrente a los datos.
- Coste mínimo de almacenamiento y mantenimiento.
- Versatilidad para la representación de relaciones.
- Establecimiento de medidas de seguridad.
- Facilidad para el cambio, actualización y optimización.

1.4 Tipos de Bases de Datos.

Las bases de datos pueden clasificarse de varias maneras, de acuerdo al criterio elegido para su clasificación.

1.4.1 Según la variabilidad de los datos almacenados.

Bases de datos estáticas

Estas son bases de datos de solo lectura, utilizadas primordialmente para almacenar datos históricos que posteriormente se pueden utilizar para estudiar el comportamiento de un conjunto de datos a través del tiempo, realizar proyecciones y tomar decisiones. Son aquellas bases de datos en las que no se puede modificar la información que está guardada, es decir, solo se puede consultar.

Bases de datos dinámicas

Estas son bases de datos donde la información almacenada se modifica con el tiempo**,** permitiendo operaciones como actualización y adición de datos, además de las operaciones fundamentales de consulta. Son aquellas bases de datos en las que la información almacenada está en constante cambio.

Teniendo en cuenta las posibilidades de actualización de la información que ofrecen las bases de datos dinámicas estas se muestran como la mejor opción para la solución de la presente base de datos a desarrollar, además de la necesidad del de interactuar de manera dinámica con la información que gestiona.

1.4.2 Según el contenido.

Bases de datos bibliográficas

Solo contienen un representante de la fuente primaria, que permite localizarla. Un registro típico de una base de datos bibliográfica contiene información sobre el autor, fecha de publicación, editorial, título, edición, de una determinada publicación. Puede contener un resumen o extracto de la publicación original, pero nunca el texto completo, porque si no estaríamos en presencia de una base de datos a texto completo (o de fuentes primarias). Como su nombre lo indica, el contenido son cifras o números. Por ejemplo, una colección de resultados de análisis de laboratorio, entre otras.

Bases de datos de texto completo

Almacenan las fuentes primarias, como por ejemplo, todo el contenido de todas las ediciones de una colección de revistas científicas.

Directorios

Un ejemplo son las guías telefónicas en formato electrónico.

Bases de datos o "bibliotecas" de información Biológica

Son bases de datos que almacenan diferentes tipos de información proveniente de las ciencias de la vida o médicas. Se pueden considerar en varios subtipos:

- Aquellas que almacenan secuencias de nucleótidos o proteínas.
- Las bases de datos de rutas metabólicas.
- Bases de datos de estructura, comprende los registros de datos experimentales sobre estructuras 3D de biomoléculas.
- Bases de datos clínicas.

- Bases de datos bibliográficas (biológicas).

1.5 Arquitectura de las bases de datos

La arquitectura que a continuación se presenta pretende explicar la estructura de los sistemas de base datos, pero no se afirma que todo sistema pueda coincidir completamente con esta arquitectura, pero si en su generalidad, la cual fue propuesta en 1975 por el Grupo de Estudio en Sistemas de Administración de Bases de Datos de ANSI-SPARC (American National Standard Institute - Standards Planning and Requirements Committee), el cual propuso una arquitectura de tres niveles para los SGBD cuyo objetivo principal era el de separar los programas de aplicación de la BD física.

Una BD se define en tres niveles de abstracción distintos:**(7)** **Nivel interno o físico**: el más cercano al almacenamiento físico, es decir, tal y como están almacenados en el ordenador. Describe la estructura física de la BD mediante un esquema interno. Este esquema se especifica con un modelo físico y describe los detalles de cómo se almacenan físicamente los datos: los archivos que contienen la información, su organización, los métodos de acceso a los registros, los tipos de registros, la longitud, los campos que los componen, etcétera. **Nivel externo o de visión**: es el más cercano a los usuarios, es decir, es donde se describen varios esquemas externos o vistas de usuarios. Cada esquema describe la parte de la BD que interesa a un grupo de usuarios en este nivel se representa la visión individual de un usuario o de un grupo de usuarios. **Nivel conceptual:** describe la estructura de toda la BD para un grupo de usuarios mediante un esquema conceptual. Este esquema describe las entidades, atributos, relaciones, operaciones de los usuarios y restricciones, ocultando los detalles de las estructuras físicas de almacenamiento. Representa la información contenida en la BD.

1.6 Modelos para el diseño de bases de datos.

Además de la clasificación por la función de las bases de datos, éstas también se clasifican de acuerdo a su modelo de administración de datos. Un modelo de datos es básicamente una "descripción" de algo conocido como *contenedor de datos* (algo en donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores. Los modelos de datos no son cosas físicas: son abstracciones que permiten la implementación de un sistema eficiente de *base de datos.*

1.6.1 Bases de datos Jerárquicas.

Son bases de datos que, como su nombre indica, almacenan su información en una estructura jerárquica. 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.

El modelo jerárquico fue desarrollado para permitir la representación de aquellas situaciones de la vida real en las que predominan las relaciones de tipo 1: N. Es un modelo muy rígido en el que las diferentes entidades de las que está compuesta una determinada situación, se organizan en niveles múltiples de acuerdo a una estricta relación PADRE/HIJO, de manera que un padre puede tener más de un hijo, todos ellos localizados en el mismo nivel, y un hijo únicamente puede tener un padre situado en el nivel inmediatamente superior al suyo. **(8)**

1.6.2 Bases de datos de red.

El modelo de datos en red general representa las entidades en forma de nodos de un grafo, y las interrelaciones entre estas mediante arcos que unen dichos nodos. En principio esta representación no impone restricción alguna acerca del tipo y el número de arcos que puede haber, con lo que se pueden modelar estructuras de datos tan complejas como sea necesario. **(9)**

1.6.3 Bases de datos orientadas a objetos.

Este modelo, bastante reciente, y propio de los modelos informáticos orientados a objetos, trata de almacenar en la base de datos los objetos completos (estado y comportamiento). Una base de datos orientada a objetos es una base de datos que incorpora todos los conceptos importantes del paradigma de objetos: Encapsulación, propiedad que permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos. Herencia, propiedad a través de la cual los objetos heredan comportamiento dentro de una jerarquía de clases. Polimorfismo, propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos. En bases de datos orientadas a objetos, los usuarios pueden definir operaciones sobre los datos como parte de la definición de la base de datos. La implementación de la operación se especifica separadamente y puede modificarse sin afectar la interfaz. Los programas de aplicación de los usuarios pueden operar sobre los datos invocando a dichas operaciones a través de sus nombres y argumentos, sea cual sea la forma en

la que se han implementado. Esto podría denominarse independencia entre programas y operaciones.

1.6.4 Base de datos relacional (Modelo relacional).

Éste es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos.

Su idea fundamental es el uso de "relaciones". Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados tuplas. Se piensa en cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla). En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos.

Cada relación tiene un o un conjunto de atributos, conocidos como clave primaria que identifican unívocamente a la fila. Las referencias de interrelaciones son manipuladas a través de llaves foráneas, que son punteros lógicos de una fila de una relación a una fila de otra relación. Para estar seguro de la validación de los datos en una base de datos relacional, Codd formuló las reglas de integridad de la entidad y las reglas de integridad referencial. Estas reglas establecen que el valor de una clave primaria no puede ser nulo, y que el valor de una clave externa debe corresponder con el valor actual de la clave primaria de la otra relación a la que se hace referencia. **(15)**

La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información. El lenguaje más habitual para construir las consultas a bases de datos relacionales es SQL, Structured Query Language o Lenguaje Estructurado de Consultas, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales. Durante su diseño, una base de datos relacional pasa por un proceso al que se le conoce como normalización de una base de datos.

Los sistemas relacionales son importantes porque ofrecen muchos tipos de procesos de datos, como: simplicidad y generalidad, facilidad de uso para el usuario final,

períodos cortos de aprendizaje y consultas de información especificadas de una forma muy sencilla.

Ventajas del modelo relacional

- Compatibilidad y estandarización.
- Fiabilidad.
- Garantía de independencia de los datos.
- Existencia de numerosos sistemas comerciales entre los que escoger y consiguiente apoyo técnico.
- Conectividad garantizada con los lenguajes de programación estándar.

1.6.4.1 Características principales de las bases de datos basadas en el modelo relacional.

A continuación se hace énfasis en características que revisten importancia para las bases de datos y nunca deben faltar:

La exigencia de integridad de los datos garantiza la calidad de los datos de la base de datos. Los aspectos que se incluye tienen que ver con la exactitud, consistencia y confiabilidad de la información. Las bases de datos tienen dentro de sus características elementos que pueden ser utilizados para garantizar la calidad de la información almacenada y procesada.

1.6.4.2 Integridad de los datos en las bases de datos.

La integridad de los datos se enmarca en lograr precisión y consistencia en los valores de los datos en las bases de datos los cuáles se logra estableciendo medidas de seguridad para el acceso a la base de datos como pueden ser el método de autenticación para manipular los datos, establecimiento de roles a los usuarios, según los cuáles van a tener acceso a diferentes datos en la base de datos así como establecer diferentes permisos, cómo pueden ser sólo lectura, de modificación de eliminación o todos, además de establecer el acceso a los datos mediante tablas virtuales que no son más que vistas de la base de datos.

Una restricción es una regla que limita los valores que pueden estar presentes en los datos de las bases de datos. El modelo relacional de Codd incluye varias restricciones que se usan para verificar la validación de los datos, *manifestando la existencia de las restricciones de entidad y las de integridad referencial*. **(16)**

Se consideran las siguientes restricciones:

- **Integridad de entidad**

  La integridad de entidad establece que una tabla va a ser única determinado por la llave primaria de la misma la cual la hace diferente a las demás. Es decir no pueden existir 2 tuplas con la misma llave primaria.

- **Integridad referencial (17)**

  La regla de integridad referencial se aplica a las claves ajenas: *si en una relación hay alguna clave ajena, sus valores deben coincidir con valores de la clave primaria a la que hace referencia, o bien, deben ser completamente nulos*.

  La regla de integridad referencial se enmarca en términos de estados de la base de datos: indica lo que es un estado ilegal, pero no dice cómo puede evitarse. La cuestión es ¿qué hacer si estando en un estado legal, llega una petición para realizar una operación que conduce a un estado ilegal? Existen dos opciones: *rechazar* la operación, o bien *aceptar* la operación y realizar operaciones adicionales compensatorias que conduzcan a un estado legal.

  Por lo tanto, para cada clave ajena de la base de datos habrá que contestar a tres preguntas:

- *Regla de los nulos:* ¿Tiene sentido que la clave ajena acepte nulos?
- *Regla de borrado:* ¿Qué ocurre si se intenta borrar la tupla referenciada por la clave ajena?
  - *Restringir:* no se permite borrar la tupla referenciada.
  - *Propagar:* se borra la tupla referenciada y se propaga el borrado a las tuplas que la referencian mediante la clave ajena.
  - *Anular:* se borra la tupla referenciada y las tuplas que la referenciaban ponen a nulo la clave ajena (sólo si acepta nulos).
- *Regla de modificación:* ¿Qué ocurre si se intenta modificar el valor de la clave primaria de la tupla referenciada por la clave ajena?
  - *Restringir:* no se permite modificar el valor de la clave primaria de la tupla referenciada.

- *Propagar:* se modifica el valor de la clave primaria de la tupla referenciada y se propaga la modificación a las tuplas que la referencian mediante la clave ajena.
- *Anular:* se modifica la tupla referenciada y las tuplas que la referenciaban ponen a nulo la clave ajena.

La integridad referencial se activa en cuanto se crea una clave foránea y a partir de ese momento se comprueba cada vez que se modifiquen datos que puedan alterarla.

- **Integridad de dominio**

  La integridad de dominio viene dada por la validez de las entradas para un atributo determinado de una tupla. Puede exigir la integridad de dominio para restringir el tipo de datos entrado, el cual puede ser determinado por el sistema, restringido por el usuario en un rango de ese mismo tipo de dato, o el usuario puede crear un nuevo tipo de dato. Además deben existir mecanismos de seguridad y restauración soportados por los sistemas gestores de bases de datos (SGBD) que deben proteger la información ante cualquier fallo del sistema.

- **Integridad definida por el usuario**

  La integridad definida por el usuario permite definir reglas de empresa específicas que no pertenecen a ninguna otra categoría de integridad. Todas las categorías de integridad admiten la integridad definida por el usuario. Se puede lograr mediante el establecimiento de procedimientos almacenados y desencadenadores. **(18)**

1.6.4.3 Redundancia en las bases de datos relacionales:

Redundancia es un estado en el cual existen datos repetidos innecesariamente. **(19)**

Los sistemas de ficheros almacenan varias copias de los mismos datos en ficheros distintos. Esto hace que se desperdicie espacio de almacenamiento, además de provocar la falta de consistencia de datos.

En los sistemas de bases de datos todos estos ficheros están integrados, por lo que no se almacenan varias copias de los mismos datos. Sin embargo, en una base de datos no se puede eliminar la redundancia completamente, ya que en ocasiones es necesaria para modelar las relaciones entre los datos.

1.6.4.4 Consistencia de los datos en las bases de datos:

Eliminando o controlando las redundancias de datos se reduce en gran medida el riesgo de que haya inconsistencias. Si un dato está almacenado una sola vez, cualquier actualización se debe realizar sólo una vez, y está disponible para todos los usuarios inmediatamente. Si un dato está duplicado y el sistema conoce esta redundancia, el propio sistema puede encargarse de garantizar que todas las copias se mantienen consistentes.

Inconsistencia es un estado en el que dos o más datos repetidos en una BD contienen diferentes valores. Generalmente es consecuencia de la actualización despareja o desordenada de datos redundantes. *Por tanto no puede haber inconsistencia si no existe primero redundancia de los datos*. **(20)**

1.6.4.5 Transacciones en bases de datos:

Una transacción no es necesariamente una sola operación de la base de datos, sino que en general es una secuencia de varias de estas operaciones que transforman un estado consistente de la base de datos en otro estado consistente sin que sea necesario conservar la consistencia en todos los puntos intermedios.

Un sistema que soporta la administración de transacciones garantiza que si la transacción ejecuta algunas transacciones y luego, por cualquier razón, ocurre una falla antes de que la transacción alcance su terminación planeada, entonces esas actualizaciones serán deshechas.

Por lo que la transacción o se ejecuta totalmente o se cancelan totalmente es decir como si esa transacción no hubiera sido ejecutada. De esta forma, es posible hacer que una secuencia de operaciones que básicamente no son atómicas parezca que lo fuera, desde el punto de vista externo.

Propiedades ACID (21)

Las transacciones cumplen con 4 propiedades importantes: *atomicidad, consistencia, aislamiento y durabilidad.*

Atomicidad: Las transacciones son atómicas, (todo o nada).

Consistencia: Las transacciones conservan la consistencia de la base de datos. Una transacción transforma el estado consistente de la base de datos en otro igual, sin necesidad de conservar la consistencia en todos los puntos intermedios.

Aislamiento: Las transacciones están aisladas entre sí. Aunque estén ejecutándose en forma concurrente, las actualizaciones de una transacción dada están ocultas ante las demás, hasta que esa transacción sea confirmada. Por ejemplo, para 2 transacciones distintas T1 y T2; T1 podrá ver las actualizaciones de T2 después de que esta haya sido confirmada o viceversa, pero no podrán suceder ambas cosas.

Durabilidad: Una vez que una transacción es confirmada, sus actualizaciones sobreviven en la base de datos aún cuando haya una caída posterior del sistema, es decir se hacen persistentes.

1.6.4.6 Concurrencia en las bases datos:

El control de concurrencia trata con los problemas de aislamiento y consistencia del procesamiento de transacciones. El control de concurrencia en un SGBD asegura que la consistencia de la base de datos se mantiene en un ambiente multiusuario. La manera más simple de lograr este objetivo es ejecutar cada transacción sola, una después de otra. En un SGBD se necesita algún tipo de **mecanismo de control de concurrencia** para asegurar que las transacciones concurrentes no interfieran entre sí.

Problemas de Concurrencia:

Existen 3 formas en que la ejecución concurrente de transacciones pueden afectarse unas a las otras, y producir resultados incorrectos y dejar la base de datos en un estado inconsistente.

La transacción que interfiere puede ser correcta por sí misma, lo que produce el resultado incorrecto general es el ***intercalado*** sin control entre las operaciones de las dos transacciones correctas.

Los problemas son:(22)

El problema de la actualización perdida.

Dada 2 transacciones A y B las cuales acceden concurrentemente a los mismos datos.

La transacción A recupera alguna tupla t en el tiempo *t1*; la transacción B recupera la misma tupla t en el tiempo *t2*; la transacción A actualiza la tupla en cuestión en el tiempo t3(con base en los valores visto en el tiempo *t1*), y la transacción B actualiza la misma tupla en el tiempo *t4*(con base en los valores visto en el tiempo *t2*), por lo que se sobrescribe y se pierde la actualización realizada por la transacción A.

El problema de la dependencia no confirmada.

El problema de la dependencia no confirmada se presenta al permitir que una transacción recupere, o lo que es peor, actualice una tupla que ha sido actualizada por otra transacción pero que esta transacción aún no ha sido confirmada. Puesto que no ha sido confirmada, sigue existiendo la posibilidad de que nunca lo sea y de que en su lugar se deshaga mediante un ROLLBACK; en cuyo caso, la primera transacción habrá visto datos que nunca existieron porque esa transacción nunca se confirmó (CONMIT).

El problema del análisis inconsistente.

El problema del análisis inconsistente ocurre cuando una transacción ha visto un estado inconsistente de la base de datos y por lo tanto, ha realizado un análisis inconsistente.

1.6.4.7 Técnicas de control de la concurrencia (23).

Existen varias técnicas para controlar la concurrencia. *Los bloqueos* son los más conocidos, aunque también se utiliza el *control multi-versión* y otras técnicas como las marcas de tiempo.

Los bloqueos como solución al problema de la concurrencia:

Una forma de controlar la concurrencia es hacer que cada transacción deba adquirir un derecho de acceso exclusivo a cada fragmento de datos que necesite modificar. A estos “derechos” se les denomina bloqueos.

Bloqueos binarios

La forma más simple de bloquear es utilizar bloqueos binarios. En un bloqueo binario, cada transacción debe solicitar el bloqueo de cada fragmento de datos A que vaya a utilizar antes de acceder a él (sea para leerlo o escribirlo), mediante una operación

bloquear(A). Deberá liberar todos los bloqueos, mediante una operación desbloquear(A) de modo que otras tareas puedan tomarlos.

Este sistema de bloqueos tiene una implementación muy simple, ya que solo requiere mantener una tabla que indica qué partes de los datos está bloqueada y por qué transacción.

El sistema de bloqueos binarios es simple pero demasiado restrictivo, ya que no permite que dos transacciones que van a leer el mismo fragmento de datos A lo hagan simultáneamente, cuando en realidad, no puede haber problemas en varios lectores simultáneos. Los bloqueos de lectura/escritura hacen más débil la restricción permitiendo la siguiente compatibilidad de bloqueos.

En este caso, las operaciones que las transacciones deben realizar son tres: desbloquear(A) y bloquear_para_lectura(A) o bloquear_para_escritura(A).

Serialización de los bloqueos de lectura/escritura

La serialización de las operaciones de lectura y escritura consiste en ordenar esas operaciones para un conjunto de transacciones concurrentes de modo que los resultados de las operaciones sean correctos.

El bloqueo en dos fases permite la serialización

El protocolo de bloqueo en dos fases fuerza a las transacciones cuando todas las operaciones de adquisición de bloqueos (bloquear_lectura, bloquear_escritura) preceden a la primera operación de desbloqueo (desbloquear). Dicho de otro modo, primero hay que adquirir todos los bloqueos, y después se pueden liberar.

Inconvenientes de los bloqueos y la serialización

Un problema del protocolo de bloqueo en dos fases es que puede llevar a situaciones de interbloqueo. La siguiente es una secuencia de operaciones que lleva al interbloqueo pero cumple perfectamente el protocolo de bloqueo en dos fases.

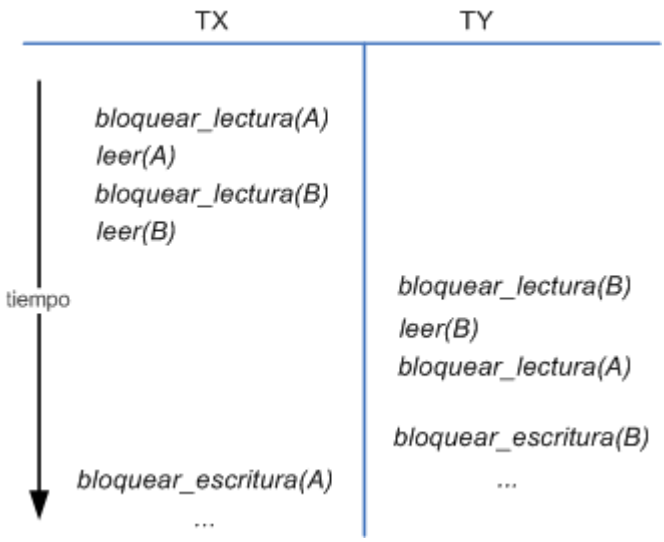

Figura 1 Situación de Interbloqueo.

El inconveniente está en que puede que en la fase de adquisición de bloqueos (fase de expansión), más de una transacción esté interesada en los mismos dos fragmentos de datos, y la mala suerte nos lleve a una situación en la que ambos quedan suspendidos, sin posibilidad de avanzar.

Guardando "instantáneas" de los datos: el Control Multi-versión:

El protocolo de bloqueo en dos fases limita considerablemente las posibilidades de concurrencia. Si se observa los problemas que causan los bloqueos no serializados, se ve que muchos de los problemas están en que una transacción lee un cierto dato y antes de escribir el resultado, otra transacción lee el dato "antiguo". En ese momento, cada transacción trabaja con un estado de información inconsistente.

Para paliar esos problemas, pero permitir la mayor concurrencia posible, se han diseñado los protocolos de control multi-versión. La idea básica es que cuando una transacción modifica un dato, se crea una nueva versión del mismo, pero se guarda la anterior. De este modo, al acabar la ejecución de las transacciones, se puede utilizar para cada una de ellas la versión de los datos "que hace la ejecución correcta", es decir, que hace la ejecución serializable.

Lógicamente, estas técnicas requieren más espacio de almacenamiento para guardar las diferentes versiones.

1.7 Bases de datos, sus principales componentes.

Las BD son de vital importancia en muchos aspectos, contienen la información específica para poder llevar a cabo la estimación más exacta de situaciones por lo que ayuda a la toma de decisiones en empresas e instituciones.

Las BD comprenden cuatro componentes: **datos, hardware, software y usuarios**.

Datos: El objetivo principal de los sistemas de BD es permitir que cada usuario se comporte como si el sistema estuviera trabajando con un solo usuario conectado. Para esto los datos en la BD serán tanto integrados como compartidos.

Datos Integrados: Unificación de varios archivos (datos) que de otro modo serían distintos, con una redundancia entre ellos mínima pero necesaria.

Datos Compartidos: Los datos individuales en la BD pueden ser compartidos entre diferentes usuarios y que cada uno de ellos puede tener acceso a la misma pieza de datos para diferentes fines.

En conclusión que la base de datos sea integrada y compartida significa que la misma sea percibida de muchas formas diferentes por distintos usuarios.

Hardware: Se refiere a los dispositivos de almacenamiento donde reside la BD, así como a los dispositivos periféricos (unidad de control, canales de comunicación) necesarios para su uso.

Software: Entre la base de datos física(es decir, los datos como están almacenados físicamente) y los usuarios del sistema, hay una capa de software conocida como Sistema de Administración de Bases de Datos (DBMS). Todas las solicitudes de acceso a la base de datos generadas por los usuarios son manejadas por el DBMS. Las cuáles son las siguientes: **(10)**

- Agregar nuevos archivos vacíos a la base de datos.
- Insertar datos dentro de los archivos existentes.
- Recuperar datos de los archivos existentes.

- Modificar datos en archivos existentes.
- Eliminar datos de archivos existentes.
- Eliminar archivos existentes de la base de datos.

Por lo tanto, una función general que ofrece el DBMS consiste en **ocultar a los usuarios de la base de datos los detalles a nivel de hardware.**

Usuarios: Se consideran tres grandes clases de usuarios.

- El **programador de aplicaciones**, quien crea programas de aplicación que utiliza la BD, en algún lenguaje de programación como puede ser COBOL, C++, Java o algún programa de alto nivel.

- Los **usuarios finales**, quien accede a la BD por medio de un lenguaje de consulta o de programas de aplicación, desde estaciones de trabajo o terminales en línea.

- El **administrador de la BD** (DBA Data Base Administrator), quien se encarga del control general del Sistema de BD.

1.8 Sistemas Gestores de Bases de Datos.

Los Sistemas Gestores de Bases de Datos (SGBD) son un tipo de software muy específico, dedicado a servir de interfaz entre las bases de datos y las aplicaciones que la utilizan.

Sus funciones son:

1.8.1 Funciones de los sistemas de gestión de bases de datos.

Codd, el creador del modelo relacional, ha establecido una lista con los ocho servicios que debe ofrecer todo SGBD.

1. Un SGBD debe proporcionar a los usuarios la capacidad de almacenar datos en la base de datos, acceder a ellos y actualizarlos. Esta es la función fundamental de un SGBD y por supuesto, el SGBD debe ocultar al usuario la estructura física interna (la organización de los ficheros y las estructuras de almacenamiento).
2. Un SGBD debe proporcionar un catálogo en el que se almacenen las descripciones de los datos y que sea accesible por los usuarios. Este catálogo

es lo que se denomina diccionario de datos y contiene información que describe los datos de la base de datos (metadatos). Normalmente, un diccionario de datos almacena:

- Nombre, tipo y tamaño de los datos.
- Nombre de las relaciones entre los datos.
- Restricciones de integridad sobre los datos.
- Nombre de los usuarios autorizados a acceder a la base de datos.
- Esquemas externos, conceptuales e internos, y correspondencia entre los esquemas.
- Estadísticas de utilización, tales como la frecuencia de las transacciones y el número de accesos realizados a los objetos de la base de datos.

Algunos de los beneficios que reporta el diccionario de datos son los siguientes:

- La información sobre los datos se puede almacenar de un modo centralizado. Esto ayuda a mantener el control sobre los datos, como un recurso que son.
- El significado de los datos se puede definir, lo que ayudará a los usuarios a entender el propósito de los mismos.
- La comunicación se simplifica ya que se almacena el significado exacto. El diccionario de datos también puede identificar al usuario o usuarios que poseen los datos o que los acceden.
- Las redundancias y las inconsistencias se pueden identificar más fácilmente ya que los datos están centralizados.
- Se puede tener un historial de los cambios realizados sobre la base de datos.
- El impacto que puede producir un cambio se puede determinar antes de que sea implementado, ya que el diccionario de datos mantiene información sobre cada tipo de dato, todas sus relaciones y todos sus usuarios.
- Se puede hacer respetar la seguridad.
- Se puede garantizar la integridad.
- Se puede proporcionar información para auditorías.

3. Un SGBD debe proporcionar un mecanismo que garantice que todas las actualizaciones correspondientes a una determinada transacción se realicen, o que no se realice ninguna. Una *transacción* es un conjunto de acciones que cambian el contenido de la base de datos. Una transacción en el sistema informático de una empresa inmobiliaria sería dar de alta a un empleado o

eliminar un inmueble. Una transacción un poco más complicada sería eliminar un empleado y reasignar sus inmuebles a otro empleado. En este caso hay que realizar varios cambios sobre la base de datos. Si la transacción falla durante su realización, por ejemplo porque falla el hardware, la base de datos quedará en un estado inconsistente. Algunos de los cambios se habrán hecho y otros no, por lo tanto, los cambios realizados deberán ser deshechos para devolver la base de datos a un estado consistente.

4. Un SGBD debe proporcionar un mecanismo que asegure que la base de datos se actualice correctamente cuando varios usuarios la están actualizando concurrentemente. Uno de los principales objetivos de los SGBD es el permitir que varios usuarios tengan acceso concurrente a los datos que comparten. El acceso concurrente es relativamente fácil de gestionar si todos los usuarios se dedican a leer datos, ya que no pueden interferir unos con otros. Sin embargo, cuando varios usuarios están accediendo a la base de datos y al menos uno de ellos está actualizando datos, pueden interferir de modo que se produzcan inconsistencias en la base de datos. El SGBD se debe encargar de que estas interferencias no se produzcan en el acceso simultáneo.
5. Un SGBD debe proporcionar un mecanismo capaz de recuperar la base de datos en caso de que ocurra algún suceso que la dañe. Como se ha comentado antes, cuando el sistema falla en medio de una transacción, la base de datos se debe devolver a un estado consistente. Este fallo puede ser a causa de un fallo en algún dispositivo hardware o un error del software, que hagan que el SGBD aborte, o puede ser a causa de que el usuario detecte un error durante la transacción y la aborte antes de que finalice. En todos estos casos, el SGBD debe proporcionar un mecanismo capaz de recuperar la base de datos llevándola a un estado consistente.
6. Un SGBD debe proporcionar un mecanismo que garantice que sólo los usuarios autorizados pueden acceder a la base de datos. La protección debe ser contra accesos no autorizados, tanto intencionados como accidentales.
7. Un SGBD debe ser capaz de integrarse con algún software de comunicación. Muchos usuarios acceden a la base de datos desde terminales. En ocasiones estos terminales se encuentran conectados directamente a la máquina sobre la que funciona el SGBD. En otras ocasiones los terminales están en lugares remotos, por lo que la comunicación con la máquina que alberga al SGBD se debe hacer a través de una red. En cualquiera de los dos casos, el SGBD recibe peticiones en forma de mensajes y responde de modo similar. Todas

estas transmisiones de mensajes las maneja el gestor de comunicaciones de datos. Aunque este gestor no forma parte del SGBD, es necesario que el SGBD se pueda integrar con él para que el sistema sea comercialmente viable.

8. Un SGBD debe proporcionar los medios necesarios para garantizar que tanto los datos de la base de datos, como los cambios que se realizan sobre estos datos, sigan ciertas reglas. La integridad de la base de datos requiere la validez y consistencia de los datos almacenados. Se puede considerar como otro modo de proteger la base de datos, pero además de tener que ver con la seguridad, tiene otras implicaciones. La integridad se ocupa de la calidad de los datos. Normalmente se expresa mediante restricciones, que son una serie de reglas que la base de datos no puede violar. Por ejemplo, se puede establecer la restricción de que cada empleado no puede tener asignados más de diez inmuebles. En este caso sería deseable que el SGBD controlara que no se sobrepase este límite cada vez que se asigne un inmueble a un empleado.

Además, de estos ocho servicios, es razonable esperar que los SGBD proporcionen un par de servicios más:

1. Un SGBD debe permitir que se mantenga la independencia entre los programas y la estructura de la base de datos. La independencia de datos se alcanza mediante las vistas o subesquemas. La independencia de datos física es más fácil de alcanzar, de hecho hay varios tipos de cambios que se pueden realizar sobre la estructura física de la base de datos sin afectar a las vistas. Sin embargo, lograr una completa independencia de datos lógica es más difícil. Añadir una nueva entidad, un atributo o una relación puede ser sencillo, pero no es tan sencillo eliminarlos.
2. Un SGBD debe proporcionar una serie de herramientas que permitan administrar la base de datos de modo efectivo. Algunas herramientas trabajan a nivel externo, por lo que habrán sido producidas por el administrador de la base de datos. Las herramientas que trabajan a nivel interno deben ser proporcionadas por el distribuidor del SGBD. Algunas de ellas son:
    - Herramientas para importar y exportar datos.
    - Herramientas para monitorizar el uso y el funcionamiento de la base de datos.
    - Programas de análisis estadístico para examinar las prestaciones o las estadísticas de utilización.
    - Herramientas para reorganización de índices.

- Herramientas para aprovechar el espacio dejado en el almacenamiento físico por los registros borrados y que consoliden el espacio liberado para reutilizarlo cuando sea necesario.

1.8.2 Objetivos de un SGBD.

Existe un conjunto de objetivos generales que deben cumplir los SGBD, de modo que faciliten el diseño de aplicaciones y los tratamientos de las mismas sean más eficientes y rápidos: **(11)**

- **Independencia de los datos y los programas de aplicación**

Con archivos de aplicación la lógica de la aplicación contempla la organización de los archivos y el método de acceso. Ejemplo, si por razones de eficiencia se utiliza un archivo secuencial indexado, el programa de aplicación debe considerar la existencia de los índices y la secuencia de archivos. Entonces es imposible modificar la estructura de almacenamiento la estrategia de acceso sin afectar el programa de aplicación (se afectan en el programa las partes del mismo que tratan los archivos).

Razones por lo que resultaría indeseable la existencia de aplicaciones y datos dependientes entre sí, por dos razones fundamentales:

1. Diferentes aplicaciones necesitarían diferentes aspectos de los mismos datos(por ejemplo puede requerirse la representación decimal o binaria).
2. Se debe poder modificar la estructura de almacenamiento o el método de acceso según los cambios según los cambios en el fenómeno o proceso de la realidad sin necesidad de modificarlos programas de aplicación.

La independencia de los datos se define como la inmunidad de las aplicaciones a los cambios en la estructura de almacenamiento y en la estrategia de acceso, y constituye el objetivo fundamental de un SGBD.

- **Minimización de la redundancia**

Los SGBD tienen como objetivo minimizar la redundancia de los datos, no la eliminarla, pues, aunque se definen las BD como no redundantes, existe una redundancia en un grado no significativo necesaria para disminuir el tiempo de acceso a los datos o para simplificar el método de direccionamiento. Lo que se trata es de eliminar la redundancia superflua.

- **Integración y sincronización de las bases de datos**

La **integración** consiste en garantizar una respuesta a los requerimientos de diferentes aspectos de los mismos datos por diferentes usuarios, de forma que, aunque el sistema almacene la información con cierta estructura y cierto tipo de representación, debe garantizar entregar al programa de aplicación los datos que solicita y en la forma en que lo solicita. Está vinculada a la sincronización, que consiste en la necesidad de garantizar el acceso múltiple y simultáneo a la BD, de modo que los datos puedan ser compartidos por diferentes usuarios a la vez. Están relacionadas, ya que lo usual es que diferentes usuarios trabajen con diferentes enfoques y requieran los mismos datos, pero desde diferentes puntos de vista.

- **Integridad de los datos.**

Consiste en garantizar la no contradicción entre los datos almacenados de modo que, en cualquier momento del tiempo, los datos almacenados sean correctos, es decir, que no se detecte inconsistencia entre los datos. Está relacionada con la minimización de la redundancia, ya que es más fácil garantizar la integridad si se elimina la redundancia.

- **Seguridad y recuperación.**

Seguridad (también llamada protección): garantizar el acceso autorizado a los datos, de forma de interrumpir cualquier intento de acceso no autorizado, ya sea por error del usuario o por mala intención.

Recuperación: que el SGBD disponga de métodos que garanticen la restauración de las bases de datos al producirse alguna falla técnica, interrupción de la energía eléctrica.

- **Facilidad de manipulación de la información**

Los usuarios de una BD pueden acceder a la misma con solicitudes para resolver muchos problemas diferentes. El SGBD debe contar con la capacidad de una búsqueda rápida por diferentes criterios, permitir que los usuarios planteen sus demandas de una forma simple, aislándolo de las complejidades del tratamiento de los archivos y del direccionamiento de los datos. Los SGBD actuales brindan lenguajes de alto nivel con diferentes grados de facilidad para el usuario no programador que garantizan este objetivo, los llamados *sublenguajes de datos*.

- **Control centralizado**

Uno de los objetivos más importantes de los SGBD es garantizar el control centralizado de la información. Permite controlar de manera sistemática y única los

datos que se almacenan en la BD, así como el acceso a ella. Lo anterior implica que debe existir una persona o conjunto de personas que tenga la responsabilidad de los datos operacionales: el administrador de la BD, que puede considerarse parte integrante del SGBD.

1.9 Características del SGBD PostgreSQL:

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional (ORDBMS) basado en el proyecto POSTGRES, de la universidad de Berkeley. El director de este proyecto es el profesor Michael Stonebraker. PostgreSQL es una derivación libre (OpenSource) de este proyecto. Fue el pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, incluido, más tarde en otros sistemas de gestión comerciales. PostgreSQL presenta características la herencia, diferentes tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. A pesar de esto.

A continuación se enumeran las principales características de este gestor de bases de datos:

1. Implementación del estándar SQL92/SQL99.
2. Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo: fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP), cadenas de bits, etc. También permite la creación de tipos propios.
3. Incorpora una estructura de datos array.
4. Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, etc.
5. Permite la declaración de funciones propias, así como la definición de disparadores.
6. Soporta el uso de índices, reglas y vistas.
7. Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
8. Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.

Presenta avanzadas funcionalidades como son consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transacciones, optimización de consultas, herencia, y arrays.

Presenta otras características denominadas ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad): **(12)**

● **Atomicidad** (Indivisible) es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.
● **Consistencia** es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper la reglas y directrices de integridad de la base de datos.
● **Aislamiento** es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que dos transacciones sobre la misma información nunca generarán ningún tipo de error.
● **Durabilidad** es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.

Además **PostgreSQL** presenta:

Altamente Extensible

PostgreSQL soporta operadores, funciones métodos de acceso y tipos de datos definidos por el usuario.

Soporte SQL Comprensivo

PostgreSQL soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92.

Integridad Referencial

PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.

API Flexible

La flexibilidad del API de PostgreSQL ha permitido a los vendedores proporcionar soporte al desarrollo fácilmente para el RDBMS PostgreSQL. Estas interfaces incluyen Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++, .NET y Pike.

Lenguajes Procedurales

PostgreSQL tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL. Otra ventaja de PostgreSQL es su habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido.

MVCC

MVCC, o Control de Concurrencia Multi-Versión (Multi-Version Concurrency Control), es la tecnología que PostgreSQL usa para evitar bloqueos innecesarios. Si alguna vez ha usado algún DBMS con capacidades SQL, tal como MySQL o Access, probablemente habrá notado que hay ocasiones en las una lectura tiene que esperar para acceder a información de la base de datos. La espera está provocada por usuarios que están escribiendo en la base de datos. Resumiendo, el lector está bloqueado por los escritores que están actualizando registros.

Mediante el uso de MVCC, PostgreSQL evita este problema por completo. MVCC está considerado mejor que el bloqueo a nivel de fila porque un lector nunca es bloqueado por un escritor. En su lugar, PostgreSQL mantiene una ruta a todas las transacciones realizadas por los usuarios de la base de datos. PostgreSQL es capaz entonces de manejar los registros sin necesidad de que los usuarios tengan que esperar a que los registros estén disponibles.

Write Ahead Logging (WAL)

La característica de PostgreSQL conocida como Write Ahead Logging incrementa la dependencia de la base de datos al registro de cambios antes de que estos sean escritos en la base de datos. Esto garantiza que en el hipotético caso de que la base de datos se caiga, existirá un registro de las transacciones a partir del cual podremos restaurar la base de datos. Esto puede ser enormemente beneficioso en el caso de caída, ya que cualesquiera cambios que no fueron escritos en la base de datos pueden ser recuperados usando el dato que fue previamente registrado. Una vez el sistema ha quedado restaurado, un usuario puede continuar trabajando desde el punto en que lo dejó cuando cayó la base de datos.

Algunos límites de PostgreSQL se incluyen a continuación**:**

- Máximo tamaño de base de datos ilimitado.
- Máximo tamaño de tabla 32 TB.
- Máximo tamaño de tupla 1.6 TB.
- Máximo tamaño de campo 1 GB.
- Máximo tuplas por tabla ilimitado.
- Máximo columnas por tabla 250 - 1600 dependiendo de los tipos de columnas.
- Máximo de índices por tabla ilimitado.

1.10 Herramienta de modelado de BD ER/Studio:

La utilización de herramientas de modelado permite a los arquitectos de datos y administradores de bases de datos junto con los desarrolladores gestionar y mantener aplicaciones que trabajan con un volumen grande de datos. Entre estas herramientas se encuentra ER/Studio de Embarcadero, que permite transformar, migrar e integrar amplias cantidades de datos procedentes de diversas fuentes.

ER/Studio es una herramienta CASE de modelado de datos fácil de usar y multinivel para el diseño de bases de datos a nivel físico y lógico. Direcciona las necesidades diarias de los administradores de bases de datos, desarrolladores y arquitectos de datos que construyen y mantienen aplicaciones de bases de datos grandes y complejas.

ER/Studio está equipado para crear y manejar diseños de bases de datos funcionales y confiables. Ofrece fuertes capacidades de diseño lógico, sincronización bidireccional de los diseños físicos y lógicos, construcción automática de bases de datos, documentación y fácil creación de reportes.

ER/Studio es simple y fácil de manejarlo y ayuda a las organizaciones para tomar decisiones en cómo resolver embotellamientos de los datos y eliminar redundancia.

Se crea un modelo de datos ofreciendo la posibilidad de crear un modelo de datos nuevo, realizar ingeniería inversa a partir de una base de datos ya existente o importar un archivo SQL o modelo de datos realizado con Erwin.

Genera scripts para diferentes SGBD como son: **(13)**

- Hitachi HiRDB
- InterBase
- Microsoft Access
- Microsoft SQL Server
- Microsoft Visual FoxPro
- MySQL
- Oracle
- PostgreSQL
- Sybase ASA
- Sybase Watcom SQL Anywhere
- Entre otros.

1.11 Herramienta EMS SQL Manager para PostgreSQL:

EMS Manager para PostgreSQL es una poderosa herramienta para la gestión y administración de bases de datos PostgreSQL.

EMS SQL Manager para PostgreSQL es una poderosa herramienta gráfica para la administración y desarrollo de PostgreSQL Database Server (servidor de bases de datos PostgreSQL). PostgreSQL Manager funciona con cualquier versión de PostgreSQL, hasta la 8.1, y soporta todas las nuevas características de PostgreSQL incluyendo espacios de tablas (tablespaces), argumentos nombrados (named arguments) en funciones y otras más. Ofrece una gran variedad de herramientas poderosas para usuarios avanzados, tales como Visual Database Designer (diseñador visual de base de datos), Visual Query Builder (constructor visual de consultas), y un poderoso editor de objetos binarios (BLOB) para satisfacer todas sus necesidades. PostgreSQL Manager cuenta con una nueva y avanzada interfaz gráfica de usuario con un sistema asistente bastante descriptivo, tan claro en su uso que ni un principiante se podrá confundir.

Características principales:

Soporte completo de PostgreSQL hasta la versión 8.1.

Ágil navegación y administración de base de datos.

Administración sencilla de todos los objetos PostgreSQL.

Herramientas de manipulación de datos avanzada.

Administración efectiva de seguridad.

Excelentes herramientas visuales y de texto para elaboración de consultas.

Acceso al servidor PostgreSQL a través del protocolo HTTP.

Impresionantes opciones de exportación e importación de datos.

Poderoso diseñador visual de base de datos.

Asistentes fáciles de usar que realizan mantenimiento de tareas de PostgreSQL.

1.12 Metodología de desarrollo:

RUP es una metodología que está guiada por una fuerte planificación durante todo el proceso de desarrollo; llamada metodología tradicional o clásica, donde se realiza una intensa etapa de análisis y diseño antes de la construcción del sistema.

RUP agrupa las actividades en 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo. EL ciclo de desarrollo en la vida de un producto de software se divide en 4 fases. Lo que se puede apreciar claramente en el gráfico:

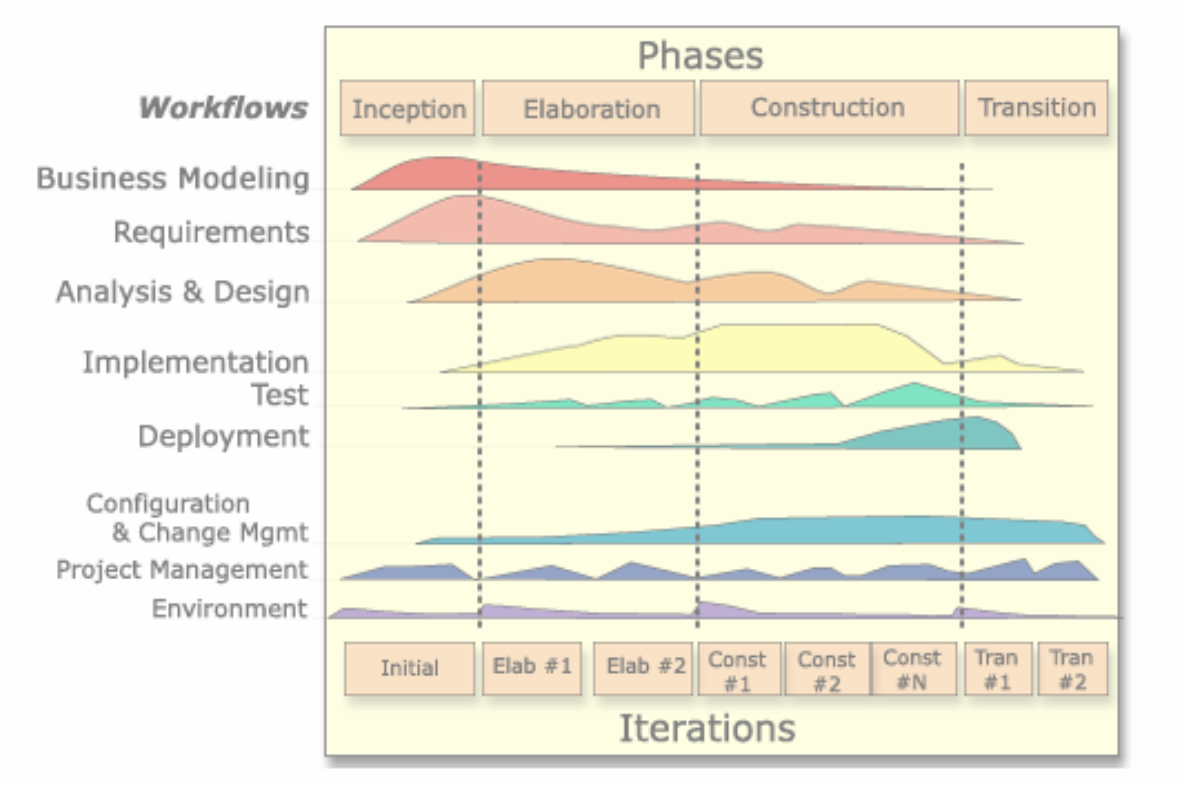

Figura 2: Flujos de trabajo y fases de la metodología RUP

Los elementos de RUP son:

Actividades: Son los procesos que se llegan a determinar en cada iteración.

Trabajadores: Son las personas o entes involucrados en cada proceso.

Artefactos: Puede ser un documento, un modelo, o un elemento de modelo.

El ciclo de vida de desarrollo del software según RUP tiene 3 características principales:

Dirigido por Casos de Uso.

Un sistema surge por la necesidad de darle servicio a un conjunto de usuarios los cuales tienen diferentes necesidades, por lo que necesitan interactuar con el mismo para recibir algún beneficio.

"Un caso de uso es un fragmento de funcionalidad de un sistema que proporciona al usuario un resultado importante." **(14)** Por lo tanto los casos de usos representan los requisitos funcionales del sistema y además guían el proceso de desarrollo del software basándose en el modelo de casos de uso. Los casos de uso se especifican, se diseñan, se implementan y se le hacen prueba para ver si realmente dan los resultados esperados, es decir son un hilo conductor durante todo el proceso de desarrollo del software.

Iterativo e Incremental:

RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto. Cada iteración trata un grupo de casos de uso empezando por los más críticos que juntos amplían las funcionalidades del producto y tratan los riesgos más importantes.

Centrado en la Arquitectura:

La arquitectura de un software se describe mediante diferentes vistas del mismo. Es una vista del diseño con las características más importantes resaltadas obviando los detalles. La arquitectura debe diseñarse para que el sistema evolucione y para que encajen en ella los casos de uso. Existe una doble dependencia entre la arquitectura y los casos de uso. RUP se desarrolla mediante diferentes iteraciones, comenzando primero por los casos de uso relevantes desde el punto de vista de la arquitectura.

1.13 Conclusiones.

Después de haber realizado el estado del arte de las bases de datos, de los sistemas gestores de base de datos y una descripción de cada herramienta a utilizar en la presente investigación se llega a un punto donde se puede comenzar a desarrollar el componente (base de datos) que podrá dar servicio a la aplicación web de gestión de los RRHH en los Polos Productivos de la Facultad 9 de la Universidad de las Ciencias Informáticas.

La presente base de datos se propone que dinámica utilizando el modelo relacional, por otra parte, como se tiene como herramienta CASE para el modelado el ER/Studio, como sistema gestor de bases de datos PostgreSQL, como herramienta para administrar la BD se tiene como propuesta EMS Manager para PostgreSQL y RUP como metodología de desarrollo.

Capítulo 2 Descripción y análisis de la solución propuesta.

Introducción

En el presente capítulo se hace una descripción de la solución propuesta. Se presenta el diagrama de clases persistentes, así como el diagrama físico relacional, se describen cada una de las clases y tablas que se encuentran en los diagramas antes mencionados. Además se hace una descripción de la arquitectura propuesta.

2.1 Descripción de la arquitectura.

La bases de datos propuesta contiene las tablas relacionadas con los RRHH de los Polos Productivos; las mismas van a estar relacionadas principalmente alrededor de los usuarios por ser la tabla principal del diseño de la presente base de datos. Se hace uso de tablas las cuales añaden campos dinámicamente con el objetivo de que se logre dinamismo en la aplicación web, además se hace uso da claves sustitutas o subrogadas con el objetivo de facilitar la manipulación de los datos, siendo todos los identificadores enteros lo que posibilita una mayor velocidad en las consultad a la base de datos, y eliminan el engorroso trabajo del uso de superclaves.

2.2 Descripción de los requisitos funcionales y no funcionales que debe cumplir la solución propuesta.

2.2.1 Requisitos Funcionales.

En la primera iteración se implementarán los casos de usos críticos del sistema, que son los que a continuación se presentan.

RF1. Autenticar usuario

RF2. Gestionar Polo

- RF2.1. Registrar Polo
- RF2.2. Modificar datos del Polo
- RF2.3. Eliminar Polo
- RF2.4. Mostrar datos de un Polo

RF3. Buscar Polo

RF4. Gestionar dato personal que se le va a pedir a los miembros del polo

- RF4.1. Registrar dato personal que se le va a pedir a los miembros del polo
- RF4.2. Eliminar dato personal que se le va a pedir a los miembros del polo
- RF4.3. Mostrar dato personal que se le va a pedir a los miembros del polo
- RF4.4. Modificar dato personal que se le va a pedir a los miembros del polo

RF5. Buscar dato personal que se le va a pedir a los miembros del polo

RF6. Gestionar Rol

- RF6.1. Registrar Rol
- RF6.2. Modificar Rol
- RF6.3. Eliminar Rol
- RF6.4. Mostrar Rol

RF7. Buscar Rol

RF8. Gestionar Categoría

RF8.1. Registrar Categoría

- RF8.2. Modificar Categoría
- RF8.3. Eliminar Categoría
- RF8.4. Mostrar Categoría

RF9. Buscar Categoría

RF10. Gestionar PC

- RF10.1. Registrar PC
- RF10.2. Modificar PC
- RF10.3. Eliminar PC
- RF10.4. Mostrar PC

RF11. Buscar PC

RF12. Gestionar tiempo de máquina

- RF12.1. Registrar tiempo de máquina
- RF12.2. Modificar tiempo de máquina
- RF12.3. Eliminar tiempo de máquina
- RF12.4. Mostrar tiempo de máquina

RF13. Buscar tiempo de máquina

RF14. Gestionar EPI

- RF14.1. Registrar EPI
- RF14.2. Modificar EPI
- RF14.3. Eliminar EPI
- RF14.4. Mostrar EPI

RF15. Buscar EPI

RF18. Gestionar tipo de dato

- RF18.1. Registrar tipo de dato
- RF18.2. Modificar tipo de dato
- RF18.3. Eliminar tipo de dato
- RF18.4. Mostrar tipo de dato

RF19. Buscar tipo de dato

RF20. Gestionar usuario

- RF20.1. Registrar usuario
- RF20.2. Modificar usuario
- RF20.3. Eliminar usuario
- RF20.4. Mostrar usuario

RF21. Buscar usuario

RF22. Gestionar datos personales

- RF22.1. Registrar datos personales
- RF22.2. Modificar datos personales
- RF22.4. Mostrar datos personales

RF23. Buscar datos personales

RF26. Gestionar evento

- RF26.1. Registrar evento
- RF26.2. Modificar evento
- RF26.3. Eliminar evento
- RF26.4. Mostrar evento

RF27. Buscar evento.

RF28. Gestionar curso optativo del segundo perfil

- RF28.1. Registrar curso optativo del segundo perfil
- RF28.2. Modificar curso optativo del segundo perfil
- RF28.3. Eliminar curso optativo del segundo perfil
- RF28.4. Mostrar curso optativo del segundo perfil

RF29. Buscar curso optativo del segundo perfil

RF30. Gestionar noticia

- RF30.1. Registrar noticia
- RF30.2. Modificar noticia
- RF30.3. Eliminar noticia
- RF30.4. Mostrar noticia

RF31. Buscar noticia

RF32. Gestionar evaluación

- RF32.1. Registrar evaluación
- RF32.2. Modificar evaluación
- RF32.3. Eliminar evaluación
- RF32.4. Mostrar evaluación

RF33. Buscar evaluación.

RF34. Generar Reporte

RF35. Exportar Reporte

RF36. Imprimir Reporte

RF37. Ver tiempo de máquina asignado al usuario que está logueado en ese momento en el sistema

RF39. Ver las evaluaciones otorgadas al usuario que está logueado en ese momento en el sistema

RF40. Ver noticia publicada en la semana

RF41. Buscar información en las noticias por un criterio dado por el usuario del sistema

2.2.2 Requisitos no funcionales.

Los requisitos no funcionales mencionados son sólo los que tienen correspondencia con el desarrollo de la BD.

Requerimientos de software:

1. El sistema deberá tener como gestor de Base de datos PostgreSQL 8.2.

Requerimientos de hardware:

- Servidor de Base Datos
    1. Periféricos: Mouse, teclado.
    2. Tarjeta de Red.
    3. 1GB de RAM o superior.
    4. 120 o más GB de disco.

Requerimientos de Seguridad

1- El sistema deberá ser capaz de permitir que cada usuario se autentique.

2- El sistema deberá garantizar que las funcionalidades se muestren de acuerdo al nivel de usuario que este activo.

3- El sistema deberá estar protegido contra acciones que no estén autorizadas o que puedan afectar la integridad de los datos.

Requerimientos de rendimiento

1- El sistema deberá tener un tiempo de respuesta rápida a cualquier solicitud, para ello se utilizará al máximo los recursos de las máquinas donde esté instalado el sistema.

2.3 Diagramas de Clases Persistentes.

Las clases persistentes tienen como origen las clases entidades mediante las cuales se modelan datos perdurables en el tiempo. Los diagramas de clases persistentes se basan en las entidades de los diagramas de clases del análisis y fundamentalmente en los diagramas de clases del diseño. A continuación se muestran los diagramas de

clases del diseño para cada módulo en los que se estructuró el diagrama general de clases persistentes.

Módulo 1 de estructura general

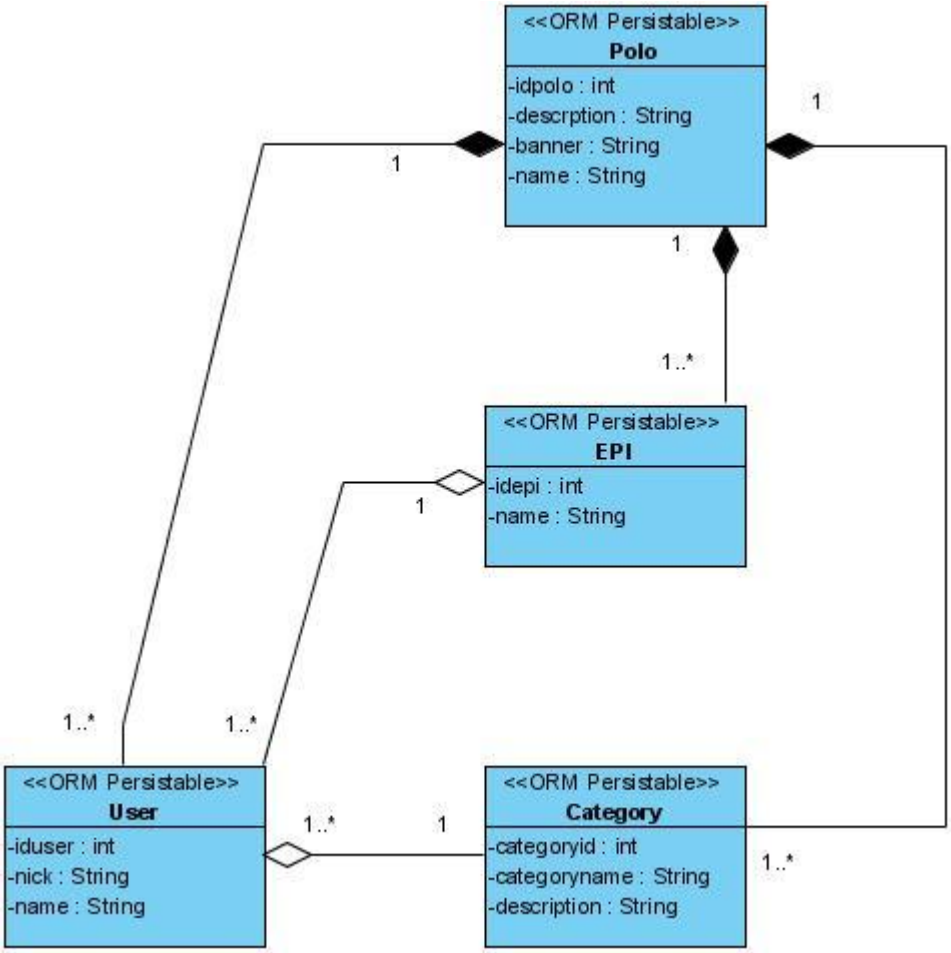

Figura 3: Diagrama de clases persistentes Módulo de estructura general

Módulo 2 Datos definidos para los usuarios

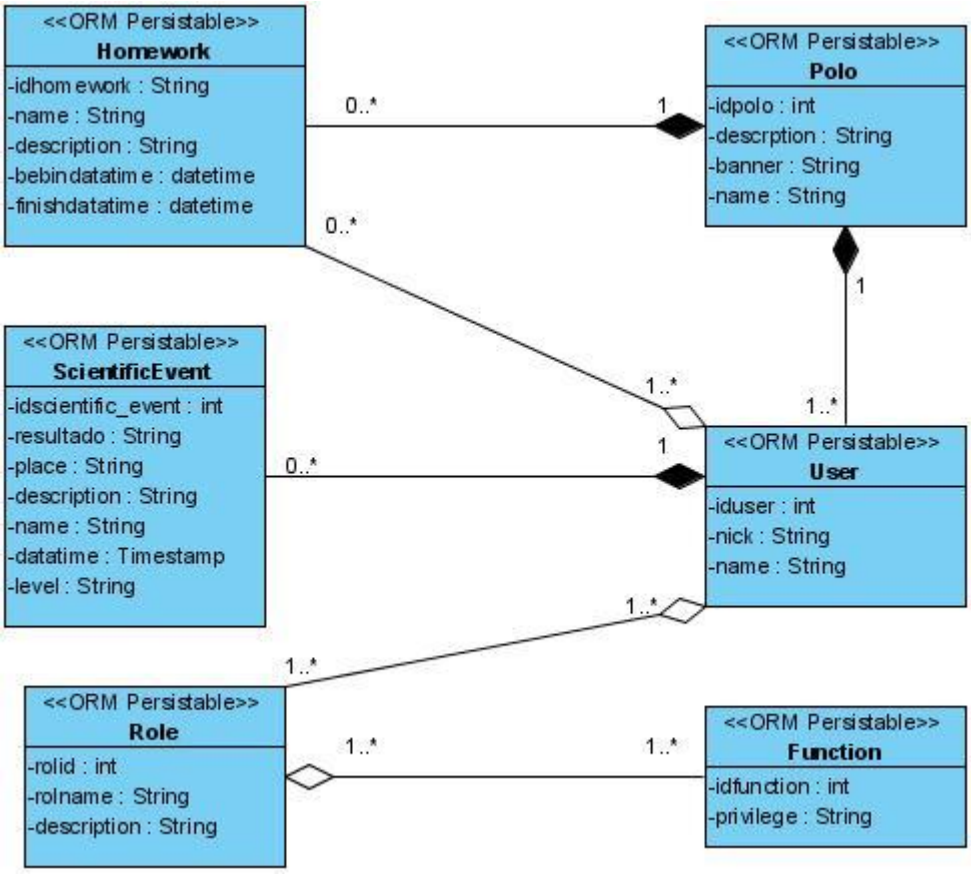

Figura 4 : Diagrama de clases persistentes Módulo datos definidos para usuarios

Módulo 3 Evaluación y cursos optativos

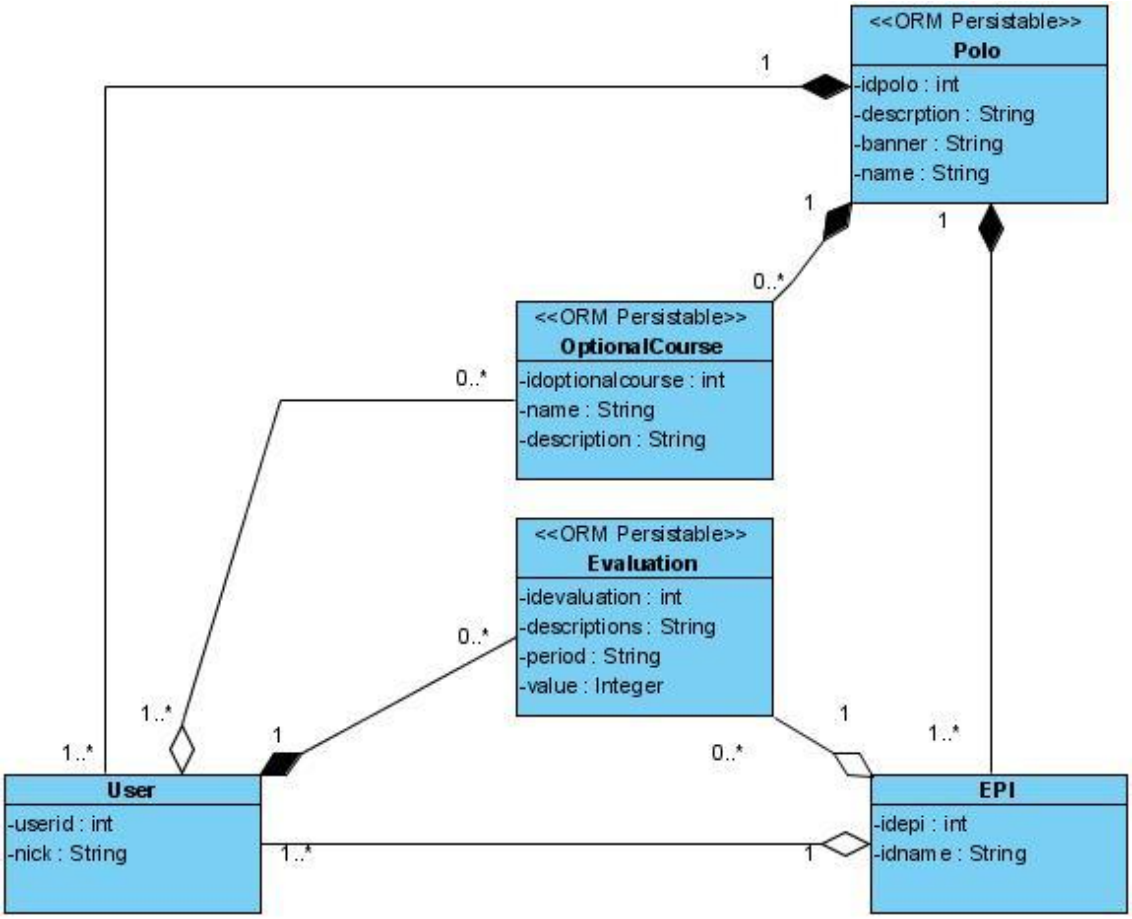

Figura 5: Diagrama de clases persistentes Módulo evaluación y cursos optativos

Módulo 4 Tiempo de Máquina y noticia

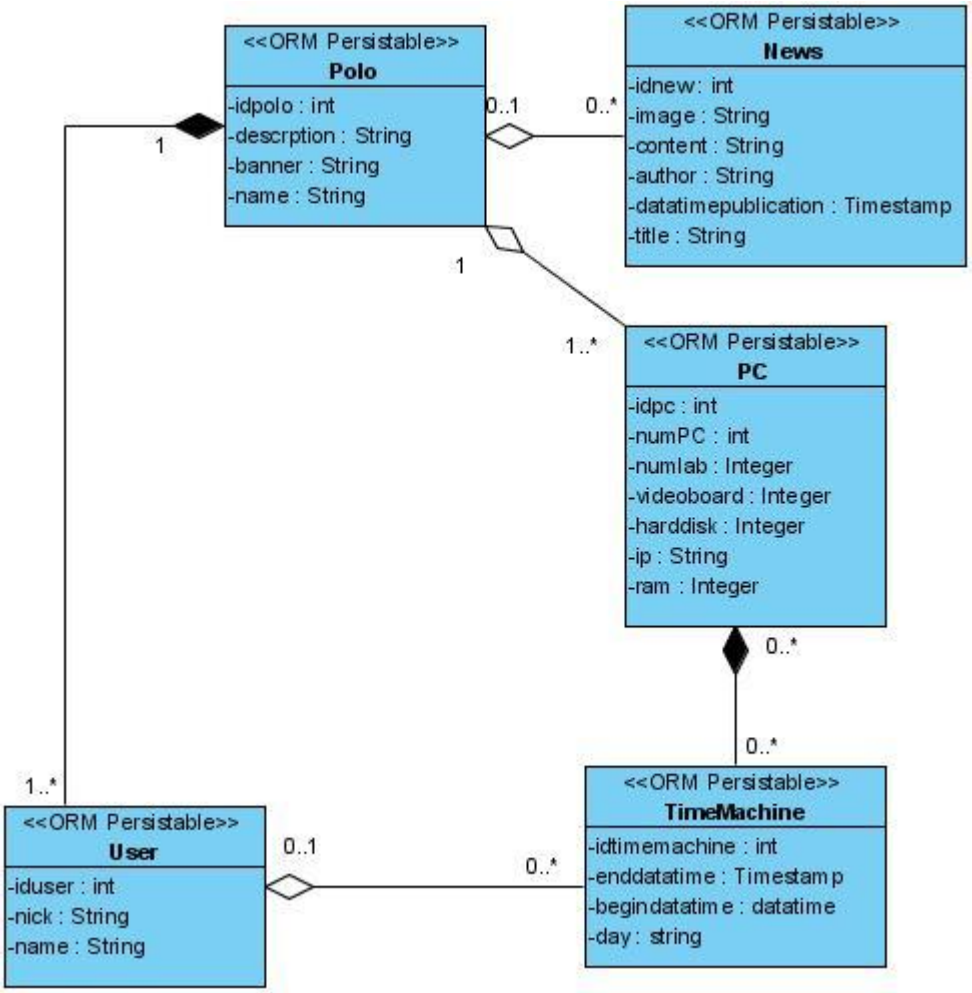

Figura 6: Diagrama de clases persistentes Módulo Tiempo de Máquina y noticias

Módulo 5 de datos dinámicos

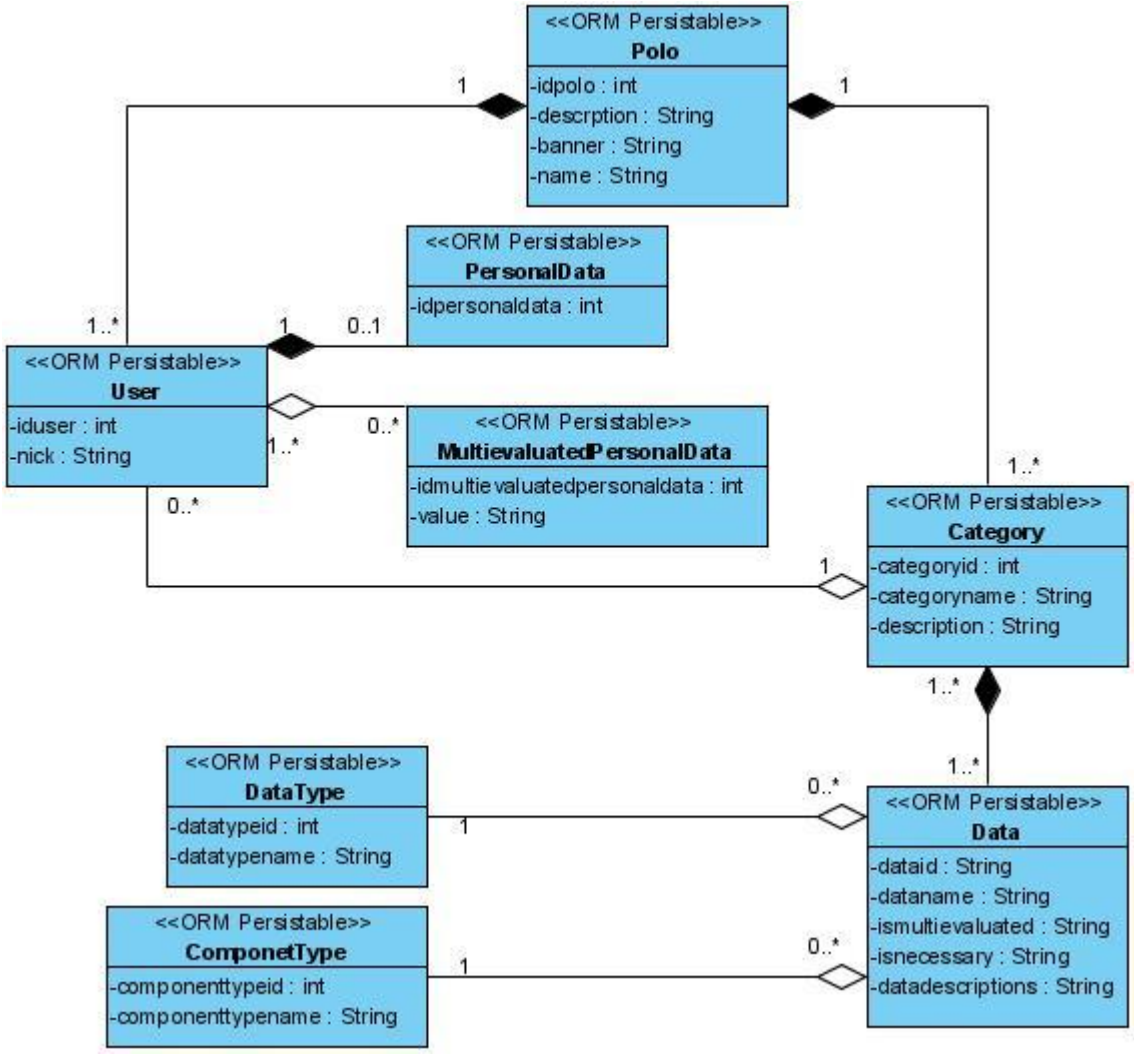

Figura 7: Diagrama de clases persistentes Módulo de datos dinámicos

Diagrama general de clases persistentes

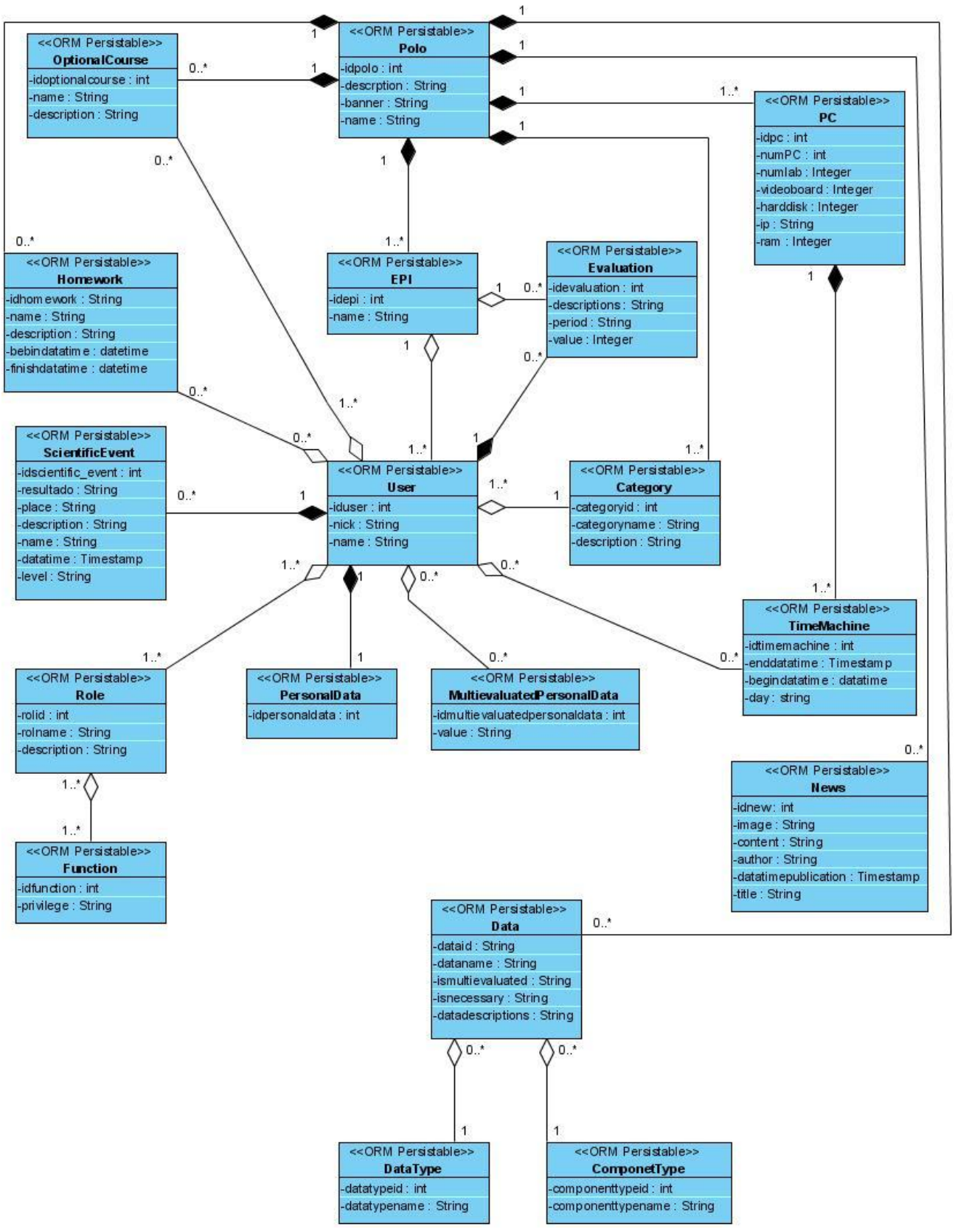

Figura 8: Diagrama de clases persistentes

2.3.1 Descripción de las clases persistentes

Tabla 1: CP Polo

Polo	
Atributo	**Tipo**
idpolo	**int**
description	**string**
banner	**string**
name	**string**

Tabla 2: CP EPI

EPI	
Atributo	**Tipo**
idepi	**int**
name	**string**

Tabla 3: CP User

User	
Atributo	**Tipo**
iduser	**int**
nick	**string**

Tabla 4: CP Category

Category	
Atributo	**Tipo**
idcategory	**int**
namecategory	**string**
description	**string**

Tabla 5: CP Homework

Homework	
Atributo	**Tipo**
id	**int**
name	**string**
description	**string**
begindatetime	**timestamp**
enddatatime	**timestamp**

Tabla 6: CP ScientificEvent

ScientificEvent	
Atributo	**Tipo**
id	**int**
result	**string**
place	**string**
description	**string**
nameevent	**string**
level	**string**

Tabla 7: CP Role

Role	
Atributo	**Tipo**
id	**int**
name	**string**
description	**string**

Tabla 8: CP Function

Function	
Atributo	**Tipo**
id	**int**
privilege	**string**

Tabla 9: CP Evaluation

Evaluation	
Atributo	**Tipo**
Id	**Int**
description	**string**
period	**string**
value	**int**

Tabla 10: CP OptativeCourse

OptativeCourse	
Atributo	**Tipo**
id	**int**
name	**string**
description	**string**

Tabla 11: CP News

News	
Atributo	**Tipo**
id	**int**
image	**string**
content	**string**

Autor	**string**
title	**string**
datetimepublication	**datetime**

Tabla 12: CP PC

PC	
Atributo	**Tipo**
id	**int**
numpc	**int**
harddisk	**int**
ip	**string**
ram	**int**
videoboard	**int**
numlab	**int**

Tabla 13: CP TimeMachine

TimeMachine	
Atributo	**Tipo**
id	**Int**
begindatatime	**datatime**
enddatatime	**datatime**
day	**string**

Tabla 14: CP Category

Category	
Atributo	**Tipo**
id	**int**

name	**string**
description	**string**

Tabla 15:CP Data

Data	
Atributo	**Tipo**
Id	**int**
name	**string**
Ismultievaluated	**bool**
isnecessary	**bool**
description	**string**

Tabla 16: CP DataType

DataType	
Atributo	**Tipo**
id	**int**
name	**string**

Tabla 17: CP ComponentType

ComponentType	
Atributo	**Tipo**
id	**int**
name	**string**

Tabla 18: CP PersonalData

PersonalData	
Atributo	**Tipo**

id	**int**

Tabla 19: CP MultievaluatedPersonalData

PersonalData	
Atributo	**Tipo**
id	**int**
value	**string**

2.4 Modelo relacional físico de la base de datos.

En esta sección se muestra el diagrama físico de la base de datos el cuál para su mejor comprensión se encuentra divido en módulos.

Módulo 1 Estructura general

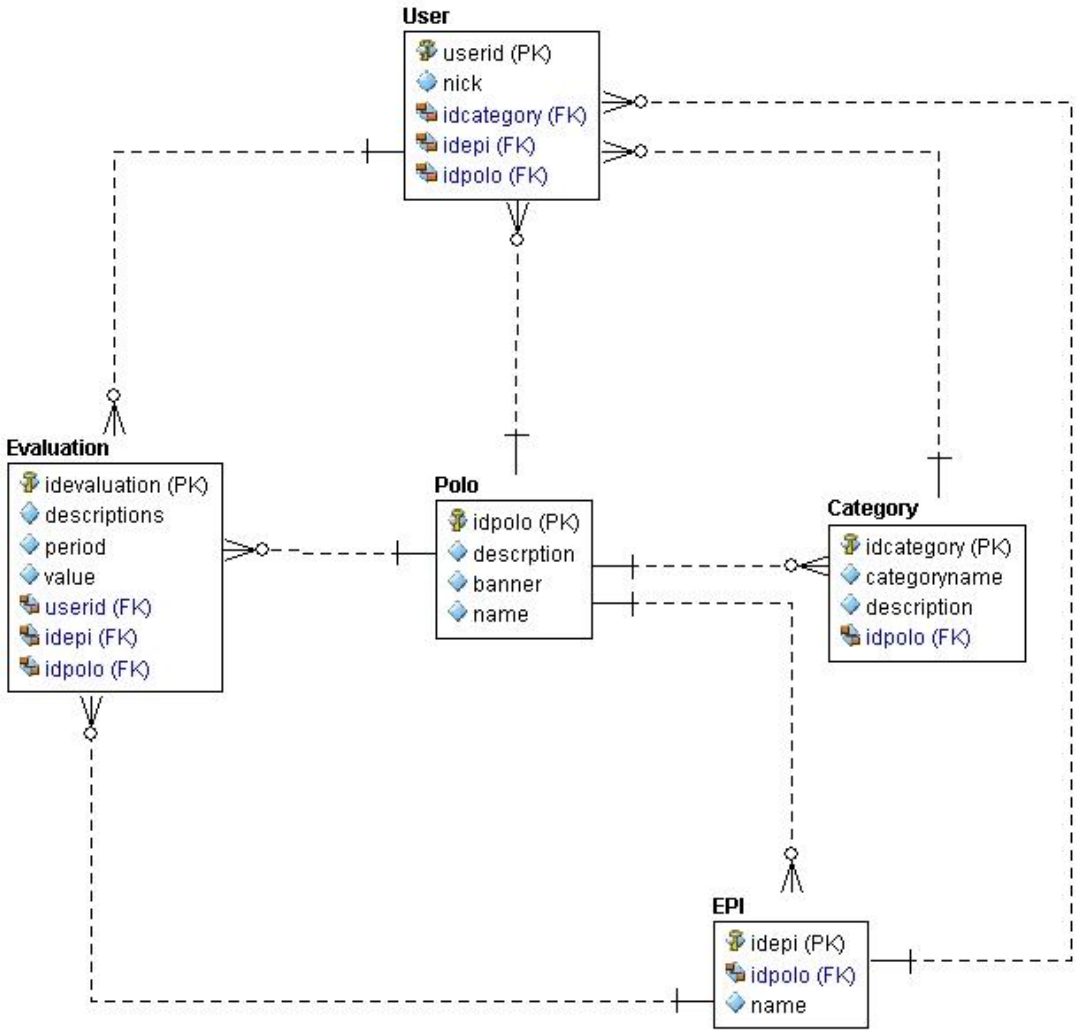

Figura 9: Diagrama relacional Módulo Estructura General

Módulo 2 Datos definidos para los usuarios

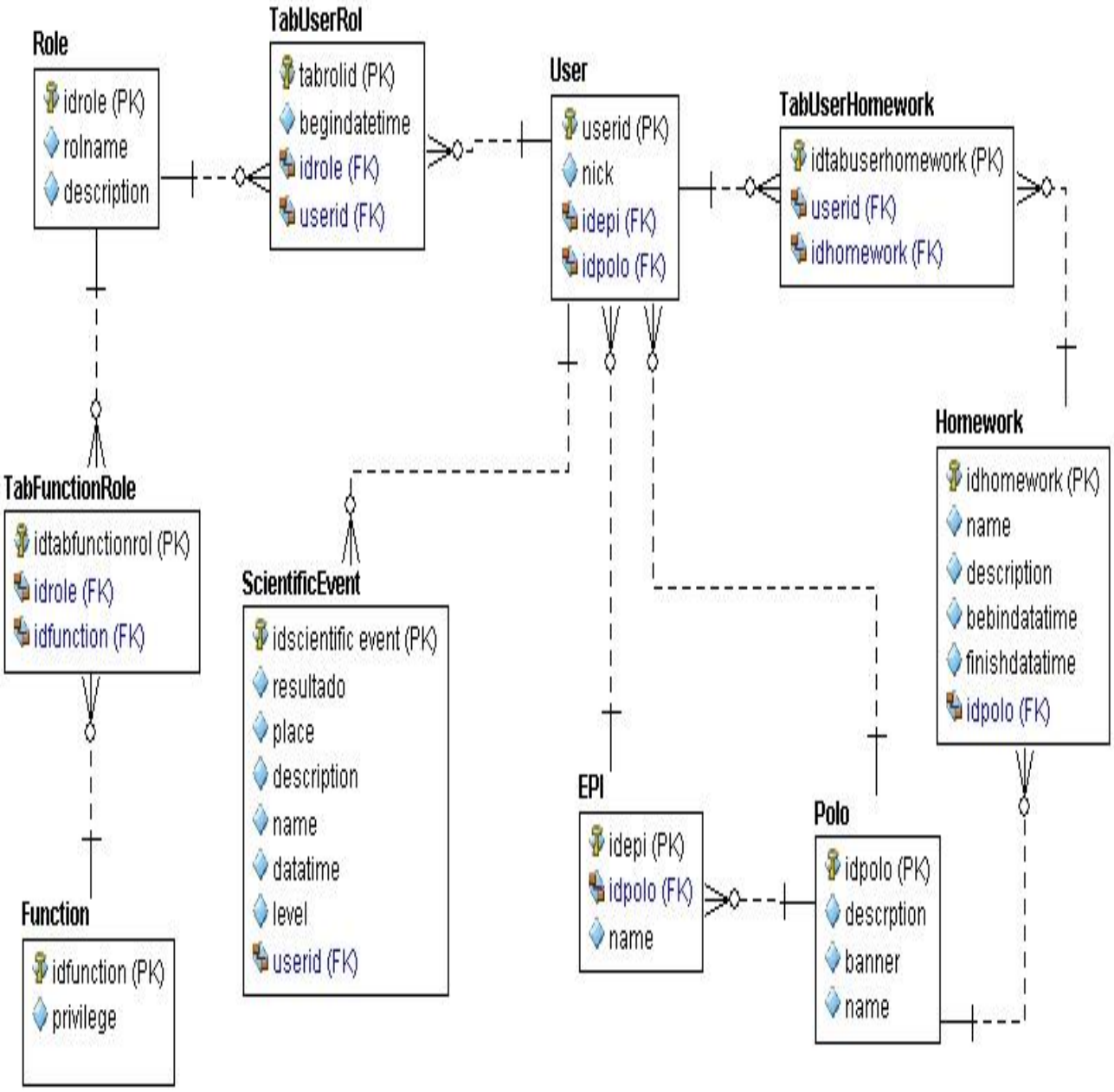

Figura 10: Módulo datos definidos para los usuarios

Módulo 3 Evaluación y cursos optativos

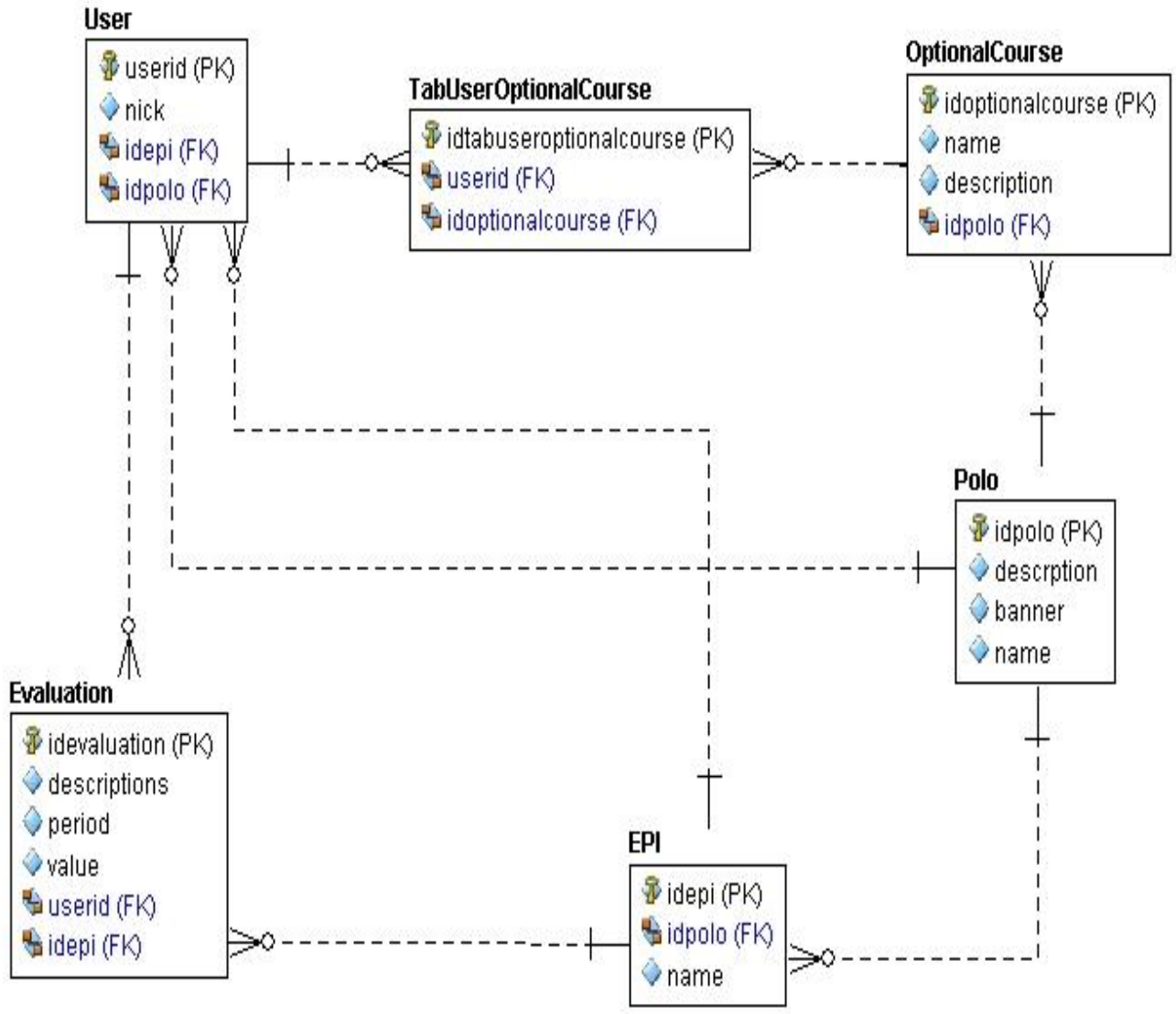

Figura 11: Evaluación y Cursos Optativos

Módulo 4 tiempo de máquina y noticias

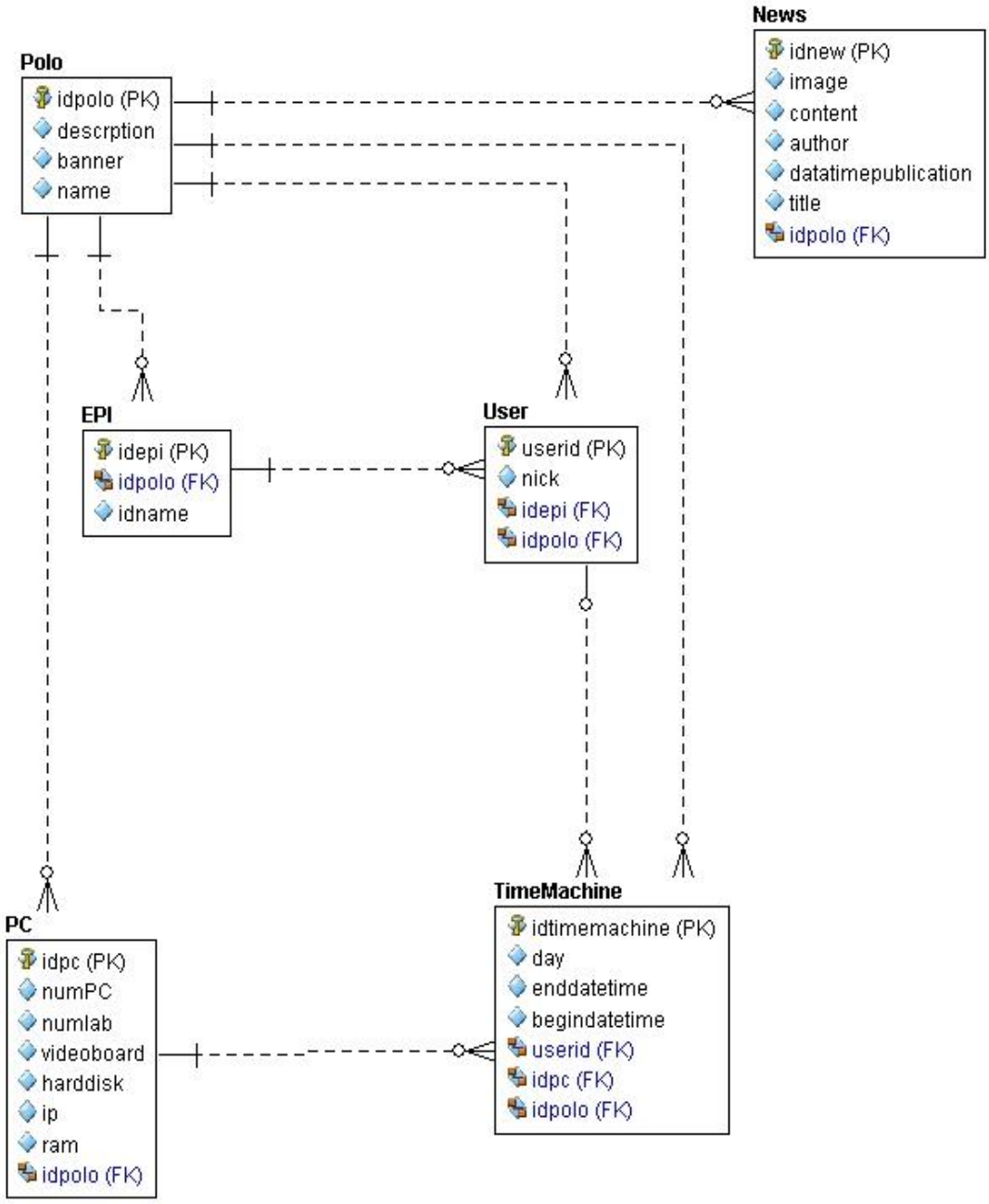

Figura 12: Módulo tiempo de máquina y noticias

Módulo 5 Datos Dinámicos

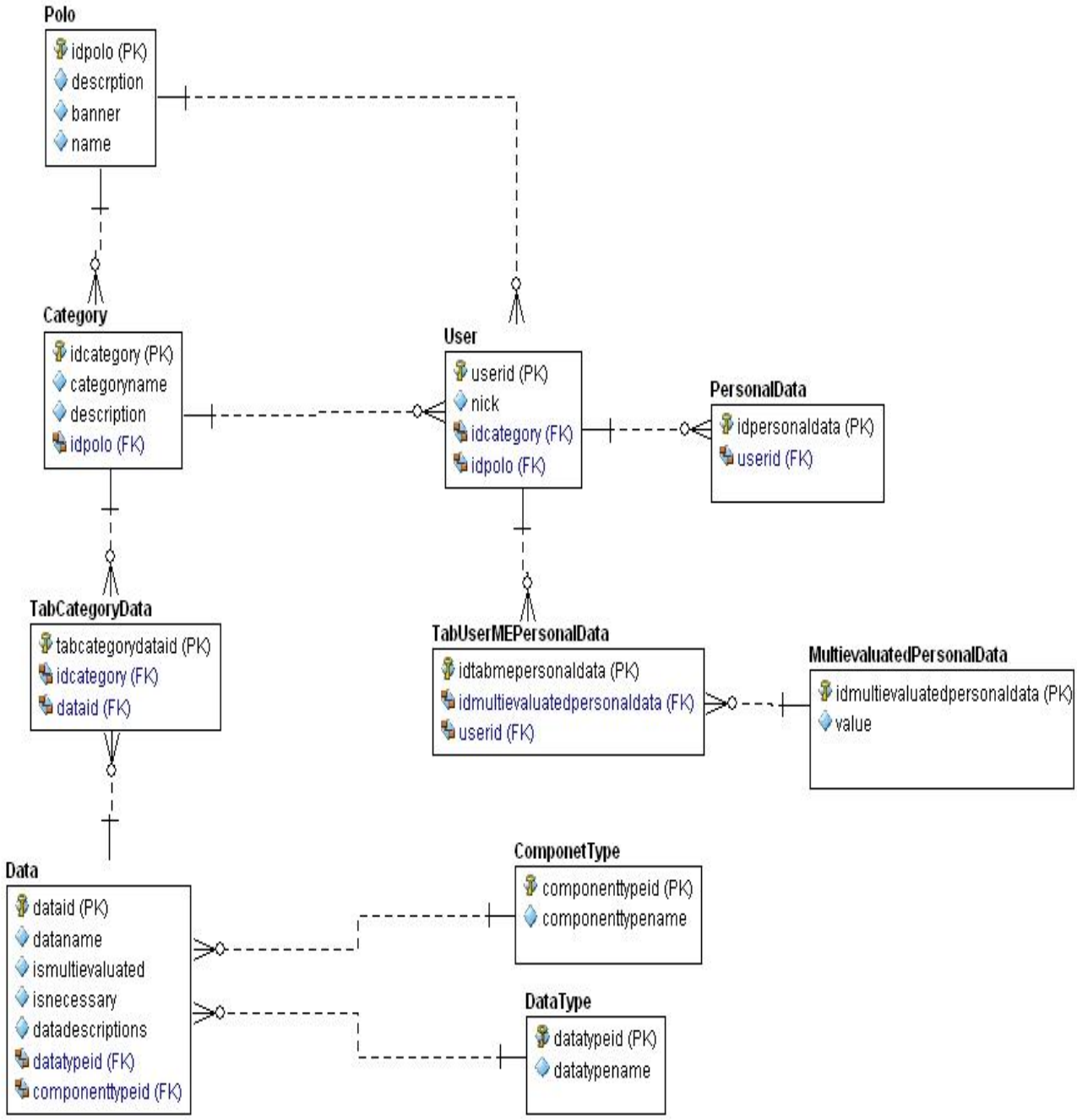

Figura 13: Módulo datos dinámicos.

Modelo relacional general de la base de datos

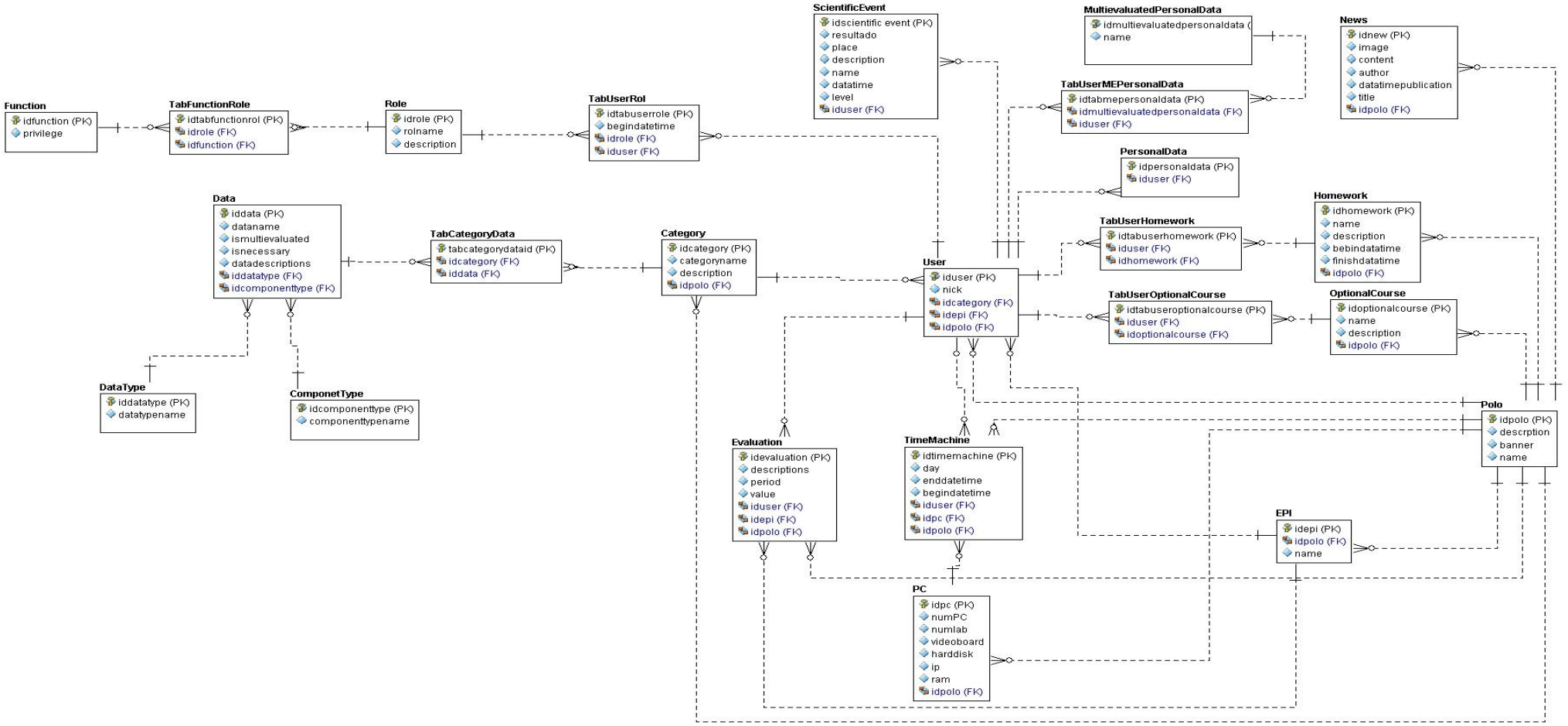

Figura 14: Diagrama relacional general

2.4.1 Descripción de las relaciones.

Tabla 20: User

Nombre: User		
Descripción: Almacena los usuarios del sistema		
Atributo	**Tipo**	**Descripción**
iduser	**integer**	**Identificador de la tabla**
nick	**varchar(30)**	**Almacena el nick del usuario UCI**
idategory(FK)	**integer**	**Almacena el id de la categoría que presenta**
idepi(FK)	**integer**	**Almacena el id de la EPI a la cual está asociado**
idpolo (FK)	**integer**	**Almacena el id del Polo al que hace referencia(pertenece)**

Tabla 21: Category

Nombre: Category		
Descripción: Almacena las categorías a que pertenecen los usuarios de los Polos		
Atributo	**Tipo**	**Descripción**
idcategory	**integer**	**Identificador de la tabla**
categoryname	**varchar(40)**	**Especifica el nombre de la categoría**
description	**text**	**Almacena la descripción de la categoría**
idpolo (FK)	**integer**	**Almacena el id del Polo al que hace referencia(pertenece)**

Tabla 22: Data

Nombre: Data		
Descripción: Metadato que almacena los datos a pedir de cada usuario por categoría		
Atributo	**Tipo**	**Descripción**

iddata	**integer**	**Identificador de la tabla**
dataname	**varchar(30)**	**Almacena el nombre del dato**
ismultievaluated	**bit**	**Especifica si el dato es multievaluado**
isnecessary	**bit**	**Especifica si el dato puede ser nulo**
description	**text**	**Describe las características del metadato**
iddatatype(FK)	**int**	**Id del tipo de dato que presenta el metadato**
idcomponenttype(FK)	**int**	**Id del tipo de componente**

Tabla 23: TabCategoryData

Nombre: TabCategoryData		
Descripción: Surge de la relación de muchos a muchos y da la posibilidad de relacionar una categoría con diferentes datos, y viceversa, ascociar un mismo dato con varias categorías.		
Atributo	**Tipo**	**Descripción**
idcategorydata	**integer**	**Identificador de la tabla**
iddategory(FK)	**integer**	**Id de la categoría referenciada**
iddata(FK)	**integer**	**Id del dato referenciado**

Tabla 24: DataType

Nombre: DataType		
Descripción: Almacena el tipo de dato que va a tomar el metadato (entero, float, varchar, texto, booleano).		
Atributo	**Tipo**	**Descripción**
iddatatype	**integer**	**Identificador de la tabla**
datatypename	**varchar(30)**	**Almacena el nombre del tipo de dato**

Tabla 25: ComponentType

Nombre: ComponentType		
Descripción: Almacena el tipo de componente html mediante el cual se va a mostrar en el formulario dinámicamente para entrar el valor del dato(text, textarea, password, checkbox, radiobuton).		
Atributo	**Tipo**	**Descripción**
idcomponenttype	**integer**	**Identificador de la tabla**
namecomponenttype	**varchar(30)**	**Almacena el nombre del tipo de componente**

Tabla 26: Role

Nombre: Role		
Descripción: Almacena los roles que presentan los usuarios del sistema		
Atributo	**Tipo**	**Descripción**
idrol	**integer**	**Identificador de la tabla**
rolname	**varchar(30)**	**Almacena el nombre del rol**
description	**text**	**Almacena la descripción del rol**
idpolo (FK)	**integer**	**Almacena el id del Polo al que hace referencia(pertenece)**

Tabla 27: TabUserRole

Nombre: TabUserRole		
Descripción: Almacena los valores de la nueva tabla creada por la relación de muchos a muchos entre la tabla usuario y rol.		
Atributo	**Tipo**	**Descripción**
idtabuserrol	**integer**	**Identificador de la tabla**
begindatatime	**datatime**	**Almacena la hora en la que se le da un rol a un usuario**
Idrol(FK)	**integer**	**Id del rol referenciado**
Iduser(FK)	**integer**	**Id del usuario referenciado**

Tabla 28: Function

Nombre: Function		
Descripción: Almacena una función determinada la cual el usuario puede hacer según el rol.		
Atributo	**Tipo**	**Descripción**
idfunction	**integer**	**Identificador de la tabla**
privelege	**varchar(100)**	**Almacena el nombre de la función**

Tabla 29: TabRolFunction

Nombre: TabRolFunction		
Descripción: Almacena las tuplas de la relación de muchos a muchos entre rol y función debido a qué un rol puede tener asignado varias funciones, y una función puede ser llevada a cabo por varios roles.		
Atributo	**Tipo**	**Descripción**
idtabrolfunction	**integer**	**Identificador de la tabla**
idtabrol(FK)	**integer**	**Id del rol referenciado**
idfunction(FK)	**integer**	**Id de la función referenciada**

Tabla 30: ScientificEvent

Nombre: ScientificEvent		
Descripción: Almacena los eventos científicos en los que han participado los usuarios.		
Atributo	**Tipo**	**Descripción**
idscientificevent	**integer**	**Identificador de la tabla**
result	**varchar(30)**	**Almacena el resultado obtenido en el evento**
place	**varchar(50)**	**Almacena el lugar donde se celebró el evento**
description	**text**	**Almacena una description acerca de la participación en el evento**
datetime	**datetime**	**Almacena la fecha en que el usuario**

		participó en el evento
iduser	**integer**	**Almacena el id del usuario referenciado**
idlevel	**integer**	**Almacena el id del nivel del evento referenciado**

Tabla 31: PersonalData

Nombre: PersonalData		
Descripción: Almacena los datos personales de un usuario		
Atributo	**Tipo**	**Descripción**
idpersonaldata	**integer**	**Identificador de la tabla**
uduser(FK)	**integer**	**Id del usuario relacionado**

Tabla 32: MultievaluatedPersonalData

Nombre: MultievaluatedPersonalData		
Descripción: Almacena los datos multievaluados		
Atributo	**Tipo**	**Descripción**
idmultievaluateddata	**integer**	**Identificador de la tabla**
value	**text**	**Almacena el valor de cada atributo multievaluado**
iduser(FK)	**integer**	**Almacena el id del usuario al que pertenece el presente dato multievaluado**

Tabla 33: EPI

Nombre: EPI		
Descripción: Almacena las EPI existentes en el Polo		
Atributo	**Tipo**	**Descripción**
idepi	**integer**	**Identificador de la tabla**
idpolo(FK)	**integer**	**Almacena la referencia al id del polo al que pertenece la EPI**

name	**varchar**	**Almacena el nombre de la EPI**

Tabla 34: Polo

Nombre: Polo		
Descripción: Almacena los datos pertenecientes al Polo		
Atributo	**Tipo**	**Descripción**
idpolo	**integer**	**Identificador de la tabla**
name	**varchar(50)**	**Almacena el nombre del polo**
description	**text**	**Almacena una descripción acerca del Polo**
banner	**varchar(100)**	**Almacena la dirección de la imagen del banner perteneciente al Polo**
iduser(FK)	**integer**	**Alamcena la referencia al usuario que es jefe de Polo**

Tabla 35: Evaluation

Nombre: Evaluation		
Descripción: Almacena la evaluación dada a un estudiante por el Polo		
Atributo	**Tipo**	**Descripción**
idevaluation	**integer**	**Identificador de la tabla**
period	**varchar(15)**	**Almacena el período en que se realiza la evaluación**
value	**integer**	**Almacena el valor de la evaluación**
description	**text**	**Almacena una descripción acerca de la evaluación**
idepi(FK)	**integer**	**Referencia al id de la EPI**
iduser(FK)	**integer**	**Referencia al id del estudiante**

Tabla 36: OptionalCourse

Nombre: OptionalCourse

Descripción: Almacena los datos de los cursos optativos		
Atributo	**Tipo**	**Descripción**
idoptionalcourse	integer	Identificador de la tabla
name	varchar(50)	Almacena el nombre del curso optativo
descripcion	text	Almacena una descripción
epi(FK)	integer	Almacena la referencia acerca de la EPI a que pertenece
idpolo(FK)	integer	Almacena la referencia al polo que pertenece

Tabla 37: TabUserOptionalCourse

Nombre: TabUserOptionalCourse		
Descripción: Almacena la relación de muchos a muchos que permite relacionar un usuario con varios cursos optativos, y un curso optativo con varios usuarios.		
Atributo	**Tipo**	**Descripción**
idtabuseroptionalcourse	integer	Identificador de la tabla
iduser(FK)	integer	Almacena el id del usuario al cual hace referencia
idoptionalcourse(FK)	integer	Almacena el id del curso optativo al cual hace referencia

Tabla 38: PC

Nombre: PC		
Descripción: Almacena los valores de la PC		
Atributo	**Tipo**	**Descripción**
idpc	integer	Identificador de la tabla
numpc	integer	Almacena el número de la PC
ip	varchar(15)	Almacena el IP de la PC
ram	integer	Almacena el tamaño de la ram de la PC
harddisk	integer	Almacena la capacidad del disco duro de la PC

videoboard	**integer**	**Almacena el tamaño de la tarjeta de video de la PC**
description	**text**	**Almacena la descripción general acerca de la PC**
lab	**integer**	**Almacena el número del laboratorio al cual pertenece**
Idpolo(FK)	**integer**	**Almacena la referencia al polo que pertenece**

Tabla 39: TimeMachine

Nombre: TimeMachine		
Descripción: Almacena los tiempos de máquina		
Atributo	**Tipo**	**Descripción**
idtimemachine	**integer**	**Identificador de la tabla**
idpc(FK)	**Integer**	**Almacena la referencia a la PC en la cual se va a brindar el tiempo de máquina**
idturn(FK)	**Integer**	**Almacena la referencia al turno en que la PC va a brindar el servicio**
iduser(FK)	**integer**	**Almacena la referencia al usuario al que se le va a brindar servicio en un turno determinado**
Idpolo(FK)	**integer**	**Almacena la referencia al polo que petenece**

Tabla 40: Homework

Nombre: Homework		
Descripción: Almacena la noticia		
Atributo	**Tipo**	**Descripción**
idhomework	**integer**	**Identificador de la tabla**
name	**varchar(50)**	**Almacena la dirección de la imagen relacionada con la noticia**
description	**text**	**Almacena el contenido de la noticia**

author	**varchar(100)**	**Almacena el nombre del autor de la noticia**
begondatatime	**datetime**	**Almacena la fecha de publicación**
Idpolo(FK)	**integer**	**Almacena el id del polo al que pertenece**

Tabla 41: TabUserHomework

Nombre: TabUserHomework		
Descripción: Almacena la noticia		
Atributo	**Tipo**	**Descripción**
idtabuserhomework	**integer**	**Identificador de la tabla**
Idhomework(FK)	**integer**	**Almacena el id de la tarea que relaciona**
Iduser(FK)	**integer**	**Almacena el id del user que referencia**

Tabla 42: News

Nombre: New		
Descripción: Almacena la noticia		
Atributo	**Tipo**	**Descripción**
idnew	**integer**	**Identificador de la tabla**
image	**varchar(50)**	**Almacena la dirección de la imagen relacionada con la noticia**
content	**text**	**Almacena el contenido de la noticia**
author	**varchar(100)**	**Almacena el nombre del autor de la noticia**
datatimepublication	**datetime**	**Almacena la fecha de publicación**
title	**varchar(50)**	**Almacena el título**
font	**varchar(30)**	**Almacena la fuente de donde se extrajo la noticia**
idpolo	**integer**	**Almacena la referencia al polo que la publica**

2.5 Optimización de consultas.

Mediante la optimización de consultas se pretende mejorar los tiempos de ejecución y por tanto de respuesta de una consulta. Para ello se debe trabajar en varios aspectos los cuáles inciden directamente en el tiempo final de las consultas realizadas.

En bases de datos relacionales el lenguaje de consultas SQL es el más utilizado por el común de los programadores y desarrolladores para obtener información desde la base de datos. La complejidad que pueden alcanzar algunas consultas puede ser tal, que el diseño de una consulta puede tomar un tiempo considerable, obteniendo no siempre una respuesta óptima. **(26)**

A la hora de diseñar la base de datos se debe tener en cuenta ajustar al máximo el tamaño de los campos con lo que se evita ocupar mayor espacio y agilizar las consultas relacionadas con los mismos, además se debe valorar el trabajo con índices de las llaves primarias y foráneas, así como atributos mediante los cuales se realizarán búsquedas o campos por los cuales se va a ordenar el resultado obtenido.

La utilización de índices aumenta el tamaño del campo por lo que dificulta las operaciones de inserción, actualización y modificación; y por otro lado agiliza los JOIN cuando forma parte de las llaves primaria y foráneas de las tablas relacionadas. Las consultas que más demoran en ejecutarse son aquellas que tienen JOINs incluidos máxime cuando las tablas involucradas presentan gran números de registros los cuales deben ser recorridos. Entonces se debe tomar la decisión de aumentar o no el tamaño de las tablas con el uso de índices produciendo una mayor velocidad de respuesta en la obtención de tuplas, a costa de un mayor tiempo de ejecución en consulatas sobre las tablas de insercion y modificación.

2.6 Conclusiones.

Una vez modelado la bases de datos lógicas estableciendo dominios y reglas de integridad primarias como las de entidad y referencial, se genera la base de datos físicas lista para montarla en el SGBD especificado por el diseñador de bases de datos con un amplio por ciento de garantías de que la base de datos tiene las características básicas para comenzar a dar servicios.

Capítulo 3 Validación del diseño realizado.

Introducción

En el presente capítulo se brinda información sobre la validación teórica realizada al diseño descrito en el capítulo 2, las reglas de integridad, la normalización y el análisis de la redundancia. Se expone la validación funcional de la base de datos, la optimización de consultas y los resultados de las pruebas realizadas.

3.1 Validación teórica del diseño.

3.1.1 Integridad.

El término integridad de datos se refiere a la corrección y completitud de los datos en una base de datos. Cuando la información de una base de datos se modifican directamente en la BD o mediante sentencias SQL como INSERT, UPDATE o DELETE la misma puede perder la integridad de los datos o quedar en un estado inconsistente. Además se debe tener una redundancia mínima lo que evita inconsistencia en la misma.

El SGBD provee una serie de restricciones que garantizan la integridad de la base de datos. PostgreSQL presenta restricciones de integridad de varios tipos, los cuales se explican a continuación:

En la base de datos fruto de la presente investigación se tiene una tabla homework cuya declaración se presenta a continuación, la cual presenta una serie de restricciones de integridad las cuales se explican a continuación:

```
CREATE TABLE "public"."homework" (
  "idhomework" SERIAL,
  "name" VARCHAR(100),
  "description" TEXT,
  "begindatatime" TIMESTAMP WITHOUT TIME ZONE,
  "finishdatatime" TIMESTAMP WITHOUT TIME ZONE,
  "idpolo" INTEGER NOT NULL,
  CONSTRAINT "PK44" PRIMARY KEY ("idhomework"),
```

CONSTRAINT "RefPolo120" FOREIGN KEY ("idpolo")

REFERENCES "public"."polo"("idpolo")

ON DELETE CASCADE

ON UPDATE CASCADE

);

Restricción de dominio:

En la tabla homework cada atributo está acompañado de su dominio lo cual obliga a que esos campos contengan ese tipo de dato.

Restricción NOT NULL

Se presenta el atributo idpolo el cual no puede ser nulo lo que se específica mediante la restricción NOT NULL.

"idpolo" INTEGER NOT NULL,

Restricción de FOREIGN KEY (llave foránea)

Además existen referencias de integridad lo cual expresa que un determinado atributo es una referencia a un atributo llave primaria de otra entidad. Con esto se logra que se mantenga la integridad y la consistencia de los datos, por ejemplo:

CONSTRAINT "RefPolo120" FOREIGN KEY ("idpolo")

REFERENCES "public"."polo"("idpolo")

ON DELETE CASCADE

ON UPDATE CASCADE

En este segmento se expresa que el atributo id de polo es una referencia a la entidad Polo. Además se deben especificar cómo se va a proyectar las operaciones de borrado y actualización de ese atributo referenciado en las entidades que lo referencian, lo cual evita inconsistencia entre el atributo referencia y el referenciado.

Restricción de PRIMARY KEY (llave primaria)

Toda entidad debe tener una llave primaria la cual identifica una tupla unívocamente de las demás. Por ejemplo:

CONSTRAINT "PK44" PRIMARY KEY ("idhomework")

Al expresar que el atributo idhomework es PRIMARY KEY (llave primaria) significa que ese campo va a ser NOT NULL y además UNIQUE otra restricción que obliga a que ese campo sea único entre todas las tuplas.

Es importante decir que las restricciones se le aplican tanto a campos específicos como a tablas lo que significa que en vez de que la restricción afecta a un solo campo lo hace a un conjunto de ellos en la tabla. Como pueden ser:

Restricciones de llave primaria, de llave foránea, de atributos únicos. Lo cual es muy importante en el caso de que se tengan superclaves en el que se debe especificar un conjunto de atributos como llave primaria o claves candidatas compuestas por varios atributos los cuales se deben especificar en la misma restricción.

3.1.2 Normalización de la Base de datos.

La normalización de bases de datos es una técnica fundamental en el diseño de bases de datos relacionales. Mediante la normalización de bases de datos se transforman datos complejos en datos más sencillos y más fácil de mantener. Se minimiza la redundancia de la información evitando anomalías de inserción o modificación, con lo que se evita la inconsistencia de los datos. Con la minimización de la redundancia se evita usar espacio en disco innecesariamente.

La normalización incluye varias fases las cuales se deben realizar en orden, es decir, sólo se puede pasar a la próxima fase una vez concluida la anterior.

Formas Normales:

Primera Forma Normal.

Segunda Forma Normal

Tercera Forma Normal

Formal Normal de Boyce-Codd (FNBD)

Existen además la cuarta (4FN) y la quinta (5FN) formas normales.

Primera Forma Normal

Una relación está en 1FN si cumple la propiedad de que sus dominios no tienen elementos que, a su vez, sean conjuntos.

Los atributos de la relación deben ser atómicos e indivisibles, los cuales se deben definir en una relación aparte y relacionarla con la que le dio origen con esto se logra obtener datos sencillos fáciles de manejar se evita anomalías de actualización o eliminación y se evita tener datos redundantes (repetidos). Se refiere a anomalía de actualización o eliminación a la necesidad de tener que modificar cada tupla que tuviera el atributo no atómico, lo que ocurre de forma similar con la eliminación del mismo.

El presente diseño se encuentra en 1FN porque todos los atributos que presenta son atómicos. Los datos multivaluados a compuestos forman parte de otra tabla.

Segunda Forma Normal

Además de estar en 1FN, asegura que los atributos no llaves dependen completamente de la llave primaria, entonces si la tabla está en primera forma normal y además el atributo de la llave primaria no es compuesto está en 2FN, en caso de que existan superclaves, los atributos no llaves deben depender completamente de la llave primaria.

Tercera Forma Normal

Una tabla está en 3FN cuando además de estar en 2FN no presenta atributos que dependan transitivamente mediante atributos no llaves de la llave primaria.

Se puede decir que el presente diseño de base de datos se encuentra en 3FN porque todos sus atributos son atómicos, además todos los atributos no llaves dependen completamente de la llave primaria, además no existen dependencias transitivas en las tablas.

Se decide llevar el modelo relacional lógico hasta la tercera forma normal y no hasta una forma superior, porque la aplicación web a la que dará servicios la presente base de datos será de gran concurrencia por parte de los usuarios, sobre todo haciendo peticiones, fundamentalmente de consultas de selección, por lo que es conveniente tener datos redundantes en pos de una menor tiempo de respuesta a las peticiones de los usuario, que seguir normalizando para eliminar aún más la redundancia, pero

disminuyendo el rendimiento de la base de datos sobre todo en las peticiones de consultas de selección.

3.1.3 Análisis de redundancia de información.

Es una necesidad y un principio evitar la redundancia de la información las bases de datos lo cual significa la repetición de la misma información, a mayor redundancia mayor posibilidad de inconsistencias. Con la normalización de la base de datos se minimiza la redundancia de la información. Aunque en ocasiones por cuestiones de lograr una rápida de respuesta del sistema ante pedidos de información se necesita lograr una redundancia controlada con el fin de obtener un menor tiempo de respuesta en operaciones muy usadas por la aplicación tal es el caso de la sentencia SELECT.

3.1.4 Análisis de la seguridad de la base de datos.

La seguridad de la información se debe llevarse contra fallos físicos, lógicos, y humanos intencionados o no. Estos fallos alteran indebida y descontroladamente los datos por lo que la BD queda en un estado desconocido.

Para garantizar la seguridad y con ello la integridad de los datos se deben seguir un grupo de estrategias como son: se deben crear vistas y procedimientos almacenados los cuales van a ser llamados desde la aplicación evitando introducir directamente el código SQL en la consulta, además se deben hacer copias de resguardo o backups de la base de datos periódicamente para poderla restablecer ante una eventual daño de la misma.

3.2 Validación funcional de la base de datos.

Las pruebas se hacen con el propósito de validar el cumplimiento de los requerimientos, además de dar una indicación de la calidad del componente o el sistema. **(24)**

3.2.1 Llenado voluminoso e inteligente de la base de datos.

EMS Data Generator for PostgreSQL permite generar datos para una o varias tablas a la vez, además permite definir el rango de valores admisibles para cada campo. Por otra parte se garantiza mediante esta herramienta que se respete la integridad referencial, pues genera los datos llaves foráneas de las tablas referenciadas para evitar errores.

Se poblaron todas las tablas, pero se hizo énfasis en aquellas donde habitualmente se tiene un mayor acceso debido a que siempre en una base de datos hay consultas que

se van a realizar habitualmente por los usuarios, pero lo que puede afectar el rendimiento de la base de datos es la concurrencia de los usuarios ejecutándolas, en este caso es evidente que una cantidad considerable de los miembros de los polos, usuarios del sistema, van a ejecutar habitualmente consultas relacionadas con las siguientes casos de uso: Ver Tareas Asignadas, Ver Tiempo de Máquina, Ver Evaluaciones.

3.2.2 Herramientas para pruebas de carga intensiva (selección de querys más voluminosos y frecuentes):

Según el diseño del sistema propuesto y específicamente la base de datos se realizarán las pruebas de carga para analizar la respuesta del servidor.

El objetivo de la pruebas de las pruebas de carga es verificar el tiempo de respuesta del sistema para transacciones, bajo diferentes condiciones de carga. Las pruebas de carga miden la capacidad del sistema para continuar funcionando apropiadamente bajo diferentes condiciones de carga. La meta de las pruebas de carga es determinar y asegurar que el sistema funciona apropiadamente aún más allá de la carga de trabajo máxima esperada. (25)

El plan de prueba de carga en la BD estará compuesto por las consultas más utilizadas en la aplicación.

Beneficios de las pruebas de carga:

- Determina el rendimiento requerido para soportar anticipadamente los picos en producción.
- Determina si el hardware del ambiente es adecuado.
- Ayuda a determinar cuántos usuarios puede manejar la aplicación antes que la performance se vea afectada.

Variables del JMeter (27)

Average (Promedio): matemáticamente se refiere a al promedio del juego de datos. Es una medida que nos da una idea de cual es el comportamiento general de la aplicación.

Standard Deviation (Desviación Standard): Es la dispersión de los datos con respecto a la media. Una norma utilizada para esta métrica es: “Los datos con una desviación estándar superior al 68% de su media debe ser tratada como sospechosa”.

Throughput (Rendimiento): Es la tasa promedio de mensajes entregados satisfactoriamente.

3.2.3 Pruebas de Carga.

Cantidad de Registros en tablas Involucradas en la prueba de Carga:

Tabla 43: Cantidad de Registros en tablas Involucradas en la prueba de Carga

Tabla	Cantidad de Registros
user	1000
tabuserhomework	1800
homework	1000
evaluation	3000
timemachine	600

Prueba de Carga ejecutando la consulta 1 **: Ver Tareas Asignadas de un Usuario**

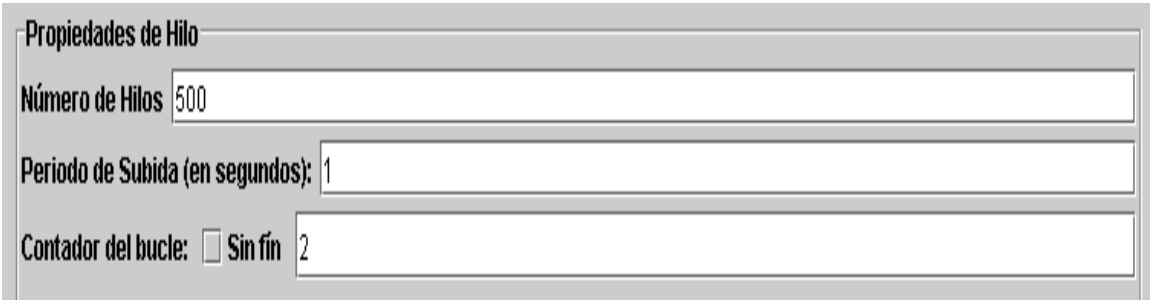

Figura 15: Configuración JMeter Consulta 1.

Label	# Muestras	Media	Mín	Máx	Std. Dev.	% Error	Rendimiento
Consulta Si...	1000	143	0	625	80.49	0.00%	390.2/sec
TOTAL	1000	143	0	625	80.49	0.00%	390.2/sec

Figura 16: Resultados Prueba Carga Consulta 1

Prueba de Carga ejecutando consulta 2 **: Ver Evaluaciones de un Usuario**

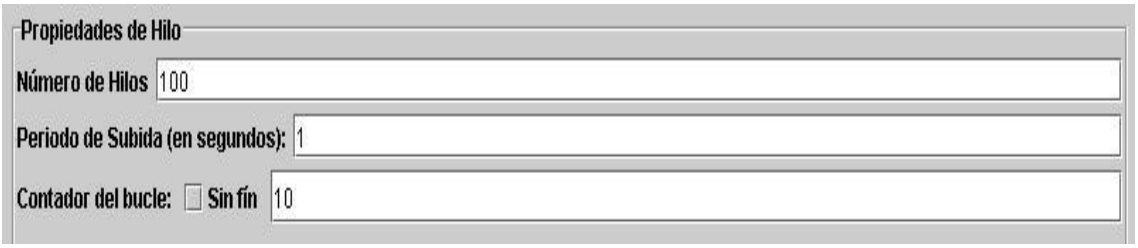

Figura 17: Configuración JMeter Consulta 2

Label	# Muestras	Media	Mín	Máx	Std. Dev.	% Error	Rendimiento
Ver Evaluaci...	1000	77	0	329	46.28	0.00%	516.0/sec
TOTAL	1000	77	0	329	46.28	0.00%	516.0/sec

Figura 18: Resultados Prueba Consulta 2

Prueba de Carga ejecutando la consulta 3**: Ver Tiempo de Máquina de un Usuario**

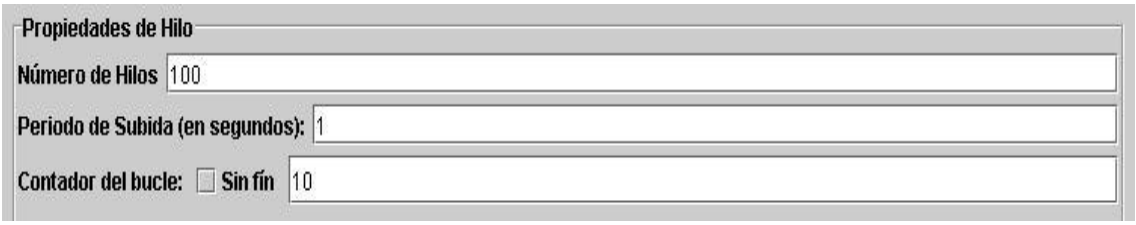

Figura 19: Configuración JMeter Consulta 3

Label	# Muestras	Media	Mín	Máx	Std. Dev.	% Error	Rendimiento
Consulta Si...	1000	83	0	422	55.68	0.00%	524.7/sec
TOTAL	1000	83	0	422	55.68	0.00%	524.7/sec

Figura 20: Resultados Prueba Consulta 3

Resultado General Prueba Carga:

Tabla 44: Resultado General Prueba Carga

Procedimiento Almacenado	Nº Conexiones	Intervalo	Bucle	Media	Desviación Estándar	Rendimiento
1	500	1	2	143	80.49	390.2/seg
2	100	1	10	77	46.28	516.0/seg
3	100	1	10	83	55.68	524.7/seg

Como se puede apreciar para un pico de usuarios, ejecutando cada consulta concurrentemente, superior a los que habitualmente se conectaran los resultados son correctos debido a que dieron un tiempo de respuesta promedio en el rango de 80 a 150 milisegundos, además en todos los casos la desviación estándar siempre fue menor al 68% de la media como establecen las métricas de calidad. Además el rendimiento siempre estuvo muy por encima del tiempo de respuesta de las conexiones lo que asegura que todos los pedidos van a atenderse correctamente sin necesidad de encolarse.

3.3 Conclusiones.

En el presente capítulo se describió la validación teórica y funcional realizada a la base de datos, se concluye que la base de datos queda normalizada a hasta la tercera forma normal con lo que se garantiza evitar redundancia en los datos y evitar anomalías de actualización. Además como parte de la validación funcional se pobló las tablas de la base de datos a lo que la misma respondió correctamente así como se le realizaron pruebas en busca de probar el rendimiento ante conexiones concurrentes picos en las cuales el servidor de bases de datos se comportó dentro de los límites establecidos.

Conclusiones Generales

En la presente investigación se le dio cumplimiento al objetivo general que fue diseñar una base de datos que permita almacenar y controlar la información de los RRHH de los polos productivos de la facultad 9, el cual se logró mediante el cumplimiento de las tareas propuestas que se dividieron en la realización del modelo lógico de la base de datos, posteriormente el diseño físico de la misma y por último la validación de la solución propuesta. Con el desarrollo de la presente base de datos se podrá tener la información centralizada y segura, de esta forma se logra minimizar la redundancia y se evita la inconsistencia en los datos. La presente base de datos desarrollada podrá brindar servicios a la aplicación web de gestión de los RRHH de los Polos Productivos de la Facultad 9 de la Universidad de las Ciencias Informáticas.

Recomendaciones

Después de haber desarrollado la base de datos concernientes a los RRHH de los polos productivos de la facultad 9 en la presente investigación se recomienda:

- El refinamiento del modelo lógico propuesto en próximas iteraciones del desarrollo del software.
- Actualización periódica de la base de datos, logrando que se mantenga la integridad de los datos.
- El empleo del gestor de bases de datos PostgreSQL para almacenar y gestionar la información en otros sistemas, como vía de lograr la soberanía tecnológica del país.

Bibliografía Citada

1. **Rodríguez, Juan.** *Workforce o la Planificación de RRHH*. *ConektIA Magazine.* [En línea] http://www.rrhhmagazine.com/conektia04/conektiaart2.asp

2. **León, Rolando A. Hernández y Coello González, Sayda.** *El paradigma cuantitativo de la investigación científica.* Ciudad de La Habana : Editorial Universitaria, 2002. 959-16-0343-6.

3. **Buades, Gaspar Rul-lán.** *Administración de recursos humanos.* Córdoba : Publicaciones ETEA. ISBN 84-86785-28-6.

4.**Márquez,Mercedes**. *Ficheros y bases de datos*. Universidad de Jaume,España. [En línea] http://www3.uji.es/~mmarques/f47/apun/node3.html.

5, 11. **García, Rosa M. Mato.** *Sistemas de Bases de Datos.* Ciudad de La Habana : Editorial Pueblo y Educación, 2005. ISBN 959-13-1273-3.

6,7,10,21,22. **Date, C. J.** *Introducción a los Sistemas de Bases de Datos.* Ciudad de La Habana : Félix Varela, 2003.

8. **Ruiz, Dr. Francisco.** *El modelo de datos jerárquico*. Universidad de Castilla La Mancha.

9. **Guzmán, Ignacio García Rodríguez de.** *Bases de datos modelo en red general.* Universidad de Castilla la Mancha .

12. **Asociación Peruana de Software Libre**. *Introducción a PostgreSQL*. [En línea] http://apesol.org.pe

13. **EmbarcaderoTechnologies.** *Ayuda del ER/Studio.* 2009.

14. **Jacobson, Ivan, Booch, Grady y Rumbaugh, James.** *El proceso unificado de desarrollo de software.* s.l. : Addison Wesley.

15. **Hansen, Gary W. y Hansen, James W.** *Diseño y Administración de Bases de datos.* 2da. s.l. : Prentice Hall. [En línea] http://bibliodoc.uci.cu/pdf/reg00071.pdf

16. **Méndez, Dr. Néstor D. Duque.***Seguridad e Integridad en Bases de Datos.* Universidad Nacional de Colombia. [En línea] http://www.virtual.unal.edu.co/cursos/sedes/manizales/4060029/lecciones/cap7-1.html.

17. **Andrés, María M. Marqués.** *Ficheros y Bases de Datos*.Universidad de Jaume. [En línea] http://www3.uji.es/~mmarques/f47/apun/node55.html.

18. **Microsoft Developer Network(MSDN).** [En línea] http://msdn.microsoft.com/es-es/library/ms184276.aspx.

19,20. **Consultoría en Marketing, Recursos Humanos y Servicios en Informática - Capacitación Laboral y Empresarial.** *BASES DE DATOS DE LA EMPRESA*. [En línea]

http://www.gestiopolis.com/administracion-estrategia/estrategia/base-de-datos-en-la-empresa.htm.

23. **Sicilia, Miguel A.** . *Control de concurrencia en bases de datos relacionales*. [En línea] http://cnx.org/content/m18939/latest/

24. **Aguilar, Ing. Violena Hernández.** *Conf. 5: Pruebas Ingeniería de Software 2*.Universidad de las Ciencias Informáticas, La Habana. [En línea] http://teleformacion.uci.cu/mod/resource/view.php?id=14103.

25. **Dirección de Calidad de Software.***Estrategia de Prueba del Laboratorio Industrial de Pruebas de Software de la UCI.*

26. **Leyva, Arnoldo Domínguez.** *MODELO LÓGICO Y FÍSICO DE LA BASE DE DATOS CORRESPONDIENTE A LOS MÓDULOS DE INVESTIGACIÓN CRIMINALÍSTICA Y ESTADÍSTICA DEL PROYECTO CICPC.* Trabajo de diploma, UCI.

27. **Sarco, José Pablo.** *Pruebas de Rendimiento.Conceptos,Documentos y Herramientas.*

Bibliografía Consultada

1. **González, Lic. Roberto Acosta.** *Sistema de Gestión para los Colaboradores de la Salud.* Ciudad de La Habana : s.n.

2. Sitio Web Oficial PostgreSQL. [En línea] http://www.postgresql.org/about/press/presskit82.html.es.

3. Sitio Web Oficial de Tecnologías Embarcadero. [En línea] http://www.embarcadero.com/products/erstudio/index.html.

4. **Gallego, Juan P. Gómez.** *Modelo objeto relacional-ORDMS.*

5. **Aliendre, Ramón, Paz, Juan y Juaniquina, Josefina.** *Comparación de rendimiento entre PostgreSQL, Oracle y SQLServer.*

6. **Castellanos, Ma. , González, Argenis y Pabón,Wilson Rojas.** *Comparación entre SGBD bajo licenciamiento libre y comercial.*TesisDoctoral,Universidad Católica de Colombia,Facultad de Ingeniería de Sistemas,2005.

7. **DanySoft.** Modelado de Bases de Datos. [En línea] http://www.danysoft.com/embarcadero.

9. **Marqués,Merche.** *Bases de datos orientadas a objetos*. Universidad de Castilla La Mancha.

10. ***González, Roberto A**..Sistema para la Gestión de los Colaboradores de la Salud.* Universidad de las Ciencias Informáticas ,La Habana, Cuba.

11. **Pastor, Patricio**. *Intranet: un sistema para la gestión de información.* Universidad de Chile. Taller en Tecnología de Redes Internet para América Latina y el Caribe, Rio de Janeiro, Brasil.

12.**Vásquez,Gabriel**. *Introducción a los sistemas de Bases de Datos.* Universidad del Cuaca,Colombia [En línea] http://atenea.unicauca.edu.co/~gavasquez/res/BD/material/IntroBD.pps.

13. **Quiroz, Javier.***El modelo relacional de bases de datos***.** Sistema Nacional de Inofmración Estadística y Geográfica. [En línea] http://www.inegi.gob.mx/inegi/contenidos/espanol/prensa/Contenidos/Articulos/tecnologia/relacional.pdf.

14. **Moreno Ortiz, Antonio**. *Diseño e Implementación de un lexicón computacional para lexicografía y traducción automática.* Universidad de Málaga.ISBN: 1139-8736.[En línea]

http://elies.rediris.es/elies9/4-2-3.htm .

15. Asignatura Diseño y Optimización de Bases de Datos. *Optimización de consultas.*Escuela Universitaria de Informática, Valencia. [En línea]

http://www.oei.eui.upm.es/Asignaturas/BD/DYOBD/OPCONS_texto.pdf .

16. **Tamargo, MSc. Lic. Lourdes Cerezal.** Diseño Físico del Data Warehouse. *BetSIME.* [En línea] http://www.betsime.disaic.cu/secciones/tec_ma_05.htm. ISBN 1029-5178 .

17. *Guía Didáctica: Modelamiento de Datos* .Universidad Técnica Particular de Loja, Ecuador. [En línea] http://www.utpl.edu.ec/eva/descargas/material/175/G181003.2.pdf .

18. **Chang Camacho, Ing. Violeta N. .** *Control de Concurrencia de Transacciones en un Sistema de Base de Datos*. Universidad Nacional de Trujillo, Perú. [En línea] http://www.informatizate.net/articulos/control_de_concurrencia_de_transacciones_en_un_sistema_de_base_de_datos_parte_01_21062004.html

19. **Dpto de Informática**. *Integridad: Control de concurrencia*. Universidad Carlos 3, Madrid, España. [En línea] http://ocw.uc3m.es/informatica/diseno-y-administracion-de-bases-de-datos/teoria/Tema4_6(Administracion_Concurrencia).pdf

20. *Entornos concurrentes*. Facultad de Informática,Universidad Nacional de La Plata,Argentina. [En línea] http://weblidi.info.unlp.edu.ar/catedras/ibd/clases/plan%202003/clase18.ppt.

21. *Administración de la base de datos*. Academia de Software Libre, Mérida, Venezuela.

[En línea] http://asl.fundacite-merida.gob.ve/courses/ED104/document/Manual_de_Base_de_Datos/capitulo_6.pdf?cidReq=ED104

22. **López, Oscar Pastor y Pons, Pedro Blesa.** *Gestión de bases de datos*. [En línea] http://books.google.com.cu/books?id=zYBWm5-X5usC&pg=PA141&lpg=PA141&dq=mecanismos+de+control+de+concurrencia%2Buniversidad&source=bl&ots=h9aOYfKvaw&sig=qt_ZGC1VaU2BbWVIRadDoVVbj8g&hl=es&ei=t4C5Sf-jKJqOMuTF8JMI&sa=X&oi=book_result&resnum=6&ct=result#PPP1,M.

23. **Galindo Gómez, J..** *Modelo Entidad Relación Extendido*. Universidad de Málaga [En línea]

http://www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r2227.PDF.

24. *Lenguaje SQL . Organización de archivos y bases de datos* .Licenciatura en Ciencias de la Computación. Universidad Nacional de San Luis, Argentina. [En línea] http://www.dirinfo.unsl.edu.ar/~bdtw/BD/teorias/apunte-sql-04.ps.