Universidad de las Ciencias Informáticas.

Facultad 8

Diseño e implementación de la Base de Datos del Sistema de planificación del Entrenamiento deportivo de Judo Femenino.

AUTOR:

Alexei Domínguez Rodríguez

TUTORES:

Arcel Labrada Batista

Yunesti Pérez la Rosa

Ciudad de la Habana, junio del 2009

"Año del 50 aniversario del triunfo de la Revolución"

Declaro que soy el autor de este trabajo titulado “Diseño e implementación de la Base de Datos del Sistema de planificación del Entrenamiento deportivo de Judo Femenino” y autorizo a la Facultad 8 de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

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

_____

Alexei Domínguez Rodríguez

Autor

_____ _____

Ing. Arcel Labrada Batista. Ing. Yunesti Pérez la Rosa

Tutor Tutor

"… no pases la vida tratando de hacer cosas extraordinarias, haz cosas ordinarias extraordinariamente bien."

Dedico este trabajo en especial a mi mamá que es mi sentido de la vida y la persona que me impulsa a que todos mis sueños se hagan realidad. A mi familia en general que siempre ha contribuido a que siga adelante. A todos mis amigos que han llenado mi corazón de alegría y han estado a mi lado en todo momento.

Primeramente

A mis padres por su amor y apoyo en todo lo que necesite, por creer en mí y a ellos le debo lo que soy.

A mi tía y a mi prima por su apoyo, dándome aliento a seguir adelante para cumplir este sueño de ser ingeniero.

A mi hermano por ayudarme todo el tiempo que estuve en la UCI.

A mi tutor Arcel por darme su apoyo en la realización de este trabajo.

A Jessy por ayudarme en los últimos momentos de mi carrera y darme su apoyo en todo lo que necesité.

A mis amigos Frank y Yailín por su apoyo durante el transcurso de la tesis.

A las jibias yami y yanisvel por hacerme estudiar tanto SQL cuando me hacían las preguntas con respuestas imposibles.

A mi grupo que pasamos por momentos difíciles en estos 5 cursos y nos apoyamos en todo para poder salir adelante, al igual a los que ya nos están en la escuela.

A Karenia mi jefa del proyecto que gracias a su insistencia con los trabajos que tenía que entregar, me mantenía desarrollando en la tesis.

A toda mi familia, amigos y los que de una forma u otra han contribuido a mi formación como profesional.

RESUMEN

La idea de desarrollar un sistema para realizar el plan de entrenamiento de judo surge a raíz de los disímiles problemas y dificultades que existe en el Instituto Nacional de Deporte, Educación Física y Recreación, al almacenar y gestionar la información que generan los entrenadores. Además de la necesidad de contar con un sistema informático para la realización de su trabajo de una forma mas ágil y eficiente.

En el desarrollo de este trabajo se presenta una investigación sobre los diferentes sistemas existentes en la búsqueda de un mayor grado de desarrollo hacia las diferentes transformaciones de la tecnología actual. Para lograr su desempeño es necesario realizar un estudio de los problemas existentes, la recopilación de datos según el plan establecido, serán analizados e interpretados para poder comprender el objetivo a ser alcanzados.

Más adelante se crea una estructura jerárquica de trabajo donde se describen los requisitos funcionales, se muestra el diseño y modelamiento de las diferentes clases que dan paso a las tablas a utilizar en la base de datos, se realiza un análisis de los procedimientos y consultas utilizado para facilitar el acceso a los datos. Al finalizar se analiza si se cumple con los resultados esperados en la realización de este trabajo.

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLA

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El Instituto Nacional de Deporte, Educación Física y Recreación (INDER) ha venido realizando una labor muy importante en el manejo y control de la información deportiva referente a todas las actividades y eventos que se realizan dentro y fuera del país, su trabajo se muestra en los grandes resultados obtenidos por los deportistas y entrenadores a través de la historia. Con el objetivo de lograr mejores resultados en las diferentes ramas deportivas es necesario la preparación y planificación de la información.

En la Escuela Cubana de Judo Femenino es muy importante la realización de los planes de entrenamiento para controlar el comportamiento físico de los deportistas, mediante estos planes se gestiona un gran volumen de información. La planificación del entrenamiento es una herramienta para el desarrollo del rendimiento deportivo, esta consiste en la realización de diferentes pruebas para determinar la condición física de cada individuo, y de acuerdo a los resultados obtenidos se elaboran una serie de actividades, a través de las cuales se garantiza el aprovechamiento máximo de su potencial físico.

Durante la investigación realizada se han detectado varias deficiencias en la elaboración del proceso de planificación del entrenamiento, debido a que la cantidad de información es muy grande y se encuentra muy dispersa, almacenada en formato duro y videos, los cuales se van deteriorando a través del tiempo y mucha información valiosa se pierde. Los entrenadores solo poseen una aplicación sencilla la cual brinda muy pocos servicios y tiene problemas de ejecución, por tanto, no pueden consultar de forma rápida los planes de entrenamientos anteriores. El proceso de búsqueda de los datos necesarios para realizar los cálculos de estadística y rendimiento es muy lento porque hay que acceder a una gran cantidad de tablas generadas por los entrenadores.

Debido a las diferentes dificultades que se han planteado anteriormente surge la necesidad de realizar este trabajo para darle solución a los problemas existentes en el deporte judo femenino, por lo que el **problema** a resolver es: ¿Cómo centralizar y agilizar el proceso de gestión de información deportiva en la escuela cubana de judo femenino?

Por lo que **el objeto de estudio** es la gestión y el procesamiento de la información deportiva generada por los entrenadores de judo femenino.

El **campo de acción** que comprende este trabajo es la gestión de la información deportiva para la planificación del plan de entrenamiento de judo femenino en Cuba.

La **idea a defender** que se plantea es: Si se desarrolla un sistema de base de datos para la escuela cubana de judo femenino, se piensa que es posible centralizar y agilizar los procesos para la gestión de la información que deben utilizar los entrenadores para planificar el entrenamiento deportivo; brindándoles la posibilidad de acceder a toda la información en el momento que sea necesario.

El **objetivo general** de este trabajo es diseñar e implementar una base de datos para centralizar y gestionar la información generada por los entrenadores para la planificación del entrenamiento deportivo de judo femenino.

Los **objetivos específicos** que se trazan en este trabajo son los siguientes:

1. Elaborar el marco teórico y conceptual de la investigación científica.
2. Realizar un estudio de los procesos relacionados con el plan de entrenamiento de judo femenino.
3. Diseñar e implementar la base de datos que permita manejar la información de una forma más rápida e interactiva, satisfaciendo las necesidades del cliente.

Para dar cumplimiento a los objetivos antes planteados de una forma eficiente es necesario desarrollar las siguientes **tareas**:

1. Estudio del estado del arte.
2. Investigación sobre las herramientas más usadas para el diseño e implementación de las base de datos.
3. Análisis de los requisitos a tener en cuenta en el desarrollo de la aplicación para que cumpla con los objetivos esperados.
4. Clasificación de las clases persistentes en el modelo de diseño para obtener un modelo claro de la base de datos.
5. Realización de un diagrama de clases persistentes.
6. Realizar un modelado relacional donde se expongan todas las clases necesarias para desarrollar el proyecto.
7. Normalización de la base de datos.
8. Descripción de las tablas de la Base de Datos.

Métodos de Investigación científica

Para la investigación realizada durante el desarrollo de este trabajo se emplearon los métodos histórico-lógicos que brinda la posibilidad de estudiar las características del objeto de investigación que no son observables directamente. Brinda la posibilidad de conocer cómo realizaban los diferentes planes de entrenamientos de judo femenino hasta la actualidad. También se utilizaron los métodos empíricos de observación y entrevista, que facilita el estudio más de cerca al objeto de investigación, sus características y resultados, así como la recopilación de la información con la que cuentan los entrenadores.

Capítulo 1. Fundamentación Teórica

Introducción

En este capítulo se expone el marco teórico y conceptual, así como las tendencias actuales similares y las tecnologías más utilizadas hoy en día, sistemas de gestión de base de datos vinculados al software a desarrollar y herramientas necesarias para su realización.

1.1 Sistemas similares de gestión de planes de entrenamientos

Las nuevas tecnologías marcan una pauta en el desarrollo de las diferentes categorías deportivas, haciendo más ágil y sencillo el trabajo a los entrenadores y especialistas. Los sistemas de gestión de planes de entrenamientos son utilizados en todos los deportes y tienen una nueva perspectiva hacia las tecnologías actuales. Se han desarrollado varios software para dar solución a una serie de dificultades que presentan los entrenadores en la elaboración de los planes de entrenamiento, como son los cálculos engorrosos y el almacenamiento de gran cantidad de información. En el estudio realizado se muestra una serie de sistemas cuyo objetivo fundamental es gestionar los datos de los planes de entrenamiento.

1.1.1 Sistemas similares a nivel internacional

ELATLETA: es la web oficial del club deportivo Paris.

En el sitio de planificación de planes de entrenamiento ELATLETA, se puede encontrar todo lo relacionado con el deporte de atletismo. El sistema gestiona mucha información dividida por temáticas entre las cuales se encuentran los planes de entrenamiento de atletismo, dividido por niveles, inicio, medio y avanzado, al igual que una serie de ejercicios por cada uno de los niveles. Cuenta con una sección administrativa donde los entrenadores se autentican y acceden al sistema, pueden realizar cambios, insertar y eliminar información referente a un atleta o planes de entrenamientos en específicos. Cuenta con una base de datos bien estructurada en la que guardan toda la información por cada uno de los atletas y también de forma general por los diferentes niveles de entrenamiento [1]. En este sistema se presentan características similares a las que necesita el software a desarrollar, pero los planes de entrenamiento se encuentran estructurados por mesosistemas, microsistemas donde cada entrenador maneja las estadísticas de los deportistas durante todo el período de preparación y competitivo, esto trae consigo que se maneje una gran

cantidad de tablas con una estructura completamente diferente a la del sistema ELATLETA por lo que no es una solución factible.

X-MEDALIST

Software para planificar y controlar el entrenamiento para deportes individuales y para la salud. Incorpora herramientas nuevas y novedosas para el mercado del software aplicado al entrenamiento, que permite a los especialistas trabajar de forma rápida y coordinada, organizando toda la actividad laboral de cada deportista. Está diseñado para realizar el trabajo rápido, seguro y cubriendo todas las actividades en la labor diaria de cada atleta. En la ficha del atleta se podrá ingresar a todos sus alumnos y deportistas. Además, entre otras cosas, lleva el registro y control de lesiones, como también la historia clínica y deportiva, el análisis de la técnica individual y el biorritmo personal. Éste sistema es el que más se asemeja al que se desea desarrollar ya que en él se realiza un análisis del atleta, donde se recoge la información correspondiente, su comportamiento, así como las pruebas médicas correspondiente al final de cada entrenamiento, pero solo es para realizar planes de entrenamientos en deportes individuales (Atletismo, Natación, Ciclismo, Patín, entrenamiento de fuerza, remo) y la salud [2]. El sistema a desarrollar abarca todos los ejercicios físicos que se realizan además de la fuerza y las pruebas médicas por lo que X-MEDALIST no logra cumplir con todos los requisitos de los planes de entrenamientos de judo.

1.1.2 Sistemas similares a nivel nacional

La automatización de la planificación del entrenamiento deportivo está presente en los diferentes deportes, por ejemplo el sistema para la planificación del atletismo, las pesas como deporte y como deporte auxiliar, la esgrima.

Software de planificación del entrenamiento en Atletismo: en su elaboración se tuvo en cuenta los elementos teóricos de la planificación en este deporte, los cuales permitieron definir los requisitos a cumplir por el sistema para lograr las expectativas de los especialistas, donde se definieron todos los planes aplicados al deporte de atletismo, así como su estructura y reglas. Para almacenar los datos necesarios se utilizó como base de datos Access, creando ficheros para ser consultados posteriormente. En el mismo se crea una interfaz gráfica amigable con la cual pueden interactuar los especialistas y consultar los datos necesarios así como realizar operaciones de forma automática. Estos procesos se desarrollan también en los planes de entrenamiento de judo, lo que se puede tener en cuenta para el desarrollo del sistema. Es una aplicación de escritorio muy sencilla donde solo se controlan

algunos datos de forma general y el gestor de base de datos no requiere de mucha potencia y no es multiplataforma, esto trae consigo que no sea recomendable su utilización en los planes de entrenamiento de judo.

Software de planificación del entrenamiento en el deporte de pesas: en él se definen las reglas y los elementos teóricos, así como los pasos para realizar la entrada de la información referente a estos planes de forma manual. El software confeccionado resuelve las dificultades analizadas en software ya existentes, además tiene la ventaja de ser confeccionado para el equipamiento más moderno disponible en el país. Todos los datos son almacenados en Access donde se realizan diversos cálculos estadísticos y consultas por diferentes categorías. Los datos están bien estructurados y organizados por tablas. Cuenta con una interfaz diseñada en Access muy amigable que brinda la posibilidad de entrada de datos por selectores, que es lo que se quiere alcanzar en los planes de entrenamiento: lograr un ahorro de tiempo y trabajo. Es una aplicación de escritorio solo utilizada en Windows lo que restringe su utilización; el gestor de base de datos de este software no es recomendable utilizarlo cuando se gestiona gran cantidad de información, que es una de las características más importantes que tienen estos planes de entrenamiento así como la tendencia actual hacia el software libre.

Software para la planificación del entrenamiento en las pesas como deporte auxiliar: se desarrolla en Access con una interfaz sencilla y fácil de usar poniendo al alcance de los especialistas la posibilidad de no tener que realizar los cálculos referente al deporte, que son muy engorrosos, y de esta forma ganan más tiempo en la atención de las diferencias individuales de cada atleta. Los datos están bien estructurados en tablas, lo que facilita el acceso a los datos de una forma más organizada. Al ser una aplicación de escritorio y solo disponible para Windows contando como gestor de base de datos a Access, no es factible su utilización en el proceso de recopilación de los datos referente a cada deportista en su entrenamiento, debido a que brinda limitaciones de procesamiento de búsquedas, no es multiplataforma y presenta bloqueos a nivel de tablas en la base de datos.

Software para la planificación del plan de entrenamiento de Esgrima: en él se da la posibilidad al usuario de añadir todas las componentes de la preparación y capacidades con las que se desee entrenar. El algoritmo fue programado en Visual Basic como un evento al hacer clic en el botón, por no obtener la salida deseada fue necesario realizar una consulta cruzada. Cuenta con consultas a la base de datos que posibilitan mostrar la información en diferentes criterios y organizada por meso ciclos

muy similar al software a desarrollar. La información está dividida en tablas que se encuentran estrechamente relacionadas. Tiene como desventaja que utiliza Access como gestor de base de datos ya que es no es multiplataforma y solo está disponible para sistemas operativos de Microsoft.

1.2 Sistemas Gestores de base de datos

Se afirma que las base de datos, son uno de los recursos más importantes para el desarrollo de cualquier área deportiva, a través de la historia se ha demostrado que son de gran influencia en el progreso de planificación de los diferentes deportes a nivel mundial, brindando un almacenamiento y acceso confiable, eficiente y práctico en el uso de la información que se produce. Las base de datos no solo han servido como repositorios de información, es una de las herramientas más importantes en la gestión de información. A nivel mundial existen muchos sistemas de gestión de base de datos que son usados por profesionales de la información, para diseñar trabajos colaborativos e interdisciplinarios. Están disponibles para el uso y desarrollo de aquellos interesados en el área de la gestión de la Información.

Como se ha visto, existe gran cantidad de base de datos especializadas en el área de planes de entrenamientos, en muchos casos se utilizan sistemas pocos complejos y fácil de usar, que traen consigo muchas deficiencias. A continuación se muestra un estudio de los diferentes gestores de base de datos más usados en internet en esta tarea de planificación y gestión de información referente al deporte, donde se realiza una descripción detallada de las ventajas y desventajas de cada uno.

1.2.1 MySQL

Es un sistema de gestión de base de datos relacional, cuyo principal objetivo de creación fue la velocidad. Licenciado bajo la GNU GPL con un esquema de licenciamiento dual. Cuenta con una licencia para uso en software privativos, de esta forma las empresas que deseen adquirirla para el desarrollo de software privados es necesario comprarla con este fin. El MySQL es propietario, el cual está sustentado por una empresa MySQL AB privada que posee la mayor parte del copyright del código. Su diseño multihilo y multiusuario le permite una gran carga de trabajo, procesando gran cantidad de información.

Es la base de datos *Open Source* más popular de Internet [3]. Es usada actualmente por muchas empresas debido a su fiabilidad y facilidad de uso por lo que persigue

cubrir todas las necesidades del cliente, principalmente la velocidad. Funciona sobre varias plataformas, permitiendo su uso en disímiles lenguajes de programación al acceder a las base de datos, esta cuenta con una gran cantidad de librerías y herramientas que alcanzan una mayor popularidad entre los sistemas de gestión de base de datos. Es desarrollado cumpliendo con el estándar SQL logrando así una gran fiabilidad por los desarrolladores.

Este sistema tiene muchas ventajas como las que se muestran a continuación.

- Consume pocos recursos por lo que puede ser ejecutado en máquinas de bajo rendimiento.
- Para lograr mejor velocidad eliminaron los subselect y las transacciones.
- Ofrece mejor rendimiento al ser usado en aplicaciones web, en Drupal y con su conexión programada en el lenguaje php.
- Los sistemas se pueden conectar usando socket TCP/IP en cualquier plataforma.
- Posee múltiples motores de almacenamiento que permiten escoger la más adecuada para cada tabla.
- Logra mayor cantidad de transacciones por segundo debido a un sistema de agrupación de transacciones.
- El uso de índices y optimización de consultas agilizan el trabajo en base de datos muy grandes ya que esto trae consigo que se gestione menos cantidad de información.
- Soporta una gran cantidad de tipos de datos diferentes hasta 32 índices por tablas, puede ser utilizada en el desarrollo de base de datos que gestionen gran volumen de información.

Al igual que otros sistemas tiene desventajas:

- Puede provocar problemas de integridad y la información se puede dañar en entornos donde se gestione mucha información simultáneamente, altas concurrencias en las modificaciones de datos.
- Es privativo lo que trae consigo que cualquier distribución de un producto debe de ser bajo la licencia de GNU PGL o la licencia comercial de MySQL.
- Su escalabilidad no es muy buena y por eso no es recomendable usarlo en base de datos muy grandes.
- Carece de transacciones rollback`s y subconsultas.

1.2.2 Oracle

Es uno de los manejadores de base de datos más potente que existe actualmente. Brinda las funcionalidades básicas de un gestor sin olvidar las opciones de subconsultas, no es lento al ser ejecutadas.

Es muy vendido a nivel mundial, y su alto precio es debido a que Oracle es un manejador de base de datos muy poderoso y solo está al alcance de medianas o grandes empresas como las multinacionales. Consume gran cantidad de recursos en el servidor que trae consigo que se necesite de un buen sistema de hardware, RAM, microprocesador, disco duro. Es capaz de almacenar gran cantidad de información, existen servidores que gestionan hasta muchos teras de capacidad. En el desarrollo de aplicaciones web no está tan extendido debido que es un sistema muy caro a diferencias de otros sistemas de gestión de base de datos como MySQL, SQL Server.

Oracle se basa en tecnología cliente-servidor, cuenta con herramientas necesarias para su utilización en el servidor así como con herramientas de programación básicas de Oracle. Para su desarrollo se utiliza el lenguaje PL/SQL y SQL para crear formularios.

Ventajas de Oracle

- Permite crear los formularios de forma similar a otras herramientas de programación conocidas como Visual Basic, Visual C.
- Puede ejecutarse en varias plataformas, desde una PC hasta un servidor que cuente con diferentes sistemas operativos.
- Usa particiones para mejorar la eficiencia de replicación y algunas versiones admiten administración de base de datos distribuidas.
- Es un sistema muy complejo que permite desarrollar diseños con triggers y procedimientos almacenados con una integridad referencial bastante potente.
- Capaz de almacenar una amplia variedad de información y recuperarla de forma rápida y fácil. Permite a las organizaciones conocer más de su negocio.
- Soporte de transacciones.
- Estabilidad.
- Escalabilidad.

Como otros sistemas de gestión de base de datos cuenta con ciertas desventajas, y esto trae consigo que no sea factible su utilización en algún software.

- Es muy riesgoso el proceso de controlador de versiones ya que es muy engorroso y puede perderse o dañarse la información.
- Las últimas versiones presentan algunos fallos en el sistema de almacenamiento por causa de las extensiones orientadas a objetos.
- Una de sus mayores desventajas radica en su precio, incluso las licencias son excesivamente caras.
- Otro problema es la necesidad de ajustar el sistema a las necesidades del cliente, no basta con instalar el Oracle en un servidor y conectar las aplicaciones al cliente, ya que si es mal configurado puede resultar extremadamente lento.

1.2.3 Microsoft SQL Server

Es un sistema de gestión de base de datos relacional que utiliza lenguaje SQL, a través del cual se puede acceder a grandes cantidades de datos de manera simultánea. Constituye la alternativa de Microsoft, es un sistema muy utilizado en aplicaciones de escritorio y otros sistemas de gestión, preferentemente con productos de Microsoft.

Tiene varias ventajas de trabajo como por ejemplo:

- Es un sistema barato.
- Es el primero en soportar la interfaz de acceso OLE DB y ADO.
- Cuando el SQL Server no tiene tareas del usuario que procesar comienza la llamada tarea de limpieza lo que hace mejorar el funcionamiento del CPU, necesita menor limpieza en las memorias intermedias durante el procesamiento de transacciones.
- Registra las transacciones de tal forma que las actualizaciones se pueden reducir al último estado en caso del servidor fallar.
- Soporta transacciones.
- Soporta procedimientos almacenados, triggers.
- Cuenta con una gran estabilidad y seguridad.
- Escalabilidad.
- Tiene un entorno gráfico administrativo muy potente.

Entre sus desventajas se encuentra:

- Presenta bloqueos a nivel de página.
- Dispositivos con crecimiento manual.
- Su licencia es propietario.
- Soportado por el sistema operativo Windows.

1.2.4 PostgreSQL

Es un servidor de base de datos relacional de código abierto orientado a objetos, publicado bajo la licencia BSD y creado en software libre por una comunidad de desarrolladores y organizaciones comerciales. El PostgreSQL permite que mientras unos usuarios accedan a una tabla otros pueden trabajar en ella sin necesidad de bloqueos. Cuenta con una amplia variedad de tipos y un entorno administrativo amigable fácil de usar. Es un sistema objeto-relacional ya que incluye la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. Es el sistema libre más avanzado soportando la gran mayoría de las transacciones SQL. Trabaja con varios lenguajes de programación como son C, C++, Java, Python, PHP.

Entre sus ventajas se puede encontrar:

- Soporta múltiples tipos de datos como los tipos de fecha, monetarios, elementos gráficos, datos de redes y también creación de tipos de datos propios.
- Permite la creación de funciones personalizadas y el manejo y configuración de disparadores.
- Posee una gran escalabilidad.
- Es capaz de ajustarse al número de CPU y la cantidad de memoria que ofrece el sistema de forma óptima, siendo capaz de soportar mayor cantidad de peticiones simultáneas sin provocar errores.
- Hace uso de rollback`s.
- Subconsultas y transacciones.
- Comprueba la integridad referencial y almacena procedimientos en su propia base de datos.

Entre sus desventajas está:

- La velocidad de respuestas que ofrece este gestor es un poco deficiente.

- Tienen un límite de 8k por filas que se pueden aumentar hasta 32 lo que trae consigo que disminuya su rendimiento.
- Es más lento que los otros gestores de base de datos. [4]
- Consume gran cantidad de recursos cuando se realizan muchas transacciones de datos.

El PostgreSQL es un magnífico gestor de base de datos que cuenta con casi todas las características de los gestores comerciales, haciendo de él una buena alternativa GPL.

Comparación entre los sistemas de gestión de base de datos (SGBD).

Tabla 1: Comparación SGBD.

Criterios de Comparación	My SQL	Oracle	SQLServer	PostgreSQL
Sistemas Operativos	Linux/Windows	Windows, Unix , Macintosh y Mainframes	Windows	Linux/Windows
Consumo de Recursos	Consume pocos recursos	Necesita de gran cantidad de recursos	Consume pocos recursos	Consume pocos recursos
Velocidad	Rápido	Rápido	Rápido	Lento
Costo	Muy costoso	Muy costoso	Muy costoso	Gratis
Rendimiento	Alto	Alto	Alto	Alto
Comercialización	Privativo	Privativo	Privativo	Libre
Volumen de datos	Alto	Alto	Alto	Alto
Integridad	Baja	Alta	Alta	Alta

Después de realizar un análisis de las diferentes características de los sistemas de gestión de base de datos relacional en la tabla anterior se decidió escoger a PostgreSQL como SGBD porque es más ventajoso, ya que es multiplataforma a diferencia de SQLServer, que es privativo y hay que pagar por una licencia al igual que Oracle (muy costoso) y MySQL. El Oracle consume gran cantidad de recursos por lo que se necesita de una PC o servidor con buenas características de hardware. En cuanto al MySQL cuando almacena o gestiona gran cantidad de datos pueden existir pérdidas o daños en la información, su integridad es baja. El PostgreSQL es escogido porque aventaja los otros SGBD en varios aspectos aunque es más lento al ejecutar los procesos, pero se adapta al sistema que se necesita para el desarrollo de la aplicación.

1.3 Herramienta Case

1.3.1 Embarcadero Erwin Studio

Es una herramienta para la arquitectura y modelamiento de base de datos. Provee de mejores reportes y mejor soporte para la implementación de prácticas y estándares en modelos complejos que contienen metadatos críticos. Mejora la calidad del modelo de datos y reduce los riesgos asociados a la seguridad de la información.

Es una herramienta multinivel destinada para la construcción del modelo físico y lógico. Está equipado para crear y manejar diseños de base de datos grandes y complejos. Cuenta con grandes capacidades de diseño lógico, construcción automática de base de datos, permite realizar ingeniería inversa partiendo del modelo físico al modelo lógico. Brinda mayor soporte en la integración de almacenamiento de datos diseñados para visualizar, documentar y compartir el conocimiento. Soporta como sistemas de gestión de base de datos a Oracle, Sybase System, Microsoft SQL Server, IBM D B/2 Universal, InterBase 4, Microsoft Access y Microsoft Visual Fox Pro [5].

1.3.2 Case Studio 2

Es un programa muy usado en la modelación de base de datos, permite crear y mantener diagramas de entidad relación (DER) y diagramas de flujo de datos (DFD), en él se crean scripts SQL de forma automática que funcionan sobre diferentes sistemas de base de datos, además se pueden utilizar plantillas para sistemas con características similares. Presenta dos variantes, una gratuita que es bastante

completa, tan sólo presenta limitaciones de pago, y otra versión completa que ofrece grandes y completas funcionalidades. Trata aspectos como contenidos y dominios para dar mejor seguridad al sistema. Contiene catálogos para guardar el modelo de diseño. Brinda soporte para ingeniería inversa, que permite identificar y estructurar base de datos ya existentes para poder trabajar con ellas. Facilita el trabajo de documentación al generar informes detallados en HTML y RTF, además de la documentación de diagramas de Flujo de datos exportables en formato XML. Exporta para varios gestores de base de datos como son, Oracle, MySQL, PostgreSQL, Access, MS SQL, Max DB, Firebird. Tiene una interfaz gráfica muy intuitiva y fácil de usar. Soportado en diferentes sistemas operativos, Windows 95/98/Me/NT/2000 /XP. Al crear el diagrama entidad relación (ERD), considera cada base de datos de opciones tales como integridad referencial, restricciones, dominios y disparadores.

De las herramientas de modelamiento de base de datos mencionadas, para el desarrollo de este trabajo investigativo se escogió el Case Studio 2 por ser una herramienta que soporta un gran número de SGBD entre ellos el PostgreSQL a diferencia del Embarcadero Erwin Estudio. Es una herramienta libre y permite generar scripts SQL para ejecutarlo en el PostgreSQL y crear la base de datos de forma automática así como generar reportes tanto del modelo físico como lógico.

1.4 Proceso y metodologías de desarrollo

El proceso de desarrollo de software día a día se va tornando más engorroso, debido a que las exigencias de los clientes crecen rápidamente, y sus deseos y necesidades se vuelcan en un reto a cumplir por los desarrolladores. Por lo que para lograr el desarrollo de un software que cumpla con los requisitos planteados por el cliente, es necesario llevar a cabo un proceso capaz de organizar y dirigir la evolución del proyecto.

En el proceso de construcción de un software es necesaria una metodología capas de organizar y dirigir la línea de vida del mismo. Por sus características que son expuestas a continuación se ha escogido el Proceso Unificado de Desarrollo (RUP).

1.4.1 Proceso Unificado de desarrollo (RUP)

Es uno de los procesos más utilizados de los existentes actualmente, ya que está pensado para adaptarse a cualquier proyecto. Fusiona completamente a un equipo de desarrollo de software, reduce la productividad de cada uno de los miembros del equipo en la asignación de tareas y facilita la interacción entre los líderes de proyectos

y demás empleados brindándoles la experiencia y las lecciones aprendidas en el desarrollo de los disímiles proyectos realizados. Este proceso está orientado a modelos de desarrollo basados en componentes, que utiliza el Lenguaje Unificado de Modelado (UML) el que define técnicas de análisis y diseño que ayudan a preparar los esquemas de un sistema de software. Entre sus principales características se encuentra que define claramente quién, cómo, cuándo y qué debe hacerse en un proyecto, está dirigido por casos de usos, que orientan al proyecto a la importancia para el usuario y lo que este quiere, además centrado en la arquitectura que relaciona la toma de decisiones que indica cómo tiene que ser construido el sistema y en que orden, también es iterativo e incremental donde se divide en micro proyectos y cumplen con el objetivo de forma más detallada, mejorando su comprensión y desarrollo. Un proyecto realizado siguiendo RUP se divide en cuatro fases:

- Inicio.
- Elaboración.
- Construcción.
- Transición.

En cada fase se ejecutarán una o varias iteraciones (de tamaño variable según el proyecto), y dentro de cada una de ellas seguirá un modelo de cascada para los flujos de trabajo.

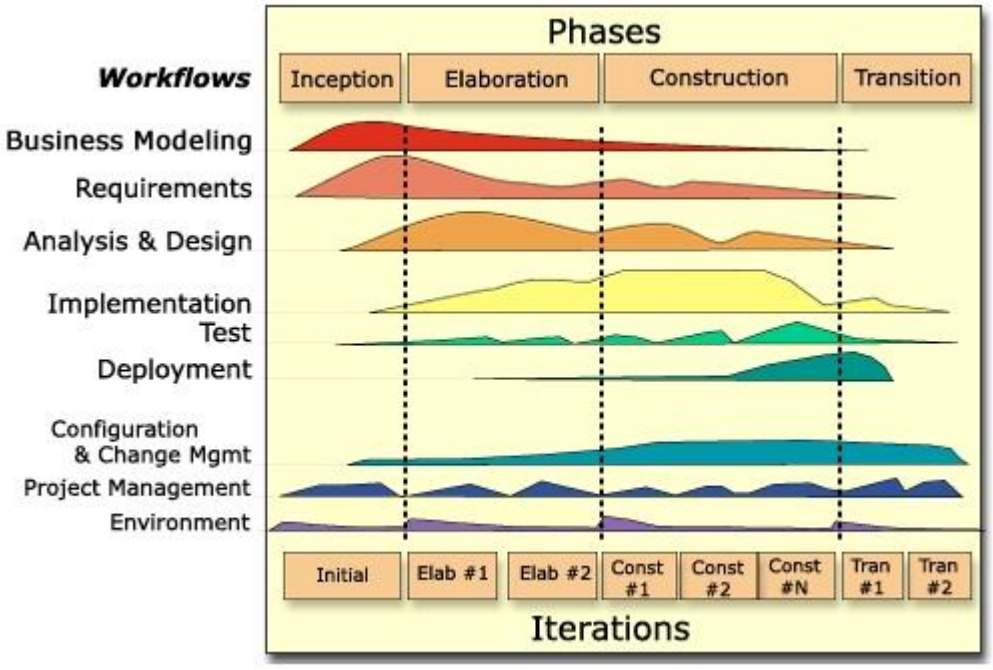

Figura 1: Fases y Flujos de trabajo de RUP.

Es una metodología que puede ser aplicada a diferentes herramientas de modelado como son: Rational Rouse Enterprise Edition 2003 y Visual Paradigm, que son capaces de manejar todos los procesos necesarios para el desarrollo del proyecto.

1.4.2 Lenguaje Unificado de Modelado (UML)

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML incluye conceptos semánticos, notación, y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. Está pensado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código así como generadores de informes. La especificación de UML no define un proceso estándar pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos.

UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Un sistema se modela como una colección de objetos discretos que interactúan para realizar un trabajo que finalmente beneficia a un usuario externo. La estructura estática define los tipos de objetos importantes para un sistema y para su implementación, así como las relaciones entre ellos. El comportamiento dinámico define la historia de los objetos en el tiempo y la comunicación entre objetos para cumplir sus objetivos. El modelar un sistema desde varios puntos de vista, separados pero relacionados, permite entenderlo para diferentes propósitos.

1.4.3 Rational Rose

Rational Rose es la herramienta CASE que es comercializada por los creadores de UML (Booch, Rumbaugh y Jacobson). En la evolución del proyecto permite la concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases y entregables. Facilita el desarrollo de un proceso cooperativo en el que todos los agentes tienen sus propias vistas de información, pero utilizan un lenguaje común para comprender y comunicar la

estructura y la funcionalidad del sistema en construcción. Esta herramienta propone la utilización de cuatro tipos de modelos para realizar el diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software. Rational Rose utiliza un proceso de desarrollo iterativo controlado, donde el desarrollo se lleva a cabo en una secuencia de iteraciones. Cada iteración comienza con una primera aproximación del análisis, diseño e implementación para identificar los riesgos del diseño, los cuales se utilizan para conducir la iteración, primero se identifican los riesgos y después se prueba la aplicación para que estos se hagan mínimos. Se puede generar código en distintos lenguajes de programación a partir de un diseño en UML. Proporciona llegar al modelo lógico desde el modelo físico, que es en si la ingeniería inversa.

1.4.4 Visual Paradigm

Es una herramienta CASE de modelado que utiliza UML como lenguaje de modelado profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue.

Permite realizar ingeniería tanto directa como inversa. La herramienta es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto; genera la documentación del proyecto automáticamente en varios formatos como Web o .Pdf, y permite control de versiones. Cabe destacar igualmente su robustez, usabilidad y portabilidad. En definitiva, VISUAL PARADIGM es una herramienta muy a tener en cuenta a la hora de ponerse manos a la obra con un proyecto importante.

Para el desarrollo de este trabajo se escogió como herramienta CASE de modelado al Visual Paradigm porque tiene un coste favorable y realiza de una forma íntegra los planes de construcción del software. Permite ingeniería inversa, del modelo físico se puede llegar al modelo lógico. Genera de forma automática códigos desde diagramas, además brinda la posibilidad de generar documentación.

1.5 Herramienta manejadora de base de datos

1.5.1 PgAdmin |||

Está diseñado para responder las necesidades de todos los usuarios, desde simples consultas SQL hasta la elaboración de funciones en base de datos complejas. La interfaz gráfica de PostgreSQL soporta todas las características y hace fácil la

administración. Es desarrollado por una comunidad de PostgreSQL expertos de todo el mundo y está disponible en más de una docena de idiomas. La más avanzada base de datos de Open Source en el mundo. La solicitud podrá ser utilizado en Linux, FreeBSD, OpenSUSE, Solaris, Mac OSX y las plataformas de Windows para la gestión de PostgreSQL 7.3 y por encima de funcionamiento en cualquier plataforma, así como comerciales y las versiones derivadas de PostgreSQL como EnterpriseDB, Mammoth PostgreSQL, Bizgres y Greenplum base de datos [6]

Características:

- Cuenta con una sintaxis SQL editor, servidor en el editor de código.
- Sentencia de SQL/ lote / Shell para programación de agente de empleo (tareas programadas).
- La conexión es utilizando TCP/IP o Unix Domain Sockets y no requiere de drives para comunicarse con el servidor.
- Es software libre.
- Tiene acceso a todos los objetos de PostgreSQL.
- Muy rápido para visualización y entrada de datos.
- Procedimientos almacenados.
- Transacciones.

1.5.2 EMS SQL Manager para PostgreSQL

EMS SQL Manager para PostgreSQL es una poderosa herramienta gráfica para procesos de administración y desarrollo de PostgreSQL. PostgreSQL Manager funciona con cualquier versión de PostgreSQL, hasta la 8.1, y soporta todas las nuevas características de PostgreSQL. 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 para facilitar el trabajo de los novatos.

Características:

- Soporte completo de PostgreSQL hasta la versión 8.1.
- Nueva interfaz gráfica de usuario último modelo.
- 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.
- Conexión vía reenvío de puerto local a través del túnel SSH.[7]

1.5.3 EMS SQL Query para PostgreSQL

SQL Query para PostgreSQL es un utilitario que te permite construir, de manera rápida y simple, consultas SQL para base de datos PostgreSQL. Tiene disponible tanto la construcción visual como la edición directa de una consulta formato texto. Una amigable interfaz gráfica de usuario: que permite conectar a base de datos, seleccionar tablas y campos para una consulta, configurar el criterio de selección de registros y agregar las condiciones de forma visual. Puedes trabajar con varias consultas de base de datos a la vez, editarlas y realizar otras operaciones con las mismas.

Características:

- Trabaja con varias consultas en diferentes ventanas.
- Permite la conexión al mismo tiempo a varias base de datos.
- Tiene un sistema avanzado de impresión.
- Permite calcular el tiempo de ejecución de las consultas.
- Permite la creación de diagramas basados en las consultas.
- Guarda un historial de todas las consultas hechas.[8]

Conclusiones

En este capítulo se analizaron las tecnologías más utilizadas a nivel mundial en el diseño de base de datos. Dentro de ellas fueron seleccionadas las más propicias para el desarrollo del sistema, teniendo en cuenta sus características y su aplicación en las nuevas tendencias actuales se utilizará el PostgreSQL como sistema de gestión de base de datos, el Case Studio 2 para el modelado de los datos debido a sus características y compatibilidad con el PostgreSQL. En el desarrollo de la ingeniería de software del sistema se decidió utilizar la metodología RUP utilizando el lenguaje de modelado UML y como herramienta CASE de modelado a Visual Paradigm que utiliza a UML como lenguaje de modelado.

Capitulo2. Descripción y análisis de solución

Introducción

En este capítulo se definen las actividades más importantes que posibilitaron la construcción de la base de datos, teniendo como resultado final el diseño de la misma.

Se hará una descripción y fundamentación de la arquitectura así como los requisitos funcionales y se determinarán las características que debe cumplir la base de datos a tener en cuenta en su construcción, como objetivo final se obtendrá el diagrama de clases persistentes, la descripción de las clases persistentes, diagrama entidad relación de la base de datos y la descripción de las tablas.

2.1 Descripción de la arquitectura y fundamentación.

La Base de Datos propuesta contiene un esquema donde se encuentran 18 tablas que se utilizan en 4 módulos. Las tablas que gestionan la mayor cantidad de datos son las tablas atleta, mesosistema y test_pedagogico, así como la relación existente entre ellas. El acceso a la Base de Datos será a través de peticiones que el servidor de páginas Web le hará al servidor de Base de Datos, donde existirán las funciones encargadas de dar respuesta a dichas peticiones. La aplicación web se encuentra en un servidor web el cual se comunica con la base de datos por el protocolo <<tcp/ip>>, para acceder a esta la comunicación es a través de <<http>>. El servidor cuenta con todas las funciones básicas (insert, select, update, delete) para realizar cualquier operación sobre la base de datos.

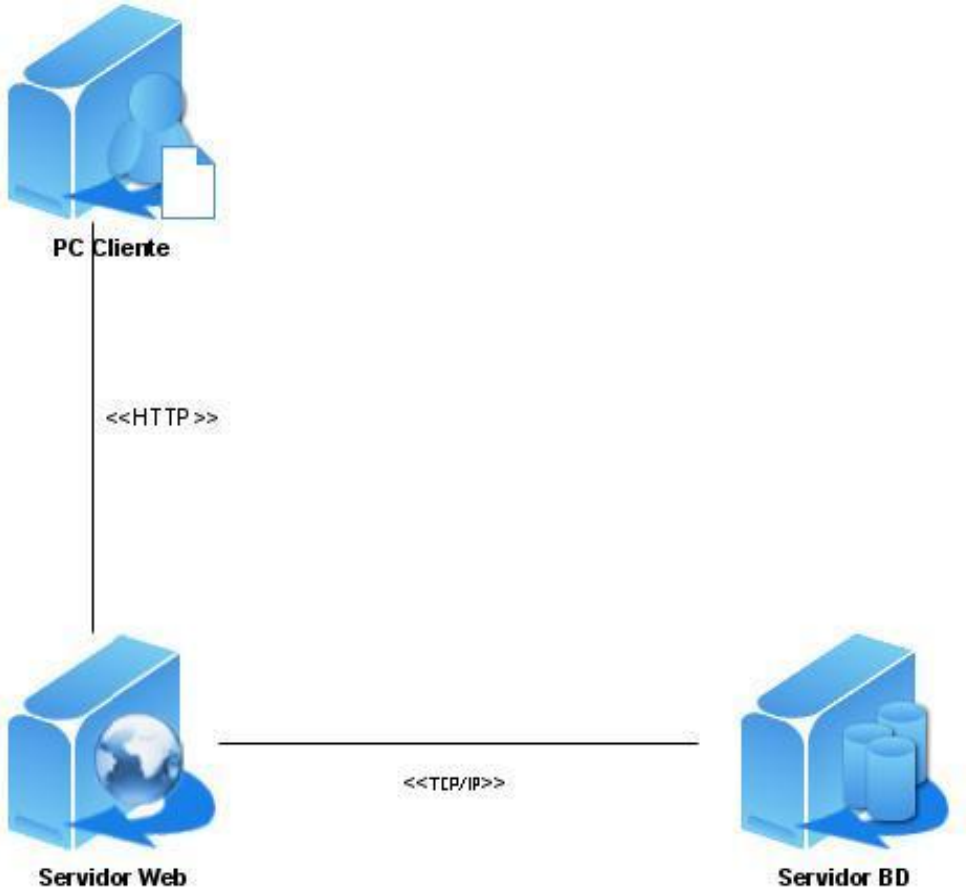

Figura 2: Diagrama de despliegue.

2.2 Análisis de optimización de Consultas.

Muchos administradores de base de datos resuelven los problemas de rendimiento solamente ajustando el rendimiento del servidor en el sistema: por ejemplo, el tamaño de la memoria, el tipo del sistema de archivos, número y tipo de procesadores de impresión. Sin embargo, muchos de los problemas de rendimiento no se pueden resolver de esta forma. Se deben solucionar mediante el análisis de las consultas y actualizaciones que la aplicación envía a la base de datos, y la forma en que estas consultas y actualizaciones de la aplicación interactúan con los datos y el esquema de la base de datos.

Las consultas son un lenguaje de acceso a las base de datos que permite realizar diversos tipos de operaciones sobre las mismas. Realizan un manejo del álgebra y cálculo relacional con el fin de recuperar la información almacenada en una base de datos así como hacer cambios sobre ella. Durante la ejecución de las consultas se pasa por 4 fases:

- Parser: Chequea la sintaxis de la consulta y genera el árbol de consulta (Query tree).
- Rewriter: Procesa el árbol de consulta y comprueba la existencia de reglas reescribiendo de nuevo la consulta si es necesario.
- Planer: Genera el plan de consulta a partir del árbol de consulta analizando todos los posibles caminos y escogiendo el de menor coste.
- Executor: Recorre recursivamente el árbol del plan de consulta y accede al disco para obtener los datos necesarios. Realiza las uniones entre tablas, cálculos y ordenamientos necesarios para devolver las tuplas resultado de las consultas.

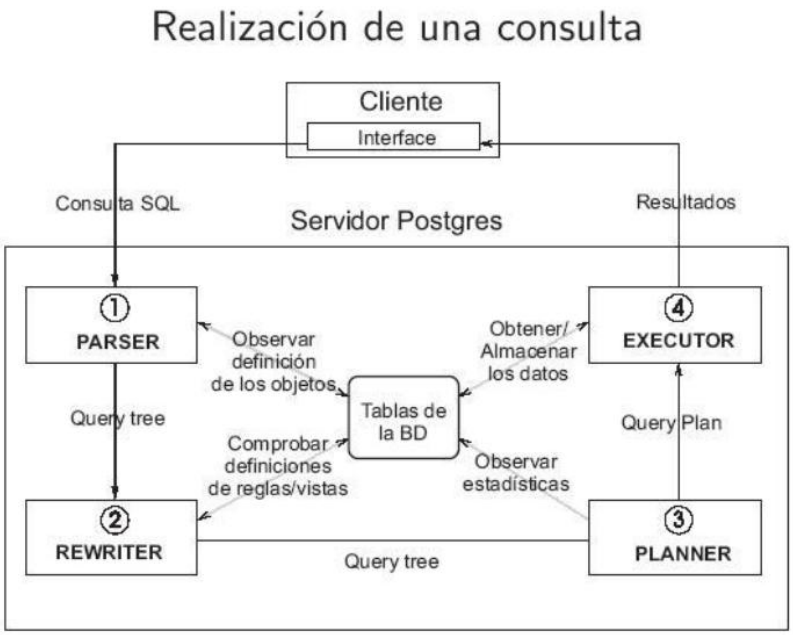

Figura 3: Diagrama de ejecución de una consulta en PostgreSQL.

La optimización de consultas es algo necesario para un mejor funcionamiento y rapidez de la Base de Datos, esto consiste en buscar las mejores opciones para realizar consultas.

Para analizar el tema de optimización de consultas en la base de datos propuesta se definieron determinados parámetros que son de alta importancia para que el diseño sea el más adecuado posible.

1. Gestión y elección de índices

Los índices son campos elegidos arbitrariamente por el constructor de la base de datos para permitir la búsqueda a partir de dicho campo a una velocidad notablemente superior. Sin embargo, esta ventaja se ve contrarrestada por el hecho de ocupar

mucha más memoria (el doble) y de requerir para su inserción y actualización un tiempo de proceso superior.

La clave de un índice es un conjunto de atributos. El índice permite, dado un valor de cada uno de los atributos del conjunto, acceder a todos los registros de la tabla sobre la cual se construye el índice que tienen esos valores en los atributos de la clave del índice.

2. Es necesario tratar de limitar el conjunto de resultados de consultas mediante el uso de la cláusula WHERE.

Esto puede resultar óptimo en el rendimiento, ya que devolverá al cliente, sólo las filas que se desean bajo la condición expuesta en la cláusula where, no todas las filas de la tabla (s). Esto puede reducir el tráfico de la red y aumentar el rendimiento global de la consulta.

3. Utilizar puntos de vista y procedimientos almacenados en lugar de pesadas consultas.

Esto puede reducir el tráfico de la red, el cliente enviará al servidor los datos de los procedimientos almacenados o sólo ver el nombre (tal vez con algunos parámetros) en lugar de las grandes preguntas de pesados textos. Esto puede ser utilizado para facilitar la gestión de permiso, también porque puede restringir el acceso del usuario a las columnas de la tabla que no deberían ver.

4. Siempre que sea posible se debe de evitar usar la cláusula SELECT COUNT (*) si se desea devolver el total de filas de la tabla.

Debido a que el SELECT COUNT (*) hace un escaneo completo de la tabla para devolver el total de filas de la tabla, puede tomar mucho tiempo para las grandes tablas. No hay otra manera de determinar el total de filas en una tabla. En este caso, se puede utilizar una tabla de índices. Existe una columna cantidad con la cantidad de datos de la tabla. Esta columna contiene el total de registros para cada tabla en su base de datos. Además tiene otra columna con los nombres de las tablas existentes. Por lo tanto, puede usar la siguiente declaración en lugar de seleccionar SELECT COUNT (*):

```
CREATE TALBE REGISTROS (

cantidad integer,

nombre_tabla character varying,
```

)

SELECT REGISTROS.cantidad from REGISTROS WHERE REGISTROS.nombre_tabla='tabla';

Actualizando los datos de la tabla incrementando o decrementando la cantidad de la tabla REGISTROS en dependencia de la operación insert o delete que se ejecute sobre las tablas de la base de datos.

También se puede utilizar la sentencia SELECT COUNT (IDENTIFICADOR) pasando el id de la tabla y de esta forma reduciendo un poco el tiempo de ejecución de la consulta.

5. Tabla de variables en lugar de tablas temporales.

Las variables tipo tabla requieren menos recursos y bloqueo que las tablas temporales, por lo que las variables tipo tablas deben utilizarse siempre que sea posible.

6. Es recomendable hacer buen uso del HAVING, GROUP BY, porque puede sobrecargar la selección.

La cláusula HAVING se utiliza para restringir el conjunto de resultados devueltos por la cláusula GROUP BY. Cuando se utiliza GROUP BY con la cláusula HAVING, la cláusula GROUP BY divide las filas agrupadas en conjuntos de filas y sus valores agregados y a continuación, la cláusula de HAVING elimina los valores no deseados agregados en grupos. Esto puede mejorar el rendimiento de su consulta.

7. La cláusula DISTINCT es costosa de ejecutar porque generalmente involucra un ordenamiento de las tuplas resultantes para eliminar duplicados. Es necesario analizar si realmente hace falta usar esa cláusula.

8. En algunas consultas el usuario almacena resultados intermedios explícitamente en tablas temporales. Estas tablas pueden bajar el rendimiento de la consulta por dos razones, una porque fuerza un orden de ejecución que quizás no sea ideal y segundo, porque obliga a ejecutar actualizaciones al Diccionario de Datos (DD) lo cual tiene el peligro de convertirlo en un cuello de botella.

9. Por otro lado, las tablas temporales pueden tener un efecto positivo al reescribir consultas que contienen subconsultas correlacionadas complejas. Las subconsultas correlacionadas pueden ejecutarse ineficientemente, pero si se logra calcular primero

las tuplas de la subconsulta y almacenarlas en una tabla temporal, se puede reescribir la consulta original, sin la subconsulta, accediendo a la tabla temporal.

10. Evitar los tipos de columna de longitud variable

Los campos con longitud fija (CHAR, por ejemplo) son más rápidos que los de longitud variable (VARCHAR, por ejemplo). Pero, como contrapartida, ocupan más espacio en disco. Aún así, para las tablas, se recomienda evitar el uso de las columnas de longitud variable (VARCHAR, BLOB, and TEXT).

11. Las condiciones de join se pueden evaluar más eficientemente contra un índice primario. Y en general, en términos de rendimiento es preferible evaluar una condición de igualdad numérica que una condición de igualdad sobre cadenas (”strings”) de caracteres.

12. Otra particularidad importante del manejador que es bueno conocer es si el orden de las tablas en la cláusula FROM puede afectar la implementación del join utilizada, posiblemente esto es relevante para joins que involucran 5 tablas o más.

Se muestra un ejemplo del tiempo de ejecución utilizando el comando EXPLAIN. Este comando explica cómo se lleva a cabo una consulta SELECT, la forma de usar los índices y la unión entre las tablas.

```
SELECT
test_pedagogico.id_test,
microsistema.numero_micro,
FROM
test_pedagogico
INNER JOIN microsistema
ON (test_pedagogico.nombre_macro = macrosistema.nombre_macro)
WHERE
(microsistema.fecha_inicio < test_pedagogico.fecha) AND
(microsistema.fecha_fin > test_pedagogico.fecha)
```

Con más de 700 registros en la tabla donde se busca, esta consulta sencilla se demoró 0.12 segundos.

```
SELECT
test_pedagogico.id_test,
microsistema.numero_micro,
FROM
```

test_pedagogico, microsistema
WHERE
(test_pedagogico.nombre_macro = macrosistema.nombre_macro) AND
(microsistema.fecha_inicio < test_pedagogico.fecha) AND
(microsistema.fecha_fin > test_pedagogico.fecha)

Para 700 registros esta consulta sencilla demoro 0.16 segundos
En este otro ejemplo:

SELECT
atleta.ci,
atleta.nombre,
atleta.apellidos,
atleta.division,
parametro_cantidad.parametro,
parámetro_cantidad.nivel
FROM atleta
INER JOIN
parametro_cantidad
ON (atleta.ci = parametro_cantidad.ci)
WHERE
parámetro_cantidad.nivel>3

Teniendo más de 1 000 registros en la tabla donde se busca, esta consulta sencilla se demoró 0.06 segundos. Ahora se probará otra forma de hacer esto.

SELECT
atleta.ci,
atleta.nombre,
atleta.apellidos,
atleta.division,
parametro_cantidad.parametro,
parámetro_cantidad.nivel
FROM atleta, parametro_cantidad
WHERE
(atleta.ci = parametro_cantidad.ci) AND
(parámetro_cantidad.nivel>3)

Esto salió mucho peor con un tiempo de 0.12 segundos.

De los resultados obtenidos se puede concluir que si se tiene un dominio bastante amplio de las maneras de hacer las consultas y con las herramientas que permitan calcular tiempos de ejecución de dichas consultas, se pueden hacer peticiones al servidor de base de datos con mejores resultados en cuanto al tiempo de respuesta.

2.3 Selección y argumentación de los requisitos funcionales.

Para que un sistema de software funcione correctamente se debe lograr una comunicación efectiva entre los usuarios y el equipo de proyecto con el objetivo de llegar a un entendimiento de lo que hay que hacer, lo que constituye la clave del éxito en la producción de un software. Aquí radica la importancia que en los últimos años se le ha dado a la identificación de los requerimientos como parte del proceso de desarrollo del software. Los requisitos se pueden clasificar en: funcionales y no funcionales. Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. Los no funcionales son propiedades o cualidades que el producto debe tener, permitiendo que el mismo sea atractivo, usable, rápido y confiable.

2.3.1 Requisitos funcionales:

RF1 Gestionar atleta

RF1.1 Insertar datos de un atleta: nombre, apellidos, número de CI, dirección particular (provincia, municipio, localidad, calle, número, entre calles), edad, edad deportiva, tiempo en el equipo nacional, foto, división, teléfono, nombre de la madre, nombre del padre, nivel de escolaridad, centro de estudio, organizaciones políticas, activo y número de pasaporte.

RF1.2 Consultar datos de un atleta

RF1.3 Modificar datos de un atleta

RF1.4 Exportar a PDF los datos de un atleta

RF2 Gestionar Test Pedagógico

RF2.1 Insertar Test Pedagógico: número macrosistema, cantidad de barra fija, paralela, soga, escalera, barra invertida, tubo, pron, halon, cuclilla, arranque de fuerza, velocidad 50 metros, velocidad 50 metros/volante, resistencia a la fuerza 200 metros, resistencia 1500 metros, flexión ventral, arqueo, híper-extensión, sapo, prueba teórica, prueba técnica (4

minutos de proyecciones, 30 segundos de proyecciones y 1 minuto de proyecciones).

RF2.2 Consultar Test Pedagógico

RF2.3 Modificar Test Pedagógico

RF2.4 Exportar a PDF los datos de un test pedagógico

RF3 Gestionar Plan Gráfico

RF3.1 Insertar Plan Gráfico: nombre del plan gráfico, año, competencia fundamental, lugar, para cada macrosistema poner la fecha de inicio y de fin de cada uno de sus mesosistemas, de cada mesosistema la cantidad de horas/sesiones, combates, rango tolerancia de peso inicial y final, volumen inicial y final e intensidad inicial y final, para cada microsistema los test pedagógicos, pruebas médicas, competencias y campos de entrenamiento que se van a realizar, además de la planificación de cada uno de los microsistemas de cada mesosistema.

RF3.2 Consultar Plan Gráfico

RF3.3 Modificar Plan Gráfico

RF3.4 Exportar a PDF los datos de un plan gráfico

RF4 Gestionar Usuario

RF4.1 Insertar un nuevo usuario. (Solicitar: nombre, apellidos, CI, usuario, contraseña y privilegio)

RF4.2 Modificar datos de un usuario

RF4.3 Eliminar cuenta de usuario

RF4.4 Consultar datos de un usuario registrado. (usuario, nombre y apellidos)

RF5 Autenticar usuario

RF5.1 Permitir que el usuario entre al sistema dado el nombre de usuario y contraseña.

RF5.2 Permitir que el usuario cambie su contraseña una vez autenticado.

RF5.3 Permitir que el usuario pueda salir de la sesión de trabajo una vez autenticado.

RF5.4 Permitir que el usuario sólo pueda acceder a los recursos que tiene privilegio.

2.3.2 Requerimientos no funcionales.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Son las características que hacen al producto atractivo, usable, rápido o confiable. En muchos casos los requerimientos no funcionales son fundamentales en el éxito del producto. Normalmente están vinculados a requerimientos funcionales.

RNF1. Requerimiento de software

PostgreSQL 8.2.1.

RNF2. Requerimiento de hardware

2.1 El servidor debe tener como mínimo las siguientes características: Una PC con procesador Pentium II, 128 megabytes (Mb) de memoria RAM y 10Gb de capacidad en disco duro.

2.2 Las computadoras cliente deben tener como mínimo las siguientes características: Procesador Pentium III, 64 Mb de memoria RAM y 10Gb de capacidad en disco duro.

RNF3. Apariencia o interfaz externa

3.1 Interfaz con un diseño sencillo, de forma tal que facilite el uso a los usuarios que no tengan entrenamiento previo en sistemas con estas características; sin dejar de ser amigable, legible, interactiva y fácil de usar.

RNF4. Usabilidad

4.1 El sistema podrá ser usado por cualquiera de las personas que integra el colectivo técnico, por lo que es necesario que cuente con una interacción mediante función que facilite el uso de la aplicación.

RNF5. Rendimiento

5.1 La aplicación debe ser rápida y precisa. Estará implementado sobre una el sistema de gestión de base de datos PostgreSQL, facilitando su uso a través de la red.

RNF6. Portabilidad

6.1 Puede funcionar en cualquier sistema operativo, debido a que el servidor web y el de base de datos son multiplataforma.

6.2 El producto debe correr sobre una plataforma web, codificada en PHP y sus sistemas de base de datos en PostgreSQL.

RNF7. Requerimiento de Soporte

7.1 Servidor de BD que soporta grandes volúmenes de datos y que tenga buena velocidad de procesamiento.

7.2 El sistema debe ser de fácil instalación.

7.3 El sistema debe estar bien documentado de forma tal que el tiempo de mantenimiento sea mínimo en caso de necesitarse.

RNF8. Requerimientos de Seguridad

8.1 El sistema debe caracterizarse por su disponibilidad e integridad.

8.2 La información debe ser confidencial, para ello se pretende establecer un sistema de permisos y usuarios para el acceso a la información. Para mantener la integridad en el mismo solo se podrá acceder al sistema después de autenticarse encriptándose la contraseña directamente en la máquina cliente utilizando el algoritmo de encriptación md5, por lo que viaja hacia el servidor de forma segura.

RNF9. Confiabilidad

9.1 El sistema debe ser capaz de mantener la integridad de los datos.

9.2 Solo el personal autorizado podrá acceder a la información solicitada.

RNF10. Requerimientos de Diseño.

10.1 Se utilizará como lenguaje de programación plpgsql.
10.2 La metodología a utilizar es RUP y el lenguaje de modelado UML.
10.3 El gestor de base de datos a utilizar será PostgreSQL.
10.4 El modelo 3 capas es el patrón de arquitectura a utilizar.

2.4 Diagrama de clases persistentes y modelo de objetos.

Una clase persistente es una clase entidad que tiene la capacidad de mantener su valor en el espacio y en el tiempo. Lo contrario son las clases temporales (transient) que son manejadas y almacenadas por el sistema en tiempo de ejecución por lo que dejan de existir cuando termina el programa.

Los diagramas de clases persistentes están agrupados en un diagrama principal y para su mejor comprensión se hace un diagrama por cada diagrama de clases del diseño. En las siguientes figuras se representan los diagramas de clases persistentes correspondiente al sistema y se describen los diferentes módulos en los que se ha fragmentado para una mejor comprensión.

DIAGRAMA DE CLASES PERSISTENTES.

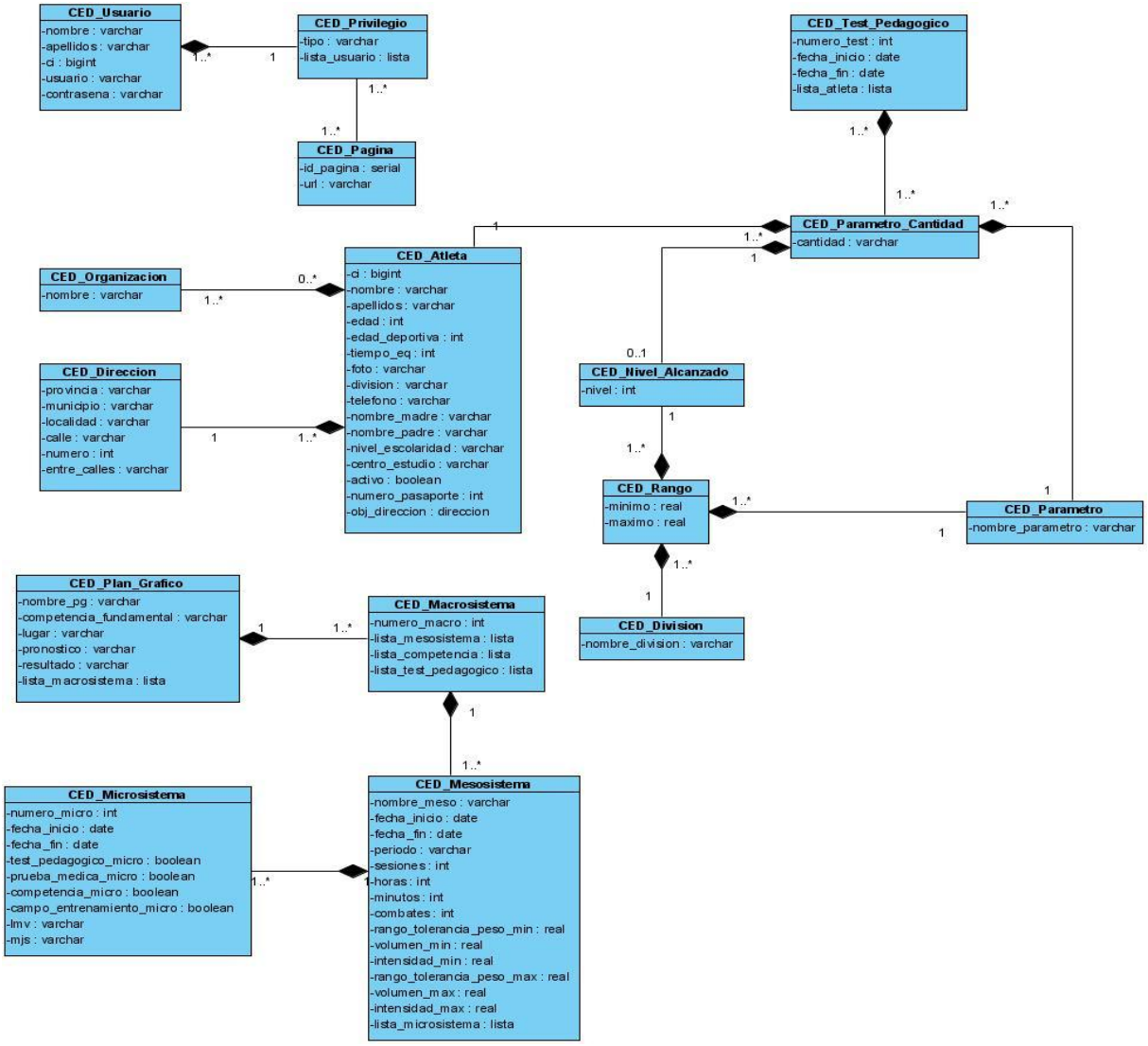

Figura 4: Diagrama de clases persistentes general.

DIAGRAMA DE CLASES PERSISTENTES

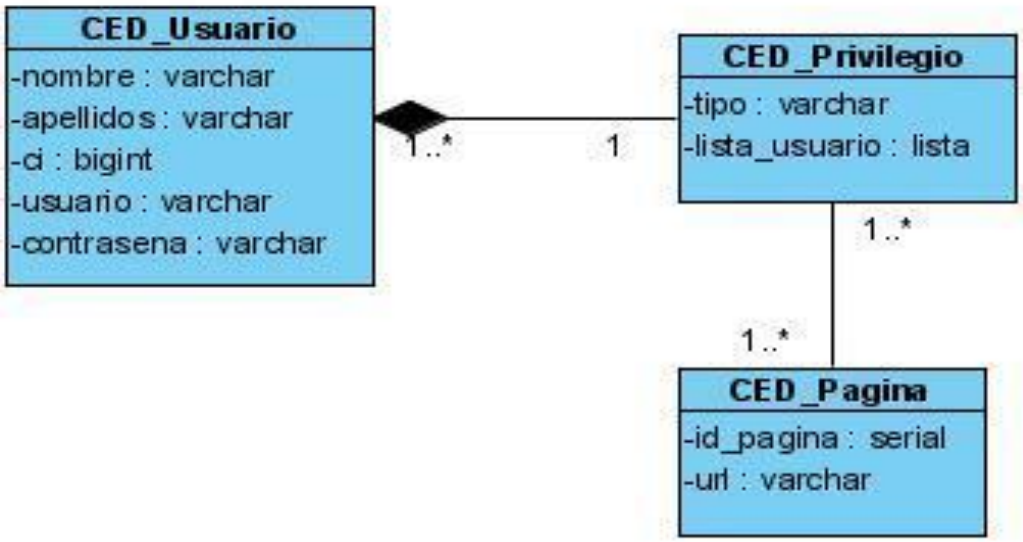

Figura 5: Diagrama de clases persistentes del módulo administrativo.

DIAGRAMA DE CLASES PERSISTENTES

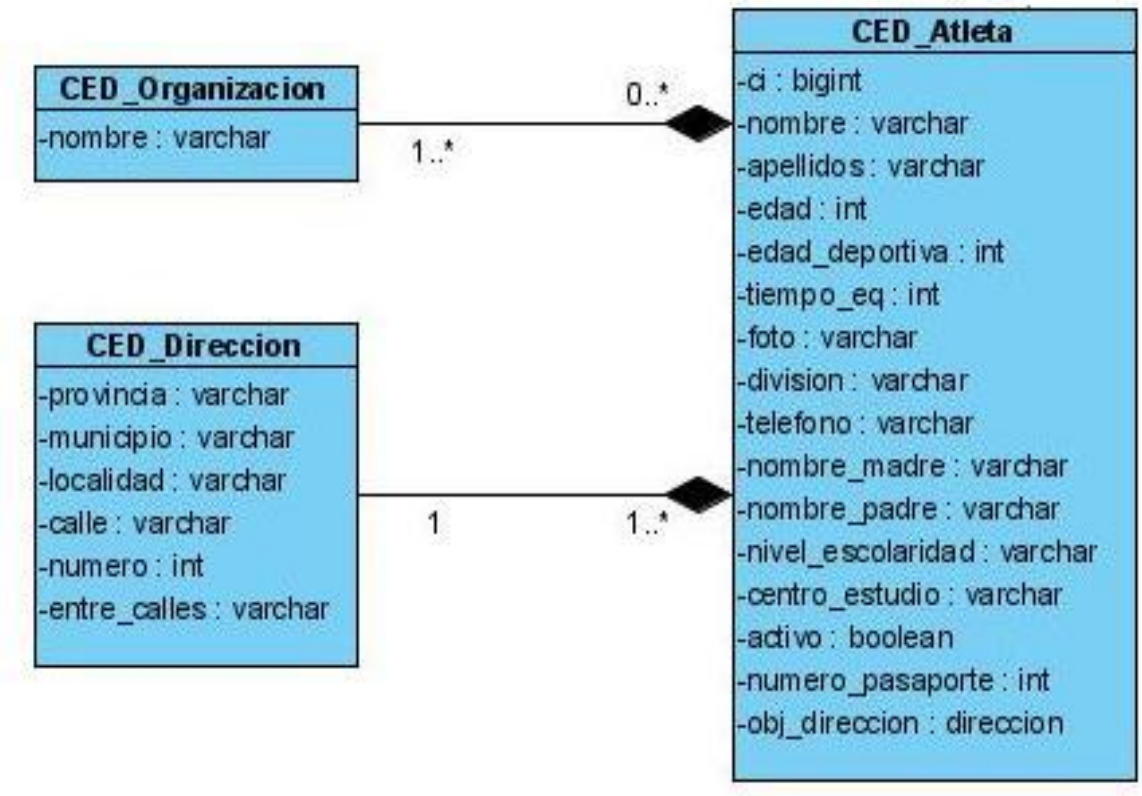

Figura 6: Diagrama de clases persistentes del módulo Atletas.

DIAGRAMA DE CLASES PERSISTENTES

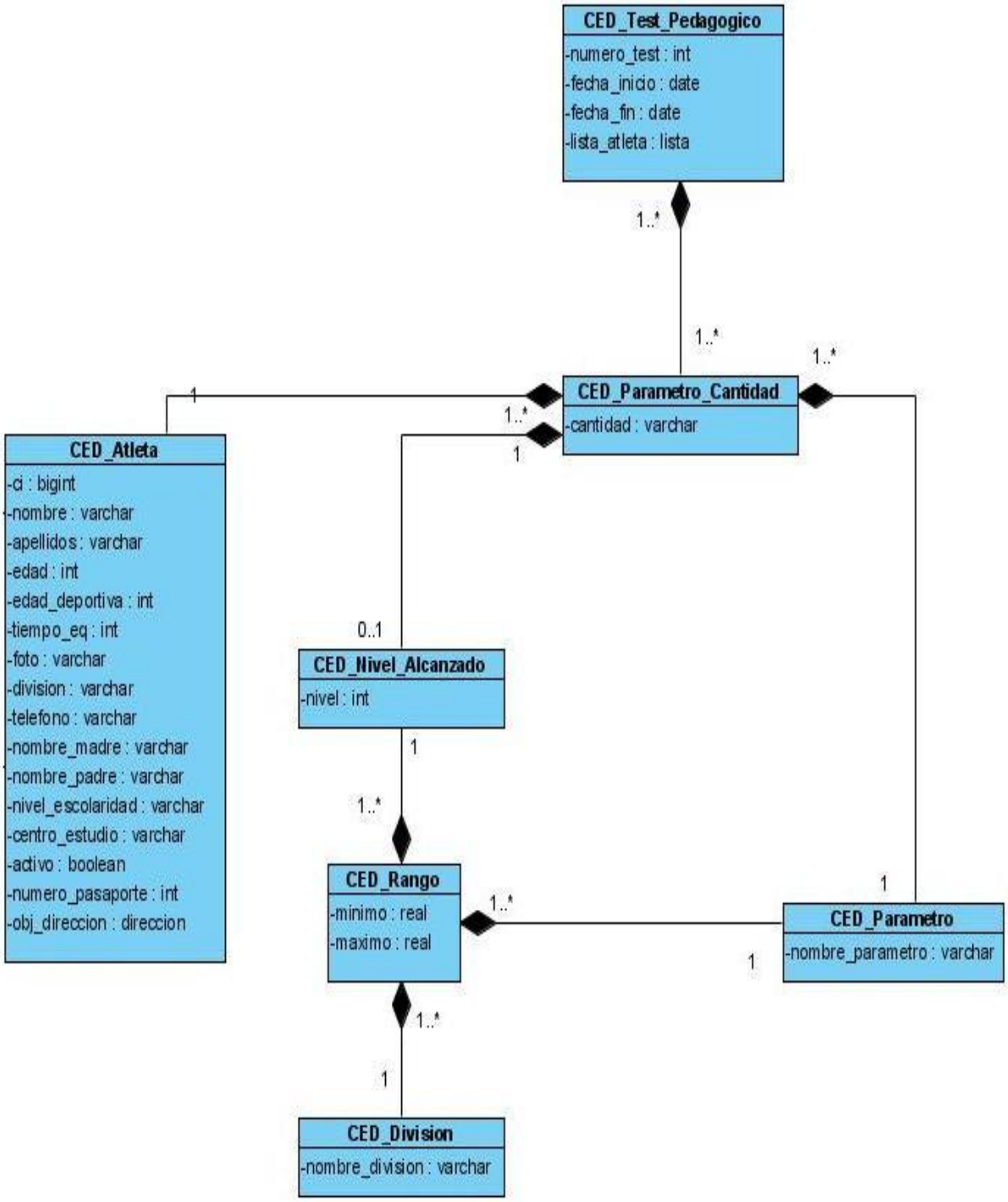

Figura 7: Diagrama de clases persistentes del módulo Test Pedagógico.

DIAGRAMA DE CLASES PERSISTENTES

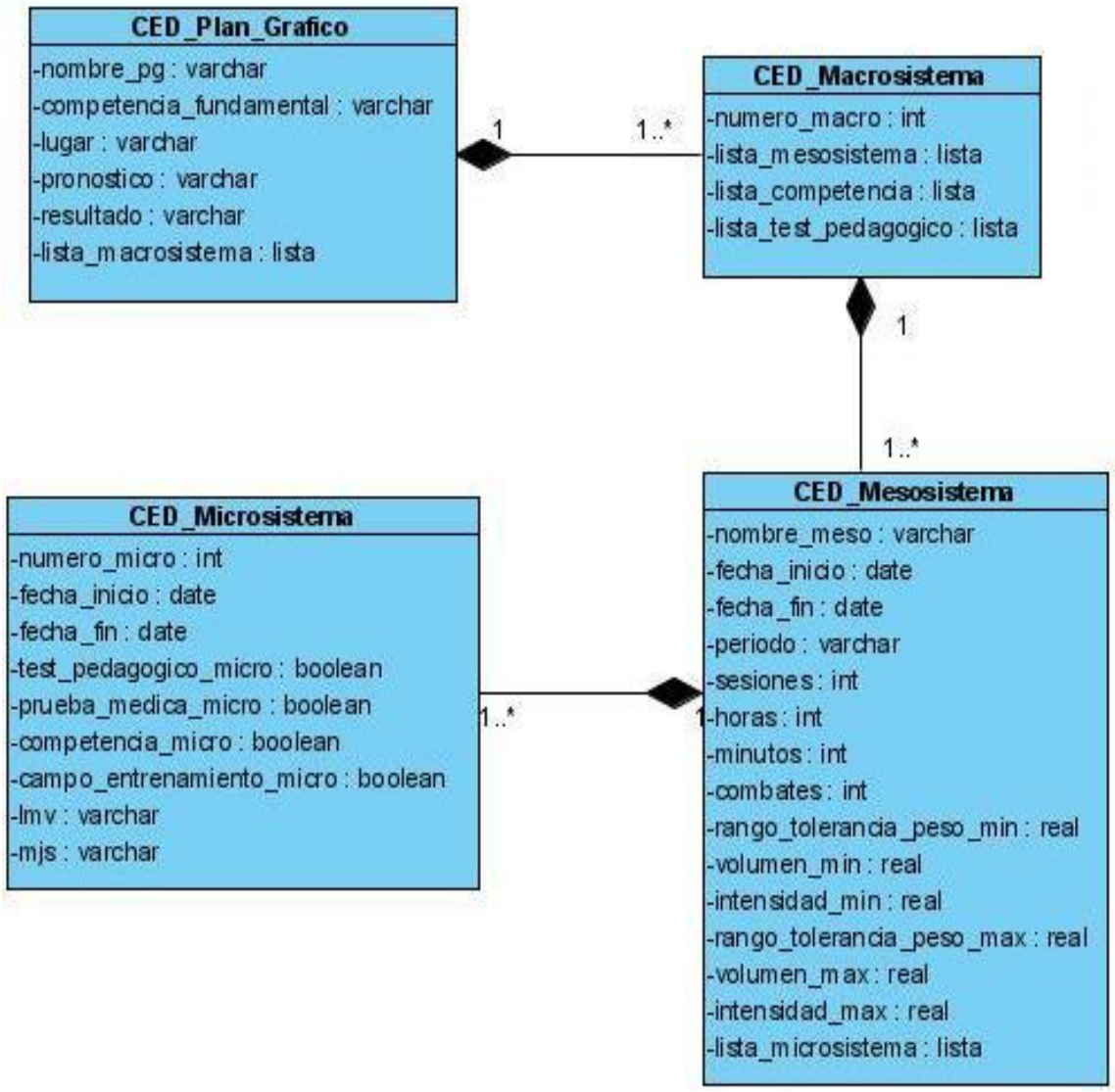

Figura 8: Diagrama de clases persistentes del módulo Plan Gráfico.

2.5 Descripción de las clases.

Las clases persistentes poseen un conjunto de atributos que se describirán en las tablas que se muestran en el Anexo 1. Cada clase tiene como métodos u operador:

- Un constructor: se llama igual que la clase y contiene cada uno de los atributos que sea necesario inicializar.
- Métodos get, que retornan el valor del atributo: contiene la estructura del get primeramente la palabra get y luego el parámetro a retornar, ejemplo: GetNombre ().
- Métodos set para modificar el valor de los atributos: contiene la estructura del set primeramente la palabra Set y luego el parámetro a cambiar, ejemplo: SetApellidos (char apellidos).

Los métodos no se pondrán en las tablas para optimizar espacio y que no se vuelvan tan extensas. A continuación se muestran las clases persistentes del modelo.

Tabla 2: Clase Atleta.

Nombre: Atleta	
Atributo	**Tipo**
ci	bigint
nombre	varchar
apellidos	varchar
edad	int
edad_deportiva	int
tiempo_equipo	int
foto	varchar
division	varchar
telefono	varchar
nombre_madre	varchar
nombre_padre	varchar
nivel_escolaridad	varchar
centro_estudio	varchar
activo	bool
numero_pasaporte	int
ogj_direccion	direccion

Tabla 3: Clase Direccion.

Nombre: Direccion	
Atributo	**Tipo**
provincia	varchar
municipio	varchar
localidad	varchar
calle	varchar
numero	int
entre_calles	varchar

Tabla 4: Clase Organizacion.

Nombre: Organizacion	
Atributo	**Tipo**
nombre	varchar

Tabla 5: Clase Usuario.

Nombre: Usuario	
Atributo	**Tipo**
nombre	varchar
apellidos	varchar
ci	bigint
usuario	varchar
contrasena	varchar

Tabla 6: Clase Privilegio.

Nombre: Privilegio	
Atributo	**Tipo**
tipo	varchar

Tabla 7: Pagina.

Nombre: Pagina	
Atributo	**Tipo**
id_pagina	serial
url	varchar

Tabla 8: Clase Plan_Grafico.

Nombre: Plan_Grafico	
Atributo	**Tipo**
nombre_pg	varchar
competencia_fundamental	varchar
lugar	varchar
pronostico	varchar
resultado	varchar

Tabla 9: Clase Macrosistema.

Nombre: Macrosistema	
Atributo	**Tipo**
numero_macro	int

Tabla 10: Clase Mesosistema.

Nombre: Mesosistema	
Atributo	**Tipo**
nombre_meso	varchar
fecha_inicio	date
fecha_fin	date
periodo	char
sesiones	int
horas	int
minutos	int
combates	int
rango_tolerancia_peso_min	real
rango_tolerancia_peso_max	real
volumen_min	real
volumen_max	real
Intensidad_min	real
Intensidad_max	real

Tabla 11: Clase Microsistema.

Nombre: Microsistema	
Atributo	**Tipo**
numero_micro	int
fecha_inicio	date
fecha_fin	date
test_pedagogico	boolean
prueba_medica	boolean
competencia	boolean
campo_entrenamiento	boolean
lmv	varchar
mjs	varchar

Tabla 12: Clase Parametro.

Nombre: Parametro	
Atributo	**Tipo**
nombre_parametro	varchar

Tabla 13: Clase Division.

Nomre: Division	
Atributo	**Tipo**
nombre_division	varchar

Tabla 14: Clase Nivel_Alcanzado.

Nombre: Nivel_Alcanzado	
Atriburo	**Tipo**
nivel	int

Tabla 15: Clase Rango.

Nombre: Rango	
Atributo	**Tipo**
minimo	real
maximo	real

Tabla 16: Clase Parametro_Cantidad.

Nombre: Parametro_Cantidad	
Atributo	**Tipo**
cantidad	varchar

Tabla 17: Clase Test_Pedagogico.

Nombre: Test_Pedagogico	
Atributo	**Tipo**
numero_test	int
fecha_inicio	date
fecha_fin	date

2.6 Diseño de la BD. Diagramas Entidad-Relación.

Las siguientes figuras representan los diagramas entidad-relación correspondiente a la base de datos propuesta para la aplicación informática.

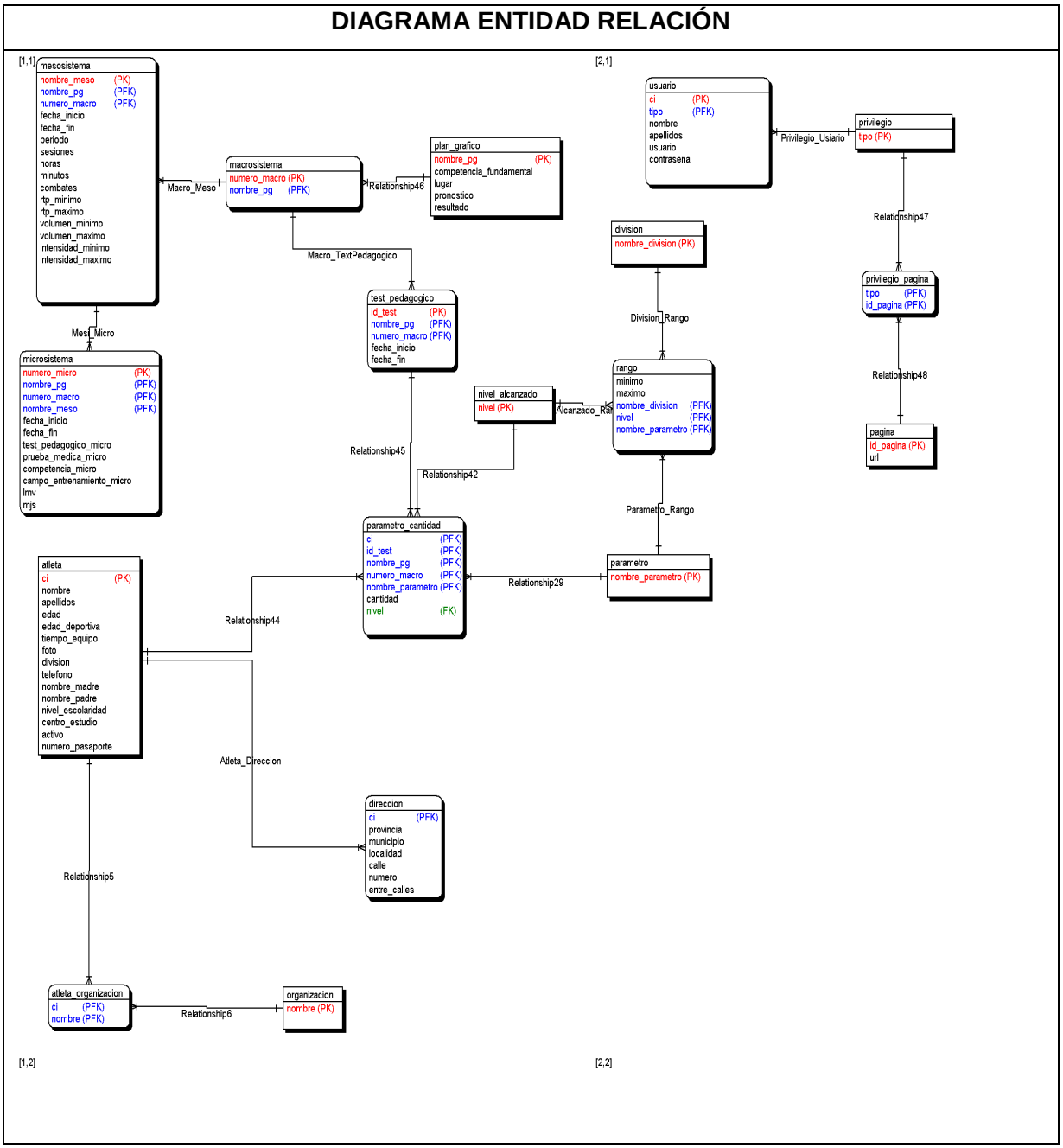

Figura 9: Diagrama Entidad-Relación General.

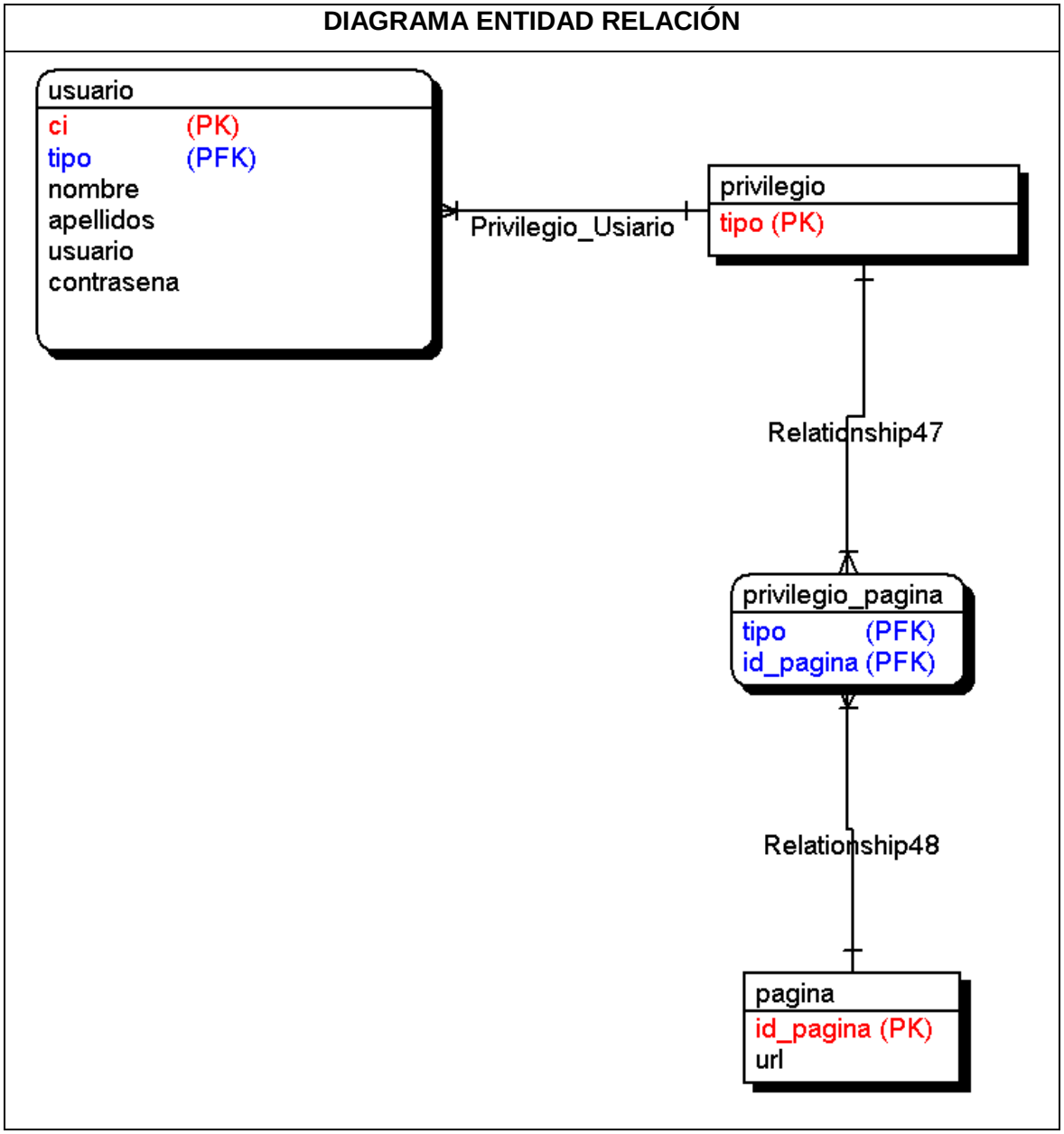

Figura 10: Diagrama Entidad-Relación del módulo Administrativo.

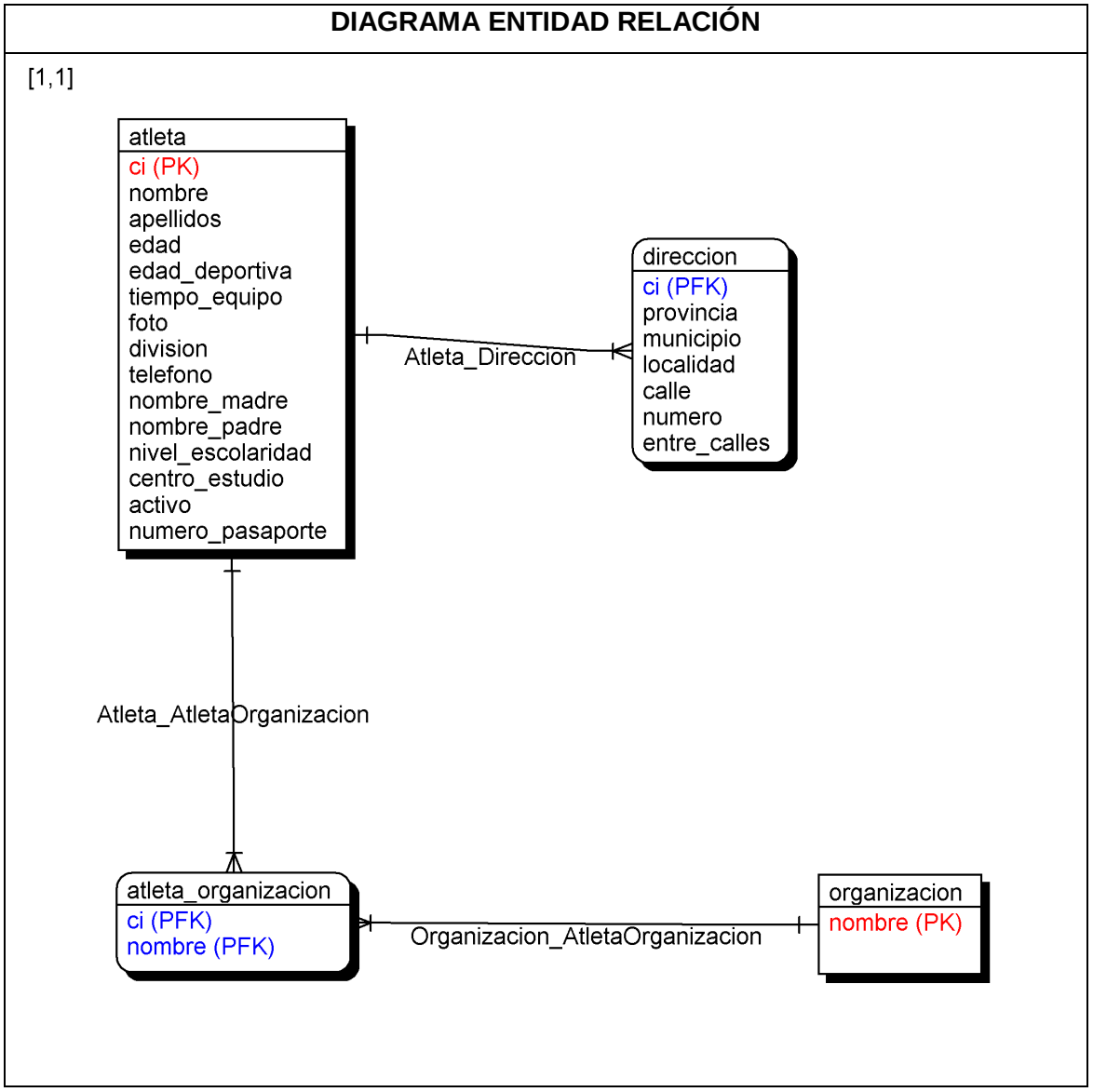

Figura 11: Diagrama Entidad-Relación del módulo Atleta.

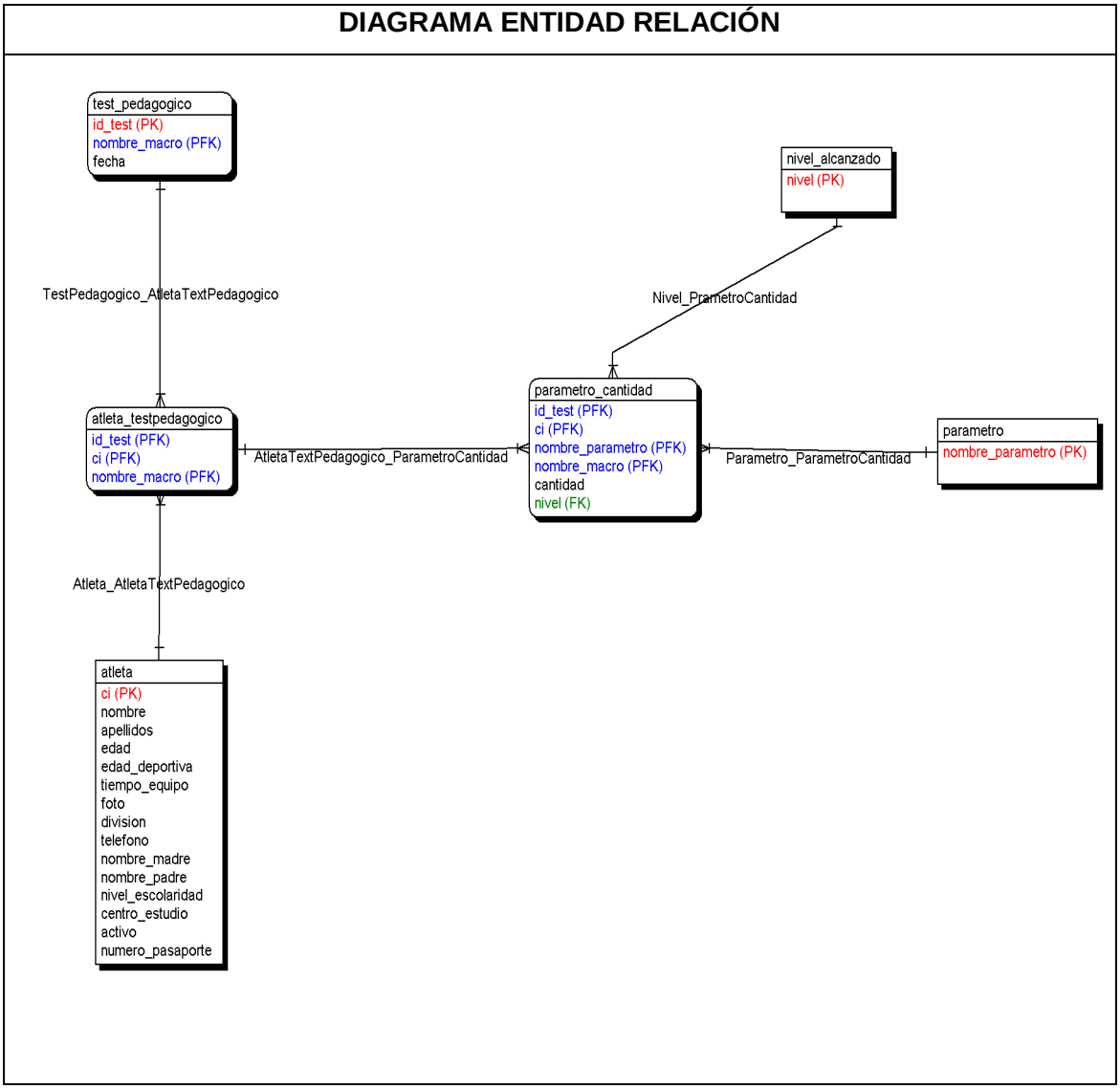

Figura 12: Diagrama Entidad-Relación del módulo Test Pedagógico.

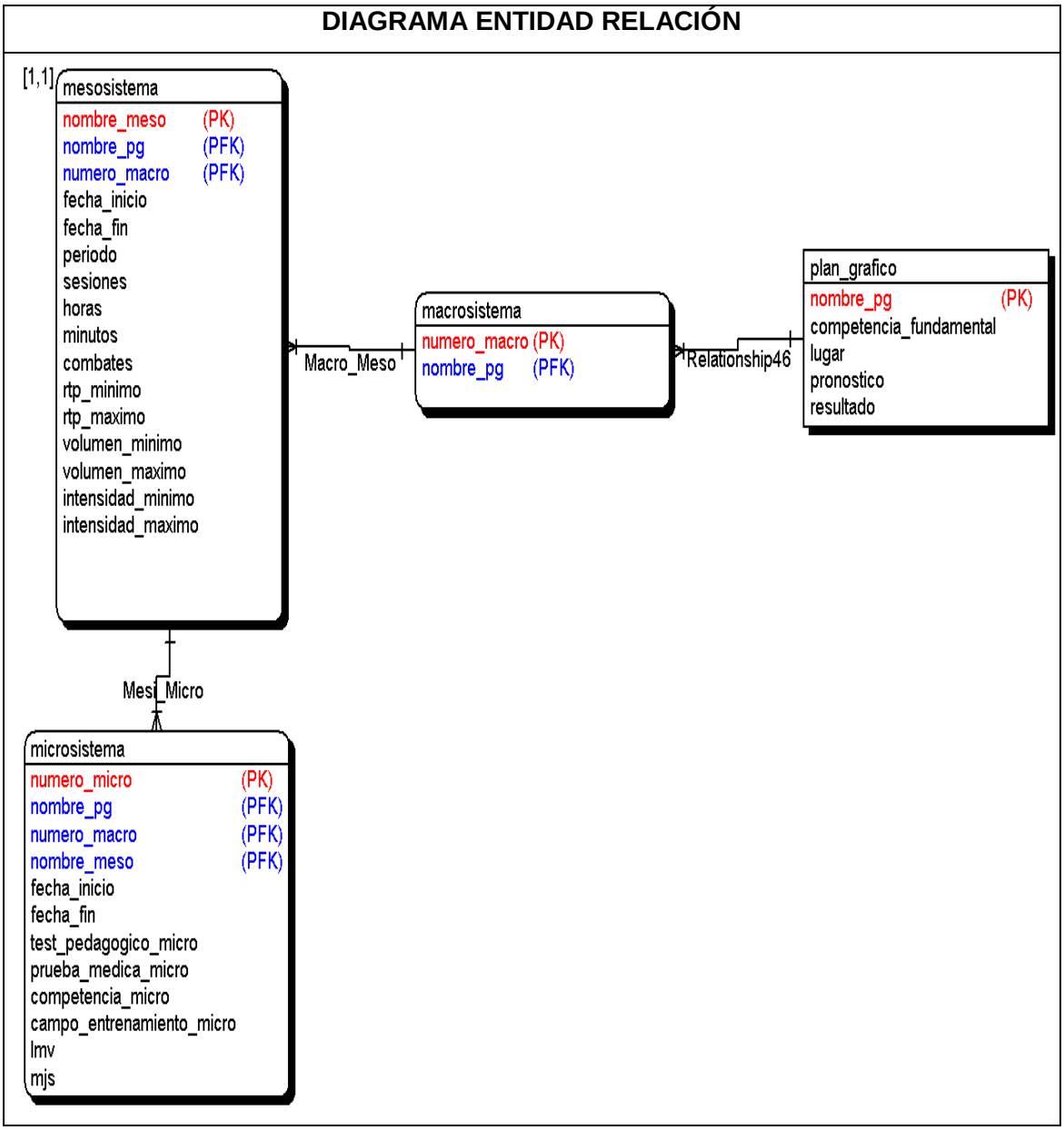

Figura 13: Diagrama Entidad-Relación del módulo Plan Gráfico.

2.7 Descripción de las tablas del diseño de la base de datos.

En esta sección se describen los datos que se almacenan en cada tabla del diagrama entidad relación de la base de datos propuesta, al igual que se explica brevemente lo que representan todos sus atributos.

Partiendo de los diagramas de clases persistentes se obtiene el modelo de datos realizando una correcta representación de las cases en tablas.

Se inicia al representar todas las clases con relaciones simples como entidades del modelo de datos, en las relacionadas de 1...n se le agrega la llave primaria del extremo 1 como llave extranjera al extremo de multiplicidad n. En el caso de las relaciones n...m es necesario crear una nueva tabla donde almacene las llaves primarias de ambas clases así como cualquier atributo que esté presente en la relación.

El Diagrama Entidad Relación está diseñado bajo el lenguaje plpgsql del gestor de base de datos Postgres por lo que lo tipos de las variables es el adecuado a este lenguaje. Ahora se muestra una descripción de las tablas de la base de datos. Para profundizar ver el Anexo 1.

Tabla 18: atleta.

Nombre: atleta		
Descripción: Almacena los datos correspondiente a cada atleta		
Atributo	**Tipo**	**Descripción**
ci	bigint	Carnet de identidad
nombre	varchar	Nombre del atleta
apellidos	varchar	Apellidos del atleta
edad	integer	Edad del atleta
edad_deportiva	integer	Edad practicando deporte
tiempo_equipo	integer	Tiempo en el equipo
foto	varchar	Nombre de la foto del atleta
division	varchar	División en la que compite
telefono	varchar	Teléfono personal
nombre_madre	varchar	Nombre de la madre
nombre_padre	varchar	Nombre del padre
nivel_escolaridad	varchar	Nivel escolar
centro_estudio	varchar	Centro de estudio
activo	boolean	Si está practicando judo
numero_pasaporte	integer	Número del pasaporte

Tabla 19: direccion.

Nombre: direccion
Descripción: Almacena la dirección del atleta

Atributo	Tipo	Descripción
ci	bigint	Carnet de identidad atleta
provincia	varchar	Provincia
municipio	varchar	Municipio
localidad	varchar	Localidad
calle	varchar	Calle
numero	integer	Número de la casa o apto
entre_calles	varchar	Entre calles si tiene

Tabla 20: organizacion.

Nombre: organizacion		
Descripción: Almacena las organizaciones políticas y de masas		
Atributo	**Tipo**	**Descripción**
nombre	varchar	Nombre de la organización

Tabla 21: atleta_organizacion.

Nombre: atleta_organizacion		
Descripción: Almacena los identificadores de la tabla competencia y la tabla organización		
Atributo	**Tipo**	**Descripción**
ci	bigint	Carnet de identidad del atleta
nombre	varchar	Nombre de la organización

Tabla 22: plan_grafico.

Nombre: plan_grafico		
Descripción: Almacena los macrosistemas		
Atributo	**Tipo**	**Descripción**
Nombre_pg	varchar	Nombre del plan grafico
competencia_fundamental	varchar	La competencia al finalizar el macrosistema
lugar	varchar	Lugar donde se realiza la competencia
pronostico	varchar	Pronóstico de la competencia
resultado	varchar	Resultado obtenido

Tabla 23: test_pedagogico.

Nombre: test_pedagogico		
Descripción: Almacena los test efectuados en el macrosistema		
Atributo	**Tipo**	**Descripción**
id_test	integer	Identificador del test
nombre_macro	varchar	Nombre macrosistema
fecha	date	Fecha de realización

Tabla 24: atleta_testpedagogico.

Nombre: atleta_testpedagogico		
Descripción: Almacena los identificadores del atleta y del test que el realiza		
Atributo	**Tipo**	**Descripción**
Id_test	integer	Identificador del test
ci	integer	Identificador del atleta
nombre_macro	varchar	Nombre de macrosistema

Tabla 25: parametro.

Nombre: parametro		
Descripción: Almacena los parámetros que se le miden al atleta en el test		
Atributo	**Tipo**	**Descripción**
nombre_parametro	varchar	Nombre del parámetro

Tabla 26: parametro_cantidad.

Nombre: parametro_cantidad		
Descripción: Almacena la cantidad de ejercicios realizados por el atleta en el test		
Atributo	**Tipo**	**Descripción**
id_test	integer	Identificador del test
ci	integer	carnet de identidad
nombre_parametro	varchar	Nombre del parámetro evaluado
cantidad	varchar	Cantidad realizada
nivel	integer	Nivel alcanzado

Tabla 27: macrosistema.

Nombre: macrosistema
Descripción: Almacena los macrosistemas

Atributo	Tipo	Descripción
numero_macro	interger	Número del macrosistema

Tabla 28: mesosistema.

Nombre: mesosistema		
Descripción: Almacena los microsistemas		
Atributo	**Tipo**	**Descripción**
nombre_meso	varchar	Nombre del mesosistema
nombre_macro	varchar	Nombre del macrosistema
fecha_inicio	date	Fecha de inicio
fecha_fin	date	Fecha de culminación
periodo	varchar	Periodo en el que se encuentra
sesiones	integer	Sesiones de entrenamiento
horas	integer	Horas de entrenamiento
minutos	integer	Minutos de entrenamiento
combates	integer	Cantidad de combates
rango_tolerancia	varchar	-
volumen	varchar	-
intensidad	varchar	-

Tabla 29: microsistema.

Nombre: microsistema		
Descripción: Almacena los microsistemas		
Atributo	**Tipo**	**Descripción**
nombre_micro	varchar	Nombre del microsistema
nombre_macro	varchar	Nombre del macrosistema
nombre_meso	varchar	Nombre del mesosistema
fecha_inicio	date	Fecha inicio de semana
fecha_fin	date	Fecha de fin de la semana
test_pedagogico	boolean	Si se realiza test
prueba_medica	boolean	Si se realiza prueba
competencia	boolean	Si se realiza competencia
campo_entrenamiento	boolean	-

lmv	varchar	Planificación por días
mjs	varchar	Planificación por días

Tabla 30: usuario.

Nombre: usuario		
Descripción: Almacena los datos de los usuarios		
Atributo	**Tipo**	**Descripción**
ci	integer	Carnet de identidad
tipo	varchar	Tipo de privilegio
nombre	varchar	Nombre del usuario
apellidos	varchar	Apellidos del usuario
usuario	varchar	Usuario asignado
contrasena	varchar	contraseña

Tabla 31: privilegio.

Nombre: privilegio		
Descripción: Almacena los privilegios asignados por el administrador		
Atributo	**Tipo**	**Descripción**
tipo	varchar	Privilegios

Tabla 32: nivel_alcanzado.

Nombre: nivel_alcanzado		
Descripción: Almacena los niveles en los que están clasificados las pruebas		
Atributo	**Tipo**	**Descripción**
nivel	integer	Niveles 0…5

Tabla 33: división.

Nombre: division		
Descripción: Almacena las divisiones por cada parámetro		
Atributo	**Tipo**	**Descripción**
nombre_division	varchar	divisiones

Tabla 34: rango.

Nombre: rango
Descripción: Almacena las llaves para relacionar las tablas división, nivel_alcanzado, parametro.

Atributo	Tipo	Descripción
minimo	real	Rango mínimo
maximo	real	Rango máximo
nombre_division	varchar	Nombre de la división
nivel	integer	Nivel alcanzado
nombre_parametro	varchar	Nombre del parámetro

Conclusiones.

Se finaliza la etapa de descripción y análisis, se confeccionaron los diagramas que fueron necesarios utilizando el Visual Paradigm y el Case Studio 2, para obtener una mejor visión de la solución del problema presentado; además se presentó el diseño de la base de datos en la cual se van a almacenar los datos persistentes resultados de la gestión de la información del sistema.

Capitulo3. Validación del diseño realizado

Introducción

Este capítulo brinda la validación final al diseño de la base de datos, la integridad, la normalización y el análisis de la redundancia; pues su objetivo principal es hacer una transformación ascendente de la calidad del diseño mediante un conjunto de procesos y normas. En la realización del diseño se tuvo en cuenta un gran número de procesos, ejemplo la normalización que a pesar de ser explicada teóricamente en el momento de llevar a cabo la descomposición de las relaciones como parte del proceso de normalización se produjo un número importante de modificaciones. También se abarca la seguridad, el control de acceso, la realización y recuperación de backups de la base de datos, la trazabilidad de acciones y la validación funcional. Se establecen ejemplos demostrativos aplicados al diseño para facilitar la comprensión del mismo. Se fundamenta con un análisis exhaustivo la redundancia de información como aspecto fundamental en esta base de datos.

3.1 Validación teórica del diseño.

3.1.1 Integridad

Cuando se modela una realidad en el nivel conceptual, junto con identificar los datos que se desea manejar y cómo éstos se relacionan, surgen 'reglas' que deben cumplir dichos datos para representar lo más exactamente posible esa realidad. Es decir, el cumplimiento de estas reglas, ayuda a mantener la integridad de los datos, como por ejemplo si se tiene los atributos de atletas como la edad y edad deportiva, estos no pueden ser negativos porque en el mundo real esto no es posible, además la edad de las personas está dada entre 1-120 años aproximadamente creándose así una regla de integridad.

El término integridad, se refiere a la corrección y completitud de la información en una base de datos; cuando los contenidos que ésta almacena se modifican, puede perderse la integridad de varias formas. La mayor parte de las base de datos están sujetas a un gran número de reglas de integridad. Estas reglas se reflejan en la especificación de la base de datos a través de las Restricciones de Integridad de distintas maneras, como por ejemplo:

definición de dominios para los datos, especificación de cardinalidad en los tipos de interrelación, relacionada con las claves primarias, relacionadas con las claves foráneas.

Dos pasos importantes en el diseño de las tablas son la identificación de valores válidos para una columna y la determinación de cómo forzar la integridad de los datos en la columna. La integridad de datos pertenece a una de las siguientes categorías:

- Integridad de entidad: la clave primaria de una entidad no puede tener valores nulos y siempre deberá ser única, por ejemplo el ci en la tabla atleta.
- Integridad de dominio: restringe los valores que puede tomar un atributo respecto a su dominio, por ejemplo, que la edad siempre sea mayor que 15 años y menor que 40 (EDAD >= 15 - 40).
- Integridad referencial: La base de datos no debe contener valores de llaves ajenas sin concordancia.
- Integridad definida por el usuario.

Integridad de entidad

Esta regla se aplica a las claves primarias de las relaciones base: ningún atributo que forme parte de una llave primaria puede aceptar valores nulos. Por definición, una clave primaria es irreducible y se utiliza para identificar de modo único los registros. Irreducible significa que ningún subconjunto de la clave primaria, sirve para identificar las tuplas de modo único. Si se permite que parte de la clave primaria sea nula, se está diciendo que no todos sus atributos son necesarios para distinguir las tuplas, con lo que se contradice la irreductibilidad. Esta regla sólo se aplica a las relaciones base y a las claves primarias, no a las claves foráneas ni alternativas.

La integridad de entidad define una fila como entidad única para una tabla determinada. La integridad de entidad exige la integridad de las columnas de los identificadores o la clave principal de una tabla, mediante índices y restricciones UNIQUE, o restricciones PRIMARY KEY.

Integridad de dominio

La integridad de dominio viene dada por la validez de las entradas para una columna determinada. Puede exigir la integridad de dominio para restringir el tipo mediante tipos de

datos, el formato mediante reglas y restricciones CHECK, o el intervalo de valores posibles mediante restricciones FOREIGN KEY, restricciones CHECK, definiciones DEFAULT, definiciones NOT NULL y reglas. Un dominio de valores puede estar asociado a cada atributo. Los límites de dominio son la forma más elemental de restricciones de integridad. Son fáciles de probar en el sistema siempre que se introduce un nuevo dato. En siguiente ejemplo se muestra que un dominio es un rango de valores aceptados por un atributo.

Tabla 35: Tipos de datos en PGPLSQL.

Tipo de dato	Longitud	Descripción
INTEGER	31 bit	entero binario con signo de palabra completa
VARCHAR(*n*)	n	cadena de caracteres de longitud variable
CHAR(*n*):	n	cadena de caracteres de longitud fija
DECIMAL (*p*[,*q*])	número decimal con signo de *p* dígitos de precisión,	asumiendo *q* a la derecha para el punto decimal. (15 _ *p* _ *q*q _ 0). Si *q* se omite, se asume que vale 0
DATATIME	8 bytes	Un valor de fecha u hora entre los años 100 y 9999
TEXT	1 bite por carácter	De 0 a 255 caracteres

Una definición bien adecuada de restricciones de dominio no solo permite probar valores insertados en la base de datos; sino que también permite probar consultas para asegurar que la comparación que se hace tiene sentido.

Integridad referencial

La integridad referencial protege las relaciones definidas entre las tablas cuando se crean o se eliminan filas. Proporcionada por las características Durabilidad, Aislamiento, Consistencia e Indivisibilidad .En base de datos se denomina ACID a la propiedad de una base de datos para realizar transacciones seguras.

La integridad referencial garantiza que los valores de clave sean coherentes en las distintas tablas. Para conseguir esa coherencia, es preciso que no haya referencias a valores inexistentes y que, si cambia el valor de una clave, todas las referencias a ella se cambien en consecuencia en toda la base de datos.

- Indivisibilidad: 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.

Por ejemplo, se tiene una tabla plan_grafico que tiene como llave primaria nombre_pg, otra tabla macrosistema que tiene como llave primaria el numero_macro. La relación entre estas tablas es de uno a muchos ya que (siempre existe 2 macrosistemas en un Plan gráfico), por lo que nombre_pg, pasa para la tabla macrosistema como llave foránea (PFK). De esta manera se conocerían los macrosistemas de un plan gráfico.

En PostgreSQL esto es posible gracias a los conjuntos de acciones.

BEGIN WORK; COMMIT WORK;

BEGIN WORK; ROLLBACK WORK;

Integridad definida por el usuario

La integridad definida por el usuario permite definir las reglas específicas que plantea el INDER que no pertenecen a ninguna otra categoría de integridad. Todas las categorías de integridad admiten la integridad definida por el usuario. Esto incluye todas las restricciones de nivel de columna y nivel de tabla en CREATE TABLE, procedimientos almacenados y desencadenadores. Por ejemplo:

- Un plan gráfico debe tener hasta 3 macrosistemas.
- Cada macrosistema tiene 5 mesosistemas (MPFG, MPEV, MPE, MOFD, MEFD).
- Los test pedagógicos solo se pueden realizar según la planificación que tengan en el plan gráfico.
- Los test pedagógicos solo se pueden realizar después de haber elaborado el plan gráfico**.**

3.1.2 Normalización de la base de datos.

La normalización es una técnica para diseñar la estructura lógica de los datos de un sistema de información en el modelo relacional, desarrollada por E. F. Codd en 1972. Es una estrategia de diseño de abajo a arriba: se parte de los atributos y estos se van agrupando en relaciones (tablas) según su afinidad.

Las ventajas de la normalización son las siguientes:

- Evita anomalías en inserciones, modificaciones y borrados.
- Mejora la independencia de datos.
- No establece restricciones artificiales en la estructura de los datos.

Uno de los conceptos fundamentales en la normalización es el de dependencia funcional. Una **dependencia funcional** es una relación entre atributos de una misma relación (tabla). Si *X* e *Y* son atributos de la relación *R*, se dice que *Y* es funcionalmente dependiente de *Y* (se denota por $X \rightarrow Y$) si cada valor de *X* tiene asociado un solo valor de *Y*(*X* e *Y* pueden constar de uno o varios atributos). A *X* se le denomina determinante, ya que *X* determina el valor de *Y*. Se dice que el atributo *Y* es completamente dependiente de *X* si depende funcionalmente de *X* y no depende de ningún subconjunto de *X*.

La normalización se lleva a cabo en una serie de pasos. Cada paso corresponde a una forma normal que tiene unas propiedades. Conforme se va avanzando en la normalización, las relaciones tienen un formato más estricto (más fuerte) y, por lo tanto, son menos vulnerables a las anomalías de actualización. El modelo relacional sólo requiere un conjunto de relaciones en primera forma normal. Las restantes formas normales son opcionales. Sin embargo, para evitar las anomalías de actualización, es recomendable llegar al menos a la tercera forma normal [9].

Existen varios niveles de normalización:

- Primera Forma Normal (1FN).
- Segunda Forma Normal (2FN).
- Tercera Forma Normal (3FN).
- Forma Normal Boyce-Codd.
- Cuarta Forma Normal.
- Quinta Forma Normal o Forma Normal de Proyección-Unión.
- Forma Normal de Proyección-Unión Fuerte.
- Forma Normal de Proyección-Unión Extra Fuerte y Forma Normal de Clave de Dominio.

Primera Forma Normal.

Una relación R se encuentra en 1FN si y solo sí por cada renglón columna contiene valores atómicos. Esta forma normal se creó para eliminar los atributos multievaluados, los atributos compuestos y las combinaciones de ellos.

De forma más específica para que una relación se encuentre en primera forma normal debe de cumplir con las siguientes condiciones:

1. Las celdas de las tablas poseen valores simples y no se permiten grupos ni arreglos repetidos como valores, es decir, contienen un solo valor por cada celda.
2. Todos los ingresos en cualquier columna (atributo) deben ser del mismo tipo.
3. Cada columna debe tener un nombre único, el orden de las columnas en la tabla no es importante.

4. Dos filas o renglones de una misma tabla no deben ser idénticas, aunque el orden de las filas no es importante.

La Base de Datos del Sistema de planificación del Entrenamiento deportivo de Judo Femenino se encuentra en primera forma normal ya que todos los datos de las relaciones existentes son atómicos. Existen dos casos, uno con atributo multievaluado y otro con atributo compuesto en la tabla atleta.
La tabla atleta está compuesto por una serie de campos de valores atómicos, además de tener dos campos uno multievaluado y otro compuesto que son organización y dirección respectivamente. Para darle solución al problema planteado es necesario realizar la normalización, por lo tanto se crea una nueva relación que se llamará organización con el identificador nombre. Esto da lugar a una relación nueva de muchos a muchos entre la nueva relación (organización) y la tabla atleta, por lo que se crea una tabla nueva llamada atleta_organizacion que contiene las llaves primarias de ambas tablas.

Segunda Forma Normal

Una relación R está en segunda forma normal si y solo si está en 1FN y los atributos no primos dependen funcionalmente de la llave primaria.

Una relación se encuentra en segunda forma normal, cuando cumple con las reglas de la primera forma normal y todos sus atributos que no son claves (llaves) dependen por completo de la clave. De acuerdo con esta definición, cada tabla que tiene un atributo único como clave, está en segunda forma normal. Las relaciones que no están en segunda forma normal pueden sufrir anomalías cuando realizan actualizaciones.

Para pasar una relación en 1FN a 2FN hay que eliminar las dependencias parciales de la clave primaria. Para ello, se eliminan los atributos que son funcionalmente dependientes y se ponen en una nueva relación con una copia de su determinante (los atributos de la clave primaria de los que dependen).

La base de datos que se está analizando cumple con la 2FN al tener todas sus tablas en 1FN y las que tienen llaves compuestas, sus atributos no llaves son dependientes de la llave en su totalidad, es decir que no tiene atributos que dependan solamente de parte de la llave primaria.

Tercera Forma Normal

La tercera forma normal plantea que la relación debe estar en segunda forma normal y que cada atributo de la relación que no está contenido dentro de la llave primaria dependa solo de la llave primaria y no de ningún otro atributo. Es decir cada atributo no llave, depende directamente y no transitivamente, de la llave primaria.

Para pasar una relación de 2FN a 3FN hay que eliminar las dependencias transitivas. Para ello, se eliminan los atributos que dependen transitivamente y se ponen en una nueva relación con una copia de su determinante (el atributo o atributos no clave de los que dependen).

La base de datos del Sistema de Planificación del Entrenamiento Deportivo de Judo Femenino se encuentra en tercera forma normal ya que no tiene atributos no llaves en las tablas que dependan transitivamente de la llave primaria y además está en segunda forma normal.

3.1.3 Análisis de redundancia de información.

La redundancia es el almacenamiento repetitivo de datos en diferentes lugares de la base de datos o ficheros. Puede provocar problemas como: un incremento del trabajo, desperdicio de espacio de almacenamiento, inconsistencia de datos; es por ello que se deben crear base de datos con un mínimo de redundancia.

En la base de datos del Sistema de Planificación del Entrenamiento Deportivo de Judo Femenino se ha disminuido considerablemente la redundancia ya que no existen datos repetidos en ninguna de las tablas, los datos insertados nunca son iguales, debido a que son datos estadísticos y planes que se realizan en diferentes periodos de tiempos por lo que la información varia constantemente.

3.1.4 Análisis de la seguridad de la base de datos.

Los datos constituyen un recurso valioso que debe ser estrictamente controlado y gestionado al igual que cualquier otro recurso corporativo. El término seguridad hace referencia a la protección de la base de datos frente a accesos no autorizados, ya sean intencionados o accidentales, es decir, hace referencia a cualquier situación o suceso, provocados de forma intencionada o accidental, que pueda afectar adversamente a un sistema y consecuentemente a la organización. Las base de datos, en particular, deben tener un sistema de seguridad sólido para controlar las actividades que pueden realizarse y determinar qué información puede verse y cuál puede modificarse. Un sistema de seguridad sólido, asegura la protección de datos sin tener en cuenta cómo los usuarios obtienen el acceso a la base de datos.

3.1.4.1 Seguridad y control de acceso.

La seguridad de la base de datos está implementada en varios niveles:

El nivel 0 es el que se encarga de las máquinas (host) y los usuarios. Es decir, permite configurar que máquina/s y/o usuario/s se pueden conectar a la Base de Datos. Utilizando las opciones del fichero de configuración pg_hba.conf.
El nivel 1 es el que se encarga de los usuarios y las Base de Datos. Es el que permite configurar a qué Base de Datos se pueden conectar. Utilizando las opciones del fichero de configuración pg_ident.conf.
El nivel 2 es el que se encarga de las tablas. Es el que permite configurar a qué tablas pueden acceder. Utilizando los comandos GRANT para dar permisos y REVOKE para eliminar los permisos.

La seguridad debería conservarse por el mantenimiento de los usuarios de las base de datos. Dependiendo del tamaño de un sistema de base de datos y de la cantidad de trabajo requerido para administrar los usuarios de base de datos, el administrador de seguridad debería ser el único usuario con los privilegios requeridos para crear, alterar o borrar un usuario de la base de datos. Por otro lado, debería haber un número de administradores con privilegios para administrar los usuarios de base de datos. Sólo personas que gozan de confianza deberían tener privilegios totales para administrar los usuarios de base de datos.

Para la seguridad de la solución propuesta es necesario el acceso por otro usuario a la base de datos que no sea el que brinda PostgreSQL por defecto “postgres”. También se debe configurar, de acuerdo con las políticas de accesos permitidas, las conexiones locales y remotas mediante el archivo de configuración pg_hba.conf.

De inicio contiene algo como lo siguiente:

```
# TYPE DATABASE USER        CIDR-ADDRESS          METHOD
# IPv4 local connections:
#host    all    all          127.0.0.1/32          md5
# IPv6 local connections:
#host    all    all          : 1/128               md5
```

3.1.4.2 Backups y recuperación de los datos.

Existen dos formas de realizar el aseguramiento de los datos.

1.-Compactado del directorio de datos.

- Detener el servicio de PostgreSQL.
- Realizar la copia en *.rar o *.zip del directorio data de la carpeta de instalación de PostgreSQL.
- Iniciar el servicio de PostgreSQL.
- Copiar el *.rar o *.zip en algún soporte. (CD, disco……).

Para recuperar la copia: descomprimir en el directorio data de la carpeta de instalación de PostgreSQL.

Ventajas:

- Es rápido y sencillo.

Desventajas:

- Recomendable parar el servicio PostgreSQL.
- Incompatibilidad entre versiones.

2.- Volcado (dump).

Para el volcado se utiliza el comando pg_dump.

pg_dumpall –-d >todas.dump

Para establecer la recuperación se utiliza ps_restore.

pg_restore –e template1 <todas.dump

Ventajas:

- Se puede hacer con el servicio corriendo.
- Se puede hacer parcial.
- Compatibilidad entre versiones.

Desventajas:

- Gran lentitud con Base de Datos grandes.
- Consumo de recursos del sistema.

3.1.5 Trazabilidad de las acciones.

La trazabilidad puede realizarse sobre tablas o sobre procesos, aunque es válido decir que también se puede crear trazabilidad sobre errores o conflictos de la aplicación (logs). Los generadores de códigos no generan trazabilidades sobre los datos.

La trazabilidad es el conjunto de acciones y tecnologías que permiten rastrear desde el nacimiento del producto en el origen hasta la entrega final al consumidor. Consiste pues en la capacidad para reconstruir la historia, recorrido o aplicación de un determinado producto. Por ejemplo es cuando se insertan, actualizan, eliminan datos de la base de datos queda guardado automáticamente cual fue la tupla afectada, quién la modificó o eliminó y en el momento que realizo la acción. Con esto, se puede hacer una revisión y tener un control de las acciones realizadas sobre los datos de un registro específico o las acciones hechas por un usuario determinado sobre los datos, logrando así tener un historial de cada uno de los usuarios.

3.2 Validación funcional.

La validación funcional de la base de datos es el aspecto que define la veracidad del diseño obtenida pues a esta valoración teórica se le efectúan pruebas, como parte de estas, se generan cierta cantidad de datos para el llenado de la base de datos y luego se definen algunas consultas para la valoración del rendimiento y calidad del diseño.

Para el llenado voluminoso e inteligente de la BD se utilizó como herramienta EMS Data Generator For PostgreSQL 2005, siendo de gran utilidad para generar los datos de pruebas a varias tablas a la misma vez. Esta herramienta da la posibilidad de escoger las tablas a las cuales se quiere llenar. También brinda la posibilidad de generar los datos a través de consultas SQL o de escogerlos por medio de listas de valores. Una de las ventajas que tiene es que genera los datos que provienen de otras tablas para evitar errores de integridad referencial.

Se generaron 1000 tuplas para las tablas test_pedagogico y atleta, 5000 tulpas para la tabla parametro_cantidad y las tuplas correspondientes a las restantes tablas. Se definió para cada campo el rango de valores admitidos en dependencia de la cantidad de tuplas a insertar, cumpliendo en todo momento las pruebas de carga intensiva y escalabilidad propuesto. Para realizar la prueba se seleccionó la siguiente consulta donde muestra un reporte con el nombre de plan gráfico, el número del macrosistema, nombre y apellido del atleta, el identificador del test, la fecha de realización y el nombre del parámetro que se mide en este caso se utilizó la Barra Fija obteniendo un

promedio de la cantidad de barra fija realizada por cada atleta y organizada de mayor a menor.

```
SELECT parametro_cantidad.nombre_pg,
      parametro_cantidad.numero_macro,
      atleta.nombre,
      atleta.apellidos,
      test_pedagogico.id_test,
      test_pedagogico.fecha_inicio as fecha_realizado,
      parametro_cantidad.nombre_parametro,
      avg (parametro_cantidad.cantidad) as promedio_cantidad
FROM atleta
    INNER JOIN parametro_cantidad on (atleta.ci = parametro_cantidad.ci)
    INNER JOIN test_pedagogico ON (parametro_cantidad.id_test =
    test_pedagogico.id_test)
WHERE parametro_cantidad.nombre_parametro = 'Barra_Fija' and atleta.activo=true
GROUP BY test_pedagogico.fecha_inicio,
        parametro_cantidad.nombre_pg,
        parametro_cantidad.numero_macro,
        atleta.nombre,
        atleta.apellidos,
        test_pedagogico.id_test,
        parametro_cantidad.nombre_parametro
        order by avg(parametro_cantidad.cantidad) desc
```

En la ejecución de esta consulta se empleo un tiempo de 36 ms obteniendo como resultado 101 tuplas. Las pruebas realizadas a una base de datos nunca pueden tomarse como resultados reales, aunque dan un aproximado al mismo, dependiendo del grado de acercamiento que se utilice a los procesos de la vida real. Aún así, la implantación y utilización de una base de datos, está marcada por factores como, cantidad de usuarios a conectarse y el grado de concurrencia de las solicitudes hechas por los mismos, los cuales hay que tener en cuenta siempre.

3.3 Valoración de resultados.

Luego de concluir el diseño y la implementación de la base de datos de la aplicación informática “Sistema de planificación del Entrenamiento deportivo de Judo Femenino” se obtuvo los resultados siguientes:

- La base de datos cumple con los requisitos funcionales.
- Las relaciones entre las tablas fueron construidas adecuadamente.
- Correcta normalización de la base de datos.
- Las restricciones de integridad garantizan que los datos introducidos o modificados sean los correctos.
- La redundancia en la información se ha reducido en gran medida.

Conclusiones

En el desarrollo del capítulo se han analizado una serie de aspectos de gran importancia a tener en cuenta en el correcto funcionamiento de la base de datos. Se concluyó con un análisis de la consistencia de la misma, se definieron restricciones de integridad para garantizar la validez de los datos introducidos o modificados en la misma, la normalización y las distintas formas de seguridad que se pueden adoptar para el control de la información. Por otra parte, las pruebas realizadas a la base de datos permitieron chequear la integridad y seguridad de los datos y el correcto funcionamiento de la misma.

CONCLUSIONES

El presente trabajo una vez concluido cumple con las funcionalidades básicas que exige el cliente y podrá hacer uso de la aplicación como parte del sistema de planificación de entrenamiento deportivo de judo femenino, que actualmente se desarrolla de forma manual haciendo uso de documentos en Microsoft Word e imprimiendo en formato duro. En el mismo se reflejó el estudio realizado, el cual permitió obtener las base para la generación de una propuesta de solución que permitiera realizar los procesos de gestión de una forma rápida y eficiente.

Este sistema ha posibilitado garantizar la integridad de la base de datos por medio de restricciones que chequean la validez de los datos y definir aspectos de seguridad de la información. Se llegó a la conclusión de que la nueva base de datos de la aplicación informática “Sistema para la planificación de entrenamiento del judo femenino” cumple con el proceso de normalización hasta la tercera forma normal, posibilitando que en las tablas se encuentren datos que realmente pertenecen a ellas y que se disminuya la redundancia en la información. Se han tomado en cuenta, además, aspectos que permitan la reducción del tiempo de ejecución de las consultas de acuerdo a las necesidades de los usuarios.

Se considera que la idea a defender planteada al inicio ha sido debidamente demostrada aportando la centralización de toda la información referente a los procesos desarrollados por la Escuela Nacional de Judo Femenino así como la rapidez en la comunicación y en las búsquedas de información por parte del personal, disminuyendo así su carga laboral. Las metas propuestas fueron alcanzadas y se le dió cumplimiento al objetivo fundamental de esta investigación, análisis, diseño e implementación de la Base de Datos.

RECOMENDACIONES

Una vez concluido el presente trabajo se recomienda:

1. Que se realice el diseño e implementación de los restantes módulos, así como la documentación correspondiente.
2. Que se realice un estudio de las nuevas tendencias para realizar mejoras en el funcionamiento de la base de datos.
3. Que se implemente las funcionalidades administrativas en el servicio web como están diseñadas en la base de datos.
4. Que se ponga en práctica la utilización de este diseño de Base de Datos.

REFERENCIAS BIBLIOGRÁFICAS

1. Planificación de atletismo.2009 Disponible en **http://www.elatleta.com/**
2. Informática & Deportes.2007.Disponible en **http://www.x-medalist.com.ar/**
3. Desarrollo Web. Disponible en **http://www.desarrolloweb.com/directorio/base_de_datos/mysql/**
4. *Reference Manual*. 2007. Disponible en **http://dev.mysql.com/doc/refman/5.0/es/index.html**
5. Modelo de base de datos con ER/studio.2005 Disponible en **http://www.estudiagratis.com**
6. Anonimo.PgAdmin PostgreSQL.2009 Disponible en **http://www.pgadmin.org/**
7. EMS SQL Management Studio.2009 Disponible en **http://www.sqlmanager.net**
8. Anónimo. EMS SQL Query for PostgreSQL. 2007; Disponible en **http://www.yonimh.net/Foro/index.php?action=profile;u=50;sa=showPosts**
9. ANDRÉS, M. M. M. Tutorial de la asignatura de Base de Datos. publicado el: 26/04/2008 de 2007, última actualización: 26/04/2008 Disponible en: **http://www3.uji.es/~mmarques/f47/apun/node1.html.**

BIBLIOGRAFÍA

1. Base de datos: El formato electrónico y la calidad de la respuesta. Ciencias de la Información Vol. 32, No. 2, agosto, 2001. Disponible en **http://www.cinfo.cu/Userfiles/file/Cinfo/cinfo2001/v32n2a2001/basedatos.pdf**
2. Sitio oficial de MySQL 2008. Disponible en **http://www.mysql.com**
3. 2007 Wikilearning. Disponible en **http://www.wikilearning.com/tutoriales/ventajas_mysql/busqueda/1**
4. TuFunción es una web de Manuel Gutiérrez Heredia ©2007. Disponible en **http://www.tufuncion.com/indices-mysql**
5. 1999 - 2009 Todoexpertos.com. Disponible en **http://www.todoexpertos.com/categorias/tecnologia-e-internet/base-de-datos/oracle/respuestas/14706/vetajas-y-desventajas**
6. Ernesto C. Quiñones A. ernestoq@apesol.org .Pg. day 2007. Disponible en **http://www.postgresql.org.pe/articles/comparativa_dbms_libres.pdf**
7. Juan Pablo Gómez Gallego y Ing. Jorge Galves 17/9/2007. Disponible en **http://www.scribd.com/doc/297224/RUP**
8. BOGGS, W. and M. BOGGS. *UML with Rational Rose 2002*, 2002.
9. HANSEN, G. W. and J. H. HANSEN. *Diseño y Administración de Base de Datos*, 2007.
10. JACOBSON, I.; G. BOOCH*, et al. El proceso unificado de desarrollo de software*, 2000.
11. RIORDAN, R. M. *Aprenda programación en SQL Server 2000 Ya*, 2001.

GLOSARIO

Atributo: Es cada una de las cualidades, propiedades o características de un elemento.

Automatización Ejecución automática de tareas industriales, administrativas o científicas haciendo más ágil y efectivo el trabajo y ayudando al ser humano.

Backups (Copias de seguridad): Copia de datos de tal forma que estas copias adicionales puedan restaurar un sistema después de una pérdida de información.

Base de datos: Una base de datos es un conjunto de datos que pertenecen al mismo contexto, almacenados sistemáticamente para su uso posterior. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta.

Centralización: Hacer que varias cosas dependan de un poder central. Reunir varias cosas en un centro común.

Escalabilidad: La capacidad del sistema informático de cambiar su tamaño o configuración para adaptarse a las circunstancias cambiantes.

Interfaz: Una interfaz es la parte de un programa informático que permite a éste comunicarse con el usuario o con otras aplicaciones permitiendo el flujo de información.

Normalización: Proceso de reducción sobre una estructura de datos que procura aumentar la integridad, disminuir la redundancia y las dependencias funcionales de esa estructura.

Prototipo: Un prototipo también se puede referir a un objeto diseñado para una demostración de cualquier tipo.

Redundancia: Repetición de una información ya dada en el mensaje.

Rol: Papel desempeñado por las personas en la sociedad.

SQL (Structured Query Language) Es un lenguaje de acceso a las Base de Datos, permite especificar todas las operaciones sobre la base de datos como por ejemplo: Inserción, Borrado, Actualización. Utiliza características de álgebra y cálculo relacional permitiendo de esta forma realizar consultas a la base de datos de forma sencilla.

Herramienta CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) Las Herramientas CASE son diversas aplicaciones informáticas

destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero.

Rollback: Es el proceso automático que se ejecuta durante una transacción si un error ocurre la base de datos regresa al estado en que se encontraba desde la última transacción.

Servicios Web: Un servicio Web es una colección de protocolos y estándares que sirve para intercambiar datos entre aplicaciones.

Scripts: Un conjunto de comandos escritos en un lenguaje interpretado para automatizar ciertas tareas de aplicación.

Trazabilidad: Aptitud de reconstruir la historia, la utilización o la localización de un producto por medio de identificaciones registradas.

Triggers: Una acción que causa la invocación automática de un procedimiento, por ejemplo conservar la integridad referencial.

Tupla: Es una hilera o fila en una tabla.

Microsoft (acrónico de Microcomputer Software) Es una empresa de Estados Unidos, fundada por Bill Gates y Paul Allen. Dueña y productora de los sistemas operativos: Microsoft DOS y Microsoft Windows, que se utilizan en la mayoría de las computadoras del planeta.

Sistema operativo (SO) Es un conjunto de programas destinados a permitir la comunicación del usuario con un computador y gestionar sus recursos de una forma eficaz.

Archivos secuenciales Un archivo organizado secuencialmente es un conjunto de registros lógicamente relacionados cuya secuencia de acceso está determinada por su ordenamiento. Los registros deben ser grabados consecutivamente.

ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad)

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.

Anexo 1: Modelo lógico y Descripción de las tablas.

Reporte de entidad

Nombre de la entidad	Tipo de entidad	Llave Primaria	cantidad atributos
atleta	independiente	ci	15
atleta_organizacion	dependiente	ci, nombre	2
direccion	dependiente	ci	7
division	independiente	nombre_division	1
macrosistema	dependiente	numero_macro, nombre_pg	2
mesosistema	dependiente	nombre_meso, nombre_pg, numero_macro	16
microsistema	dependiente	numero_micro, nombre_pg, numero_macro, nombre_meso	12
nivel_alcanzado	independiente	nivel	1
organizacion	independiente	nombre	1
pagina	independiente	id_pagina	2
parametro	independiente	nombre_parametro	1
parametro_cantidad	dependiente	ci, id_test, nombre_pg, numero_macro, nombre_parametro	7
plan_grafico	independiente	nombre_pg	5
privilegio	independiente	tipo	1
privilegio_pagina	dependiente	tipo, id_pagina	2
rango	dependiente	nombre_division, nivel, nombre_parametro	5
test_pedagogico	dependiente	id_test, nombre_pg, numero_macro	5
usuario	dependiente	ci, tipo	6

Entidad 'atleta'

Nombre de la entidad	atleta
Tipo de entidad	independiente

Atributos

Llave	**Atributo/nombre de rol**	**Tipo de dato**	**Not null**	**Unique**	**Nota**
PK	ci	Bigint	YES	NO	
	nombre	Varchar(n)	YES	NO	
	apellidos	Varchar(n)	YES	NO	
	edad	Integer	YES	NO	
	edad_deportiva	Integer	NO	NO	
	tiempo_equipo	Integer	NO	NO	
	foto	Varchar	NO	NO	
	division	Varchar(n)	NO	NO	
	telefono	Varchar(n)	NO	NO	
	nombre_madre	Varchar(n)	NO	NO	
	nombre_padre	Varchar(n)	NO	NO	
	nivel_escolaridad	Varchar(n)	NO	NO	
	centro_estudio	Varchar(n)	NO	NO	
	activo	Boolean	YES	NO	
	numero_pasaporte	Integer	NO	NO	

Relaciones

Nombre de la Relación	**Tipo**	**Entidad Padre**	**Entidad Hija**	**Card.**
Relación5	Identifying	atleta	atleta_organizacion	1:N
Atleta_Direccion	Identifying	atleta	direccion	1:N
Relación44	Identifying	atleta	parametro_cantidad	1:N

Entidad 'atleta_organizacion'

Nombre de la entidad	atleta_organizacion
Tipo de entidad	dependiente

Atributos

Llave	**Atributo/nombre de rol**	**Tipo de dato**	**Not null**	**Unique**	**Nota**
PFK	ci	Bigint	YES	NO	
PFK	nombre	Varchar(n)	YES	NO	

Relaciones

Nombre de la Relación	**Tipo**	**Entidad Padre**	**Entidad Hija**	**Card.**
Relación5	Identifying	atleta	atleta_organizacion	1:N
Relación6	Identifying	organizacion	atleta_organizacion	1:N

Entidad 'direccion'

Nombre de la entidad	direccion
Tipo de entidad	dependiente

Atributos

Llave	**Atributo/nombre de rol**	**Tipo de dato**	**Not null**	**Unique**	**Nota**
PFK	ci	Bigint	YES	NO	
	provincia	Varchar(n)	YES	NO	
	municipio	Varchar(n)	YES	NO	
	localidad	Varchar(n)	NO	NO	
	calle	Varchar(n)	NO	NO	
	numero	Integer	NO	NO	
	entre_calles	Varchar(n)	NO	NO	

Relaciones

Nombre de la Relación	**Tipo**	**Entidad Padre**	**Entidad Hija**	**Card.**
Atleta_Direccion	Identifying	atleta	direccion	1:N

Entidad 'division'

Nombre de la entidad	division
Tipo de entidad	independiente

Atributos

Llave	**Atributo/nombre de rol**	**Tipo de dato**	**Not null**	**Unique**	**Nota**
PK	nombre_division	Varchar(n)	YES	NO	

Relaciones

Nombre de la Relación	**Tipo**	**Entidad Padre**	**Entidad Hija**	**Card.**
Division_Rango	Identifying	division	rango	1:N

Entidad 'macrosistema'

Nombre de la entidad	macrosistema
Tipo de entidad	dependiente

Atributos

Llave	**Atributo/nombre de rol**	**Tipo de dato**	**Not null**	**Unique**	**Nota**
PK	numero_macro	Integer	YES	NO	
PFK	nombre_pg	Varchar(n)	YES	NO	

Relaciones

Nombre de la Relación	**Tipo**	**Entidad Padre**	**Entidad Hija**	**Card.**
Macro_Meso	Identifying	macrosistema	mesosistema	1:N
Macro_TextPedagogico	Identifying	macrosistema	test_pedagogico	1:N
Relación46	Identifying	plan_grafico	macrosistema	1:N

Entidad 'mesosistema'

Nombre de la entidad	mesosistema
Tipo de entidad	dependiente

Atributos

Llave	Atributo/nombre de rol	Tipo de dato	Not null	Unique	Nota
PK	nombre_meso	Varchar(n)	YES	NO	
PFK	nombre_pg	Varchar(n)	YES	NO	
PFK	numero_macro	Integer	YES	NO	
	fecha_inicio	Date	YES	NO	
	fecha_fin	Date	YES	NO	
	periodo	Varchar(n)	NO	NO	
	sesiones	Integer	NO	NO	
	horas	Integer	NO	NO	
	minutos	Integer	NO	NO	
	combates	Integer	NO	NO	
	rtp_minimo	Real	NO	NO	
	rtp_maximo	Real	NO	NO	
	volumen_minimo	Real	NO	NO	
	volumen_maximo	Real	NO	NO	
	intensidad_minimo	Real	NO	NO	
	intensidad_maximo	Real	NO	NO	

Relaciones

Nombre de la Relación	Tipo	Entidad Padre	Entidad Hija	Card.
Macro_Meso	Identifying	macrosistema	mesosistema	1:N
Mesi_Micro	Identifying	mesosistema	microsistema	1:N

Entidad 'microsistema'

Nombre de la entidad	microsistema
Tipo de entidad	dependiente

Atributos

Llave	Atributo/nombre de rol	Tipo de dato	Not null	Unique	Nota
PK	numero_micro	Integer	YES	NO	
PFK	nombre_pg	Varchar(n)	YES	NO	
PFK	numero_macro	Integer	YES	NO	
PFK	nombre_meso	Varchar(n)	YES	NO	
	fecha_inicio	Date	YES	NO	
	fecha_fin	Date	YES	NO	
	test_pedagogico_micro	Boolean	NO	NO	
	prueba_medica_micro	Boolean	NO	NO	
	competencia_micro	Boolean	NO	NO	
	campo_entrenamiento_micro	Boolean	NO	NO	
	lmv	Varchar	NO	NO	
	mjs	Varchar	NO	NO	

Relaciones

Nombre de la Relación	Tipo	Entidad Padre	Entidad Hija	Card.
Mesi_Micro	Identifying	mesosistema	microsistema	1:N

Entidad 'nivel_alcanzado'

Nombre de la entidad	nivel_alcanzado
Tipo de entidad	independiente

Atributos

Llave	Atributo/nombre de rol	Tipo de dato	Not null	Unique	Nota
PK	nivel	Integer	YES	NO	

Relaciones

Nombre de la Relación	Tipo	Entidad Padre	Entidad Hija	Card.
NivelAlcanzado_Rango	Identifying	nivel_alcanzado	rango	1:N
Relación42	Non-identifying	nivel_alcanzado	parametro_cantidad	1:N

Entidad 'organizacion'

Nombre de la entidad	organizacion
Tipo de entidad	independiente

Atributos

Llave	Atributo/nombre de rol	Tipo de dato	Not null	Unique	Nota
PK	nombre	Varchar(n)	YES	NO	

Relaciones

Nombre de la Relación	Tipo	Entidad Padre	Entidad Hija	Card.
Relación6	Identifying	organizacion	atleta_organizacion	1:N

Entidad 'pagina'

Nombre de la entidad	pagina
Tipo de entidad	independiente

Atributos

Llave	Atributo/nombre de rol	Tipo de dato	Not null	Unique	Nota
PK	id_pagina	Serial	YES	NO	
	url	Varchar	YES	NO	

Relaciones

Nombre de la Relación	Tipo	Entidad Padre	Entidad Hija	Card.
Relación48	Identifying	pagina	privilegio_pagina	1:N

Entidad 'parametro'

Nombre de la entidad	parametro
Tipo de entidad	independiente

Atributos

Llave	Atributo/nombre de rol	Tipo de dato	Not null	Unique	Nota
PK	nombre_parametro	Varchar(n)	YES	NO	

Relaciones

Nombre de la Relación	Tipo	Entidad Padre	Entidad Hija	Card.
Parametro_Rango	Identifying	parametro	rango	1:N
Relación29	Identifying	parametro	parametro_cantidad	1:N

Entidad 'parametro_cantidad'

Nombre de la entidad	parametro_cantidad
Tipo de entidad	dependiente

Atributos

Llave	Atributo/nombre de rol	Tipo de dato	Not null	Unique	Nota
PFK	ci	Bigint	YES	NO	
PFK	id_test	Integer	YES	NO	
PFK	nombre_pg	Varchar(n)	YES	NO	
PFK	numero_macro	Integer	YES	NO	
PFK	nombre_parametro	Varchar(n)	YES	NO	
	cantidad	Real	NO	NO	
FK	nivel	Integer	NO	NO	

Relaciones

Nombre de la Relación	Tipo	Entidad Padre	Entidad Hija	Card.
Relación29	Identifying	parametro	parametro_cantidad	1:N
Relación42	Non-identifying	nivel_alcanzado	parametro_cantidad	1:N
Relación44	Identifying	atleta	parametro_cantidad	1:N
Relación45	Identifying	test_pedagogico	parametro_cantidad	1:N

Entidad 'plan_grafico'

Nombre de la entidad	plan_grafico
Tipo de entidad	independiente

Atributos

Llave	**Atributo/nombre de rol**	**Tipo de dato**	**Not null**	**Unique**	**Nota**
PK	nombre_pg	Varchar(n)	YES	NO	
	competencia_fundamental	Varchar(n)	YES	NO	
	lugar	Varchar(n)	YES	NO	
	pronostico	Varchar(n)	YES	NO	
	resultado	Varchar(n)	NO	NO	

Relaciones

Nombre de la Relación	**Tipo**	**Entidad Padre**	**Entidad Hija**	**Card.**
Relación46	Identifying	plan_grafico	macrosistema	1:N

Entidad 'privilegio'

Nombre de la entidad	privilegio
Tipo de entidad	independiente

Atributos

Llave	**Atributo/nombre de rol**	**Tipo de dato**	**Not null**	**Unique**	**Nota**
PK	tipo	Varchar	YES	NO	

Relaciones

Nombre de la Relación	**Tipo**	**Entidad Padre**	**Entidad Hija**	**Card.**
Privilegio_Usiario	Identifying	privilegio	usuario	1:N
Relación47	Identifying	privilegio	privilegio_pagina	1:N

Entidad 'privilegio_pagina'

Nombre de la entidad	privilegio_pagina
Tipo de entidad	dependiente

Atributos

Llave	**Atributo/nombre de rol**	**Tipo de dato**	**Not null**	**Unique**	**Nota**
PFK	tipo	Varchar	YES	NO	
PFK	id_pagina	Integer	YES	NO	

Relaciones

Nombre de la Relación	**Tipo**	**Entidad Padre**	**Entidad Hija**	**Card.**
Relación47	Identifying	privilegio	privilegio_pagina	1:N
Relación48	Identifying	pagina	privilegio_pagina	1:N

Entidad 'rango'

Nombre de la entidad	rango
Tipo de entidad	dependiente

Atributos

Llave	**Atributo/nombre de rol**	**Tipo de dato**	**Not null**	**Unique**	**Nota**
	minimo	Real	NO	NO	
	maximo	Real	NO	NO	
PFK	nombre_division	Varchar(n)	YES	NO	
PFK	nivel	Integer	YES	NO	
PFK	nombre_parametro	Varchar(n)	YES	NO	

Relaciones

Nombre de la Relación	**Tipo**	**Entidad Padre**	**Entidad Hija**	**Card.**
Division_Rango	Identifying	division	rango	1:N
Parametro_Rango	Identifying	parametro	rango	1:N
NivelAlcanzado_Rango	Identifying	nivel_alcanzado	rango	1:N

Entidad 'test_pedagogico'

Nombre de la entidad	test_pedagogico
Tipo de entidad	dependiente

Atributos

Llave	**Atributo/nombre de rol**	**Tipo de dato**	**Not null**	**Unique**	**Nota**
PK	id_test	Integer	YES	NO	
PFK	nombre_pg	Varchar(n)	YES	NO	
PFK	numero_macro	Integer	YES	NO	
	fecha_inicio	Date	YES	NO	
	fecha_fin	Date	YES	NO	

Relaciones

Nombre de la Relación	**Tipo**	**Entidad Padre**	**Entidad Hija**	**Card.**
Macro_TextPedagogico	Identifying	macrosistema	test_pedagogico	1:N
Relación45	Identifying	test_pedagogico	parametro_cantidad	1:N

Entidad 'usuario'

Nombre de la entidad	usuario
Tipo de entidad	dependiente

Atributos

Llave	**Atributo/nombre de rol**	**Tipo de dato**	**Not null**	**Unique**	**Nota**
PK	ci	Bigint	YES	NO	
PFK	tipo	Varchar	YES	NO	
	nombre	Varchar(n)	YES	NO	
	apellidos	Varchar(n)	YES	NO	
	usuario	Varchar(n)	YES	NO	
	contrasena	Varchar(n)	YES	NO	

Relaciones

Nombre de la Relación	Tipo	Entidad Padre	Entidad Hija	Card.
Privilegio_Usiario	Identifying	privilegio	usuario	1:N

Reporte de Atributos				
Nombre de atributo	**Nombre de la entidad**	**Tipo de dato**	**PK**	**FK**
activo	atleta	Boolean	NO	NO
apellidos	atleta	Varchar(n)	NO	NO
apellidos	usuario	Varchar(n)	NO	NO
calle	direccion	Varchar(n)	NO	NO
campo_entrenamiento_micro	microsistema	Boolean	NO	NO
cantidad	parametro_cantidad	Real	NO	NO
centro_estudio	atleta	Varchar(n)	NO	NO
ci	atleta_organizacion	Bigint	YES	YES
ci	atleta	Bigint	YES	NO
ci	parametro_cantidad	Bigint	YES	YES
ci	direccion	Bigint	YES	YES
ci	usuario	Bigint	YES	NO
combates	mesosistema	Integer	NO	NO
competencia_fundamental	plan_grafico	Varchar(n)	NO	NO
competencia_micro	microsistema	Boolean	NO	NO
contrasena	usuario	Varchar(n)	NO	NO
division	atleta	Varchar(n)	NO	NO
edad	atleta	Integer	NO	NO
edad_deportiva	atleta	Integer	NO	NO
entre_calles	direccion	Varchar(n)	NO	NO
fecha_fin	mesosistema	Date	NO	NO
fecha_fin	microsistema	Date	NO	NO
fecha_fin	test_pedagogico	Date	NO	NO
fecha_inicio	mesosistema	Date	NO	NO
fecha_inicio	microsistema	Date	NO	NO
fecha_inicio	test_pedagogico	Date	NO	NO
foto	atleta	Varchar	NO	NO
horas	mesosistema	Integer	NO	NO
id_pagina	pagina	Serial	YES	NO
id_pagina	privilegio_pagina	Integer	YES	YES
id_test	parametro_cantidad	Integer	YES	YES
id_test	test_pedagogico	Integer	YES	NO

intensidad_maximo	mesosistema	Real	NO	NO
intensidad_minimo	mesosistema	Real	NO	NO
lmv	microsistema	Varchar	NO	NO
localidad	direccion	Varchar(n)	NO	NO
lugar	plan_grafico	Varchar(n)	NO	NO
maximo	rango	Real	NO	NO
minimo	rango	Real	NO	NO
minutos	mesosistema	Integer	NO	NO
mjs	microsistema	Varchar	NO	NO
municipio	direccion	Varchar(n)	NO	NO
nivel	rango	Integer	YES	YES
nivel	nivel_alcanzado	Integer	YES	NO
nivel	parametro_cantidad	Integer	NO	YES
nivel_escolaridad	atleta	Varchar(n)	NO	NO
nombre	usuario	Varchar(n)	NO	NO
nombre	atleta	Varchar(n)	NO	NO
nombre	organizacion	Varchar(n)	YES	NO
nombre	atleta_organizacion	Varchar(n)	YES	YES
nombre_division	division	Varchar(n)	YES	NO
nombre_division	rango	Varchar(n)	YES	YES
nombre_madre	atleta	Varchar(n)	NO	NO
nombre_meso	microsistema	Varchar(n)	YES	YES
nombre_meso	mesosistema	Varchar(n)	YES	NO
nombre_padre	atleta	Varchar(n)	NO	NO
nombre_parametro	parametro	Varchar(n)	YES	NO
nombre_parametro	parametro_cantidad	Varchar(n)	YES	YES
nombre_parametro	rango	Varchar(n)	YES	YES
nombre_pg	test_pedagogico	Varchar(n)	YES	YES
nombre_pg	microsistema	Varchar(n)	YES	YES
nombre_pg	macrosistema	Varchar(n)	YES	YES
nombre_pg	plan_grafico	Varchar(n)	YES	NO
nombre_pg	parametro_cantidad	Varchar(n)	YES	YES
nombre_pg	mesosistema	Varchar(n)	YES	YES
numero	direccion	Integer	NO	NO
numero_macro	microsistema	Integer	YES	YES
numero_macro	mesosistema	Integer	YES	YES
numero_macro	parametro_cantidad	Integer	YES	YES

numero_macro	macrosistema	Integer	YES	NO
numero_macro	test_pedagogico	Integer	YES	YES
numero_micro	microsistema	Integer	YES	NO
numero_pasaporte	atleta	Integer	NO	NO
periodo	mesosistema	Varchar(n)	NO	NO
pronostico	plan_grafico	Varchar(n)	NO	NO
provincia	direccion	Varchar(n)	NO	NO
prueba_medica_micro	microsistema	Boolean	NO	NO
resultado	plan_grafico	Varchar(n)	NO	NO
rtp_maximo	mesosistema	Real	NO	NO
rtp_minimo	mesosistema	Real	NO	NO
sesiones	mesosistema	Integer	NO	NO
telefono	atleta	Varchar(n)	NO	NO
test_pedagogico_micro	microsistema	Boolean	NO	NO
tiempo_equipo	atleta	Integer	NO	NO
tipo	usuario	Varchar	YES	YES
tipo	privilegio	Varchar	YES	NO
tipo	privilegio_pagina	Varchar	YES	YES
url	pagina	Varchar	NO	NO
usuario	usuario	Varchar(n)	NO	NO
volumen_maximo	mesosistema	Real	NO	NO
volumen_minimo	mesosistema	Real	NO	NO

Atributo 'activo'

Nombre de atributo	activo	**Nombre de la entidad**	atleta
Tipo de dato	Boolean		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	YES		

Atributo 'apellidos'

Nombre de atributo	apellidos	**Nombre de la entidad**	atleta
Tipo de dato	Varchar(n) (30)		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	YES		

Atributo 'apellidos'

Nombre de atributo	apellidos	**Nombre de la entidad**	usuario
Tipo de dato	Varchar(n) (30)		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	YES		

Atributo 'calle'

Nombre de atributo	calle	**Nombre de la entidad**	direccion
Tipo de dato	Varchar(n) (20)		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'campo_entrenamiento_micro'

Nombre de atributo	campo_entrenamiento_micro	**Nombre de la entidad**	microsistema
Tipo de dato	Boolean		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'cantidad'

Nombre de atributo	cantidad	**Nombre de la entidad**	parametro_cantidad
Tipo de dato	Real		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'centro_estudio'

Nombre de atributo	centro_estudio	**Nombre de la entidad**	atleta
Tipo de dato	Varchar(n) (20)		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'ci'

Nombre de atributo	ci	**Nombre de la entidad**	atleta_organizacion
Tipo de dato	Bigint		
Llave Primaria	YES		
Llave Foránea	YES	**Entidad Padre**	atleta
Not null	YES		

Atributo 'ci'

Nombre de atributo	ci	**Nombre de la entidad**	atleta
Tipo de dato	Bigint		
Llave Primaria	YES		
Llave Foránea	NO		
Not null	YES		
Check	length(ci)=11		

Atributo 'ci'

Nombre de atributo	ci	**Nombre de la entidad**	parametro_cantidad
Tipo de dato	Bigint		
Llave Primaria	YES		
Llave Foránea	YES	**Entidad Padre**	atleta
Not null	YES		

Atributo 'ci'

Nombre de atributo	ci	**Nombre de la entidad**	direccion
Tipo de dato	Bigint		
Llave Primaria	YES		
Llave Foránea	YES	**Entidad Padre**	atleta
Not null	YES		

Atributo 'ci'

Nombre de atributo	ci	**Nombre de la entidad**	usuario
Tipo de dato	Bigint		
Llave Primaria	YES		
Llave Foránea	NO		
Not null	YES		

Atributo 'combates'

Nombre de atributo	combates	**Nombre de la entidad**	mesosistema
Tipo de dato	Integer		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'competencia_fundamental'

Nombre de atributo	competencia_fundamental	**Nombre de la entidad**	plan_grafico
Tipo de dato	Varchar(n) (20)		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	YES		

Atributo 'competencia_micro'

Nombre de atributo	competencia_micro	**Nombre de la entidad**	microsistema
Tipo de dato	Boolean		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'contrasena'

Nombre de atributo	contrasena	**Nombre de la entidad**	usuario
Tipo de dato	Varchar(n) (20)		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	YES		

Atributo 'division'

Nombre de atributo	division	**Nombre de la entidad**	atleta
Tipo de dato	Varchar(n) (5)		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'edad'

Nombre de atributo	edad	**Nombre de la entidad**	atleta
Tipo de dato	Integer		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	YES		
Check	edad>10 and edad<40		

Atributo 'edad_deportiva'

Nombre de atributo	edad_deportiva	**Nombre de la entidad**	atleta
Tipo de dato	Integer		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		
Check	edad_deportiva<30		

Atributo 'entre_calles'

Nombre de atributo	entre_calles	**Nombre de la entidad**	direccion
Tipo de dato	Varchar(n) (20)		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'fecha_fin'

Nombre de atributo	fecha_fin	**Nombre de la entidad**	mesosistema
Tipo de dato	Date		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	YES		
Check	fecha_fin>fecha_inicio		

Atributo 'fecha_fin'

Nombre de atributo	fecha_fin	**Nombre de la entidad**	microsistema
Tipo de dato	Date		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	YES		
Check	fecha_fin=fecha_inicio+6		

Atributo 'fecha_fin'

Nombre de atributo	fecha_fin	**Nombre de la entidad**	test_pedagogico
Tipo de dato	Date		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	YES		
Check	(fecha_fin>fecha_inicio)and(fecha_fin-fecha_inicio<6)		

Atributo 'fecha_inicio'

Nombre de atributo	fecha_inicio	**Nombre de la entidad**	mesosistema
Tipo de dato	Date		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	YES		

Atributo 'fecha_inicio'

Nombre de atributo	fecha_inicio	**Nombre de la entidad**	microsistema
Tipo de dato	Date		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	YES		

Atributo 'fecha_inicio'

Nombre de atributo	fecha_inicio	**Nombre de la entidad**	test_pedagogico
Tipo de dato	Date		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	YES		

Atributo 'foto'

Nombre de atributo	foto	**Nombre de la entidad**	atleta
Tipo de dato	Varchar		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'horas'

Nombre de atributo	horas	**Nombre de la entidad**	mesosistema
Tipo de dato	Integer		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'id_pagina'

Nombre de atributo	id_pagina	**Nombre de la entidad**	pagina
Tipo de dato	Serial		
Llave Primaria	YES		
Llave Foránea	NO		
Not null	YES		

Atributo 'id_pagina'

Nombre de atributo	id_pagina	**Nombre de la entidad**	privilegio_pagina
Tipo de dato	Integer		
Llave Primaria	YES		
Llave Foránea	YES	**Entidad Padre**	pagina
Not null	YES		

Atributo 'id_test'

Nombre de atributo	id_test	**Nombre de la entidad**	parametro_cantidad
Tipo de dato	Integer		
Llave Primaria	YES		
Llave Foránea	YES	**Entidad Padre**	test_pedagogico
Not null	YES		

Atributo 'id_test'

Nombre de atributo	id_test	**Nombre de la entidad**	test_pedagogico
Tipo de dato	Integer		
Llave Primaria	YES		
Llave Foránea	NO		
Not null	YES		

Atributo 'intensidad_maximo'

Nombre de atributo	intensidad_maximo	**Nombre de la entidad**	mesosistema
Tipo de dato	Real		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		
Check	intensidad_maximo>intensidad_minimo		

Atributo 'intensidad_minimo'

Nombre de atributo	intensidad_minimo	**Nombre de la entidad**	mesosistema
Tipo de dato	Real		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'lmv'

Nombre de atributo	lmv	**Nombre de la entidad**	microsistema
Tipo de dato	Varchar		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'localidad'

Nombre de atributo	localidad	**Nombre de la entidad**	direccion
Tipo de dato	Varchar(n) (30)		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'lugar'

Nombre de atributo	lugar	**Nombre de la entidad**	plan_grafico
Tipo de dato	Varchar(n) (20)		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	YES		

Atributo 'maximo'

Nombre de atributo	maximo	**Nombre de la entidad**	rango
Tipo de dato	Real		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'minimo'

Nombre de atributo	minimo	**Nombre de la entidad**	rango
Tipo de dato	Real		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'minutos'

Nombre de atributo	minutos	**Nombre de la entidad**	mesosistema
Tipo de dato	Integer		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'mjs'

Nombre de atributo	mjs	**Nombre de la entidad**	microsistema
Tipo de dato	Varchar		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'municipio'

Nombre de atributo	municipio	**Nombre de la entidad**	direccion
Tipo de dato	Varchar(n) (30)		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	YES		

Atributo 'nivel'

Nombre de atributo	nivel	**Nombre de la entidad**	rango
Tipo de dato	Integer		
Llave Primaria	YES		
Llave Foránea	YES	**Entidad Padre**	nivel_alcanzado
Not null	YES		

Atributo 'nivel'

Nombre de atributo	nivel	**Nombre de la entidad**	nivel_alcanzado
Tipo de dato	Integer		
Llave Primaria	YES		
Llave Foránea	NO		
Not null	YES		

Atributo 'nivel'

Nombre de atributo	nivel	**Nombre de la entidad**	parametro_cantidad
Tipo de dato	Integer		
Llave Primaria	NO		
Llave Foránea	YES	**Entidad Padre**	nivel_alcanzado
Not null	NO		

Atributo 'nivel_escolaridad'

Nombre de atributo	nivel_escolaridad	**Nombre de la entidad**	atleta
Tipo de dato	Varchar(n) (20)		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'nombre'

Nombre de atributo	nombre	**Nombre de la entidad**	usuario
Tipo de dato	Varchar(n) (20)		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	YES		

Atributo 'nombre'

Nombre de atributo	nombre	**Nombre de la entidad**	atleta
Tipo de dato	Varchar(n) (20)		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	YES		

Atributo 'nombre'

Nombre de atributo	nombre	**Nombre de la entidad**	organizacion
Tipo de dato	Varchar(n) (5)		
Llave Primaria	YES		
Llave Foránea	NO		
Not null	YES		

Atributo 'nombre'

Nombre de atributo	nombre	**Nombre de la entidad**	atleta_organizacion
Tipo de dato	Varchar(n) (5)		
Llave Primaria	YES		
Llave Foránea	YES	**Entidad Padre**	organizacion
Not null	YES		

Atributo 'nombre_division'

Nombre de atributo	nombre_division	**Nombre de la entidad**	division
Tipo de dato	Varchar(n) (20)		
Llave Primaria	YES		
Llave Foránea	NO		
Not null	YES		

Atributo 'nombre_division'

Nombre de atributo	nombre_division	**Nombre de la entidad**	rango
Tipo de dato	Varchar(n) (20)		
Llave Primaria	YES		
Llave Foránea	YES	**Entidad Padre**	division
Not null	YES		

Atributo 'nombre_madre'

Nombre de atributo	nombre_madre	**Nombre de la entidad**	atleta
Tipo de dato	Varchar(n) (20)		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'nombre_meso'

Nombre de atributo	nombre_meso	**Nombre de la entidad**	microsistema
Tipo de dato	Varchar(n) (7)		
Llave Primaria	YES		
Llave Foránea	YES	**Entidad Padre**	mesosistema
Not null	YES		

Atributo 'nombre_meso'

Nombre de atributo	nombre_meso	**Nombre de la entidad**	mesosistema
Tipo de dato	Varchar(n) (7)		
Llave Primaria	YES		
Llave Foránea	NO		
Not null	YES		

Atributo 'nombre_padre'

Nombre de atributo	nombre_padre	**Nombre de la entidad**	atleta
Tipo de dato	Varchar(n) (20)		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'nombre_parametro'

Nombre de atributo	nombre_parametro	**Nombre de la entidad**	parametro
Tipo de dato	Varchar(n) (20)		
Llave Primaria	YES		
Llave Foránea	NO		
Not null	YES		

Atributo 'nombre_parametro'

Nombre de atributo	nombre_parametro	**Nombre de la entidad**	parametro_cantidad
Tipo de dato	Varchar(n) (20)		
Llave Primaria	YES		
Llave Foránea	YES	**Entidad Padre**	parametro
Not null	YES		

Atributo 'nombre_parametro'

Nombre de atributo	nombre_parametro	**Nombre de la entidad**	rango
Tipo de dato	Varchar(n) (20)		
Llave Primaria	YES		
Llave Foránea	YES	**Entidad Padre**	parametro
Not null	YES		

Atributo 'nombre_pg'

Nombre de atributo	nombre_pg	**Nombre de la entidad**	test_pedagogico
Tipo de dato	Varchar(n) (30)		
Llave Primaria	YES		
Llave Foránea	YES	**Entidad Padre**	macrosistema
Not null	YES		

Atributo 'nombre_pg'

Nombre de atributo	nombre_pg	**Nombre de la entidad**	microsistema
Tipo de dato	Varchar(n) (30)		
Llave Primaria	YES		
Llave Foránea	YES	**Entidad Padre**	mesosistema
Not null	YES		

Atributo 'nombre_pg'

Nombre de atributo	nombre_pg	**Nombre de la entidad**	macrosistema
Tipo de dato	Varchar(n) (30)		
Llave Primaria	YES		
Llave Foránea	YES	**Entidad Padre**	plan_grafico
Not null	YES		

Atributo 'nombre_pg'

Nombre de atributo	nombre_pg	**Nombre de la entidad**	plan_grafico
Tipo de dato	Varchar(n) (30)		
Llave Primaria	YES		
Llave Foránea	NO		
Not null	YES		

Atributo 'nombre_pg'

Nombre de atributo	nombre_pg	**Nombre de la entidad**	parametro_cantidad
Tipo de dato	Varchar(n) (30)		
Llave Primaria	YES		
Llave Foránea	YES	**Entidad Padre**	test_pedagogico
Not null	YES		

Atributo 'nombre_pg'

Nombre de atributo	nombre_pg	**Nombre de la entidad**	mesosistema
Tipo de dato	Varchar(n) (30)		
Llave Primaria	YES		
Llave Foránea	YES	**Entidad Padre**	macrosistema
Not null	YES		

Atributo 'numero'

Nombre de atributo	numero	**Nombre de la entidad**	direccion
Tipo de dato	Integer		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'numero_macro'

Nombre de atributo	numero_macro	**Nombre de la entidad**	microsistema
Tipo de dato	Integer		
Llave Primaria	YES		
Llave Foránea	YES	**Entidad Padre**	mesosistema
Not null	YES		

Atributo 'numero_macro'

Nombre de atributo	numero_macro	**Nombre de la entidad**	mesosistema
Tipo de dato	Integer		
Llave Primaria	YES		
Llave Foránea	YES	**Entidad Padre**	macrosistema
Not null	YES		

Atributo 'numero_macro'

Nombre de atributo	numero_macro	**Nombre de la entidad**	parametro_cantidad
Tipo de dato	Integer		
Llave Primaria	YES		
Llave Foránea	YES	**Entidad Padre**	test_pedagogico
Not null	YES		

Atributo 'numero_macro'

Nombre de atributo	numero_macro	**Nombre de la entidad**	macrosistema
Tipo de dato	Integer		
Llave Primaria	YES		
Llave Foránea	NO		
Not null	YES		
Check	numero_macro<=2		

Atributo 'numero_macro'

Nombre de atributo	numero_macro	**Nombre de la entidad**	test_pedagogico
Tipo de dato	Integer		
Llave Primaria	YES		
Llave Foránea	YES	**Entidad Padre**	macrosistema
Not null	YES		

Atributo 'numero_micro'

Nombre de atributo	numero_micro	**Nombre de la entidad**	microsistema
Tipo de dato	Integer		
Llave Primaria	YES		
Llave Foránea	NO		
Not null	YES		

Atributo 'numero_pasaporte'

Nombre de atributo	numero_pasaporte	**Nombre de la entidad**	atleta
Tipo de dato	Integer		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'periodo'

Nombre de atributo	periodo	**Nombre de la entidad**	mesosistema
Tipo de dato	Varchar(n) (20)		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'pronostico'

Nombre de atributo	pronostico	**Nombre de la entidad**	plan_grafico
Tipo de dato	Varchar(n) (10)		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	YES		

Atributo 'provincia'

Nombre de atributo	provincia	**Nombre de la entidad**	direccion
Tipo de dato	Varchar(n) (20)		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	YES		

Atributo 'prueba_medica_micro'

Nombre de atributo	prueba_medica_micro	**Nombre de la entidad**	microsistema
Tipo de dato	Boolean		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'resultado'

Nombre de atributo	resultado	**Nombre de la entidad**	plan_grafico
Tipo de dato	Varchar(n) (10)		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'rtp_maximo'

Nombre de atributo	rtp_maximo	**Nombre de la entidad**	mesosistema
Tipo de dato	Real		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		
Check	rtp_maximo>rtp_minimo		

Atributo 'rtp_minimo'

Nombre de atributo	rtp_minimo	**Nombre de la entidad**	mesosistema
Tipo de dato	Real		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'sesiones'

Nombre de atributo	sesiones	**Nombre de la entidad**	mesosistema
Tipo de dato	Integer		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'telefono'

Nombre de atributo	telefono	**Nombre de la entidad**	atleta
Tipo de dato	Varchar(n) (10)		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'test_pedagogico_micro'

Nombre de atributo	test_pedagogico_micro	**Nombre de la entidad**	microsistema
Tipo de dato	Boolean		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Atributo 'tiempo_equipo'

Nombre de atributo	tiempo_equipo	**Nombre de la entidad**	atleta
Tipo de dato	Integer		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		
Check	tiempo_equipo<20		

Atributo 'tipo'

Nombre de atributo	tipo	**Nombre de la entidad**	usuario
Tipo de dato	Varchar		
Llave Primaria	YES		
Llave Foránea	YES	**Entidad Padre**	privilegio
Not null	YES		

Atributo 'tipo'

Nombre de atributo	tipo	**Nombre de la entidad**	privilegio
Tipo de dato	Varchar		
Llave Primaria	YES		
Llave Foránea	NO		
Not null	YES		

Atributo 'tipo'

Nombre de atributo	tipo	**Nombre de la entidad**	privilegio_pagina
Tipo de dato	Varchar		
Llave Primaria	YES		
Llave Foránea	YES	**Entidad Padre**	privilegio
Not null	YES		

Atributo 'url'

Nombre de atributo	url	**Nombre de la entidad**	pagina
Tipo de dato	Varchar		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	YES		

Atributo 'usuario'

Nombre de atributo	usuario	**Nombre de la entidad**	usuario
Tipo de dato	Varchar(n) (20)		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	YES		

Atributo 'volumen_maximo'

Nombre de atributo	volumen_maximo	**Nombre de la entidad**	mesosistema
Tipo de dato	Real		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		
Check	volumen_maximo>volumen_minimo		

Atributo 'volumen_minimo'

Nombre de atributo	volumen_minimo	**Nombre de la entidad**	mesosistema
Tipo de dato	Real		
Llave Primaria	NO		
Llave Foránea	NO		
Not null	NO		

Reporte de Relaciones

Nombre de la Relación	Tipo de Relación	Entidad Padre	Entidad Hija	Card.
Atleta_Direccion	Identifying	atleta	direccion	1:N
Division_Rango	Identifying	division	rango	1:N
Macro_Meso	Identifying	macrosistema	mesosistema	1:N
Macro_TextPedagogico	Identifying	macrosistema	test_pedagogico	1:N
Mesi_Micro	Identifying	mesosistema	microsistema	1:N
NivelAlcanzado_Rango	Identifying	nivel_alcanzado	rango	1:N
Parametro_Rango	Identifying	parametro	rango	1:N
Privilegio_Usiario	Identifying	privilegio	usuario	1:N
Relación29	Identifying	parametro	parametro_cantidad	1:N
Relación42	Non-identifying	nivel_alcanzado	parametro_cantidad	1:N
Relación44	Identifying	atleta	parametro_cantidad	1:N
Relación45	Identifying	test_pedagogico	parametro_cantidad	1:N
Relación46	Identifying	plan_grafico	macrosistema	1:N
Relación47	Identifying	privilegio	privilegio_pagina	1:N
Relación48	Identifying	pagina	privilegio_pagina	1:N
Relación5	Identifying	atleta	atleta_organizacion	1:N
Relación6	Identifying	organizacion	atleta_organizacion	1:N

Relación 'Atleta_Direccion'

Nombre de la Relación	Atleta_Direccion		
Tipo de Relación	identifying	**Cardinalidad**	1:N
Entidad Padre	atleta		
Entidad Hija	direccion		

Parcialidad

Padre	mandatory
Hijo	mandatory

Integridad Referencial

	Insert	Update	Delete
Padre	----	RESTRICT	RESTRICT
Hijo	NONE	NONE	----

Llaves

Tipo de llave	Padre Llave	Hijo Llave	Nombre Rol
Llave Primaria	ci	ci	----

Usuario-definición de variables

Nombre	Valor
Match Tipo	Default
Deferrable (Deferre constraint to the end of the transaction)	No
Deferred (Check constraint at the end of the transaction)	No

Relación 'Division_Rango'

Nombre de la Relación	Division_Rango		
Tipo de Relación	identifying	**Cardinalidad**	1:N
Entidad Padre	division		
Entidad Hija	rango		

Parcialidad

Padre	mandatory
Hijo	mandatory

Integridad Referencial

	Insert	Update	Delete
Padre	----	RESTRICT	RESTRICT
Hijo	NONE	NONE	----

Llaves

Tipo de llave	Padre Llave	Hijo Llave	Nombre Rol
Llave Primaria	nombre_division	nombre_division	----

Usuario-definición de variables

Nombre	Valor
Match Tipo	Default
Deferrable (Deferre constraint to the end of the transaction)	No
Deferred (Check constraint at the end of the transaction)	No

Relación 'Macro_Meso'

Nombre de la Relación	Macro_Meso		
Tipo de Relación	identifying	**Cardinalidad**	1:N
Entidad Padre	macrosistema		
Entidad Hija	mesosistema		

Parcialidad

Padre	mandatory
Hijo	mandatory

Integridad Referencial

	Insert	Update	Delete
Padre	----	CASCADE	RESTRICT
Hijo	NONE	NONE	----

Llaves

Tipo de llave	Padre Llave	Hijo Llave	Nombre Rol
Llave Primaria	numero_macro	numero_macro	----
	nombre_pg	nombre_pg	----

Usuario-definición de variables

Nombre	Valor
Match Tipo	Default
Deferrable (Deferre constraint to the end of the transaction)	No
Deferred (Check constraint at the end of the transaction)	No

Relación 'Macro_TextPedagogico'

Nombre de la Relación	Macro_TextPedagogico		
Tipo de Relación	identifying	**Cardinalidad**	1:N
Entidad Padre	macrosistema		
Entidad Hija	test_pedagogico		

Parcialidad

Padre	mandatory
Hijo	mandatory

Integridad Referencial

	Insert	**Update**	**Delete**
Padre	----	CASCADE	RESTRICT
Hijo	NONE	NONE	----

Llaves

Tipo de llave	**Padre Llave**	**Hijo Llave**	**Nombre Rol**
Llave Primaria	numero_macro	numero_macro	----
	nombre_pg	nombre_pg	----

Usuario-definición de variables

Nombre	**Valor**
Match Tipo	Default
Deferrable (Deferre constraint to the end of the transaction)	No
Deferred (Check constraint at the end of the transaction)	No

Relación 'Mesi_Micro'

Nombre de la Relación	Mesi_Micro		
Tipo de Relación	identifying	**Cardinalidad**	1:N
Entidad Padre	mesosistema		
Entidad Hija	microsistema		

Parcialidad

Padre	mandatory
Hijo	mandatory

Integridad Referencial

	Insert	Update	Delete
Padre	----	CASCADE	RESTRICT
Hijo	NONE	NONE	----

Llaves

Tipo de llave	Padre Llave	Hijo Llave	Nombre Rol
Llave Primaria	nombre_meso	nombre_meso	----
	nombre_pg	nombre_pg	----
	numero_macro	numero_macro	----

Usuario-definición de variables

Nombre	Valor
Match Tipo	Default
Deferrable (Deferre constraint to the end of the transaction)	No
Deferred (Check constraint at the end of the transaction)	No

Relación 'NivelAlcanzado_Rango'

Nombre de la Relación	NivelAlcanzado_Rango		
Tipo de Relación	identifying	**Cardinalidad**	1:N
Entidad Padre	nivel_alcanzado		
Entidad Hija	rango		

Parcialidad

Padre	mandatory
Hijo	mandatory

Integridad Referencial

	Insert	Update	Delete
Padre	----	RESTRICT	RESTRICT
Hijo	NONE	NONE	----

Llaves

Tipo de llave	Padre Llave	Hijo Llave	Nombre Rol
Llave Primaria	nivel	nivel	----

Usuario-definición de variables

Nombre	Valor
Match Tipo	Default
Deferrable (Deferre constraint to the end of the transaction)	No
Deferred (Check constraint at the end of the transaction)	No

Relación 'Parametro_Rango'

Nombre de la Relación	Parametro_Rango		
Tipo de Relación	identifying	**Cardinalidad**	1:N
Entidad Padre	parametro		
Entidad Hija	rango		

Parcialidad

Padre	mandatory
Hijo	mandatory

Integridad Referencial

	Insert	Update	Delete
Padre	----	RESTRICT	RESTRICT
Hijo	NONE	NONE	----

Llaves

Tipo de llave	Padre Llave	Hijo Llave	Nombre Rol
Llave Primaria	nombre_parametro	nombre_parametro	----

Usuario-definición de variables

Nombre	Valor
Match Tipo	Default
Deferrable (Deferre constraint to the end of the transaction)	No
Deferred (Check constraint at the end of the transaction)	No

Relación 'Privilegio_Usiario'

Nombre de la Relación	Privilegio_Usiario		
Tipo de Relación	identifying	**Cardinalidad**	1:N
Entidad Padre	privilegio		
Entidad Hija	usuario		

Parcialidad

Padre	mandatory
Hijo	mandatory

Integridad Referencial

	Insert	**Update**	**Delete**
Padre	----	RESTRICT	RESTRICT
Hijo	NONE	NONE	----

Llaves

Tipo de llave	**Padre Llave**	**Hijo Llave**	**Nombre Rol**
Llave Primaria	tipo	tipo	----

Usuario-definición de variables

Nombre	**Valor**
Match Tipo	Default
Deferrable (Deferre constraint to the end of the transaction)	No
Deferred (Check constraint at the end of the transaction)	No

Relación 'Relación29'

Nombre de la Relación	Relación29		
Tipo de Relación	identifying	**Cardinalidad**	1:N
Entidad Padre	parametro		
Entidad Hija	parametro_cantidad		

Parcialidad

Padre	mandatory
Hijo	mandatory

Integridad Referencial

	Insert	Update	Delete
Padre	----	RESTRICT	RESTRICT
Hijo	NONE	NONE	----

Llaves

Tipo de llave	Padre Llave	Hijo Llave	Nombre Rol
Llave Primaria	nombre_parametro	nombre_parametro	----

Usuario-definición de variables

Nombre	Valor
Match Tipo	Default
Deferrable (Deferre constraint to the end of the transaction)	No
Deferred (Check constraint at the end of the transaction)	No

Relación 'Relación42'

Nombre de la Relación	Relación42		
Tipo de Relación	non-identifying	**Cardinalidad**	1:N
Entidad Padre	nivel_alcanzado		
Entidad Hija	parametro_cantidad		

Parcialidad

Padre	mandatory
Hijo	mandatory

Integridad Referencial

	Insert	Update	Delete
Padre	----	RESTRICT	RESTRICT
Hijo	NONE	NONE	----

Llaves

Tipo de llave	Padre Llave	Hijo Llave	Nombre Rol
Llave Primaria	nivel	nivel	----

Usuario-definición de variables

Nombre	Valor
Match Tipo	Default
Deferrable (Deferre constraint to the end of the transaction)	No
Deferred (Check constraint at the end of the transaction)	No

Relación 'Relación44'

Nombre de la Relación	Relación44		
Tipo de Relación	identifying	Cardinalidad	1:N
Entidad Padre	atleta		
Entidad Hija	parametro_cantidad		

Parcialidad

Padre	mandatory
Hijo	mandatory

Integridad Referencial

	Insert	Update	Delete
Padre	----	RESTRICT	RESTRICT
Hijo	NONE	NONE	----

Llaves

Tipo de llave	Padre Llave	Hijo Llave	Nombre Rol
Llave Primaria	ci	ci	----

Usuario-definición de variables

Nombre	Valor
Match Tipo	Default
Deferrable (Deferre constraint to the end of the transaction)	No
Deferred (Check constraint at the end of the transaction)	No

Relación 'Relación45'

Nombre de la Relación	Relación45		
Tipo de Relación	identifying	**Cardinalidad**	1:N
Entidad Padre	test_pedagogico		
Entidad Hija	parametro_cantidad		

Parcialidad

Padre	mandatory
Hijo	mandatory

Integridad Referencial

	Insert	**Update**	**Delete**
Padre	----	CASCADE	RESTRICT
Hijo	NONE	NONE	----

Llaves

Tipo de llave	**Padre Llave**	**Hijo Llave**	**Nombre Rol**
Llave Primaria	id_test	id_test	----
	nombre_pg	nombre_pg	----
	numero_macro	numero_macro	----

Usuario-definición de variables

Nombre	**Valor**
Match Tipo	Default
Deferrable (Deferre constraint to the end of the transaction)	No
Deferred (Check constraint at the end of the transaction)	No

Relación 'Relación46'

Nombre de la Relación	Relación46		
Tipo de Relación	identifying	**Cardinalidad**	1:N
Entidad Padre	plan_grafico		
Entidad Hija	macrosistema		

Parcialidad

Padre	mandatory
Hijo	mandatory

Integridad Referencial

	Insert	**Update**	**Delete**
Padre	----	CASCADE	RESTRICT
Hijo	NONE	NONE	----

Llaves

Tipo de llave	**Padre Llave**	**Hijo Llave**	**Nombre Rol**
Llave Primaria	nombre_pg	nombre_pg	----

Usuario-definición de variables

Nombre	**Valor**
Match Tipo	Default
Deferrable (Deferre constraint to the end of the transaction)	No
Deferred (Check constraint at the end of the transaction)	No

Relación 'Relación47'

Nombre de la Relación	Relación47		
Tipo de Relación	identifying	**Cardinalidad**	1:N
Entidad Padre	privilegio		
Entidad Hija	privilegio_pagina		

Parcialidad

Padre	mandatory
Hijo	mandatory

Integridad Referencial

	Insert	**Update**	**Delete**
Padre	----	RESTRICT	RESTRICT
Hijo	NONE	NONE	----

Llaves

Tipo de llave	Padre Llave	Hijo Llave	Nombre Rol
Llave Primaria	tipo	tipo	----

Usuario-definición de variables

Nombre	Valor
Match Tipo	Default
Deferrable (Deferre constraint to the end of the transaction)	No
Deferred (Check constraint at the end of the transaction)	No

Relación 'Relación48'

Nombre de la Relación	Relación48		
Tipo de Relación	identifying	Cardinalidad	1:N
Entidad Padre	pagina		
Entidad Hija	privilegio_pagina		

Parcialidad

Padre	mandatory
Hijo	mandatory

Integridad Referencial

	Insert	Update	Delete
Padre	----	RESTRICT	RESTRICT
Hijo	NONE	NONE	----

Llaves

Tipo de llave	Padre Llave	Hijo Llave	Nombre Rol
Llave Primaria	id_pagina	id_pagina	----

Usuario-definición de variables

Nombre	Valor
Match Tipo	Default
Deferrable (Deferre constraint to the end of the transaction)	No
Deferred (Check constraint at the end of the transaction)	No

Relación 'Relación5'

Nombre de la Relación	Relación5		
Tipo de Relación	identifying	**Cardinalidad**	1:N
Entidad Padre	atleta		
Entidad Hija	atleta_organizacion		

Parcialidad

Padre	mandatory
Hijo	mandatory

Integridad Referencial

	Insert	**Update**	**Delete**
Padre	----	RESTRICT	RESTRICT
Hijo	NONE	NONE	----

Llaves

Tipo de llave	**Padre Llave**	**Hijo Llave**	**Nombre Rol**
Llave Primaria	ci	ci	----

Usuario-definición de variables

Nombre	**Valor**
Match Tipo	Default
Deferrable (Deferre constraint to the end of the transaction)	No
Deferred (Check constraint at the end of the transaction)	No

Relación 'Relación6'

Nombre de la Relación	Relación6		
Tipo de Relación	identifying	**Cardinalidad**	1:N
Entidad Padre	organizacion		
Entidad Hija	atleta_organizacion		

Parcialidad

Padre	mandatory
Hijo	mandatory

Integridad Referencial

	Insert	Update	Delete
Padre	----	RESTRICT	RESTRICT
Hijo	NONE	NONE	----

Llaves

Tipo de llave	Padre Llave	Hijo Llave	Nombre Rol
Llave Primaria	nombre	nombre	----

Usuario-definición de variables

Nombre	Valor
Match Tipo	Default
Deferrable (Deferre constraint to the end of the transaction)	No
Deferred (Check constraint at the end of the transaction)	No