Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

METODOLOGÍA PARA EL DISEÑO DE BASES DE DATOS EFICIENTES PARA PROYECTOS DE TAMAÑO MEDIANO DE LA FACULTAD DE INGENIERÍA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Christoper Emanuel Santisteban González
Asesorado por el Ing. Roberto Estuardo Ruiz Cruz

Guatemala, Junio de 2014

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

METODOLOGÍA PARA EL DISEÑO DE BASES DE DATOS EFICIENTES PARA PROYECTOS DE TAMAÑO MEDIANO DE LA FACULTAD DE INGENIERÍA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

CHRISTOPER EMANUEL SANTISTEBAN GONZÁLEZ
ASESORADO POR EL ING. ROBERTO ESTUARDO RUIZ CRUZ

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, JUNIO DE 2014

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	Ing. Alfredo Enrique Beber Aceituno
VOCAL II	Ing. Pedro Antonio Aguilar Polanco
VOCAL III	Inga. Elvia Miriam Ruballos Samayoa
VOCAL IV	Br. Walter Rafael Véliz Muñoz
VOCAL V	Br. Sergio Alejandro Donis Soto
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	Ing. Roberto Estuardo Ruiz Cruz
EXAMINADOR	Ing. Ricardo Morales Prado
EXAMINADOR	Ing. César Augusto Fernández Cáceres
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

METODOLOGÍA PARA EL DISEÑO DE BASES DE DATOS EFICIENTES PARA PROYECTOS DE TAMAÑO MEDIANO DE LA FACULTAD DE INGENIERÍA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha 3 de agosto de 2013.

Christoper Emanuel Santisteban González

Guatemala, 3 de Agosto de 2013

Ingeniero
Carlos Azurdia
Coordinador del Seminario de Investigación
Facultad de Ingeniería
Universidad de San Carlos de Guatemala

Estimado Ingeniero:

Habiendo revisado el trabajo de tesis titulado "METODOLOGÍA PARA EL DISEÑO DE BASES DE DATOS EFICIENTES PARA PROYECTOS DE TAMAÑO MEDIANO DE LA FACULTAD DE INGENIERÍA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA" presentado por el estudiante **CHRISTOPER EMANUEL SANTISTEBAN GONZALEZ** quien se identifica con número de carné **200511998**, manifiesto, que dicho trabajo ha llenado los requerimientos del programa dentro del cual se efectuó, por lo que lo doy como aprobado en mi calidad de asesor de tesis.

Atentamente,

Roberto Estuardo Ruiz Cruz
Ingeniero en Ciencias y Sistemas
colegiado 11,455

Ing. Estuardo Ruiz Cruz
Colegiado 11455
Asesor de tesis

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 14 de Agosto de 2013

Ingeniero
Marlon Antonio Pérez Turk
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Pérez:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación del estudiante **CHRISTOPER EMANUEL SANTISTEBAN GONZÁLEZ** con carné **2005-11998,** titulado: **"METODOLOGÍA PARA EL DISEÑO DE BASES DE DATOS EFICIENTES PARA PROYECTOS DE TAMAÑO MEDIANO DE LA FACULTAD DE INGENIERÍA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA",** y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

ESCUELA DE CIENCIAS Y SISTEMAS

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
TEL: 24767644

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, del trabajo de graduación ***"METODOLOGÍA PARA EL DISEÑO DE BASES DE DATOS EFICIENTES PARA PROYECTOS DE TAMAÑO MEDIANO DE LA FACULTAD DE INGENIERÍA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA"****, realizado por el estudiante CHRISTOPER EMANUEL SANTISTEBAN GONZÁLEZ, aprueba el presente trabajo y solicita la autorización del mismo.*

"ID Y ENSEÑAD A TODOS"

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION DE INGENIERIA EN CIENCIAS Y SISTEMAS

Ing. Marlon Antonio Pérez Türk
Director, Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 26 de mayo 2014

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

Ref.DTG.247-2014

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al trabajo de graduación titulado: **METODOLOGÍA PARA EL DISEÑO DE BASES DE DATOS EFICIENTES PARA PROYECTOS DE TAMAÑO MEDIANO DE LA FACULTAD DE INGENIERÍA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA,** presentado por el estudiante universitario: **Christoper Emanuel Santisteban González** y después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, se autoriza la impresión del mismo.

IMPRÍMASE.

Ing. Murphy Olympo Paiz Recinos
Decano

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERÍA

Guatemala, mayo de 2014

ACTO QUE DEDICO A:

Dios	Por dame la vida y las fuerzas necesarias para lograr mis metas y ser mi apoyo en todo momento.
Padre	Por estar conmigo y darme su apoyo en todo momento, ser mi inspiración y ejemplo, sus invaluables consejos y creer siempre en mí.
Hermanos	Por el apoyo, cariño y paciencia.
Novia	Por su paciencia, amor, dedicación, apoyo y las noches de desvelo.
Amigos	Por su ayuda y amistad a lo largo de la carrera.

AGRADECIMIENTOS A:

Ing. Estuardo Ruiz	Por su valioso tiempo y dedicación en la realización del presente trabajo de graduación.
Ing. Miguel Marín	Por su paciencia y apoyo en la realización de las prácticas.
Universidad de San Carlos de Guatemala	Por haber sido un segundo hogar todos estos años, haberme permitido adquirir los conocimientos y las habilidades para poder superarme personal y profesionalmente.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

GLOSARIO

Atomicidad La atomicidad es la propiedad que asegura que una operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias. Se dice que una operación es atómica cuando es imposible para otra parte de un sistema encontrar pasos intermedios. Si esta operación consiste en una serie de pasos, todos ellos ocurren o ninguno. Por ejemplo, en el caso de una transacción bancaria o se ejecuta tanto el depósito y el crédito o ninguna acción es realizada. Es una característica de los sistemas transaccionales.

Cardinalidad Es el número de elementos en un conjunto. Si dos conjuntos tienen el mismo número de elementos se dice que tienen la misma cardinalidad.

Clustering Se aplica a los conjuntos o conglomerados de computadoras construidos mediante la utilización de hardwares comunes y que se comportan como si fuesen una única computadora.

Concurrencia Juntarse o coincidir en un mismo lugar o tiempo diferentes personas, sucesos o cosas. Ejemplo: varios procesos compartiendo un mismo recurso al mismo tiempo.

DBMS	En inglés *Database Management System* (DBMS), es una agrupación de programas que sirven para definir, construir y manipular una base de datos.
Foregin Key	También llamada clave externa, foránea o simplemente “fk”, es uno o más campos de una tabla que hacen referencia al campo o campos de clave principal de otra tabla, una clave externa indica cómo están relacionadas las tablas. Los datos en los campos de clave externa y clave principal deben coincidir, aunque los nombres de los campos no sean los mismos.
Front-End	Es la parte del software que interactúa con el o los usuarios.
Integridad	La integridad de datos se refiere a los valores reales que se almacenan y se utilizan en las estructuras de datos de la aplicación. La aplicación debe ejercer un control deliberado sobre todos los procesos que utilicen los datos para garantizar la corrección permanente de la información.
Normalización	Es posible garantizar la integridad de los datos mediante la implementación escrupulosa de varios conceptos clave, como los que se incluyen a continuación:

- Normalizar datos.
- Definir reglas de negocio.
- Proporcionar integridad referencial.
- Validar los datos.

La normalización o estandarización es la redacción y aprobación de normas que se establecen para garantizar el acoplamiento de elementos construidos independientemente, así como garantizar el repuesto en caso de ser necesario, garantizar la calidad de los elementos fabricados, la seguridad de funcionamiento y trabajar con responsabilidad social.

La normalización es el proceso de elaborar, aplicar y mejorar las normas que se aplican a distintas actividades científicas, industriales o económicas con el fin de ordenarlas y mejorarlas. La asociación estadounidense para pruebas de materiales (ASTM) define la normalización como el proceso de formular y aplicar reglas para una aproximación ordenada a una actividad específica para el beneficio y con la cooperación de todos los involucrados.

Primary Key Una llave o clave primaria es un indicador único en una tabla.

Redundancia

La redundancia hace referencia al almacenamiento de la información varias veces en diferentes lugares. La redundancia de datos puede provocar problemas como:

- Incremento del trabajo: como un mismo dato está almacenado en dos o más lugares, esto hace que cuando se graben o actualicen los datos, deban hacerse en todos los lugares a la vez.
- Desperdicio de espacio de almacenamiento: ya que los mismos datos están almacenados en varios lugares distintos, ocupando así más bytes del medio de almacenamiento. Este problema es más evidente en grandes bases de datos.
- Inconsistencia de datos: esto sucede cuando los datos redundantes no son iguales entre sí. Esto puede suceder, por ejemplo, cuando se actualiza el dato en un lugar, pero el dato duplicado en otro lugar no es actualizado.

Repositorio

Un sitio centralizado donde se almacena y mantiene información.

SGBD

Véase DBMS.

SQL

Es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas. Una de sus características es el manejo del álgebra y el cálculo relacional que permiten efectuar consultas con el fin de recuperar de forma sencilla información de interés de bases de datos, así como hacer cambios en ella.

Tupla

Un registro (también llamado fila o tupla) representa un objeto único de datos implícitamente estructurados en una tabla. En términos simples, una tabla de una base de datos puede imaginarse formada de filas y columnas o campos. Cada fila de una tabla representa un conjunto de datos relacionados, y todas las filas de la misma tabla tienen la misma estructura. Un registro es un conjunto de campos que contienen los datos que pertenecen a una misma repetición de entidad. Se le asigna automáticamente un número consecutivo (número de registro) que en ocasiones es usado como índice aunque lo normal y práctico es asignarle a cada registro un campo clave para su búsqueda.

RESUMEN

Desde hace mucho tiempo se ha tenido la necesidad de almacenar información, ya sea en piedra, papel, un disquete, un disco duro, etc. El almacenar la información solo es una parte del objetivo, se debe poder recuperar la información, y aún más, compartir la información para que tenga sentido almacenarla. Las bases de datos surgieron para satisfacer esta necesidad, permiten almacenar grandes cantidades de información, y que ésta sea consultada de manera rápida, además, también permiten limitar el acceso a la información, otorgando la autoridad de decidir con quién se quiere compartir la misma.

Una base de datos es un repositorio que permite almacenar grandes cantidades de información, la cual está relacionada, organizada y tiene sentido; permite independencia física y lógica, es decir, que como los usuarios perciben la información no depende de cómo está se almacena, además no depende de su ubicación física; evitan tener copias de la misma información (redundancia) y permite el acceso a la información de manera concurrente, es decir, que varias personas pueden consultar la información sin que una se vea afectada por lo que la otra hace o dependa de este.

Una base de datos es una herramienta que brinda distintas características para el resguardo de la información, pero se debe saber que existen reglas y mejores prácticas para diseñar las estructuras (modelos) donde la información será almacenada, estas reglas se llaman reglas de integridad.

Los conceptos básicos de integridad en el modelo relacional son el de llave primaria, llave foránea, valores nulos y las reglas de integridad de relaciones y referencial. Una llave primaria es un conjunto de uno o más atributos de una tabla, que tomados colectivamente permiten identificar un registro como único, una llave foránea de una relación es un atributo que hace referencia a una llave primaria de otra relación; esto da pie a que una relación pueda tener varias llaves foráneas, un valor nulo es un valor que está fuera de la definición de cualquier dominio el cual permite dejar el valor del atributo latente.

Las otras dos reglas de integridad tienen que ver precisamente con los conceptos antes mencionados y son la integridad de relaciones, es decir, que ningún atributo que forme parte de una llave primaria puede aceptar valores nulos. La integridad referencial garantiza que una entidad (fila o registro) siempre se relaciona con otras entidades válidas, es decir, que existen en la base de datos. Implica que en todo momento dichos datos sean correctos, sin repeticiones innecesarias, datos perdidos ni relaciones mal resueltas.

Utilizando de base las reglas de integridad se pretende encontrar al menos tres factores que tengan una relación directa en el rendimiento de la base de datos, y en base a estos factores elaborar una metodología para la optimización del diseño de la base de datos. Las pruebas a realizar son:

- Rendimiento de los campos enteros autoincrementales en comparación de los campos enteros no autoincrementales.
- Rendimiento de los campos enteros apropiados autoincrementales en comparación de los campos enteros apropiados no autoincrementales.
- Rendimiento de los campos enteros apropiados autoincrementales con llaves foráneas en comparación de los campos enteros apropiados no autoincrementales sin llaves foráneas.

- Rendimiento de los campos enteros autoincrementales con índices *clustered* en comparación de los campos enteros autoincrementales con índices *non-clustered* y campos enteros autoincrementales sin índices.

En base a los resultados de las pruebas anteriormente descritas, se determina que una metodología propuesta debe tener la siguiente estructura, la cual puede tomarse como una guía rápida para aplicar a los modelos de bases de datos creados:

Primero: el análisis de los requerimientos, se debe conocer cuál es el objetivo de la base de datos, se definen detalles como: la carga, la estabilidad, los tiempos de respuesta aceptados, etc.

Segundo: como parte del diseño se debe tomar en cuenta las reglas de negocio, como el dominio de los datos, el papel que juegan los usuarios, las entidades a utilizar y las relaciones entre ellas.

Tercero: el utilizar la nomenclatura adecuada da visibilidad respecto a lo que se tiene y a lo que se quiere llegar, permite validar de manera gráfica las formas normales, las entidades, atributos y las relaciones entre ellas, además, se puede visualizar rápidamente los tipos de datos y si existe redundancia de datos.

Cuarto: determinar de acuerdo al objetivo (modelo de negocios) las entidades que tendrán la mayor carga y analizar si el uso de índices es factible. También se debe tomar en consideración aquellos campos a utilizar para relacionar la información y sobre los cuales se harán consultas. Utilizar los *constrains* y los índices *unique*.

OBJETIVOS

General

Definir una metodología que sirva de guía para el diseño de bases de datos eficientes para proyectos de tamaño mediano en la Facultad de Ingeniería en la Universidad de San Carlos de Guatemala.

Específicos

1. Definir al menos 3 factores que tengan una relación directa con el rendimiento de una base de datos, los cuales serán tomados de las 12 reglas de integridad de Codd y los principios de diseño y optimización de bases de datos.

2. Establecer por cada factor si mejora o empeora el rendimiento individualmente mediante pruebas de estrés.

3. Establecer si existe relación entre los factores que pueda perjudicar el rendimiento si se usan al mismo tiempo mediante pruebas de estrés.

INTRODUCCIÓN

Un motor de base de datos está pensado que cumpla funciones básicas como almacenar la información, mantenerla integra y que pueda ser accedida cuando sea necesaria.

Las bases de datos no están pensadas para almacenar pequeñas cantidades de información, sino manejar grandes volúmenes y por esta razón es que un diseño de base de datos eficiente es importante; un buen diseño es la diferencia entre una base de datos que responde en minutos y una que responde en segundos o incluso milisegundos. También se debe tomar en consideración que el análisis y diseño es una de las etapas más importantes, puesto que una vez hecho el análisis y diseño los cambios posteriores son más costosos.

El presente trabajo de graduación surge de la necesidad de materializar una metodología, que sirva de base para el diseño de bases de datos y hacer que este sea eficiente para los proyectos de bases de datos de la Escuela de Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala.

Se presentan a continuación una serie de hipótesis, las cuales mediante pruebas de rendimiento se negaran o afirmarán, y de esta manera se recopilará información para formar la metodología a proponer.

1. INTRODUCCIÓN A LOS SISTEMAS DE BASES DE DATOS

1.1. Necesidad de almacenar la información

Los sistemas de bases de datos surgieron ante la necesidad de almacenar registros de información que fueran perdurables y que además, estuvieran disponibles cuando fueran necesarios; el acceso a la información debía estar siempre disponible a un número no limitado de personas (aun cuando el acceso a la información se diera de manera concurrente) y a la vez debía ser restringido, limitando la cantidad de información accesible a cada usuario.

El uso de bases de datos permite centralizar en un repositorio la información, el medio de acceso a la información y los usuarios.

Figura 1. **Repositorio de información centralizada**

Fuente: elaboración propia, con base al programa Microsoft Word 2010.

1.2. ¿Qué son las bases de datos?

Una base de datos es un repositorio que permite almacenar grandes cantidades de información, la cual está relacionada, organizada y tiene sentido.

1.2.1. Definición

Una base de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

1.2.2. Características

- Independencia lógica y física de los datos: la manera como los usuarios perciben la información no es dependiente de como esta se almacena, permitiendo independencia lógica, es decir que, el ingreso, modificación, eliminación y/o consulta de la información es totalmente transparente al usuario, además, no depende de su ubicación física, la ubicación de los archivos de una base de datos puede ser alterada sin llegar a perjudicar el contenido de ella.

- Reduce la redundancia: copias de la misma información se mantienen al mínimo posible. La redundancia permite almacenar la misma información más de una vez, se usa normalmente para facilitar la lectura de la información, pero puede ser perjudicial, se debe llevar un control, para que todas las copias sean actualizadas al mismo tiempo, por ello, se debe evitar y minimizar la redundancia.

- Acceso concurrente: la concurrencia habla atender a varias personas a la vez, en términos de base de datos, la concurrencia permite el acceso a la información a múltiples usuarios y/o aplicaciones simultáneamente.

- Integridad de los datos: asegura que la información almacenada tiene sentido, no pierda su relación y esté disponible a las personas adecuadas.

- Seguridad de acceso y auditoría: controla el acceso de los usuarios a la información, concediendo o denegándolo.

- Respaldo y recuperación: es importante hacer notar que el único método de almacenamiento permanente es aquel en el que se guardan copias exactas de una misma información en medios no volátiles, por ello, una característica importante de una base de datos es el respaldo, una característica que permite hacer copias idénticas de la base de datos y que estas posteriormente, puedan ser restauradas en caso de un desastre.

1.2.3. Tipos de bases de datos

Las bases de datos pueden clasificarse de varias maneras, de acuerdo al contexto que se esté manejando, la utilidad de las mismas o las necesidades que satisfagan:

1.2.4. Según la variabilidad de los datos almacenados

- Bases de datos estáticas: son bases de datos de solo lectura, utilizadas primordialmente para almacenar datos históricos que posteriormente se pueden utilizar para estudiar el comportamiento de un conjunto de datos a

través del tiempo, realizar proyecciones, tomar decisiones y realizar análisis de datos para inteligencia empresarial.

- Bases de datos dinámicas: estas son bases de datos donde la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización, borrado y adición de datos, además de las operaciones fundamentales de consulta. Un ejemplo de esto puede ser la base de datos utilizada en un sistema de información de un supermercado, una farmacia, un videoclub o una empresa.

1.2.5. Según el contenido

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

- Bases de datos de texto completo: almacenan las fuentes primarias, como por ejemplo, todo el contenido de todas las ediciones de una colección de revistas científicas.

- Directorios: un ejemplo son las guías telefónicas en formato electrónico.

- Bases de datos o "bibliotecas" de información química o biológica.

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

  - Las que almacenan secuencias de nucleótidos o proteínas.
  - Las bases de datos de rutas metabólicas.
  - Bases de datos de estructura, comprende los registros de datos experimentales, por ejemplo: sobre estructuras 3D de biomoléculas.
  - Bases de datos clínicas.
  - Bases de datos bibliográficas (biológicas, químicas, médicas y de otros campos).

1.2.6. Modelos de bases de datos

Además de la clasificación por la función de las bases de datos, también se pueden clasificar de acuerdo a su modelo de administración de datos.

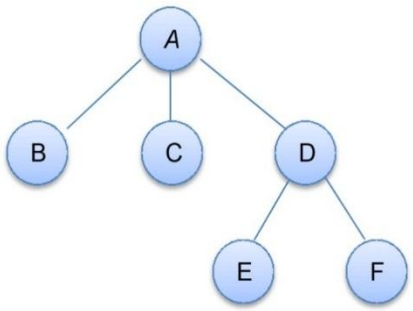
Un modelo de datos es básicamente una descripción de algo conocido como contenedor de datos, así como de los métodos para almacenar y recuperar la información. Los modelos de datos no son cosas físicas: son abstracciones que permiten la implementación de un sistema eficiente de base de datos; por lo general se refieren a algoritmos, y conceptos matemáticos.

- Bases de datos jerárquicas: en este modelo los datos se organizan en una forma similar a un árbol (visto al revés), en donde un nodo padre de información puede tener varios hijos. El nodo que no tiene padres es llamado raíz, y a los nodos que no tienen hijos se los conoce como hojas.

Las bases de datos jerárquicas son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos permitiendo crear estructuras estables y de gran rendimiento.

Una de las principales limitaciones de este modelo es su incapacidad de representar eficientemente la redundancia de datos.

Figura 2. **Modelo de base de datos jerárquico**

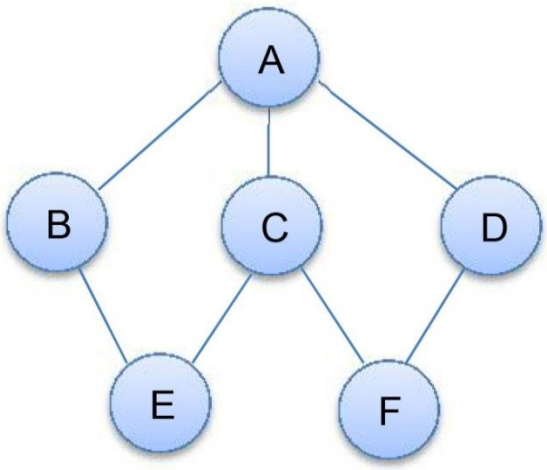


Fuente: elaboración propia, con base al programa Microsoft Word 2010.

- Base de datos de red: este es un modelo ligeramente distinto del jerárquico; su diferencia fundamental es la modificación del concepto de nodo: se permite que un mismo nodo tenga varios padres (posibilidad no permitida en el modelo jerárquico).

Fue una gran mejora con respecto al modelo jerárquico, ya que ofrecía una solución eficiente al problema de redundancia de datos; pero, aun así, la dificultad que significa administrar la información en una base de datos de red ha significado que sea un modelo utilizado en su mayoría por programadores más que por usuarios finales.

Figura 3. **Modelo de base de datos en red**



Fuente: elaboración propia, con base al programa Microsoft Word 2010.

- Bases de datos transaccionales: son bases de datos cuyo único fin es el envío y recepción de datos a grandes velocidades, estas bases son muy poco comunes y están dirigidas por lo general al entorno de análisis de calidad, datos de producción e industrial, es importante entender que su fin único es recolectar y recuperar los datos a la mayor velocidad posible, por lo tanto la redundancia y duplicación de información no es un problema como con las demás bases de datos, por lo general para poderlas aprovechar al máximo permiten algún tipo de conectividad a bases de datos relacionales.

Un ejemplo habitual de transacción es el traspaso de una cantidad de dinero entre cuentas bancarias. Normalmente se realiza mediante dos operaciones distintas, una en la que se decrementa el saldo de la cuenta origen y otra en la que incrementa el saldo de la cuenta destino. Para garantizar la atomicidad del sistema (es decir, se hacen las dos operaciones o ninguna) las dos operaciones

deben ser atómicas, ya que el sistema debe garantizar que, bajo cualquier circunstancia (incluso una caída del sistema), el resultado final es que, o bien se han realizado las dos operaciones, o bien no se ha realizado ninguna.

Si un sistema de base de datos transaccional pierde la mitad de la energía eléctrica a mitad de una transacción, está parcialmente terminada se retrotraerá y la base de datos será restaurada al estado en que se encontraba antes de la transacción iniciada.

Imagine que un *front-end* de la aplicación es el envío de un pedido de un cliente a un sistema de base de datos. Este envía la solicitud al producto al cliente y restar el producto del inventario. El mismo está a punto de enviar la solicitud de crear una factura para el cliente y de repente se bloquea.

Una base de datos transaccional puede revertir la transacción de forma parcial.

- Bases de datos relacionales: este es el modelo utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos. Su idea fundamental es el uso de relaciones. Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados tuplas. Pese a que esta es la teoría de las bases de datos relacionales creadas por Codd, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar. Esto es pensando en cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla).

En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser recuperada o almacenada mediante consultas que ofrecen una amplia flexibilidad y poder para administrar la información.

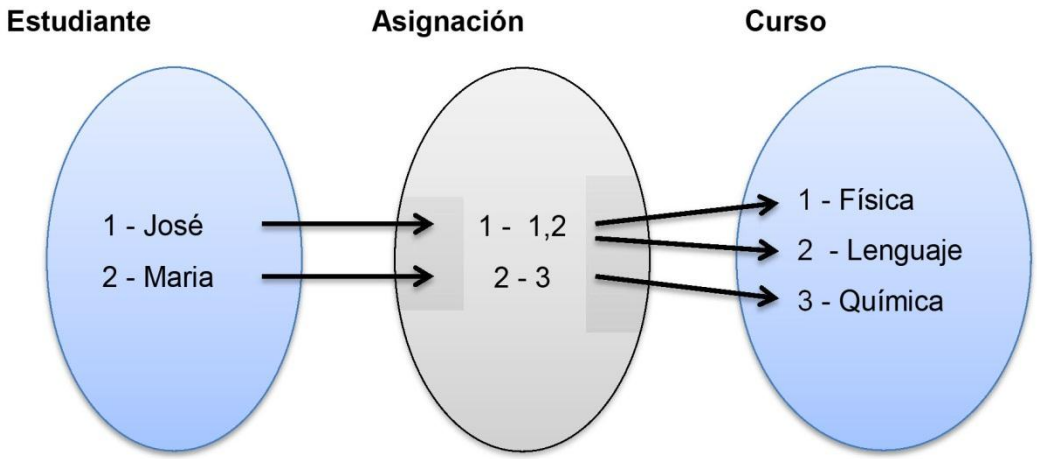
El lenguaje más habitual para construir las consultas a bases de datos relacionales es SQL, *Structured Query Language* o Lenguaje Estructurado de Consultas, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

Durante su diseño, una base de datos relacional pasa por un proceso al que se le conoce como normalización de una base de datos.

Durante los años 80 la aparición de dBASE produjo una revolución en los lenguajes de programación y sistemas de administración de datos. Aunque nunca debe olvidarse que dBase no utilizaba SQL como lenguaje base para su gestión.

Es una base de datos que utiliza el modelo relacional. Su idea fundamental es el uso de relaciones. Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados registros o tuplas.

Figura 4. **Relaciones en el modelo de base de datos relacional**



Fuente: elaboración propia, con base al programa Microsoft Word 2010.

- Bases de datos multidimensionales: una base de datos multidimensional, es aquella que almacena sus datos con varias dimensiones, es decir que en vez de un valor, se encuentran varios dependiendo de los ejes definidos o una base de datos de estructura basada en dimensiones orientada a consultas complejas y alto rendimiento. Puede utilizar un SGBDR en estrella (Base de datos Multidimensional a nivel lógico) o SGBDM (Base de datos Multidimensional a niveles lógico y físico o Base de datos Multidimensional Pura).

En una base de datos multidimensional, la información se representa como matrices multidimensionales, cuadros de múltiples entradas o funciones de varias variables sobre conjuntos finitos. Cada una de estas matrices se denomina cubo.

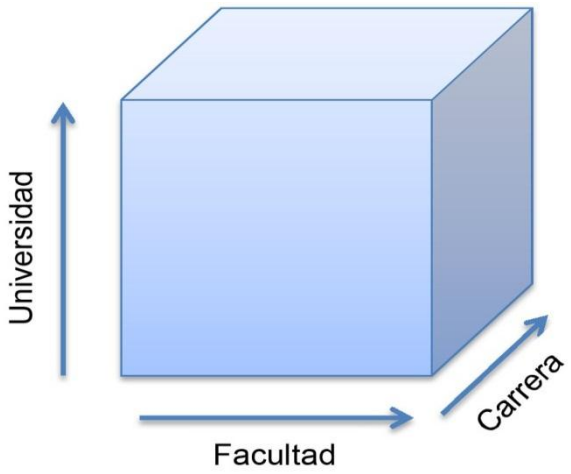
La estructura básica es un hipercubo compuesto por dos elementos: un conjunto de dimensiones y una función que mapea coordenadas formadas por valores de cada una de las dimensiones en tuplas o booleanos. Una dimensión

es un nombre con un dominio asociado.

Los cubos de información o cubos OLAP funcionan como los cubos de rompecabezas en los juegos, en el juego se trata de armar los colores y en el data *warehouse* se trata de organizar los datos por tablas o relaciones; los primeros (el juego) tienen 3 dimensiones, los cubos OLAP tienen un número indefinido de dimensiones, razón por la cual también reciben el nombre de hipercubos.

Eso facilita el manejo de grandes cantidades de datos dentro de empresas, dándole a esto una amplia aplicación dentro de varias áreas y diferentes campos del conocimiento humano.

Figura 5. **Modelo de base de datos multidimensional**



Fuente: elaboración propia, con base al programa Microsoft Word 2010.

- Bases de datos orientadas a objetos: este modelo, bastante reciente, y propio de los modelos informáticos orientados a objetos, trata de

almacenar en la base de datos los objetos completos (estado y comportamiento).

Una base de datos orientada a objetos es una base de datos que incorpora todos los conceptos importantes del paradigma de objetos:

- Encapsulación: propiedad que permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos.
- Herencia: propiedad a través de la cual los objetos heredan comportamiento dentro de una jerarquía de clases.
- Polimorfismo: propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos.

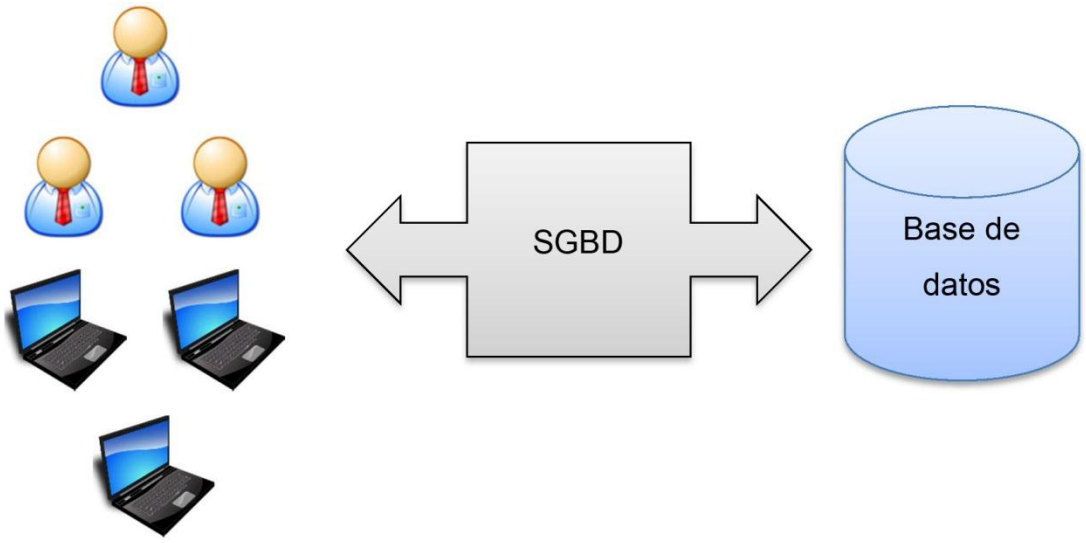
En bases de datos orientadas a objetos, los usuarios pueden definir operaciones sobre los datos como parte de la definición de la base de datos. Una operación (llamada función) se especifica en dos partes. La interfaz (o signatura) de una operación incluye el nombre de la operación y los tipos de datos de sus argumentos (o parámetros). La implementación (o método) de la operación se especifica separadamente y puede modificarse sin afectar la interfaz. Los programas de aplicación de los usuarios pueden operar sobre los datos invocando a dichas operaciones a través de sus nombres y argumentos, sea cual sea la forma en la que se han implementado. Esto podría denominarse independencia entre programas y operaciones.

1.3. Sistemas manejadores de bases de datos

Un Sistema Gestor de Bases de Datos (SGBD) o DBMS (*DataBase Management System*) es una colección de programas cuyo objetivo es servir de

interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos.

Figura 6. **Función de un sistema gestor de base de datos**



Fuente: elaboración propia, con base al programa Microsoft Word 2010.

Las características de un SGBD son:

- Abstracción de la información. Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción.

- Independencia. La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.

- Redundancia mínima. Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. De entrada, lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.

- Consistencia. En aquellos casos en los que no se ha logrado esta redundancia nula, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.

- Seguridad. La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra segura frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado pero despistado. Normalmente, los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.

- Integridad. Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.

- Respaldo y recuperación. Los SGBD deben proporcionar una forma eficiente de realizar copias de respaldo de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.

- Control de la concurrencia. En la mayoría de entornos (excepto quizás el doméstico), lo más habitual es que sean muchas las personas que acceden a una base de datos, bien para recuperar información, bien para almacenarla. Y es también frecuente que dichos accesos se realicen de forma simultánea. Así pues, un SGBD debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.

1.3.1.1. Ventajas de usar un SGBD

- Proveen facilidades para la manipulación de grandes volúmenes de datos (ver objetivos). Entre estas:

    - Simplifican la programación de equipos de consistencia.
    - Manejando las políticas de respaldo adecuadas, garantizan que los cambios de la base serán siempre consistentes sin importar si hay errores correctamente, etc.
    - Organizan los datos con un impacto mínimo en el código de los programas.
    - Disminuyen drásticamente los tiempos de desarrollo y aumentan la calidad del sistema desarrollado si son bien explotados por los desarrolladores.
    - Usualmente, proveen interfaces y lenguajes de consulta que simplifican la recuperación de los datos.

1.3.1.2. Niveles de un SGBD

Los SBD pueden ser estudiados desde tres niveles distintos:

- Nivel interno o físico: es el nivel real de los datos almacenados, considerado como el nivel inferior de abstracción, es decir cómo se almacenan los datos, ya sea en registros, o de otra forma. Este nivel es usado por muy pocas personas que deben estar cualificadas para ello, lleva asociada una representación de los datos, denominados Esquema Físico.

- Nivel conceptual o lógico: es el correspondiente a una visión de la base de datos desde el punto de vista del mundo real. Es decir, se trata con la entidad u objeto representado, sin importar como está representado o almacenado. Este nivel lleva asociado el Esquema Conceptual.

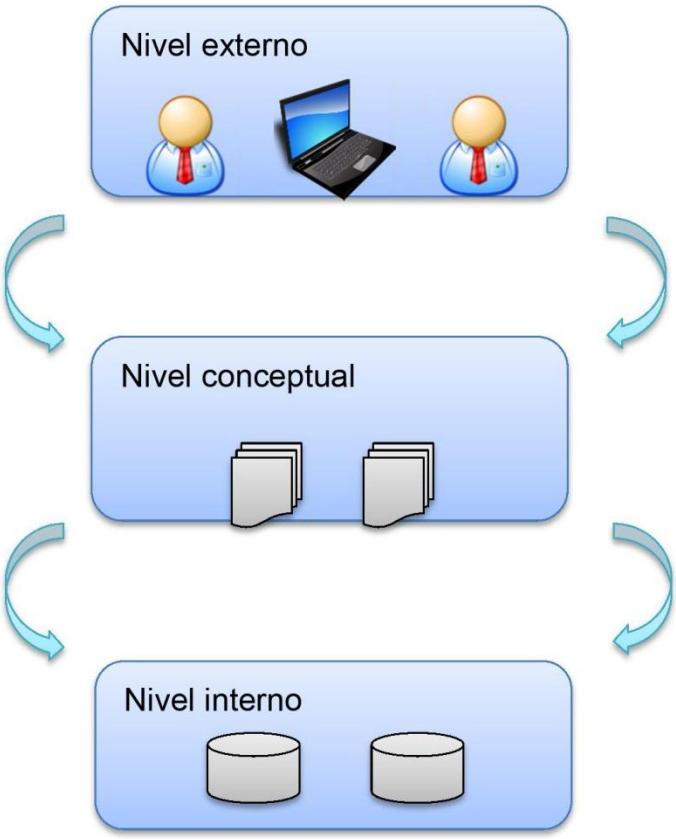
- Nivel externo o visión: son partes del esquema conceptual. El nivel conceptual presenta toda la base de datos, mientras que los usuarios por lo general solo tienen acceso a pequeñas parcelas de esta. El nivel visión es el encargado de dividir estas parcelas. Un ejemplo sería el caso del empleado que no tiene por qué tener acceso al sueldo de sus compañeros. El esquema asociado a este nivel es el Esquema de Visión.

Los tres niveles indicados, componen lo que se conoce como arquitectura de base de datos a tres niveles. En ocasiones puede encontrarse el nivel conceptual subdivido en dos niveles, conceptual y lógico. El primero de ellos corresponde a la visión del sistema global desde un punto de vista organizativo independiente, no informático, el segundo correspondería a la visión de la base de datos expresada en términos del sistema que se va a implantar con medios informáticos.

A menudo el nivel físico no es facilitado por muchos DBMS, esto es, no permiten al usuario elegir como se almacenan sus datos y vienen con una forma estándar de almacenamiento y manipulación de los datos.

La arquitectura a tres niveles se puede representar como sigue:

Figura 7. **Niveles de abstracción de un SGBD**



Fuente: elaboración propia, con base al programa Microsoft Word 2010.

1.3.2. SGBD comerciales

Actualmente en el mercado existen Sistemas Gestores de Bases de Datos de diferentes tipos, como libres y gratuitos, no libres (requieren una licencia pagada), libres pero no gratuitos, a continuación se hace una recopilación de ellos:

1.3.2.1. Sistemas libres

- PostgreSQL (http://www.postgresql.org Postgresql) Licencia BSD
- Firebird basada en la versión 6 de InterBase, Initial Developer's PUBLIC LICENSE Version 1.0.
- SQLite (http://www.sqlite.org SQLite) Licencia Dominio Público
- DB2 Express-C (http://www.ibm.com/software/data/db2/express/)
- Apache Derby (http://db.apache.org/derby/)
- MariaDB (http://mariadb.org/)
- MySQL (http://dev.mysql.com/)
- Drizzle (http://www.drizzle.org/)

1.3.2.2. Sistemas no libres

- MySQL: Licencia Dual, depende del uso. No se sabe hasta cuándo permanecerá así, ya que ha sido comprada por Oracle. Sin embargo, existen 2 versiones: una gratuita que sería equivalente a la edición express SQL server de Microsoft Windows, y otra más completa de pago.
- Advantage Database
- dBase
- FileMaker
- Fox Pro

- gsBase
- IBM DB2: Universal Database (DB2 UDB)
- IBM Informix
- Interbase de CodeGear, filial de Borland
- MAGIC
- Microsoft Access
- Microsoft SQL Server
- NexusDB
- Open Access
- Oracle
- Paradox
- PervasiveSQL
- Progress (DBMS)
- Sybase ASE
- Sybase ASA
- Sybase IQ
- WindowBase
- IBM IMS Base de Datos Jerárquica
- CA-IDMS

1.3.2.3. Sistemas no libres y gratuitos

- Microsoft SQL Server Express Edition (Es una edición gratis de SQL Server ideal para desarrollo y pequeñas aplicaciones)
- Microsoft SQL Server Compact Edition Basica
- Sybase ASE Express Edition para Linux (edición gratuita para Linux)
- Oracle Express Edition 10 (solo corre en un servidor, capacidad limitada)
- DB2 Express-C

Los SGDB más comúnmente utilizados son:

- MySQL
- SQL Server
- Oracle
- PostgreSQL
- DB2

1.4. Structured Query Language

Se vio anteriormente que el SGBD brinda un medio de acceso a los usuarios y las aplicaciones a la información almacenada en la base de datos, este acceso se facilita mediante el SQL –*Structured Query Language*- lenguaje de consulta estructura, permite realizar consultas, actualizaciones y/o modificaciones a la información contenida en la base de datos; es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas. Una de sus características es el manejo del álgebra y el cálculo relacional que permiten efectuar consultas con el fin de recuperar de forma sencilla información de interés de bases de datos, así como hacer cambios en ella.

El SQL es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales y permite así gran variedad de operaciones.

Es un lenguaje declarativo de "alto nivel" o "de no procedimiento" que, gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros —y no a registros individuales— permite una alta productividad en codificación y la orientación a objetos. De esta forma, una sola sentencia puede

equivaler a uno o más programas que se utilizarían en un lenguaje de bajo nivel orientado a registros.

1.5. Lenguaje de definición de datos – DDL

El lenguaje de definición de datos (en inglés *Data Definition Language*, o DDL), es el que se encarga de la modificación de la estructura de los objetos de la base de datos. Incluye órdenes para modificar, borrar o definir las tablas en las que se almacenan los datos de la base de datos. Existen cuatro operaciones básicas: CREATE, ALTER, DROP y TRUNCATE, ejemplo:

1.5.1. CREATE

Se permite “crear” objetos, por ejemplo:

Una base de datos

```
CREATE DATABASE <nombre_de_base_de_datos>;
```

Una tabla

```
CREATE TABLE <nombre_de_la_tabla> (
    [lista de atributos]
)
```

Una vista

```
CREATE VIEW <nombre_de_la_vista>
AS
    [datos a mostrar]
```

1.5.2. ALTER

Permite “modificar” objetos ya existentes, por ejemplo:

Agregar una columna a una tabla:

ALTER TABLE <nombre_de_la_tabla> ADD COLUMN <nombre_columna> INT;

Añadir una llave:

ALTER TABLE <nombre_tabla> ADD PRIMARY KEY (<campo>);

Cambiar el tipo de dato de una columna:

ALTER TABLE <nombre_tabla> ALTER <nombre_columna> <tipo_dato>;

1.5.3. DROP

Permite “borrar” objetos, por ejemplo:

Una base de datos:

DROP DATABASE <nombre_de_base_de_datos>;

Una tabla

DROP TABLE <nombre_de_la_tabla>;

Una vista:

DROP VIEW <nombre_de_la_vista>;

1.5.4. TRUNCATE

Borra todos los registros de una tabla, ejemplo:

```
TRUNCATE TABLE <nombre_de_la_tabla>;
```

1.6. Lenguaje de manipulación de datos – DML

Un lenguaje de manipulación de datos (*Data Manipulation Language*, o DML en inglés) es un lenguaje proporcionado por el sistema de gestión de base de datos que permite a los usuarios llevar a cabo las tareas de consulta o manipulación de los datos, organizados por el modelo de datos adecuado.

El lenguaje de manipulación de datos más popular hoy día es SQL, usado para recuperar y manipular datos en una base de datos relacional. Existen tres operaciones básicas: INSERT, UPDATE, DELETE.

1.6.1. INSERT

Insertar nuevos registros en una tabla:

```
INSERT INTO <nombre_de_la_tabla> (<lista_de_campos_a_insertar>)
VALUES (<lista_de_valores>);
```

Copiar una tabla:

```
SELECT <lista_de_columnas>
INTO <nombre_tabla_destino>
FROM <nombre_tabla_origen>
```

1.7. Introducción al análisis y diseño de bases de datos

Se empezará definiendo conceptos introductorios al modelado de las bases de datos:

Como explica NetXell Mycrosystems un dato en particular es la unidad de información más pequeña que tiene un significado en la vida real, por ejemplo nombre o apellido; un registro es un grupo relacionado de datos en particular que se manejan como unidad, por ejemplo nombre, apellido, número de identificación; un archivo es una colección de registros.

Se definirá una base de datos como una colección interrelacionada de información almacenada que sirve las necesidades de múltiples usuarios dentro de una o más organizaciones, una colección de tablas en el modelo relacional.

Las bases de datos son administradas por personas las cuales se llaman DBA (*Database administrator*), mediante sistemas gestores de bases de datos SGDB o DBMS por sus siglas en inglés; los sistemas gestores de bases de datos provee:

- Disponibilidad de la información: hacer una colección de información íntegra disponible a una amplia gama de usuarios.
- Integridad: asegurar la validez y calidad de la información.
- Seguridad: limita los acceso no deseados a la información
- Control de gestión: administración, mantenimiento

Sparx Systems define los siguientes niveles de abstracción en el modelado de datos:

- Conceptual: documenta las entidades básicas de un sistema propuesto y las relaciones entre ellos.
- Lógico: especifica las entidades y las relaciones sin detalles de implementación.
- Físico: define la estructura de una base de datos para un formato de tecnología en específico.

Estos niveles definen las principales etapas del proceso de diseño de una base de datos. El siguiente cuadro es un resumen de los aspectos que se deben tomar en cuenta para el modelado de datos en los niveles conceptual, lógico y físico.

Fernando Caparrini explica que el modelado es un plan figurado para el diseño de bases de dato. El modelado de clases orientado a objetos puede no ser el método más simple o el mejor, pero es el más común y puede ser mal utilizado si no se le conoce. El modelo relacional ha probado ser eficiente y flexible, es esencialmente un conjunto de tablas, las cuales se componen de una o más columnas. Roberto Encalada define que los objetivos del diseño deben proveer:

- Buena documentación
- Fácil mantenimiento
- Agilizar el desarrollo
- Reducir los recursos necesarios
- Optimizar el rendimiento
- Asegurar la calidad

Después del diseño, refinar el esquema y la definición de vistas, se tienen los esquemas conceptuales y externos de la base de datos. El siguiente paso es

escoger índices, tomar decisiones acerca del *clustering* y refinar los esquemas conceptuales y externos si fuera necesario para alcanzar las metas de rendimiento como lo explica Finkelstein.

Así como un buen diseño provee de beneficios para el modelo, un mal diseño causa un rendimiento pobre, baja calidad, difícil entendimiento, uso inapropiado de los recursos; algunas de las causas suele ser:

- Requisitos mal definidos: los requisitos deben definir de manera precisa e inequívoca la necesidad (u oportunidad de negocio) detectada y han de ser completos, coherentes, diseñables y comprobables. No tiene sentido definir de manera incompleta la necesidad identificada, dejando al azar que finalmente el sistema satisfaga al usuario. No tiene sentido establecer requisitos antagónicos; sino se establecen las adecuadas prioridades entre ellos que ayuden a resolver objetivamente los conflictos que se presenten. No tiene sentido requerir nada que no pueda influir en el proceso de diseño, y finalmente no tiene sentido requerir nada que no pueda comprobarse de manera objetiva que ha sido satisfecho explica Rupak Ganguly.

- Cambios continuos en los requisitos: aún en el caso de que el análisis de una determinada necesidad u oportunidad se haya llevado a cabo de manera correcta y esa necesidad u oportunidad haya sido transformada en el conjunto completo y coherente de requisitos diseñables y demostrables, la continua modificación y/o incorporación de requisitos entraña un enorme riesgo para la correcta ejecución del proyecto.

- Falta de conocimientos técnicos: la transformación de necesidades u oportunidades en sistemas que las satisfagan requieren de amplios

conocimientos, cada vez en más disciplinas, que han de ser además debidamente integrados.

- Inadecuada gestión de riesgos: la mala gestión de riesgos, puede conducir al desastre a cualquier proyecto, tanto en lo técnico como en lo económico. En particular, la inclusión de demasiadas tecnologías no maduras, supone en general un riesgo extraordinario, no siempre bien gestionado explica Rupak Ganguly.

1.8. Reglas de integridad

Los conceptos básicos de integridad en el modelo relacional son el de llave primaria, llave foránea, valores nulos y un par de reglas de integridad.

Una llave primaria es un conjunto de uno o más atributos de una tabla, que tomados colectivamente permiten identificar un registro como único.

Una llave foránea de una relación es un atributo que hace referencia a una llave primaria de otra relación; esto da pie a que una relación pueda tener varias llaves foráneas.

Un valor nulo es un valor que está fuera de la definición de cualquier dominio el cual permite dejar el valor del atributo "latente".

Las dos reglas de integridad tienen que ver precisamente con los conceptos antes mencionados y son:

Integridad de relaciones:

Ningún atributo que forme parte de una llave primaria puede aceptar valores nulos.

1.9. Integridad referencial

Garantiza que una entidad (fila o registro) siempre se relaciona con otras entidades válidas, es decir, que existen en la base de datos. Implica que en todo momento dichos datos sean correctos, sin repeticiones innecesarias, datos perdidos ni relaciones mal resueltas.

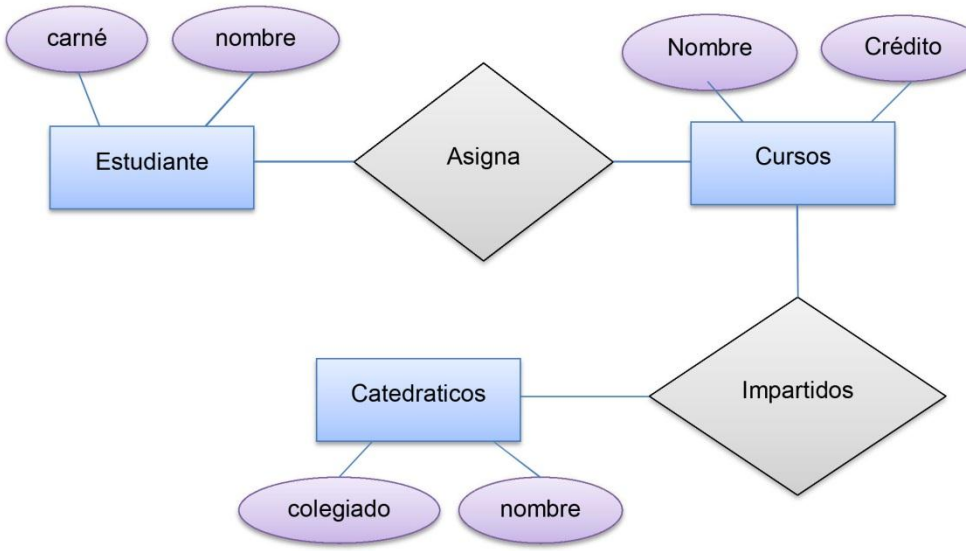
1.10. Diagramas ER

"Cuando se utiliza una base de datos para gestionar información, se está plasmando una parte del mundo real en una serie de tablas, registros y campos ubicados en un ordenador; creándose un modelo parcial de la realidad. Antes de crear físicamente estas tablas en el ordenador se debe realizar un modelo de datos.

Se suele cometer el error de ir creando nuevas tablas a medida que se van necesitando, haciendo así el modelo de datos y la construcción física de las tablas simultáneamente. El resultado de esto acaba siendo un sistema de información parcheado, con datos dispersos que terminan por no cumplir adecuadamente los requisitos necesarios.

Existen distintas notaciones para representar diagramas entidad relación, las cuales se describen e ilustran más adelante; a continuación se muestra un ejemplo de un diagrama entidad relación.

Figura 8. **Ejemplo de diagrama entidad-relación**



Fuente: elaboración propia, con base al programa Microsoft Word 2010.

1.11. Entidades y relaciones

El modelo de datos más extendido es el denominado Entidad/Relación (E/R) En el modelo E/R se parte de una situación real a partir de la cual se definen entidades y relaciones entre dichas entidades:

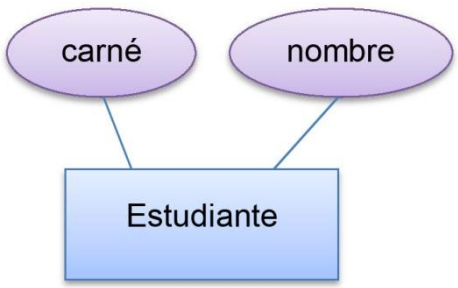
- Entidad: objeto del mundo real sobre el que se quiere almacenar información (ej: una persona). Las entidades están compuestas de atributos que son los datos que definen el objeto (para la entidad persona serían DPI, nombre, apellidos, dirección, etc). De entre los atributos habrá

uno o un conjunto de ellos que no se repite; a este atributo o conjunto de atributos se le llama clave de la entidad, (para la entidad persona una clave podría ser DPI). En toda entidad siempre hay al menos una clave que en el peor de los casos estará formada por todos los atributos de la tabla. Ya que pueden haber varias claves y se necesita elegir una, se hará atendiendo a estas normas:

- Que sea única.
- Que se tenga pleno conocimiento de ella.- ¿Por qué en las empresas se asigna a cada cliente un número de cliente?
- Que sea mínima, ya que será muy utilizada por el gestor de base de datos.

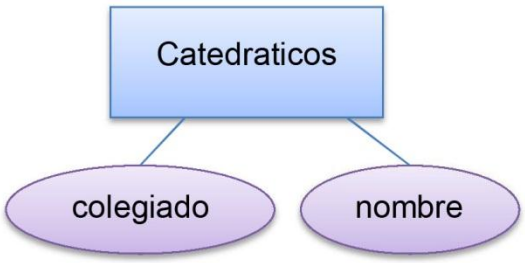
Siguiendo el ejemplo presentado anteriormente, se puede ilustrar las entidades de la siguiente manera:

Figura 9. **Ejemplo de una entidad "estudiante" con sus atributos**



Fuente: elaboración propia, con base al programa Microsoft Word 2010.

Figura 10. **Ejemplo de una entidad “catedrático” con sus atributos**

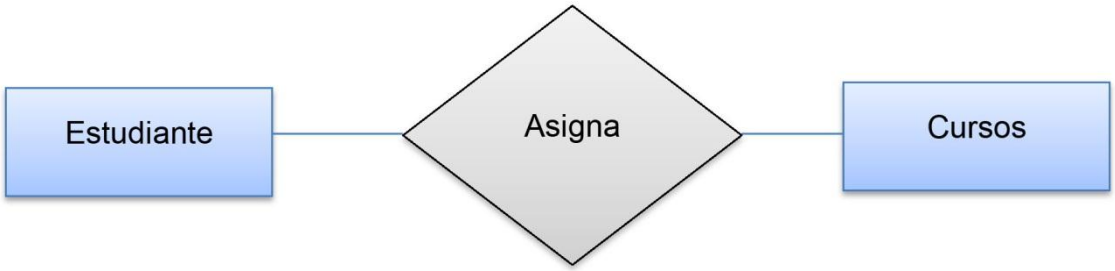


Fuente: elaboración propia, con base al programa Microsoft Word 2010.

- Relación: asociación entre entidades, sin existencia propia en el mundo real que se está modelando, pero necesaria para reflejar las interacciones existentes entre entidades. Las relaciones pueden ser de tres tipos:

    - Relaciones 1-1.- Las entidades que intervienen en la relación se asocian una a una (Ej: la entidad HOMBRE, la entidad MUJER y entre ellos la relación MATRIMONIO).
    - Relaciones 1-n.- Una ocurrencia de una entidad está asociada con muchas (n) de otra (Ej: la entidad EMPERSA, la entidad TRABAJADOR y entre ellos la relación TRABAJAR-EN).
    - Relaciones n-n.-Cada ocurrencia, en cualquiera de las dos entidades de la relación, puede estar asociada con muchas (n) de la otra y viceversa (Ej: la entidad ALUMNO, la entidad CURSO y entre ellos la relación MATRÍCULA).".

Figura 11. **Ejemplo de una relación entre dos entidades**



Fuente: elaboración propia, con base al programa Microsoft Word 2010.

1.12. Notaciones

Las notaciones sirven para representar de manera gráfica, ordenada y estandarizada las conceptualizaciones que quieren plasmar del mundo real. Las notaciones más útiles (descritas más adelante) son la notación de Peter Chen y la notación UML. La importancia de la notación de Peter Chen permite conceptualizar gráficamente las relaciones entre las entidades, y ver sus atributos de manera simple. Si bien la notación de Chen da una vista rápida para entender el sistema, no provee información detallada, por ejemplo, tipos de datos, longitud, contraints, etc., la cual se pueden representar utilizando la notación UML, a continuación se describirá y ejemplificará cada notación:

1.12.1. Chen

Utiliza rectángulos para representar a las entidades, y los diamantes para representar las relaciones adecuadas para objetos de primera clase: se puede tener atributos y relaciones de los suyos. Conjuntos de entidades se dibujan como rectángulos, la relación de grupos como los diamantes. Si un conjunto de

entidades participa en un conjunto de relaciones, que están conectados con una línea.

Los atributos son dibujados como óvalos y se conectan con una línea exactamente a una entidad o conjunto de relaciones.

Restricciones de cardinalidad se expresan de la siguiente manera:

- Una línea doble indica una restricción de participación, la totalidad (*surjectivity*): todas las entidades en el conjunto de entidades que participan en al menos una relación en el conjunto de relaciones.

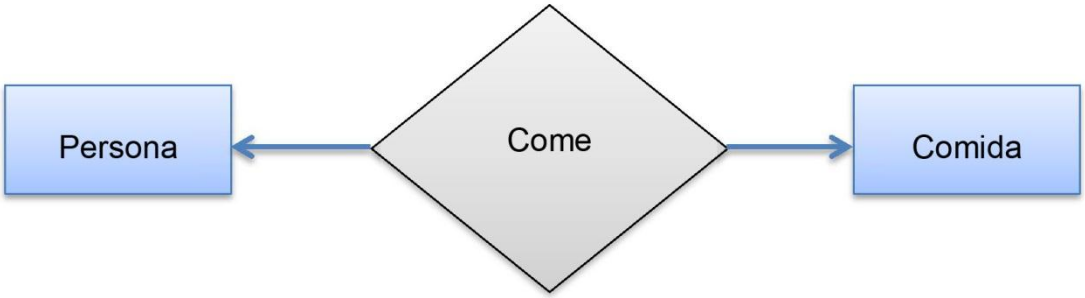
- Una flecha de la entidad creada para establecer la relación indica una restricción de clave, es decir la inyectividad: cada entidad del conjunto de entidades pueden participar como máximo en una relación en el conjunto de relaciones.

- Una línea gruesa indica a la vez, es decir, *bijectivity*: cada entidad en el conjunto de entidades se trata exactamente de una relación.

- Un nombre subrayado de un atributo indica que se trata de una clave: dos entidades diferentes o las relaciones con este atributo siempre tienen diferentes valores para este atributo.

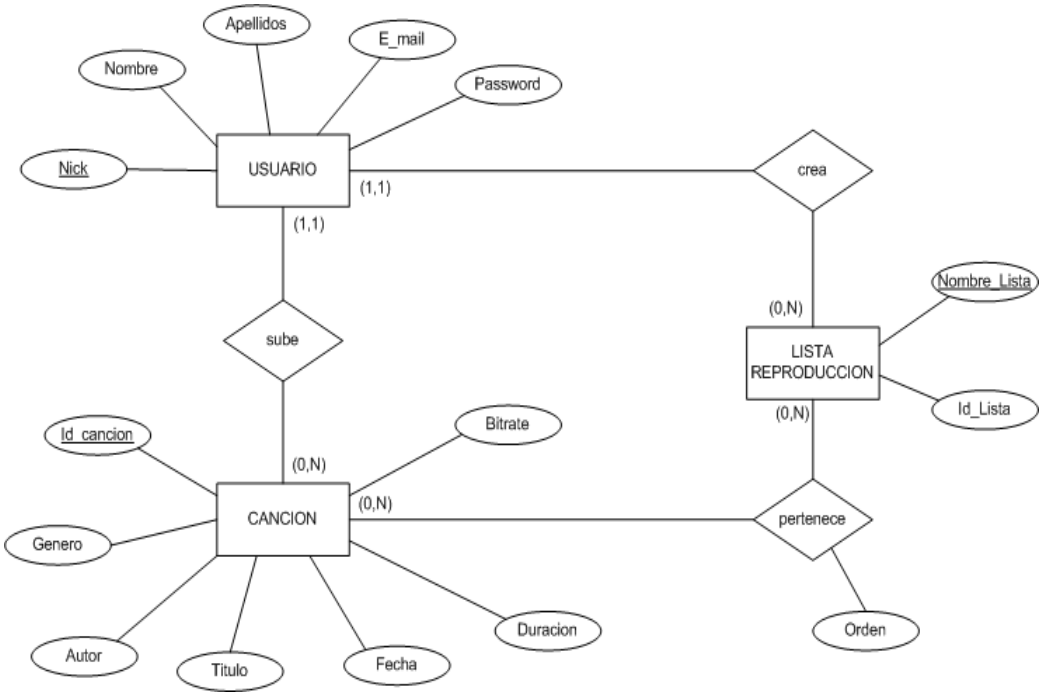
Los atributos son a menudo omitidos, ya que pueden saturar un diagrama, diagrama de otras técnicas a menudo atributos de entidad dentro de la lista de los rectángulos dibujados por los conjuntos de entidades. (8).

Figura 12. **Notación Peter Chen**



Fuente: elaboración propia, con base al programa Microsoft Word 2010.

Figura 13. **Diagrama entidad relación con notación Chen**

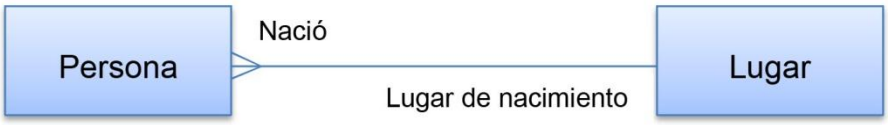


Fuente: infosol.es. Consulta: 15 de marzo de 2013.

1.12.2. Crow

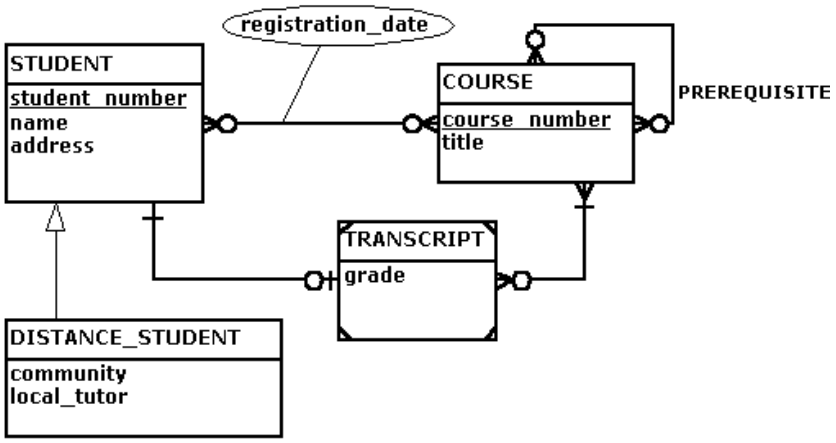
La notación de pata de gallo se utiliza en la notación de Barker, SSADM e Ingeniería de la información. Diagramas de pata de gallo representan a entidades como las cajas, y las relaciones como las líneas entre cajas. Formas diferentes en los extremos de estas líneas representan cardinalidad de relación. (8).

Figura 14. **Notación Crow´s foot**



Fuente: elaboración propia, con base al programa Microsoft Word 2010.

Figura 15. **Diagrama entidad relación con notación Crow´s Foot**



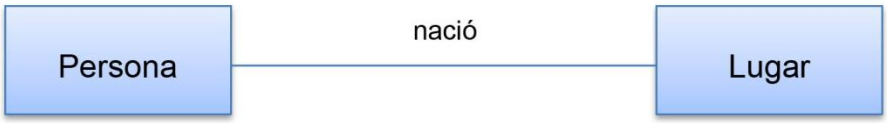
Fuente: college.yukondude.com. Consulta: 15 de marzo de 2013.

1.12.3. UML

Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un plano del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

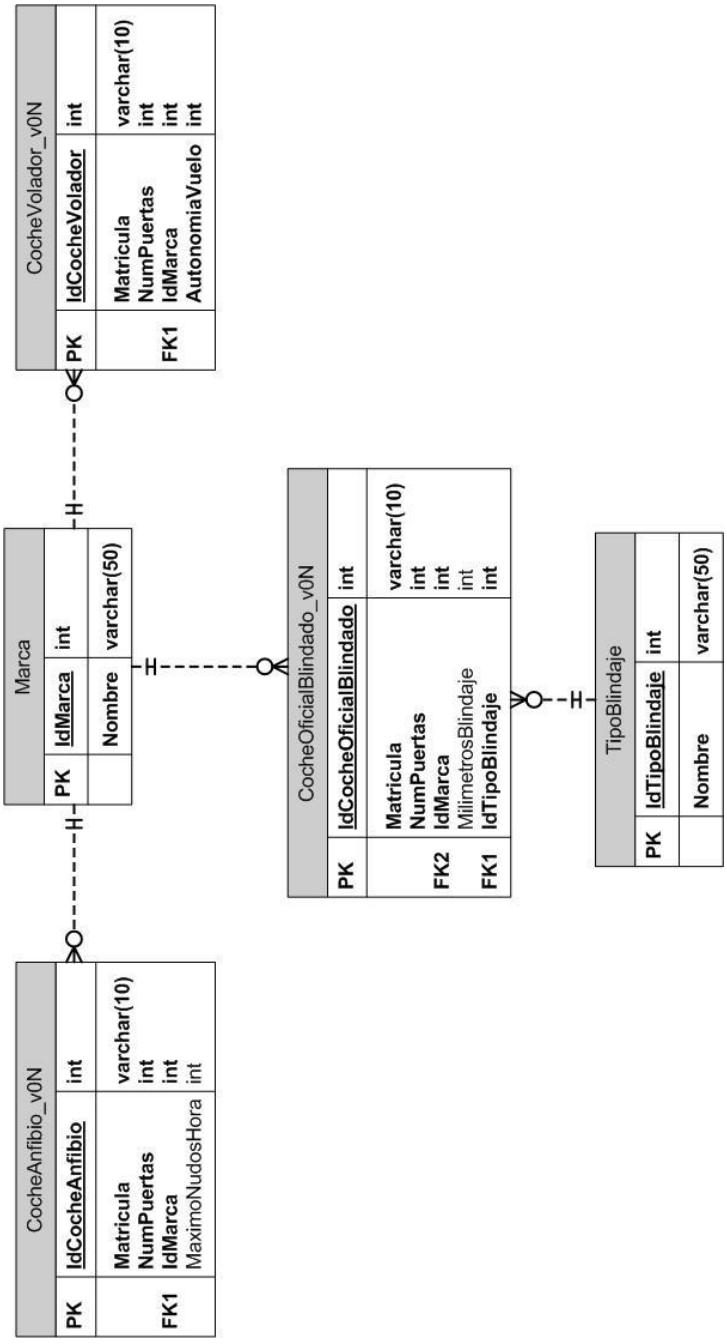
Robert Garvey explica que es importante resaltar que UML es un lenguaje para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software, pero no especifica en sí mismo qué metodología o proceso usar.

Figura 16. **Notación UML**



Fuente: elaboración propia, con base al programa Microsoft Word 2010.

Figura 17. **Diagrama entidad relación con notación UML**



Fuente: miguelmatas.es. Consulta: 15 de marzo de 2013.

1.13. Importancia de la eficiencia

Eficacia mide los resultados alcanzados en función de los objetivos que se han propuesto, presuponiendo que esos objetivos se mantienen alineados con la visión que se ha definido. Mayor eficacia se logra en la medida que las distintas etapas necesarias para arribar a esos objetivos, se cumplen de manera organizada y ordenada sobre la base de su prioridad e importancia, como lo explica la Universidad de Granada.

Cumplir con el objetivo no es suficiente, si se aspira a un modelo de calidad, ahí es donde entra en juego la eficiencia.

Eficiencia consiste en la medición de los esfuerzos que se requieren para alcanzar los objetivos. El costo, el tiempo, el uso adecuado de factores materiales y humanos, cumplir con la calidad propuesta, constituyen elementos inherentes a la eficiencia. Los resultados más eficientes se alcanzan cuando se hace uso adecuado de estos factores, en el momento oportuno, al menor costo posible y cumpliendo con las normas de calidad requeridas.

1.14. Pruebas de rendimiento

Es un método utilizado para determinar cómo se comportará un sistema en una situación de extrema carga y así poder prevenir desastres y/o optimizar el uso de los recursos. (10).

Se puede realizar pruebas de rendimiento mediante aplicaciones especializadas como SQL Profiler, Apache Benchmarking, SQL Trace/Profile, pero también se puede hacer de la manera más básica escribiendo sentencias SQL y haciendo múltiples llamadas a la base de datos, repitiendo el proceso una

cantidad significativa de veces como se explica en una publicación en línea hecha en TechNet.

Las pruebas de rendimiento pueden servir para diferentes propósitos. Pueden demostrar que el sistema cumple los criterios de rendimiento. Pueden comparar dos sistemas para encontrar cuál de ellos funciona mejor. O pueden medir que partes del sistema o de carga de trabajo provocan que el conjunto rinda mal. Para su diagnóstico, los ingenieros de software utilizan herramientas como pueden ser monitorizaciones que midan qué partes de un dispositivo o software contribuyen más al mal rendimiento o para establecer niveles (y umbrales) del mismo que mantenga un tiempo de respuesta aceptable.

Es fundamental para alcanzar un buen nivel de rendimiento de un nuevo sistema, que los esfuerzos en estas pruebas comiencen en el inicio del proyecto de desarrollo y se amplié durante su construcción. Cuanto más se tarde en detectar un defecto de rendimiento, mayor es el coste de la solución. Esto es cierto en el caso de las pruebas funcionales, pero mucho más en las pruebas de rendimiento, debido a que su ámbito de aplicación es de principio a fin.

En las pruebas de rendimiento, a menudo es crucial (y con frecuencia difícil de conseguir) que las condiciones de prueba sean similares a las esperadas en el uso real. Esto es, sin embargo, casi imposible en la práctica. La razón es que los sistemas en producción tienen un carácter aleatorio de la carga de trabajo y aunque en las pruebas se intente dar lo mejor de sí para imitar el volumen de trabajo que pueda tener el entorno de producción, es imposible reproducir exactamente la variabilidad de ese trabajo, salvo en el sistema más simple.

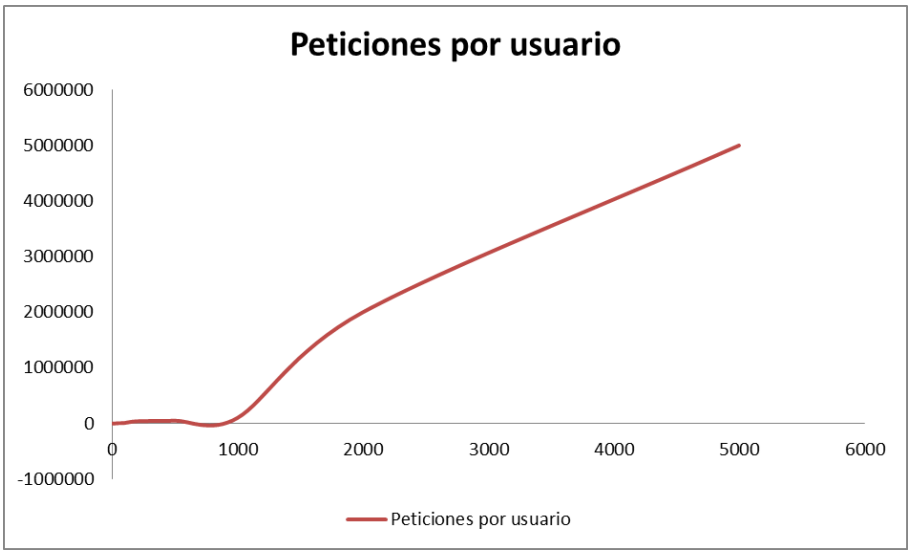
Los nuevos conceptos en la implementación de la arquitectura (por ejemplo: SOA) han añadido complejidad adicional a las pruebas de rendimiento. Los

servicios o recursos de la empresa (que comparten infraestructura o plataforma) requieren pruebas de rendimiento coordinadas (con la creación del volumen y carga de todos los sistemas que comparten la infraestructura o plataformas) para reproducir realmente el estado del entorno de producción. Debido a la complejidad, coste y tiempo necesario en torno a esta actividad, algunas organizaciones emplean herramientas que pueden mostrar y crear condiciones (también conocido como "ruido") en los entornos de pruebas de rendimiento para comprender la capacidad y las necesidades de recursos y verificar/validar los niveles de calidad

1.14.1. Pruebas de carga

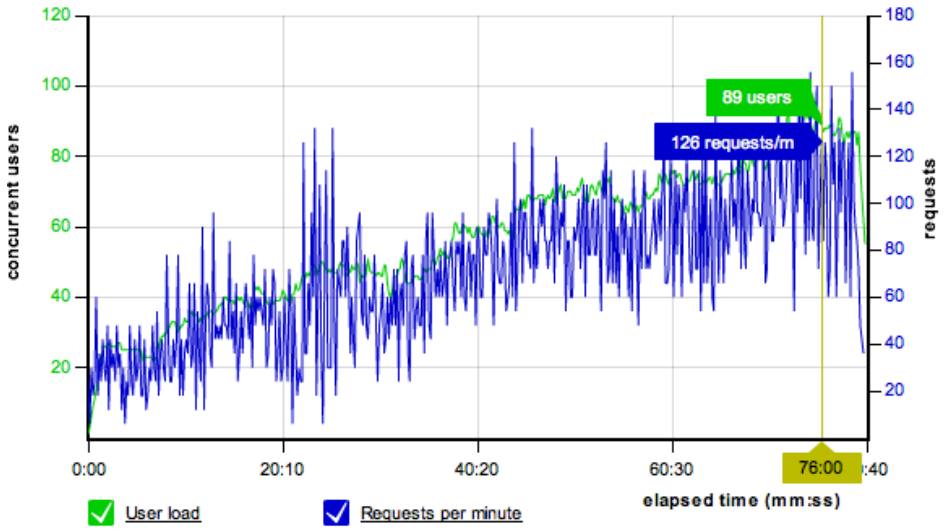
Este es el tipo más sencillo de pruebas de rendimiento. Una prueba de carga se realiza generalmente para observar el comportamiento de una aplicación bajo una cantidad de peticiones esperada. Esta carga puede ser el número esperado de usuarios concurrentes utilizando la aplicación y que realizan un número específico de transacciones durante el tiempo que dura la carga. Esta prueba puede mostrar los tiempos de respuesta de todas las transacciones importantes de la aplicación. Si la base de datos, el servidor de aplicaciones, etc. También se monitorizan, entonces esta prueba puede mostrar el cuello de botella en la aplicación.

Figura 18. **Prueba de rendimiento: prueba de carga obtenida**



Fuente: elaboración propia, con base al programa Microsoft Excel 2010.

Figura 19. **Prueba de rendimiento: prueba de carga esperada**



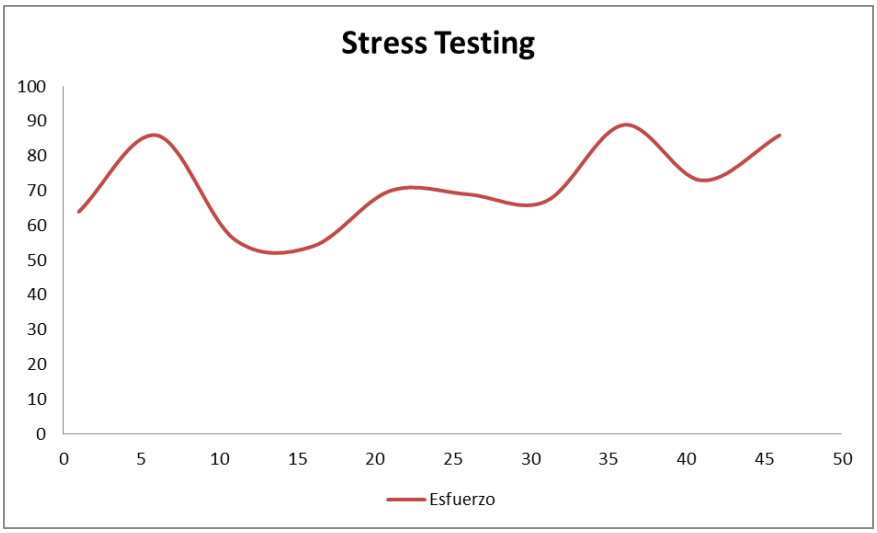
Fuente: loadstorm.com. Consulta: 6 de mayo de 2012.

Durante una prueba de carga lo que se espera es que a medida que la cantidad de usuarios se incremente, también lo haga la cantidad de peticiones, y se busca que el servidor sea no solo capaz de soportar dicha carga, sino que responda en un lapso de tiempo adecuado.

1.14.2. Prueba de estrés (también llamada prueba de esfuerzo)

Esta prueba se utiliza normalmente para romper la aplicación. Se va doblando el número de usuarios que se agregan a la aplicación y se ejecuta una prueba de carga hasta que se rompe. Este tipo de prueba se realiza para determinar la solidez de la aplicación en los momentos de carga extrema y ayuda a los administradores para determinar si la aplicación rendirá lo suficiente en caso de que la carga real supere a la carga esperada.

Figura 20. **Prueba de estabilidad (*soak testing)***

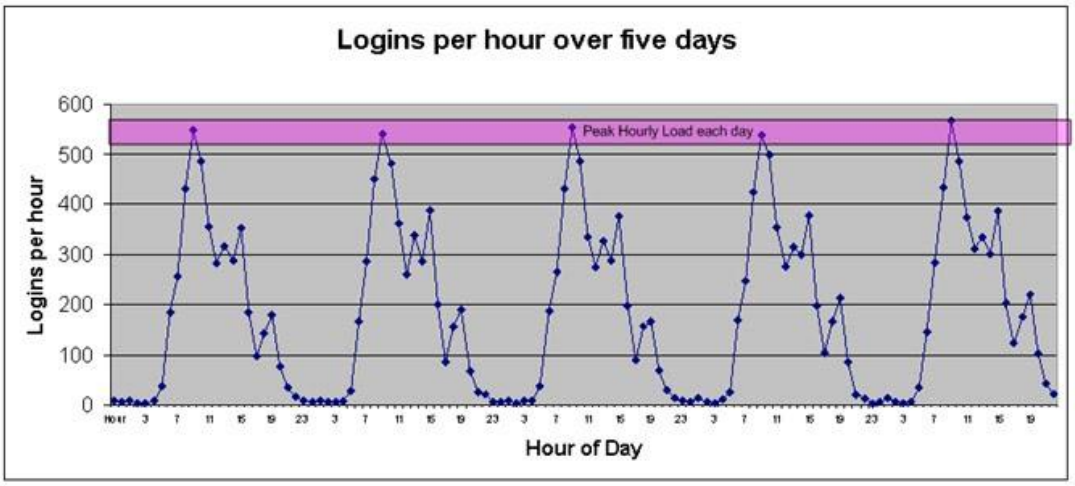


Fuente: elaboración propia, con base al programa Microsoft Excel 2010.

1.14.3. Prueba de estabilidad (*soak testing*)

Esta prueba normalmente se hace para determinar si la aplicación puede aguantar una carga esperada continuada. Generalmente esta prueba se realiza para determinar si hay alguna fuga de memoria en la aplicación.

Figura 21. **Prueba de rendimiento: prueba de estabilidad**

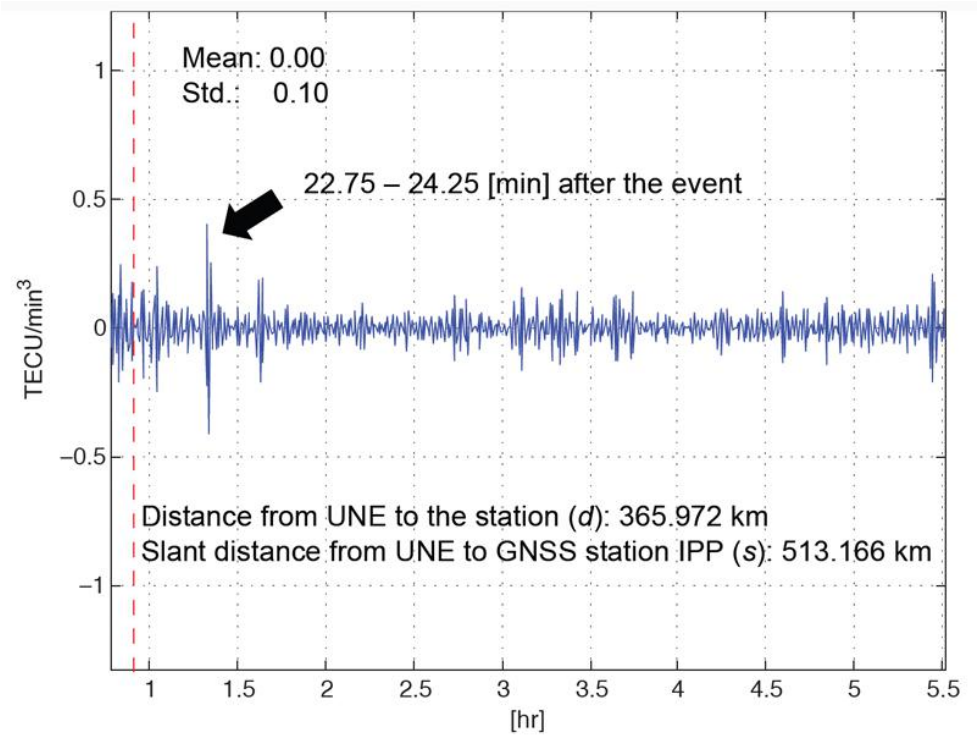


Fuente: sharadabreddy.blogspot.com. Consulta: 6 de mayo de 2012.

1.14.4. Pruebas de picos (*spike testing*)

La prueba de picos, como el nombre sugiere, trata de observar el comportamiento del sistema variando el número de usuarios, tanto cuando bajan, como cuando tiene cambios drásticos en su carga. Esta prueba se recomienda que sea realizada con un software automatizado que permita realizar cambios en el número de usuarios mientras que los administradores llevan un registro de los valores a ser monitorizados.

Figura 22. **Prueba de rendimiento: prueba de picos**



Fuente: researchnews.osu.edu. Consulta: 6 de Mayo de 2012.

1.15. Consideraciones a tomar en cuenta para realizar pruebas de estrés

- Decidir usar recursos internos o externos para ejecutar las pruebas, en función de la experiencia de la casa (o falta de ella).
- Reunir u obtener requisitos de rendimiento (especificaciones) de los usuarios y/o analistas.
- Desarrollar un plan de alto nivel, incluyendo los requisitos, recursos, plazos e hitos.

- Elaborar un plan de pruebas de rendimiento detallado (incluyendo los escenarios detallados y casos de prueba, cargas de trabajo, información del entorno, etc).
- Elegir la/s herramienta/s de prueba.
- Especificar los datos de prueba necesarios y la distribución de ellos (a menudo pasado por alto, y a menudo el fracaso de una prueba de rendimiento válida).
- Desarrollar *scripts* de prueba de concepto para cada aplicación/componente sometido a la prueba, utilizando la herramienta de prueba elegida y estrategias.
- Desarrollar un plan de prueba de rendimiento detallado, incluyendo todas las dependencias y los plazos.
- Instalar y configurar los simuladores de peticiones y controladores.
- Configurar el entorno de prueba (lo ideal es que sea idéntico hardware a la plataforma de producción), configurar los *router*, aislar la red (no se quiere alterar los resultados por parte de otros usuarios), desplegar la aplicación en el servidor, desarrollar la base de datos de prueba, etc.
- Ejecutar las pruebas, probablemente en repetidas ocasiones (iterativamente), a fin de ver si el desconocimiento de cualquier factor podría afectar a los resultados.
- Analizar los resultados - ya sea de aceptando/rechazando, o investigando el camino crítico y recomendando medidas correctivas.

2. MODELACIÓN DE UN PROBLEMA DE LA VIDA REAL (CASOS DE ESTUDIO)

2.1. Caso de estudio

Se pretende modelar un sistema básico que lleve el control de clientes y sus compras (facturas), únicamente, pueden existir clientes que no tengan ninguna factura de compra, pero todas las facturas deben ser asociadas a un cliente, y este debe tener un tipo de cliente. De los clientes interesa guardar el nombre completo, fecha de nacimiento, dirección, código alternativo (para futuros descuentos). Los tipos de clientes pueden ser: básico, intermedio (recibe 15 % de descuento) y avanzado (recibe 35 % de descuento). La factura debe contener la fecha, serie y número de factura, el total de la venta y el impuesto que se pagó (IVA 12 %). El modelo a usar se detalla a continuación:

Tabla: Invoice

Atributos:

DocumentType	INT
DocumentName	VARCHAR(50)
DocumentDate	DATETIME
Value	NUMERIC(12, 4)
CustomerID	INT
Tax	NUMERIC(12, 4)

Tabla: Customer

Atributos:

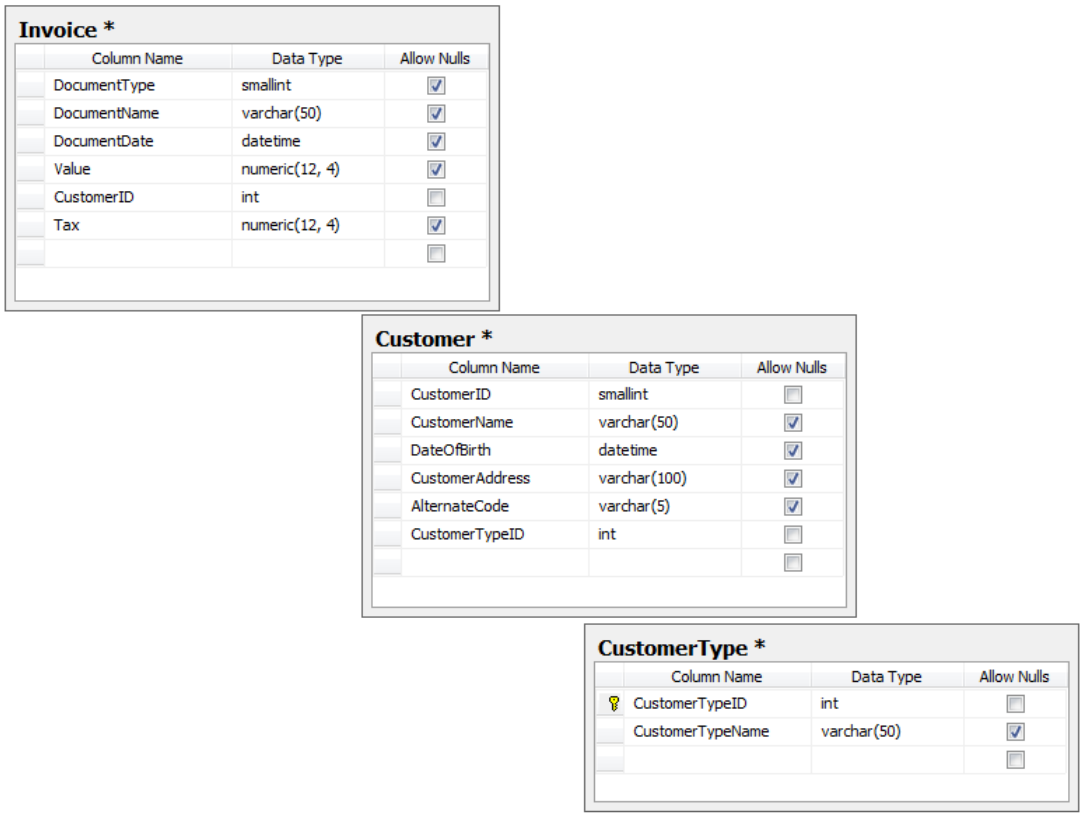
CustomerID	INT
CustomerName	VARCHAR(50)
DateOfBirth	DATETIME
CustomerAddress	VARCHAR(100)
AlternateCode	VARCHAR(5)
CustomerTypeID	INT

Tabla: CustomerType

Atributos:

CustomerTypeID	INT
CustomerTypeName	VARCHAR(50)

Figura 23. **Diagrama de relación del caso de estudio**



Fuente: elaboración propia, con base al programa SQL Server Management Studio 2008 R2.

2.2. Pruebas de rendimiento

Las pruebas de rendimiento han sido realizadas bajo el sistema descrito a continuación:

Servidor de base de datos:

- Procesador: Intel Xeon X5355 2.66Mhz (2 procesadores)
- Sistema operativo: Windows Server 2008 x64

- Memoria RAM: 2GB
- DBMS: SQL Server 2008 R2 Standard Edition 64 bits

A continuación se detallan las pruebas de rendimiento realizadas al modelo:

- Campos autoincrementales: el valor de un campo auto-incremental es asignado por el sistema, dependiendo la configuración (incremental 1 en 1, 2 en 2, 3 en 3, etc.), dado que representa una carga para el sistema, debe de afectar directamente el rendimiento, ya que el sistema primero debe conocer cuál fue el último valor asignado e incrementarlo.

    - Prueba: rendimiento de los campos enteros autoincrementales en comparación de los campos enteros no autoincrementales.

- Tipos de datos: los tipos de datos definen el tipo de valor a almacenar en los campos de las tablas (enteros, cadenas, fechas, bit, XML, etc), las buenas prácticas residen en saber y analizar los valores esperados y en base a ello, asignar un tipo de dato apropiado. En el modelo originalmente se tienen campos de tipo INT cuyo tipo de dato puede almacenar hasta 2 000 000 000 enteros positivos como lo menciona David Kroenke, lo apropiado seria usar tipos SMALLINT que permite almacenar hasta 32 000 enteros positivos, ya que se harán pruebas sobre únicamente 15 000 registros.

    - Prueba: rendimiento de los campos enteros apropiados autoincrementales en comparación de los campos enteros apropiados no autoincrementales.

- Llaves foráneas: las llaves foráneas son estructuras que permiten obligar que exista una relación (en el modelo, no existe un cliente sin un tipo de cliente, no existe una factura sin un cliente). Antes de insertar un nuevo registro las llaves foráneas permiten validar que existan los valores que se están referenciando, ahora bien, se pueden “ahorrar” esos recursos si se tiene otra manera de validar la integridad de la información que se quiere insertar, por ejemplo, validación en una aplicación que inserta los valores.

    - Prueba: rendimiento de los campos enteros apropiados autoincrementales con llaves foráneas en comparación de los campos enteros apropiados no autoincrementales sin llaves foráneas.

- Índices: los índices son estructuras que permiten un rápido acceso a los datos, cada nuevo registro insertado es ordenado de acuerdo a la clave definida, esto conlleva el uso de recursos para llevar a cabo este proceso. Un índice en una tabla actúa tipificando un índice de un libro, si se desea insertar una nuevo página se debe de verificar el orden que se desea mantener. Si la información no precisa ser ordenada los índices pueden ser una pérdida de recursos, de lo contrario, se va a estudiar dos tipos de índices, los agrupados (*clustered*) y los no agrupados (*non-clustered*).

    - Prueba: rendimiento de los campos enteros autoincrementales con índices *clustered* en comparación de los campos enteros autoincrementales con índices *non-clustered* y campos enteros autoincrementales sin índices.

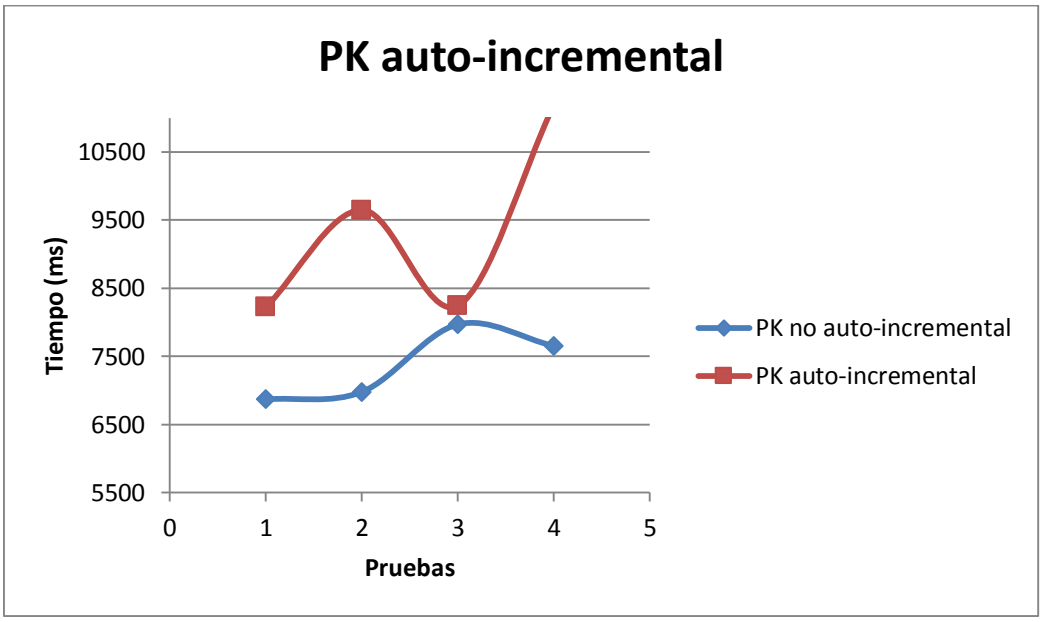
2.3. Análisis de resultados de las pruebas de rendimiento

Se han hecho una serie de 4 corridas de prueba para garantizar la exactitud e independencia de los resultados, los tiempos están medidos en milisegundos.

Prueba # 1: rendimiento de los campos enteros autoincrementales en comparación de los campos enteros no autoincrementales.

Lo que se pretende analizar es el impacto que tienen los campos autoincrementales en el rendimiento de una base de datos. Un campo auto-incremental incrementa su valor cada vez que se inserta un nuevo registro, evitando al desarrollador o administrador de la base de datos, llevar el control de qué valor debe llevar ese campo; dado que el motor de la base de datos asume dicha responsabilidad, supone una leve carga, a continuación se estudia el efecto que tiene, y la magnitud de la carga que representa.

Figura 24. **Comportamiento resultado de la prueba no. 1 realizada**



Fuente: elaboración propia, con base al programa Microsoft Excel 2010.

Tabla I. **Comportamiento resultado de la prueba no. 1 realizada**

Prueba	Prueba1	Prueba2	Prueba3	Prueba4	Promedio
PK no auto-incremental	6 870	69 80	7 970	7 653	7 368,25
PK auto-incremental	8 223	9 650	8 246	11 193	9 328

Relación
21,01 %

Fuente: elaboración propia, con base al programa Microsoft Word 2010.

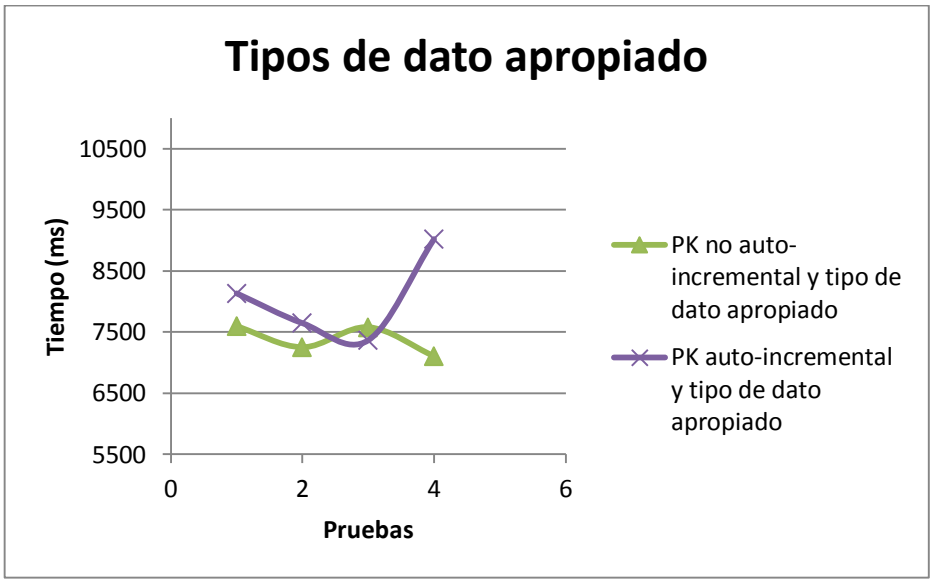
Análisis de resultado: los campos autoincrementales tienden a ser costosos a medida que se incrementa la cantidad de registros, mientras que un valor no incremental se mantiene estable, en este aspecto se puede decir que los campos no incrementales son un 21 % más eficiente que los autoincrementales.

Prueba # 2: rendimiento de los campos enteros apropiados autoincrementales en comparación de los campos enteros apropiados no autoincrementales.

Dado que almacenar tipos de datos en la base de datos representa un consumo variable de memoria, se supone que a tipos de datos más grandes menos es el rendimiento de la base de datos. El escenario es el siguiente: se necesita llevar un control de los empleados de una empresa, para lo cual, se crea un atributo EmpleadoID que es del tipo INT; si se analiza más a fondo, se ve que el tipo de dato INT puede almacenar una cantidad de 32 000 millones de números, una cantidad que seguramente, la empresa nunca tendrá de empleados. Se estima que la empresa contrata dos personas al mes, y lleva en operaciones 3 años. Esto supone que a la fecha, se tendría un aproximado de 108 empleados, por lo que se estaría desperdiciando 21 999 millones de valores; lo óptimo sería utilizar un valor de tipo TINYINT que permite almacenar 256 enteros positivos, o en su defecto utilizar un tipo de datos SMALLINT que permite almacenar 32 000 enteros positivos.

A continuación se estudia el efecto que tiene no utilizar el tipo de dato apropiado:

Figura 25. **Comportamiento resultado de la prueba no. 2 realizada**



Fuente: elaboración propia, con base al programa Microsoft Excel 2010.

Tabla II. **Comportamiento resultado de la prueba no. 2 realizada**

Prueba	Prueba1	Prueba2	Prueba3	Prueba4	Promedio
PK no auto-incremental y tipo de dato apropiado	7 596	7 253	7 580	7 106	7 383,75
PK auto-incremental y tipo de dato apropiado	8 133	7 646	7 366	9 020	8 041,25

Relación
8,18 %

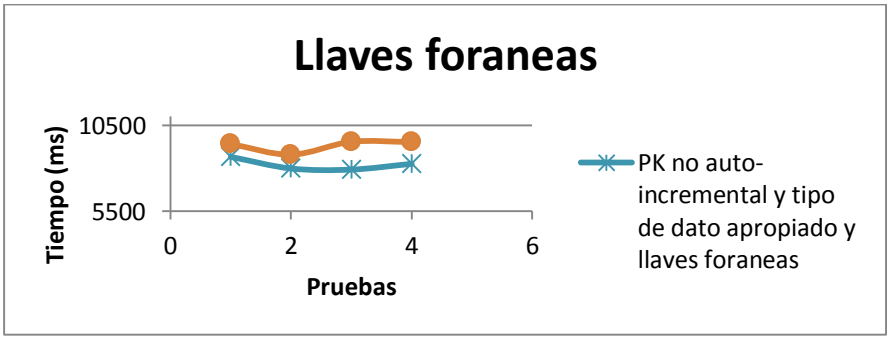
Fuente: elaboración propia, con base al programa Microsoft Word 2010.

Análisis de resultados: un tipo de dato apropiado tiende a representar un menor gasto de recursos frente a uno no apropiado, se ve que un tipo de dato apropiado es 8 % más eficiente.

Prueba # 3: rendimiento de los campos enteros apropiados autoincrementales con llaves foráneas en comparación de los campos enteros apropiados no autoincrementales sin llaves foráneas.

Las llaves permiten asegurar la integridad de la información. El escenario es el siguiente: se necesita almacenar la información de empleados y el departamento al cual pertenece, no pueden haber empleados sin un departamento asignado, más si departamentos que no tengan empleados; para asegurar que no exista un empleado sin departamento se crea una llave foránea que relaciona el empleado a un departamento, esto supone una carga a la base de datos, ya que cada vez que se haga una inserción en “empleado”, esta deberá verificar que el departamento al cual se le está asignando realmente exista. A continuación se estudia la magnitud que representa ésta carga para la base de datos:

Figura 26. **Comportamiento resultado de la prueba no. 3 realizada**



Fuente: elaboración propia, con base al programa Microsoft Excel 2010.

Tabla III. **Comportamiento resultado de la prueba no. 3 realizada**

Prueba	Prueba1	Prueba2	Prueba3	Prueba4	Promedio
PK no auto-incremental y tipo de dato apropiado y llaves foráneas	8 660	7 983	7 920	8 256	8 204,75
PK auto-incremental y tipo de dato apropiado y llaves foráneas	9 406	8 793	9 540	9 523	9 315,5

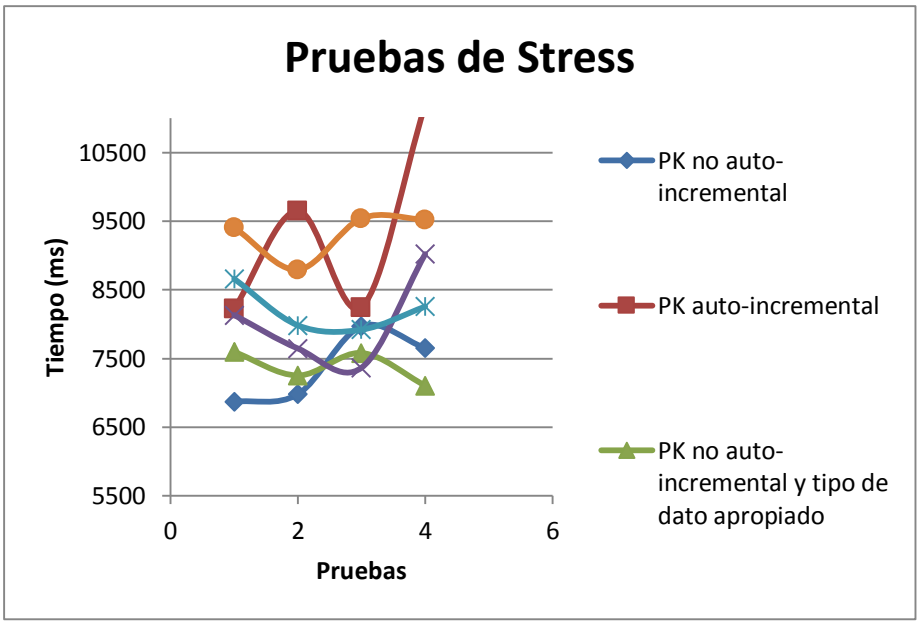
Relación
1,92 %

Fuente: elaboración propia, con base al programa Microsoft Word 2010.

Análisis de resultados: las llaves foráneas proveen integridad al sistema, pero representan un costo de 12 % más de recursos a un sistema sin llaves. Una solución temporal para la carga masiva de datos (si fuera un proceso de una vez), es verificar que la información a insertar es válida, remover las llaves que puedan afectar el proceso, insertar la información, volver a crear las llaves; esto permitirá hacer una carga masiva de la información (considerando que es un único proceso) sin afectar el rendimiento.

En el siguiente cuadro comparativo se puede ver la comparación de todas las pruebas realizadas, las cuales muestran que en términos de recursos, los campos autoincrementales y llaves foráneas representan el mayor costo, por lo que son factores a evitar.

Figura 27. **Cuadro comparativo del rendimiento de cada factor de optimización**



Fuente: elaboración propia, con base al programa Microsoft Excel 2010.

3. METODOLOGÍA PARA EL DISEÑO DE BASES DE DATOS EFICIENTES PARA PROYECTOS DE TAMAÑO MEDIANO DE LA FACULTAD DE INGENIERÍA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Una metodología es un conjunto de procedimientos ordenados utilizados para alcanzar un objetivo explica Elias Mejia. El objetivo que se persigue es diseñar una base de datos eficiente, que no utilice más recursos de los necesarios; se han identificado una serie de 5 pasos distribuidos en 4 áreas principales, las cuales se describen a continuación:

- Análisis
- Diseño
- Validación y verificación
- Buenas practicas

3.1. Análisis

El análisis de los requerimientos es el primer paso fundamental para un sistema exitoso, este define el enfoque que se le dará a la información que se necesita almacenar, ya sea integridad de la información, estabilidad, baja latencia, rápido tiempo de respuesta, etc.

Paso 1: definir los objetivos de la base de datos

Definir el objetivo ayuda a fijar la meta que se quiere lograr, si se está diseñando una base de datos para un buscador de internet, gustaría tener una respuesta rápida y resultados precisos, si por el contrario, se diseña una base de datos para datos financieros, se necesita que la información sea integra y tenga un nivel de seguridad elevado, si se diseña una base de datos para una biblioteca se necesitará que la información este ordenada, disponible para consultarse aun cuando la información fuera de hace décadas.

Paso 2: definir las reglas del negocio

Las reglas de negocio definen normas o políticas que rigen los procesos dentro de una organización en específico, es de vital importancia ya que le dan significado a la información, por ejemplo, el código de barras de un producto es relevante para un control de inventarios, pero no es relevante para una persona en particular que desea adquirir el producto; el tiempo de garantía de un producto podría ser irrelevante en un control de inventarios, pero puede llegar a ser sumamente importante en una venta de electrodomésticos. Las reglas del negocio pueden carecer de sentido fuera de la organización, pero se deben tomar en cuenta durante el proceso de diseño reglas como:

- Pueden haber clientes con 3 nombres
- No se pueden registrar personas menores de 19 años
- Se debe registrar al menos 2 direcciones
- Se debe registrar al menos 2 números telefónicos
- Solo se hacen entregas a domicilio a zonas contiguas a la tienda

Las reglas de negocio además de brindar una perspectiva del negocio, dicen que información (que posteriormente se trasladarán en entidades) se debe almacenar y que no, por ejemplo: se necesita almacenar solamente un nombre y un apellido; además, permite saber el dominio de los datos, por ejemplo: cada cliente al firmar el contrato adquiere un código BX que identifica su número de casillero, el cual está formado por letras y números; conocer el dominio de los datos permite escoger el mejor tipo de dato para la información a almacenar, por ejemplo: si se necesita almacenar una edad, se podría escoger un tipo de dato que permita almacenar números enteros positivos hasta 256, no necesita almacenar miles de números, ya que no existen personas que excedan los 150 años, pero si se necesita almacenar un código de rastreo de un paquete (*tracking number*) se debe usar un campero que permita almacenar enteros positivas de al menos 10 cifras.

Dejando de lado la información, las reglas de negocio también permiten definir los usuarios que tendrán acceso a la información y el nivel de acceso, por ejemplo: el gerente de mercadeo debe tener acceso a consultar reportes de ventas mensuales, pero, el gerente de personal no. El encargado de bodega necesita tener control sobre los inventarios de productos, pero un cajero, necesita únicamente datos de clientes y compras para facturar.

Diseño

Paso 3: elegir una nomenclatura adecuada

El diseño de sistemas ofrece una amplia variedad de diagramas, los cuales son utilizados con distintos propósitos, entre ellos se pueden mencionar:

- Diagramas de clases
- Diagramas entidad-relación
- Diagramas de procesos
- Diagramas de flujo
- Historias de usuarios
- Casos de uso
- Etc.

El objetivo es seleccionar el diagrama que mejor describa la estructura del sistema y permita ver de una manera clara como se va a satisfacer los requerimientos dados.

Usualmente los proyectos a los que se enfoca esta investigación, surgen a partir de un enunciado de requerimientos, por lo que la mejor manera de modelar la estructura es usar diagramas entidad-relación.

Paul Nielsen explica que los diagramas entidad-relación permiten crear un modelo parcial de la realidad en una serie de tablas, registros y campos; un diagrama entidad relación se compone de: entidades, atributos y las relaciones entre las entidades (descrito en el Capítulo I – Introducción al análisis y diseño de bases de datos),

Una vez hecho el análisis (paso 1 y paso 2) se puede tener una buena idea de las entidades, sus atributos y las relaciones que se van a usar, las cuales se deben modelar en un diagrama entidad-relación.

Validación y verificación

Paso 4: validar y verificar

Es importante verificar que el modelo realizado cumpla con las especificaciones dadas y además, está hecho de la mejor manera.

La validación permite saber si ha construido el sistema correcto, de conformidad con los requerimientos solicitados. Una de las maneras de realizarlo, tomando los requerimientos uno a uno, y verificando si el modelo es capaz de satisfacer dicho requerimiento.

Verificar permite saber si el sistema está construido correctamente, de la mejor manera, una de las técnicas que se puede aplicar para “verificar” es aplicar las formas normales, usualmente hasta la tercera forma normal - 3FN suele ser suficiente.

Buenas prácticas

Paso 5: optimizar estructuras internas

El quinto paso contiene recomendaciones, las cuales se recomienda leer y aplicar en la medida de lo posible, y son:

- De acuerdo con el objetivo de su modelo, identifique las entidades que tendrán la mayor carga, o de las cuales es espera una respuesta rápida y analice la posibilidad de crear índices de búsqueda.
- Por ejemplo, si es una base de datos de clientes, será útil crear índices de acuerdo a los criterios de las búsquedas, como el nombre y/o apellido.

- Si se tiene un sitio web de búsquedas de productos, será útil crear índices de acuerdo a los criterios de búsqueda, como la categoría de un producto, el color del producto, el nombre, etc.
- Aplicar índices del tipo *unique* permite asegurar que no existirán valores repetidos, lo cual puede ser muy útil en los siguientes casos:
    - Número de identificación de una persona, por ejemplo, número de cédula o DPI.
    - Nombre de usuario
    - Registros mediante correo electrónico
    - Etc.

3.2. Estructura de la metodología

En resumen, la metodología presentada contiene la siguiente estructura, la cual puede tomarse como una guía rápida para aplicar a los modelos de bases de datos creados:

- Analizar requerimientos
    - Objetivos de la base de datos
        - Especialización
        - Alta carga
        - Integridad
        - Baja latencia
        - Rápida respuesta
        - Estabilidad
    - Reglas de negocio
        - Entidades (que se debe almacenar y que no)
        - Dominio de los datos, tipos de datos

        - Papel usuarios
        - Acceso a la información (índices)
        - Integridad

- Escoger nomenclatura adecuada
    - Validar y verificar
        - Formas normales
        - Entidades, relaciones y atributos
        - Tipos de datos
        - Redundancia
    - Optimizar estructuras internas
        - Índices
        - Políticas retención datos, históricos

CONCLUSIONES

1. Definir una metodología ha facilitado establecer los pasos a seguir para hacer un diseño eficiente de bases de datos; los resultados muestran que definir una metodología específica para los proyectos de bases de datos de la Facultad de Ingeniería, facilita el aprendizaje y mejora en un promedio de 2 – 6 puntos la nota de promoción de laboratorio, y en un promedio de 13 - 16 puntos las notas de proyecto y prácticas. Ver apéndice 1.

2. Si bien se hace notorio el incremento de las notas finales y proyectos entre los semestres comparados, no se puede estimar de manera concluyente un porcentaje de mejora, se concluye que si existe una mejora significativa, pero no estimarla en puntos dado que se necesitaría hacer una comparación más exhaustiva.

3. Tomando como base las 12 reglas de integridad de Codd, se aplicaron pruebas de estrés para determinar que los siguientes factores tienen una relación directa con el rendimiento de una base de datos:

    - Campos autoincreméntales: los campos autoincreméntales afectan de manera directa y proporcional el rendimiento de manera negativa, a mayor cantidad de registros las pruebas de estrés mostraron que el tiempo de respuesta se incrementa.
    - Tipos de datos apropiados: cuando se utilizan apropiadamente los tipos de datos, estos afectan de manera positiva el rendimiento, la regla debe de ser utilizar siempre el tipo de dato más pequeño.

- Llaves foráneas: si bien las llaves proveen de integridad a la información contenida en la base de datos, afectan de manera negativa el rendimiento, si se puede proveer de integridad de manera externa por ejemplo mediante aplicaciones y/o validaciones del lado del usuario, se puede prescindir de las llaves.

En resumen, el performance se ve afectado por:

- Campos autoincrementales de manera negativa
- Tipos de datos apropiados de manera positivo
- Llaves foráneas de manera negativa

4. Las pruebas de estrés mostraron que el uso de campos autoincreméntales anulan el efectivo positivo que tiene el escoger adecuadamente los tipos de datos, es decir, que el efecto negativo que tienen sobrepasa el efecto positivo de los tipos de datos adecuados.

5. Las pruebas de estrés mostraron que la combinación de campos autoincrementares y llaves foráneas tienen un efecto contraproducente en el rendimiento, este puede minimizarse usando tipos de datos apropiados, pero la mejora no es notable.

RECOMENDACIONES

1. Si necesita realizar cargas masivas de datos sobre tablas con campos autoincrementales, se recomienda desactivar dichos campos, borrar los índices y al terminar la carga activar los campos y crear de nuevo los índices.

2. Evitar el uso de campos autoincrementales en tablas con grandes cantidades de información.

3. Para determinar el tipo de dato apropiado basta con saber el dominio de datos a almacenar y aplicar la regla de utilizar siempre el tipo de dato más pequeño.

4. Usar validación externa para la integridad de la información en la medida de lo posible.

5. Evitar la mezcla de campos autoincrementales y llaves foráneas, ya que juntos tienen un efecto negativo mayor que cada uno por separado.

6. Evitar la mezcla de campos autoincrementales y tipos de datos apropiados, ya que los campos autoincreméntales neutralizan el efecto positivo de uso apropiado de los tipos de datos.

7. El uso de campos no autoincrementales y tipos de datos apropiados es la mejor opción si se busca el máximo rendimiento.

BIBLIOGRAFÍA

1. CAPARRINI, Fernando. *El modelo de datos entidad-relacion E/R* [en línea]. Sevilla, España. Departamento de Ciencias de la Computación de la Universidad de Sevilla. <http://www.cs.us.es/cursos/bd-2003/HTML/modeloER.htm> [Consulta: 29 de junio de 2012].

2. ENCALADA, Rodolfo. *¿Eficiencia o eficacia?* [en línea]. <http://habilidadesgerenciales.bligoo.com/content/view/311036/Eficiencia-o-Eficacia.html> [Consulta: 15 de julio 2012].

3. ENTERPRISE Architect. *Data modeling: from conceptual model to DBMS* [en línea]. Sparx Systems, 2011. <http://www.odbms.org/wp-content/uploads/2013/11/Data_Modeling_ConcepttoDBMS.pdf> [Consulta: 23 de septiembre de 2012].

4. FINKELSTEIN, Martin S. *Physical Database Design for Relational Databases* [en línea]. IBM Almaden research center. <http://ece.ut.ac.ir/classpages/F84/AdvancedDatabase/Paper/DB_Paper/p91-finkelstein.pdf> [Consulta: 1 de octubre 2012].

5. GANGULY, Rupak. *How to: stress test/load test databases* [en línea]. <http://blog.webintellix.com/2009/07/how-to-stress-test-load-test-databases.html> [Consulta: 14 de septiembre de 2012].

6. GARVEY, Robert. *Online database: design and optimización* [en línea]. Kansas, Estados Unidos. Witan Inc. <http://www.openmpe.com/cslproceed/HPIUG82/P43.pdf> [Consulta: 23 de octubre 2012].

7. HALPIN, Terrence Aidan. *Database schema transformation and optimizacion* [en línea]. <http://link.springer.com/chapter/10.1007/BFb0020532#page-2> [Consulta: 17 de septiembre 2012].

8. KROENKE, David. *Procesamiento de bases de datos: fundamentos, diseño e implementación*. Naucalpan de Juárez, México: Pearson Prentice Hall, 2010. 220 p.

9. MEJIA, Elías. *Metodología de la investigación científica* [en línea]. Lima, Peru. <http://es.scribd.com/doc/56942915/Libro-Metodologia-de-La-Investigacion-2005> [Consulta: 3 de septiembre 2012].

10. NETXELL Mycrosystems. *Importancia del análisis y diseño orientado a objetos* [en línea]. <http://www.netxell.com/home/index.php?option=com_content&view=article&id=14%3Aimportancia-del-analisis-y-diseno-orientado-a-objetos&catid=1%3Ablog&Itemid=6&lang=> [Consulta: 26 de agosto de 2012].

11. NIELSEN, Paul. *Optimization theory* [en línea]. Washington, Estados Unidos. <http://www.sqlserverbible.com/files/optimizationtheory> [Consultado: 16 de octubre de 2012].

12. RAMAKRISHNAN, Raghu. *Database management system: physical database design* [en línea]. <http://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/slides/slides3ed-english/Ch20_DB_Tuning-95.pdf> [Consulta: 5 de septiembre de 2012].

13. SCHNAITTER, Karl. *On-line index selection for physical database tuning*. UMI Dissertation Publishing, 1996. 610 p.

14. SPARKS, Geoffrey. *Database modelling in UML* [en línea]. Australia. <http://www.sparxsystems.com/downloads/whitepapers/Database_Modeling_In_UML.pdf> [Consultado: 12 de agosto de 2012].

15. STEPHENS, Jon. *Beginning MySQL Database design and optimization: from novice to professional.* Nueva York, Estados Unidos: Apress, 2004. 190 p.

16. TECHNET. *Int, bigitn, smallint y tinyint (transact-sql)* [en línea]. <http://technet.microsoft.com/es-es/library/ms187745.aspx> [Consulta: 5 de octubre de 2012].

17. TEOREY, Toby. *Database modeling and design.* 5ta edición. Michigan, Estados Unidos: Publicaciones Morgan Kaufmann. 2011. 352 p.

18. TOMAN, David. *Database tuning and physical design: query optimization, index selection, and schema, query and transaction tuning* [en línea]. Waterloo, Canada. <https://cs.uwaterloo.ca/~david/cs348/lect-TUNING.pdf> [Consulta: 6 de octubre de 2012].

19. Universidad de Granada. *Fundamentos de diseño de bases de datos: modelado de datos* [en línea]. <http://elvex.ugr.es/idbis/db/docs/intro/C%20Modelado%20de%20datos.pdf> [Consulta: 2 de noviembre 2012].

APÉNDICES

Apéndice 1: Comparativa de notas finales de laboratorio aplicando la metodología propuesta.

Sistemas de Bases de Datos 1 (Notas de Laboratorio)				
	Aprobados	**Reprobados**	**Promedio Nota Final**	**Promedio Proyectos**
1er Semestre 2012	41	7	58.55	84.44
2do Semestre 2012	27	10	53.79	65.4
		Promedio	56.17	74.92
Aplicando la metodología				
1er Semestre 2013	9	29	62.94	91.24
		Mejora	6.77	16.32

Sistemas de Bases de Datos 2 (Notas de Laboratorio)				
	Aprobados	**Reprobados**	**Promedio Nota Final**	**Promedio Proyectos**
Diciembre 2012	33	5	57.41	88.2
Aplicando la metodología				
1er Semestre 2013	5	19	60.19	74.65
		Mejora	2.78	13.55

Fuente: reportes finales de auxiliaturas de los cursos Bases de Datos 1 y Bases de Datos 2, del primer semestre 2012 al primer semestre 2013 y curso de vacaciones diciembre 2012.