ESPECIFICACIÓN DE TEMPORALIDAD EN MODELOS CONCEPTUALES PARA BASES RELACIONALES Y ORIENTADAS A OBJETOS.

Temporality specification in models for relational and object oriented databases

RESUMEN

El grupo de investigación GIA del programa Ingeniería en Sistemas y Computación de la Universidad Tecnológica de Pereira, en torno al proyecto "*Marco de referencia para la gestión conceptual del cambio ontológico*", está trabajando en la extensión de la especificación de modelos conceptuales de datos, para que incluyan en su expresividad los aspectos de temporalidad, seguridad y la representación de metadatos.
Cuando se construye un modelo que representa un problema del mundo real, las caracteristicas estáticas y dinámicas pueden ser especificadas obteniendo un modelo conceptualmente equivalente; sin embargo, en algunas ocasiones es necesario representar cambios e implementar los cambios del sistema conservando registros históricos de eventos ocurridos. Este artículo explora como especificar temporalidad en modelos conceptuales de bases de datos relacionales y orientados a objetos.

PALABRAS CLAVES: Temporalidad, modelos conceptuales, bases de datos, orientación a objetos

ABSTRACT
GIA, as a research group in the Systems and Computing Engineering of the Universidad Tecnológica de Pereira, around de project "Framework for management of ontological changes", is working in the specification of data models that see the time as an important variable, so the data security and the metadata representation to think in a conceptual.
When you talk about a problem in the real world, the static and dynamic characteristics could be specified obtaining and conceptually equivalent model, all the time you include the time as a variable that let you to implement the changes in the system preserving historic records of the last events. This paper explore how is the specification of temporality in models for relational and object oriented databases.

KEYWORDS: Temporality, conceptual models, databases, object oriented systems

CARLOS AUGUSTO MENESES ESCOBAR
Ingeniero de Sistemas
Profesor Auxiliar
Universidad Tecnológica de Pereira
cmeneses@utp.edu.co

Jorge Alberto Gálvez Correa
Ingeniero Eléctrico
Profesor Auxiliar
Universidad Tecnológica de Pereira
jgalvez@utp.edu.co

Julio César Chavarro Porras
Ingeniero de Sistemas
Profesor Asistente
Universidad Tecnológica de Pereira
jchavar@utp.edu.co

1. INTRODUCCION

El modelado de los aspectos dinámicos de un sistema, usualmente conlleva el manejo de datos, comúnmente soportados en Sistemas de Gestión de Bases de Datos (SGDB). En éste proceso de modelado, las operaciones que implican cambios en la base de datos pueden requerir de un tratamiento especial, debido a que si no se hace un registro histórico de los cambios, entonces no se podrá proporcionar información sobre eventos pasados que guardan relación con el instante de tiempo en que han ocurrido.

Lo anterior implica la necesidad de tener que incluir características temporales en los elementos que forman parte del modelo conceptual, extendiéndolo para que su expresividad sea más acorde a los requerimientos del sistema. Las bases de datos relacionales y las orientadas a objetos, buscan especificar de una forma especializada y exhaustiva la formulación de un esquema conceptual dentro de un dominio dado.

El diseño de una base de datos es un proceso complejo que abarca decisiones a muy distintos niveles: desde las técnicas de abstracción de los datos para identificar y representar los aspectos que se enmarcan en el dominio del problema, hasta el diseño e implementación del modelo físico.

Aunque en los modelos conceptuales es posible representar la temporalidad, es necesario tratar en detalle el proceso de transformación de un modelo conceptual al

Fecha de Recepción: 25 de Enero de 2011
Fecha de Aceptación: 15 de Abril de 2011

modelo lógico, para evitar pérdida de la semántica del modelo.

Por ejemplo, la representación de los aspectos temporales en modelos orientados a objetos, generalmente son tratados como en los modelos relacionales, sin tener en cuenta apectos dinámicos de la orientación a objetos. En esta dirección se viene trabajando (por ejemplo, en la Universidad Politécnica de Valencia España [7], entre otros) en la definición de una extensión temporal al método "OO-Method" para producción automática de software a partir de modelos conceptuales orientados a objetos.

La potencialidad de los datos temporales es notable, ya que puede facilitar el acceso a los estados actuales y anteriores del sistema, proporcionar información de la evolución histórica. Lo anterior permite analizar, planificar y construir soluciones que estén orientadas al soporte de decisiones.

Este artículo presenta, en la sección 2 una revisión temática de los conceptos generales del modelado de bases de datos, en la sección 3 se presentan los conceptos de granularidad temporal, la sección 4 propone los aspectos temporales que afectan un modelo conceptual. La sección 5, plantea algunas directrices para la etapa de implementación. Finalmente, la sección 6 presenta algunas conclusiones y trabajos futuros.

2. MODELADO DE BASES DE DATOS

Una base de datos es básicamente un conjunto de datos en el dominio de un problema, los cuales están, organizados y relacionados para responder a los requerimientos de información en el mismo dominio. Las bases de datos (BD) generalmente hacen parte de un entorno computacional, conocido como el Sistema de Gestión de la Base de Datos (SGDB), el cual además de los datos del dominio contiene datos que describen a la base de datos (se conocen como metadatos), el software de gestión (que permite definir y manipular los datos) y los usuarios del sistema.

Las BD y los SGBD surgen como solución a los problemas que se presentan cuando los datos son manejados a partir de sistemas con ficheros; entre los problemas que podemos destacar, se encuentran la redundancia de los datos, la dependencia (lógica y fisica) de los datos respecto a los programas, y el control de la manipulación de datos y las consultas que son dependientes de los programas de aplicación..

Diseñar una base de datos consta de 3 fases: diseño conceptual, diseño lógico y diseño físico. Un *esquema conceptual* es una descripción de alto nivel de la estructura de la base de datos por medio del cual se abstraen las características de los elementos que forman parte del dominio del problema en el mundo que se quiere modelar, independientemente del SGBD que se vaya a utilizar para manipularla.

En un *modelo conceptual* se utiliza un lenguaje para describir esquemas conceptuales con un alto nivel de abstracción. El objetivo del diseño conceptual es describir el contenido de información de la base de datos y no las estructuras de almacenamiento que se necesitarán para manejar esta información. [7]. El modelo más usado es el *modelo entidad – relación*.

El modelo Entidad – Relación (ER) fue introducido por Peter Chen en 1976 [9]. Este modelo está formado por un conjunto de conceptos que permiten describir la realidad mediante un conjunto de representaciones gráficas y lingüísticas. La base del modelo son el concepto de entidad, de relación (entre las entidades) y atributos. Se agregaron además como extensión del modelo [1] (en 1989 Codd) los conceptos de atributos compuestos y jerarquías de generalización (modelo entidad relación extendido).

Uno de los problemas más comunes que enfrenta el diseñador de la base de datos con marcas temporales, es determinar el grado de una relación, el cual corresponde al número de entidades que participen de ella y el cual puede variar. A veces no es fácil determinar qué tipo de relación tienen un par de conjuntos de entidades o clases, pues depende del contexto del problema y suele confundirse el diseñador al no tener claro hasta que punto el esquema permite modelar instancias temporales. Como distinguir cuando la cardinalidad de una relación varia, dependiendo si se tiene en cuenta o no el comportamiento histórico de la relación.

En que situaciones es o no equivalente que una relación de muchos a muchos se puede modelar como dos relaciones de 1:n?, como también que una relación 1:1 en algunos casos puede desaparecer? El diseño lógico que parte del esquema conceptual da como resultado un *esquema lógico*, el cual es una descripción de la estructura estática de la base de datos en términos de las estructuras de datos que puede procesar un tipo de SGBD.

Un *modelo lógico* usa un lenguaje para especificar esquemas lógicos (modelo relacional, modelo de red, etc.). En los *modelos lógicos*, las descripciones de los datos tienen una correspondencia sencilla con la estructura física de la base de datos.

Algunos de los Modelos logicos son: jerarquico, lista invertida, red, relacional, modelo objeto, objeto relacional. En general todo modelo de datos se compone de estructuras, operadores y un conjunto de restricciones que garantiza su consistencia. A nivel lógico, vale la pena preguntarse si la parte estática y dinámica puede ser representada conservando la expresividad de la abstracción, en un modelo Relacional que permita de

forma eficiente hacer un manejo adecuado através de operaciones de persistencia.

En los sistemas de gestión de bases de datos orientados a objetos, un objeto se concibe como una abstracción del mundo real (en su parte estática similar a una entidad) y consta de un Oid (que puede ser la llave primaria), atributos propios del objeto y un conjunto de métodos que corresponden al comportamiento de un objeto.

El modelo Objeto – Relacional (OR). Este modelo se concibe a partir del modelo orientado a objetos y el modelo relacional a comienzos de los años 90. El modelo OR introduce a nivel lógico una mayor expresividad permitiendo modelar espectos dinámicos de los objetos. Cómo se podría garantizar que el modelo lógico conserve los aspectos dinámicos inherentes al modelo conceptual? Este interrogante abre la puerta para pensar en la formulación de una metodología que extienda y aumente la expresividad de los modelos existentes.

El SGBD se encarga de administrar el esquema físico el cual corresponde a una descripción (estructura de almacenamiento y métodos utilizados) de la implementación de una base de datos en memoria, y se interactúa con él, a través de un lenguaje de definición de datos (DDL).

3. GRANULARIDAD EN LAS MARCAS TEMPORALES

La granularidad, es entendida en el contexto de las bases de datos, como la duración del punto de tiempo que se modela.

La temporalidad en bases de datos, tiene que ver con la especificación de propiedades históricas que manejan los elementos que se representan en el dominio del problema. Todo elemento de un modelo de datos es factible de tener un comportamiento evolutivo que hay que registrar para poder tener información actual e histórica, que es la base de la planificación futura. El artículo referenciado en [7], sirve de base para documentar la especificación de temporalidad aquí expuesta.

Para capturar la expresividad temporal se tienen 4 formas distintas de describir la granularidad en las marcas temporales [5]:

a. *Instante temporal*. Representa un punto en el tiempo.
b. *Intervalo temporal*. Definido por 2 instantes temporales
c. *Elemento temporal*. Unión finita de intervalos temporales.
d. *Duración temporal*. Periodo de tiempo que no tiene en cuenta instantes. Por ejemplo: 1 hora (60 minutos), un mes (28 a 31 dias). [3]

Adicional a las anteriores, se incluyen otras formas para poder describir esta granularidad en las marcas temporales, entre las cuales se mencionan:

e. *Colección de instantes temporales*. Representa la unión finita de instantes temporales, permitiendo llevar un registro histórico no contínuo sino discreto de un elemento que en el tiempo va evolucionando o es sometido a cambios. Un ejemplo, es cuando se guarda el registro fotográfico de una persona o cualquier paisaje, donde para notar cambios hay que tomar instantes en el tiempo.
f. *Conjunto de Duración temporal*. Representa la unión finita de periodos de duración temporal. Sirve para llevar registro histórico de eventos ocurridos sin importar la especificación de los instantes. Por ejemplo se quiere representar para un jugador de futbol, cuantos minutos ha jugado en cada partido de futbol sin importar en que periodo específico.

En la medida de lo posible, es aconsejable utilizar el intervalo y el elemento temporal en lugar de la duración y el conjunto de duración temporal respectivamente, para ofrecer mayor expresividad.

4. TEMPORALIDAD EN MODELOS CONCEPTUALES

La temporalidad en un modelo conceptual se puede hacer en cinco niveles diferentes de especificación: [7]:

a. Clase
b. Atributo
c. Relación de Agregación
d. Relación de Herencia
e. Relación de agente activador de servicios (en modelos OO)

a. Clase. Contiene un registro histórico de la evolución de sus objetos. Se asocia a cada objeto un elemento temporal como una lista de intervalos en el cual el último se cierra en el momento de eliminar el objeto.

b. Atributo. Permite guardar los cambios de valor a lo largo de la vida del objeto. Se asocia al atributo una lista de pares (valor, intervalo temporal). E*jemplo*: Se tiene una clase de objetos *Empleado* declarada temporal, entonces cada instancia de la clase se asocia a distintos intervalos de existencia.

codigo	*Nombre*	*Existencia*	*Salario*
c1	Jaime Diaz	[2, 8], [15, null]	500 [2,6], 620 [7,8], 750 [15,null]
C2	Maria Perez	[6,null]	750 [6,10], 830 [11,null]
C3	Luis Ramirez	[5,17]	780 [5,11], 870 [12,17]

Tabla 1. Clase *Empleado* y atributo *salario* temporales.

c. Relación de agregación (*Parte-de*). Se mantiene la historia de la relación en el tiempo, guardando información sobre los intervalos en los que un objeto está compuesto de (o forma parte) de otros.

d. Relación de Herencia (*es-un*). Conserva un registro histórico de los intervalos de existencia asociados a los roles que asumen los objetos de la clase especializada.

e. Relación de Agente. Captura los instantes de tiempo en que un agente (objeto del sistema que actúa como activador de servicios) activa un servicio o transacción. El resultado es un "log" que permite a los usuarios controlar la ejecución del sistema.

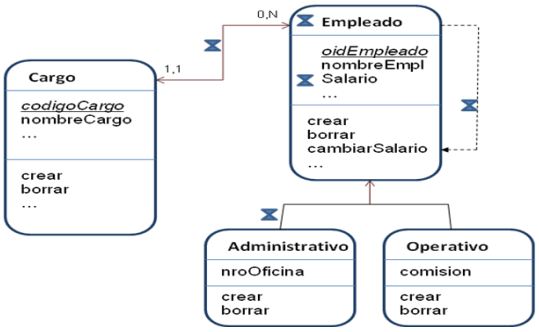

Diagrama 1. Niveles de temporalidad

5. IMPLEMENTACIÓN DE LA TEMPORALIDAD EN EL DISEÑO DE LAS BD

La introducción de aspectos temporales en el modelo orientado a objetos, conlleva tener que tomar algunas decisiones como son las de representar los cambios de estado de los objetos en una misma tabla o separar los datos históricos de los datos actuales evaluando las implicaciones que se derivan en cada caso. En [7], se hace un tratamiento a los datos históricos separándolos de los actuales, de tal forma que para cada nivel de temporalidad en el modelo conceptual, se da un tratamiento que permite conservar la expresividad de estos aspectos.

a. Clase. Una clase temporizada, se debe representar con una "tabla base" (donde se incluye para cada objeto el inicio de existencia) y se asocia otra "tabla histórica", donde se incluyen las instancias eliminadas de la tabla base y en la que se cierra el intervalo de existencia de cada objeto.
Tabla base: (oid, {atributos})
Tabla histórica: (oid, inicioExistencia, finExistencia, {atributos}) Si es modelo conceptual porque tablas?

El intervalo de existencia de un objeto histórico está determinado por [inicioExistencia, finExistencia], correspondiente al periodo desde que se crea el objeto y hasta el instante en que ocurre el evento de destrucción del mismo.
La generación de una instancia en la tabla histórica se puede implementar:

- Logrando que el servicio *new* en la clase base, ejecute el *insert* en la tabla histórica (No se requiere inicioExistencia en la tabla base) y que el servicio *destroy* en tabla base ejecute la *actualización* en la tabla histórica. [7]

- Ejecutando el servicio *insert* en la tabla histórica cuando ocurra el evento *destroy* en la tabla base (en este caso habria que incluir inicioExistencia como parte de los datos en esta tabla) y que el servicio *destroy* en tabla base ejecute igualmente la *actualización* en la tabla histórica.

Es de importancia evaluar y definir cuál debe ser el tratamiento para las llaves primarias y foráneas de tal manera que se mantenga la integridad referencial de la base de datos. El tratamiento de temporalidad para objetos que tienen restricción existencial con respecto a otros, requieren un manejo especial para no violar la integridad de la base de datos. En cualquier caso, se puede definir un disparador ("trigger", si lo permite el SGBD) asociado a la inserción y otro al borrado de tuplas.

b. Atributo. Un atributo especificado como temporal, genera en la base de datos la creación de una "tabla histórica de atributo", asociada a la tabla base de la clase (la cual se debe tratar como temporal si no ha sido especificada como tal). Solo se pueden especificar como temporales, atributos variables.
En los atributos se pueden tener distintos tipos de marcas temporales dependiendo de la representación que de ellos se tenga.

En la tabla base se agrega el instante de inicio de existencia para un valor del atributo temporal, y en la tabla histórica del atributo se cierra ese intervalo de existencia.

Tabla base: (oid, atributoTemp, {otros atributos})
Tabla histórica Atributo: (oid, atributoTemp, inicioExistenciaAt, finExistenciaAt).

Donde *oid* es la llave primaria del objeto, *atributoTemp* es el atributo temporizado y el intervalo de existencia del atributo está dado por [inicioExistenciaAt, finExistenciaAt].

El tratamiento de temporalidad en un atributo se puede implementar de varias maneras:

- Cuando se crea el objeto (*new* en la tabla base), el inicio de existencia para cada uno de los atributos temporales, es similiar al de este. En este caso o por

cada servicio que modifique el valor de un atributo temporal, se abre el intervalo de existencia para el nuevo valor del atributo generando una instancia (servicio *insert*) en la tabla histórica de ese atributo. El servicio *destroy* en tabla base y cada servicio que modifique al atributo, ejecuta la *actualización* en la tabla histórica de atributo cerrando el intervalo de existencia para el valor anterior. [7].

- Al crear el objeto (servicio *new* en la tabla base), o al activar cada servicio que modifique el valor del atributo temporal, se ejecuta la *actualizacion* del inicio del intervalo de existencia para nuevo valor del atributo (en este caso si se requiere *inicioExistenciaAtributoTemp* en la tabla base). Al borrar el objeto (servicio *destroy* en la tabla base) o al ejecutar cada servicio que modifique el valor del atributo temporal, se genera una instancia (activa el servicio *insert*) en la tabla histórica del atributo conteniendo el valor anterior.

- Otra forma de tratar el manejo de atributos temporales (analizando sus implicaciones), es teniendo solo la tabla histórica de clase y cada modificación en un atributo temporal o la destrucción del objeto, ejecuta el servicio *insert* en la tabla histórica de la clase y en ese caso se guarda un estado del objeto cerrando su intervalo de existencia, habria que tener un atributo para representar el intervalo de existencia por cada atributo temporal en la tabla base.

c. Relaciones de Agregación (parte-de). Una relación de agregación temporal, se implementa como una restricción de clave ajena entre las tablas base de las clases relacionadas. Hay que distinguir los tipos de relaciones M:M, 1:M y 1:1 para determinar donde ubicar la clave ajena (en la compuesta o en la componente).

Para recoger los intervalos de tiempo en los que se mantiene la relación, se crea una "tabla histórica de agregación", asociada a las tablas históricas de las clases relacionadas (las cuales se deben tratar como temporales si no han sido especificadas como tal), incluyendo la identificación de las instancias involucradas, con el intervalo de existencia de dicha relación. Las tablas base de las clases no se ven afectadas.

Tabla histórica Agregación:
(oidObjeto1, oidObjeto2, … inicioExistenciaAgregacion, finExistenciaAgregacion).

Donde *oidOBjeto1, oidObjeto2,..* son las llaves primarias de los objetos históricos involucrados, y el intervalo de existencia de las relación de agregación está dado por [inicioExistenciaAgregación, finExistenciaAgregación].

Se puede implementar la generación de una instancia en la tabla histórica de agregación de la siguiente manera:

- La activación del servicio *new* en la clase base de la relación donde están las claves ajenas y de cada servicio que afecte a alguna de estas, ejecuta el servicio *insert* en la tabla histórica de la agregación. El servicio *destroy* en tabla base y cada servicio que modifique alguna de las claves ajenas, ejecuta la *actualización* en la tabla histórica de agregación cerrando el intervalo de existencia para el valor anterior. [7]
- De la misma forma que con atributos temporales, otra forma de implementación consiste en ejecutar el *insert* de la tabla histórica de agregación cuando ocurra el evento de modificación de la clave ajena o *destroy* en la tabla base (donde está la clave ajena y en estos casos habria que incluir *inicioExistenciaAgregación* como parte de los datos en esta tabla).
- Considerando los posibles tipos de relaciones con sus cardinalidades mínimas y máximas, se tienen otras alternativas de solución (donde también hay que revisar sus implicaciones), en las que se tenga un tratamiento especial a las relaciones de agregación con restricción existencial, en las que probablemente se puede tener una generación automática en cascada de tablas históricas para las clases que cumplan con esas restricciones y además, este concepto se extiende a las clases temporales.

d. Relaciones de Herencia (es-un). En una relación de herencia temporizada, se trata como temporal la clase especializada y todas las clases (aunque no se hayan definido como tal) que se encuentran en la línea jerárquica ascendente hasta la raíz del árbol de herencia. Esto con el fin de garantizar que el objeto temporal especializado se trate como un todo y se recogen los intervalos de tiempo en los que las relaciones de herencia existieron. [7]

e. Relaciones de Agente. El tratamiento de una relación temporal de agente, se implementa con la creación de una tabla histórica de agente (o "log") que contiene la información histórica de las actividades de ejecución de servicio.

Debido a que la ejecución de un servicio es una actividad atómica, se utiliza como marca temporal el instante de tiempo en que ocurre dicho evento. Se debe tener un "log" por cada clase agente temporizada que se tenga,

Tabla histórica Agente:
(oidAgente, instanteEjecuciónServicio, claseServicio, servicioEjecutado, argumentos, resultado, mensaje).

Donde *oidAgente* es la identificacion del agente (cuya clase debe ser temporizada sino lo es), *claseServicio* es la clase a la cual pertenece el servicio, *argumentos* son los

valores de los atributos afectados, *resultado* indica si se realizó o no la transacción adecuadamente y *mensaje* es de error o de información.

Cuando un agente activa la ejecución de un servicio cuya relación es temporizada, se implementa haciendo que el servicio ejecutado active el *insert* de la tabla histórica de agente. Si la clase agente tiene especializaciones, en cascada cada una de ellas, automáticamente se vuelven temporizadas, pues cada especialización hereda el derecho a activar los servicios de sus ancestros [7]. El tratamiento para relaciones de agente expuesto anteriormente puese presentar algunos inconvenientes de manejo con los atributos incluidos como *argumentos* y *resultados* en el "log", debido a su dominio y cardinalidad.

Un tratamiento diferente de las relaciones de agente, que puede resolver este inconveniente consiste en especificar a nivel conceptual una extensión del modelo que incluya relaciones entre los servicios y las clases que estos afectan, pues probablemente hay que registrar históricamente los cambios en los objetos (cuyas clases deben ser temporizadas) afectados por servicios que son activados por agentes (y su relación es temporizada). Esto se puede lograr si se representa en el modelo conceptual una buena especificación de los metadatos.
Por lo tanto la activación de un servicio automáticamente guardaría un registro histórico de todos los objetos que afecta y su relación con la instancia creada en el "log" correspondiente.

6. CONCLUSIONES

Buscando una forma de representación de esquemas a nivel conceptual, el grupo GIA está en un proceso de formulación de un modelo conceptual capaz de permitir la esquematización de abstracciones en el dominio del problema enriquecido con el tratamiento temporal. El tratamiento de las propiedades temporales en el modelo de ejecución conlleva algunas decisiones que permiten obtener distintos enfoques en los cuales se busca conservar la expresividad de esas propiedades. Es de interés, poder revisar cuales son las implicaciones de los distintos enfoques y poder determinar cómo se puede aportar al mejoramiento de la representación de los aspectos temporales en el modelo de ejecución.

En este sendero se perfilan propuestas que permitan además plantear un modelo a nivel conceptual y orientado a objetos para el modelado de bases de datos en los que se introducen conceptos adicionales que enriquecen la expresividad del modelo en aspectos relevantes como son los metadatos, la seguridad y la temporalidad. Junto con el modelo se debe formular y desarrollar una metodología que incluya los aspectos anteriores mencionados y con la que se garantice no solo la seguridad sino también la integridad referencial de las bases de datos.

4. Referencias Bibliográficas

[1] Codd, E. F. "*The Relational Model for Database Management*". Addison - Wesley. 1990.
[2] Date, C. J. "*Introducción a los Sistemas de Bases de Datos*". Sistemas técnicos de edición S.A.. 1986.
[3] Edelweiss, N. "Bancos de Dados Temporais: Teoría e Prática". Report UFMG p 225, Joao Paulo Kitajima 1998.
[4] Gómez, J. "El Modelo Objeto – Relacional ORDBMS"
[5] Jensen, C. S. (Ed). A consensus glossary of temporal database concepts. ACM SIGMOD Record, New York, v.23 n.l, p52-64, Mar 94.
[6] Korth, H. F.; Silberschatz A.; Sudarshan S. "*Fundamentos de Bases Datos*". Mc Graw Hill. 1998.
[7] Meneses, C; Pastor, O; Molina, J; Insfrán, E. "*Especificación de Temporalidad en Entornos Automáticos de Producción de Software a Partir de Modelos Conceptuales Objetuales*". Revista "Computación y Sistemas" Vol 4 Nro 4 abr-jun 2001 Mexico
[8] Wiederhold, G. "*Diseño de Bases de Datos*". Mc Graw Hill. 1991.
[9] Chen, Peter. "The Entity Relationship Model - Toward A Unified View of Data".