DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA, UNAM
CURSOS ABIERTOS

CURSO: **CC110 Taller de Análisis y Diseño de Bases de Datos Relacionales**
FECHA: **2 al 11 de diciembre de 1996**

EVALUACIÓN DEL PERSONAL DOCENTE

(ESCALA DE EVALUACION 1 A 10)

CONFERENCISTA	DOMINIO DEL TEMA	USO DE AYUDAS AUDIOVISUALES	COMUNICACIÓN CON EL ASISTENTE	PUNTUALIDAD
ING. JESSICA BRISEÑO CORTES				

Promedio _____

EVALUACIÓN DE LA ENSEÑANZA

CONCEPTO	CALIF
ORGANIZACIÓN Y DESARROLLO DEL CURSO	
GRADO DE PROFUNDIDAD DEL CURSO	
ACTUALIZACION DEL CURSO	
APLICACIÓN PRACTICA DEL CURSO	

Promedio _____

EVALUACIÓN DEL CURSO

CONCEPTO	CALIF.
CUMPLIMIENTO DE LOS OBJETIVOS DEL CURSO	
CONTINUIDAD EN LOS TEMAS	
CALIDAD DEL MATERIAL DIDÁCTICO UTILIZADO	

Promedio _____

Evaluación total del curso_____

Continúa...2

1. ¿Le agradó su estancia en la División de Educación Continua?

SI [ ] NO [ ]

Si indica que "NO" diga porqué. _____

_____

2. Medio a través del cual se enteró del curso

Periódico *Excélsior*	
Periódico *La Jornada*	
Folleto anual	
Folleto del curso	
Gaceta UNAM	
Revistas técnicas	
Otro medio (Indique cual)	

3 ¿Qué cambios sugeriría al curso para mejorarlo?

_____

_____

_____

_____

_____

_____

4 ¿Recomendaría el curso a otra(s) persona(s) ?

SI [ ] NO [ ]

5. ¿Qué cursos sugiere que imparta la División de Educacion Continua?

_____

_____

_____

_____

_____

_____

6 Otras sugerencias

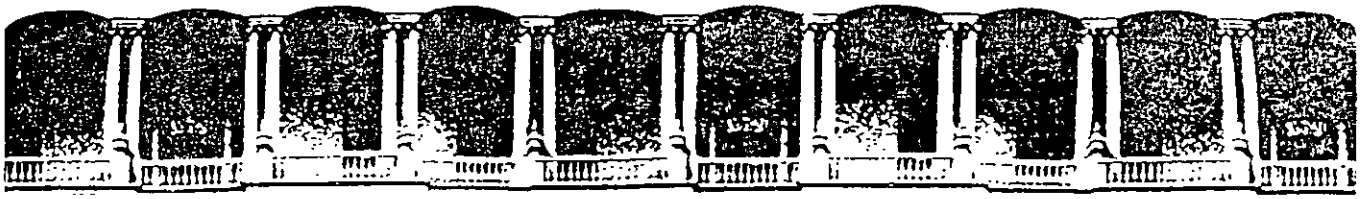
_____

_____

_____

_____

_____



FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

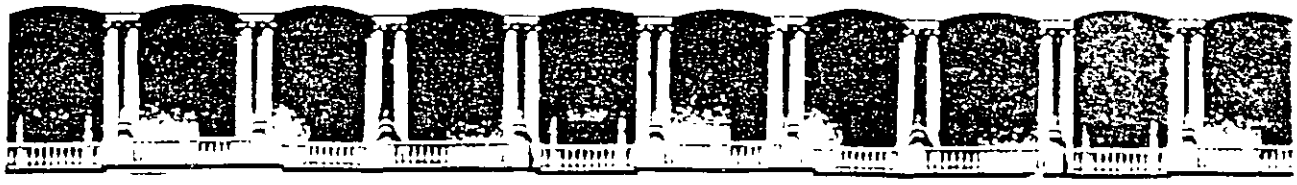
TALLER DE ANALISIS Y DISEÑO DE BASES DE DATOS RELACIONALES

2 al 11 de diciembre de 1996

DIRECTORIO DE PROFESORES

ING. JESSICA BRISEÑO CORTES
Norte 91 No. 4729
Col. Nva. Tenochtitlán
Delegación Gustavo A. Madero
C.P. 07890 México, D.F.
TEL: 751 92 56

'pmc.



FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

TALLER DE ANALISIS Y DISEÑO DE BASES DE DATOS RELACIONALES

Jéssica Briseño C.
J. Antonio Chávez F.

Palacio de Minería Calle de Tacuba 5 Primer piso Del. Cuauhtémoc 06000 México D.F. APDO. Postal M-2285
Teléfonos 510-6957 512-5121 521-7335 521-1987 Fax 510-0573 512-5121 521-4020 AL. 26

CONTENIDO

BLIBLIOGRAFIA

"OBJECT ORIENTED DATABASE DESING Concepts and Applicaton"
Kenmore S. Brathwaite
Academic Press

"CLIENT/SERVER COMPUTING"
Dawna Travis Dewire
McGraw Hill

"DATABASE SYSTEMS Principles, Design and Implementation"
Catherine M. Ricardo
Maxwell MacMillan International Editions

"ENTERPRISE DATABASE IN A CLIENT/SERVER ENVIORONMENT"
Michel M. Gorman
A Wiley-QED Publication

"ANALISIS Y DISEÑO ORIENTADO A OBJETOS"
James Martin, James J. Odell
Prentice Hall

"THE DATABSE BOOK"
Mary E.S. Loomis
Maxwell MacMillan International Editions

"FUNDAMENTOS DE BASE DE DATOS"
Henry F. Korth, Abraham Silberschatz
McGraw Hill, 2a. ed.

"INGENIERIA DE SOFTWARE"
Roger S. Pressman
McGraw Hill, 2a. ed.

Análisis: El Modelo Conceptual de Datos

OBJETIVO:

En este capítulo se estudiará la metodología utilizada para la Creación de un Modelo Conceptual de Datos en base a los requerimientos establecidos.

En este capítulo el asistente:

- Aprenderá a identificar las entidades a partir de los requerimientos de la empresa.

- Aprenderá a identificar las relaciones existentes entre entidades.

- Determinará los atributos de las entidades y relaciones.

- Identificará superentidades y subentidades.

- Identificará relaciones recursivas y N-arias.

- Conocerá los DER como herramienta gráfica para representar un

modelo de Bases de Datos.

MODELO CONCEPTUAL DE DATOS

El Modelo Conceptual de Datos o Modelo Lógico de Datos es el esquema resultado en la etapa de Análisis, que representa los requerimientos de datos del Sistema de Información que se esta modelando.

El Modelo Conceptual de Datos debe cumplir con las siguientes características:

Claro y preciso, esto es, debe de describir completamente los datos, la semántica de los mismos, sus relaciones, la seguridad, la integridad y los alcances del Sistema de Información que representa.

Constante, es decir debe contemplar requerimientos presentes y futuros para evitar cambios constantes en el modelo.

Flexible, debe de poderse adaptar a nuevos requerimientos sin tener que sufrir cambios significativos. Un buen modelo conceptual no debe permitir que cambios en el Modelo Conceptual afecten el Modelo de Vistas. El Esquema de Vistas debe ser derivable del Modelo Conceptual.

Integración de aplicaciones, el Modelo Conceptual debe de servir como base para el desarrollo de las múltiples aplicaciones que se den en el Sistema de Información, no solamente de aquellas que se proyecten en las etapas de planeación para el Sistema de Información en Bases de Datos, sino también para aquellas que se presenten durante toda la vida útil de la

Base de Datos.

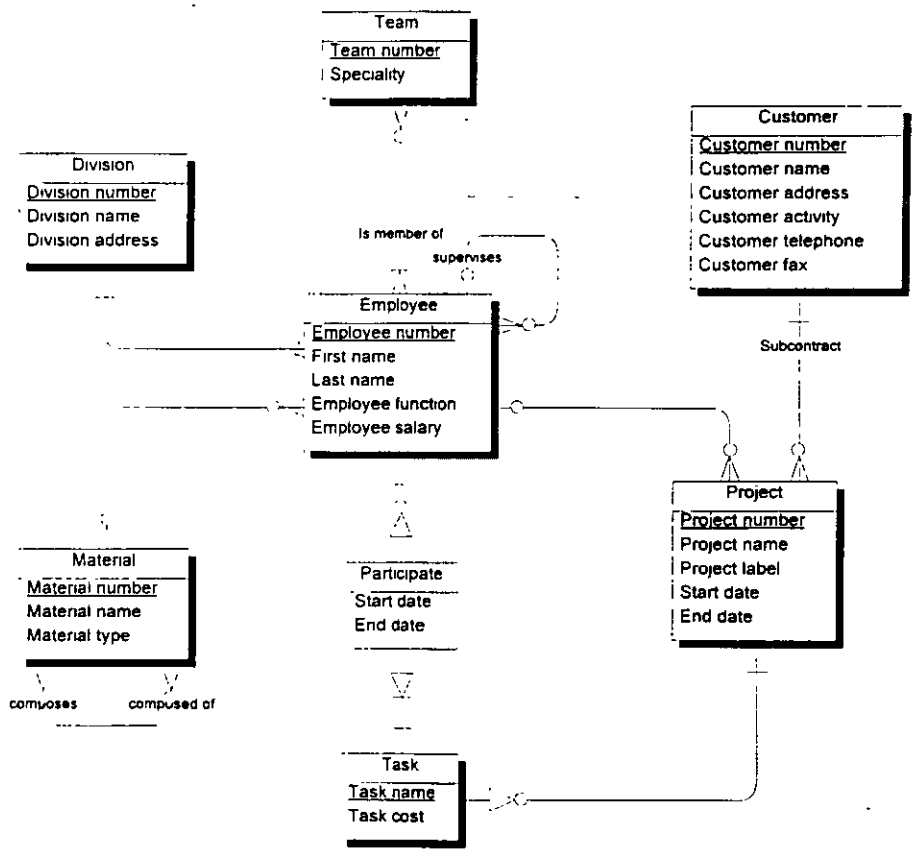
Independiente del software o hardware utilizado para su implementación.

MODELO ENTIDAD-RELACION

El Modelo Entidad-Relación fue desarrollado por Peter Chen en los 70´s.

El Modelo Entidad-Relación está basado en la percepción de un mundo real que se compone de un conjunto de objetos básicos llamados **Entidades**, descritos por sus **atributos**, y de **Relaciones** entre esos objetos.

El Modelo Entidad-Relación proporciona una ayuda gráfica conocida como Diagrama Entidad-Relación(DER), que permite la representación de la estructura lógica del modelo de datos.



El análisis para la generación del Modelo Conceptual de Datos debe tomar como entrada los Requerimientos del Sistema de Información de la empresa.

El procedimiento para la generación del Modelo Conceptual de Datos es el siguiente:

1. Seleccionar las Entidades y el tipo de entidad.

2. Seleccionar las Relaciones entre las entidades que están dentro del Alcance de Integración de la Empresa.

3. Determinar el tipo de relación entre las entidades.

4. Determinar el grado de asociación o cardinalidad entre las entidades.

5. Asignar atributos a las entidades y Relaciones.

ENTIDADES

Es un objeto que existe y que es distinguible de otros objetos.

Es algo (persona, lugar, objeto, concepto) a lo que la Empresa le reconoce poder existir en forma independiente y que puede ser definido en forma única.

Una entidad es identificable en forma única y tiene atributos que la describen.

Ejemplo:

PELICULA
CLIENTE
EMPLEADO
DEPARTAMENTO

Una instancia de una entidad es un elemento de ese tipo de entidad, por ejemplo: "Los gritos del silencio" es una instancia de PELÍCULA, "Sistemas" es una instancia de DEPARTAMENTO, etc.

Al conjunto de todas las instancias de una entidad se le denomina **conjunto de entidades** o **tipo de entidad**.

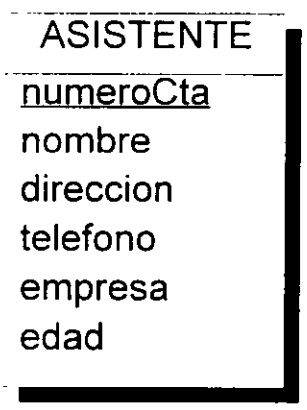
Por simplicidad se denomina **entidad** al conjunto de entidades.

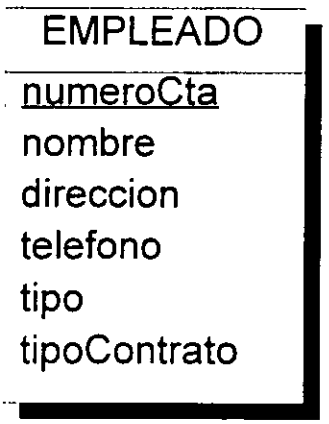
Representación de Entidades en un DER

Existen varias convenciones para la representación de DER´s. Para representar las entidades se utilizan rectángulos, opcionalmente, se pueden tener algunas variaciones en su representación:

- Identificados con el nombre de la Entidad.
- Identificados con el nombre de la Entidad, además de la lista de atributos.
- Rectángulos con esquinas redondeadas para indicar cierto tipo de entidad.

Ejemplos:





¿Como obtener las entidades?

Las entidades se obtienen del análisis de los requerimientos de la empresa. Cada una de las entidades tienen múltiples instancias las cuales deben poder ser identificadas unas de otras por medio de sus atributos.

Si las instancias no pueden ser identificadas unas de otras, entonces no se trata de una entidad.

Una entidad es descrita por más de un atributo.

Generalmente las entidades son sustantivos; pero no todos los sustantivos representan entidades.

Se deberán ignorar los sustantivos que:

- No tengan importancia para la empresa.
- Aquellos que denoten documentos que contienen información proveniente de otras entidades, por ejemplo: recibo de nomina, credencial de socio, etc. Se considerarán a estos como entidades cuando sea necesario mantener información propia del documento, como por ejemplo, número de folio, fecha de expedición, etc.

Ejemplo:

Una empresa de Capacitación y consultoría en sistemas imparte cursos. Para poder inscribirse a un curso de los publicados por dicha empresa, se debe enviar una solicitud de inscripción por cada participante en donde se indíque: nombre del asistente, nombre de la empresa en la que trabaja, puesto, dirección, teléfono. En la solicitud también se debe indicar el curso que solicita.

En la descripción anterior se puede identificar a la entidad ASISTENTE, el cuál es un objeto de importancia en el Sistema de Información que se esta modelando.

La entidad ASISTENTE tiene una serie de atributos que la describen: nombre, dirección, empresa, puesto, tel., etc.

Las instancias de ASISTENTE son todas las personas que envíen una solicitud de inscripción. Dichas instancias son identificables unas de otras por medio de sus atributos.

En este ejemplo, la solicitud no es una entidad, ya que no es un objeto que interese a la empresa como tal. Es importante la información que guarda, pero no la que la describe.

Determinación de la llave primaria

Por definición, toda ocurrencia de una Entidad debe ser identificable en forma única. Es por ello, que se debe encontrar un identificador para las instancias de la Entidad, es decir una llave.

Existen varios tipos de llaves:

Super llave. Es un conjunto de uno o más atributos que, tomados en conjunto, permiten identificar en forma única una instancia dentro del conjunto de Entidades.

Llave candidato. El concepto de Super llave no es suficiente, puesto que puede contener atributos extraños. Las Super llaves con el menor conjunto de atributos se conocen como llaves candidato y es posible que existan diferentes conjuntos de atributos que puedan servir como llaves candidato.

Llave primaria. Una llave primaria es una llave candidato que ha sido seleccionada por el diseñador de la base de datos como el medio de identificar las instancias de una entidad.

Las características necesarias para una llave primaria son las siguientes:

- Única
- Conocible en cualquier tiempo

Las características deseables de una llave primaria son las siguientes:

- Estable
- No descriptiva
- Pequeña y simple

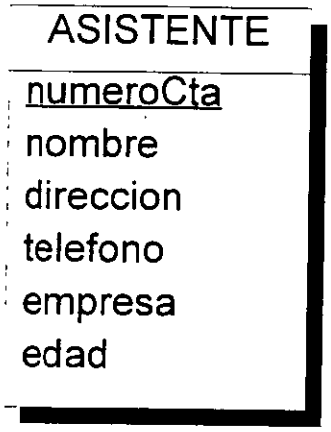
Cuando la llave primaria no cumple con las características mencionadas, se puede crear una llave primaria no natural, es decir, una que no haya sido obtenida de los atributos mencionados en los requerimientos de la empresa. Este tipo de llaves primarias no tienen significado alguno en el contexto de la empresa.

Ejemplo:

Para los requerimientos de la empresa de Capacitación y Consultoría en sistemas, la llave primaria de la entidad ASISTENTE podría ser nombre, dirección y edad. En lugar de la llave primaria natural se podría crear una: número de cuenta.

Representación de la llave primaria

Para representar la llave primaria en un DER, los atributos que la forman se subrayan:



Documentación de las entidades

Una vez que se han determinado las entidades, es necesario asignarles un nombre, para lo cual se pueden tomar en cuenta los siguientes puntos:

- Utilizar mayúsculas.
- Utilizar sustantivos en singular (EMPLEADO, DEPARTAMENTO, PELÍCULA, etc.).
- Eliminar homónimos.
- Describir el significado de la entidad en un Diccionario, utilizando ejemplos y contra-ejemplos.
- Indicar la llave primaria.

Ejemplo:

EMPLEADO

Llave primaria: númeroCta
Un empleado es para la empresa toda persona que tiene una relación laboral mediante un contrato suscrito mediante la Ley Laboral del Trabajo, por ejemplo: empleado de confianza, de medio tiempo, jubilados. No se consideran empleados a aquellas personas que presten sus servicios bajo pago por honorarios o bien empleados temporales o contratados por otra empresa.

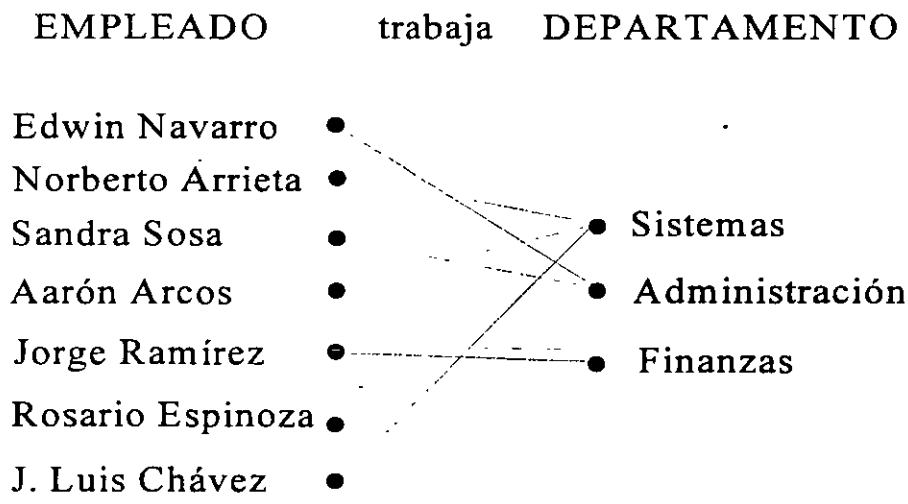
LABORATORIO 1

1. Leer la descripción de requerimientos que se incluye para el ejercicio 1, y hacer una lista de entidades. Especificar la llave primaria de cada entidad.

RELACIONES

Es el vínculo o conexión que existe entre dos o más Entidades.

Por ejemplo, la entidad DEPARTAMENTO puede relacionarse con la entidad EMPLEADO vía la relación **trabaja.**



Una instancia de la relación es un elemento de esa relación entre entidades, por ejemplo: Jorge Ramírez-trabaja-Finanzas.

Al conjunto de todas las instancias de una relación se le denomina **conjunto de Relaciones** o **tipo de Relación**.

Por simplicidad se denomina relación al "conjunto de Relaciones".

¿Como obtener las relaciones?

Las asociaciones o relaciones se obtienen del análisis de los requerimientos de la empresa.

Generalmente las relaciones son verbos; pero no todos los verbos representan relaciones.

Se deberán ignorar los verbos que:

- Denoten actividades que no tengan importancia para la empresa.
- Aquellos que denoten procesos, por ejemplo, "generar nómina".

Cardinalidad de la relación

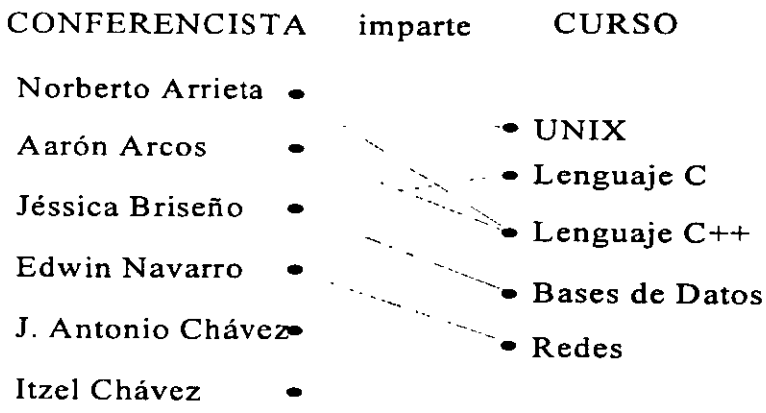
El grado de asociación o cardinalidad, representa en forma cuantitativa, el número de instancias de una entidad con las que puede estar asociada una instancia de otra entidad.

Una asociación puede tener tres tipos de cardinalidad:

TIPO		PUEDE INCLUIR			
1:1	(uno a uno)		1:0	0:1	1:1
1:N	(uno a muchos)	1:0	0:1	1:1	1:N
M:N	(muchos a muchos)	1:0	0:1	1:1	1:N
		N:1	M:N		

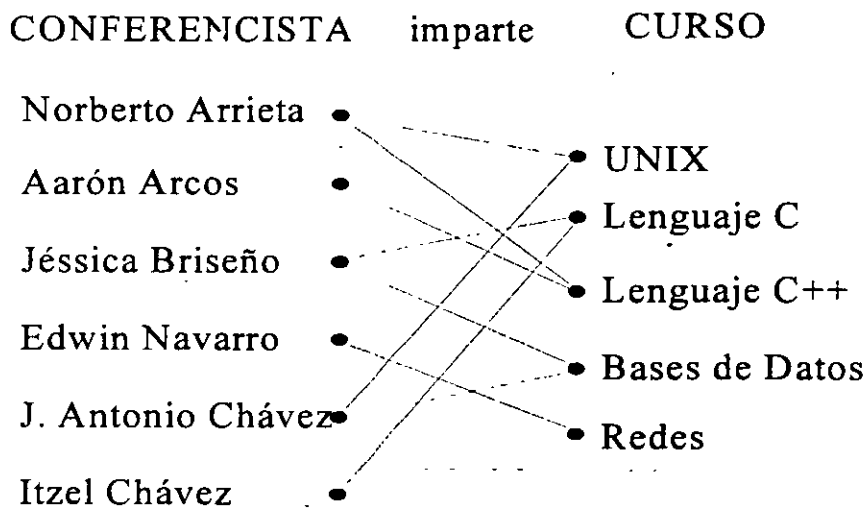
La cardinalidad de las relaciones refleja las reglas de empresa presentadas en los requerimientos de la misma. Las reglas de empresa son definiciones que se obtienen del análisis de los requerimientos. Por ejemplo:

"Un conferencista puede impartir muchos cursos"
"Un curso puede ser impartido por varios conferencistas"



La cardinalidad que incluye una categoría de relación puede ser reducida con reglas de empresa más estrictas, como:

"Un conferencista debe impartir cursos"
"Un curso debe ser impartido"



Cardinalidad Máxima

La cardinalidad máxima representa el número máximo de instancias con las que esta relacionada una instancia de una entidad.

Para determinar la cardinalidad máxima, primero se deberá leer la relación en una dirección y luego en la otra.

Ejemplo:

Para la relación **trabaja** entre EMPLEADO y DEPARTAMENTO:

1. Se lee la relación de EMPLEADO a DEPARTAMENTO tomando como referencia una instancia de EMPLEADO, para de esta forma obtener la cardinalidad máxima de EMPLEADO hacia DEPARTAMENTO: un empleado trabaja en **un** departamento.

2. Se lee la relación en el otro sentido, DEPARTAMENTO a EMPLEADO, tomando como referencia una instancia de DEPARTAMENTO, para de esta forma obtener la cardinalidad máxima de DEPARTAMENTO hacia EMPLEADO: en un departamento trabajan **muchos** empleados.

muchos **1**

La cardinalidad máxima es N:1

Cardinalidad mínima

La cardinalidad mínima representa el número mínimo de instancias con las que esta relacionada una instancia de una entidad.

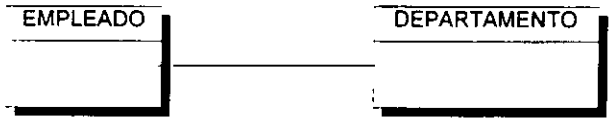
Las reglas de empresa que excluyen relaciones 1:0 se traducen como la obligatoriedad de que una entidad participe en una relación con otra Entidad. Cuando no se excluyen las relaciones 1:0, se trata de relaciones opcionales, sin obligatoriedad.

Una Entidad con obligatoriedad se puede considerar como una Entidad débil puesto que para existir depende de la existencia de otra Entidad.

El procedimiento utilizado para determinar la cardinalidad mínima de una relación es semejante al utilizado para la cardinalidad máxima, solamente que se plantea la obligatoriedad de la relación:

1. Se lee la relación de EMPLEADO a DEPARTAMENTO: un empleado esta obligado a trabajar en **un** departamento.

2. Se lee la relación en el otro sentido, DEPARTAMENTO a EMPLEADO: no hay obligatoriedad de que en un departamento trabajen empleados.



0 **1**

La cardinalidad mínima es 0:1

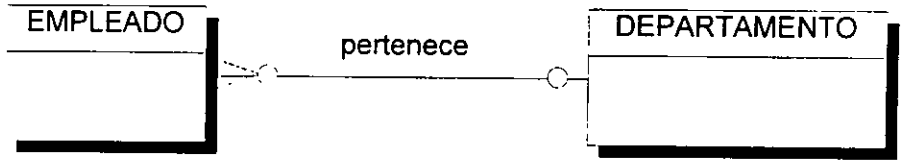
Relaciones N:1 (muchos a uno)

Una relación con cardinalidad máxima de muchos a uno tiene un grado de asociación de cero, uno o muchos en una dirección y de cero o uno en la otra.

Las relaciones N:1 son las más comunes.

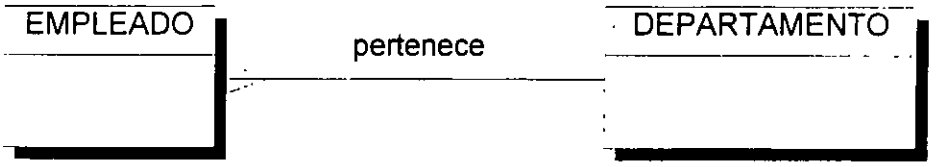
Ejemplo:

- Cardinalidad mínima opcional en ambas direcciones de la relación.



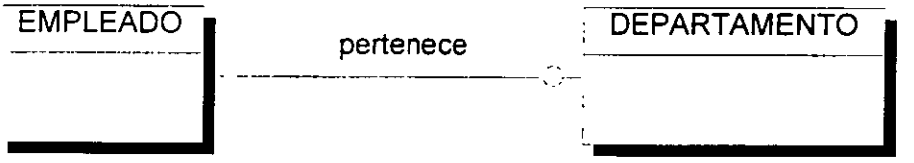
No todos los empleados trabajan o pertenecen a un departamento y existen departamentos en los que no trabaja ningún empleado.

- Obligatoriedad en ambos sentidos de la relación.

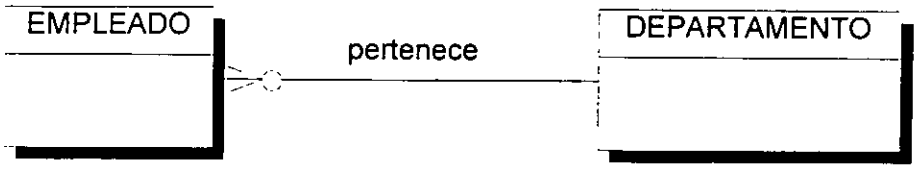


Todos los empleados pertenecen a uno y solo un departamento. Todos los departamentos cuentan con empleados, al menos uno.

- Cardinalidad opcional en el alguna de las direcciones.



Existen empleados que no pertenecen o trabajan en un departamento. Por otra parte, todos los departamentos cuentan con empleados, al menos uno.



Todos los empleados pertenecen a uno y sólo un departamento. Por otra parte, existen departamentos que no cuentan con empleados.

Relaciones 1:1 (uno a uno)

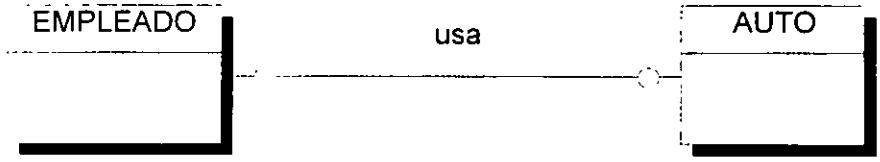
Una relación con cardinalidad máxima de uno a uno tiene un grado de asociación de cero o uno en ambas direcciones de la relación.

Relaciones con cardinalidad mínima obligatoria en ambos sentidos no son muy utilizadas ya que se pueden modelar como una sola entidad.

Ejemplo:

Una empresa cuenta con un lote de autos los cuales son asignados a los empleados.

- Cardinalidad opcional en ambas direcciones de la relación.

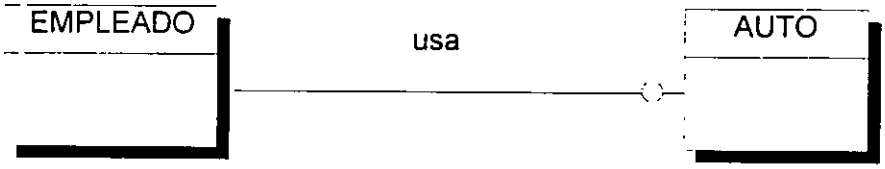


La empresa cuenta con autos los cuales no son utilizados por los empleados y solamente algunos empleados utilizan autos de la compañía.

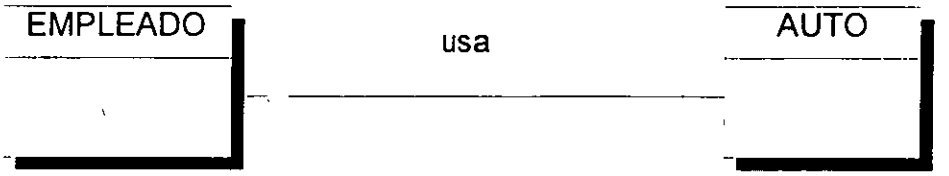
- Obligatoriedad en ambas direcciones de la relación.

Todos los empleados usan auto de la compañía. Por otra parte, los autos de la compañía solamente son utilizados por empleados. Cada auto es utilizado por un solo empleado y un empleado utiliza un solo auto.

- Cardinalidad mínima opcional en alguna dirección de la relación.



No todos los empleados usan autos de la compañía, pero todos los autos son utilizados por los empleados. Cada auto es utilizado por un solo empleado y los empleados que utilizan auto, solamente utilizan uno.



La empresa cuenta con autos para ser utilizados, uno por cada empleado y además cuenta con otros autos los cuales no son utilizados por empleados.

Relaciones N:N (muchos a muchos)

Una relación con cardinalidad máxima de muchos a muchos, tiene un grado de asociación de cero, uno o muchos en ambas direcciones de la relación.

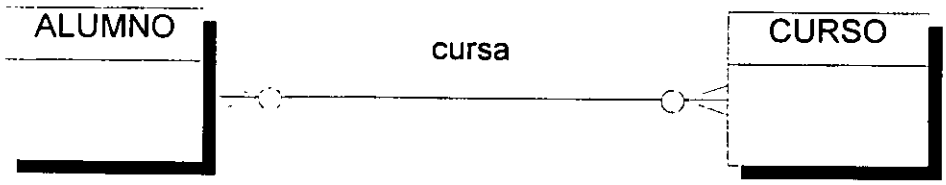
Las relaciones N:N son muy utilizadas, generalmente la cardinalidad mínima es opcional en ambas direcciones o bien, existe cardinalidad mínima obligatoria en alguna.

Este tipo de relaciones permiten implementar relaciones 1:N o 1:1 que requieren información de estados pasados, presentes y/o futuros.

Ejemplo:

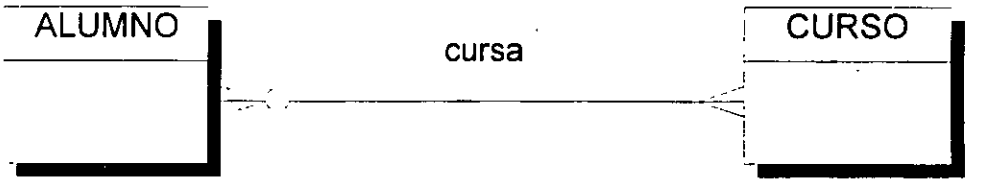
Una empresa de capacitación imparte cursos a empresas. La empresa mantiene la información de los cursos programados en un mes. Existen empresas que mandan a capacitar a sus empleados en varios cursos durante el mes.

- Cardinalidad mínima opcional en una dirección.



Hay alumnos registrados, que no se han inscrito a un curso. Por otra parte, hay cursos en los que no hay alumnos inscritos.

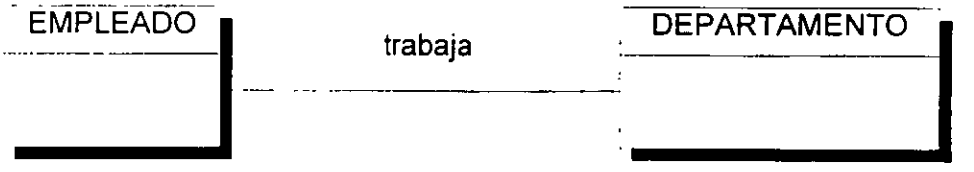
- Cardinalidad mínima obligatoria en alguna de las direcciones.



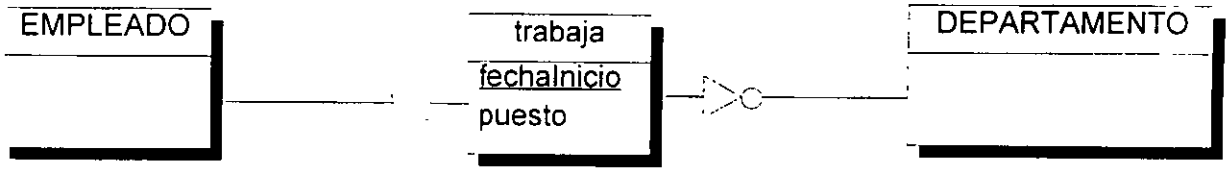
Para que un alumnos pueda estar registrado, necesita al menos estar inscrito a un curso.

- Representación de estados presentes y pasados.

Si para la relación EMPLEADO-trabaja-DEPARTAMENTO, se quisiera mantener el historial de los departamentos en los que ha trabajado el empleado, esta se podría modelar como una relación N:N:



Cuando la relación N:N tiene atributos, como por ejemplo, en la relación EMPLEADO-trabaja-DEPARTAMENTO, esta relación se modela como dos relaciones de dependencia N:1.



Documentación de las relaciones

Una vez que se han determinado las entidades y relaciones entre ellas, es necesario asignar un nombre a las relaciones, para lo cual se pueden tomar en cuenta los siguientes puntos:

- Utilizar verbos en voz activa (trabaja, pertenece, renta, cursa, etc.).
- Eliminar homónimos.
- Utilizar sinónimos cuando sea necesario.
- Describir el significado de la relación entre las entidades participantes en un Diccionario, utilizando ejemplos y contra ejemplos.
- Indicar la cardinalidad máxima y la mínima.

Ejemplo:

EMPLEADO-trabaja-DEPARTAMENTO

Cardinalidad máxima: N:1
Cardinalidad mínima,
EMPLEADO a DEPARTAMENTO: obligatoria
DEPARTAMENTO a EMPLEADO: obligatoria

La relación trabaja entre EMPLEADO y DEPARTAMENTO representa el hecho de que un empleado este asignado a un departamento, de acuerdo a su contrato de trabajo. No representa el hecho de que un empleado pueda realizar alguna tarea en un departamento.

LABORATORIO 2

1. Revisar la descripción de requerimientos nuevamente. Hacer una lista de relaciones, y especificar qué relaciones están involucradas en cada relación.

2. Dibujar un Diagrama de Entidades y Relaciones.

3. Determinar la máxima y mínima cardinalidad de cada relación. Representar las respuestas en el diagrama.

ATRIBUTOS

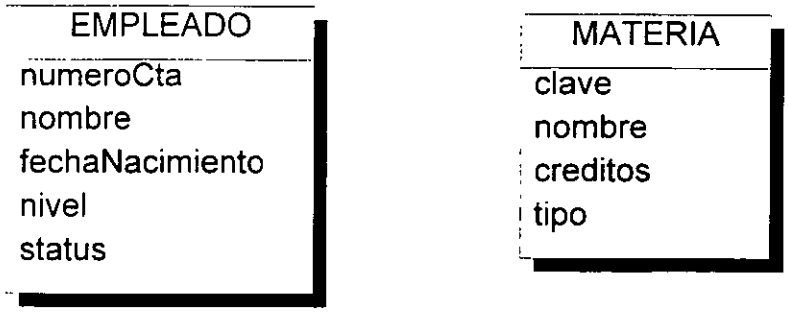
Es una propiedad de una Entidad. Los atributos describen, identifican, clasifican o determinan el estado de una entidad.

Por ejemplo, número, nombre y RFC pueden ser atributos de la entidad CLIENTE.

Los atributos deben ser asignados solamente a una entidad.

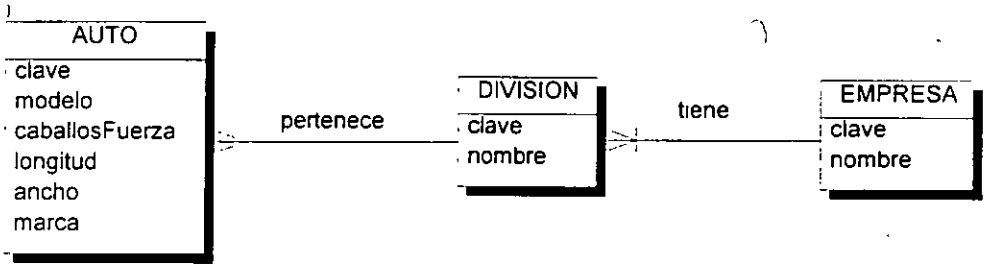
Las llaves foráneas (FK) no son atributos, estas se presentan solamente hasta la etapa de Diseño de la Base de Datos.

Ejemplo:

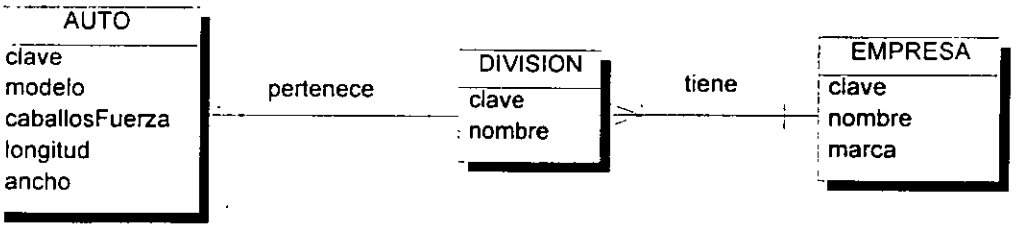


Hay que determinar cuales son atributos indirectos de la entidad, producto de la relación con otra entidad.

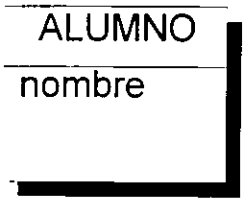
Ejemplo:

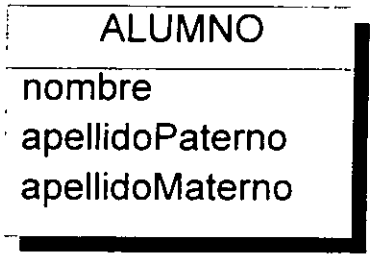


Un automóvil es fabricado en una división y su marca depende, no de la división sino de la empresa a la que pertenece la división: el mustang es fabricado por la división Mercury la cuál pertenece a la empresa Ford Motor Co.; por lo tanto, la marca es Ford:



Es necesario descomponer los atributos, cuando los requerimientos indiquen la necesidad de manejar los componentes de manera individual:





Se deberán descomponer los atributos codificados en componentes más simples y claros:

- Atributos que contienen fechas generalmente no se deberán descomponer en día, mes, año.

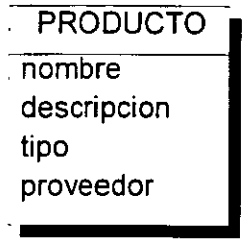
- Atributos que contengan información de códigos postales, números de cuenta, rfc, etc., generalmente no se deberán descomponer.

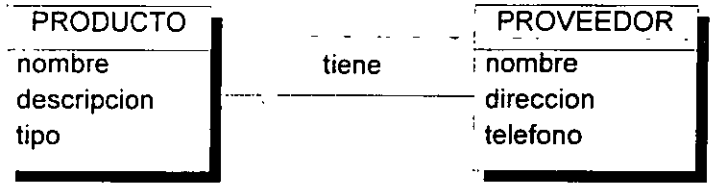
Los atributos derivables se deben eliminar en esta etapa. Por ejemplo el salario neto de los empleados obtenido de la suma del salario base más comisiones, más prestaciones, no se debe considerar como atributo directo.

Otros atributos que involucren sumas, conteo, promedios, etc. de otros atributos o de instancias, se deberán eliminar ya que el mantenerlos

provocará redundancia en el modelo.

Si un atributo tiene atributos que lo describen, que sean necesarios en el modelo de datos, entonces se trata de una entidad. La nueva entidad participa de una relación con la entidad de donde se originó el supuesto atributo.





Cardinalidad

La cardinalidad de un atributo indica el número de valores que existen para un atributo en una instancia de una entidad.

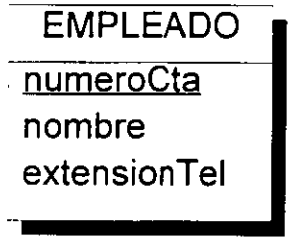
La cardinalidad mínima puede ser opcional, lo que indica que una instancia puede no tener valor para el atributo, u obligatoria. Los atributos que conforman la llave primaria tienen cardinalidad mínima obligatoria.

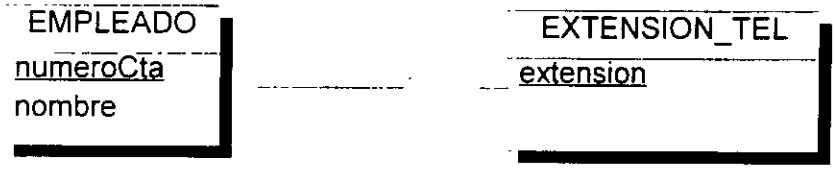
Generalmente la cardinalidad mínima no se representa en el DER, solamente se documenta en el Diccionario de Datos.

La cardinalidad máxima puede ser singular, lo que indica que una instancia debe de tener un solo valor para el atributo, o plural.

Los atributos plurales generan relaciones de dependencia, (ENTIDAD_DEPENDIENTE - ENTIDAD_INDEPENDIENTE) cuya cardinalidad máxima es N:1 y mínima 0:1.

La llave primaria de una Entidad Dependiente esta formada en parte por la llave primaria de la entidad de la que depende:





Documentación de los atributos

En el Diccionario hay que documentar los atributos de cada entidad, indicando:

- Nombre
- Tipo: tipo de dato asociado al atributo.
- Dominio: el conjunto de valores válidos para el atributo. El dominio esta determinado, en primera instancia por el tipo de dato, y en segunda por las reglas de empresa que apliquen al atributo.
- Cardinalidad
- Descripción

Ejemplo:

edad
Tipo: entero
Dominio: edad en el rango de 18 a 45
Cardinalidad mínima: requerida
Cardinalidad máxima: singular
Descripción: edad del empleado

LABORATORIO 3

1. En una tercera revisión de la descripción de requerimientos, determinar los atributos de las entidades y relaciones.

2. Especificar la máxima(múltiple) y mínima(opcional) cardinalidad para cada atributo de las entidades y relaciones. Escibir las respuesta en el DER.

3. En alguna de las entidaes existe un atributo que representa la Oficina DMV en donde se aplican los exámenes, ¿debe crearse adicionalmente una entidad para representar a dichas oficinas? Sustentar la respuesta.

4. ¿Existen Entidades Dependientes? Representarlas en el DER.

Generalización y especialización

La **generalización** es el proceso que permite reunir a un conjunto de entidades con atributos comúnes en una entidad de nivel superior llamada superentidad.

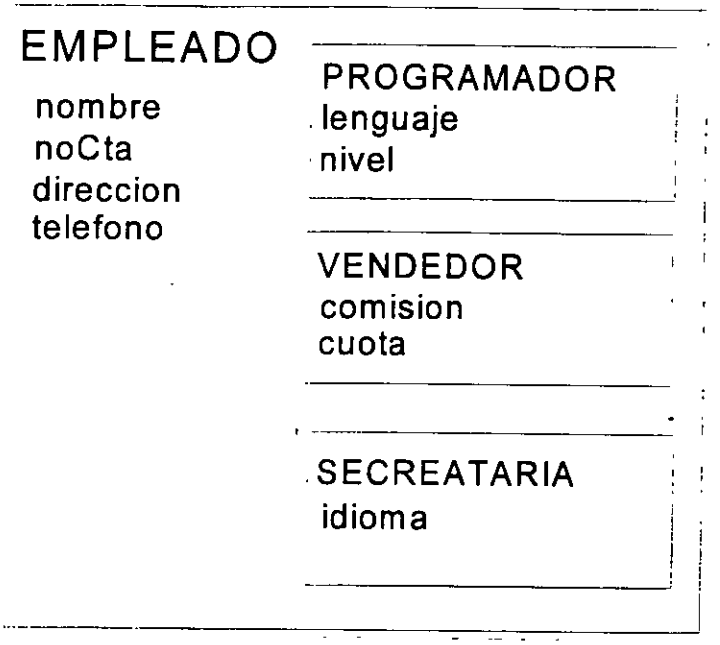
A cada una de las entidades agrupadas se les conoce como subentidades.

La **especialización** consiste en generar entidades con atributos particulares a partir de una entidad. En este caso la entidad de la cual parte la especialización también es llamada superentidad, y las entidades generadas subentidades.

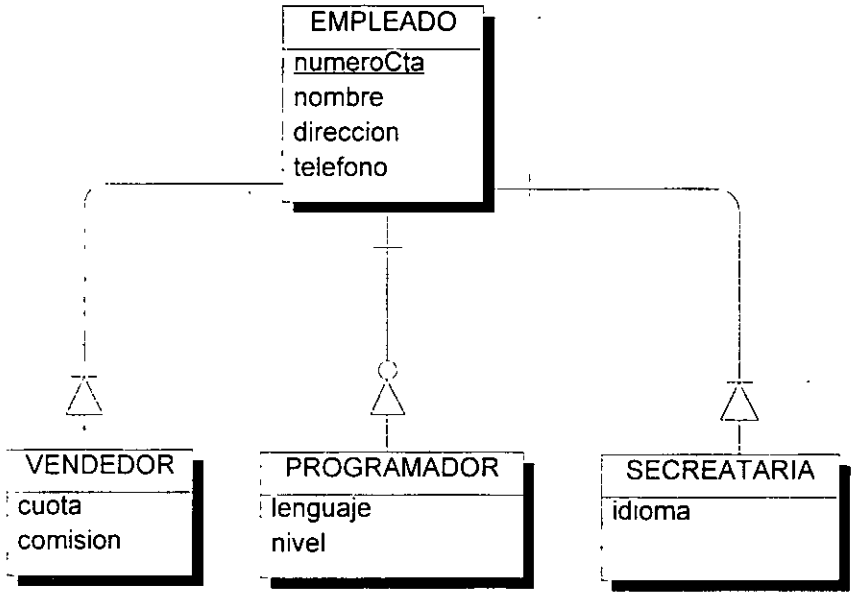
La llave primaria, atributos y relaciones de una superentidad también lo son de las subentidades. Lo contrario no aplica.

Las subentidades son mutuamente excluyentes.

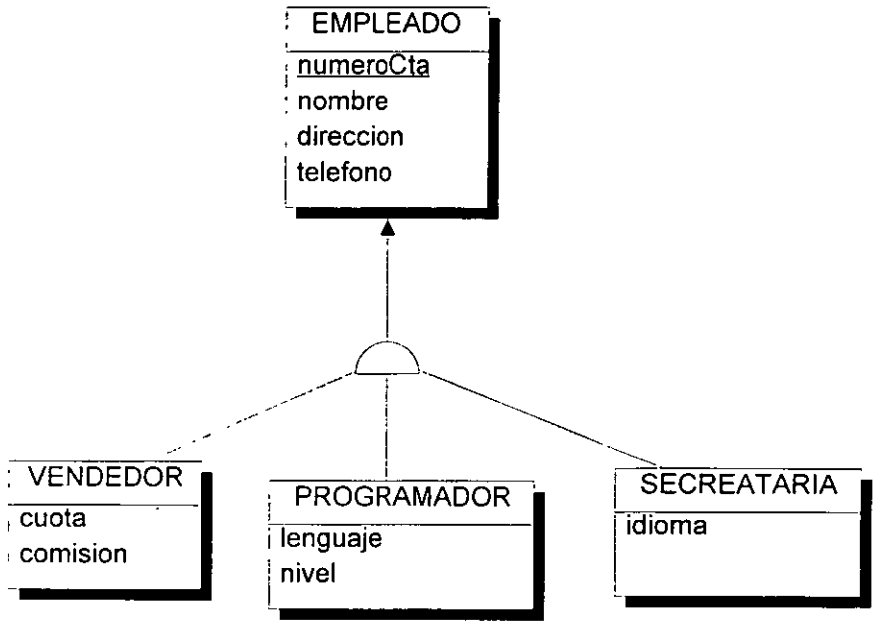
Ejemplo:


EMPLEADO
nombre
noCta
direccion
telefono
PROGRAMADOR
lenguaje
nivel
VENDEDOR
comision
cuota
SECREATARIA
idioma

Un esquema de subentidades se puede representar en el DER como relaciones de dependencia de las subentidades con la superentidad, conocidas como relaciones ISA (ES-UN):



Opcionalmente se puede utilizar la siguiente notación:

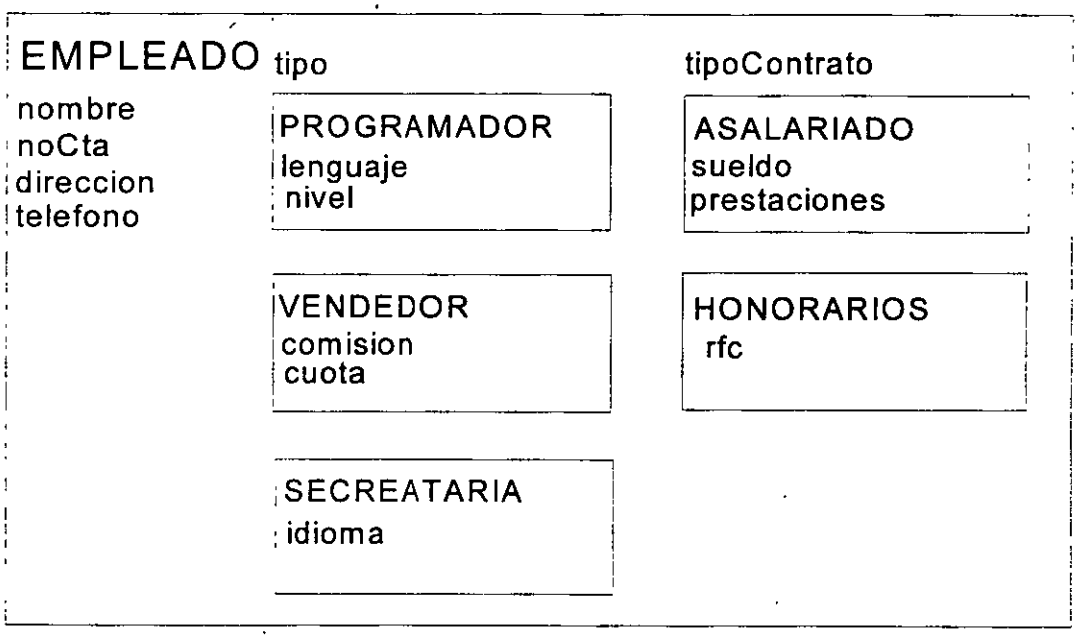


CLASES

Una clase es un conjunto de subentidades mutuamente excluyentes.

Las subentidades de una entidad se pueden agrupar en varias clases. Cada clase tiene asociado un atributo de clasificación en la superentidad.

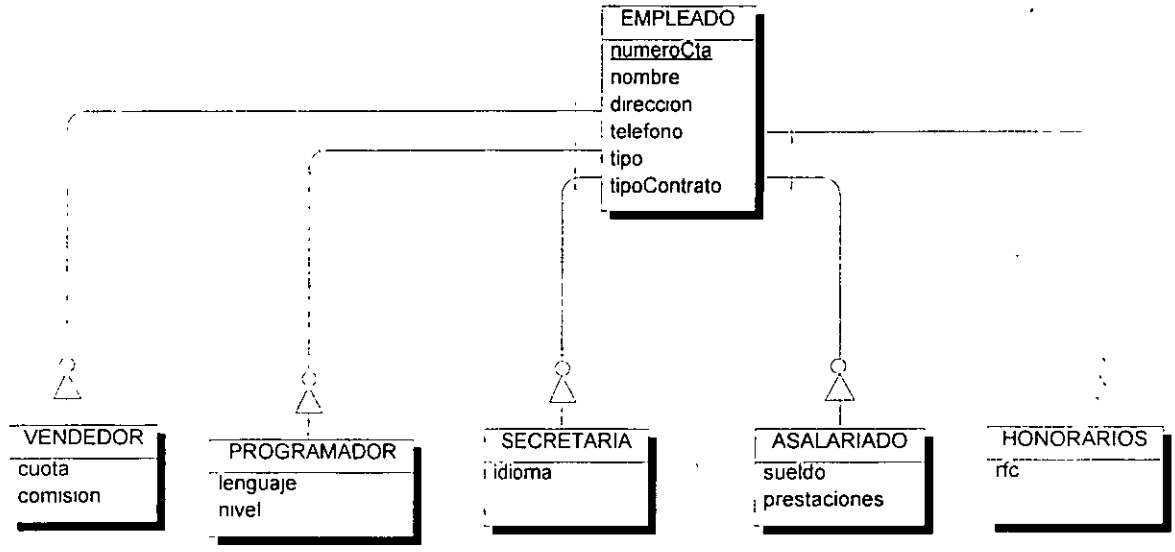
Ejemplo:

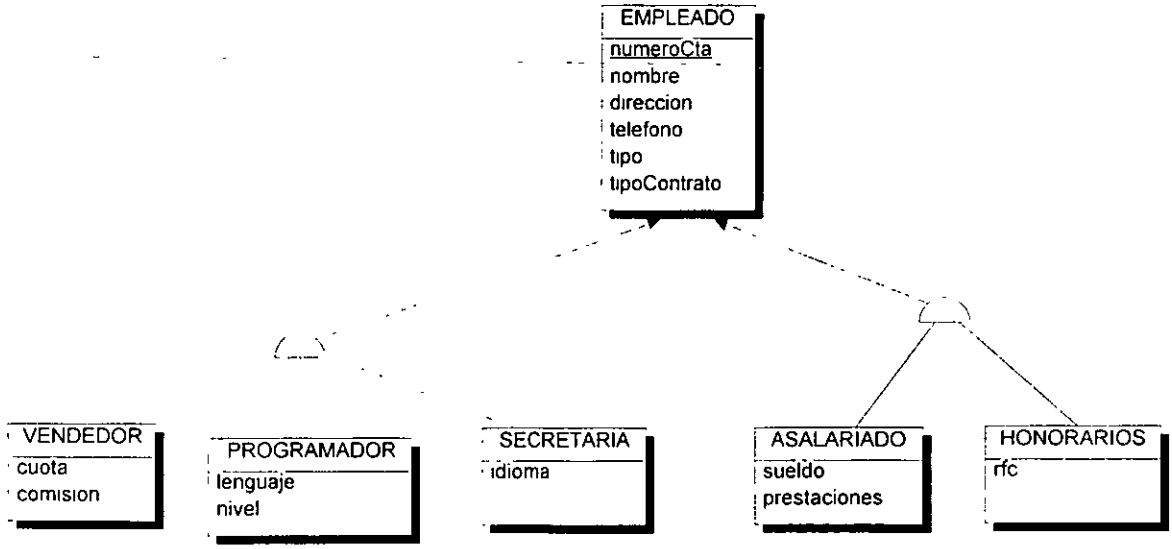


Puesto que en una clase, las subentidades son mutuamente excluyentes, no es necesario, que al menos existan dos subentidades en una clase. Si en una clase existe una subentidad solamente, se supone que implícitamente existe la que forman las instancias que no tienen los atributos de la subentidad.

En nuestro ejemplo, para la clase formada por ASALARIADO y HONORARIOS, si el RFC fuera un atributo con cardinalidad opcional uno, en EMPLEADO la subentidad HONORARIOS no tendría ningún atributo. En este caso se podría omitir la representación de HONORARIOS.

Representación de clases en el DER





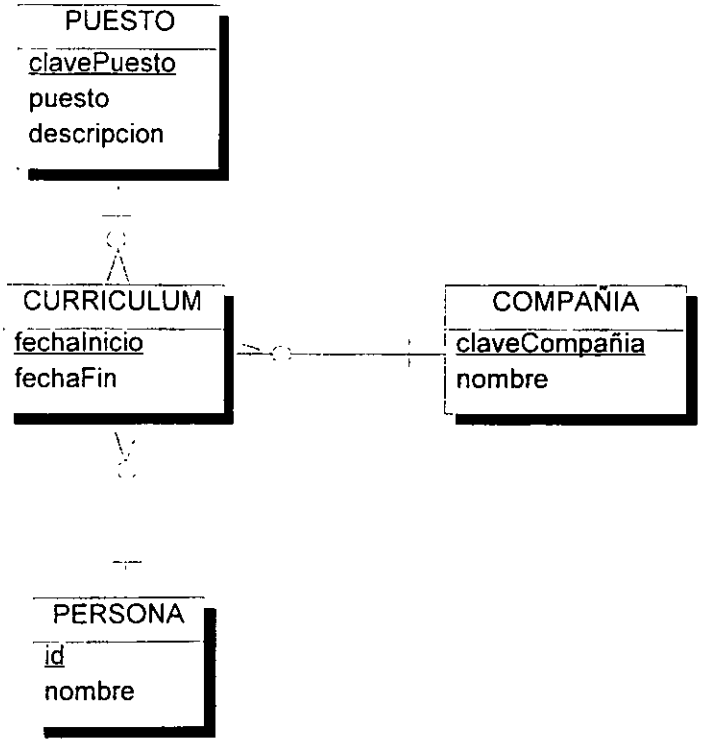
RELACIONES N-ARIAS

Las relaciones N-arias, son aquellas que existen entre más de dos Entidades.

Este tipo de relaciones se representa en el DER mediante una entidad de relación, la cuál mantiene relaciones de obligatoriedad N:1 hacia las entidades que participan de la relación N-aria.

La llave primaria de la entidad de relación esta formada por las llaves primarias de las entidades que relaciona, además de otros atributos en algunos casos.

Ejemplo:

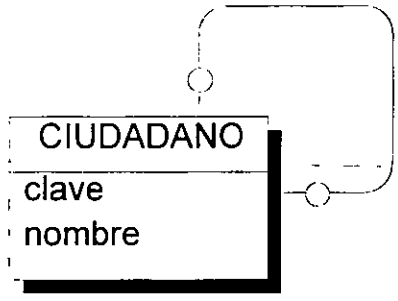


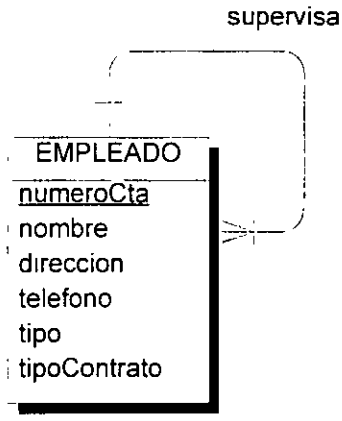
RELACIONES RECURSIVAS

Son las que se dan entre instancias del mismo conjunto de Entidades.

En estos casos el modelo se analiza, en cuanto a cardinalidad, exactamente igual a los casos no-recursivos; pero, se debe indicar el papel que juega cada Entidad en la relación.

Ejemplo: considere la Entidad CIUDADANO y la relación MATRIMONIO, que debe ser entre dos ciudadanos:





o la relación que representa el que un empleado supervise a varios empleados:

CONTENIDO DEL DICCIONARIO

Para cada entidad:

- Nombre
- Descripción
- Llave primaria
- En el caso de subentidades:

  Superentidad
  Clase
  Atributo de clasificación

- Número aproximado de instancias

Para cada relación:

- Nombre (entidad - relación - entidad)
- Descripción
- Cardinalidad máxima
- Cardinalidad mínima

- En el caso de relaciones recursivas, rol de cada entidad

Para cada atributo (entidad o relación)

- Nombre
- Descripción
- Tipo de dato
- Dominio
- Cardinalidad máxima
- Cardinalidad mínima
- Si es atributo derivable, indicar fórmula

LABORATORIO 4

1. Crear superentidades nuevas, alrededor de las entidades ya existentes del DER del laboratorio anterior, que se consideren similares y adecuadas para reunirse en una genérica. Revisar y modificar en caso necesario las relaciones y atributos de las superentidades creadas. Revisar y modificar en caso necesario las llaves primarias de la super y subentidades.

2. Leer la descripción adicional de requerimientos, que proporciona mayor detalle a la información de los vehículos. ¿Qué nuevas subentidades se pueden encontrar dentro de VEHICULO? Desarrollar un diagrama preliminar de VEHICULO que incluya sus subentidades.

3. Organizar las nuevas subentidades en clases (una clase será en función del tipo de vehículo y otra en función del uso del mismo). Documentar los atributos de clasificación.

4. En algunas ocasiones una infracción como conducir en estado de ebriedad puede causar un accidente, y en consecuencia causa un citatorio para la Corte. A pesar de que el DMV no tiene como función elaborar estos citatorios, desea relacionar infracciones y accidentes para facilitar su labor a la Corte. ¿Cómo puede hacerse?

5. Uno de los analistas de datos revisa el DER, y sugiere que la entidad ACCIDENTE debe tener un atributo NUMERO_PLACA (del oficial), de manera que se pueda determinar el oficial responsable, y otro atributo NUMERO_PLACA (del vehículo), para documentar a los vehículos involucrados. Argumentar si la sugerencia es adecuada o no.

6. Reasignar los atributos indirectos.

7. ¿Cuáles de las entidades y relaciones incluyen estados pasados o futuros? ¿Se deben hacer cambios en este sentido para que otras entidades y relaciones que actualmente sólo incluyen estados presentes, incluyan también pasados o futuros?

8. ¿Qué subentidades adicionales y clases podríamos anticipar? Deberíamos modelar OFICIAL como subentidad de PERSONA? Argumentar.

9. Se ha propuesto un nuevo requerimiento: El DMV debe ahora tener la capacidad de determinar al conductor de cada vehículo en un accidente. Recordar que un accidente puede involucrar a varios vehículos y conductores. ¿Cómo se modelaría este nuevo requerimiento?

10. Dibujar el DER final.

Página intencionalmente blanca.

El Modelo Relacional de Bases de Datos

OBJETIVO:

En este capítulo se estudiará la terminología asociada al Modelo Relacional de Bases de Datos.

En este capítulo el asistente:

- Definirá el concepto de Modelo Relacional de bases de datos.

- Conocerá el significado del valor nulo.

- Definirá los conceptos Llave Primaria y Llave Foránea.

- Definirá lo que es Integridad de la Base de Datos.

EL MODELO RELACIONAL

El modelo relacional fué originalmente propuesto por Codd en 1970 en un escrito titulado "A Relational Model of Data for Large Shared Data Banks".

El primer producto relacional que fué desarrollado en base a esta teoría se llamo System R, un DBMS desarrollado por IBM en los 70´s.

El modelo relacional esta definido en base a las estructuras de datos que permiten representarlo y a las operaciones que se pueden realizar sobre los datos.

ESTRUCTURA DEL MODELO RELACIONAL

En el modelo relacional, los datos y las relaciones entre los datos se representan por medio de **tablas**, a las cuales se les asigna un nombre único.

Cada una de las tablas esta compuesta por **columnas** con nombres únicos, las cuales tienen asociado un **tipo de dato**.

El contenido de una tabla es un conjunto de **renglones**, los cuales tienen un valor para cada una de las columnas de la tabla. Los valores permitidos para cada columna estan limitados al tipo de dato de la columna.

Ejemplos:

DEPARTAMENTOS

NO_DEPTO	NOMBRE
10	Sistemas
20	Finanzas
30	Contabilidad
40	Ingeniería

PROYECTOS

NO_PROY	DESCRIPCION
1	Diseño del Sistema de Inventarios
2	Sistema de Nomina

3	Sistema Integral de Servicios
4	Evaluación de Equipo de Cómputo

EMPLEADOS

NO_CTA	NOMBRE	DIR	TEL	NO_DEPTO
18456	Antonio	Revolucion 123	733-22-89	10
18272	Jéssica	Norte 86B 98	657-28-92	40
72638	Gerardo	Rio chico 22	512-38-39	10
28289	José	Palmas 926	833-32-21	30
29829	Raúl	Rosas 83-5	937-00-52	30
87719	Arturo	Churubusco 45	723-45-11	30
32983	Aarón	Taxqueña 372	632-73-21	20
32732	Sandra	Av. Central 13	743-03-01	40
32903	Ramón	Marina Nal. 86	732-37-77	20
95672	David	Almaraz 2-1	664-83-00	10
48568	Edwin :	Cozumel 11-3	731-82-83	40
84324	Jorge	Fco. Sosa 266	527-73-12	20

ASIGNADO

NO_CTA	NO_PROY
28289	2
95672	3
48568	4
84324	3
84324	4
18456	1
29829	2
32983	1
32732	4
18272	1
29829	1
72638	2
18272	3

87719	3
32732	2
32903	1

El modelo relacional tiene un fuerte apoyo matemático que permite fundamentar su estructura y las operaciones que se pueden realizar con los datos.

Un **dominio** es un conjunto de valores. Un atributo tiene asociado un conjunto de valores permitidos, es decir, un dominio.

Los matemáticos definen una **relación** como un subconjunto de un producto cartesiano de una lista de dominios, por lo tanto, las tablas son esencialmente relaciones.

Una relación es un conjunto de **tuplas** con valores para cada uno de los atributos.

Características de una relación

Una relación en el modelo relacional tiene las siguientes características:

- Los dominios de los atributos deben ser atómicos, es decir los elementos del dominio son unidades indivisibles.
- Cada atributo tiene un nombre único en la relación.
- Los valores de una tupla corresponden a los dominios de los atributos.
- El orden de los atributos no tiene importancia.
- Cada tupla es única, no existen tuplas duplicadas.
- El orden de las tuplas no es importante.

OPERACIONES

Las operaciones que se pueden llevar a cabo en el modelo relacional tienen un fundamento matemático conocido como **Algebra Relacional**.

El Algebra Relacional es un lenguaje de consulta abstracto procedural. Consta de un conjunto de operaciones que toman como entrada una o dos relaciones.

Las operaciones definidas en el Algebra Relacional tienen la propiedad de cerradura, ya que el resultado de la operación es una relación.

Existen cinco operaciones fundamentales:

- Proyección π
- Elección σ
- Producto Cartesiano X
- Unión **U**
- Resta de conjuntos -
- Intersección de conjuntos $\cap$

Para ejemplificar las operaciones, se utilizaran las siguientes relaciones:

ALUMNO

no_cta	nom_alum	cve_carr
105	Sandra	027
107	José	032
101	Luis	032
103	Lourdes	027
106	Juan	029
102	Santiago	029
108	Verónica	027

PROFESOR

cve_prof	nom_prof
acf000	Antonio
rrh000	Ramón
enp000	Edwin
jbc000	Jéssica

CARRERA

cve_carr	carrera
027	Industrial
032	Computación
039	Electrónica

Proyección

Es una operación unitaria, ya que actúa sobre una sola relación. Se representa con la letra griega π. Se utiliza para proyectar atributos, de una relación, su sintaxis es la siguiente:

$$\pi_{\text{atributo 1, atributo 2, ..., atributo n}} (\text{Relación})$$

Ejemplo: Listar a los alumnos y su número de cuenta

$$\pi_{\text{nom\_alum, no\_cta}} (\text{ALUMNO})$$

nom_alum	no_cta
Sandra	105
José	107
Luis	101
Lourdes	103
Juan	106
Santiago	102
Verónica	108

Ejemplo: obtener el nombre de los profesores:

$$\pi_{nom\_prof}(PROFESOR)$$

nom_prof
Antonio
Ramón
Edwin
Jéssica

Elección

Al igual que la proyección la elección es una operación unitaria. Se representa con la letra griega σ, y su función es elegir aquellas tuplas que cumplan con la condición especificada. Su sintaxis es la siguiente:

$$\sigma_{condición}(Relación)$$

donde:

condición: es una expresión la cual puede tener operadores de relación (>, <, >=, <=, =, !=) y operadores lógicos (AND, OR, NOT).

Ejemplo: ¿Qué alumnos cursan la carrera cuya clave de carrera es 027?

$$\sigma_{cve\_carr=027}(ALUMNOS)$$

nom_alum	cve_carr
Sandra	027
Lourdes	027
Véronica	027

Las operaciones del Algebra Relacional tienen la propiedad de cerradura; por lo que el resultado de una operación se puede utilizar como argumento para otra.

Ejemplo: ¿Cual es la clave de la carrera de Computación?

$$\pi_{\text{cve\_carr}} (\sigma_{\text{carrera= "Computación"}} (\text{CARRERA}))$$

cve_carr
027

Producto cruz

Es una operación binaria ya que actúa sobres dos relaciones, se representa por una cruz X. Esta operación permite combinar ó relacionar informacion de dos o mas relaciones. Su sintaxis es la siguiente:

(Relación1) X (Relación2)

Esta operación da como resultado una relación que tiene **nXm** tuplas y todos los atributos de cada relación.

donde:

n= número de tuplas de Relación1
m= número de tuplas de Relación2

Los atributos de la relación resultado son los atributos de las relaciones sobre las que se realiza la operación, el nombre de estos esta precedido por el nombre de la relación de la que provienen, de esta forma se sigue cumpliendo el que los nombres de los atributos sean únicos.

Ejemplo:

(ALUMNO X CARRERA)

ALUMNO. no_cta	ALUMNO. nom_alum	ALUMNO. cve_carr	CARRERA. cve_carr	CARRERA. carrera
105	Sandra	027	027	Industrial
105	Sandra	027	032	Computación
105	Sandra	027	029	Electrónica
107	José	032	027	Industrial
107	José	032	032	Computación
107	José	032	029	Electrónica
101	Luis	032	027	Industrial
101	Luis	032	032	Computación
101	Luis	032	029	Electrónica
103	Lourdes	027	027	Industrial
103	Lourdes	027	032	Computación
103	Lourdes	027	029	Electrónica
106	Juan	029	027	Industrial
106	Juan	029	032	Computación
106	Juan	029	029	Electrónica
102	Santiago	029	027	Industrial
102	Santiago	029	032	Computación
102	Santiago	029	029	Electrónica
108	Verónica	027	027	Industrial
108	Verónica	027	032	Computación

108	Verónica	027	029	Electrónica

Ejemplo: ¿Qué alumnos cursan la carrera de Computación?

$$\pi_{\text{nom\_alum}}(\sigma_{\text{carrera = "Computacion"}}(\sigma_{\text{ALUMNO cve\_carr= CARRERA.cve\_carr}}(\text{ALUMNO X CARRERA})))$$

nom_alum
José
Luis

Unión

Es una operación binaria, con la cual podemos obtener todos las tuplas de las relaciones involucradas. Su sintaxis es la siguiente:

$$\text{Relación1} \cup \text{Relación2}$$

Resta

Es una operación binaria, con la cual obtenemos todos las tuplas que se encuentran en la primera relación, pero que no se encuentran en la segunda. Su sintaxis es la siguiente:

$$\text{Relación1} - \text{Relación2}$$

Intersección

Esta operación es adicional ya que se puede obtener de las operaciones anteriores. El resultado son aquellas tuplas que se encuentran en ambas relaciones.

$$\text{Relación1} \cap \text{Relación2}$$
$$=$$
$$\text{Relación1} - (\text{Relación1} - \text{Relación2})$$

Esta operación no es conmutativa: $A - B \neq B - A$

Para realizar las operaciones Union, Resta e Intersección se debe cumplir con dos condiciones:

- Las relaciones involucradas deben tener el mismo múmero de atributos.
- Los dominios de los atributos de las relaciones involucradas deben ser los mismos.

VISTAS

No es conveniente que todos los usuarios vean el modelo conceptual completo. Las condiciones de seguridad pueden requerir que se "escondan" ciertos datos a algunos usuarios.

Aparte de las cuestiones de seguridad, se puede querer crear un conjunto de relaciones que correspondan mejor con una cierta imágen que el usuario tiene de la base de datos, que el modelo conceptual.

Cualquier relación que no es parte del modelo conceptual, pero es visible al usuario como una "relación virtual", se llama **vista**. Es posible tener un gran número de vistas sobre cualquier conjunto de relaciones reales dado.

Las relaciones del modelo conceptual pueden modificarse, es por ello, que no es posible almacenar una relación correspondiente a una vista. En cambio, una vista debe volverse a calcular para cada consulta que se refiera a ella.

Aunque las vistas son una herramienta útil para consultas, presentan problemas importantes si las actualizaciones se expresan usando vistas.

VALORES NULOS

El símbolo **NULL** representa un valor no conocido en la base de datos, un valor que no existe.

Un valor nulo no lo representa cero o la cadena "", ya que estos, son valores conocidos.

Todas las comparaciones que impliquen valores nulos en las operaciones relacionales son falsas por definición.

Las operaciones aritméticas que impliquen valores nulos, dan como resultado un valor desconocido, es decir NULL.

Llave primaria(PK)

Una **llave primaria** es un atributo o conjunto de atributos que identifican a las tuplas de una relación. Cada relación debe contar con una llave primaria.

Las características necesarias para una llave primaria son las siguientes:

- Unica
- Conocible en cualquier tiempo

Las características deseables de una llave primaria son las siguientes:

- Estable
- No descriptiva
- Pequeña y simple

Cuando la llave primaria esta formada por más de un atributo, se denomina **llave primaria compuesta**.

Una relación puede tener más de un atributo o combinación de atributos que pueden actuar como llave primaria. A cada una de estas combinaciones o atributos se les llama **llave candidato**.

Solamente puede existir una llave primaria por relación, las llaves candidatos

restantes se denominan **llaves alternas**.

Ejemplos:

EMPLEADO

numeroCta	nombre	rfc	direcció n	salario	noDepto

La llave primaria es numeroCta
rfc es una llave candidato, por lo tanto también es una llave alterna

CUENTA

banco	noCta	saldo	fechaApertura

La llave primaria es banco, noCta, la cuál es una llave primaria compuesta

Llave foránea(FK)

Una llave foránea en una relación es un atributo o conjunto de atributos que forman la llave primaria de otra o la misma relación.

Las llaves foráneas permiten llevar a cabo operaciones binarias para obtener información de un conjunto de relaciones.

Los valores de los atributos que conforman una llave foránea deben corresponder a valores existentes en la llave primaria o, tener valor nulo.

Es conveniente que la llave foránea tenga el mismo nombre y tipo de la llave primaria de la que proviene.

La llave primaria de una relación puede estar formada en parte por una llave foránea, en este caso, la llave foránea no puede tener valores nulos.

Ejemplos:

DEPARTAMENTO

noDepto	nombreDepto	**numeroCta**

EMPLEADO

numeroCta	nombre	rfc	dirección	salario	**noDepto**

numeroCta es llave foránea en DEPARTAMENTO
noDepto es llave foránea en EMPLEADO

BANCO

banco	nombre	dirección

CUENTA

banco	noCta	saldo	fechaApertura

La llave primaria de CUENTA esta formada en parte por una llave foránea

INTEGRIDAD

El concepto de Integridad es sencillo: los datos deben ser válidos.

Ejemplos:

- No pueden existir renglones repetidos en una tabla.

- El número de departamento de un empleado debe de pertenecer a un departamento válido.

- En una Base de Datos de Reservación de líneas aéreas, el número de reservaciones no debe ser mayor que el cupo en un vuelo.

- El salario de un empleado debe estar entre N$ 1,000 y N$ 8,000.

Se definen los siguientes tipos de Integridad:

Integridad de Entidades: cada tupla en una relación debe ser única.

Integridad de Dominio: cada valor de un atributo pertenece a un dominio previamente definido. Si la llave primaria es única, se garantiza que una tupla es única en una relación.

Integridad Referencial: cada llave foránea debe estar asociada a una llave primaria válida, o debe tener un valor nulo.

Integridad del Negocio: Los datos de la Base de Datos deben cumplir con las reglas del negocio.

Integridad Referencial

Cada llave foránea debe estar asociada a una llave primaria válida, o debe tener un valor nulo.

Para poder mantener esta relación se pueden considerar las siguientes reglas:

Para la llave foránea:

- Insert/Update

  - No permitir el insert/update si no existe una llave primaria asociada (RESTRICT).

  - Asignar valor nulo a la llave foránea sino existe la correspondiente llave primaria(NULLIFY).

  - Asignar valor default a la llave foránea sino existe la correspondiente llave primaria(DEFAULT).

  - Si la llave primaria asociada a la llave foránea no existe, dar de alta la tupla asociada a la llave primaria.

- Para Delete no se debe considerar acción alguna.

Para la llave primaria:

- Para Update:

  - Modificar todas las llaves foráneas asociadas (UPDATE CASCADE).

  - Evitar la modificación si existen llaves foráneas asociadas (RESTRICT).

- Para Delete:

  - Borrar todas las llaves foráneas asociadas (DELETE CASCADE).

  - Evitar el borrado si existen llaves foráneas asociadas (RESTRICT).

  - Asignar NULL a las llaves foráneas asociadas (NULLIFY).

  - Asignar un valor de default a las llaves foráneas asociadas (DEFAULT).

- Para Insert no se debe considerar acción alguna.

Ejemplo:

Suponga las relaciones ALUMNO y CARRERA:

ALUMNO

no_cta	nom_alum	cve_carr
105	Sandra	027
107	José	032
101	Luis	032
103	Lourdes	027
106	Juan	029
102	Santiago	029
108	Verónica	027

CARRERA

cve_carr	carrera
027	Industrial
032	Computación
039	Electrónica

¿ Qué acciones se podrían llevar a cabo para mantener la integridad de los datos, si para el alumno Luis se quiere modificar su clave de carrera a 25?

¿ Que acciones se podrían llevar a cabo si se desea cambiar la clave de la

carrera de Computación?

REGLAS DE CODD

En 1985 Codd publicó una serie de reglas o principios que un DBMS debe cumplir para considerarse "completamente relacional":

Regla Cero:

Un DBMS debe manejar los datos almacenados utilizando únicamente sus características relacionales.

Regla 1, Representación de la información:

Toda la información, en el nivel lógico, debe ser representada como valores en tablas.

Regla 2, Garantía de acceso:

Debe ser posible accesar cualquier dato en la base de datos si se hace referencia al nombre de la base de datos, la columna y el valor de la llave primaria.

Regla 3, Representación de valores nulos:

Se deben poder representar valores nulos, sin importar el tipo de dato. El valor nulo debe ser distinto de cero o cualquier valor numérico, así como de la cadena vacía.

Regla 4, Catálogo Relacional:

El catálogo de la base de datos, que contiene la descripción lógica de la misma, debe ser representado como datos ordinarios.

Regla 5, Lenguaje comprensible:

Sin importar el número de lenguajes soportados, se debe proporcionar un lenguaje que, por medio de sentencias expresadas como cadenas de caracteres, permita la definición de datos y vistas, la manipulación de datos, la definición de las reglas de integridad y los esquemas de autorización.

Regla 6, Modificación de vistas:

Una vista que sea teoricamente modificable, puede ser modificada.

Regla 7, Operaciones Insert, Delete y Update:

Una relación que pueda ser utilizada para consulta, también lo puede ser para las operaciones insert, update y delete.

Regla 8, Independencia física:

Las aplicaciones de la base de datos son inmunes a cambios relacionados con los métodos de acceso a los datos o con los esquemas de almacenamiento de la base de datos.

Regla 9, Independencia lógica:

Cambios generados a nivel lógico que no afecten la información a nivel lógico, no requieren de modificaciones en las aplicaciones.

Regla 10, Reglas de integridad:

Las reglas de integridad de entidades e integridad referencial deben ser indicadas en el lenguaje que proporciona el DBMS y almacenadas como datos en el catálogo, no es necesario programarlas en las aplicaciones.

Regla 11, Independencia de la distribución de datos:

Los programas de aplicación y los comandos del usuario que utilicen el lenguaje de definición de datos no deben sufrir cambios por efectos de distribución de la base de datos.

Regla 12, No subversión:

El DBMS debe tener el control total de los datos, no es posible que por medio de lenguajes de bajo nivel se puedan violar las reglas de integridad impuestas en la base de datos.

LABORATORIO

Utilizando las tablas de la página 3 y 4 (DEPARTAMENTOS, PROYECTOS, EMPLEADOS y ASIGNADO), esprese las siguientes consultas en algebra relacional:

1. El nombre de los empleados del departamento 10.

2. El nombre de los empleados del departamento Sistemas.

3. Los empleados de Ingeniería que estan asignados al proyecto de Sistema de Nómina.

4. Los empleados asignados al proyecto de Sistema de Nómina que no estan asignados al proyecto de Diseño del Sistema de Inventarios.

5. ¿Qué departamentos participan en el proyecto de Sistema Integral de Servicios?

Diseño de Bases de Datos Relacionales

OBJETIVO:

En este capítulo se estudiarán las técnicas para la obtención de tablas a partir del modelado representado por medio de un Diagrama de Entidades y Relaciones.

En este capítulo el asistente:

- Aprenderá a generar un Modelo Relacional de Bases de Datos a partir del Modelo Coceptual.

- Aprenderá a diseñar las entidades y relaciones entre estas.

- Entenderá el concepto de redundancia.

- Comprenderá el proceso de normalización de un diseño de Bases de Datos.

DISEÑO RELACIONAL DE BASES DE DATOS

El diseño relacional consiste básicamente en la generación del Modelo Relacional de la base de datos a partir del Modelo Concepual.

El Modelo Conceptual de la base de datos es independiente del DBMS a ser utilizado y también del Modelo de Base de Datos utilizado por este.

El Diseño relacional se lleva a cabo en dos etapas:

- Mapeo del DER a tablas
- Redefinición del diseño tomando en cuenta consideraciones de rendimiento

El objetivo de la primera etapa es generar un esquema de tablas con las siguientes características:

- Esquema normalizado (redundancia minimizada)
- Sin pérdida de información, es decir que toda la información del Modelo Conceptual se pueda obtener relacionalmente

Esta primera etapa se puede llevar a cabo independientemente del DBMS a ser utilizado.

El objetivo de la segunda etapa es generar un esquema de tablas que proporcione las condiciones para que las aplicaciones de la base de datos tengan un alto rendimiento.

En esta segunda etapa se deben considerar las aplicaciones críticas de la base de datos y algunas características del DBMS a ser utilizado.

REPRESENTACION DE ENTIDADES

Cada entidad independiente representa una tabla.

La llave primaria de tablas independientes no contiene llaves foráneas.

Cada atributo de una entidad representa una columna de la tabla.

Los atributos con cardinalidad opcional, se mapean a columnas que aceptan valores nulos.

ASISTENTE		
NUMEROCTA	<pk>	not null
NOMBRE		not null
DIRECCION		not null
TELEFONO		null
EMPRESA		not null
EDAD		not null

Representación de Relaciones

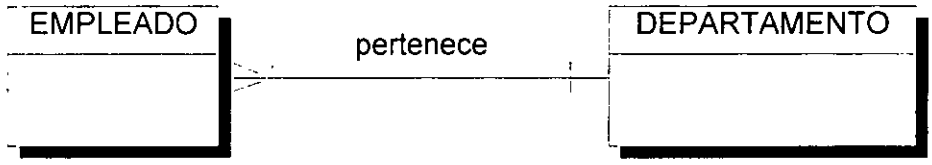
Las relaciones representadas en el Modelo Conceptual, se representan con llaves foráneas en las tablas relacionadas, o se generan tablas que mantienen la información de la relación.

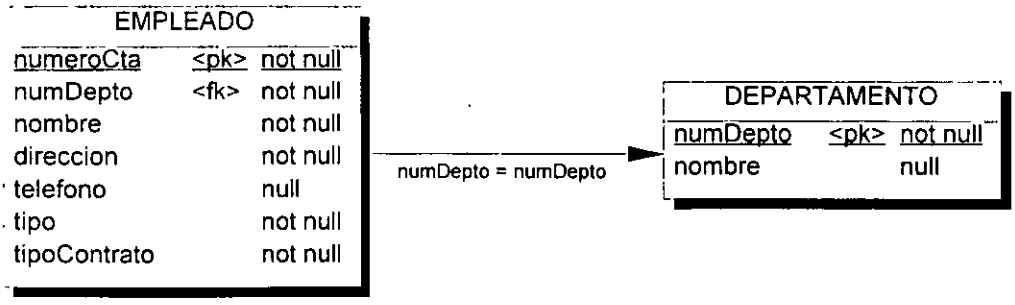
Relaciones N:1 (muchos a uno)

En relaciones N:1, no importando la cardinalidad de la misma, la llave foránea se coloca en la entidad con el grado N.

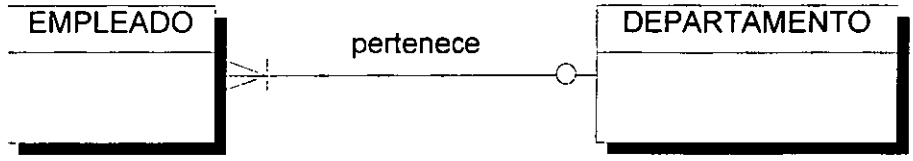
La cardinalidad mínima en la entidad de grado N, determina si la llave foránea acepta o no valores nulos.

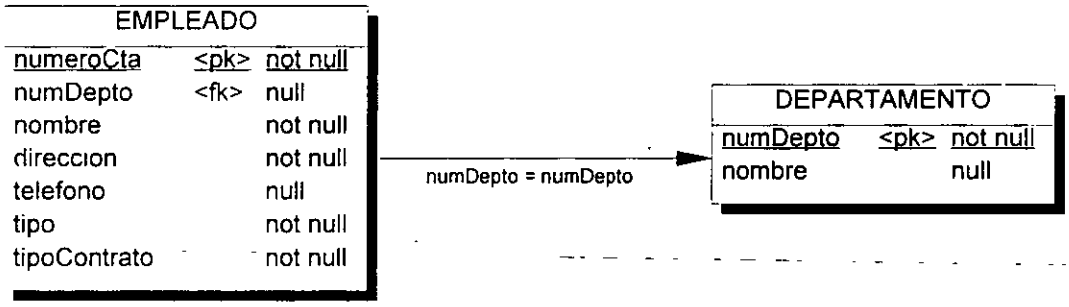
- Para una relación 1:N con cardinalidad mínima 1:1





- Relación 1:N con cardinalidad mínima 1:0.



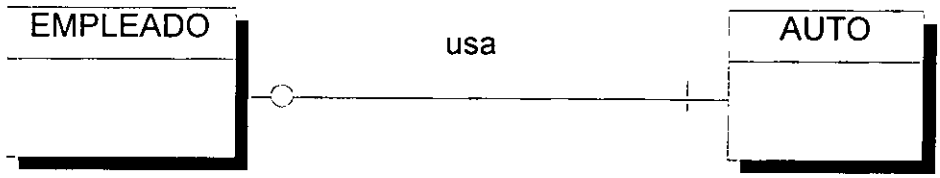


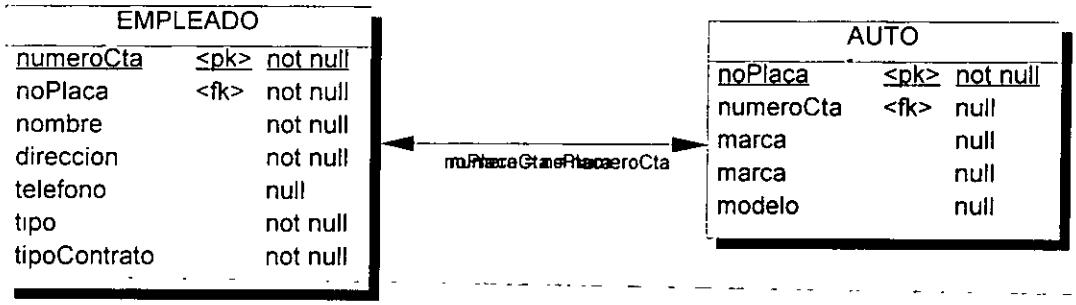
Relaciones 1:1 (uno a uno)

La llave foránea se coloca en la entidad con cardinalidad mínima cero.

- Cardinalidad mínima 1:0

- Cardinlaidad mínima 0:1

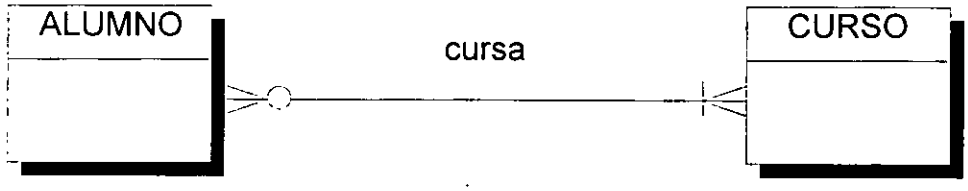


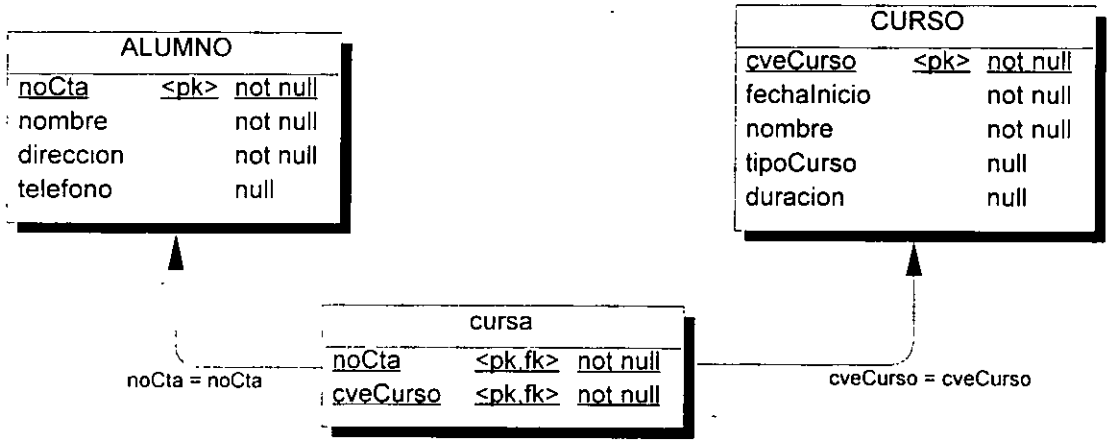


Cuando la cardinalidad mínima es 0:0 se puede colocar la llave foránea en cualquiera de las tablas relacionadas.

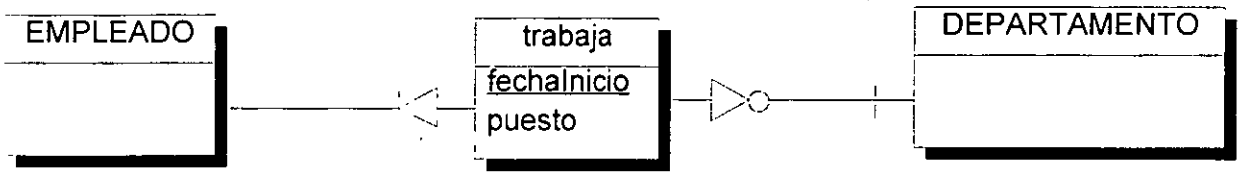
Relaciones N:N (muchos a muchos)

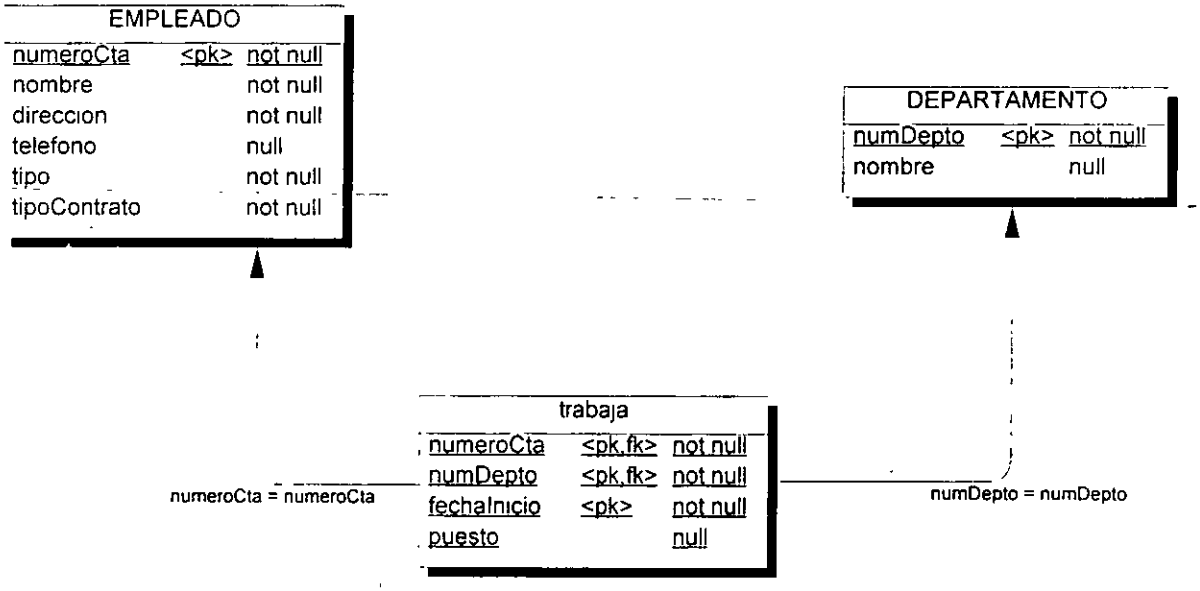
En relaciones N:N, no importando la cardinalidad mínima, se genera una tabla de relación, cuya llave primaria esta compuesta de las llaves primarias de las tablas que forman parte de la relación.





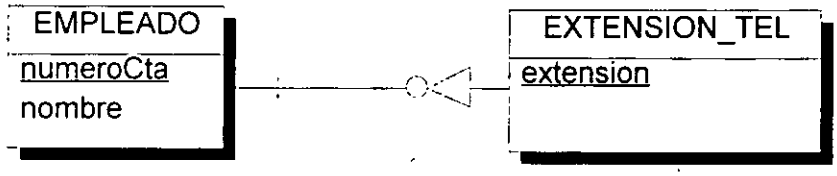
Para relaciones N:N que se modelaron como dos relaciones de dependencia, la entidad generada será una tabla, cuya llave primaria contiene las llaves primarias de las entidades que se realacionan:

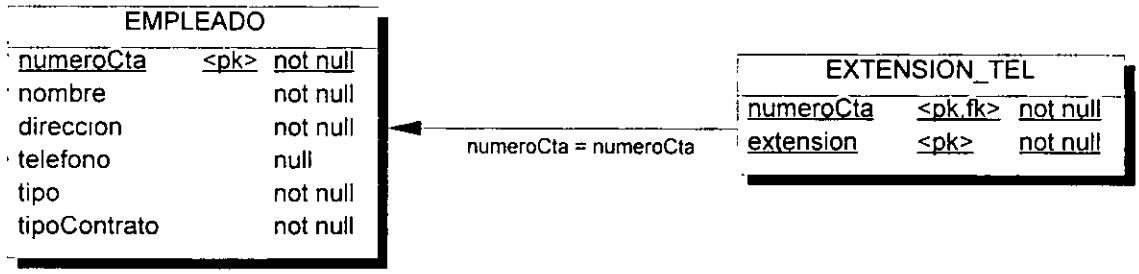




Relaciones de dependencia

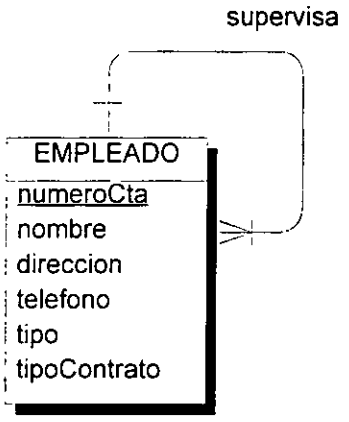
Las entidades dependientes, las cuales participan en una relación de dependencia, tienen una llave primaria que contiene la llave primaria de la entidad de la que depende.

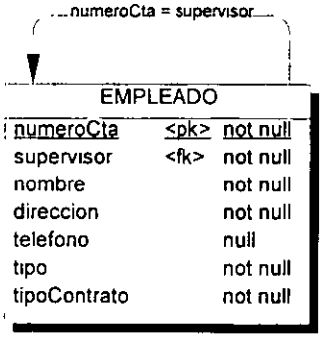




Relaciones recursivas

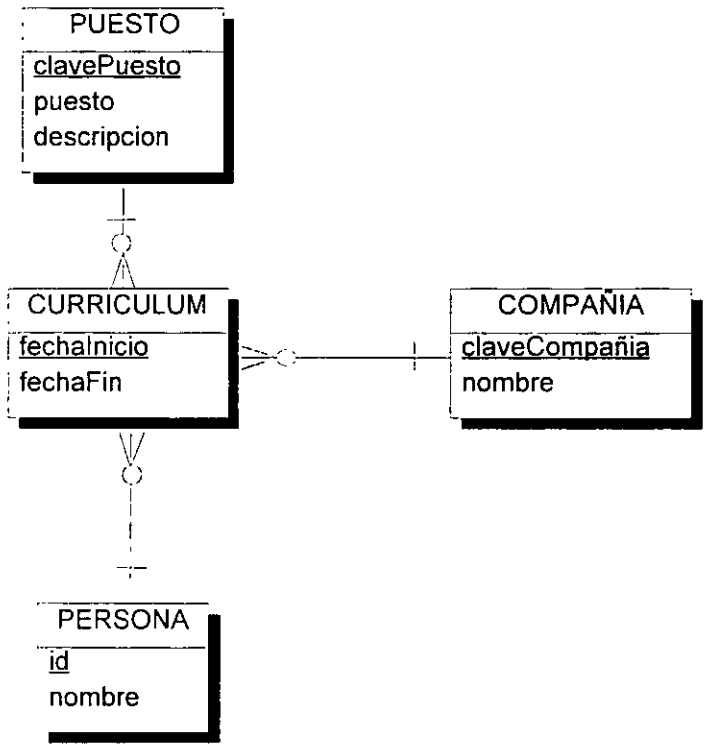
En relaciones recursivas es importante considerar la cardinalidad máxima de la relación, dependiendo de ella se determina si es necesaria una llave foránea en la tabla o es necesario crear una tabla de relación, de acuerdo a las reglas indicadas para las relaciones binarias.

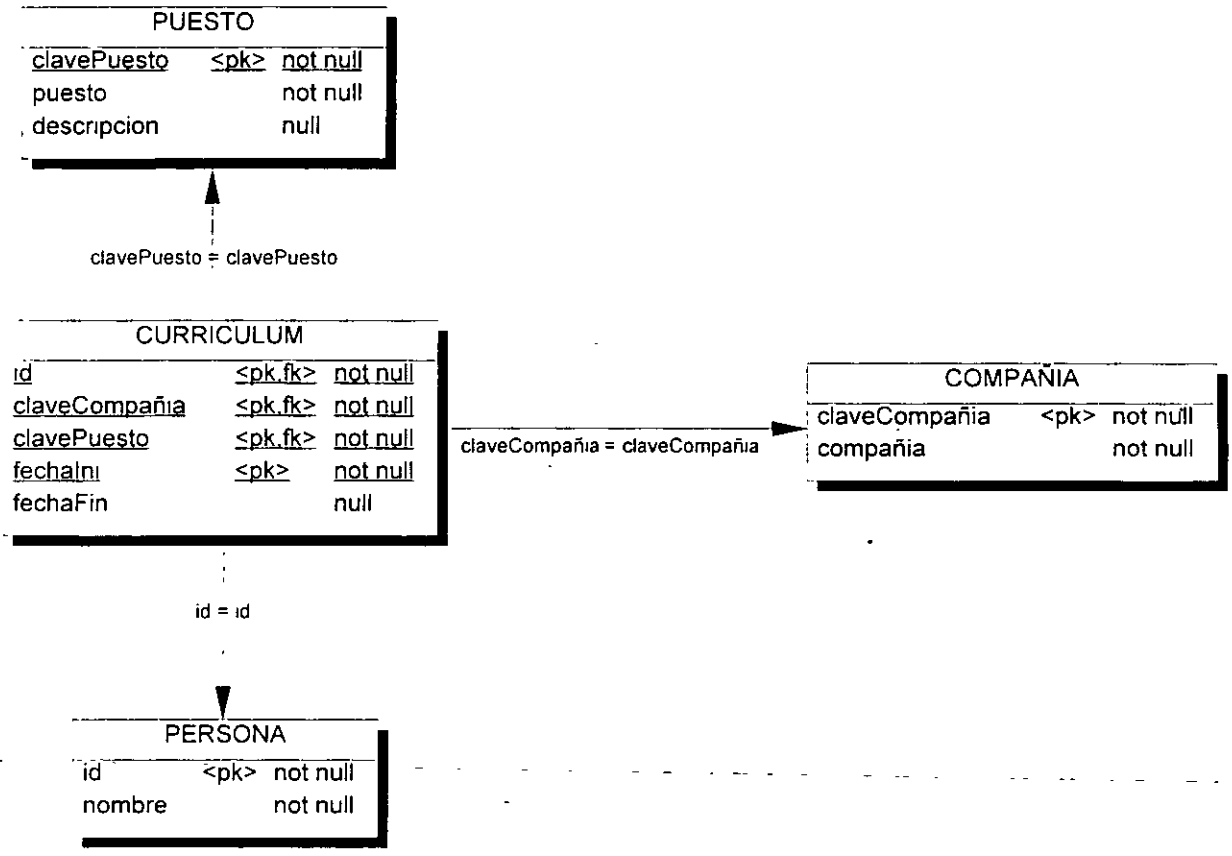




Relaciones N-arias

No importan mucho los grados de asociación o de pertenencia, de cualquier forma se debe representar la Asociación y formar una llave primaria con la concatenación de los identificadores de las Entidades que participan.



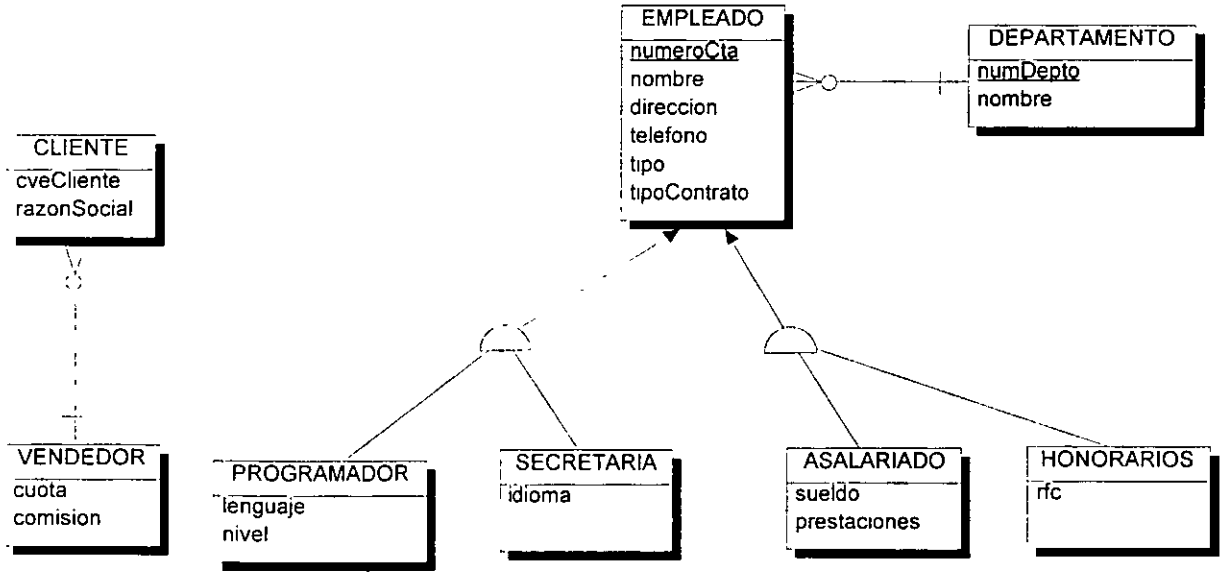

PUESTO
clavePuesto <pk> not null
puesto not null
descripcion null
clavePuesto = clavePuesto
CURRICULUM
id <pk,fk> not null
claveCompañia <pk,fk> not null
clavePuesto <pk,fk> not null
fechaIni <pk> not null
fechaFin null
claveCompañia = claveCompañia
COMPAÑIA
claveCompañia <pk> not null
compañia not null
id = id
PERSONA
id <pk> not null
nombre not null

Superentidades y subentidades

La llave primaria, atributos y relaciones de una superentidad también lo son de las subentidades, pero en forma indirecta. Lo contrario no aplica.

La superentidad es una tabla que contiene la llave primaria y atributos comunes, además de las relaciones comunes a las subentidades.

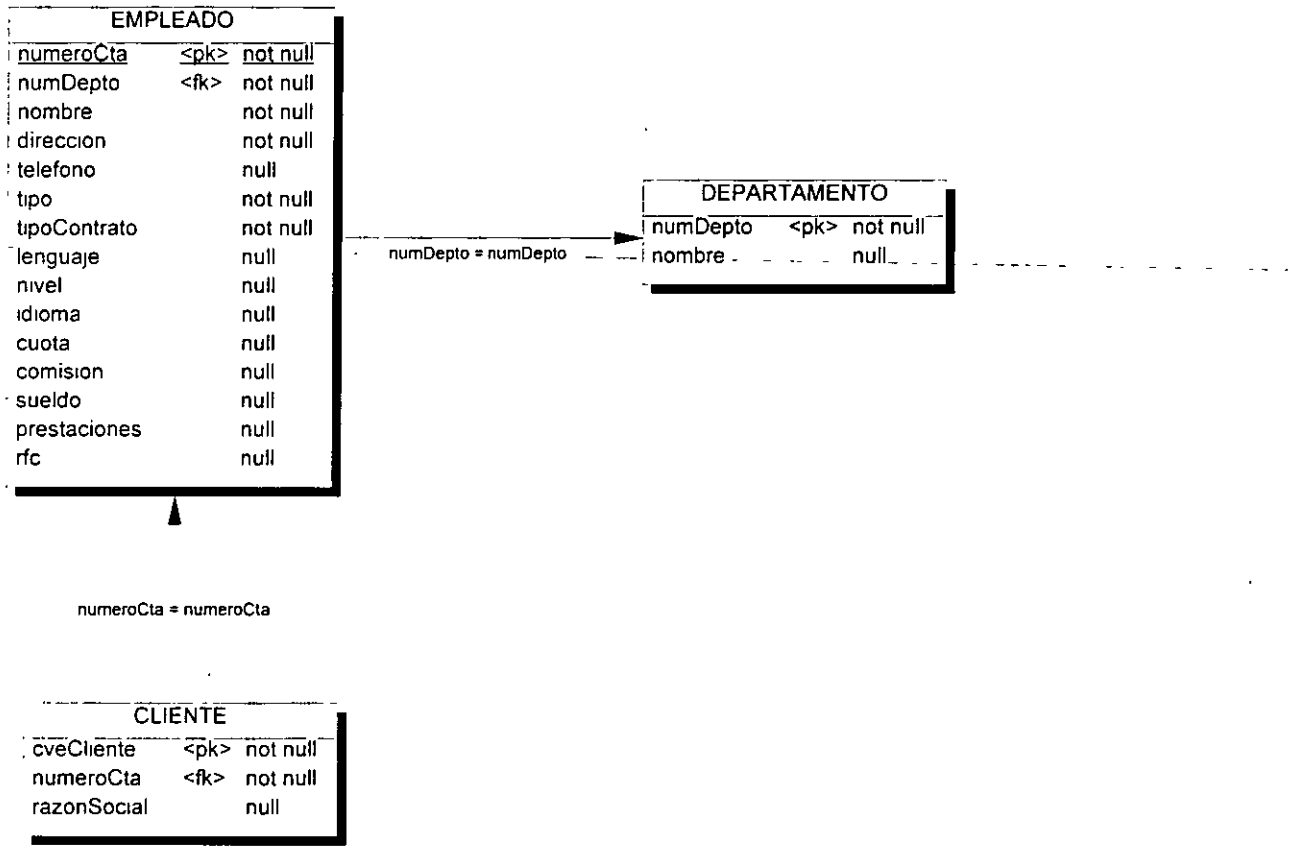
Cada una de las subentidades son entidades dependientes que se representan mediante una tabla que contiene la llave primaria de la superentidad y los atributos propios de la subentidad. Cada subentidad mantiene las relaciones propias de la misma.



El diseño de subentidades se puede llevar a cabo de tres formas:

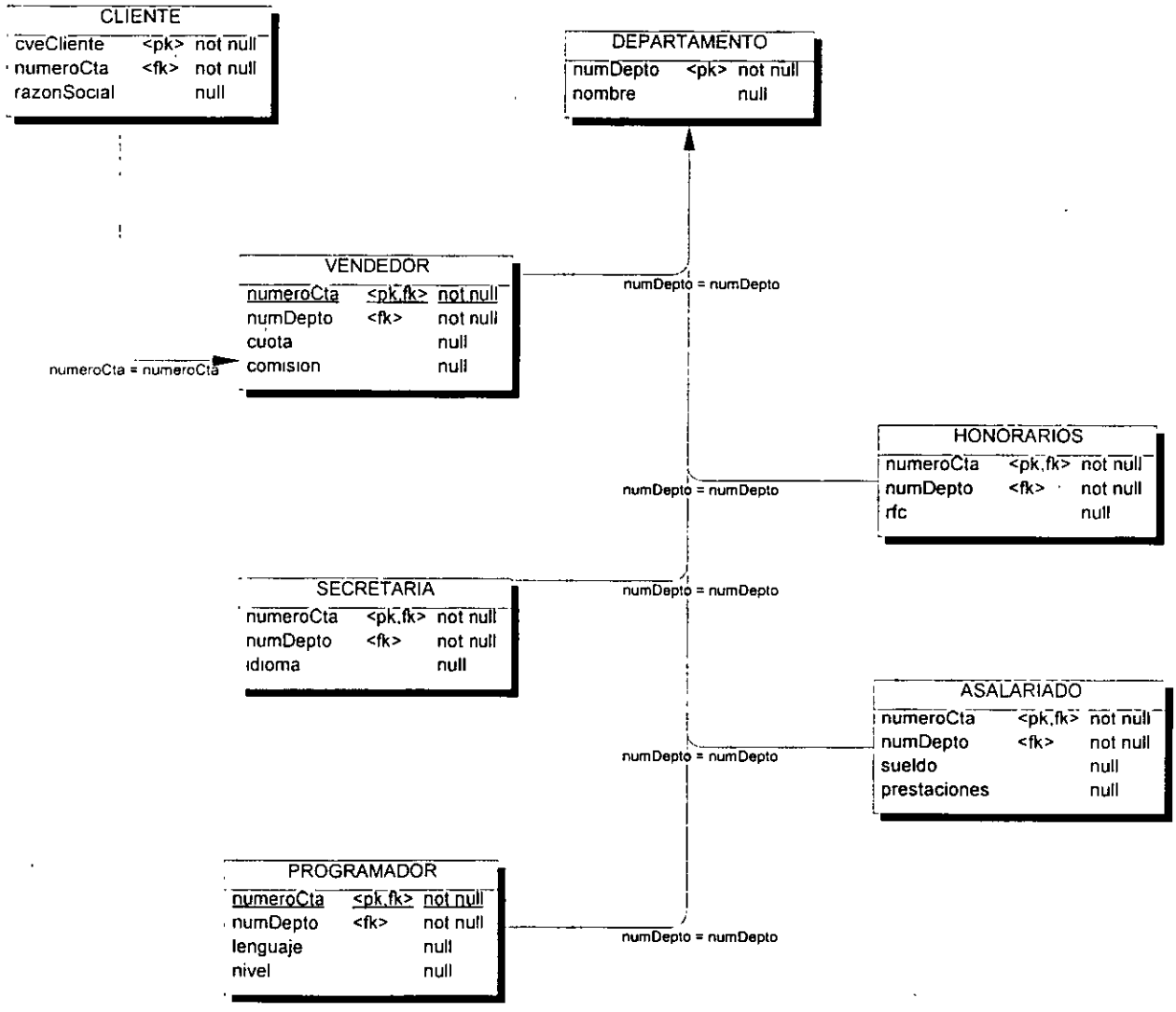
1. Generalización

Las subentidades no generan tablas, solamente se genera una tabla correspondiente a la superentidad, la cuál tiene todos los atributos comunes, así como los particulares a las subentidades, estos últimos deben indicarse como NULL (aceptan valores nulos).



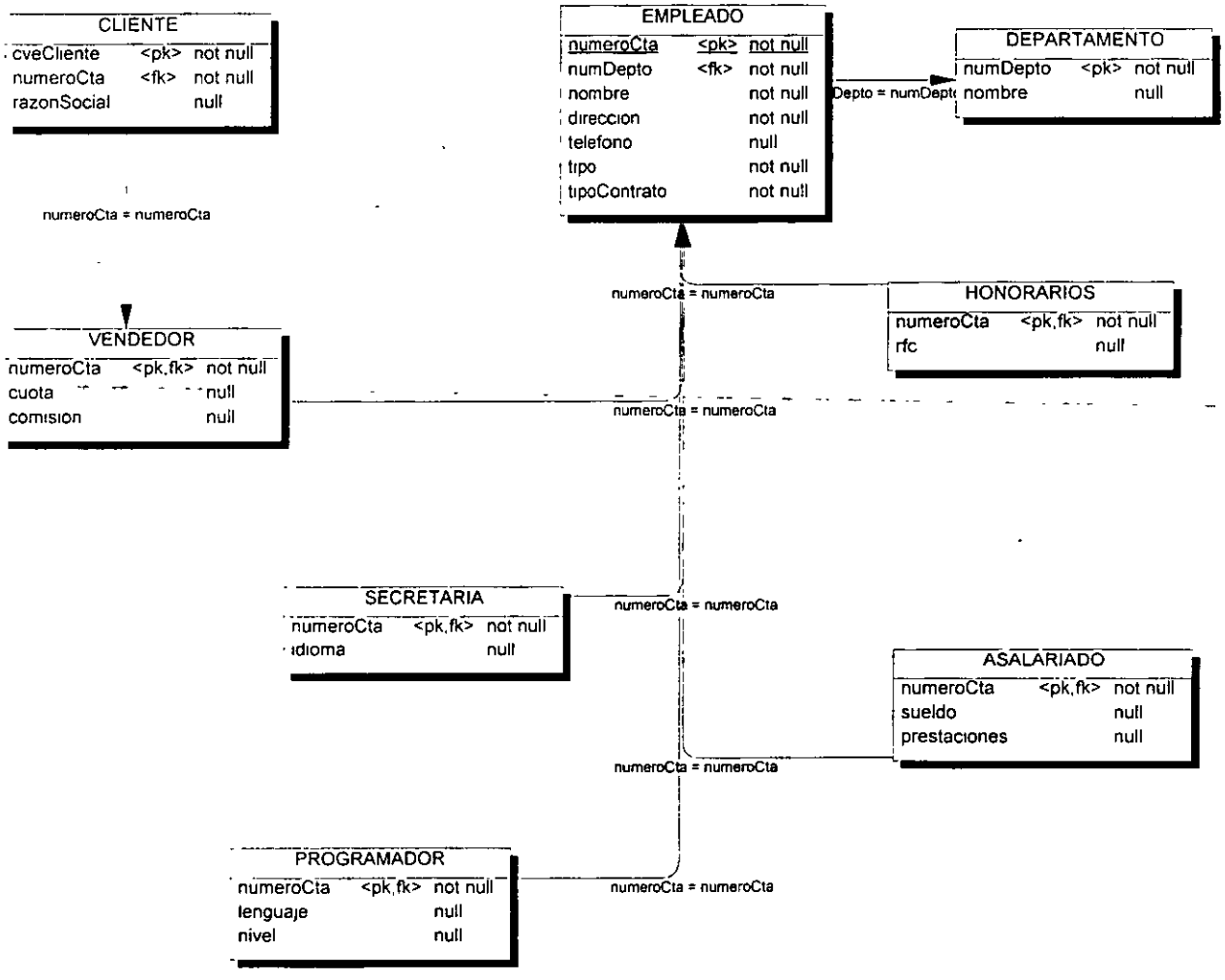
2. Especialización total

Se generan tablas para cada una de las subentidades, las cuales heredan los atributos, la llave primaria y las relaciones de la superentidad. No se crea una tabla para la superentidad.



3. Especialización parcial.

Para cada subentidad se genera una tabla cuyos atributos son aquellos particulares a la subentidad. Esta tabla tiene como llave primaria, la llave primaria de la superentidad y mantiene solamente las relaciones propias a ella. Para la superentidad se genera una tabla con los atributos y relaciones comunes a las subentidades.



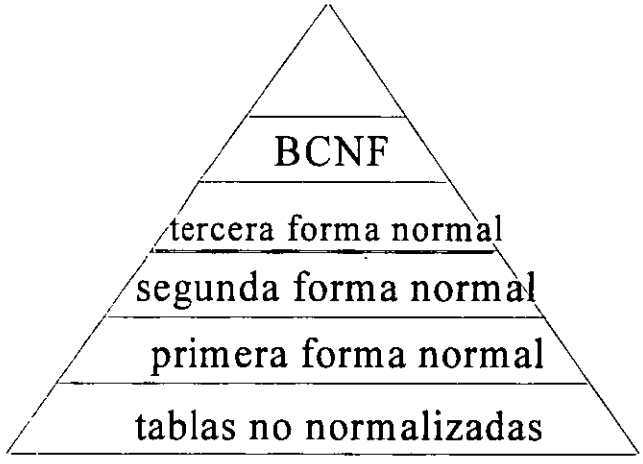
NORMALIZACION

Normalización es una técnica desarrollada para asegurar que las estructuras de datos sean eficientes. Los beneficios de la Normalización son:

- Minimiza la redundancia
- Libera de dependencias indeseables de inserción, borrado y actualización.
- Minimiza la reestructuración de datos cuando se introduce algo nuevo. Se mejora la independencia de datos, permitiendo que las extensiones a la base de datos tengan poco o ningún efecto sobre los programas o aplicaciones que tienen acceso a ella.
- No se introducen restricciones artificiales a las estructuras de datos.

Aunque se han definido más estados de normalización, sólo se han aceptado ampliamente tres. Estos se conocen como **primera, segunda, y tercera formas normales**, o 1NF, 2NF y 3NF respectivamente.

El normalizar es un proceso ascendente, en el que se parte de un universo de relaciones y atributos, y se avanza de forma en forma hasta llegar a la tercera forma normal.



Las etapas de normalización se muestran adelante con un ejemplo, dada la relación no normalizada:

ORDEN (#orden, fecha, #proveedor, nombre_proveedor,
dirección_proveedor, #producto, descripción_producto,
cantidad_producto, precio_total_producto, precio_total_orden)

Primera Forma Normal

Un registro en primera forma normal no incluye grupos repetidos. Es decir cada uno de sus campos debe tener un sólo valor.

En la relación ORDEN, se observa que para una misma orden habrá varios productos, por lo que #producto y otros atributos serán grupos repetidos. En 1NF habría que separar:

ORDEN (#orden, fecha, #proveedor, nombre_proveedor, dirección_proveedor, precio_total_orden)

PRODUCTO_ORDENADO (#orden, #producto, descripción_producto, precio_producto, cantidad_producto, precio_total_producto)

Segunda Forma Normal

Cada atributo depende de la totalidad de la llave, y no de sólo de una parte de ella.

Se puede observar en PRODUCTO_ORDENADO que descripción_producto depende sólo de #producto, y no tiene que ver con #orden. En 2NF quedaría:

ORDEN (#orden, fecha, #proveedor, nombre_proveedor, dirección_proveedor, precio_total_orden)

PRODUCTO (#producto, descripción_producto, precio_producto)

PRODUCTO_ORDENADO (#orden, #producto, cantidad_producto, precio_total_producto)

Tercera Forma Normal

Todos los atributos dependen solamente de la llave y no de otros atributos no llave.

En la relación ORDEN se presenta este problema:

nombre_proveedor y direccion_proveedor dependen de #proveedor

De modo que las tablas en 3NF quedarían como:

ORDEN (#orden, fecha, #proveedor, precio_total_orden)

PROVEEDOR (#proveedor, nombre_proveedor, dirección_proveedor)

PRODUCTO (#producto, descripción_producto, precio_producto)

PRODUCTO_ORDENADO (#orden, #producto, cantidad_producto, precio_total_producto)

LABORATORIO

1. Implementar todas las relaciones muchos a uno.

2. Implementar las relaciones muchos a muchos.

3. Implementar las relaciones uno a uno.

4. Implementar los esquemas de superentidades y subentidades. Argumente la opción elegida para la implementación.

5. Se incluirán atributos a las tablas de la siguiente manera:

INVOLUCRADO	**PARTICIPA**	**PERSONA**
VIN	PERSONA_ID	PERSONA_ID
EVENTO_ID	EVENTO_ID	NOMBRE_PERSONA
DESCRIPCION_DAÑO	ALCOHOL	CALLE
CIA_GRUA	NUM_LICENCIA	CIUDAD
NUMERO_PLACA	NOMBRE_PERSONA	ESTADO
		CODIGO_POSTAL

¿En qué forma normal se encuentra cada tabla? Si la tabla no está en 3FN, ¿cómo se puede lograr esta forma?

Para la tercera tabla, suponer que en CODIGO_POSTAL se encuentran de manera implícita las identificaciones del estado y ciudad.

Consideraciones de Performance en el Diseño de Bases de Datos Relacionales

OBJETIVO:

En este capítulo se estudiaran las técnicas para la obtención de un buen Diseño de Bases de Datos Relacionales.

En este capítulo el asistente:

- Conocerá el concepto de redundancia.

- Aprenderá a determinar las características que debe cumplir un buen Diseño de Bases de Datos.

- Aprenderá a aplicar técnicas de desnormalización en el Diseño de Bases de Datos para mejorar el rendimiento de las aplicaciones.

REDUNDANCIA

La repetición de información no implica necesariamente redundancia. La redundancia se puede definir como la repetición de hechos.

Suponga los dos casos siguientes:

1.

EMPLEADO

no_empleado	nombre	departamento
10	Antonio Chávez Flores	Sistemas
15	Edwin Navarro Pliego	Administración
20	Jéssica Briseño C.	Sistemas
25	Ramón Ramírez Hdez.	Sistemas

En este primer caso existe repetición de información (Sistemas) pero no redundancia.

2.

EMPLEADO

no_empleado	nombre	num_depto	nombre_depto
10	Antonio Chávez Flores	5	Sistemas
15	Edwin Navarro Pliego	7	Administración
20	Jéssica Briseño C.	5	Sistemas
25	Ramón Ramírez Hdez.	5	Sistemas

En este caso se repite el hecho de que el departamento 5 es el departamento llamado Sistemas.

Un Diseño de Bases de Datos con tablas normalizadas minimiza la redundancia de datos, pero, ¿qué sucede con el rendimiento?

Pueden existir consultas críticas que involucren la obtención de datos de varias tablas, en este caso el costo de los joins puede ser muy alto. Por ejemplo, obtener el estado de cuenta de un cliente a partir de las tablas CLIENTE, MOVIMIENTOS, SALDO, CUENTA.

Existen algunas aplicaciones que requieren el cálculo de datos en base a ciertas columnas de una o varias tablas. El cálculo de estos datos implica tiempo. Por ejemplo, el cálculo del salario mensual de un empleado en base a los campos prestaciones, salario_gravable, bono, incentivo.

En muchas ocasiones solamente una parte de los datos de una tabla se utiliza constantemente.

CARACTERISTICAS DE UN BUEN DISEÑO

Un buen diseño debe:

- **Minimizar la redundancia de información.**

- **Fácil de comprender**

  Las tablas que componen el diseño no deben de estar demasiado fragmentadas, ni tampoco deben contener demasiada información, ya que sería difícil determinar la entidad que representan.

  Es necesario que las tablas tengan una estructura que permitan fácilmente determinar la información que representan.

- **Ayudar a que el performance de las consultas y movimientos en la BD sea aceptable**

  La información no utilizada debe separarse.

  La información que se accesa junta debe permanecer como tal.

Un Diseño de Bases de Datos difícilmente puede satisfacer todos estos requerimientos.

ASPECTOS A EVALUAR

En un ambiente de operación real pueden existir aplicaciones de cualquier tipo sobre la Base de Datos, por lo que hay que determinar cuales son las consultas o movimientos más críticos e importantes. Para estos hay que determinar:

- La frecuencia con que se llevan a cabo
- El número de usuarios concurrentes que las ejecutan
- El volúmen de información procesada
- Procesos en ejecución cuando estos se realizan
- El tipo de operación (en línea o batch)
- El usuario que las ejecuta
- El tiempo de respuesta requerido

VENTAJAS Y DESVENTAJAS DE LA NORMALIZACION

VENTAJAS

- Reduce la redundancia de datos.

- Las tablas son más pequeñas, lo que implica que la Base de datos ocupa menos espacio de disco.

- Como las tablas son más pequeñas los accesos a disco para leer una tabla se reducen.

- Las modificaciones son más eficientes.

- Puede reducir la contención de datos.

Suponga los dos esquemas siguientes:

1. Tablas normalizadas

   EMPLEADO (noCta, nombre, dirección, salario, noDepto)

   DEPTO (noDepto, nombre)

2. Tabla no normalizada

   EMPLEADO (noCta, nombre, dirección, salario, noDepto, nombreDepto)

suponga que existen 12000 empleados asignados a 50 departamentos, y que el departamento de "Sistemas" cambiará de nombre, ahora se llamará departamento de "Informática y Sistemas". ¿Bajo qué esquema es más fácil hacer el cambio conservando la integridad de la Base de Datos?

DESVENTAJAS

- Debido a que el número de tablas se incrementa, para las consultas que requieren de mucha información, muy probablemente se tengan que llevar a cabo joins entre varias tablas.

- Una consulta que involucra joins tiene, generalmente, un tiempo de respuesta mayor a una consulta sobre una sola tabla.

Suponga los siguientes esquemas:

1. Tablas Normalizadas

PELICULA (noPel, nombre, ...)
RENTA (noPel, cveSocio, noCopia)
SOCIOS (cveSocio, nombre, direccion, ...)

2. Tablas no normalizadas

PELICULA (noPel, nombre, ...)
RENTA (noPel, cveSocio, nomPel, nomSocio, noCopia)
SOCIOS (cveSocio, nombre, direccion, ...)

Suponga que se desea hacer un reporte que muestre las películas rentadas y el nombre del socio que las rento, ¿bajo qué esquema sería más eficiente la consulta?

DESNORMALIZACION

La desnormalización tiene como objetivo obtener un mejor performance de cierto tipo de aplicaciones sobre la Base de Datos.

Cuando se desnormaliza se debe partir de que existe un diseño normalizado.

La desnormalización se da a nivel de columna o tabla.

La desnormalización mejora el performance de cierto tipo de aplicaciones, por lo que hay que tener conocimiento de como se van ha utilizar los datos.

La desnormalización reduce el número de joins, el uso de llaves foráneas y se puede reducir el número de tablas.

El costo de desnormalizar se refleja al momento de hacer actualizaciones y mantener la integridad de la Base de Datos.

Tácticas de Desnormalización

- Redefinición de columnas.

- Redefinición de tablas.

Duplicación de columnas

Las columnas que se van a duplicar deben de cumplir con:

- ser del mismo tipo
- tener el mismo dominio
- tener el mismo nombre

Las columnas a duplicar no deben ser muy volátiles, es decir, no deben cambiar constantemente, ya que de lo contrario se tendría que requerir un control más estricto para mantener la integridad de la Base de Datos.

La duplicación puede darse mediante una copia exacta o como una columna calculada con base en otras.

Duplicación de columnas, copias exactas

Ejemplo:

PELICULA (cvePel, nomPel, ...)

RENTA (cvePel, cveSocio, **nomPel**, **nomSocio,** noCopia)

SOCIOS (cveSocio, nomSocio, direccion, ...)

En este caso el reporte de películas rentadas indicando el nombre del socio tiene un mejor tiempo de respuesta.

Las columnas repetidas no son volátiles, ¿Cambia el nombre del socio respecto a su clave de socio?

Ejemplo:

PELICULA (cvePel, nomPel,precio ...)
COPIA (cvePel, noCopia,**precio**)

En este caso la columna precio es muy volátil, por lo que para mantener la integridad de la Base de Datos, cada vez que cambie el costo de la renta de una pelicula, se debería de hacer el cambio en las dos tablas. En el caso de un esquema normalizado, ¿cómo se mantiene la integridad en este caso?

Duplicación de columnas, columnas derivadas

Una columna derivada es aquella cuyos valores son obtenidos de un cálculo con base en los registros o columnas de una o varias tablas:

- Suma de varias columnas
- Conteo de registros
- etc.

Ejemplos:

PELICULA (cvePel, nomPel, genero, **numCopias**)

COPIA (cvePel, noCopia)

ALUMNO (noCta, nombre, **promedio,** ...)

CURSO (noCta, cveMat, calificación, ...)

MATERIA (cveMat, nombre, ...)

EMPLEADO (noEmp, nombre, salBase, comisión, prestaciones, **salTotal)**

Redefinición de columnas

La redefinición de columnas se da sobre llaves primarias cuando estas son muy grandes.

Se redefine la llave primaria para que esta sea más pequeña, lo que provoca que:

- Las llaves foráneas en otras tablas sean más pequeñas.

- Los joins sean más rápidos, ya que se hacen con campos de relación más pequeños.

Ejemplo:

Suponga el siguiente esquema:

PRODUCTO (**tipo, marca**, precio, ...)
VENDE (***numTienda, tipo, marca***, cantidad, fecha, ...)
TIENDA (**numTienda**, dirección, tel, ...)

Redefiniendo la llave primaria:

PRODUCTO (**cvePro**,tipo, marca, precio, ...)
VENDE (***numTienda, cvePro,*** cantidad, fecha, ...)
TIENDA (**numTienda**, dirección, tel, ...)

Redefinición de tablas

El performance de las aplicaciones de la Base de Datos se puede ver afectado por alguna de las siguientes razones:

- Los renglones contienen más atributos de los que se necesitan, de modo que el tamaño del registro es muy grande y el tiempo de lectura de n registros se ve afectado.

- La tabla contiene renglones que son muy poco utilizados, lo que provoca que la lectura de datos sea más lenta.

Existen dos métodos para redefinir tablas:

- Duplicación (por renglón o por columna)
- Segmentación (por renglón o por columna)

Para elegir entre un método y otro se deben considerar los siguientes aspectos.

- Integridad de la Base de Datos.
- Redundancia.
- Espacio en disco adicional.

Duplicación/segmentación de tablas por columna

Ejemplo:

La tabla de empleados contiene 20 columnas, por lo que cada registro ocupa 350 bytes. Las aplicaciones más comunes solamente requieren del nombre, número de cuenta, salario y número de departamento del empleado; estos cuatro campos ocupan en total 40 bytes. La llave primaria de la tabla es el número de cuenta:

EMPLEADO (noCta, nombre, direccion, tel, edad, sexo, salario, noDepto, fechaContrato, nomEsposa, numHijos, ...)

Duplicación:

EMPLEADO (noCta, **nombre, salario, noDepto**, direccion, tel, edad, sexo, fechaContrato, nomEsposa, numHijos, ...)

EMP_DUP (noCta, **nombre, salario, noDepto**)

Ventajas:

- Se reduce la contención de datos

- En el caso de que se requiera hacer una consulta que involucre a todas las columnas, solamente se definirá sobre la tabla EMPLEADO. No se requieren joins.

Desventajas:

- Requiere más espacio en disco

- Se requiere de mayor esfuerzo para mantener la integridad. Cuando se cambie el salario o el número de departamento de un empleado, el cambio se deberá realizar en ambas tablas.

La duplicación de tablas no representa un esquema de desnormalización.

Para mantener el modelo conceptual que percibe el usuario, se pueden definir vistas sobre las tablas resultantes de la duplicación.

Segmentación:

EMP_PERSONAL (noCta, direccion, tel, edad, sexo, fechaContrato, nomEsposa, numHijos, ...)

EMP_NOMINA (noCta, nombre, salario, noDepto)

Ventajas:

- Reduce la contención de datos.
- Requiere menos espacio en disco.
- Mantener la integridad no requiere de esfuerzo adicional.

Desventajas:

- Una consulta que involucre la información de todas las columnas requiere de un join.

Duplicación/segmentación de tablas por renglón

Ejemplo:

La empresa X mantiene una tabla de empleados en donde se tienen registrados todos los empleados que alguna vez han trabajado en dicha empresa. La tabla contiene 100000 registros; sin embargo, las aplicaciones de nómina y asistencia que son las más importantes y que requieren un buen tiempo de respuesta, solamente utilizan la información de los 5000 empleados que actualmente se encuentran contratados.

Duplicación:

EMPLEADO (noCta, nombre, ...) 100000 registros

EMP_ACTIVO (noCta, nombre, ...) 5000 registros

Ventajas:

- Se reduce la contención de datos.
- En el caso de que se requiera hacer una consulta que involucre a todos los registros, solamente se definirá sobre la tabla EMPLEADO. No se requieren uniones.

Desventajas:

- Requiere más espacio en disco
- Se requiere de mayor esfuerzo para mantener la integridad. Cuando se cambie los datos de un empleado activo, el cambio se deberá realizar en ambas tablas.

Segmentación:

EMP_INACTIVO (noCta, nombre, ...) 95000 registros

EMP_ACTIVO (noCta, nombre, ...) 5000 registros

Ventajas:

- Reduce la contención de datos.
- Requiere menos espacio en disco.
- Mantener la integridad no requiere de esfuerzo adicional.

Desventajas:

- Una consulta que involucre la información de todos los registros requiere de la union de las dos tablas.

En general es más recomendable la segmentación en ambos casos.

LABORATORIO

Realice los cambios necesarios en el Diseño de la Base de Datos para cumplir con los siguientes requerimientos:

1. Los oficiales del DMV cuentan en sus patrullas con una computadora que les permite accesar la Base de Datos. Es importante, que los oficiales puedan determinar rápidamente en base al número de placa de un vehículo, lo siguiente:

- El número de veces que se ha infraccionado
- El número de veces que ha participado en un accidente

Y también es importante conocer para un conductor, tomando como base su nombre, sus antecedentes en accidentes e infracciones.

2. Para realizar citatorios a la Corte es determinante poder determinar el número de puntos acumulados por un conductor.

3. El DMV desea guardar un histórico de todas las licencias que ha tenido un conductor.

4. Se desea también, tener un histórico que permita determinar todos los dueños que ha tenido un vehículo.

Página intencionalmente blanca.

Diseño Físico de Bases de Datos

OBJETIVO:

En este capítulo se presentaran los aspectos importantes que se deben considerar al implementar una Base de Datos.

En este capítulo el asistente:

- Entenderá el concepto de Integridad.

- Conocerá la función de los índices.

- Conocerá la función de los constraints.

- Conocerá la función de los triggers.

DISEÑO FISICO DE BASES DE DATOS

Implementar una Base de Datos es transformar el Modelo Lógico de la misma en una serie de estructuras físicas que la representen. El Diseño Físico es el proceso de determinar las estructuras de datos y mecanismos a ser utilizados para la implementación de la Base de Datos.

El Diseño Físico de la Base de Datos es particular a un DBMS.

Para llevar a cabo un buen Diseño Físico es conveniente conocer las aplicaciones críticas que accesarán la base de datos.

El Diseño de la Base de Datos incluye conceptos de:

- Distribución de la Base de Datos
- Almacenamiento de la Base de Datos
- Tipos de datos
- Estructuras de acceso directo a los datos
- Integridad
- Esquema de seguridad
- Esquemas de respaldo y recuperación

Bases de Datos Distribuídas

El Modelo Lógico de la Base de Datos puede ser implementada físicamente en varias computadoras conectadas en red. El resultado es, una Base de Datos Distribuída.

Un sistema distribuído de bases de datos consiste en un conjunto de localidades, cada una de las cuales mantiene un sistema de bases de datos local.

Cada localidad puede procesar transacciones locales o bien, puede participar en la ejecución de transacciones globales, es decir, aquellas que acceden datos de varias localidades.

LOCALIZACION DE LOS DATOS

Uno de los aspectos más importantes en un esquema distribuído es determinar en donde se localizarán los datos. Existen tres alternativas:

1. Replicación

En este esquema se mantienen copias idénticas de las tablas en diferentes localidades, lo que resulta en repetición de la información.

Ventajas y desventajas:

- Disponibilidad
- Mayor paralelismo
- Tiempo de latencia en las actualizaciones
- Sincronización de las actualizaciones

2. Fragmentación

En este esquema las tablas se dividen en varios fragmentos. Cada fragmento se almacena en una localidad diferente.

La fragmentación puede ser:

- Horizontal
- Vertical
- Mixta

3. Esquema híbrido

Este esquema es una combinación de los dos anteriores, las tablas se dividen en fragmentos para los cuales se mantienen copias idénticas en las diferentes localidades.

Para el diseño de esquemas distribuídos se deben considerar los siguientes aspectos:

- Transparencia y autonomía de la red
- Procesamiento distribuído de consultas
- Esquemas de recuperación
- Protocolos para manejo de transacciones
- Control de concurrencia
- Manejo de bloqueos

Almacenamiento de la Base de Datos

Uno de los aspectos importantes en el diseño físico de la base de datos es determinar los tipos de medios de almacenamiento para los datos. Generalmente se utiliza almacenamiento en disco.

Hay que determinar primero, el tamaño que ocupará la base de datos incluyendo tablas, índices, catálogo, etc.

Una vez que se ha determinado el espacio en disco de la base de datos, se necesita seleccionar el número, tipo y configuración de los discos a ser utilizados.

Hay que diseñar un esquema de objetos a discos para poder determinar que objetos residirán en que discos.

Tipos de Datos

Desde la creación del Modelo Conceptual, a cada uno de los atributos de las entidades se les puede asignar un tipo de dato; sin embargo, es hasta la etapa de diseño físico donde se debe hacer la elección correcta del tipo de dato asociado a cada una de las columnas de las tablas de la base de datos.

Los tipos de datos son particulares a un DBMS.

Al momento de elegir los tipos de datos se debe considerar:

- Espacio en disco que requieren
- Dominio asociado
- Overhead ocacionado por su representación
- Compatibilidad con los ambientes de desarrollo
- Restricciones de procesamiento

Estructuras de acceso directo a los datos

Muchas consultas hacen referencia a solo una pequeña parte de los datos de una tabla. Es ineficiente que el sistema tenga que leer todos los renglones y comprobar las condiciones de la consulta. Lo ideal es que el DBMS pueda localizar directamente los registros.

Para permitir estas formas de acceso, existen estructuras de datos adicionales asociadas a las tablas.

INDICES

Un índice es una estructuras de datos, generalmente un árbol binario o un árbol B+, que establece un orden lógico de los registros de una tabla de acuerdo a las columnas en las cuales se define el índice.

Existen dos tipos de índices:

- **Clustered**, el orden lógico del índice corresponde al orden físico de los datos.

- **Nonclustered**, el índice solamente establece un criterio de búsqueda, que corresponde con el orden físico de los datos.

HASH

Una desventaja de los esquemas de indexado es que se debe accesar a la estructura de índices para localizar los datos. La técnica de asociación (hashing) consiste en la construcción de llaves de busqueda a partir de una función de asociatividad.

Es importante diseñar correctamente la función de asociatividad. Existen varias técnicas conocidas como **funciones de asociación dinámica**, que permiten modificar de manera dinámica la función de asociación para compensar el crecimiento de los datos.

Al elegir las estructuras de acceso directo a los datos se debe considerar:

- Tiempos de acceso
- Tiempos provocados por insert, update y delete.
- Espacio requerido
- Distribución de los registros

Integridad

El concepto de Integridad es sencillo: los datos deben ser válidos.

Ejemplos:

- No pueden existir renglones repetidos en una tabla.

- El número de departamento de un empleado debe de pertenecer a un departamento válido.

- En una Base de Datos de Reservación de líneas aéreas, el número de reservaciones no debe ser mayor que el cupo en un vuelo.

- El salario de un empleado debe estar entre N$ 1,000 y N$ 8,000.

Integridad de Entidades

Cada renglón en una tabla debe ser único.

Si la llave primaria es única, se garantiza que un renglón es único en una tabla.

Los **constraints** son reglas que se indican al momento de crear una tabla y que sirven para la implementación de la integridad de entidades. Para el caso de asegurar que la llave sea única.

Otra forma de implementar la Integridad de Entidades es mediante el uso de índices únicos.

Integridad de Dominio

Cada valor de un campo pertenece a un dominio previamente definido.

En un principio el dominio para un campo puede ser limitado por el tipo de dato al que esta asociado; sin embargo, muchas veces hay que restringir más el dominio.

Por ejemplo, si se desea que el salario de un empleado este en el rango de 1,000 a 8,000 y que el RFC tenga el formato adecuado, se deben crear restricciones de dominio adicionales a las que impone un tipo de dato.

Uno de los mecanismos para implementar la Integridad de Dominio lo constituyen los constrains.

Otro mecanismo para implementar la Integridad de Dominio es el uso de **reglas.**

Integridad Referencial

Cada llave foránea debe estar asociada a una llave primaria válida, o debe tener un valor NULL.

Para poder mantener esta relación se pueden considerar las siguientes reglas:

Para la llave foránea:

- Insert/Update

  - No permitir el insert/update si no existe una llave primaria asociada (RESTRICT).

  - Asignar valor nulo a la llave foránea sino existe la correspondiente llave primaria(NULLIFY).

  - Asignar valor default a la llave foránea sino existe la correspondiente llave primaria(DEFAULT).

  - Si la llave primaria asociada a la llave foránea no existe, dar de alta la tupla asociada a la llave primaria.

- Para Delete no se debe considerar acción alguna.

Para la llave primaria:

- Para Update:

  - Modificar todas las llaves foráneas asociadas (UPDATE CASCADE).

  - Evitar la modificación si existen llaves foráneas asociadas (RESTRICT).

- Para Delete:

  - Borrar todas las llaves foráneas asociadas (DELETE CASCADE).

  - Evitar el borrado si existen llaves foráneas asociadas (RESTRICT).

  - Asignar NULL a las llaves foráneas asociadas (NULLIFY).

  - Asignar un valor de default a las llaves foráneas asociadas (DEFAULT).

- Para Insert no se debe considerar acción alguna.

Las reglas de Integridad Referencial se pueden implementar con constrains.

Otro mecanismo para implementar las reglas de Integridad Referencial son los **triggers**.

Los triggers permiten la implementación de reglas de Integridad mucho más complejas conocidas como **Integridad del Negocio** y que no se pueden implementar mediante algún otro mecanismo.

Por ejemplo, en una Base de Datos de reservación de vuelos, es necesario verificar que las reservaciones no sobrepasen el cupo del vuelo.

La integridad de la Base de Datos puede ser implementada en el DBMS o en las aplicaciones. Para determinar en donde debe ser implementada se debe considerar:

- Necesidad de centralización del control de la integridad
- Necesidad de mantener la integridad de la base de datos en cualquier instante.
- Tiempo de proceso adicional
- Complejidad de las reglas de integridad
- Mantenimiento

EJERCICIOS

EJERCICIO 1: Departamento de vehículos de California

El Departamento de Vehículos Motorizados (DMV) del estado de California está desarrollando una nueva base de datos, para auxiliar tres áreas funcionales, relacionadas entre si:

1. Pago anual de derechos de vehículos.
2. Expedición de licencias de conductores.
3. Seguimiento de accidentes e infracciones.

Para tal propósito, se han especificado los siguientes requerimientos, basados en entrevistas con el personal adecuado, y basados también en la revisión de los documentos del DMV.

PAGO ANUAL DE DERECHOS DE VEHICULOS

La Unidad de Registro de Vehículos del DMV, lleva el control de información de automóviles, autobuses, motocicletas, camiones de carga, aviones y barcos. Cada vehículo tiene grabado en el bloque del motor los siguientes datos: año de fabricación, marca, modelo y clave de identificación, proporcionado por la compañía que lo fabricó. Esta clave es única para cada vehículo. La clave de identificación del vehículo se conoce en el DMV como VIN (número de identificación del vehículo). La mayoría de los vehículos tienen placas, con excepción de los aviones y barcos.

En California, los vehículos deben pagar derechos anualmente; el DMV debe almacenar la fecha de pago de derechos más reciente, nombre completo y dirección (para efectos dfe correo) del dueño, y precio estimado de venta del vehículo. Las tarifas de derechos son calculadas como la cantidad que resulte mayor entre $15.00 dólares o el 4% del precio de venta del vehículo. También se debe tener almacenado el estado en el cual fué registrado el vehículo.

EXPEDICION DE LICENCIAS DE CONDUCTORES

El DMV administra y almacena la información sobre los exámenes que se aplican a los conductores de vehículos y mantiene además, un registro de las licencias.

Cada exámen consiste en 5 partes: estacionamiento, vuelta en U, reversa, uso de señales direccionales y control de velocidad. El exámen se califica en una escala del 0 al 100 por el oficial examinador, excepto en la parte de control de velocidad, la cual es evaluada en una escala de 0 a 30.

En algunos casos, los conductores de vehículos reprueban el exámen varias veces antes de lograr aprobarlo. Cada persona puede realizar sólo un exámen por día. Se debe llevar un registro histórico de exámenes, tanto aprobados como reprobados. Para análisis estadístico interno, el DMV lleva registro del nombre del Centro DMV que efectúa cada exámen, así como del oficial responsable de la aplicación del mismo.

El DMV mantiene una lista de todos los oficiales de policia de California, por número de placa, departamento al que pertenecen, rango y nombre. Frecuentemente se encuentran números de placa duplicados entre departamentos (y en algunas ocasiones dentro del mismo departamento), de manera que el número de placa no es suficiente para identificar a oficiales; el rango y departamento son también necesarios.

Cuando un conductor presenta un exámen, se registran en el sistema su nombre completo, dirección, fecha de naciemiento y color de ojos y cabello. Se le asigna al conductor un número único de licencia, y una fecha en que expira (normalmente 4 años después de la fecha de expedición), lo cual es registrado. Las licencias de conductores son expedidas con uno o más códigos de autorización según la función del vehículo, como 'M' (motocicleta), 'CF' (comercial/flete), o 'CP' (comercial/pasajeros), o bien, se expide una

licencia sin código de autorización para quienes han hecho el exámen pro aún no lo han pasado.

SEGUIMIENTO DE ACCIDENTES E INFRACCIONES

La Oficina de Registros del DMV mantiene una lista con la historia de todos los accidentes y otra con las infracciones reportadas en California.

El reporte de accidente es escrito por un oficial en el lugar en que ocurre, e incluye los nombres y números de licencias de los conductores involucrados, así como los números de las placas de los vehículos. El oficial también debe escribir cualquier daño a cada uno de los vehículos o heridas de los conductores.

Si alguno de los vehículos se encuentra fuera de sus estado, se documenta el estado de registro del vehículo. Si el oficial considera que alguno de los conductores está en estado de ebriedad, puede aplicar un examen de sobriedad, y anotar en el reporte el nivel de alcohol en sangre del conductor. También se anotan en el reporte la fecha, hora, ubicación, condiciones climatológicas (códigos de visibilidad, lluvia, y condiciones de pavimento), y una narración de como ocurrió el accidente.

Por otra lado, el DMV registra las infracciones cometidas, contenidas en el Código de Tráfico del Estado de California, como exceso de velocidad, estacionamiento en lugar prohibido, conducción en estado de ebriedad, etc.

La información sobre infracciones es capturada en el sistema del DMV por oficinistas, de acuerdo a la información de las boletas de infracción. Estas boletas contienen el número de boleta de infracción, oficial, fecha, hora, ubicación, placa de automóvil, el estado de registro del vehículo y el tipo de infracción, así como el artículo del Reglamento de Tránsito del Estado de California que se está desobedeciendo. Si el conductor se encuentra presente (por ejemplo, infracción por exceso de velocidad, vuelta prohibida, etc.) el oficial anota también la clave de su licencia de conducir, nombre y

resultado del examen de sobriedad (cuando se considere necesario).

En algunas ocasiones, una infracción involucra daño al vehículo, lo cual es descrito en la boleta de infracción y almacenado en la base de datos del DMV. Si la infracción es tal que el vehículo sea removido del lugar, se registra también el nombre de la compañía de gruas (No son del DMV, sino particulares contratadas por el Departamento), para beneficio del conductor.

Algunas infracciones, las que así lo ameriten, acumulan puntos, los cuales se cargan al registro del conductor, para posibles citatorios en la Corte.

Departamento de vehículos de California

REQUERIMIENTOS ADICIONALES

Los tipos especiales de vehículos tienen requerimientos de datos especiales.

El DMV debe almacenar el desplazamiento y clase de motor (diesel, gas, etc.) de los barcos.

Para autobuses, camiones de carga, autos y motocicletas, se tiene que almacenar el dato de caballos de fuerza (proporcionado por el fabricante), y la fecha de la última verificación de emisión de contaminantes.

Para autobuses y camiones de carga, se debe almacenar el número de ejes, para el cálculo de la cuota de caminos.

El DMV registra el aeropuerto al que está asignado cada avión, y su código de la Administración de Aviación Federal, pero no almacena accidentes o infracciones para barcos o aviones.

Para taxis, aviones comerciales, autobuses de transporte público y barcos de transporte colectivo (ferries), el DMV también expide y registra un código de ocho caracteres, denominado "medallon", la fecha de la inspección más reciente y la capacidad del vehículo (máximo número de pasajeros). Se requiere autorización especial para operar vehículos comerciales tales como taxis y autobuses.

EJERCICIO 2: Control de Departamentos

... parece que nadie puede estar en su lugar, siempre entrando y saliendo, no es posible mantenerlo manualmente mas, es demasiado trabajo ...

Tengo que administrar 1480 departamentos de 15 edificios, bueno 1479 desde que yo vivo en el 149. Pero de todas maneras, son demasiados. Algunos con una recamara, algunos con dos recamaras y otros con tres recamaras, algunos con chimenea, algunos con camas de agua, algunos con lavaplatos y otros con calefacción, es realmente complicado controlarlos.

Todo parece indicar que necesito una computadora. Algo que me diga cuantos inquilinos tengo en cada departamento, creo que lo ideal es uno en cada departamento, pero no me interesa mucho quien vive ahí, solo quiero saber quien paga. Pero no, este sistema no va a controlar los pagos, solo lo quiero como referencia, su nombre, el numero de alguna identificación, etc.

Pero también quiero ser capaz de saber el número de servicios que tiene cada departamento, ya sabes, las camas, los lavaplatos, la chimenea, etc., además de el edificio en donde se localizan.

Y también necesito saber que departamentos están vacíos, y si mis inquilinos están rentando mas de un departamento, ya sabes, no me gustaría que se hicieran malos manejos...

Y casi lo olvido, ¿recuerdas el gran estacionamiento cruzando la calle? Bueno, tenemos unos 300 lugares disponibles, pero estos lugares están reservados por los inquilinos, y a varios inquilinos no les gusta que coloquemos el número de departamento sobre el estacionamiento, ellos quieren que sea confidencial, entonces necesito controlar los estacionamientos también.

EJERCICIO 3: Base de Datos para control de un video club

El video club "**Patito S.A. de C.V**" usa actualmente un sistema de control de rentas de películas en base a los siguientes archivos:

SOCIOS	**RENTAS**
Nombre socio	Nombre Socio
Dirección	Nombre de película
Teléfono	Tipo
No. tarjeta	Formato
	Fecha de renta
	Fecha de devolución
	Importe

La utilización de estos archivos ha resultado ser impractica y costosa ya que se tiene gran redundancia, mucha inconsistencia y un alto costo de mantenimiento. De esta manera, el dueño desea un nuevo sistema en Bases de Datos ya que quiere eliminar estos problemas

Se deben considerar los siguientes requerimientos:

1. Se desea tener una clave de socio que cuente con la siguiente información: dirección, teléfono, numero de tarjeta de crédito, el nombre de dos titulares indicando cual es el principal.

2. Se tiene un lote de 2000 películas diferentes cada uno de las cuales tiene de 5 a 20 copias, para las cuales se desea saber su nombre, su genero (comedia, terror, infantil, suspenso, etc.), clasificación (A, B, C, XXX) y el director.

3. El precio que se debe pagar por la renta diaria de una película depende de la categoría de esta, se tienen 5 categorías diferentes (línea dorada, roja, super hits, etc.).
4. El dueño desea saber en un momento dado cuantas copias de una película se han rentado, cuales son estas y de que formato son (beta o VHS). Por otra parte desea saber la fecha a partir de la cual se puso a disposición de los socios una copia, esto con el fin de dar el mejor servicio y poder determinar cuales son las copias que ya se necesitan reemplazar.

5. Un socio puede rentar hasta tres películas; si no devuelve una película, se le aplica una multa por cada día que transcurra después de la fecha de entrega igual al 20% del precio por renta diaria de la película en cuestión.

EJERCICIO 4: Base de Datos para control de pedidos

Queremos diseñar una Base de Datos para almacenar información sobre nuestros clientes y sus pedidos.

Los artículos comercializados por nuestra organización son, algunos fabricados en fábricas propias y otros hay que adquirirlos con proveedores externos. Es necesario conocer los datos de los proveedores: razón social, dirección, teléfono, etc. para efecto de generar facturas, pedidos, etc. Nuestra Base de Datos no mantendrá la información de los pedidos que nosotros realizamos.

Cada cliente dispone de un crédito, un descuento por compras masivas, y de un saldo que varía conforme va pagando los pedidos entregados. Los pedidos se entregan en una dirección de entre varias posibles de cada cliente.

Los pedidos se componen de dos partes: un encabezado de pedido y varias líneas de detalle. El encabezado contiene la fecha del pedido, un número de pedido y el cliente que lo hace. Las líneas de detalle contienen la clave de identificación del articulo, la descripción del mismo y la cantidad pedida. Toda esta información se desea guardar en la Base de Datos.

Existen artículos que se deben vender en conjunto, es decir, el usuario final para poder utilizar un articulo necesita de otro, por lo que se deben proporcionar ambos. No debemos, en este caso, vender piezas sueltas.

Se desea también llevar un control del inventario de artículos. Para ello es necesario conocer el nivel mínimo de existencia, para que en caso de ser necesario, ya sea en forma automática o manual, con la información de la base de datos se genere un reporte indicando los pedidos que se deben realizar a nuestros proveedores.

La operación más importante en la organización es el manejo de los estados de cuenta de nuestros clientes, mensualmente se les hace una notificación de este aspecto. Por otra parte, se pretende que el departamento de ventas y compras pueda obtener de la base de datos:

- Reportes semanales de ventas por tipo de producto, cliente, fecha.
- Estimaciones de ventas en base a las ventas del mes anterior.
- Generación de pedidos a tiempo a nuestros proveedores.
- Listas de inventario de productos.
- Identificar pedidos de piezas sueltas, para productos que se deben vender en conjunto con otros.

En la empresa se manejan alrededor de 10000 pedidos diarios, por lo que es importante considerar un buen diseño de la base de datos, no solamente para mantener la integridad necesaria de los datos, sino también para que las aplicaciones y consultas del departamento de ventas y compras se realicen con un excelente tiempo de respuesta.

EJERCICIO 5: Sistema de Administración Escolar

La Universidad Héroes de América , UHA, tiene un sistema de administración escolar que esta hecho en base a manejo de archivos planos. La UHA ha adquirido equipo de cómputo del más moderno y quiere sacar el mayor provecho de éste. Para comenzar, se he decidido diseñar un nuevo sistema de administración escolar, pero para ello se plantea diseñar la Base de Datos.

Para esto se han proporcionado los siguientes requerimientos:

1. Las inscripciones se llevan a cabo semestralmente, cada semestre esta identificado por el año y el semestre (ej: 92-1, 92-2, 93-1, etc.). Por cada semestre es necesario conocer la fecha de inicio y finalización, esto para poder emitir constancias de inscripción.

2. La UHA cuenta con 9000 alumnos en promedio, cada uno es identificado por un número de cuenta, ademas se desea conocer su nombre, su dirección(calle, número, colonia y código postal), su teléfono, el nombre de su tutor, el número de créditos cursando en el semestre, el número de créditos acumulados, su promedio y el porcentaje de avance de su carrera.

3. Se imparten 10 carreras diferentes identificadas mediante una clave, de las cuales se necesita conocer su nombre, el total de créditos para aprobar la carrera, el costo de inscripción a la carrera, el total de alumnos inscritos en la carrera actualmente y el número de semestres en el que se cursa la carrera.

4. Para cursar una carrera se deben de aprobar de 40 a 50 materias (dependiendo de la carrera), algunas de las materias son comunes a varias carreras. El número de materias diferentes que se imparten son 300, las cuales están identificadas por una clave; para ellas se desea conocer su nombre, la cantidad de créditos que aportan, el total de inscritos en cada

materia por semestre y el número de profesores que la imparten.

Las materias son clasificadas para determinar el costo por cursar cada una de ellas, las materias más caras son las que requieren uso de laboratorio y prácticas profesionales y las más baratas son las materias introductorias de los primeros semestres.

5. El costo de cursar materias en un semestre esta determinado por el costo de las materias que se cursan más una cuota de inscripción semestral fijada antes de comenzar el proceso de inscripción.

6. Para impartir estas materias, la UHA cuenta con una planta docente de 150 profesores, cada uno de ellos identificados por su RFC, ademas se requiere conocer su nombre, su dirección, su teléfono y la cantidad de materias que ha impartido en cada semestre, asi como una lista de todas las que puede impartir, esto para poder tomar en cuenta profesores cuando alguno se da de baja. Cada profesor puede impartir desde 1 hasta 10 materias por semestre.

7. Cada inicio de semestre se entrega a los alumnos una Historia Académica en donde se presentan las materias que curso en cada semestre, la calificación que obtuvo, el profesor con quien tomo la materia, el número de semestres que lleva inscrito en la carrera (para determinar elementos fosilizados) y su porcentaje de avance en la carrera.

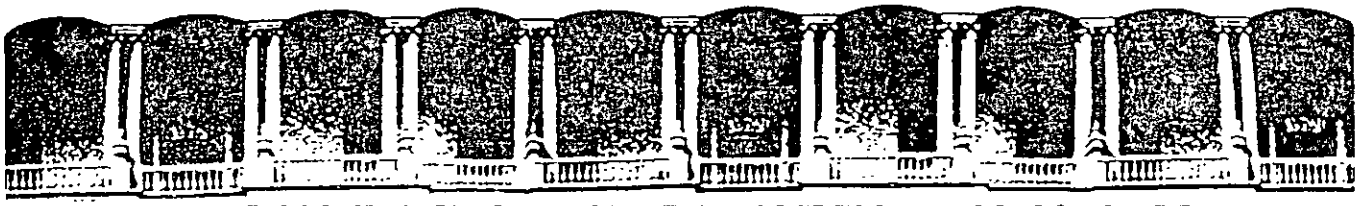
8. La UHA cuenta con 50 salones y 10 laboratorios identificados mediante una clave, en los cuales se imparten las materias, se pretende generar un horario para cada uno de los salones y laboratorios en donde se especifique las materias que se imparten en el, así como la hora y el profesor que las imparte, esto con el propósito de pegar dicho horario en la puerta del salón.

9. Para atender las llamadas telefónicas que se hacen a los alumnos, se necesita conocer, para cualquier alumno, el salón o laboratorio donde se puede localizar en horario de clases.

11. La oficina de Atención a alumnos proporciona servicios para emitir documentos como: constancias de inscripción, constancias de promedio y avances, constancias de terminación de estudios, etc., además proporciona servicios de cambios de calificaciones

12. La oficina de Exámenes profesionales necesita información para determinar las materias que ha cursado un alumno y el número de veces que las ha cursado, calificaciones, semestres en los que ha estado inscrito, etc.

13. En esta primera etapa no se requiere manejar la información de exámenes extraordinarios, coordinadores de carrera ni cursos intersemestrales.



FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

TALLER DE ANALISIS Y DISEÑO DE BASES DE DATOS RELACIONALES

MATERIAL DIDACTICO

A N E X O

DICIEMBRE 1996

Palacio de Minería Calle de Tacuba 5 Primer piso Deleg. Cuauhtémoc 06000 México, D.F. Tel.: 521-40-20 Apdo. Postal M-2285

TALLER DE ANALISIS Y DISEÑO DE BASES DE DATOS RELACIONALES

MATERIAL ANEXO

2 AL 13 DE DICIEMBRE DE 1996

Introducción al ambiente de Bases de Datos

OBJETIVO:

En este capítulo se estudiarán los conceptos básicos de la teoría de Bases de Datos.

En este capítulo el asistente:

- Definirá el término Base de Datos.

- Definirá el término Sistema Manejador de Bases de Datos (***Data Base Management System*, DBMS**).

- Conocerá las ventajas de desarrollar sistemas en ambientes de Bases de Datos.

- Identificará a los diferentes tipos de usuarios de una Base de Datos.

- Conocerá los diferentes modelos de Bases de Datos .

INTRODUCCION

En sus orígenes, el término **Base de Datos** sólo era utilizado dentro de los cerrados ambientes de los grandes centros de cómputo. Las Bases de Datos fueron una atractiva alternativa para el desarrollo de sistemas de información más versátiles y eficientes.

Actualmente, este concepto ha sido muy difundido gracias a la introducción de computadoras pequeñas y accesibles, y a la aparición de una gran cantidad de software de Bases de Datos, que permiten la distribución de los recursos en los Sistemas de Bases de Datos y el desarrollo de aplicaciones más eficientes y amigables, dando origen a esquemas Cliente/Servidor y de Bases de Datos distribuídas.

Sin embargo, no obstante que la tecnología de Bases de Datos ha avanzado considerablemente, la clave en el éxito de la adopción de esquemas de Bases de Datos, radica en el Diseño de la Base de Datos.

DISEÑO DE BASES DE DATOS

Diseñar una Base de Datos no consiste simplemente en mapear los antiguos sistemas de archivos a un esquema de tablas.

El diseñar una Base de Datos consiste en generar un modelo que represente los requerimientos de la empresa, que obtenga ventaja de todas las características del Modelo Relacional, así como de las facilidades del DBMS a utilizar, para brindar el mejor rendimiento posible a las aplicaciones generadas en la base de datos.

En este curso se presentará un metodología para el análisis y diseño lógico de la Base de Datos partiendo del análisis de requerimientos del sistema.

Por otra parte se presentarán una serie de técnicas para el Diseño Lógico y Físico que permiten la implementación de Bases de Datos de alto performance.

En los últimos temas del curso se tratarán temas realacionados con las nuevas Tecnologías de Bases de Datos, como lo son: Arquitectura Cliente/Servidor, Bases de Datos Distribuídas y Bases de Datos Orientadas a Objetos.

SISTEMAS DE INFORMACION DE PROCESAMIENTO DE ARCHIVOS

Antes de la aparición de las Bases de Datos, los datos de los sistemas de cómputo eran mantenidos en archivos.

En un Sistema de Información de Procesamiento de Archivos típico, cada departamento tenía sus propias aplicaciones y archivos, los cuales estaban particularmente diseñados para esas aplicaciones.

El departamento o área de trabajo en cuestión fijaba las políticas y estándares para el formato y mantenimiento de los archivos.

En toda la empresa existían una gran variedad de archivos muchos de los cuales mantenían información semejante.

Suponga el siguiente ejemplo:

Una empresa bancaria, mantiene su sistema de ahorros en un sistema de archivos. El sistema tiene diversos programas de aplicación que permiten al usuario manejar los archivos:

- Programa de cargos y abonos a una cuenta.
- Programa de altas, bajas y cambios de cuentas.
- Programa para generar estados de cuenta mensuales.
- Programa para obtener el saldo de una cuenta.

Estos programas de aplicación los han escrito programadores de sistemas en respuesta a las necesidades de la empresa.

Según surge la necesidad, se agregan nuevos programas de aplicación al sistema y probablemente nuevos archivos.

A los programas ya existentes se les hacen cambios "parches", para cumplir con las necesidades de la empresa. Por ejemplo, suponga que la directiva del banco ha decidido que el saldo mínimo de las cuentas de ahorro será de $500; por lo que es necesario modificar el programa de cargos y abonos y todos aquellos programas que hagan validaciones al saldo de la cuenta bancaria.

El típico *Sistema de Infromación de Procesamiento de Archivos* descrito está apoyado por un sistema operativo convencional. Los registros permanentes se almacenan en varios archivos, y se escribe un número de diferentes programas de aplicación para extraer y añadir registros.

Este tipo de sistemas tiene un número de desventajas importantes:

- Redundancia e inconsistencia de los datos
- Dificultad para tener acceso a los datos
- Aislamiento de los datos
- Anomalías del acceso concurrente
- Problemas de seguridad
- Problemas de integridad

BASE DE DATOS

Conjunto de datos interrelacionados con redundancia controlada para servir a una o más aplicaciones, los datos son independientes de los programas que los usan.

La Base de Datos representa la información de una empresa y su modelo debe de ser tal, que responda a todas las espectativas de información de la misma.

ARQUITECTURA DE LA BASE DE DATOS: ESQUEMAS

La arquitectura de una Base de Datos, es el modelo que describe los diferentes aspectos acerca de su estructura e incluye la estructura lógica y física.

En 1971 el grupo denominado Database Task Group y The Conference on Data Systems and Languages (CODASYL DBTG) generaron una primer propuesta para la estandarización de la arquitectura de los sistemas de bases de datos.

En 1975 The Standars Planning and Requirements Committee of the American National Standars Institute Committee on Computers and Information Processing publicó un documento semejante, ANSI/X3/SPARC.

Como resultado de estos reportes, se define la arquitectura de una base de datos en tres niveles de abstracción. A estos tres niveles se les conoce como **esquemas** o **modelos.**

Los esquemas determinan la estructura de la base de datos, no la información que se almacena en un momento dado.

El objetivo es que estos esquemas sean independientes. A la capacidad de modificar una definición de esquema en un nivel sin afectar la definición del esquema en el nivel inmediato superior se denomina **independencia de datos.**

Los esquemas así definidos son:

- **Esquema de Vistas o Esquema Externo**

El Esquema de Vistas define el modelo de la base de datos que percibe el usuario final. Este esquema esta formado por varios Esquemas Externos, los cuales representan el modelo que cada usuario percibe.

- **Esquema Lógico o Conceptual**

En este esquema se describen cuáles son los datos reales que están almacenados en la Base de Datos y que relaciones existen entre ellos. Un mismo modelo tendrá diferentes esquemas físicos al ser implantado en diferentes DBMS.

- **Esquema Físico**

En este esquema se define la implementación física de la base de datos, incluye las estructuras de datos y tipos de archivos utilizados para almacenar la información en los dispositivos de almacenamiento.

Las ventajas que se obtienen de una arquitectura como esta son las siguientes:

- Diferentes usuarios tienen diferentes vistas de la misma información.

- Los usuarios no deben preocuparse de las estructuras de datos complejas utilizadas para la implementación de la base de datos.

- Se puede cambiar la estructura del Modelo Conceptual sin afectar la del Modelo Externo que perciben los usuarios.

- El Modelo Conceptual no se ve afectado por cambios realizados a nivel físico.

MODELOS DE BASES DE DATOS

Estos modelo son utilizados para representar el Esquema Conceptual de la Base de Datos.

Estos modelos permiten especificar la estructura lógica de la base de datos y algunos aspectos de implementación de la misma.

Un Modelo de Bases de Datos es un conjunto de herramientas conceptuales que sirven para la descripción de los datos, relaciones entre ellos, semántica asociada y restricciones de consistencia.

Los diversos Modelos de Bases de Datos que se han propuesto se dividen dos grupos:

- Modelos lógicos basados en objetos
- Modelos lógicos basados en registros

Modelos Lógicos basados en Objetos

Los modelos lógicos basados en objetos se usan para describir datos en los niveles conceptual y de visión. Se caracterizan por el hecho de que proporcionan capacidad de estructuración bastante flexible, permiten especificar restricciones de datos explícitamente y son independientes de la forma en que los datos se almacenan y manipulan.

Algunos de los modelos más extensamente conocidos son:

- El modelo Orientado a Objetos.
- El modelo binario.
- El modelo semántico de datos.
- El modelo infológico.

EL MODELO ORIENTADO A OBJETOS

El Modelo Orientado a Objetos se basa en una colección de objetos, cada uno de los cuales representa un ente abstracto del Sistema de Información a modelar.

El objeto contiene información, **atributos,** que representan su estado. Por otra parte, los objetos tienen asociado código, **métodos**, que opera sobre el objeto.

Los objetos que tienen los mismos tipos de valores y los mismos métodos se agrupan en una **clase**. Una clase puede ser vista como una definición de tipo para objetos.

La única forma en la que un objeto puede accesar la información de otro objeto, es por medio de un método de ese otro objeto, es decir *enviando mensajes* al objeto. Los métodos constituyen la interfaz a un objeto y son el único medio de modificar la información interna del objeto, es decir, su **estado**.

Cada objeto tiene su propia identidad, independientemente de los valores de sus atributos. Así, dos objetos que contienen los mismos valores para sus atributos, son distintos. La distinción entre los objetos se mantiene en el nivel físico por medio de identificadores de objeto.

Modelos Lógicos basados en Registros

Los modelos lógicos basados en registros se utilizan para describir datos en los niveles conceptual y físico.

Los modelos lógicos basados en registros se llaman así, porque la Base de Datos está estructurada en registros de formato fijo de varios tipos. Cada registro define un número fijo de campos, o atributos, y cada campo normalmente es de longitud fija. Esto contrasta con los modelos orientados a objetos en los que los objetos pueden estar compuestos por objetos a un nivel de anidamiento de profundidad arbitraria.

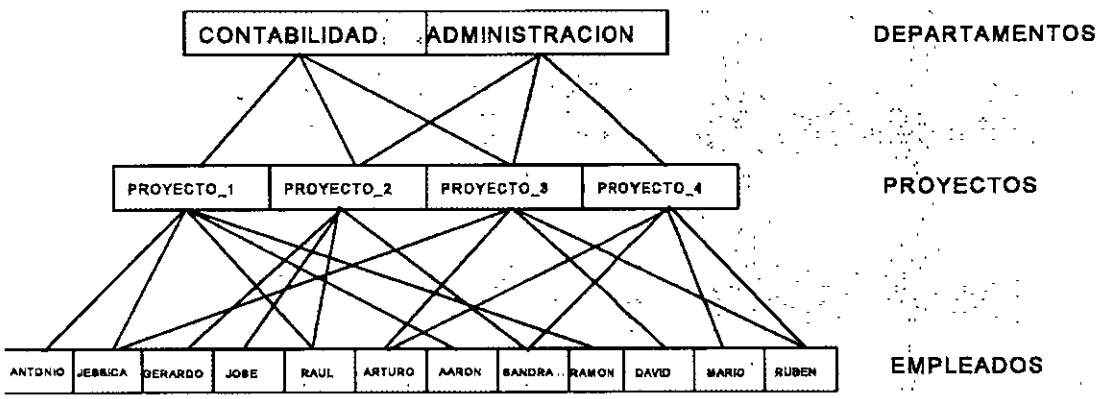
Los modelos lógicos basados en registros, no incluyen un mecanismo para la representación directa de código en la Base de Datos. En cambio, tienen asociados lenguajes que permiten expresar consultas y actualizaciones sobre la Base de Datos.

Dentro de los modelos lógicos basados en registros tenemos los siguientes:

- Modelo de red
- Modelo jerárquico
- Modelo relacional

MODELO DE RED

En este modelo, los datos se representan como una colección de registros, la relación entre ellos se da por medio de apuntadores. Los registros se organizan como una colección de gráficas arbitrarias.



Desventajas:

- Los apuntadores o direcciones se deben almacenar junto con los datos

- Para recuperar información se debe "navegar" a través de la gráfica

- No se puede obtener información no planeada antes de modelar

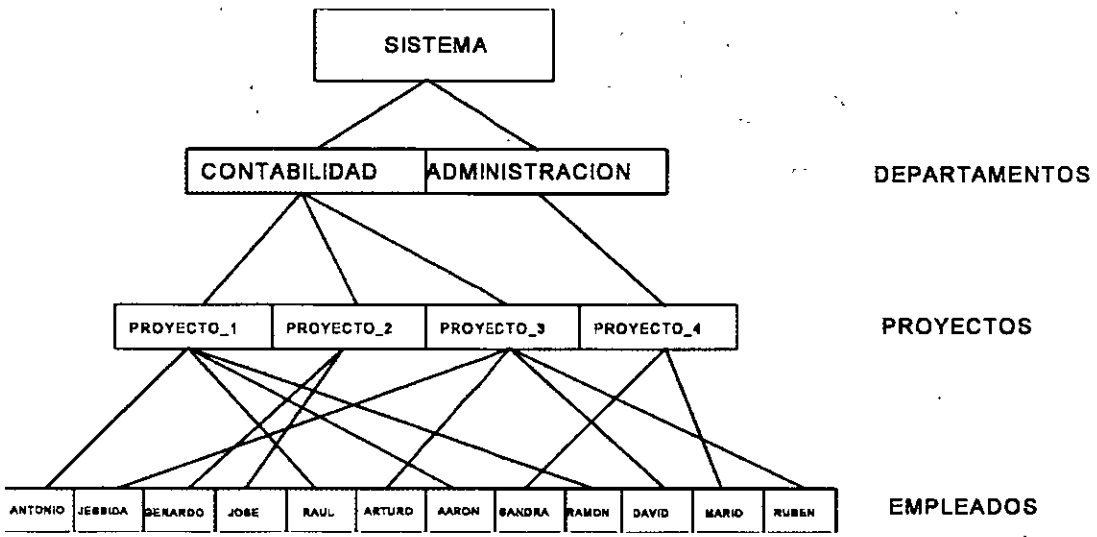
- Modificar la estructura de la Base de Datos implica redefinir todo el esquema

Ventajas:

- Acceso rápido a los datos debido a los apuntadores

MODELO JERARQUICO

En este modelo, los datos se representan como una colección de registros, mientras que la relación entre ellos se da por medio de ligas o apuntadores. Se diferencia del modelo de red, en que los registros están organizados como colecciones de árboles en vez de gráficas arbitrarias.



Desventajas:

- No puede haber ciclos y sólo puede haber asociaciones 1:N y 1:1
- Los apuntadores o direcciones se deben almacenar junto con los datos
- Para recuperar información se debe recorrer el árbol
- No se puede obtener información no planeada antes de modelar
- Modificar la estructura de la Base de datos implica redefinir todo el esquema
- Se pueden representar asociaciones M:N manteniendo datos duplicados

Ventajas:

- Acceso rápido a los datos debido a los apuntadores

MODELO RELACIONAL

En el modelo relacional, los datos y las relaciones entre los datos se representan por medio de una serie de tablas, cada una de las cuales esta compuesta por columnas con nombres únicos. Una columna de una tabla representa una relación entre un conjunto de valores. Existe una correspondencia entre el concepto de tabla y el concepto matemático de relación, del cual recibe su nombre el modelo relacional.

DEPARTAMENTOS

NO_DEPTO	NOMBRE
10	Sistemas
20	Finanzas
30	Contabilidad
40	Ingeniería

EMPLEADOS

NO_CTA	NOMBRE	DIR	TEL	NO_DEPTO
18456	Antonio	Revolución 123	733-22-89	10
18272	Jéssica	Norte 86B 98	657-28-92	40
72638	Gerardo	Río chico 22	512-38-39	10
28289	José	Palmas 926	833-32-21	30
29829	Raull	Rosas 83-5	937-00-52	30
87719	Arturo	Churubusco 45	723-45-11	30
32983	Aarón	Taxqueña 372	632-73-21	20
32732	Sandra	Av. Central 13	743-03-01	40
32903	Ramón	Marina Nal. 86	732-37-77	20
95672	David	Almaraz 2-1	664-83-00	10
48568	Edwin	Cozumel 11-3	731-82-83	40
84324	Jorge	Fco. Sosa 266	527-73-12	20

Ventajas:

- Tiene una base matemática, conocida como Algebra y Cálculo Relacional.

- Se pueden representar fácilmente asociaciones M:N.

- No existen apuntadores u otro tipo de información que no sea la que creó la necesidad de la Base de Datos.

- Las operaciones efectuadas para obtener información se realizan a nivel de la tabla completa y no a nivel de registros.

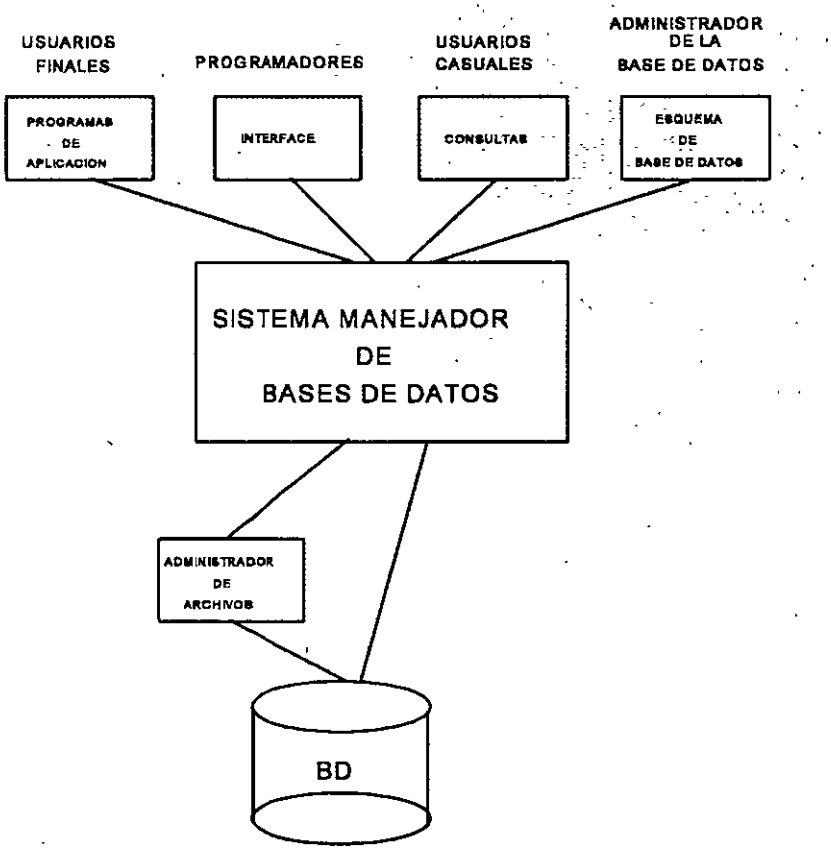
- No es necesario diseñar el esquema de la Base de Datos de acuerdo a las operaciones o consultas que se van a llevar a cabo. Es posible obtener información no prevista.

- Se puede modificar la estructura de la Base de Datos sin que esto obligue a un cambio de las aplicaciones.

- La forma de explotar la información es por medio de operaciones relacionales, a través de lenguajes de cuarta generación.

SISTEMA MANEJADOR DE BASES DE DATOS

Conjunto de programas que sirven para administrar, controlar, accesar y manipular una base de datos.



EJEMPLOS:

- SYBASE

- Oracle

- Informix

- Ingres

Generalmente, las base de datos requieren una gran cantidad de almacenamiento, las base de datos de las empresas se miden en términos de gigabytes o terabytes. Puesto que la memoria principal de las computadoras no puede almacenar esta información, la base de datos se almacena en disco. Es de gran importancia que el sistema de base de datos minímice la necesidad de mover los datos entre la memoria y el disco.

Las funciones que debe cumplir un DBMS son las siguientes:

- Simplificar y facilitar la definición y el acceso a los datos.
- Interactuar con el sistema de archivos.
- Permitir la implantación de esquemas de integridad.
- Permitir la implantación de esquemas de seguridad.
- Esquemas de respaldo y recuperación.
- Control de concurrencia

LENGUAJES

Una de las funciones de un DBMS es la de simplificar la definición y el acceso a los datos. Para definir el esquema de la Base de Datos, así como para explotar la información que se encuentra en ella es necesario contar con un medio de comunicación proporcionado por el DBMS, es decir, con un lenguaje. Existen dos tipos de lenguajes básicamente, los cuales en la mayoría de los casos están inmersos en uno:

- Lenguaje de Definición de Datos (**DDL, Data Definition Language**)

- Lenguaje de Manipulación de Datos (**DML, Data Manipulation Language**)

- Lenguaje de Control de Datos (**DCL, Data Control Language**)

El Lenguaje de Manipulación de Datos puede ser de alguno de los siguientes tipos:

- Procedural: se especifíca que datos se necesitan y cómo obtenerlos.

- No Procedural: se especifíca únicamente los datos que serán recuperados y no la forma de obtenerlos.

Los DML no procedurales normalmente son más sencillos de aprender y de usar que los procedurales. Sin embargo, puesto que el usuario no tiene que especificar como conseguir los datos, estos lenguajes pueden generar código que no sea tan eficiente como el producido por los lenguajes procedurales.

Un lenguaje será más productivo en tanto más No Procedural sea y más versátil para aplicaciones especiales en tanto más Procedural se comporte.

Una **consulta** es una sentencia de DML que solicita la recuperación de datos. Al subconjunto de instrucciones de DML que permiten realizar consultas se le llama **lenguaje de consulta.**

SQL (Structured Query Language)

SQL es un lenguaje tanto para la definición de los datos como para la manipulación de ellos que se ha convertido en estándar en el campo de los DBMSs.

Características:

- Es un lenguaje estándar reconocido por ANSI e ISO.

- Se encuentra implementado en la mayoría de los DBMS más populares.

- Es un 4GL.

- Es muy fácil de utilizar.

- No incluye referencias físicas de los datos.

- Es utilizado desde muchos programas de aplicación que forman parte de un DBMS.

- Es utilizado para la obtención, modificación y definición de los datos, así como para la Administración de la Base de Datos.

SISTEMAS DE BASES DE DATOS

Los DBMS nos ayudan a implementar sistemas de información bajo la perspectiva de Bases de Datos, proporcionándonos sistemas con las siguientes características:

- Eficiente control de los datos.
- Interacción con el sistema operativo para efectos del almacenamiento, recuperación y actualización de los datos en la base de datos.
- Implantación de la integridad.
- Seguridad.
- Esquemas de respaldo y recuperación.
- Control de concurrencia.
- Herramientas para la explotación de los datos.
- Fácilidad para explotar la Base de datos.
- Performance.
- Implantación de estándares.
- Rápido desarrollo de aplicaciones.
- Escalamiento.

Desventajas:

- Altos costos en la conversión o adaptación de sistemas ya existentes.
- Costos adicionales por hardware y software más sofisticado.
- Altos costos de los productos DBMS.
- Costos adicionales por la necesidad de contar con personal especializado.

USUARIOS DE LA BASE DE DATOS

Existe una gran variedad de aplicaciones que se pueden desarrollar y utilizar en una base de datos, por otra parte, existen una gran cantidad de usuarios que tienen necesidad de obtener información de la base de datos. Cada uno de estos usuarios tienen en general una concepción diferente de la estructura lógica y física de la base de datos.

El DBMS debe contar con las facilidades, herramientas e interfaces necesarias para poder satisfacer las necesidades de cada uno de los diferentes tipos de usuarios de la base de datos. Podemos definir los siguientes tipos de usuarios:

- Usuarios finales

- Usuarios casuales

- Programadores de aplicaciones

- Programadores especializados

- El Administrador de la Base de Datos

EL ADMINISTRADOR DE LA BASE DE DATOS

Una de las razones por las que se utilizan sistemas en Bases de Datos es el tener un control centralizado de la información que se maneja y de los programas que la accesan. El Administrador de la Base de Datos o DBA (*Data Base Administrator*) es un usuario especial que tiene como función controlar la Base de Datos y la forma en como es accesada.

Las funciones principales del DBA son:

- Definición del esquema lógico y de vistas de la Base de Datos.

- Definición de las estructuras de almacenamiento y de los métodos de acceso.

- Modificación del esquema y de la organización física.

- Autorización para acceso a los datos.

- Especificación de restricciones de integridad.

- Especificación de políticas para con la Base de Datos.

LABORATORIO

1. ¿Qué es una Base de Datos?

2. ¿Qué es un DBMS?

3. ¿Qué representa el Modelo Conceptual de una Base de Datos?

4. ¿Cuáles son las caractterísticas del Modelo Relacional?

5. ¿Cuál es el medio de comunicación mediante el cuál se define y explota una Base de Datos en un DBMS?

6. ¿Qué esun lenguaje de consulta?

7. ¿Es SQL un lenguaje de consulta?

Página intencionalmente blanca.

El Análisis y Diseño de Bases de Datos en el entorno de la Ingeniería de Software

OBJETIVO:

En este capítulo se determinará el papel que juega el Análisis y Diseño de Bases de Datos como disciplinas de la Ingeniería de Software.

En este capítulo el asistente:

- Definirá el concepto de Ingeniería de Software.

- Conocerá el ciclo de vida aplicado a la automatización de procesos.

- Conocerá el ciclo de vida aplicado al modelado de datos.

INGENIERIA DE SOFTWARE

La Ingeniería de Software consiste en el establecimiento y uso de principios y metodologías delingeniería robustos, orientados a obtener software de calidad que sea funcional y que se adapte a las necesidades actuales y futuras de la empresa.

La Ingeniería de Software abarca un conjunto de tres elementos claves: **métodos, herramientas y procedimientos,** que facilitan el controlar el proceso de desarrollo de software y suministran las bases para construir software de alta calidad de una forma productiva.

Los métodos de la Ingeniería de Software suministran el "cómo" construir técnicamente el software, abarcan tareas como:

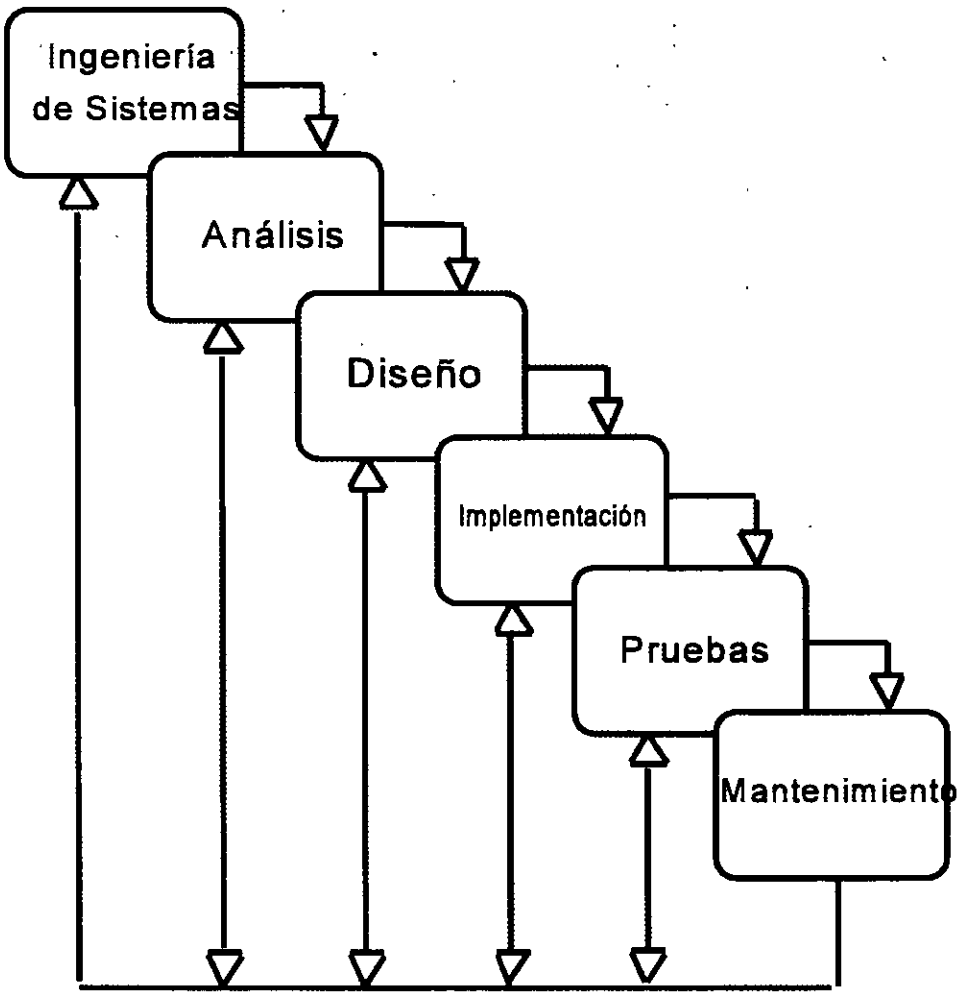
- Planificación y estimación de proyectos.
- Análisis de los requerimientos del sistema y del software.
- Diseño de los datos.
- Arquitectura de programas y procedimientos.
- Codificación.
- Pruebas.
- Mantenimiento.

Las herramientas de la Ingeniería de Software suministran un soporte automático o semiautomático para los métodos (por ejemplo herramientas CASE).

Los procedimientos de la Ingeniería de Software definen la secuencia en la que se aplican los métodos, la secuencia en la generación de documentos, los controles de calidad, y las guías que facilitan el establecer el desarrollo del software.

EL CICLO DE VIDA

El ciclo de vida en la Ingeniería de Software exige un enfoque sistemático, semi-secuencial del desarrollo del software que comienza en el nivel del sistema y progresa a través del análisis, diseño, codificación, pruebas y mantenimiento.



Las etapas de la Ingenierá de Software se aplican tanto a **Procesos**, como a **Datos**.

En el caso de procesos, el ciclo de vida nos conduce a la creación de programas que automatizan los procesos.

Para poder automatizar procesos, antes se debe analizar y diseñar los datos.

La aplicación de las etapas de la Ingeniería de Software a datos conduce a la creación de un Modelo de Datos, que puede ser, un esquema de archivos planos, o bien una Base de Datos. Este modelo será utilizado en el proceso de automatización de procesos.

No todos los sistemas basados en computadora hacen uso de una Base de Datos, para aquellos que lo hacen, la Base de Datos es la guía para todas las funciones del sistema.

El análisis, diseño e implementación de una Base de Datos forman parte del ciclo de vida de la Ingeniería de Software y son actividades que no dependen del desarrollo de alguna aplicación en particular y pueden iniciarse una vez que se ha definido el alcance del sistema de la información.

EL CICLO DE VIDA EN LA AUTOMATIZACION DE PROCESOS (DESARROLLO DE SOFTWARE)

Ingeniería y Análisis del Sistema

Debido a que el software es siempre un componente de un sistema mayor, el trabajo comienza estableciendo los requerimientos de todos los elementos del sistema y luego asignando algún subconjunto de estos requerimientos al software.

En esta etapa se fijan las limitantes del software, los alcances, sus funciones, sus interfaces (conexión con otros elementos del sistema global), los datos utilizados y los generados, etc.

Análisis

Para comprender la naturaleza de los programas a construir, el Ingeniero de Software debe comprender el dominio de los procesos a automatizar, las funciones, rendimientos e interfaces requeridas.

En esta etapa se análizan detalladamente cada uno de los procesos involucrados, también se documentan.

Este proceso se realiza en estrecha comunicación con el cliente.

La documentación generada en esta etapa, refleja, en forma detallada, los requerimientos del sistema.

Diseño

El diseño de software es un proceso que se enfoca a la generación de la arquitectura del software y detalle procedimental.

El proceso de diseño traduce los requerimientos en una representación del software que pueda ser establecida de forma que obtenga la calidad requerida antes de que comience la codificación.

Implementación

En el caso de procesos, esta etapa se conoce comunmente como codificación.

En esta etapa, el diseño debe traducirse en una forma legible para la máquina.

Si el diseño se ejecuta de una manera detallada, la codificación puede realizarse mecánicamente.

Pruebas

Las pruebas se enfocan sobre la lógica interna del software, asegurando que todas las sentencias sean probadas, y sobre las funciones externas, realizando pruebas para asegurar que la entrada definida producirá los resultados que se requieren.

Mantenimiento

El software sufrirá cambios después de que sea liberado.

Los cambios pueden ser debido a nuevas adaptaciones (Mantenimiento adaptativo) o a que se han encontrado errores (Mantenimiento correctivo).

EL CICLO DE VIDA EN LA GENERACION DE UN MODELO DE DATOS (DESARROLLO DE BASES DE DATOS)

Ingeniería y Análisis del Sistema

En esta etapa se determina la información que maneja el sistema, las características de esta información.

Se determina también el alcance del modelo que se desea generar y sus limitantes (la información que se deberá considerar).

En esta atapa se deben reflejar las reglas del negocio como requerimientos del modelo de datos. Hay que puntualizar que cada empresa tiene reglas y requerimientos diferentes.

Por ejemplo, si se desea desarrollar un modelo de datos para un banco, se debe determinar la información del cliente que se utiliza, los tipos de cuentas que maneja el banco, restricciones en cuanto a movimientos, reglas de integridad, etc.

Los requerimientos necesarios se obtienen en base a:

- Entrevistas con el cliente
- Cuestionarios
- Lectura de manuales de procedimientos, reglamentos, reportes, etc.
- Prototipos

Los requerimientos para el modelo de datos deben reflejar el sistema de información de la empresa, así como también futuros cambios.

El documento de requerimientos generado es la base y guía del análisis y diseño de la Base de Datos.

Análisis

Esta etapa es una etapa de retroalimentación con el cliente, en la que seguramente se proponen y cambian requerimientos.

En esta etapa se analizan los requerimientos y se obtiene una descripción del sistema de información a modelar, se genera el **Modelo Conceptual de Datos.**

El Modelo Conceptual de Datos refleja los datos, alcances, restricciones y reglas del sistema de información. Este modelo es independiente del Modelo de Datos a utilizar(jerárquico, red, relacional, etc.).

Diseño

El diseño de Bases de Datos incluye el Diseño Lógico de Bases de Datos y el Diseño Físico de la Base de Datos.

En el Diseño Lógico se hace un mapeo del Modelo Conceptual de Datos al Modelo Lógico de Bases de Datos a utilizar.

En el Diseño Físico se determina la forma en como se implementará la Base de Datos, en base al DBMS utilizado (indices, dispositivos, esquemas de protecciones, etc.).

Implementación

En esta etapa se genera la Base de Datos de acuerdo al Diseño Físico.

Mantenimiento y Administración

En esta etapa se generan cambios a la base de datos y se llevan a cabo tareas de administración que permitan recuperar la base de datos en caso de que sucedan fallas.

LABORATORIO

1. ¿Cuál es el objetivo de la Ingeniería de Software?

2. ¿Cuáles son las herramientas de la Ingeniería de Software?

3. ¿Las etapas en el ciclo de vida de la Ingeniería de Software deben ser estrictamente secuenciales, es decir, si se retrocede, no se esta haciendo un desarrollo adecuado?

4. La Ingeniería de Software se aplica a datos y a procesos, ¿ en qué secuencia se debe aplicar? ¿porqué?

5. ¿En que etapa del ciclo de vida se diseña la interface gráfica de una aplicación?

6. ¿El Diseño Lógico de la Base de Datos es particular a un DBMS?

7. Suponga que tiene que desarrollar el sistema de nómina de su empresa. Este sistema existe actualmente bajo un esquema de archivos en COBOL; sin embargo la funcionalidad del mismo ya no es la adecuada. El nuevo sistema será desarrollado utilizando tecnología de Bases de Datos, se utilizará SQL Server como DBMS y Visual Basic para el desarrollo de la aplicación.

Explique brevemente, ¿qué tareas debe llevar a cabo y en que secuencia?