VNIVERSIDAD NACIONAL
AVTNOMA DE
MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

A · C · A · T · L · Á · N

PASOS EN EL DISEÑO LÓGICO DE UNA BASE DE DATOS RELACIONAL (APLICANDO EL MODELO ENTIDAD RELACIÓN)

TESIS

Que para obtener el título de
Licenciado en Matemáticas Aplicadas y Computación
presenta :

Braulio González Alvarez

MÉXICO
1 · 9 · 9 · 6

TESIS CON
FALLA DE ORIGEN

Agradecimientos

Gracias a Dios por la salud que ha dado a mi familia.

A mis padres por dar el ejemplo de amor y respeto en la pareja, que me ha permitido tener una familia unida por siempre, y por todo el apoyo que he recibido siempre de ellos en cualquier tarea que he emprendido.

A Manolo por lo dedicado en toda tarea que emprende.

A Zory por todo el cariño que le tiene a mis padres.

A mi tío Moisés por mostrar siempre ese carácter de iniciativa y superación.

A la Universidad que me ha dado tanto y que me ha permitido crecer profesionalmente.

A mi Asesor y Sinodales por sus observaciones durante el desarrollo de este trabajo.

A Carmen Alcántara por la corrección de estilo en este trabajo.
A Aracely Ibarra por sus observaciones en la presentación de este trabajo.

A mis amigos por estar siempre a mi lado
Alfonso, Carlos, Charly, Carmen, Jesús, Rodrigo, Víctor, Lael, Leo, Héctor, Rosario, Sergio, Manolo, Vane.

PASOS EN EL DISEÑO LÓGICO

DE UNA BASE DE DATOS RELACIONAL

(APLICANDO EL MODELO ENTIDAD RELACIÓN)

ÍNDICE

Capítulo IV EJEMPLO

INTRODUCCIÓN

A lo largo de nuestra vida cotidiana nos encontramos con problemas en cuanto a obtención y manejo de información, también durante nuestra formación profesional, así como en el campo de trabajo.

Este trabajo trata sobre el estudio de un caso en particular, el cual presenta una aplicación al campo profesional; en específico al manejo de información de una empresa (restaurante).

El restaurante se analiza como una organización que iteractúa para satisfacer las necesidades de información que en él se presentan a través del **modelo entidad relación**.

Cabe mencionar que un restaurante se compone de diferentes áreas, tales como almacén, ventas, contabilidad, etc. , sin hacer de menos las demás que lo conforman. Y son precisamente estas áreas las que se tomaron para ejemplificar el **modelo-ER** y poder dar la información suficiente para la toma de decisiones dentro de la empresa.

Las áreas seleccionadas se analizan y desarrollan por separado. Cómo iteractúan en su propia área sería un trabajo muy extenso, por ello se obtiene **información concreta** de cada una de estas partes sin profundizar en el procesamiento de la misma, solamente en lo necesario.

Por **información concreta** se entenderá la información arrojada después de efectuarse procedimientos internos propios de cada una de las áreas de esta empresa.

Objetivo :

Mostrar la aplicación del **modelo entidad relación (modelo-ER)** en el diseño de una **base de datos (BD)**, ya que éste permite el diseño de la estructura lógica de un problema determinado. Así como ejemplificar los pasos en el ciclo de vida de una base de datos con la información obtenida de la empresa (restaurante), para poder tener la información confiable en la toma de decisiones dentro de la misma.

PRÓLOGO

Durante la investigación y recopilación de información para el desarrollo de este trabajo, la estructura del libro **Database Modeling and Desing : The fundamental principles , second edition 1994, del autor Toby J, Teorey;** me pareció excelente en la secuencia de su capitulado y en los temas que abarca; dicho libro es de lo más actual que se ha escrito de bases de datos y modelo entidad relación, debido a esto se tomó parte de su estructura para ejemplificar y desarrollar el trabajo, además de utilizar bibliografía de publicación reciente.

Este trabajo es una propuesta para mejorar la comprensión de las bases de datos y del modelo entidad relación, así como para abordar los problemas que se presentan a lo largo del ciclo de vida de una base de datos.

Convenciones

Durante el desarrollo del trabajo se utilizarán **abreviaturas de términos** que son manejados constantemente, que aparecerán sólo la primera vez que se mencione el término utilizado. Le seguirá su abreviatura correspondiente con letra mayúscula obscura entre paréntesis, con la que se conocerá a ese término durante el trabajo. Las siguientes veces que se haga referencia a este término sólo se colocará su abreviatura; por ejemplo, el término base de datos será de uso continuo, para tal caso, la primera vez que se mencione se colocará **base de datos (BD),** las siguientes veces que se haga referencia a él sólo se colocará su abreviatura correspondiente **BD**. Esto se presenta en el **apéndice a).**

Habrá ocasiones en que se haga referencia más de una vez a una abreviatura, incluyendo su término correspondiente, esto se hace para que no se pierda la idea que representa ésta.

También se debe considerar que la implementación y manipulación de la **BD** será realizada en el sistema manejador de bases de datos **SQL-server Sybase**. Durante el desarrollo de este trabajo este manejador será conocido simplemente como **Sybase**.

Por otra parte, debido al gran número de pasos ejemplificados y desarrollados, se decidió poner pequeños ejemplos que mejor se presten para la comprensión de este trabajo, desarrollados en cada capítulo, dejando para los apéndices el complemento de modelos, descripciones, figuras, consultas y tablas que se omitieron en cada capítulo, tales son los siguientes casos : **la integración del diseño lógico de la base de datos, la descripción de la base de datos, resumen de tablas en la base de datos, la transformación del modelo entidad relación en tablas, los requerimientos de análisis y una introducción a SQL-server Sybase.**

Al inicio de cada capítulo se dará una introducción del mismo, así como recordar qué etapa del diseño se desarrolla y para qué servirá dentro de este trabajo.

Capítulo I
Se presentará una idea general de lo que son las bases de datos, es decir, un aspecto teórico de conceptos, definiciones, herramientas (Software y Hardware), los usuarios que manipulan la **BD** , los beneficios de utilizar una **BD** , la abstracción de datos para los usuarios, instancias y esquemas; la independencia y dependencia de los datos y ciclo de vida de una **BD** , así como la importancia de utilizar una **BD.**

Capítulo II
Este capítulo trata sobre la teoría existente, acerca del **modelo-ER**, el cual está dividido en tres subtemas :
II.A trata de los principios y fundamentos de construcción del **modelo-ER**, tales como sus características, definiciones y ventajas, que permitirán la definición de una **BD** relacional, la cual se utilizará más adelante.
II.B se muestra el papel que juega el **modelo-ER**, en el diseño lógico de una base de datos, y se ejemplifica la teoría descrita en esta parte del trabajo.
II.C se presenta la transformación del **modelo-ER** en esquemas de tablas, que más adelante, cuando se normalice la **BD**, serán transformados a tablas, utilizando **Sybase**.

Capítulo III
Trata sobre la unificación de los esquemas de tablas, esto se logra por medio de la **normalización** de la **BD**.
Aquí se describen las 5 formas normales para poder tener información confiable, libre de redundancia e inconsistencia, es decir, una **BD** normalizada.

Capítulo IV
En este capítulo una vez que la **BD** tiene una forma normal, la cual se consiguió en el capítulo III, se muestra la transformación de la forma normal de la **BD** en tablas en el **apéndice e),** además se presentan varios ejemplos en los que se aplica la teoría descrita en los anteriores capítulos, en los que se hacen consultas a la **BD**, utilizando **Sybase ,** por medio de las cuales se solucionarán los requerimientos de información planteados por los usuarios en el análisis.

Al final de estos capítulos se darán las conclusiones y experiencias vividas durante el desarrollo de este trabajo.

Como parte importante y como un suplemento, se muestran varios apéndices, que como su nombre lo indica son parte que añade y subsana omisiones durante el desarrollo del mismo, estos apéndices se presentan al final de los capítulos y son una herramienta de referencia importante para la mejor comprensión de las bases de datos y abreviaturas utilizadas alrededor de este tema; a continuación se muestra el resumen de éstos.

Apéndices

Apéndice a) Abreviaturas y Definiciones
Este apéndice muestra abreviaturas de palabras que se utilizan continuamente, además de algunas definiciones a las cuales se hará referencia una o dos veces durante el desarrollo del mismo.

Apéndice b) Integración del diseño lógico de la base de datos
Este apéndice es parte del capítulo II, donde se muestran las herramientas necesarias para construir el diseño lógico de la base de datos, además de explicar en detalle los pasos por los que atraviesa este diseño lógico. Por la manera como esta definido este trabajo es mejor tomar un ejemplo que se preste para la total comprensión del capítulo II, de tal forma que se muestre la integración al problema planteado en este apéndice, omitiendo detalles de cómo se realizó.

Apéndice c) Descripción de la base de datos
Descripción detallada de tablas, atributos y llaves.

Apéndice d) Resumen de tablas de la base de datos
Contiene un resumen de tablas que serán útiles como referencia rápida, cuando no se recuerde la estructura de la base de datos.

Apéndice e) Transformación del modelo-ER en tablas
Muestra la transformación de esquemas de tablas ya normalizadas, con sus correspondientes atributos identificadores y descriptores, a tablas que realmente contendrá la base de datos, utilizando **Sybase**.

Apéndice f) Requerimientos de información
Contiene el total de requerimientos de información para cada uno de los usuarios en el área de alimentos y bebidas.

Apéndice g) Introducción a SQL-server Sybase
Este apéndice muestra una visión de lo que es un lenguaje estructurado de consulta, el cual servirá para una mejor comprensión del capítulo IV.

También se debe considerar el software empleado para el desarrollo de este trabajo :

Para capturar de este trabajo se utilizó el procesador de texto **Word V6.0** para windows.
Copyright 1983 - 1993 Microsoft Corporation.
Durante los **capítulos I, II, III, IV**, así como sus apéndices.

Para elaborar las **figuras que aparecen durante los capítulos** se utilizó el procesador **Visio V6.0** para windows.
Copyright 1991 - 1992 Shapeware Corporation.

Para elaborar las tablas que aparecen en el **capítulo III**, se utilizó la hoja de cálculo **Excel V5.0**
Copyright 1985 - 1994 Microsoft Corporation.

Para elaborar las portadas de cada capítulo se utilizó **Powerpoint V3.0** .
Copyright 1987 - 1993 Microsoft Corporation.

Capítulo I

Bases de Datos

Fundamentos

CAPÍTULO I
BASES DE DATOS

Las base de datos y un **sistema manejador de bases de datos** son dependientes el uno del otro, ya que una base de datos no puede iteractuar sin el soporte de un sistema manejador de bases de datos. De la misma manera, un sistema manejador de bases de datos no podría manipular una base de datos si no existiera la base de datos o un análisis y diseño de la misma para su implementación. A continuación se mencionan los **beneficios de utilizar bases de datos**, ya que durante el desarrollo de este trabajo se realizará un análisis del problema abordado, de tal forma que al transformar el modelo entidad relación en tablas en el capítulo IV, se tengan todos los beneficios que se describen en este capítulo. También se verá la **abstracción de datos** para los usuarios, ya que alrededor de los niveles de abstracción gira el almacenamiento, creación y manipulación de datos, que ayudarán a solucionar requerimientos de información en un momento determinado **(instancia)**. Además se mostrará **la independencia y dependencia de los datos**, las cuales son de suma importancia, ya que si se modifica uno de los niveles de abstracción de los datos no se debe de alterar el **esquema** general de la base de datos o las aplicaciones que las manipulen. Se mencionará la **importancia de utilizar una dase de datos** y se mostrarán los pasos en el **ciclo de vida de una base de datos**.
La importancia de este capítulo es el hecho de que durante el capítulo II se realizará el diseño lógico de una base de datos con todas las bondades que se mencionaron aquí. Una vez teniendo éste, se normalizará en el capítulo III, para su posterior transformación en el capítulo IV, utilizando un sistema manejador de bases de datos, el cual permitirá la implementación, manipulación y posibles modificaciones a la base de datos para que ésta mejore su operación conforme pasa el tiempo.

I.1 Bases de datos

Para iniciar se dará una visión general de las **bases de datos** , las cuales son una de las áreas de las ciencias de computación de más rápido desarrollo, que tuvo sus inicios en 1960. Hoy en día, la cantidad de información que se almacena en una base de datos se mide, sin exagerar, en cientos de megabytes. Muchas organizaciones dependen de la manipulación eficiente de dicha información, que se logra utilizando un **Sistema Manejador de Bases de Datos (SMBD)**, el cual facilita la definición de una base de datos y la manipulación de la misma por programas (Software).

Una **base de datos (BD)** es una colección de datos almacenados interrelacionados, que sirven para satisfacer las necesidades de múltiples usuarios dentro de una organización, (estos datos están relacionados con cualquier cosa que sea significativa para los mismos); para el desarrollo de este trabajo se entenderá como **base de datos** la descripción de una **base de datos relacional**, que es aquella cuyos datos y relaciones entre los mismos se realizan mediante una colección de tablas, cada una de las cuales tiene un nombre y número de columnas con nombres únicos, además de que las relaciones entre éstas se tienen por medio de llaves.

La preferencia de utilizar una **BD**, en lugar de archivos aislados, es el hecho de conservar la **integración** de datos, facilitar el acceso y las transacciones , además de disminuir la **redundancia** e **inconsistencia** ; de estas preferencias se hablará más adelante en este capítulo.

Ahora se mostrará una idea de lo que es un **SMBD,** el cual, como se mencionó anteriormente, es una herramienta muy importante para una **BD** , para esto se definirán las partes que lo forman, las cuales son de suma importancia, debido a que son dependientes unas de otras para que el **SMBD** iteractúe correctamente con la **BD**.

1.2 Componentes de un SMBD

Un **SMBD** tiene los siguientes componentes :
Datos, Hardware, Software y Usuarios

a) Datos

El primer componente son los **datos** , que se convierten en información cuando son utilizados. Estos son almacenados en un sistema y se dividen en una o más **BD**. Pueden ser de diferentes tipos, por ejemplo : caracter, numérico, alfanumérico, fecha etc.
Para este trabajo se definen dos tipos de datos :
Datos de entrada, se refieren a los datos que entran al sistema, por lo regular desde una terminal. Estos a su vez son manipulados dependiendo de las necesidades que se tengan.
Datos de salida, son mensajes o informes que genera el sistema, ya sean impresos o desplegados en una terminal.

b) Hardware (HW)

Este es otro componente necesario para que iteractúe el **SMBD,** el cual está formado por las unidades de almacenamiento secundario, tales como unidad de discos donde se encuentra almacenada la **BD,** además de teclado, monitor, mouse, disco duro, CPU etc.
Se supone que la **BD** es demasiado grande para caber en memoria principal del computador. Por lo regular se encuentra grabada en disco duro. Es muy importante definir el **HW** dependiendo de las necesidades que se tengan al momento de implementar el sistema.

Para la mejor comprensión de esta parte del trabajo es necesario tener presente la **figura 1.1** ya que alrededor de ésta giran los conceptos referentes al **SMBD**. Para la mejor comprensión del **SW** se mostrarán los siguientes componentes : estructura de un **SMBD**, base de datos, diccionario de datos y funcionamiento del **SMBD**.

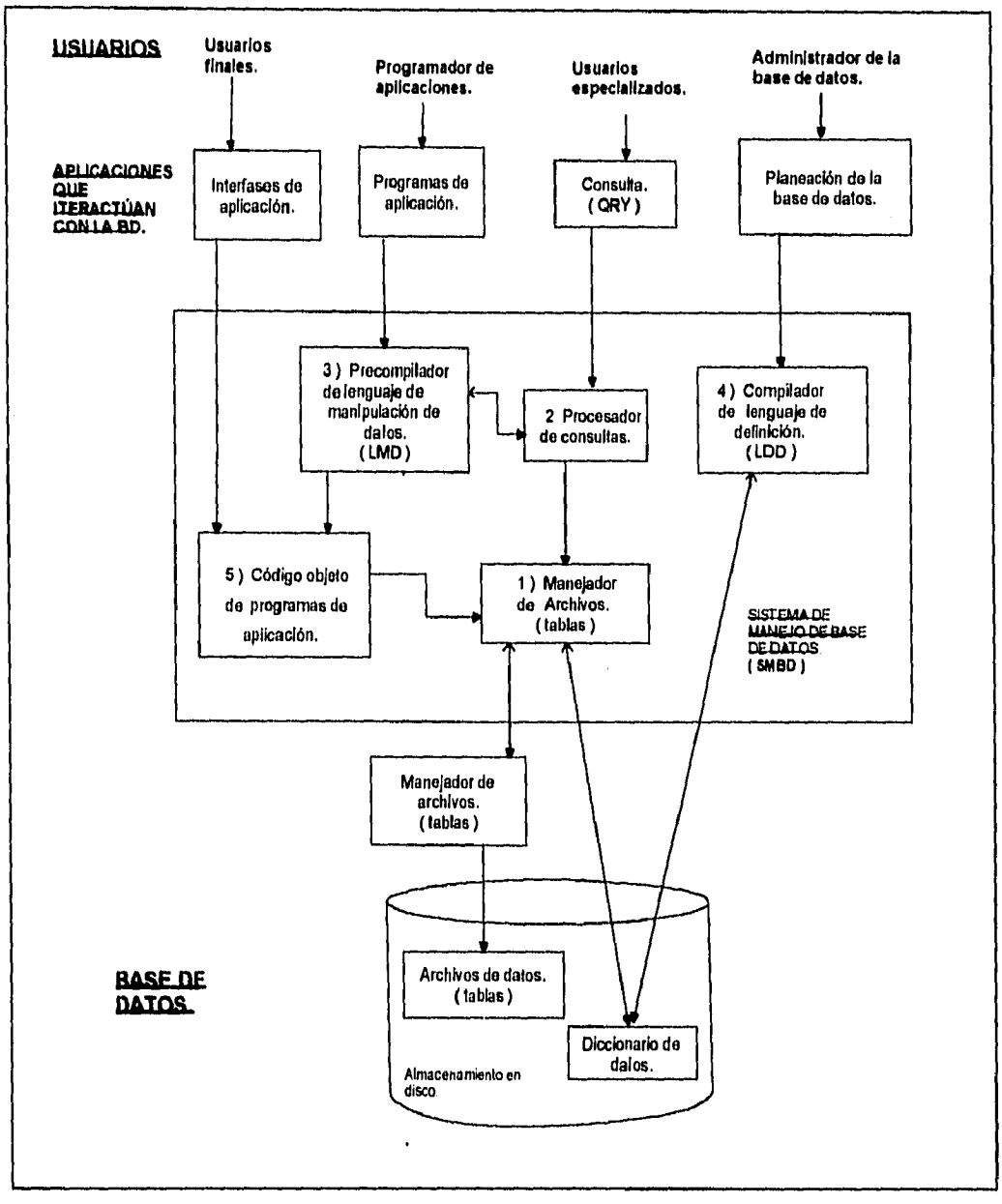


Figura 1.1 Estructura del SMBD.

[1]**figura 1.1**

c) Software (SW)

Aquí es donde entra el **SMBD,** entre el almacenamiento de la **BD** físicamente y los usuarios del sistema . (Ver figura 1.1)

La figura 1.1 se tomó del libro [Henry F. Korth, 21]

c.1) Estructura de un SMBD

En la descripción del **SW** se muestra la **estructura de un SMBD ,** la cual tiene un **manejador de BD.** Se define como la interfaz de los datos almacenados en disco y las aplicaciones que los manipulan, Está formado por cinco componentes :

El primero es el **manejador de archivos o tablas,** el cual manipula el espacio de memoria en discos.

El segundo se conoce como **procesador de consultas.** Compila instrucciones en un **lenguaje de manipulación de datos (LMD) ,** a instrucciones en bajo nivel para que pueda realizar su actividad en la **BD.**

El tercero es el **precompilador del lenguaje de manipulación de datos** , se encarga de compilar instrucciones en **LMD** para convertirlas en código apropiado que iteractúe con procedimientos del lenguaje principal.

Aquí se verá que el **Lenguaje de manipulación de datos o lenguaje de consultas (LMD),** tiene las siguientes funciones :

Recuperación de información almacenada **(Consultas),** inserción **(altas),** supresión **(bajas),** modificación **(Cambios).**

Además de permitir la manipulación de datos de manera eficiente (esto si se definieron algoritmos que permitan realizar operaciones óptimas con los datos), en este trabajo se utilizará el **lenguaje estructurado de consulta (SQL),** el cual está contenido dentro de **Sybase** que se utilizará para ejemplificar este trabajo y que permite la manipulación de datos, especificándole cómo obtener los datos que se necesiten.

El cuarto es el **compilador del lenguaje de definición de datos,** se encarga de compilar instrucciones en **lenguaje de definición de datos (LDD)** en **metadatos** , es decir, datos de los datos .

El esquema de definición de datos se realiza por medio de un **LDD,** que al compilar instrucciones, así como detalles de la implementación de los esquemas, obtiene tablas. Estas se almacenan en un **diccionario de datos**, es decir, un **directorio,** el cual es un archivo que contiene **metadatos.**

El quinto componente de la estructura del **SMBD**, es el **código objeto de programas de aplicación,** el cual es código ya compilado y listo para ejecutarse.

c.2) Base de datos

Por otra parte, como segundo componente del **SW** se tiene la **base de datos,** la cual tiene dos componentes :

Los **Archivos de datos o tablas,** los cuales contienen la **BD** y el **diccionario de datos**, que contiene metadatos.

c.3) El Diccionario de datos

Es el tercer componente del **SW**, el cual es uno de los recursos más importantes del **Administrador de la base de datos (ABD) ,** este diccionario es una **BD** que contiene datos acerca de los datos, es decir descripciones de todos los objetos del sistema. El diccionario de

datos puede integrarse incluso en la **BD** que describe y, por tanto, incluir su propia descripción.

c.4) Funciones del manejador de BD
Aquí se muestran las **funciones del manejador de BD,** que es el cuarto componente del **SW** y realiza las siguientes tareas :

La primera función es **iteractuar con el manejador de archivos.** Este compila sentencias en **LMD** a comandos de bajo nivel que manipularán la **BD,** es decir , realiza la manipulación de información.

La segunda es **implementar integridad,** de tal manera que los datos que se almacenen deberan de satisfacer **restricciones de consistencia,** esto se puede lograr por medio de una adecuada definición de **Reglas del negocio**, es decir, políticas de seguridad y restricciones que deberá tener la **BD**.

Otra función es **implementar la seguridad,** aquí se verifica que se cumplan las restricciones de seguridad definidas con anterioridad.

También se deben **verificar copias de seguridad y de respaldo,** ya que un sistema puede fallar como cualquier automóvil, un tren o cualquier medio mecánico o electrónico, por esto debe ser capaz de recuperar y hacer copias de seguridad, para dejarlo como estaba antes de que ocurriera la falla.
Estas fallas pueden ocurrir por defectos fisicos de los dispositivos de almacenamiento, cortes de energía eléctrica, por problemas de comunicación o por las aplicaciones que manipulan los datos. Nota 1)

La quinta función, como una tarea de vital importancia, debe procurar que el manejador sea capaz de **controlar la concurrencia.** Cuando se manipula la **BD** concurrentemente, es el responsable de conservar la consistencia de los datos. Este control se puede tener con una adecuada definición de **reglas del negocio**.

Hasta este momento se han descrito los **datos, Hardware y Software** necesarios para el funcionamiento de un **SMBD**, ahora se definirán los usuarios que trabajan con la **BD**.

d) Usuarios de la BD
Como último componente de un **SMBD** se tiene a los **usuarios de la BD,** éstos se clasifican en cuatro diferentes tipos :
El primero son los **usuarios finales,** es decir los usuarios que solamente operan el sistema sin tomar en cuenta cómo iteractúa internamente la **BD**.

Nota 1)

En un manejador pequeño estas tareas se deja que los usuarios las realicen, en **Sybase** se pueden crear aplicaciones que realicen estas tareas en un lapso de tiempo determinado, sin la intervención de los usuarios finales.

El segundo tipo es el **programador de aplicaciones,** el cual se encarga de crear **aplicaciones** para manipular la **BD.** Estas aplicaciones se crean por medio de un **LMD** que son precompilados, una vez compilados son llamados a procedimientos en lenguaje principal y se crea un código objeto, como se puede ver en la siguiente figura 1.2

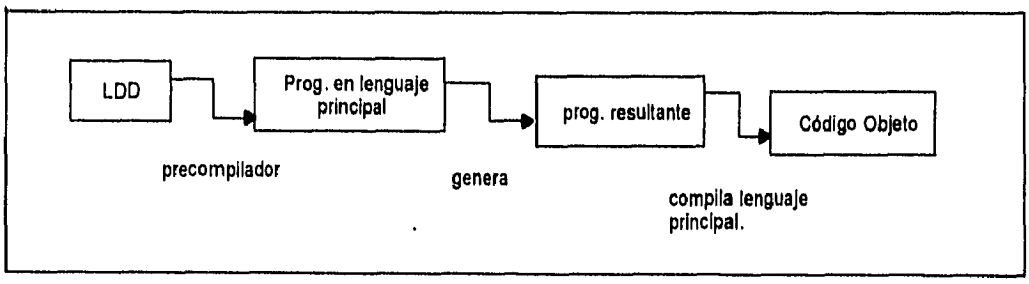


Figura 1.2 Compilación de aplicaciones.

EL tercer tipo son los **usuarios especializados,** se encargan de accesar a la **BD** por medio de un **LMD** , puede ser utilizando un **SQL,** en el cual se especifican sus requerimientos de información.

El cuarto tipo de usuario es el **administrador de la base de datos (ABD),** realiza las siguientes tareas :
Tiene el control centralizado y de todas las aplicaciones que accesan a la **BD**.
Define esquema, identificando entidades, atributos y definiendo vistas.
Realiza el análisis y requerimientos de información para ejecutarlos por medio de un **LDD** y dar origen a tablas con sus respectivas características, esto se verá representado en el nivel conceptual .
Define estructuras de almacenamiento y del método de acceso, creando aplicaciones con estas estructuras para ser compiladas por el **LDD y LMD.**
Modifica el esquema y organización física por medio de un LMD o LDD .
Por ejemplo, añadiendo chequeos a las tablas.
Da autorización para chequeo de tablas, dando jerarquías para acceso a la BD.
Especifica restricciones de integridad.
Se relaciona con los usuarios verificando que los datos que se requieren sean los que estén disponibles.
Define estrategias de respaldo y recuperación de datos.
Define procedimientos de validación, tales como actualizar o borrar un dato y validar la transacción.
Controla el desempeño y responde a los cambios de los requerimientos, es decir, se encarga de realizar los ajustes a medida que los requerimientos cambian.
Por ejemplo, programación de aplicaciones, creando rutinas de reorganización, es decir reordenar la **BD** para recuperar el espacio ocupado por datos obsoletos, creando rutinas de registro de eventos diarios, rutinas de recuperación y rutinas de análisis estadístico.

I.3 Beneficios de utilizar una base de datos.

Teniendo en mente la descripción de **BD** y los componentes de un **SMBD**, se describirán los beneficios que proporciona el utilizar una **BD**, para esto cabría hacer la siguiente pregunta :
¿ Por qué es necesario utilizar una **BD** al momento de desarrollar un sistema ?
Se utiliza una **BD** por lo siguiente :
Proporciona a la empresa un **control centralizado** de los datos, es decir, que sea **integrada y compartida** y capaz de controlar el **acceso concurrente**, de tal manera que al realizar un análisis y diseño lógico de una **BD**, se deben tomar en cuenta los siguientes puntos :

1.- Evitar la redundancia
Es el almacenamiento duplicado de registros en diferentes tablas, de los mismos datos, esto implica más espacio físico de almacenamiento y más tiempo en la manipulación de datos.
Por ejemplo :
El atributo **em_nombre** en tabla **empleados** y **ve_em_nombre** en tabla **ventas**, este problema se soluciona declarando identificadores que relacionen las entidades para tener acceso a los atributos descriptivos, que darán la información necesaria asociada a este identificador, por ejemplo :
cl_id en tabla **clientes** y **cl_id** en tabla **ventas**.
Los atributos descriptivos e identificadores mencionados anteriormente, se describirán más adelante en el capítulo II.

2.- Evitar inconsistencia
Esta se presenta cuando se tiene información duplicada y los datos que deben ser iguales son diferentes, es decir, los datos no concuerdan entre sí, por ejemplo, cuando dos datos se presentan por dos entradas distintas, al actualizarse un registro no se actualizan los demás, que se encuentran duplicados en las diferentes tablas y por esto no concuerdan, esto se evita cuando la información se representa por un solo atributo.
Por ejemplo :
Cuando se actualice **pr_tentrega** en tabla **proveedores**, no se actualiza **co_pr_tentrega** en tabla **compras**.

3.- Tener facilidad para accesar los datos
Se tiene desarrollando un sistema de recuperación de datos en general, esto se define en el análisis y es llevado a la práctica, codificando algoritmos adecuados y ejecutándolos con un **SQL** y así optimiza la manipulación de datos .

4.- Evitar aislamiento de los datos o independencia de los mismos
Esto es cuando los datos están en diferentes formatos y cuando se quiera un cambio, no afecta un esquema diferente , es decir, al añadir un tipo de cliente al sistema y modificar la estructura de las tablas (el nivel conceptual), no se tienen que reescribir las aplicaciones.

5.- Evitar anomalías del acceso concurrente

Para lograr esto se requiere que la **BD** sea **compartida,** es decir, que partes individuales de la **BD** se puedan compartir por diferentes usuarios, en el caso de que éstos utilicen concurrentemente una parte de la **BD** con diferentes necesidades de manipulación, por ejemplo :

Cuando varios usuarios actualizan datos de un cliente, al mismo tiempo se pueden provocar inconsistencias en los mismos si no se prevee alguna forma de solución.

Por ejemplo : Para el acceso concurrente (una regla del negocio), se bloquea el registro de un determinado cliente hasta que se realice la actualización. Esta regla del negocio asegura que la información que se consulte sea la misma en cualquier parte de la **BD,** ya que no se permiten actualizaciones cuando un usuario se posiciona en un registro para realizar cambios, siempre y cuando un usuario no se haya posesionado anteriormente.

6.- Tener seguridad en el acceso a la información

Se debe dar permiso a usuarios de manipular información, dependiendo del puesto que tengan dentro de una empresa (no todos los usuarios pueden accesar a toda la información), para esto se deben crear políticas de seguridad, es decir, que el acceso sea sólo por canales establecidos.

7.- Tener integridad de los datos

Es decir, tener las **tablas unificadas** donde se elimine parcial o totalmente la **redundancia** entre los mismos.

8.- Definir las reglas del negocio (políticas a cumplir)

Se deben definir las reglas del negocio, éstas son políticas y restricciones en datos que se tomaron al momento de analizar y desarrollar la **BD** para la manipulación de la misma. Estas se definen después de haber realizado un análisis de todas las partes a iteractuar en la **BD**.

Para el acceso concurrente se pueden tener rutinas de chequeo al momento de actualizar un registro.

Por ejemplo :

Se pueden consultar muchos registros por diferentes usuarios, pero cuando se actualiza uno, se bloquea la tabla automáticamente en una página, donde se encuentra el registro a modificar, hasta que se libere nuevamente.

Otro podría ser en el caso de modificar una promoción para un determinado día, bloquea el registro perteneciente a la misma , pero que además sólo se pueda modificar una vez al día.

Así se pueden tener diferentes reglas del negocio que se definieron para conservar la consistencia de la **BD**.

Los tres niveles de abstracción en los cuales se pueden ver representados los datos son de suma importancia, ya que alrededor de estos niveles gira el almacenamiento, creación y manipulación de los datos, que ayudarán a solucionar los requerimientos de información planteados en el análisis y que serán resueltos con la ayuda el **modelo-ER.**

El objetivo de esta parte del trabajo es proporcionar una visión abstracta de los datos, puesto que muchos usuarios no tienen experiencia en computadores, se les disminuye la complejidad a través de diversos niveles de abstracción, para simplificar su iteracción con el sistema.

Estos niveles de abstracción, transparentes para un usuario, se dan cuando el sistema no muestra cómo iteractúa con los datos, debido a la complejidad de las operaciones que realiza. Se compone de los siguientes niveles :

I.4 Abstracción de los datos

El primero es el **nivel físico,** describe cómo se almacenan los datos físicamente en el disco, como posiciones de memoria consecutivas o bits. Estos bien pueden ser llamados campos de almacenamiento, que es la unidad más pequeña en que se puede almacenar información.

EL segundo se conoce como **nivel conceptual,** es usado por el **ABD**, quien decide qué información se va a almacenar en la **BD**, es decir, describe los datos que son almacenados, pero no describe cómo, además las definiciones y las relaciones que hay entre ellos.
Por ejemplo, el tipo de los datos al definir la siguiente TABLA que forma parte de la aplicación que se presentará más adelante y las relaciones que hay entre ellos.

CREATE TABLE **clientes**
(cl_id char(5);
em_id char(5);
cl_nombre varchar(40);
cl_direccion varchar(40);
cl_ciudad varchar(30);
cl_telefono varchar(30);
cl_antiguedad char(2);
cl_frecuencia char(2);
cl_profesion varchar(30);
cl_rcf char(13))

La interpretación del ejemplo anterior se define como una tabla con el nombre de clientes, que contendrá los siguientes atributos, por ejemplo, **cl_ nombre** con un dominio cadena tipo caracter de 40 posiciones y con un contra dominio variable, dependiendo de lo que se capture, así para los demás atributos.
La información definida en la tabla se puede almacenar en un registro de campos almacenados y asociados, llamados registros de almacenamiento, la creación de tablas se define con más detalle en el **apéndice e),** donde se crean las tablas que contendrá la **BD**.
El tercer nivel es el conocido como **visual o vista externa,** en este nivel sólo se visualiza parte de la **BD** a la vez , dependiendo de las necesidades que se tengan y los permisos de manipulación de información, de tal manera que es el contenido de una **BD** tal como lo ve un cierto tipo de usuario, y se compone de múltiples ocurrencias . Los niveles de abstracción de los datos se pueden ver gráficamente en las **figura 1.3 y 1.4**

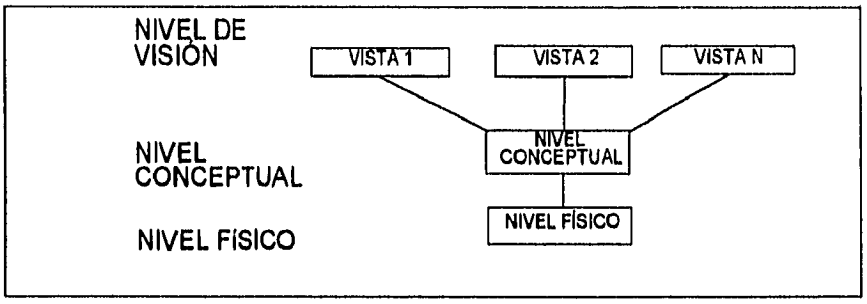


Figura 1.3 Niveles de abstracción de los datos.

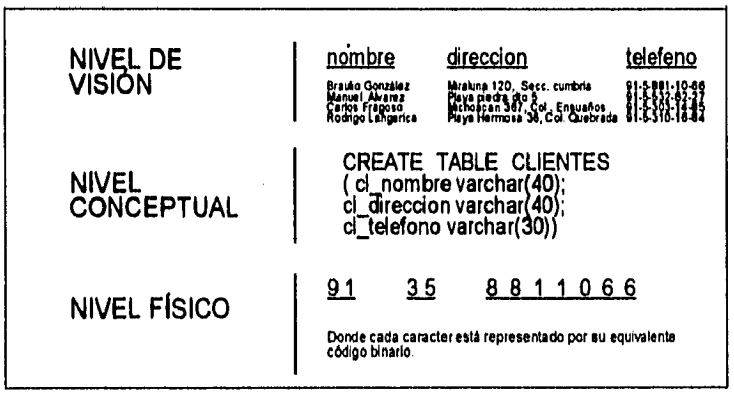


Figura 1.4 Niveles de abstracción de los datos.

Nótese que en las figura 1.3 y 1.4 (parte de la aplicación) , en el nivel conceptual, se describen sólo algunos campos que contendrá la tabla clientes cuando se transforme en el **apéndice e).**

I.5 Instancias y esquemas

Las instancias y esquemas es conviene definirlos haciendo la comparación con el mundo real que nunca deja de cambiar y está en constante evolución, por ello, en una **BD** es igual, ya que ésta se encuentra cambiando constantemente según se actualice o se suprima información.

Una **instancia de la BD** es una colección de información almacenada en un momento dado y por ello solamente existe al momento que se consulta la información. Estas instancias son de suma importancia, ya que alrededor de ellas se obtiene la información necesaria en un determinado momento.

Un **esquema** es el diseño lógico global de la **BD**, el cual es analizado y desarrollado para que realice de manera óptima la manipulación de datos, el cual está constituido por pequeños esquemas que forman parte del esquema general.

I.6 Independencia y dependencia de los datos

La independencia y dependencia de los datos es de suma importancia al momento de realizar el análisis y diseño lógico de la **BD**, debido a que de esto depende su funcionamiento óptimo cuando se realicen modificaciones al nivel físico y conceptual.

La **Independencia** de los datos es cuando se modifica un esquema sin afectar el esquema superior siguiente, es decir, sin afectar aplicaciones existentes. Para esta independencia se tienen dos niveles :

Independencia física de los datos. Se da cuando se modifica el esquema físico de los datos sin tener que volver a escribir las aplicaciones.
Por ejemplo, cuando se hace una actualización directamente en una de las tablas y al manipular ese dato con una aplicación existente no provoque ningún tipo de problema.

Independencia lógica de los datos es cuando se modifica el esquema conceptual sin tener que volver a escribir las aplicaciones.
Por ejemplo, cuando se modifica la estructura de una tabla y al referirse a esta tabla por medio de un programa que la manipule no se tenga que volver a escribir la aplicación.

Dependencia de datos es la manera como los datos se organizan y la forma como se accesan, depende de los requerimientos de la aplicación, además del conocimiento de la organización de la **BD** y de las técnicas de acceso que forman parte de la lógica de los programas de aplicación.
Para el ejemplo del (restaurante) las aplicaciones serán de acuerdo a los requerimientos de información por parte de los usuarios que laboran en él.

Una vez que se definieron los anteriores conceptos, es importante definir los pasos que involucra **el ciclo de vida de una BD,** ya que para su cumplimiento requiere de la aplicación de los conceptos y herramientas (Software y Hardware) descritos anteriormente, así como de información que se dará en los siguientes capítulos.
El **ciclo de vida de una BD** muestra qué pasos son necesarios en el diseño metódico, partiendo de un análisis y diseño lógico de la **BD**, el cual es independiente al ambiente del sistema, al diseño físico y a la distribución de red, basado en los detalles del **SMBD** elegido para la implementación de la **BD**.

I.7 Ciclo de vida de una base de datos

El ciclo de vida de una **BD** involucra diferentes pasos en el diseño global del esquema de la **BD**, una vez que se completó el diseño, el ciclo de vida continúa con la implementación y el mantenimiento.

Consta de los siguientes pasos :

1) Análisis de requerimientos :
Determinado por entrevistas entre diseñadores y usuarios, produciendo especificaciones formales de requerimientos. Estas especificaciones incluyen requerimientos y especificaciones de datos, relaciones entre los datos y plataforma para la implementación de la **BD**.

2) Diseño lógico :
Muestra el esquema global y sus relaciones entre datos, utiliza conceptos y definiciones tales como el **modelo-ER**. La metodología del desarrollo del esquema global es la misma para una **BD-distribuida** que para una **BD-centralizada**. Esta última es la que ejemplificaré en este trabajo.
El diseño lógico tiene los siguientes pasos :

a) **Modelo-ER :**
Los requerimientos son analizados y modelados.

b) **Integración de vistas :**
Esto se da cuando el diseño es realizado por varias personas o se utiliza la metodología del diseño descendente por un diseñador, dando origen a múltiples vistas de datos y relaciones. Para eliminar redundancia e inconsistencia y tener una vista global del problema se recurre a este paso.

c) **Transformación del modelo-ER en esquemas de tablas :**
Cada relación asociada con entidades es transformada en esquemas de tablas.
Los incisos a, b y c se describen con más detalle en el Capítulo II.

d) Normalización de tablas.
Aquí se reducen al máximo las tablas, tratando de evitar inconsistencias y redundancia. Para esto se utilizarán **dependencias funcionales (DFs)**, que son derivadas del **modelo-ER**. Estas representan dependencias alrededor de elementos que son llaves de entidades, así como **dependencias funcionales multivaluadas DFMs** , las cuales representan dependencias alrededor de llaves y no llaves con entidades. Estas dependencias son derivadas de los requerimientos de aplicación, además se utilizan **Dependencias de Unión (DUs)** que sirven para reducir al máximo las tablas.
También se muestra que una tabla candidata es derivada de **DFs y DFMs,** usando técnicas de normalización. Por último, la redundancia que todavía pudiera encontrar después de reducir al máximo las tablas, es analizada para su posible eliminación, este paso se describe con más detalle en el capítulo III.

3) Refinamiento
Aquí se seleccionan procesos dominantes de los básicos y de alta frecuencia, que surgen al manipular la **BD**, evaluando total de costos por **QRY**. Esta evaluación no se desarrollará debido a que se encuentra fuera de las especificaciones de este trabajo.

4) Distribución de datos
En este trabajo no se distribuirá la **BD**, por la razón anterior.

5) Esquema global y diseño físico
Aquí es donde se presentan las tablas en su correspondiente definición en un lenguaje **LDD**, este paso se verá ejemplificado en el capítulo II, en las figuras para los casos de transformación, y se mostrará en su totalidad en el **apéndice e)** , donde se ve la transformación del diseño lógico en tablas. Respecto al diseño físico, no se profundizará en este tema debido a que se encuentra fuera de las especificaciones del trabajo, por lo cual sólo se mencionará que representa a la ' forma como están ordenados los datos para hacer más óptima su manipulación, por ejemplo, índexes para mejorar el tiempo de consulta.

6) Implementación, monitoreo y modificaciones de la BD.
En este último paso del ciclo de vida de una **BD** se retoma la definición de datos en **LDD** y la manipulación por medio de un **LMD**, ya que éste se ejemplificará en el capítulo IV y en el **apéndice e),** donde la implementación de la **BD** será utilizando **Sybase** , el cual contiene los dos lenguajes, el **LDD y LMD.** Por ejemplo, " **create table** " representa comandos de **LDD** y un " **select** " representa un comando de **LMD**.
Respecto a este paso se verá con más detalle en el capítulo IV y en los **apéndices e) y g).**

Cuando se implementa la **BD** surgen detalles que es necesario corregir para el mejoramiento de la misma, de este modo el ciclo continúa y el monitoreo tiende a perfeccionar la **BD.**

Resumen

Este capítulo muestra diferentes conceptos que es necesario comprender para saber a dónde se quiere llegar durante el desarrollo de este trabajo. Por lo tanto, en este capítulo se mostró una idea general de lo que son las bases de datos, así como los componentes de un **SMBD**, tales como **datos, SW, HW y usuarios** los cuales son importantes para que iteractúe la **BD**, también se mencionaron los beneficios que se obtienen al utilizar una **BD** para implementar un sistema, tales como **evitar** la redundancia, inconsistencia, el aislamiento de los datos, anomalías del acceso concurrente, así como **tener** facilidad para accesar los datos, mantener su integridad , seguridad , así como definir las reglas del negocio, también se mencionaron los tres **niveles de abstracción**, los cuales son : físico, conceptual y visual, asimismo se vieron **las instancias y los esquemas** al igual que se habló de la importancia de la independencia y dependencia de los datos. Como parte final de este capítulo se vio el ciclo de vida de una **BD**, el cual sirve para tener una visión más clara de cómo se avanza en este trabajo, así como los temas pertenecientes a este ciclo que no se desarrollaron debido a que se encuentran fuera de las especificaciones de este trabajo.

Capítulo II

Modelo Entidad Relación

CAPÍTULO II
MODELO ENTIDAD RELACIÓN

II.A Principios y fundamentos de construcción del modelo-ER

II.A.1 Antecedentes del modelo-ER

[2] El esquema fue formalizado en 1960 por **Charles Bachman**, usando **rectángulos** para representar tipos de registros, además de renglones direccionados de un tipo de registro para otro, denotando relaciones de uno para muchos, alrededor de instancias de registros de diferentes tipos.

La metodología que se utilizará en este trabajo fue formalizada por **Peter Chen** en 1976, usando **rectángulos** para especificar **entidades**, también **rombos para especificar relaciones entre entidades,** las cuales son diferenciadas por números o letras, **conectando el rombo con el rectángulo**, donde el **modelo-ER** es la aproximación más usada para capturar los requerimientos de información del mundo real, además de entendible por los diseñadores de la **BD**. Este modelo proporciona el mejor camino para el diseño lógico de una **BD** y por lo tanto un diseño óptimo para manipular la **BD** por los usuarios, también se mostrará que esta forma de diseño permite la fácil definición e implementación de requerimientos en un **SMBD**. En la figura 2.1 se muestra la notación ocupada por **Chen.**

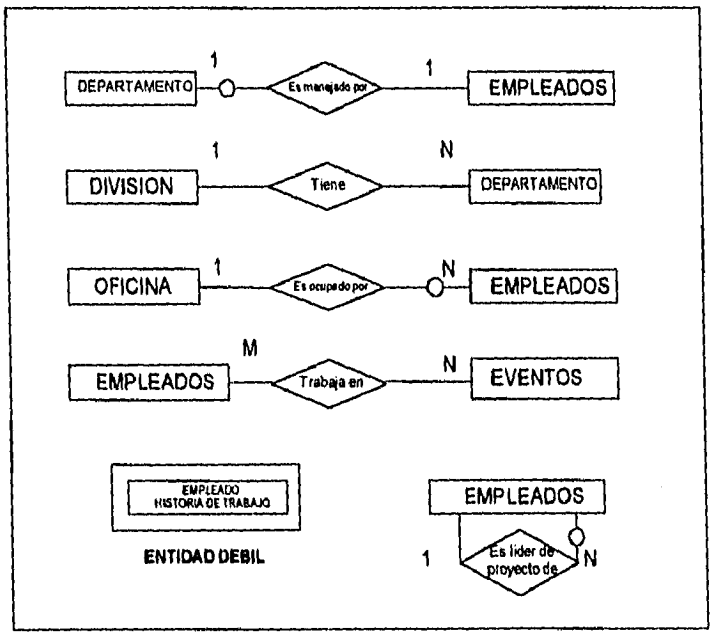


Figura 2.1 Notación de Chen en la construcción del modelo-ER.

[2] [Toby J. Teorey, 2]

Por ejemplo, para el caso de la primer relación que se muestra en la figura anterior, se observa que ésta involucra **dos entidades : departamento y empleados**. Los nombres de estas entidades **se muestran dentro del rectángulo que las representa** en este modelo, también se puede ver que **la entidad departamento es opcional y tiene una cardinalidad de uno** y que **la entidad empleado tiene una cardinalidad de uno y es indefinida**, además, esta relación representa **el roll "es manejado por "** , el cual es representado por medio de un rombo que sirve para hacer la relación entre las entidades descritas anteriormente, de esta manera se van a leer las relaciones que poco a poco se presentan en este trabajo, más adelante se darán definiciones de términos, de los cuales se habla en la anterior interpretación, tales como **opcional, cardinalidad, indefinida y roll** , y que ayudarán a la mejor comprensión del **modelo-ER**.

II.A.2 Objetos para la construcción del modelo-ER

El **modelo-ER** es una abstracción de un mundo real basado en tres tipos de objetos :

1) Entidades

Los primeros se conocen como **entidades,** los cuales son objetos perfectamente identificados y con características propias, estos pueden ser una persona, un lugar, cualquier cosa o un evento de información de interés. El nombre de la entidad se escribe dentro del rectángulo que la representa.

Una **ocurrencia de una entidad** es llamada **entidad instancia o entidad ocurrencia ,** además, cuando se realiza una operación de venta que involucra varias entidades del mismo tipo se le llama **conjunto de entidades**. También se tienen las llamadas **entidades débiles.** Estas se dan cuando las entidades no tienen atributos para formar una llave primaria y dependen de la existencia de un evento determinado para que existan, se identifican representando el nombre de la entidad dentro de un rectángulo doble, además, denotan que todas las ocurrencias son dependientes de la existencia en la **BD** de una asociación con una **entidad fuerte.** Estas **entidades fuertes** son con las que constantemente iteractúa la **BD** y que forzosamente tienen una **llave primaria (PK).**

Por ejemplo, una entidad débil, historia de un empleado, depende de que el empleado exista, además se tiene otro tipo de entidades, las llamadas **entidades virtuales**, que son aquellas que no existen en la **BD**, pero que sirven para describir posibles ocurrencias dentro de una entidad mandatoria. [Nota 2)]

2) Atributos

El segundo objeto con el que trabaja el **modelo-ER** se conoce como **atributos.** Estos son valores descriptivos de los elementos de las entidades, que proveen una descripción detallada acerca de la entidad a la que pertenecen.

Cuando se presenta una **ocurrencia de un atributo** en una entidad o relación es llamada **valor de atributos. Para el desarrollo de este trabajo se representan los atributos en la**

Nota 2)

Las entidades mandatorias se explican más adelante, cuando se muestren las existencias de entidades.

parte superior de la entidad que éstos describan. Para poder comprender lo descrito anteriormente se utilizan dos tipos de atributos :
identificadores y descriptores.

El **Identificador** es una llave usada para determinar una instancia única de la entidad. Se puede identificar fácilmente representado de la siguiente forma: las primeras dos o tres letras, por lo general, son un **alias** de la entidad, es decir, una abreviatura con la que también se conoce a la entidad que se está refiriendo. Después le seguirá un **guión bajo** seguido de las letras **id**, éstas indicarán que se trata de un identificador, puede ser **llave primaria (PK) o llave foránea (FK)** , dependiendo del papel que desempeñe el atributo en la entidad. Como ejemplo considérese el atributo **al_id** de la **figura 2.2** . Las primeras dos letras **al** son el **alias** con el que se conoce la **entidad almacén**, después se ve el **guión bajo** seguido de las letras **id,** que indican que se trata de un identificador.

El **Descriptor (no es llave),** describe características no únicas de una entidad, se puede identificar con su respectivo **alias** de la entidad a la que describe, es decir, las primeras dos o tres letras seguidas del **guión bajo** y la palabra o conjunto de letras que darán una idea del atributo que describe, como ejemplo considérense los atributos de la **figura 2.2** , **al_descripcion** y **al = alias,** de la entidad almacen, seguido del **guión bajo** y la palabra **descripción,** la cual indica que este atributo contiene la descripción de un artículo o un alimento en el almacén; **uso_id** , **uso = alias** de la entidad **lugar_uso** , seguido del **guión bajo** y la letras **id** que indica que este atributo contiene el identificador de lugar donde se utiliza el artículo o alimento en el restaurante.

Otro tipo de atributo es el conocido como **atributo complejo**, se conoce como tal debido a que es una generalización de varios atributos descriptivos, para ejemplificar este tipo de atributo se toma el atributo descriptivo **pr_direccion**, el cual se entiende como el atributo donde se indica la dirección donde se puede localizar a uno de los proveedores. Este atributo se describe por partes, de la misma manera que los anteriores atributos descriptivos, es decir, **pr_ciudad** , **pr = alias** de la entidad proveedores, seguido del **guión bajo** y la palabra **ciudad** , la cual indica que este atributo contiene la ciudad donde radica uno de los proveedores, asimismo describe los casos para **pr_calle, pr_estado y pr_codigop**.

Como se explicó en los anteriores ejemplos, tanto los atributos identificadores como los descriptivos pueden ser atributos singulares o una composición de ellos, es decir, atributos compuestos; también se encuentran los **atributos multivaluados**. Estos atributos se utilizan para describir varios valores para un atributo en un momento determinado. Se representan durante el desarrollo de este trabajo con **(m)** al final del nombre con que se conoce al atributo.

3) Relaciones
El tercer objeto del **modelo-ER** es el conocido como **relaciones** , las cuales **representan ocurrencias** de necesidades de información del mundo real, alrededor de una o más entidades.

Las **ocurrencias** de una relación son **llamadas instancias,** que son solicitudes de información en un momento dado. A las **relaciones del mismo tipo se les llama conjunto de relaciones.** Nota 3)

También se tienen los **rolles**, estos juegan en el **modelo-ER** un papel muy importante y son definidos como la función que tiene una entidad en una relación.
Por ejemplo :
El **roll** " **trabaja en** " define la función de la entidad empleados y departamento, donde se toman los atributos necesarios para ejemplificar dónde trabaja el empleado.
Respecto al **nombre de la relación, se considera igual al nombre que recibe el roll** en una relación cualquiera. El nombre se escribe dentro del rombo que indica dicha relación. Para el desarrollo de este trabajo se encontrarán nombres de relaciones que por su longitud no se pueden escribir dentro del rombo correspondiente, para este caso el nombre de la relación se colocará en la parte superior de las entidades que relaciona , resaltado con letras mayúsculas obscuras. Nota 4)

Los conceptos descritos anteriormente se pueden ver representados en la figura 2.2

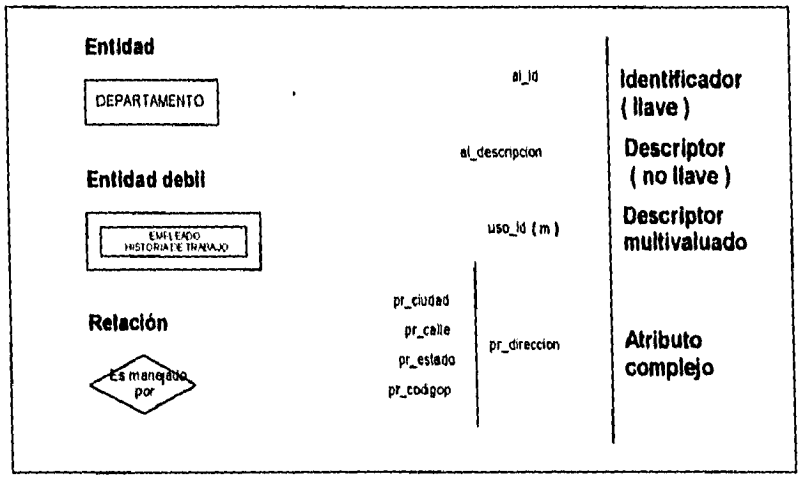


Figura 2.2 Conceptos básicos de construcción del modelo-ER.

Ahora se muestra la importancia del grado de una relación, debido a que entre más entidades contenga una relación, más compleja es la manipulación de entidades. Además de que se requiere de un buen conocimiento de la **BD**, se recomienda tener relaciones con el menor número de entidades, ya que dependiendo del número de entidades que se necesiten para

---

Nota 3)

El significado de una relación es indicado por la conectividad de sus ocurrencias, las cuales se denotan como relaciones de **uno a uno, uno a varios y varios a varios**. Estas conectividades se mostrarán más adelante para su mejor comprensión.

Nota 4)

Cuando se forman relaciones sin importar el grado de la relación y el roll que se maneje en ese momento, se da origen a **vistas**, las cuales son una visión de determinados datos de interés en un momento dado. Estas vistas darán la información necesaria para la toma de decisiones en la empresa, en el capítulo IV, se ejemplifican algunas vistas.

extraer la información requerida, será el grado de la relación que se manipule. A continuación se dan definiciones para los diferentes tipos (grados) de relaciones que se manejan en este trabajo:

II.A.3 Grado de una relación

Es el número de entidades asociadas en una relación, por ejemplo, **una relación binaria es de segundo grado, una ternaria es de tercer grado y una relación de n-entidades es una relación n-aria**. Ahora se describen los tipos de relaciones que hay, así como la forma como se construyen.

Una **relación binaria recursiva** se da cuando los elementos de una entidad están relacionados con elementos de la misma entidad. Esta se construye con una entidad y un rombo con ambos conectores dirigidos hacia la misma entidad.
También se tienen las **relaciones binarias,** relaciones entre dos entidades unidas por un rombo con sus respectivos conectores.
Otras son las **relaciones ternarias,** relaciones de tres entidades unidas por un rombo a través de sus respectivos conectores. Este tipo de relación se utiliza cuando una relación binaria no es suficiente para describir el objetivo de la relación.
El último tipo de relaciones son las **relaciones n-arias.** Estas describen una relación de N entidades alrededor de un rombo con N conectores.

Una vez que se comprendió la forma como se asocian entidades a una relación, como parte importante y para comprender el flujo de información que se da a partir de las relaciones y sus conectores, se describe la **cardinalidad** de una relación, también conocida como **conectividad** de una entidad a otra. [Nota 5)]

II.A.4 Cardinalidad de una relación

Describe el mapeo de asociación y el número de ocurrencias posibles para dicha relación. Los valores que puede tomar son conocidos como el número de entidades con las que se asocia una entidad a otra entidad. Para la mejor comprensión de la cardinalidad, primero se muestra una definición formal de los diferentes tipos de cardinalidades que se utilizaron en este trabajo, más adelante, con figuras se interpretan y se dan ejemplos de las mismas ; estas pueden ser :

[3] ***uno a uno***

Sea la entidad **A** está asociada con una entidad **B** y una entidad **B** está asociada con sólo una entidad **A**.
(entidad fuerte a otra entidad fuerte).

Nota 5)

Más adelante, en este capítulo, se presentan figuras que harán más comprensible el grado de las relaciones.

[3] Las definiciones de cardinalidad se tomaron del libro | Henry F. Korth, 31 |

uno a varios
Sea una entidad **A** está asociada a varias entidades **B** y una entidad **B** está asociada con sólo una entidad **A.**
(una entidad fuerte a otras subordinadas).

varios a uno
Sea una entidad **A** está asociada con una entidad **B** y una entidad **B** sin embargo, puede estar asociada con varias entidades **A.**
(entidades subordinadas a una entidad fuerte).

varios a varios
Una entidad **A** está asociada a varias entidades **B** y varias entidades **B** están asociadas a varias entidades **A.**
(una entidad fuerte a otra fuerte).
La máxima cardinalidad es muchos (N) y la mínima cardinalidad es uno (1).

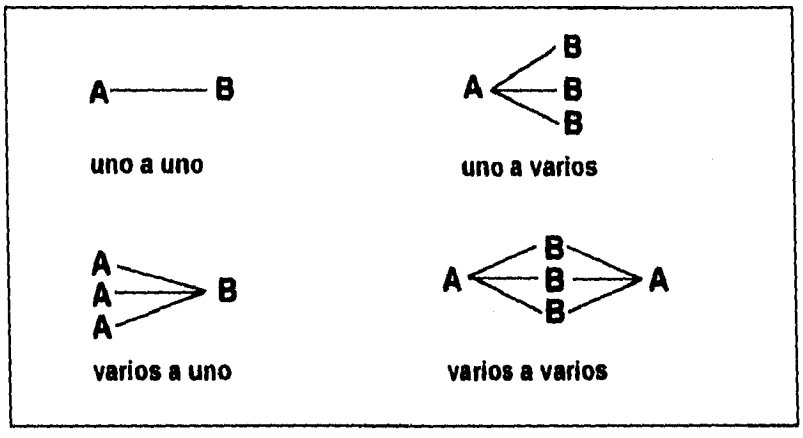


Figura 2.2 a) Posibles cardinalidades.

La cardinalidad de asignación para un conjunto de relaciones es determinada por los requerimientos de información del mundo real y del conjunto de relaciones que está manejando, es decir, de las entidades que se relacionan para satisfacer los requerimientos de información.

En esta parte del trabajo se define la importancia de la existencia de las entidades en una relación.

II.A.5 Existencia de entidades en una relación

Ahora se muestran algunas entidades cuya existencia depende de la existencia de otra entidad, a esto se le llama **existencia por dependencia o existencia.** Un ejemplo de existencia por dependencia puede ser cuando un platillo con identificador único depende de varios alimentos, entonces la existencia del platillo depende de la existencia de los alimentos a utilizar, (si no existe **B** no existe **A**).

(Es posible que se supriman los alimentos una vez elaborado el platillo y que éste exista debido a que el platillo se encuentra preparado, pero cuando se intente preparar nuevamente este platillo, no podrá existir, ya que no existen alimentos de los cuales depende su preparación).

La existencia de una entidad en una relación también se define como : **mandatoria u opcional**

Es **mandatoria** cuando necesariamente una entidad debe existir en una relación, es definida por una línea perpendicular en la conexión entre la línea y la entidad, **su cardinalidad mínima es de uno**. Por ejemplo, en la **figura 2.3** en la relación " **es ocupada por** " , se muestra que una oficina puede tener cero o N empleados, pero un empleado tiene exactamente una oficina, por eso la entidad oficina es mandatoria pues los empleados necesitan una oficina para laborar.

Es **opcional** cuando la ocurrencia de una entidad no necesariamente debe existir. Es definida por un 0 entre la conexión de la línea y la entidad. Por ejemplo en la **figura 2.3** la entidad empleados puede o no puede ser el manejador de cualquier departamento, ocasionando que la entidad departamento y empleados en la relación " **es manejado por** " , sea opcional. Para este caso la cardinalidad es cero entre la entidad y la relación.

Cuando no se encuentra un **cero** o **una línea perpendicular** en una línea que conecta la entidad con la relación, se dice que **la existencia de las entidades es desconocida, en este caso la cardinalidad mínima es 1**. En la **figura 2.3** se puede ver que la **existencia es desconocida** en el roll " **se asigna un** ". Este ejemplo se lee de la siguiente forma: Un empleado tiene un teléfono, pero también se le puede localizar en otros teléfonos, por tanto es desconocido el teléfono donde se le puede encontrar.

Los anteriores conceptos de relaciones se pueden ver representados gráficamente en la **figura 2.3**

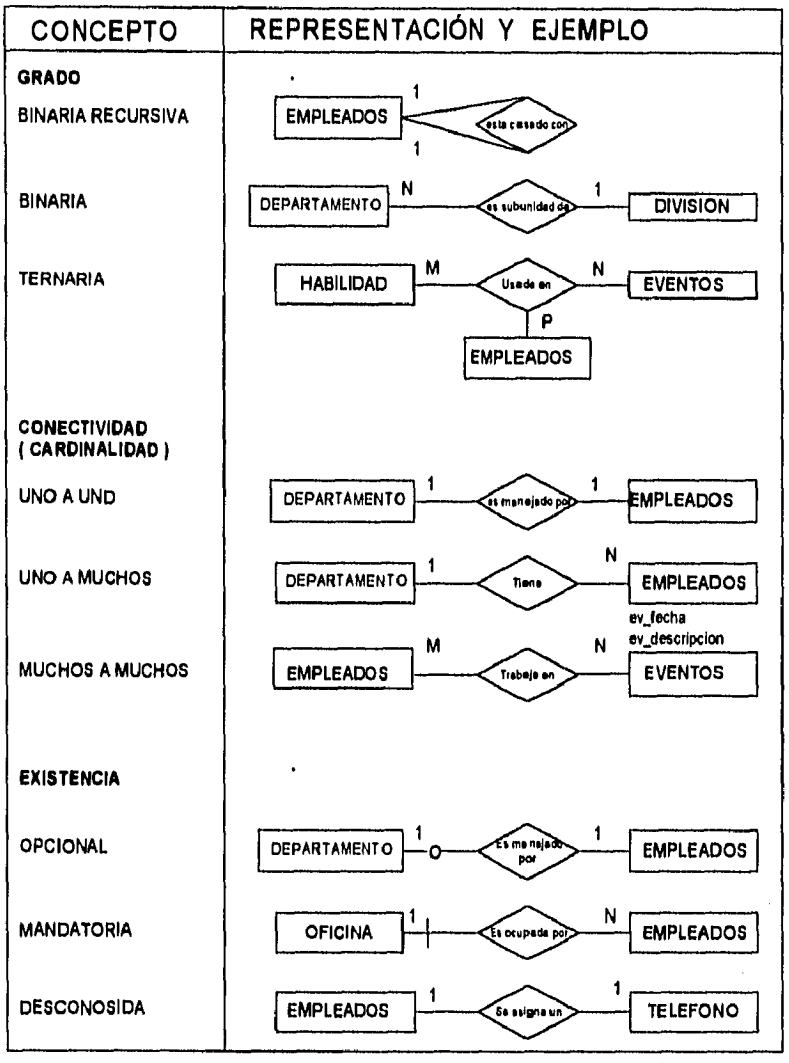


Figura 2.3 Tipos de relación en la construcción del modelo-ER.

De la gráfica anterior se interpretan las cardinalidades de la siguiente forma :

Cardinalidad para la **relación uno a uno** es máxima =1 y mínima = 1 para ambas entidades. Lo que representa esta relación es lo siguiente : un departamento es manejado por un empleado y cada empleado solamente maneja un departamento.

Cardinalidad para la **relación uno a muchos,** es mínima = 1 para la entidad departamento y máxima = N para la entidad empleados, representando lo siguiente : Cada empleado trabaja en un solo departamento.

Cardinalidad para la **relación muchos a muchos** es mínima = M para la entidad empleados, y máxima = N para la entidad departamento, la cual representa lo siguiente, cada empleado puede tener uno o más eventos a su cargo y un evento puede estar manejado por uno o más empleados.

Aquí se ha llegado a una etapa del **modelo-ER**, la cual se verá de una forma más general, de manera que se pueda " **generalizar** " **en una pequeña figura todo un análisis de un problema determinado**. Esta generalización se conoce como construcción avanzada del **modelo-ER.**

II.A.6 La construcción avanzada del modelo-ER
Se generalizará en : supertipos y subtipos

Hasta aquí el **modelo-ER** ha sido usado para tener una representación abstracta de comunicación y relaciones de datos para los usuarios finales, también se usará para representar **la generalización de relaciones** con entidades en las que se encuentran atributos en común y que pueden ser generalizados en **superclases o supertipos de entidad.**
La especialización es lo contrario de generalización, indica que los **subtipos** son especializaciones de un **supertipo.**
Los anteriores conceptos son fáciles de comprender si se parte de una simple abstracción, por ejemplo, si se parte de algo general para dividirlo en partes que conserven características propias de la estructura general a la que pertenecen , tal es el caso de la entidad **empleados** **(supertipo)** y cocineros, meseros, personal de mantenimiento etc. **(subtipos)**, éstas son **entidades virtuales,** las cuales se clasifican de esta manera ya que no existen y a su vez son la representación de determinadas partes de la empresa con características similares, por esto se pueden unir en una general llamada empleados, aplicando la generalización.

Por otra parte, un **supertipo** de entidad en una relación puede ser **subtipo** en otra relación de entidades, esta combinación de **supertipo/subtipos** recibe el nombre de **generalización de hierarchy.**
Las generalizaciones se dividen en dos tipos : **segmentos,** representados por la letra (**S**) **figura 2.4 a)** y **complementos,** representados por la letra (**C**) **figura 2.4 b).** Estos complementos son parte de un **supertipo.**
También se tiene la **Integración,** que está formada por la integración entre **supertipos y subtipos** de entidades y se lee como **parte de**. Por ejemplo, un platillo está formado de alimentos y accesorios para su presentación, también puede ser como se muestran los productos terminados de **SW**, formados de programas y guía de usuario. Esto se puede entender como la integración de **supertipos y subtipos** para obtener productos terminados, representados por la letra (**I**) como se muestra en la **figura 2.4 c).**

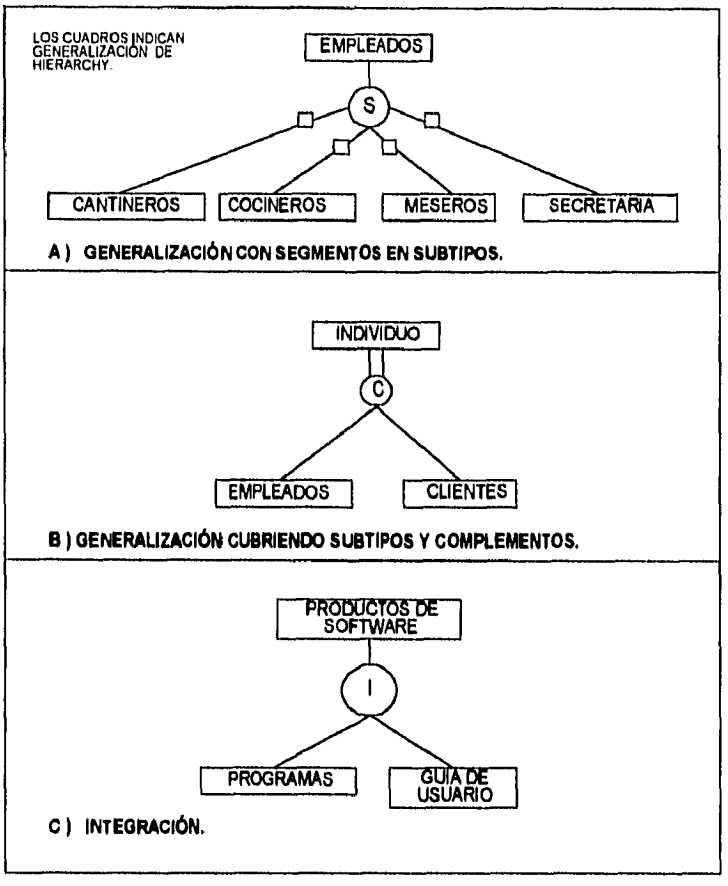


Figura 2.4 Generalización, complementos e integración.

II.A.7 Importancia de las relaciones ternarias

Ahora se muestra la importancia de las relaciones ternarias en una relación, ya que éstas son parte de la solución de requerimientos de información. Se utilizan para describir un evento que una relación binaria no puede resolver.

En los siguientes ejemplos se muestran los cuatro casos más comunes que se presentan en la construcción de relaciones ternarias, estos casos interpretarán a la cardinalidad del **modelo-ER**, además de las **dependencias funcionales (DFs)** asociadas al caso correspondiente.

En la interpretación de estas relaciones se mostrará cómo se lee la cardinalidad de la relación, es decir, las posibles interpretaciones de datos que se tienen en las tablas y cómo se obtendrán, además de las restricciones en cuanto a la manipulación de datos.

Por ejemplo, en la primera interpretación en la relación **" usa recetario "** :

1) Un cocinero usa exactamente una receta para cada platillo, es decir, no puede usar dos recetas para preparar un platillo.

2) Cada receta pertenece a un cocinero para cada platillo, es decir, la receta es manejada por un cocinero en especial que prepara el platillo correspondiente.

3) Un cocinero tiene habilidad para preparar varios platillos y tener diferentes recetas para diferentes platillos, esta última aserción indica que un cocinero no se limita a preparar un solo

platillo y que éste puede tener diferentes recetas para diferentes platillos, pero cuando prepare un platillo deberá ocupar una receta.

De esta manera se interpretan y se comprenden las cardinalidades de asignación de cada relación. Para el caso de las **dependencias funcionales (DFs)** mostradas en esta relación, se definirán más adelante con más detalle en el capítulo III, donde **em_id** es una **llave primaria (PK)** de la entidad empleados, **pl_id** es **llave primaria (PK)** de la entidad platillos, **re_id** es **llave primaria (PK)** de la entidad recetas. [Nota 6)] Los diferentes casos de relaciones ternarias se pueden ver en la siguiente figura 2.5

Nota 6)

A partir de estas **llaves primarias (PKs)** se obtendrá la información descriptiva necesaria para esta relación. De la misma forma que se representó e interpretó la anterior relación, se interpretarán las relaciones n - **arias**, además de las **dependencias funcionales (DFs)** asociadas a la misma.

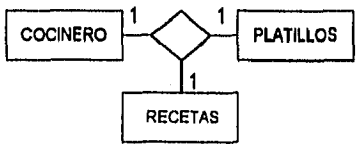
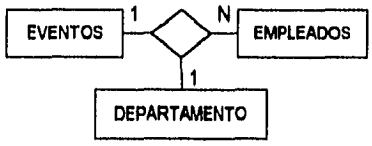
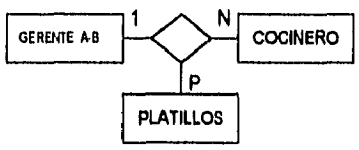
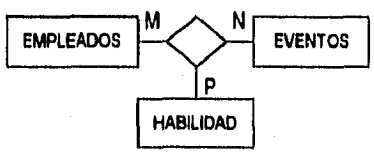
INTERPRETACIÓN	REPRESENTACIÓN GRAFICA	DEPENDENCIAS FUNCIONALES (DF)
1) UN COCINERO USA EXACTAMENTE UNA RECETA PARA CADA PLATILLO. 2) CADA RECETA PERTENECE A UN COCINERO PARA CADA PLATILLO. 3) UN COCINERO TIENE HABILIDAD PARA COCINAR VARIOS PLATILLOS Y TENER DIFERENTES RECETAS PARA DIFERENTES PLATILLOS. RELACIÓN TERNARIA UNO A UNO A UNO.	**RELACIÓN : USA RECETARIO**  	em_id, pl_id -> re_descripcion em_id, re_id -> pl_nombre pl_id, re_id - > em_id
1) CADA EMPLEADO ESTA ASIGNADO A SOLO UN EVENTO DE TRABAJO EN UN DEPARTAMENTO, PERO PUEDE HABER UN DEPARTAMENTO DIFERENTE ASIGNADOS A UN EVENTO DIFERENTE. 2) EN UN DEPARTAMENTO, UN EMPLEADO TRABAJA EN SOLO UN EVENTO. 3) EN UN DEPARTAMENTO EN PARTICULAR PUEDE HABER EMPLEDOS ASIGNADOS PARA REALIZAR UN EVENTO. RELACIÓN TERNARIA UNO A UNO A VARIOS.	**RELACIÓN : ASIGNADO A**  	em_id, de_nombre - > ev_descripcion em_id, ev_descripcion - > de_nombre
1) UN COCINERO TRABAJA BAJO ORDENES DEL GERENTE DE A-B Y PUEDE ESTAR TRABAJANDO EN VARIOS PLATILLOS. 2) UN PLATILLO BAJO LA DIRECCIÓN DE UN GERENTE DE A-B, PUEDE TENER VARIOS COCINEROS. 3) UN COCINERO QUE TRABAJA EN UN PLATILLO Y DEBE TENER SOLO UN GERENTE DE A-B QUE LO SUPERVICE. RELACIÓN TERNARIA UNO A VARIOS A VARIOS	**RELACIÓN : SUPERVISADO POR**  	pl_nombre, em_id - > pu_id
1) UN EMPLEADO PUEDE USAR DIFERENTES HABILIDADES EN CUALQUIER EVENTO. 2) CADA EVENTO TIENE VARIOS EMPLEADOS CON DIFERENTES HABILIDADES. RELACIÓN TERNARIA VARIOS A VARIOS A VARIOS	**RELACIÓN : USA HABILIDAD EN**  	NINGUNA

Figura 2.6 Propiedades y tipos de relaciones ternarias.

Resumen

Hasta este momento se han mostrado los conceptos básicos en la construcción del **modelo-ER**, tal como una **entidad,** la cual puede ser una persona, un lugar o un evento de información de interés; los **atributos** son objetos que proveen información descriptiva acerca de la entidad a la que pertenecen, estos atributos pueden ser únicos (identificadores) y no únicos (descriptores), respecto a las **relaciones,** se vio que éstas describen la conectividad entre instancias de entidades, también se vio el **grado de una relación** que es el número de entidades asociadas a una relación. Respecto a los conceptos de **existencia en una relación,** se mencionó que éstos determinan si una **entidad es mandatoria u opcional**. También se mencionaron los conceptos de **generalización**, que permiten la formación de **supertipos y subtipos** de abstracciones, asimismo se trato la **importancia de las relaciones ternarias.**

II.B Modelo-ER en el diseño lógico de una BD

En esta parte del trabajo se observará una aproximación gráfica de los requerimientos de información por medio del **modelo-ER**, para lo que se tendrán requerimientos de análisis y diseño lógico. Estos dos pasos por lo regular se realizan juntos, para llegar a un modelo lógico de la **BD**.

El **modelo-ER** permite clasificar entidades, atributos, identificar **generalizaciones de hierarchy**, definir relaciones alrededor de entidades , además de otras abstracciones. El diseño lógico de una **BD**, además de auxiliarse del **modelo-ER**, es acompañado por variedades de diseño tales como el diseño descendente.

En la práctica, si se utiliza el diseño descendente puede ser que de aquí salgan los requerimientos de análisis, pero además de estas técnicas se utiliza el **modelo-ER** como una herramienta poderosa para atacar cualquier tipo de problema lógico que se presente, proporcionando una visión más amplia de cada una de las partes que forman los requerimientos al momento de realizar el diseño; se iniciará el diseño lógico de la **BD**, dando una visión de cómo se lleva a cabo el análisis de requerimientos.

II.B.1 Análisis de requerimientos

Es el paso más importante, ya que de aquí dependen la definición y especificación formal de requerimientos del **SW** a implementar. Para realizar el análisis y diseño lógico se deben considerarse los siguientes pasos :

1) Saber cúal es el problema a resolver.
2) Buscar literatura relacionada con el problema.
3) Saber el giro de la empresa y qué requerimientos tiene de información.
4) Una vez que se sabe qué necesidades se tienen de información, así como el giro de la empresa, se procura entrevistar a los usuarios.
5) Estas entrevistas darán un panorama más amplio del flujo de datos, así como de las diferentes áreas que conforman la empresa. Al tener esta visión del problema sigue el paso 6.

6) Interpretar los requerimientos en términos de objetos primitivos, así como los diferentes departamentos que iteractúan para proporcionar información confiable para la toma de decisiones dentro de la empresa , es decir, tener una idea de lo que se trata el problema a analizar.
7) Identificar generalizaciones de hierarchy
8) Una vez que se tiene la anterior información, se describen los objetos y las relaciones que éstos involucran.
9) Lo siguiente es determinar el tipo de transacción, es decir, el flujo que tendrá la información y la iteración con los objetos.

10) También se deberá describir el mantenimiento, integridad, seguridad y reglas del negocio.
11) Después se tiene que hacer un análisis de **SW** y **HW** a utilizar.
12) Una vez que se completaron los anteriores pasos, es necesario hacer documentos formales de especificaciones de la **BD**, para entregarlos a las personas que requirieron el **SW** y de esta forma estar de acuerdo con lo propuesto en los documentos, contra lo que se habló en las entrevistas. Estos documentos son de suma importancia, debido a que una vez aceptados por las personas que requirieron el **SW**, se procede al diseño tal como se acordó.
13) Por último se define el diccionario de datos. Esta es una herramienta muy importante para el diseñador de aplicaciones y el **administrador de la base de datos (ABD).**

Del primero al quinto paso se entiende que ya se realizó y se obtuvo la información necesaria para ejemplificar el desarrollo de este trabajo. Los pasos que se desarrollarán a continuación son del sexto al noveno, donde ejemplifican el diseño lógico de una **BD**. Respecto al paso diez, se ve conforme avanza este trabajo.

En el paso 11 se mostrarán las ventajas de elegir Sybase para implementar la **DB** en lugar de un manejador de archivos.

Los manejadores de archivos son productos tales como **dbase, foxpro y paradox** , que forman parte de la familia de **xbase** (sistemas manejadores de archivos). Estos incluyen las dos partes que forman un sistema **" frond - end "** (herramientas de aplicación), así como un **" back - end "** (base de datos) que maneja los archivos de datos.
Algunos de estos productos soportan **ODBC** (conectividad abierta para bases de datos) y pueden accesar datos de **BD** relacionales, tal como **Sybase**, es decir, algunos pueden usarse como **" frond - end "** y **Sybase** como **" back - end "**.
Estos productos tienen una arquitectura similar y solamente pueden usarse, no implementa la arquitectura cliente servidor.
Cuando se usen aplicaciones de múltiples usuarios en cada PC, éstos deben ejecutar el procesamiento de los datos, lo cual no es un servidor. Esta arquitectura de servidor de archivos es una gran diferencia con **Sybase**, además estos productos tienen muy poco o no soportan proceso de transacciones.

Ventajas de Sybase
Estas ventajas son :
Se puede usar como una **BD** local, como un **servidor** que soporte la integridad de la **BD** o un **cliente** que haga peticiones al servidor en múltiples aplicaciones.
Soporta procesos de transacciones, teniendo su propia bitácora. Esto asegura la consistencia de la **BD**, ya que no se registran físicamente en la **BD** hasta que se hayan ejecutado sin ninguna corrupción.

Escalabilidad
Las aplicaciones que corren en una **PC** pueden ser diseñadas para correr en un ambiente multiusuario dentro de una red, además tiene el standard de **ANSI SQL** (standard de instrucciones para un lenguaje estructurado de consulta).

Arquitectura Cliente / servidor
Soporta la arquitectura **cliente / servidor**, donde los clientes hacen peticiones de datos a través de la red al servidor y éste se encarga de procesarla y mandar los requerimientos al cliente, con esto se gana que todo el procesamiento se realice en el servidor.
Con esta arquitectura se reduce el tráfico en la red, ya que sólo peticiones van al servidor y de esta manera las respuestas son más rápidas para el cliente, además de reducir el código fuente en los clientes, ya que por lo regular se hacen validaciones de ciertos datos y se llama a rutinas en el servidor para que ejecuten el procesamiento.

Proceso de transacciones
Sybase soporta completamente el proceso de transacciones con las instrucciones **COMMIT** para ejecutar y grabar físicamente en la **BD** una transacción o **ROLLBACK** para deshacerla y dejar la **BD** como estaba, esto asegura que la **BD** no se corrompa dado el caso que ocurriera alguna falla del sistema.

Otras ventajas
Sybase ofrece otras ventajas tales como mejor **performance**, mejora la integridad de los datos y la funcionalidad de la **BD**, por consiguiente.
Es soportado por múltiples plataformas como windows 3.x, windows 95, windows NT, Netware de novell, OS/2, MS-DOS, además de otras plataformas más robustas.
También tiene **procedimientos almacenados**, los cuales son sentencias en **SQL**, almacenados en la **BD** y ejecutados cuando se invoque a éstos por la aplicación de algún cliente, con esto se logra que cualquier cambio a la programación de los objetos de la **BD** sólo se haga en el servidor, y por lo tanto tener un control centralizado de las aplicaciones que iteractúan con la **BD**, además éstos accesan a la **BD** sin provocar tráfico en la red, ya que no se ejecutan desde un cliente.
También soporta **triggers**, que son procedimientos asociados con una o más tablas. Estos son invocados automáticamente cuando algún registro de la **BD** es insertado, actualizado o borrado. Los triggers son usados para implementar las reglas del negocio, además de soportar borrado y modificaciones en cascada.

Conectividad a través de ODBC
Este tipo de conectividad asegura que se puede escoger cualquier ambiente gráfico de " **front - ends** ". Esto incluye herramientas tales como Powersoft Powerbuilder, Powersoft Infomaker, Microsoft Acces, Microsoft Visual Basic, Logic Works Erwin, etc.
Además una característica muy importante de **Sybase** la da su optimizador de **QRYs**, ya que dependiendo de la consulta, este optimizador tiene la capacidad de escoger la mejor ruta para su ejecución, ya que por medio de ésta se examina la distribución de los datos.
Sybase tiene otras características adicionales ,como respaldos en línea, soporta múltiples **BD**, soporta diferentes protocolos de comunicación, etc.

Requisitos de HW
Computadoras compatibles con IBM, con CD-rom optativo, DOS versión 3.x, Windows 3.x, Windows 95, Windows NT 3.x u OS/2 2.x.

Requisitos del servidor de la BD
Computadora compatible con IBM, Windows 3.x, Windows NT 3.x, Netware de Novell 3.x u OS/2 2.x. Procesador intel 80386 o posterior, 8 **MB** de **RAM** disponible.

Requisitos de la red
NetBIOS, TCP/IP o IPX de Netware de Novell, (los clientes de DOS soportan NetBIOS e IPX solamente).

Estadísticas de producto :
Bases de datos
63 bases de datos por servidor, tamaño de la **BD hasta 12 TB**, hasta 32767 tablas por consulta.

Tablas
32,767 tablas por base de datos, 999 columnas por tabla, 32,767 índices por tabla, tamaño de la tabla hasta 1024 **GB**, filas por tabla limitadas por el tamaño de la tabla, 999 columnas por índice compuesto, 128 caracteres por nombre de objeto de la base de datos.

Procedimientos almacenados y triggers (disparadores)
Tamaño máximo de 2 GB, 32,767 procedimientos almacenados por base de datos, anidamiento limitado sólo por el espacio en disco.
Por las características mencionadas anteriormente se decidió utilizar **Sybase** en lugar de un manejador de archivos, además de que Sybase es un producto a muy bajo costo.

En el paso 12 se entenderá que ya se aceptó el **SW** por los usuarios finales, en el paso 13 sólo se definirán los datos pertenecientes a la creación de la **DB**, éstos se pueden ver en el **apéndice e).**

La forma como se realizará el análisis y diseño permitirá transformar **el modelo lógico en entidades normalizadas** y éstas en relaciones. Además de dar **reglas de integridad,** también se utilizan **generalizaciones,** integrando éstas a esquemas que serán **vistas** para los usuarios finales.
A grandes rasgos, esto es lo que se hace en el análisis. Ahora se tocarán con más detalle los anteriores pasos. [Nota 7)]

II.B.2 Diseño lógico de una BD
Para realizar el diseño lógico de una **BD** se tienen los siguientes pasos :

1) Clasificación de entidades y atributos
El primer paso es la clasificación de entidades y atributos. Esta clasificación se lleva a cabo durante una entrevista, preguntando por ejemplo :
¿ Cuáles son las áreas que conforman la empresa ?
¿ Qué tipo de información se maneja en la misma ?
¿ Quién trabaja en cada una de estas áreas ?
Estas son algunas de las preguntas que se pueden formular para tener una idea general y poder abstraer los nombres que deben llevar las entidades que se identificaron, así como los atributos que pertenecen a las mismas, además del **roll** que tendrán éstas en el sistema.
Al final de este paso, las entidades deberán tener descripción del objeto al que pertenecen, también se identificarán **atributos** con características propias, así como **atributos multivaluados,** colocándolos en entidades separadas para simplificar el modelo. Estos atributos se verán más adelante, en el capítulo III, al momento de normalizar las tablas.

Nota 7)

Respecto a conceptos que se tocaron aquí por primera vez, tales como **entidades normalizadas, integridad, seguridad y reglas del negocio,** se explicarán con más detalle en los capítulos III y IV.

2) Identificar generalizaciones de hierarchy

El segundo paso es identificar generalizaciones de hierarchy. Se definieron como propiedades heredables de un objeto general, dividiéndose en dos o más objetos que conservan características similares, tales como : un empleado tiene varios objetos que guardan propiedades similares a las del objeto general, como se puede ver en la siguiente figura :

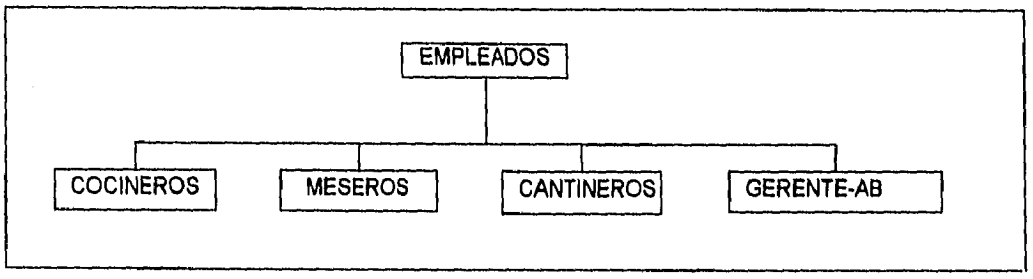


Figura 2.6 Generalizaciones de hierarchy.

Además deben tener identificadores con características únicas para poder referirse a los identificadores que conforman la generalización sin afectar los demás atributos, por ejemplo :

empleados	**em_id**
cocineros	**em1_id**
meseros	**em2_id**
cantineros	**em3_id**
gerente	**em4_id**

Para la figura anterior se puede observar que tanto cocineros, meseros, cantineros y el gerente, son empleados que trabajan dentro de la empresa y que conservan características similares que les da el hecho de ser empleados, por eso se ocupa una generalización de hierarchy. [Nota 8)]

3) Definir relaciones

El tercer paso es definir las relaciones, aquí se identifican posibles entidades a relacionarse, así como **el grado, conectividad, si es mandatoria u opcional** , además de **eliminar las relaciones redundantes**, esto es, aquéllas que representen la misma idea; para realizar esto se debe tener en cuenta que dos o más relaciones son permitidas con la misma entidad, siempre y cuando su significado sea diferente para realizar transacciones en los **rolles** que les corresponden.

Respecto a las relaciones ternarias, como se mencionó en la primera parte de este capítulo, una relación ternaria **se utilizará cuando una relación binaria no sea suficiente para describir un determinado roll**. De la misma manera, cuando una relación ternaria no sea suficiente para describir otro determinado **roll** se utilizará una **relación n - aria** que sí satisfaga el requerimiento de información.

Nota 8)

Para los ejemplos que se desarrollan en este trabajo, es importante tener en cuenta este tipo de generalizaciones.

Una vez que se comprendieron los anteriores pasos para el diseño lógico, se mostrarán las ventajas que proporciona el **modelo-ER**.

II.B.3 Ventajas de utilizar el modelo-ER

Las entrevistas con usuarios finales dan como resultado necesidades de información que serán resueltas por medio de relaciones entre entidades.
La sintaxis de las gráficas hace entendible la representación de **transacciones** y **rolles** que tienen las entidades dentro de la organización, además el **modelo-ER** permite identificar entidades opcionales y mandatorias, también permite, una vez que se tienen las gráficas, la representación en tablas que conformarán una **BD**. Esta representación gráfica (**modelo-ER**), es lo más parecido a los requerimientos de información del mundo real, éstas se verán más adelante en los capítulos III y IV.

Otra ventaja de utilizar el **modelo-ER** es que permite la **integración de vistas,** éstas se dan en el caso de que un grupo de desarrollo realizara el diseño lógico y por esto, tendrían varias vistas o cuando se emplea la técnica del diseño descendente, de tal forma que para tener una vista general de un problema determinado se tengan que integrar los pequeños problemas ya analizados y resueltos, para que al integrarse resuelvan un problema en particular, planteado en una primera instancia. Para lograr esto se comparan y verifican estas vistas, para evitar que se encuentren relaciones redundantes, ya que podrían surgir diferencias en niveles de abstracción, conectividad o redundancia en atributos. Para realizar la integración de una forma ordenada se muestran los siguientes pasos :

II.B.4 Pasos en la integración de vistas

1) Análisis de preintegración

Aquí se realiza el análisis de preintegración y se debe evaluar qué tipo de relación es conveniente para representar el **roll** que tendrán las entidades en la **BD**.

2) Comparación de esquemas

En este paso se comparan esquemas durante el desarrollo de los mismos, para evitar que se tenga un conflicto al momento de la integración, un ejemplo de conflicto podría ser cuando haga referencia a diferentes llaves en una entidad en diferentes vistas, ya que el identificador por el que se está realizando la relación debe ser el mismo en las diferentes vistas.

3) Conformación de esquemas

El tercer paso es la conformación de esquemas, aquí se utilizan técnicas tales como **segmentos y complementos,** así como **supertipos y subtipos** para obtener las nuevas relaciones.

4) Integración y reestructuración de esquemas.

El cuarto paso es la integración y reestructuración de esquemas, donde remueven todas las redundancias tanto en atributos , como en esquemas de **hierarchy**.

Para poder ilustrar toda la teoria descrita anteriormente, se analizan por separado las entidades **empleados, departamento y puestos** de la siguiente figura 2.7 .

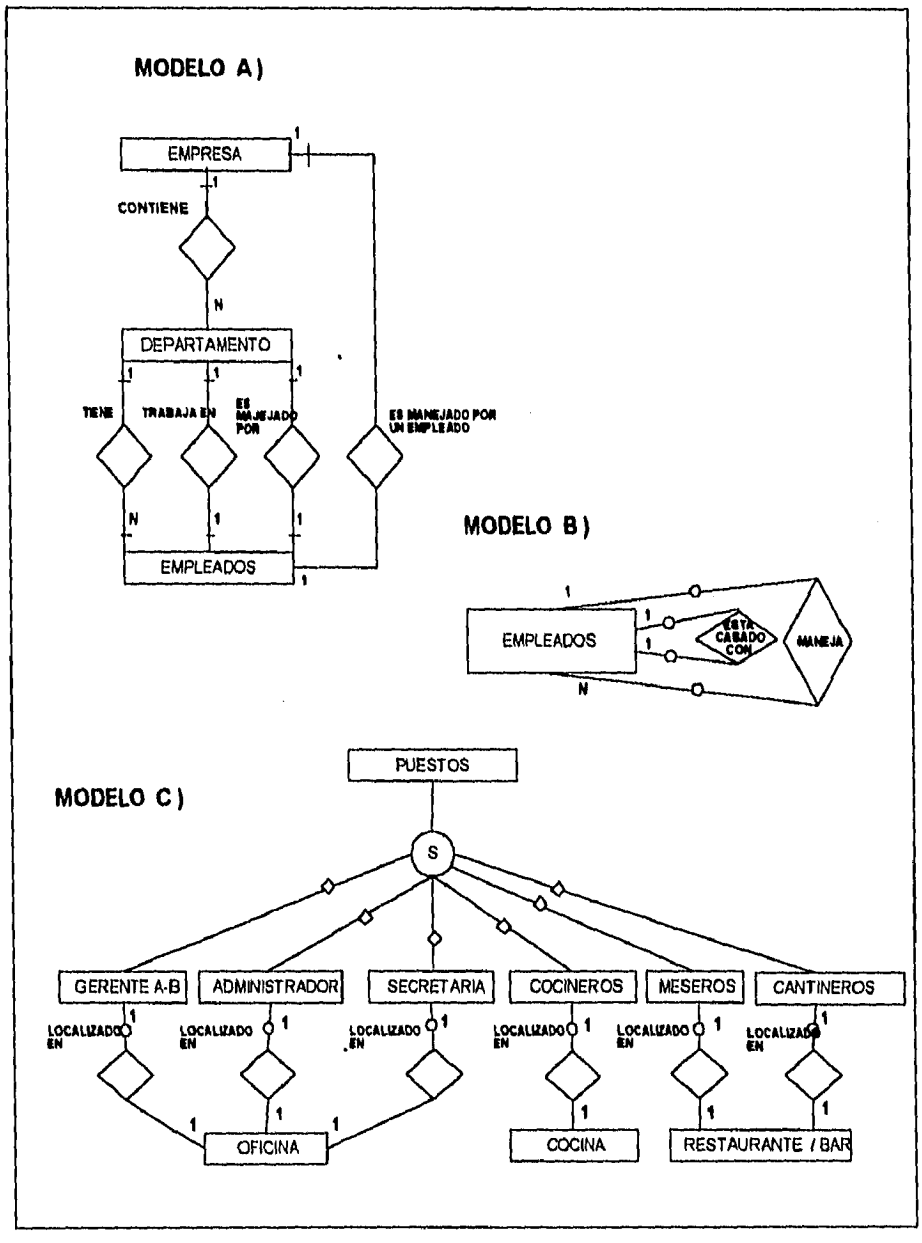


Figura 2.7 Vistas de las entidades empleados, departamento y puestos.

Estas entidades son una parte de las que conforman toda la organización, por lo tanto se mencionan algunas ; las cardinalidades, rolles y existencias que tendrán estas entidades.

Cardinalidad para las entidades de la figura 2.7
Modelo a)
Se puede ver que una empresa, en este caso es el restaurante, está dividida en dos o más departamentos y que varios empleados pertenecen a estos departamentos. En este modelo se utilizan relaciones binarias, además se observa que estas entidades son mandatorias, ya que sus existencias son necesarias en esta relación.

Modelo b)
Se observa que un empleado puede "**estar casado con**" sólo un empleado, además este empleado puede tener a su cargo uno o más empleados. En este modelo se utilizan relaciones binarias recursivas, también se ve que esta entidad para estos rolles es opcional.

Modelo c)
Se observa que puestos es una **generalización de hierarchy**, que contiene los puestos asignados a un empleado cuando se da de alta en la empresa. Estos puestos son entidades virtuales, que a su vez son otra **generalización de hierarchy**. Aquí se utilizan tanto entidades virtuales opcionales como indefinidas.

Respecto a los rolles que juegan las entidades en esta figura :
Modelo a)
Se tienen los siguientes rolles : una empresa " **contiene** " varios departamentos, un departamento " **tiene** " varios empleados, un empleado " **trabaja en** " un departamento, un departamento " **es manejado por** " un empleado y por último una empresa " **es manejada por un** " empleado.

Modelo b)
Se tiene el **roll** un empleado " **esta casado con** " sólo un empleado, también se puede ver el **roll** " **maneja** " varios empleados.

Modelo c)
Se tiene el **roll** para los posibles puestos que se asignan a un empleado, además del **roll** donde es " **localizado** " .

Respecto a la existencia de entidades figura 2.7
Para los rolles que se manejan en esta figura, las entidades empresa, departamento y empleados son mandatorias debido a que forzosamente deben existir, por ejemplo, una empresa, en este caso el restaurante, debe estar dividido en departamentos y estos deben contener empleados que laboren dentro. Respecto a las entidades virtuales puestos, unas son opcionales y otras mandatorias, pero éstas deben estar contenidas dentro de la entidad empleados, con excepción de oficinas, cocina, restaurante y bar.

Como se pudo ver en estos modelos separados a, b y c, se utilizan conceptos propios del **modelo-ER**, tales como grados, conectividad y existencia de una relación, además de segmentos y complementos propios de generalizaciones de hierarchy.

Como siguiente paso se muestra cómo agrupar los modelos a, b y c en un sólo modelo, además de quitar las relaciones redundantes que se pueden encontrar. Si se retoma el modelo c), se puede ver que el **roll** , **" localizado en "** , es redundante. En este modelo se puede sustituir por un solo roll, conservando sus conectividades con las entidades virtuales que se están relacionando, como se muestra en la **figura 2.8** de relaciones redundantes.

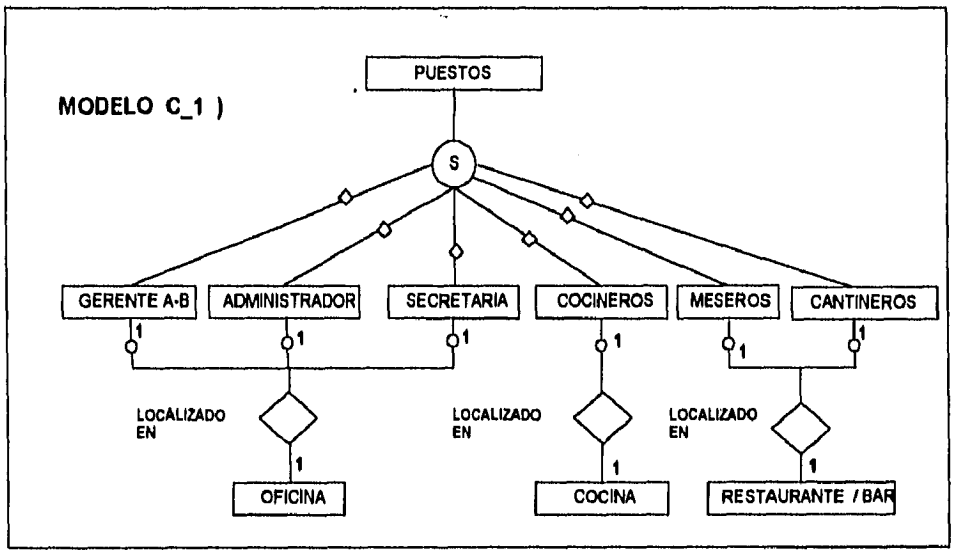


Figura 2.8 Vista de la entidad puestos (eliminando redundancias).

Una vez que se elimina la redundancia del modelo c), como lo muestra **figura c_1)** , se agrupa en el esquema global representado con los modelos a, b y c_1, como se muestran en la **figura 2.9** .

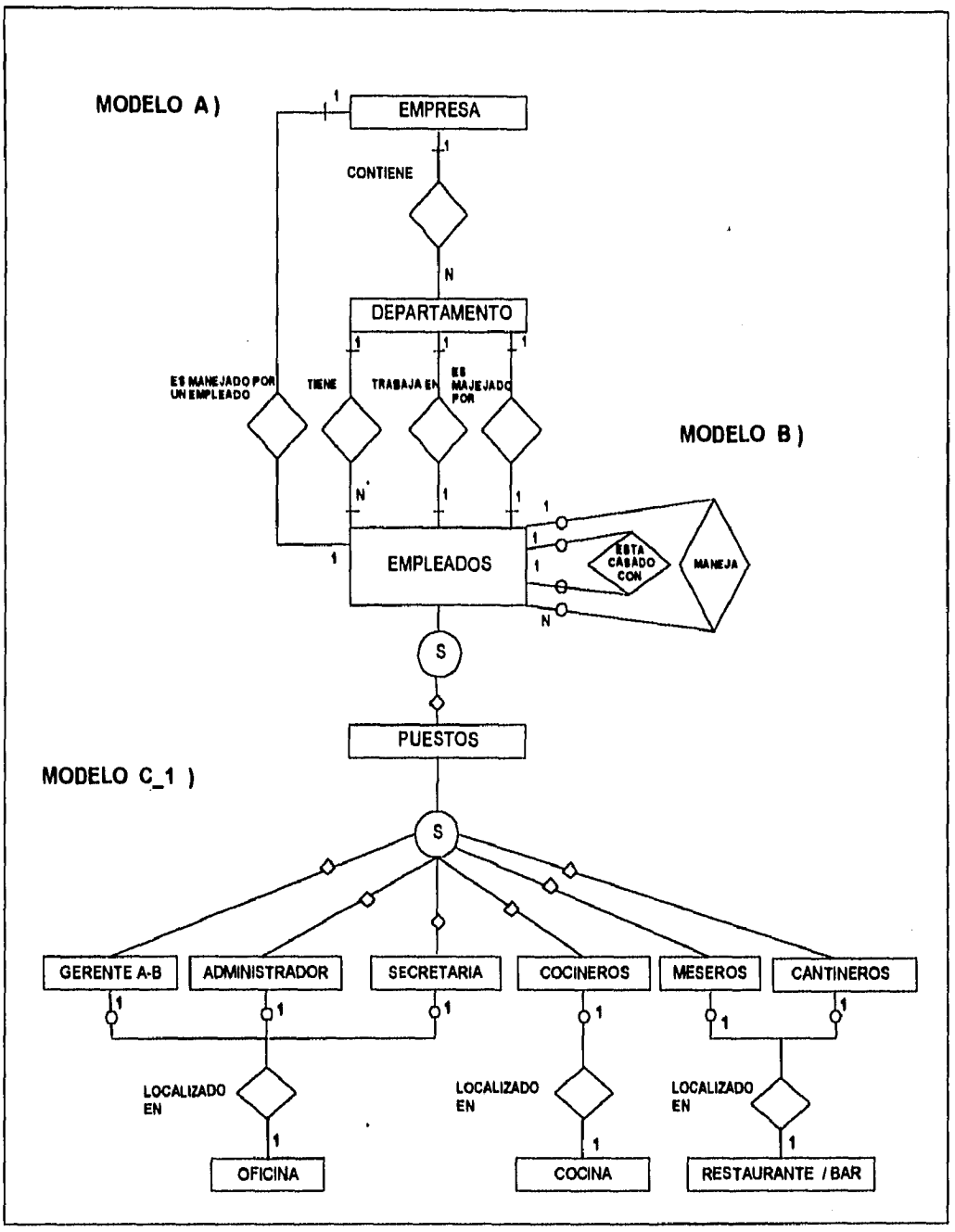


Figura 2.9 Agrupación de vistas.

Este esquema es una simple integración de vistas que conforman el esquema general de la **BD**.
Nota 9)

Nota 9)

Estos esquemas se transformarán en tablas, como se muestra más adelante.

Una vez que se comprendieron los anteriores pasos, como agrupar una vista, en esta parte del trabajo se presenta la forma de agrupar entidades y relaciones.

II.B.5 Agrupación de entidades y relaciones

Este concepto es muy importante , debido a que se puede resumir una vista con características extensas en una vista que conservará las características anteriores, pero ésta representada de una manera más explícita, además de dar un método para organizar esquemas en la **BD**.

Esta agrupación se utilizará cuando se tenga una **BD** o estructuras de información muy grandes, relacionadas con diferentes componentes que no se puedan entender fácilmente.

Para agruparlas se utilizan los siguientes pasos :

1) Se definen puntos a agrupar dentro de las áreas funcionales

En este primer paso se definen puntos que se pueden agrupar dentro de las áreas funcionales, es decir, se identifican entidades dominantes en la relación; en las relaciones n-arias se debe encontrar el foco de la relación, si éste no existe, se considera la relación como general y se debe agrupar como un grupo general.

2) Se definen las formas de agrupar entidades

En el segundo paso se definen las posibles formas en que se pueden agrupar las entidades, aquí se deben identificar entidades a agrupar que pertenezcan a la misma área, si se encuentra una visión clara para agrupar una relación no se debe separar.

3) Forma avanzada de agrupar entidades

En el tercer paso se presenta una forma más avanzada de agrupar entidades. Esta agrupación será recursiva, siempre y cuando se permita sin ninguno de los problemas mencionados en los dos pasos anteriores.

4) Validar el diagrama de agrupación.

En este paso se verifica que el diagrama al que se ha llegado después del proceso tenga el mismo significado en cada nivel que el proporcionado por los usuarios finales.

Para ilustrar los cuatro anteriores pasos obsérvese, por ejemplo, la entidad puestos en la **figura 2.9** , donde **las entidades virtuales gerente A-B, administrador y secretaria** se pueden agrupar en una tipo **clustered**, asimismo para **cocineros** tipo **clustered** y **meseros, cantineros** tipo **clustered**. Finalmente la agrupación permite la simplicidad y comprensión en las relaciones. Nota 10)

En este paso del diseño lo que se hace es tomar pequeñas partes de un problema en particular de una determinada área de la empresa. Cuando este problema ya se analizó y se dio solución a cada uno de los pequeños problemas que se abordaron, se representan por una figura en particular, es este caso será **un rectángulo** dentro del cual tendrá el nombre significativo del

---

Nota 10)

Por entidad agrupada se entenderá la agrupación de entidades y sus correspondientes relaciones dentro de su nivel de abstracción.

problema que se abordó y la palabra **clustered**, tomando en cuenta que este rectángulo conservará la conectividad con el resto de la integración.
Cabe mencionar que el rectángulo que está representando al problema integrado, está respaldado por el **modelo-ER** correspondiente, es decir, este **rectángulo con la palabra clustered** representa un análisis general que se abordó y dio solución.

Cuando se agrupan entidades, mucha veces como diseñador se encuentran con agrupaciones que están anidadas dentro de otra agrupación, es decir, contenidas unas dentro de otras, para este caso se colocará al lado del nombre de la agrupación un número encerrado entre paréntesis, el cual indicará el nivel de anidamiento contenido en la agrupación, por ejemplo, si no se encuentra un número entre paréntesis al final del nombre de la agrupación, esto indicará que esa agrupación es la primera que se realiza con el problema analizado. Si se encuentra un (1), indicará que el nivel de anidamiento es uno y por lo tanto tiene una agrupación dentro de la misma. Si se encuentra un (2), este indicará que la agrupación tiene dos niveles de agrupación dentro de la misma.

En los siguientes ejemplos se aplicará la teoría descrita anteriormente. Para poder lograr esto mostraré las siguientes figuras que representan una colección de entidades y relaciones en un objeto de alto nivel de abstracción. Estas figuras son una innovación para este modelo y fueron tomadas del libro **[Toby J. Teorey , 65]** , las cuales usarán cuando se presenten los tres casos mostrados en la siguiente figura 2.10 .

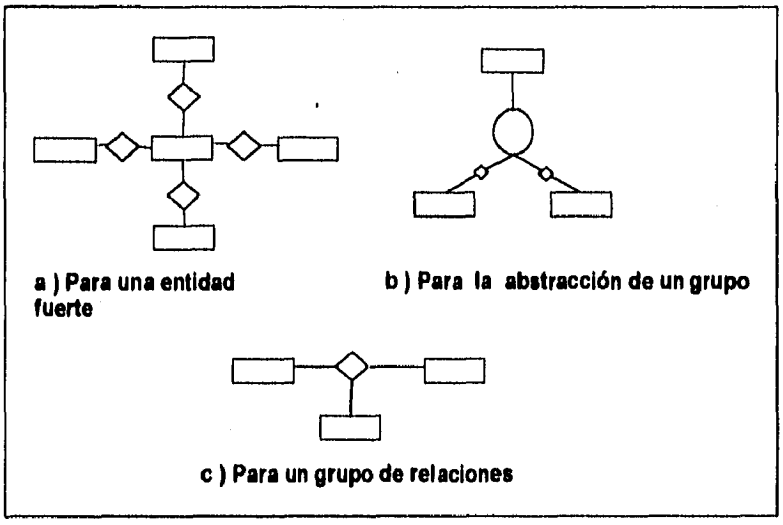


Figura 2.10 Formas de agrupar entidades (entidades CLUSTERED).

Para ejemplificar la agrupación de entidades, es decir **entidades clustered** , se retoma el ejemplo de la **figura 2.9** , donde se aplica la teoría descrita anteriormente y que se mostrará en la **figura 2.11 .**

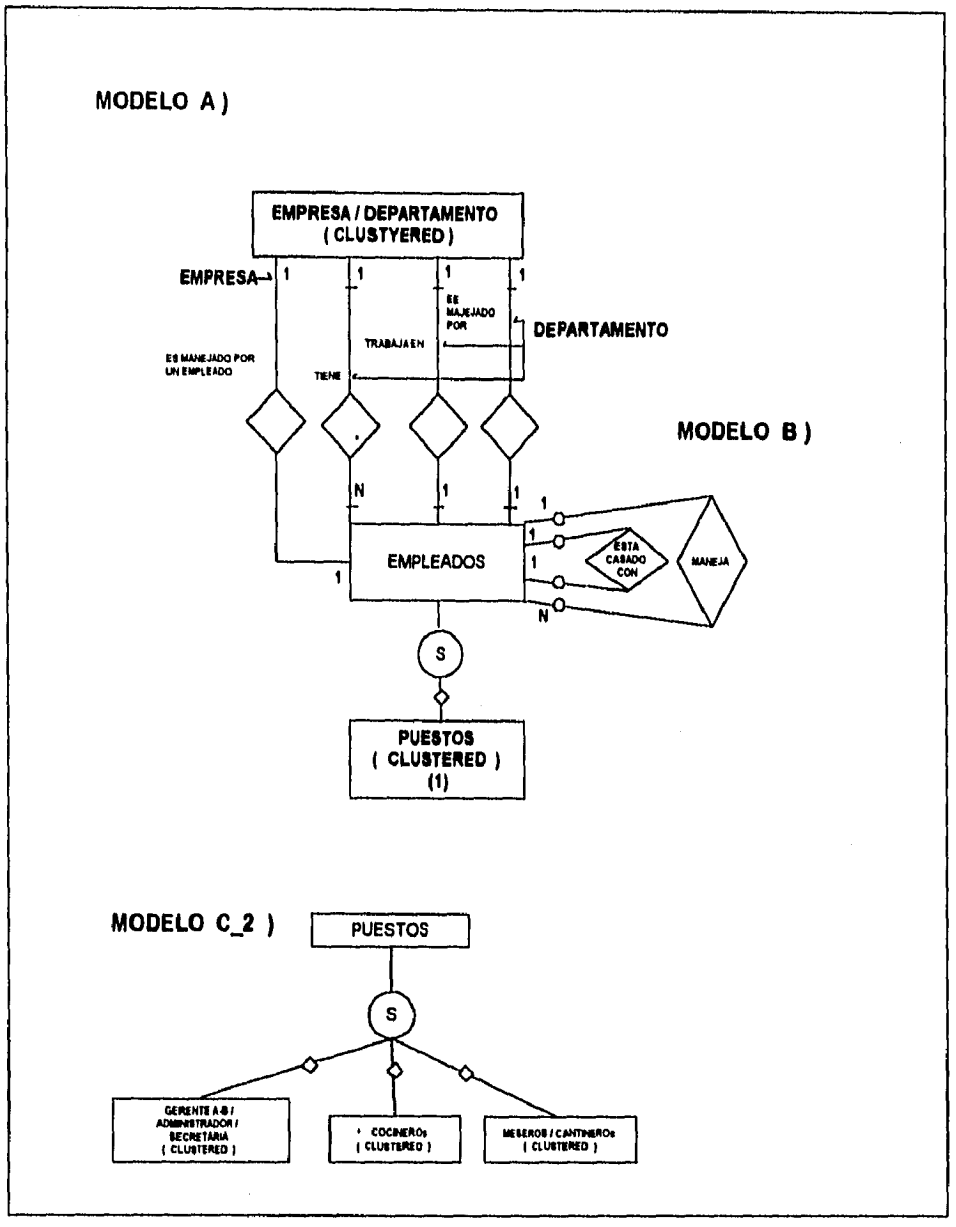


Figura 2.11 Agrupación de entidades (entidades CLUSTERED).

En el caso de la figura anterior, se observa que la relación binaria que hay entre la entidad empresa y departamentos se puede representar en una **entidad** tipo **clustered,** llamada **[empresa/departamento CLUSTERED]** , conservando su conectividad y existencias respectivas. Como la entidad **clustered** abarca un solo nivel de abstracción, no se coloca ningún número entre paréntesis, para indicar que tiene un solo nivel de abstracción; también se puede ver que en la entidad puestos se pueden agrupar las entidades virtuales gerente A-B, administrador y secretaria en una tipo **clustered,** con un nivel de abstracción y ésta a su vez en otra tipo clustered llamada **[puestos CLUSTERED]** , la cual tiene un nivel de abstracción uno, indicado entre paréntesis.

En general, la teoría descrita hasta este momento proporciona las herramientas para poder construir el diseño lógico de un problema determinado.
Los problemas descritos anteriormente del **modelo-ER** , así como la total integración del diseño lógico de la **BD** se muestran en el **apéndice b).** Nota 11)

Resumen

En esta parte del capítulo se resalta la importancia del análisis de requerimientos y se proponen pasos para llevar a cabo el diseño lógico, tales como identificar entidades y atributos, siguiendo la identificación de generalizaciones de hierarchy, así como el hecho de definir relaciones. También se mencionaron las ventajas de utilizar el **modelo-ER**, ya que la sintaxis entre las gráficas hace más entendible la representación de las transacciones y los diferentes rolles de las entidades dentro de la empresa, así como la integración de vistas. También se explicaron los pasos en la integración de vistas, así como la agrupación de entidades.

Nota 11)

En la integración que se presenta en el **apéndice b)** no se darán detalles de cómo se realizaron las abstracciones correspondientes, como se especificó en los anteriores ejemplos.

II.C Transformación del modelo-ER en esquemas de tablas

En esta parte del trabajo, para la forma como se desarrolla el diseño lógico de la **BD** y la transformación en tablas del **modelo-ER**, utilizando un **SMBD** y como **la mejor alternativa, como primer paso se propone crear esquemas de tablas, es decir, tablas que no han sido transformadas utilizando un SMBD**. Estos esquemas de tablas, por el momento, únicamente contendrán atributos identificadores, los cuales ayudarán a realizar el diseño lógico de la **BD** y todas las abstracciones que éste involucra. Para poder realizar esta tarea es necesario estar verdaderamente involucrado con el diseño lógico y conocer muy bien los objetos primitivos que se describieron en el análisis, tales como entidades, y descubrir sus atributos identificadores que ayudarán a relacionar las entidades.

Los esquemas de tablas en esta parte del trabajo sólo contendrán identificadores, más adelante, en el capítulo **III**, se normalizará la **BD**. En esta etapa los esquemas de tablas serán analizados con todos sus atributos, sean estos identificadores, descriptores y multivaluados. En ese momento, cuando la **BD** es normalizada, además de estar libre de inconsistencias y redundancias, es conveniente transformar los **esquemas de tablas** en tablas y las relaciones en vistas, utilizando **Sybase,** ya que éste permite la **definición** y **manipulación** de datos, de acuerdo a las necesidades, además de ser uno de los manejadores de **BD** más utilizados actualmente en México.

La transformación se llevará a cabo de la siguiente forma : se mostrará la sintaxis de un determinado comando y se ejemplificará cada uno de los pasos de transformación en este manejador. Esto se verá con más a detalle en el **apéndice e)** , donde se muestra la transformación del diseño lógico en tablas, utilizando **Sybase.**

II.C.1 Llaves

Tomando en cuenta lo descrito anteriormente, se deben considerar las siguientes reglas y definiciones de **llaves**, para tener bases sólidas y un mejor entendimiento al momento de transformar el **modelo-ER** en esquemas de tablas y relaciones (vistas).

[4] **Las llaves** son construcciones lógicas, no tienen parte física en la **BD.** Una variación de éstas son las **llaves compuesta,** formadas por la unión de varias columnas.
Otro tipo de llave se conoce como **super llave,** ésta es una llave que permitirá identificar una entidad de otras. Este identificador se tendrá por la combinación de columnas de dicha entidad. Por ejemplo, cuando se tiene el nombre de un cliente existe la posibilidad de que otro tenga el mismo nombre, por esto se debe buscar una combinación de columnas que no permita la duplicidad del nombre del cliente.
Otro tipo es la **llave candidata,** ésta es la combinación de atributos que no pueden ser **superllaves.**
La transformación de esquemas de tablas involucra los siguientes tipos de llaves :

[4] Las definiciones de llaves se tomaron del libro [Henry F. Korth, 33-36]

1) Llave primaria
El primero es conocido como **llave primaria (PK)**, la cual es un identificador único, no debe contener valores nulos; además de servir para relacionar entidades y dar origen a relaciones, puede estar formada por una o más columnas que referirán a otra llave primaria.

2) Llave foránea
El segundo tipo de llave es conocido como **llave foránea (FK)**, la cual es una llave que juega dos papeles ; en una tabla puede ser **llave foránea (FK)** y en otra **llave primaria (PK).**

También se debe considerar a los **discriminadores** , identificadores únicos que se crean al realizar una operación.

II.C.2 Valores que pueden tomar los atributos

Ahora se deben considerar tres reglas para valores que pueden tomar los atributos al formar esquemas de tablas y vistas.

1) Valores nulos son permitidos en valores de atributos pertenecientes a una entidad.
2) Valores nulos no son permitidos en una tabla para una **llave foránea** asociada a una **entidad mandatoria**.
3) Valores nulos no son permitidos en **llaves primarias y foráneas** en una relación, ya que sólo atributos que tengan valor pueden ser relacionados.

Después de tener presentes los anteriores valores, se proponen tres pasos para **transformar entidades en esquemas de tablas y relaciones en vistas,** éstos son :

II.C.3 Pasos para transformar entidades en esquemas

1) Transformar cada entidad en esquemas de tablas, éstas deben contener sólo atributos llaves, los no llaves se integran a los esquemas cuando se normaliza la **BD**.
2) Transformar las relaciones binarias muchos a muchos o relaciones binarias recursivas en esquemas de tablas de relaciones con sus respectivas llaves de entidades. Estas transformaciones de relaciones también son conocidas como creación de vistas, que solucionarán los problemas planteados en el análisis.
3) Transformar cada relación ternaria o n-aria en un esquema de tabla con sus respectivos atributos llaves. [Nota 12)]

A continuación se presentan cuatro ejemplos generales, carentes de conectividad y existencia, con los que uno se puede encontrar al momento de transformar los esquemas en tablas.

Nota 12)

Los atributos descriptivos se integrarán a estos esquemas cuando de normalice la BD.

Para el mejor entendimiento de estas transformaciones, a los atributos identificadores se les conocerá simplemente como identificadores.

II.C.4 Casos al transformar los esquemas

Los siguientes casos son aquellos con los que constantemente uno como diseñador se puede encontrar

1) Relaciones binarias recursivas
Esta relación contendrá dos identificadores, uno será **PK** y el otro un identificador que lo relacionará con una ocurrencia dentro de la misma entidad.

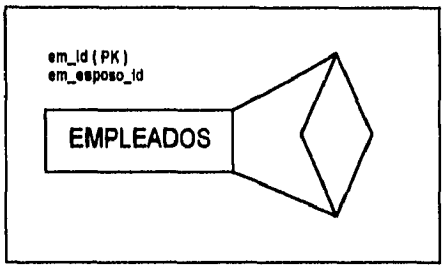


Figura 2.12 Relación binaria recursiva.

Aquí se puede ver que **em_id, es PK** , que identifica al empleado y **em_esposo_id** es un identificador que sirve para relacionarlo con otro empleado en una ocurrencia determinada.

2) Relaciones binarias
Cuando se inicie la transformación ,en tablas de entidades que están modeladas en relaciones binarias. Estas tendrán N-1 variaciones de conectividad con la entidad que se está relacionando, además las tablas tendrán una llave primaria **(PK)** y su correspondiente llave foránea **(FK)** , la cual en una entidad será **PK** y en la otra **FK**, además de los atributos descriptivos y multivaluados, que serán integrados al momento de normalizar la **BD**.

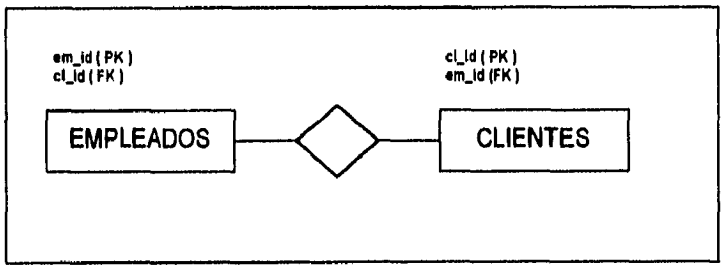


Figura 2.13 Relación binaria.

En esta relación se puede observar, para el caso de la entidad empleados , que el identificador **em_id** es **PK** y el otro identificador **cl_id** es **FK** ; si se observa a la inversa la entidad clientes, el identificador **cl_id** es **PK** y el identificador **em_id** es **FK**.

3) Relaciones ternarias

Cuando se presenten **modelos-ER** que involucren tres entidades, tienen N-1 variaciones de conectividad en la entidad fuerte, donde N es el número de entidades involucradas en esta relación; la cual contendrá tres identificadores, uno de los cuales será **PK** y los otros dos serán **FKs**. Respecto a las otras dos entidades involucradas en la relación ternaria, contendrán una **FK** relacionada a la **PK** de la entidad fuerte.

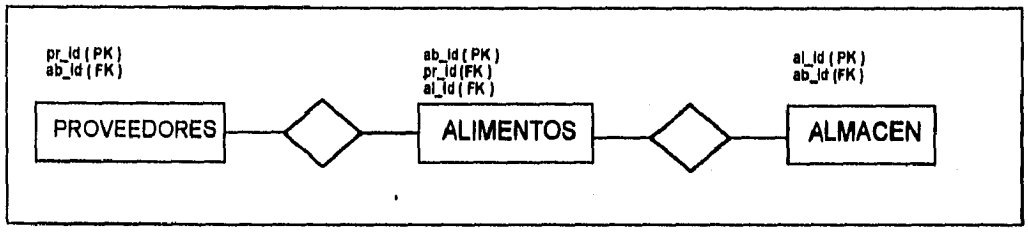


Figura 2.14 Relación ternaria.

En la figura anterior, para la entidad alimentos el identificador **ab_id** es **PK** y los otros dos identificadores son **llaves foráneas (FKs)**. Para la entidad almacen **al_id** es **PK** y **ab_id** es **FK** que la relaciona con la entidad fuerte. el mismo criterio se sigue para proveedores.

Por entidad fuerte se entenderá aquella entidad que se tiene como foco de la relación.

4) Relaciones n-arias

Las relaciones **n - arias** también contendrán N-1 variaciones de conectividad en la entidad fuerte. Para este caso, cuando más de tres entidades estén contenidas en un **modelo-ER**, la entidad fuerte contendrá un número igual de identificadores al de las entidades que se están relacionando, es decir, si se tienen cuatro entidades involucradas, se deberán tener cuatro identificadores en la entidad fuerte, donde un identificador será **PK** y los otros tres identificadores serán **FKs**. Las demás entidades involucradas contendrán una **FK** que la relacionará con la entidad fuerte, además de su **PK** .

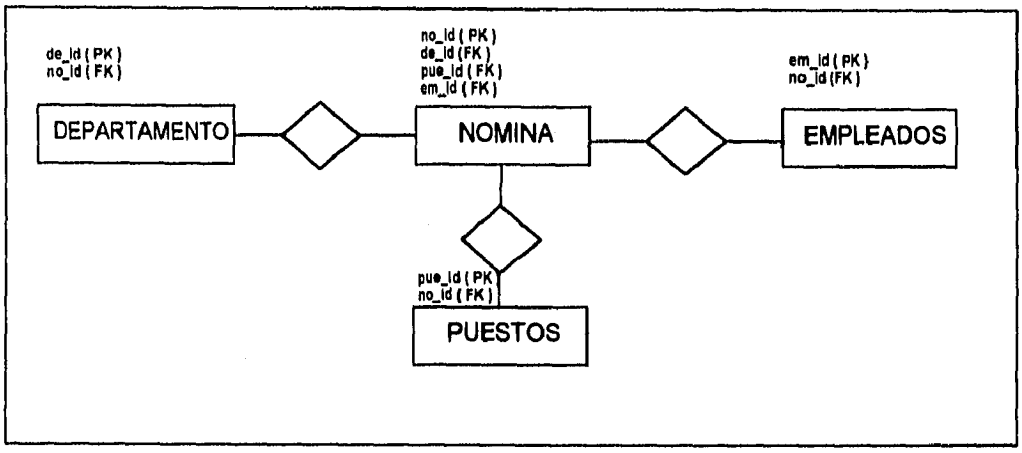


Figura 2.15 Relación n-aria.

En la anterior figura se puede ver que en la entidad nómina, el identificador **no_id** es **PK** y los otros identificadores son **FKs** , para esta entidad.
Estos son los casos que se pueden encontrar en cuanto a relaciones se refiere, al momento de transformar el **modelo-ER** a tablas.

Para poder ejemplificar la transformación del modelo en tablas, únicamente se muestra la transformación de identificadores, dejando para el **apéndice e)** la transformación completa del diseño lógico de la **BD** utilizando **Sybase**.

Una vez que se comprendieron los cuatro ejemplos generales que se pueden presentar en esquemas de relaciones, se mostrarán los cuatro tipos de relaciones con sus correspondientes cardinalidades y existencias que se pueden encontrar cuando se realiza el diseño lógico de una **BD**.

Una aportación importante de las siguientes figuras es el hecho de que muestran, como una introducción, la transformación de identificadores en tablas, que como se mencionó anteriormente, se mostrará más adelante, pero que en esta etapa del trabajo es muy didáctico ver cómo evoluciona la transformación conforme avanza el diseño, para los posibles casos en que uno como diseñador se pueda encontrar a través del ciclo de vida de una **BD**.

II.C.5 Casos para relaciones binarias recursivas

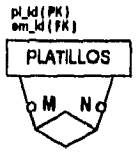
INTERPRETACIÓN A LA CARDINALIDAD.	MODELO-ER	TRANSFORMACIÓN DEL MODELO-ER EN TABLAS.
1) UN DEPARTAMENTO ES MANEJADO POR UN EMPLEADO. 2) UN EMPLEADO MANEJA SOLO UN DEPARTAMENTO RELACIÓN UNO A UNO	**RELACION : ES MANEJADO POR**  	CREATE TABLE **empleados** (em_id char(4) not null, de_id char(4) not null)
1) UN EMPLEADO TIENE A SU CARGO N EMPLEADOS. RELACIÓN UNO A VARIOS	**RELACION : TIENE A SU CARGO**  	CREATE TABLE **empleados** (em_id char(4) not null, em_depende_id char(4) not null)
1) UNO O MAS EMPLEADOS, SON ASIGNADOS A UNO O MAS PLATILLOS. RELACIÓN VARIOS A VARIOS	**RELACION : SON ASIGNADOS.**  	CREATE TABLE **platillos** (pl_id char(4) not null, em_id char(4) not null)

FIGURA 2.16 CASOS DE TRANSFORMACIÓN EN RELACIONES BINARIAS

La figura anterior muestra los posibles casos para una relación binaria recursiva, donde se puede ver que la cardinalidad y existencia se interpretan de la siguiente forma :

Para el roll " **es manejado por** " , se tiene una **cardinalidad** de uno a uno, debido a que sólo un empleado puede manejar sólo un departamento. Respecto a su **existencia se puede ver** que esta entidad para este roll es opcional, debido a que un empleado puede o no manejar un departamento.

En el roll " **tiene a su cargo** " se tiene una **cardinalidad** de uno a varios, ya que un empleado tiene a su cargo uno o más empleados. Aquí se observa que la **existencia** de esta entidad mandatoria, ya que un empleado forzosamente da las órdenes a otro empleado, también es opcional, ya que un empleado puede o no depender de las órdenes de otro empleado, como en el caso del gerente de A-B que solamente recibe órdenes del dueño de la empresa.

En el roll " **son asignados** " , se tiene una **cardinalidad** de varios a varios, ya que varios platillos pueden ser asignados para su preparación a varios empleados; para este roll, la existencia de esta entidad es opcional, ya que los platillos pueden ser asignados a uno o más empleados.

II.C.6 Casos para relaciones binarias

INTERPRETACIÓN A LA CARDINALIDAD	MODELO-ER	TRANSFORMACIÓN DEL MODELO-ER EN TABLAS
1) UN DEPARTAMENTO ES MANEJADO POR UN EMPLEADO. 2) UN EMPLEADO MANEJA SOLO UN DEPARTAMENTO. RELACIÓN BINARIA UNO A UNO	**RELACION : ES MANEJADO POR** de_id (PK) em_id (FK) em_id (PK) de_id (FK) 1 1 DEPARTAMENTO — EMPLEADOS	CREATE TABLE **departamento** (de_id char(6) not null , em_id char(6) not null) CREATE TABLE **empleados** (em_id char(6) not null , de_id char(6) not null)
1) CADA DEPARTAMENTO TIENE UN EMPLEADO QUE LO MANEJA, PERO UN EMPLEADO PUEDE MANEJAR MAS DE UN DEPARTAMENTO. RELACIÓN BINARIA UNO A UNO	**RELACIÓN : ES MANEJADO POR** de_id (PK) em_id (FK) em_id (PK) de_id (FK) 1 1 DEPARTAMENTO — EMPLEADOS	CREATE TABLE **departamento** (de_id char(6) not null , am_id char(6) not null) CREATE TABLE **empleados** (em_id char(6) not null , de_id char(6) not null)
1) ALGUNOS ACCESORIOS SON PERMITIDOS PARA USO DE LOS COCINEROS, PERO NO NECESARIAMENTE PARA TODOS LOS COCINEROS. RELACIÓN BINARIA UNO A UNO	**RELACIÓN : TIENE PERMITIDO** em_id (PK) ac_id (FK) ac_id (PK) em_id (FK) 1 1 EMPLEADOS — ACCESORIOS	CREATE TABLE **empleados** (em_id char(6) not null , ac_id char(6) not null) CREATE TABLE **accesorios** (ac_id char(6) not null , em_id char(6) not null)
1) CADA EMPLEADO TRABAJA EXACTAMENTE EN UN DEPARTAMENTO Y CADA DEPARTAMENTO TIENE ALMENOS UN EMPLEADO RELACIÓN BINARIA UNO A VARIOS	**RELACIÓN : TIENE** de_id (PK) em_id (FK) em_id (PK) de_id (FK) 1 N DEPARTAMENTO — EMPLEADOS	CREATE TABLE **departamento** (de_id char(6) not null , em_id char(6) not null) CREATE TABLE **empleados** (em_id char(6) not null , de_id char(6) not null)
1) CADA DEPARTAMENTO ENTREGA UNO O MAS REPORTES PERO UN REPORTE NO NECESARIAMENTE ES ENTREGADO POR UN DEPARTAMENTO. RELACIÓN BINARIA UNO A VARIOS	**RELACIÓN : ENTREGA REPORTES** de_id (PK) re_id (FK) re_id (PK) de_id (FK) 1 N DEPARTAMENTO — REPORTES	CREATE TABLE **departamento** (de_id char(6) not null , re_id char(6) not null) CREATE TABLE **reportes** (re_id char(6) not null , de_id char(6) not null)
1) LOS CLIENTES QUE SON CONSTANTES EN SU CONSUMO PERTENECEN AL CLUB DORADOS, Y ESTE CLUB PUEDE TENER O NO CLIENTES PERTENECIENTES A EL. RELACIÓN BINARIA VARIOS A VARIOS	**RELACIÓN : PERTENECE A** cl_id (PK) do_id (FK) do_id (PK) cl_id (FK) M N CLIENTES — DORADO	CREATE TABLE **clientes** (cl_id char(6) not null , do_id char(6) not null) CREATE TABLE **dorados** (do_id char(6) not null , cl_id char(6) not null)

Figura 2.17 Casos de transformación en relaciones binarias.

La figura 2.17 muestra los posibles casos para una relación binaria, de los cuales se pueden interpretar las existencias de la siguiente manera :
En el roll " **es manejado por** " , la existencia de las entidades departamento y empleados es mandatoria, por lo tanto deben de existir.
En el roll " **tiene permitido** " , las entidades empleados y accesorios son opcionales.

En el roll " **tiene** " , las entidades departamento y empleados son mandatorias y deben de existir.
Para el roll " **entregn reportes** " , la entidad reportes es indefinida y la entidad departamento es opcional.
En el roll " **pertenece a** " , la entidad clientes y dorados son opcionales.
Respecto a la interpretación de **cardinalidades** se puede leer en la primer columna de la figura 2.17 .

II.C.7 Casos para relaciones ternarias

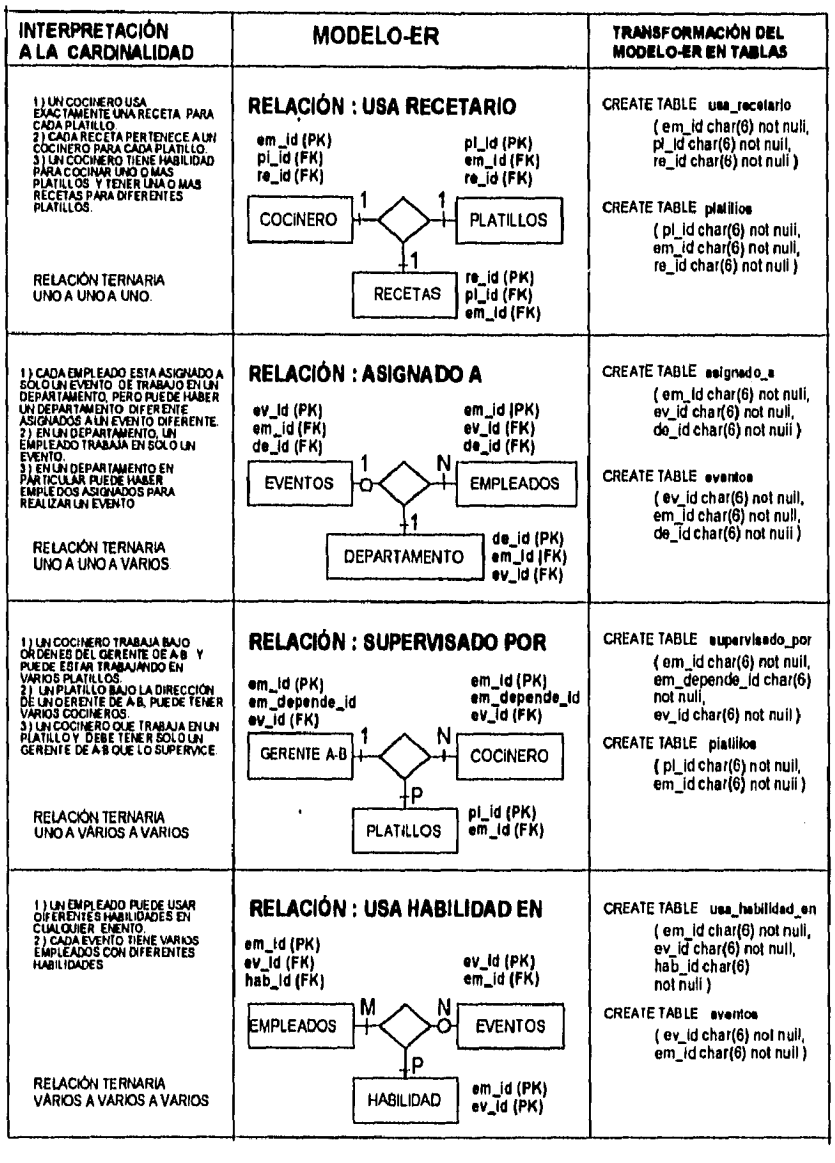


Figura 2.18 Casos de transformación en relaciones ternarias.

La anterior figura muestran los posibles casos para una relación ternaria, de las cuales su existencia se interpreta de la siguiente forma :

En el caso del roll " **usa recetario** " , las tres entidades involucradas deben de existir, por lo tanto son mandatorias.
En el roll " **asignado a** " , las entidades empleados y departamentos son mandatorias y deben existir, la entidad eventos es opcional debido a que el restaurante puede o no tener eventos.
En el roll " **supervisado por** " , las entidades virtuales gerente A-B y cocineros, están contenidas dentro de la entidad empleados, la cual debe existir, también la entidad platillos debe existir por lo tanto son mandatorias.
En el roll " **usa habilidad en** " , la entidad virtual habilidad está contenida dentro de la entidad empleados, éstas deben existir, por lo tanto son mandatorias. Respecto a la entidad eventos, es opcional debido a que el restaurante puede o no tener un evento en una fecha determinada.
Respecto a la interpretación de la carnalidad, se puede ver en la primer columna de la figura 2.18 .

II.C.8 Casos para relaciones n-arias

INTERPRETACIÓN A LA CARDINALIDAD	MODELO-ER	TRANSFORMACIÓN DEL MODELO-ER EN TABLAS
1) UN DEPARTAMENTO PERTENECE A UNA NOMINA. 2) N DEPARTAMENTOS PERTENECEN EXACTAMENTE A UNA NOMINA. 1) CADA PUESTO ES REGISTRADO EN UNA NOMINA. 2) UNA NOMINA REGISTRA N PUESTOS. 1) UNA NOMINA REGISTRA N EMPLEADOS. 2) N EMPLEADOS ESTAN REGISTRADOS EN UNA NOMINA.	RELACIÓN : NOMINA NOMINA (CLUSTERED)	CREATE TABLE nomina (no_id char(6) not null, de_id char(6) not null, pue_id char(6) not null, em_id char(6) not null) CREATE TABLE puestos (pu_id char(6) not null, no_id char(6) not null)

Figura 2.19 a) Caso de transformación en una relación n-aria.

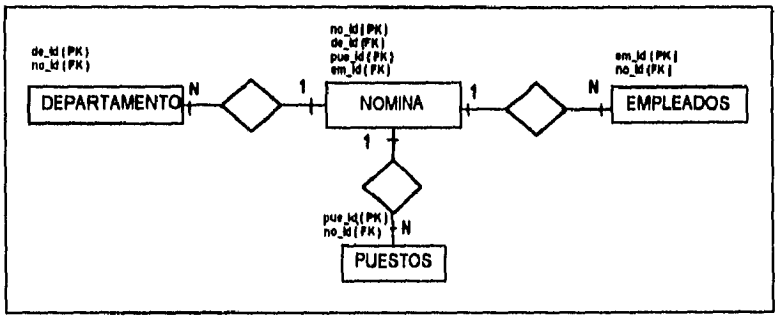


Figura 2.19 b) Agrupación de la entidad Nomina (CLUSTERED).

La anterior figura muestra los posibles casos para una relación n-aria, de la cual, respecto a su **existencia**, se puede ver que una nómina depende de la existencia de las entidades empleados,

puestos y departamentos, por lo tanto, estas entidades forzosamente deben existir para que exista la nómina.

Respecto a la interpretación a la **cardinalidad**, se puede ver en la primer columna de la gráfica anterior.

Una aportación importante de estas gráficas es que muestran la interpretación a la cardinalidad y existencia de las relaciones.
Los esquemas de entidades y relaciones serán mostrados implícitamente en el **modelo-ER** que éste represente, siguiendo la propuesta hecha en este capítulo en su parte de **II.A** , en la cual se sugiere que los atributos involucrados en una relación serán mostrados en la parte superior de la entidad que describan.

Para la mejor comprensión de los esquemas de entidades, los atributos llaves se mostrarán en la parte superior de la entidad que identifiquen. Cuando se encuentre esta forma de agruparlos, se debe asociar inmediatamente que esa entidad, con sus atributos en la parte superior, representa un esquema de una entidad, antes de que sea transformada en tablas o vistas.

Como parte adicional en estas figuras, se mostró como adelanto cómo se transforman estos esquemas de tablas en tablas, utilizando **Sybase**.
Cabe mencionar que esta transformación se muestra con más detalle en el capítulo IV y en el **apéndice e).**

Por ejemplo, para el caso de la relación **" es manejado por "** , se utiliza el comando **create table**, el cual **sirve para crear tablas,** donde el comando create table **crea la tabla empleados** con sus correspondientes identificadores **em_id , de_id tipo char de cuatro posiciones y que éstos no acepten valores nulos**.
Como se puede ver representado en la **figura 2.16** , en casos de transformación binaria recursiva.

Resumen
En este capítulo se presentaron los diferentes casos que uno como diseñador se puede encontrar cuando transforma el **modelo-ER** en esquemas de tablas, así como su carnalidad y cómo se transforman éstos en tablas.

Capítulo III

Normalización

CAPÍTULO III
NORMALIZACIÓN

En este capítulo se normalizará la **BD**, con esto se procura tener los esquemas de las tablas libres de redundancias e inconsistencias, es decir, tener tablas con todas las ventajas y beneficios que se mencionaron en el capítulo I ; también se considera este capítulo de vital importancia para el ciclo de vida de una **BD**, ya que de esto depende que la manipulación de la **BD** se realice de una manera óptima, libre de anomalías en la representación de los datos, tales como inconsistencias y redundancias. Desde mi punto de vista, es el principal motivo por el que un diseñador elige una **BD** con todos los pasos que ésta implica en su ciclo de vida para implementar un sistema. En este capítulo se conocerán **las cinco formas normales** con las que se puede normalizar una **BD**.
El objetivo de una **BD** es generar un conjunto de esquemas relacionales, que como se mencionó anteriormente, permitan almacenar información sin redundancia, así como recuperarla fácilmente, esto se logrará diseñando un esquema que tenga una **forma normal**, utilizando **Dependencias Funcionales (DFs), Dependencias Funcionales Multivaluadas (DFMs) y Dependencias de Unión (DUs).** Estas dependencias se definirán más adelante.
A continuación se mencionarán las **tres desventajas** en las que se puede caer si se diseña mal una **BD** :

III.1 Desventajas del mal diseño de una BD

1) Repetición de información

Es la primer desventaja y se caracteriza por el desperdicio de espacio, que complica las actualizaciones a la **BD**, ya que se deben actualizar más tablas donde se encuentre la información repetida.

2) Incapacidad para representar cierta información

Esta desventaja se provoca debido a que no se realizó un diseño lógico relacional de forma óptima y por lo tanto se omitieron ciertas relaciones importantes, que en este caso permitirán relacionar ciertas tablas y extraer datos de salida para proporcionar la información en una instancia determinada.

3) Pérdida de información

Para evitar la pérdida de información se debe descomponer un esquema con muchos atributos, en varios esquemas pequeños con pocos atributos, utilizando el diseño descendente, de tal manera que se proyecte el problema general, difícil de solucionar en una primera instancia, a pequeños problemas que formen parte del esquema general, de tal forma que la visión y solución de estos pequeños problemas sea más fácil para el diseñador.

Para poder prevenir los tres puntos anteriores se deben tomar en cuenta las siguientes definiciones, antes de pasar a normalizar la **BD**.

III.2 Prevención de un mal diseño de la BD

Definición

[5] **Descomposición de productos sin pérdida**

Sea **R** un esquema de relaciones de dos o más entidades y **F** un conjunto de dependencias funcionales en **R**, pertenecientes a las entidades que se están relacionando. **R1 y R2** forman una descomposición de productos sin pérdida de **R**, si por lo menos una de las dependencias funcionales está en **F**, de tal forma que :

R1 ^ R2 -> R1

R1 ^ R2 -> R2

De la anterior definición se deduce que se puede descomponer una relación sin pérdida de información, siempre y cuando exista intersección en un atributo perteneciente a ambas relaciones ; este atributo en una tabla será **PK** y en otra una **FK** que relacione las dos entidades y que de esta forma se pueda accesar a la información que contienen ambas entidades.

Por ejemplo, en la **figura 3.1** , el atributo **d**, en la **DF** de **a** es una **FK** y en la **DF** de **d** es una **PK**; de tal forma que esta llave sirve para relacionar estas dos entidades sin que se encuentre pérdida de información.

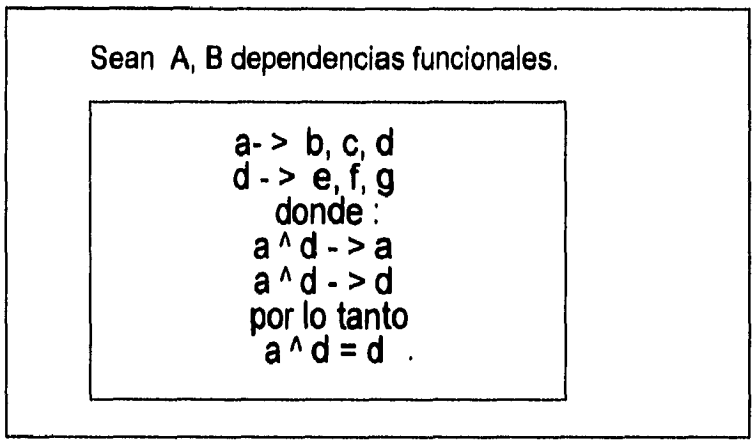


Figura 3.1 Descomposición de productos sin pérdida.

Otro aspecto muy importante, es conocido como **la conservación de dependencias,** ya que si se realiza una actualización a la **BD,** el sistema debe probar que no se haya creado una relación ilegal, es decir, una que no satisfaga todas las **DFs** dadas, además de probar que las actualizaciones, altas y bajas, se realicen en todas las tablas involucradas, en sus respectivas **DFs**. Para probar la manipulación eficiente de los datos es conveniente diseñar esquemas de **BD** relacionales, que permitan validar una actualización, es decir, crear programas que permitan validar, por ejemplo, que los datos estén relacionados y que por algún motivo se modifiquen, las actualizaciones se realicen en todas las tablas involucradas y que además los datos sean del mismo tipo, ya que si el sistema permite que se relacionen datos de diferente tipo, conforme se actualice información perderá su consistencia; ahora se explicará a dar la siguiente definición que permitirá comprender mejor la conservación de dependencias.

[5] Esta definición se tomo del libro [Henry F. Korth, 192]

Definición

[6] Conservación de dependencias

Sea **F** un conjunto de **DFs** en el esquema general de relaciones en la **BD** representada por **R,** y sea **R1, R2... Rn** una descomposición del esquema general **R,** es decir un conjunto de relaciones; aquí se van a tener varias restricción de **F** a **Ri,** debido a que los atributos pertenecientes a **Ri** se encuentran relacionados con atributos pertenecientes a otros esquemas de tablas, tales como **R1, R2, ... Rn**, donde **Ri** será un esquema de una tabla cualquiera en el conjunto de todas las **DFs** en **F,** que incluyen únicamente atributos de **Ri**.

Para poder entender la anterior definición se debe entender que **F** es un conjunto **DFs** , con sus correspondientes atributos, conservando sus **DFs** con sus **PKs,** donde al actualizar o realizar la alta o baja de un de registro, se debe verificar que los datos que se manipulan sean del mismo tipo y que además la manipulación afecte todos los registros relacionados en las diferentes tablas, la anterior definición se puede ver representada en la siguiente figura, que muestra la conservación de dependencias.

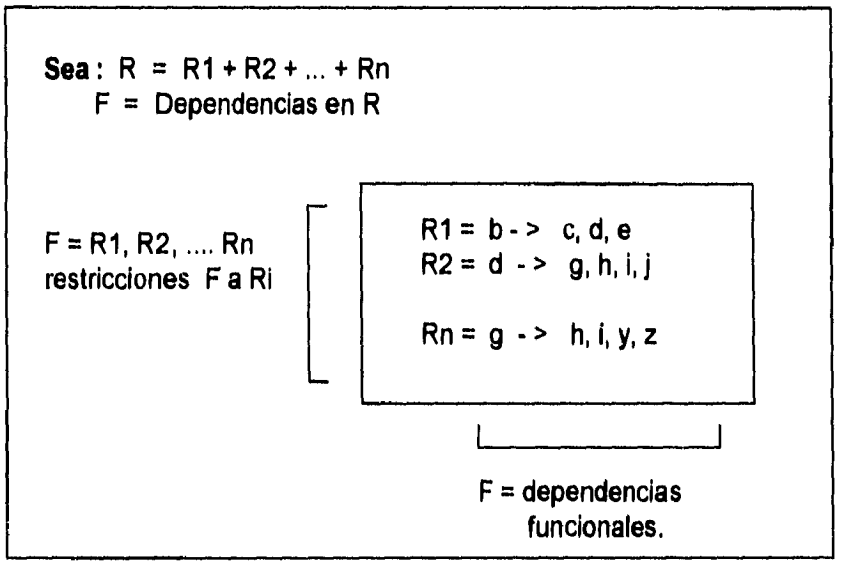


Figura 3.2 Conservación de dependencias.

La anterior figura se interpreta de la siguiente forma : **R1, R2, ... Rn** son un conjunto de **DFs**, de tal manera que si actualizo los atributos dependientes de la **PK d** , en la relación **R2** se deberá checar que al actualizar los atributos relacionados con la **PK**, en este caso es **d**, se actualicen y se verifiquen las dependencias de esta relación; este mismo criterio se debe seguir según sea el caso de la relación que se tome para ejemplificar la conservación de dependencias.

Una vez que se comprendió la teoría descrita anteriormente, se recordarán que las tablas son resultado del diseño lógico de una **BD**, las cuales se derivaron del **modelo-ER**. Suponiendo que se colocara toda la información de una empresa en una sola tabla, como son todos los

[6] Esta definición se tomo del libro [Henry F. Korth, 193]

departamentos, esto causaría que el costo de manipular dicha tabla fuera muy alto, además de redundancia en la información.
Si se quisiera manipular dicha tabla podría afectar registros que no se desean y por tanto perder información valiosa para la empresa. Para evitar este tipo de problemas se **divide la información en diferentes áreas, identificándolas como únicas llamadas entidades**, conteniendo **atributos** para describir el área a la que pertenecen.

La aportación del anterior párrafo, como ya se mencionó, es utilizar la metodología del diseño descendente, debido a que se pueden analizar mejor pequeños problemas pertenecientes al esquema general, de tal forma que una vez solucionados y analizados, se integren e implementen las tablas, libres de inconsistencias y redundancias, conservando tanto sus **DFs** como las **DFMs**.

La figura 3.3 a) representa una tabla sin normalizar que se presenta parte de la información de la empresa (**almacén**), donde se puede ver que hay atributos repetidos, como son **uso_id, al_fsalida, al_existencias y al_fcaducidad.**

almacen

al-id	al-descripcion	pr-cia	ab-id
A00001	VINO, BLANCO	OXO S.A. DE C.V.	A00001
B00002	QUESO, CHI	COMERCIAL S.A. DE C.V.	B00001
C00003	LECHE, DESCR	ALPURA S.A. DE C.V.	C00001
D00004	CARNE, RES	COMERCIAL S.A. DE C.V.	D00001

al-cminima	al-cmaxima	al-existencias	al-fcaducidad
35	130	50	10-05-1995 12:00AM
5	10	6	05-06-1995 12:00AM
25	40	35	30-07-1995 12:00AM
25	50	35	30-07-1995 12:00AM

al-fingreso	pr-id	tipo-id	ab-nombre
12-04-1995 12:00AM	P00001	T1	VINO TINTO
04-05-1995 12:00AM	P00002	T2	QUESO, CHI
15-07-1995 12:00AM	P00003	T2	LECHE DESCR
15-07-1995 12:00AM	P00004	T2	CARNE, RES

al-fsalida	uso-id	al-existencias	al-fcaducidad
25-05-1995 12:00AM	U01	50	10-05-1995 12:00AM
30-05-1995 12:00AM	U01	6	05-06-1995 12:00AM
17-07-1995 12:00AM	U02	35	30-07-1995 12:00AM
17-05-1995 12:00AM	U02	35	30-07-1995 12:00AM

pr-nombre	pr-ciudad	pr-direccion	pr-telefono
CARMEN ALCANTARA	TOLUCA, MEX	AV. CEYLAN 123, ANAHUAC	310-25-81
MANUEL ALVAREZ	D.F. , MEX	GRANADOS 35, POLANCO	555-67-78
ZORAIDA GONZALEZ	GUADALAJARA, JAL	VALLARTA 34, PALMAS	23-45-67
JESUS ARENAS	MOCHIS, SINALOA	OLVERA 456, NUEVA	34-78-98

al-fsalida	uso-id
25-05-1995 12:00AM	U01
30-05-1995 12:00AM	U01
17-07-1995 12:00AM	U02
17-05-1995 12:00AM	U02

Figura 3.3 a) Tabla sin normalizar.

Una vez comprendidos los anteriores conceptos y para evitar los problemas antes mencionados, se procede a **Normalizar la BD.**

III.3 Normalización

Definición

Por **normalización** se entenderá la creación de una forma normal de la **BD**, partiendo de analizar cada uno de los atributos que conforman las tablas y buscando relaciones (óptimas) entre éstos, además de identificar el **dominio** de cada atributo, es decir, todos los posibles valores que pueden tomar dichos atributos. Para lograr esto se retomarán conceptos de **llaves** definidos en el capítulo anterior, que, se utilizarán durante la normalización de tablas.

Definición

[7] Primera forma normal (1FN)

Una tabla toma la **primera forma normal,** si y sólo si todas las columnas constan sólo de **valores únicos**, es decir, que no se repitan nombres de atributos en todas las tablas ni tampoco en si mismas. Cuando no son únicos los atributos puede ser que se encuentren con la misma definición y con el mismo nombre en otra tabla, este problema causaría redundancia en la información.

La ventaja de la **1FN** es que no permite la redundancia y al momento de realizar una consulta **(QRY)**, no se confunde el **LMD** al seleccionar atributos para su consulta, una ventaja de la **1FN** sobre una tabla sin normalizar .es que simplifica la representación y facilita el desarrollo de aplicaciones de consulta.

Lo descrito anteriormente se puede ver representado en la **figura 3.3 b)**, donde los nombres de los atributos no se repiten en las tablas.

almacen

al-id	al-descripcion	al-cia	ab-id
A00001	VINO, BLANCO	OXO S.A. DE C.V.	A00001
B00002	QUESO, CHI	COMERCIAL S.A. DE C.V.	B00001
C00003	LECHE, DESCR	ALPURA S.A DE C.V.	C00001
D00004	CARNE, RES	COMERCIAL S.A. DE C.V.	D00001

al-cminima	al-cmaxima	al-existencias	al-fcaducidad
35	130	50	10-05-1995 12:00AM
5	10	6	05-06-1995 12:00AM
25	40	35	30-07-1995 12:00AM
25	50	35	30-07-1995 12:00AM

al-fingreso	pr-id	tipo-id	ab-nombre
12-04-1995 12:00AM	P00001	T1	VINO TINTO
04-05-1995 12:00AM	P00002	T2	QUESO, CHI
15-07-1995 12:00AM	P00003	T2	LECHE DESCR
15-07-1995 12:00AM	P00004	T2	CARNE, RES

pr-nombre	pr-ciudad	pr-direccion	pr-telefono
CARMEN ALCANTARA	TOLUCA, MEX	AV. CEYLAN 123, ANAHUAC	310-25-81
MANUEL ALVAREZ	D.F. , MEX	GRANADOS 35, POLANCO	555-67-78
ZORAIDA GONZALEZ	GUADALAJARA, JAL	VALLARTA 34, PALMAS	23-45-67
JESUS ARENAS	MOCHIS, SINALOA	OLVERA 456, NUEVA	34-78-98

al-fsalida	uso-id
25-05-1995 12:00AM	U01
30-05-1995 12:00AM	U01
17-07-1995 12:00AM	U02
17-05-1995 12:00AM	U02

Figura 3.3 b) Tabla en la (1FN).

[7] Definición traducida del libro [Toby J. Teorey, 92]

Definición
[8] **Segunda forma normal (2FN)**
Una tabla tiene la segunda forma normal, si y sólo si tiene la **1FN** y cada atributo tiene una **dependencia funcional (DF)**, ésta implica identificar una **llave primaria (PK)**, que cumpla esta dependencia. Un atributo es dependiente de una **PK** si está colocado al lado derecho de la misma, por consiguiente, el atributo colocado en la parte izquierda será la **PK**, también se puede dar esta dependencia usando dependencias transitivas.

Para el mejor entendimiento de la definición anterior, como siguiente paso se tienen que comprender las **DFs**.

Definición
[9] **Dependencia Funcional (DF)**
La dependencia funcional es la propiedad de uno o más atributos para determinar únicamente el valor de los otros atributos, es decir, el valor de los atributos colocados a la derecha de la **llave primaria (PK)** dependerá de ésta misma, de tal forma que cualquier evento que se presente con esta **PK**, afecta los atributos que dependan directamente de esta llave.

Por ejemplo, a continuación se tienen tres **DFs**, en la primera se ve que **cl_id** es **PK** y de esta dependen los valores de los atributos colocados a la derecha de este, tal es el caso para **cl_nombre**, **cl_direccion** y **ve_id**, de tal forma que al realizar una operación con esta llave, automáticamente esta operación afectará los atributos dependientes de la misma.
1) cl_id - > cl_nombre , cl_direccion, ve_id

La segunda **DF** se tiene para la **PK de_id**, ya que el atributo **de_nombre** también depende del tipo de manipulación que se realice con este atributo.
2) de_id - > de_nombre

La tercer **DF** es representada por la **PK ve_id** de la cual dependen los valores que tomen **ve_fventa** y **ve_descripcion**.
3) ve_id - > ve_fventa, ve_descripcion

Dependencia transitiva :
Lo que indica esta dependencia es que se puede accesar a la información que se tenga en otra tabla, si se tiene. Para este caso una **PK = cl_id** y una **FK = ve_id,** se relacionan y permiten el acceso a los datos requeridos.
cl_id -> ve_id
ve_id -> ve_fventa
Por dependencia transitiva se puede accesar a los datos que dependen de la **FK ve_id**, como se puede ver a continuación
cl_id -> ve_fventa

---

[8] Definición traducida del libro [Toby J. Teorey, 95]
[9] Definición traducida del libro [Toby J. Teorey, 95]

Cuando se tiene la normalización de las tablas en la **2FN** se forman más de dos tablas, conservando sus **DFs.** Las **DFs** mencionadas anteriormente, se pueden ver representadas en las tablas de la **figura 3.4** , dondè se muestran las tablas con sus correspondientes atributos identificadores y descriptores, que sirven para ilustrar la definición anterior, tal como se representa en una tabla implementada, además de dar los nombres de las tablas, tal como se implementarán en el **apéndice e).**

cl-id	cl-nombre	cl-direccion	ve-id
CL0001	CARLOS FRAGOSO	NARANJOS 35, ARCOS DEL ALBA	V00001
CL0002	SERGIO CHAINE	SANTA MONICA 35, SATELITE	V00002
CL0003	BRAULIO GONZALEZ	MIRALUNA 120, CUMBRIA	V00003
CL0004	JESUS ALVAREZ	VILLA PALEMO 16, VILLAS SOL	V00004

departamento

de-id	de-nombre
D00001	ALIMENTOS Y BEBIDAS
D00002	COCINA
D00003	RESTARURANTE
D00004	RELACIONES PUBLICAS

ventas

ve-id	ve-fventa	ve-descripcion
V00001	10-12-95 11:00PM	UNA BOTELLA VINO NAC.
V00002	13-12-95 11:00PM	CENA DOS PERSONAS
V00003	14-12-95 11:00PM	HELADOS Y CAFE
V00004	15-12-95 11:00PM	EVENTO BODA SR. PEREZ

POR DEPENDENCIA TRANSITIVA

cl-id	ve-fventa
CL0001	10-12-95 11:00PM
CL0002	13-12-95 11:00PM
CL0003	14-12-95 11:00PM
CL0004	15-12-95 11:00PM

Figura 3.4 Muestra dependencias funcionales.

Una vez que se tienen los esquemas de tablas en la **1FN** y **2FN,** se explicará la **3FN.**

Definición

[10] Tercera forma normal (3FN)

Una tabla está en la tercera forma normal, si y sólo si para cada **DF** no trivial **X - > A,** donde **X** y **A** son de uno a más atributos simples o una composición de atributos, de tal forma que forzosamente tenga una de estas dos condiciones, además, el atributo **X** es una **super llave,** es decir una combinación de atributos que permitirán identificar una entidad de otra, o si el atributo **A** es miembro de una **llave candidata,** es decir una combinación de atributos que no pueden ser super llaves; **A** será el primer atributo de una nueva tabla.

Si el atributo **A** es miembro de una **llave candidata, A** es llamado **atributo primo.**

[10] Definición traducida del libro [Toby J. Teorey, 99]

Para interpretar la definición anterior se debe tener en cuenta que **A** es una **DF** trivial de la forma **YZ - > Z**.
Esta forma normal remueve **dependencias transitivas de los atributos con la PK**, estas dependencias transitivas se dan cuando algunos de los atributos de una entidad dependen de una **PK**, únicamente por su relación con otro atributo que sí depende directamente de ésta, los atributos transitivos dependientes son agrupados, clasificándolos por su intermediario común. Por cada grupo obtenido se crea una nueva entidad.
Por ejemplo, tomando la notación para la **3FN** se tiene :
X = re_id
A = em_id, de_id, de_nombre, de_cempleados
re_id - > em_id, **de_id**, de_nombre, de_cempleados
donde **de_id** es un atributo primo.
Por lo tanto, **de_id**, de_nombre, de_cempleados, son miembros de una **llave candidata**, es decir formarán una nueva tabla, donde **de_id** será el primer atributo de la nueva tabla, o sea un **atributo primo**, ya que será la **PK** de la nueva tabla.
Retomando la notación ocupada en la anterior definición, se tiene :

YZ = em_id, **de_id**
Z = **de_id**, de_nombre, de_cempleados
em_id - > **de_id**
de_no - > de_nombre, de_cempleados

Retomando el ejemplo anterior para los atributos **X** y **A**, si se borra el atributo **re_id** = R00001 en la tabla **reporte_1**, se borrará también información asociada a ese atributo, como es el caso de información perteneciente a un **departamento**, para evitar esto se recurre a normalizar las tablas en la **3FN** como se presenta en la **figura 3.5 a).**

Para la **figura 3.5 b)** , lo que se evita es que al borrar por ejemplo el atributo **re_id** = R00001, se borren también todos los atributos asociados con el identificador borrado, por eso se separa el atributo primo **de_id** con sus correspondientes atributos, para formar una tabla candidata llamada **departamento,** independiente de la tabla original, para que al momento de borrar un registro, por ejemplo, **re_id** = R00001 de la tabla **reporte_2,** no se borre información importante.

reporte_1

re-id	em-id	de-id	de-nombre	de-empleados
R00001	EM0001	D00001	COCINA	6
R00002	EM0002	D00002	BAR	3
R00003	EM0003	D00003	ALMACEN	2
R00004	EM0004	D00004	MANTENIMIENTO	1

Figura 3.5 a)

reporte_2

re-id	em-id
R00001	EM0001
R00002	EM0002
R00003	EM0003
R00004	EM0004

departamento

de-id	de-nombre	de-empleados
D00001	COCINA	6
D00002	BAR	3
D00003	ALMACEN	2
D00004	MANTENIMIENTO	1

Figura 3.5 b)

Después de haber normalizado la **3FN,** aún restan anomalías por normalizar, las cuales se eliminarán por la forma normal de **Boyce Codd,** que es una variación de normalización más fuerte que la **3NF.**

Definición

[11] **Forma Normal de Boyce - Codd (FNBC)**

Sea : **A** una tabla, **R** está en **FNBC,** si para cada **DF** no trivial **X - > A, X** es una **super llave.**

Recordando que una **super llave** es un conjunto de uno o más atributos, que considerados conjuntamente permitirán identificar de forma única a una entidad de otras.

Esto es, si las **dependencias no triviales** son **DF** de la parte izquierda de la dependencia, es decir dependientes de una **llave**, para los atributos en su parte dependiente.

La **FNBC** se puede ver representada en la **figura 3.5 b)** , donde los atributos **re_id** y **em_id** de la tabla reporte forman una super llave que contendrá reportes de las actividades del empleado, además de prevenir que se pierdan datos en una instancia determinada.

Al llegar a la **FNBC** se han descrito las características estructurales de una visión de la **BD.** Usando bloques simples de construcción, el modelo normalizado reduce la redundancia y hace más explícitas las dependencias funcionales que pueden utilizarse para la integración e implementación de la **BD.**

La **FNBC** es una estricta forma de normalización, más que la **3FN,** ya que ésta elimina la segunda condición para la **3FN,** la cual deja del lado derecho de la **DF** para ser un **atributo primo,** por lo tanto, cada lado izquierdo de una **DF** en la tabla tiene que ser una **super llave.**

Una vez que se cumplió lo anterior, se puede afirmar que las tablas están en la **1FN, 2FN, 3FN y FNBC.**

[11] Definición traducida del libro [Toby J. Teorey, 99]

Lo que indica la **FNBC** es que en este paso se deben tener perfectamente identificados los atributos descriptores e identificadores, además de que en esta parte del diseño se ha logrado :

Mejorar las consultas y actualizaciones en tiempo.
Mejorar el almacenamiento en espacio.
Integridad, esto es, prevenir anomalías al manipular un registro.

Ahora se mostrará cómo eliminar ciertas anomalías que se pueden encontrar en las tablas candidatas.

Normalización de tablas candidatas derivadas del modelo-ER

La normalización de tablas candidatas (paso II d en el ciclo de vida de una **BD**) es acompañada por el análisis de **DFs** asociadas con esas tablas. Para poder eliminar algunas anomalías que todavía restan en los esquemas de las tablas, se deben utilizar **DFs** primarias y **DFs** secundarias.

La Dependencia Funcional primaria representa dependencia alrededor de **DFs,** que son **llaves de entidades, es decir, las dependencias dentro de la entidad**; esta dependencia es derivada del **modelo-ER**.

La Dependencia Funcional secundaria representa **dependencias alrededor de elementos de datos que componen una entidad en particular, esto es, dependencias dentro de la entidad**. Esta dependencia es obtenida explícitamente de requerimientos de análisis.
Para seguir con la normalización se debe tener en cuenta que una tabla puede ser eliminada cuando todos sus atributos están contenidos en otra tabla, además se debe considerar que hay dos formas normales más para eliminar el resto de las anomalías que pueden ocurrir, éstas son la **4FN** y **5FN**. Antes de definir la **4FN** es conveniente definir las **dependencias funcionales multivaluadas**, ya que éstas se utilizan para poder llegar a la **4FN**.

Dependencias Funcionales Multivaluadas (DFMs)

Cuando no existen dependencias funcionales en absoluto, la teoría desarrollada hasta ahora no proporciona ninguna base formal para descomponer la estructura en proyecciones. Para solucionar esto existe una base formal de un nuevo tipo de dependencias llamadas **dependencias funcionales multivaluadas (DFMs)**, las cuales son una generalización de las **DFs.**

Definición

[12] **Dependencia Funcional Multivaluada (DFMs)**

Sea la declaración de **DFM** X - >> Y, se lee **"el atributo A multidetermina al atributo B "** al presentar, una relación **P (A, B, C)**, se cumple, si y sólo si el conjunto de valores de **B**, que corresponden a un par (valor de **A** y valor de **C**), dado en la relación **P**, depende sólo del

[12] Definición traducida del libro [Toby J. Teorey, 111]

valor de **A** y es independiente del valor de **C**. Las **DFM** existen sólo si la relación **P** tiene tres atributos.
Dada una relación **P (A, B, C)**, la **DFM A - >> B** se cumple, si y sólo si también se cumple la **DFM A - >> C**. Las **DFMs** siempre se presentan juntas, en parejas, de esta manera es más común expresarlas con la siguiente notación **A - >> B, C**.

Una **DF** es una **DFM** donde el conjunto de valores dependientes se componen de un solo valor.
La relación **P** con atributos **A, B, C,** se puede descomponer sin pérdida en dos proyecciones **P1 (A, B)** y **P2 (A, C)**, si y sólo si la **DFM A - >> B, C** se cumple en la relación **P.**

La **DFM** expresa lo siguiente :
Si se tiene una tabla con **DF** en por lo menos 3 atributos, se puede descomponer en dos o más tablas, es decir, si se encuentran **DF** dentro de estos atributos, por ejemplo :
Si el atributo **B** tiene valores iguales correspondientes a un identificador **A** y el atributo **C** tiene valores pares distintos correspondientes a los valores contenidos en el atributo **B**, se puede descomponer en dos o más tablas, como lo muestra la **figura 3.6** , es decir, son **DFMs** en los atributos que componen dicha tabla.

Tabla a)
almacen

A	B	C
ab-id	ab-precio	uso-id
A00001	50	U01
A00001	50	U01
A00001	50	U02
A00001	50	U02
B00001	70	U03
B00001	70	U03

Tabla b)
alimentos_1

A	B	C	
ab-id	ab-precio	ab-uso1	ab-uso2
A00001	50	U01	U02
B00001	70	U03	U04

Una vez que se comprendió la **FNBC** y las **DFMs**, se definirá la **4FN**.

Definición
[13] Cuarta forma normal (4FN)
Una tabla está en la **4FN,** si y sólo si está en la **FNBC** y evaluada por **DFMs** está automáticamente en la **4FN.**

[13] Definición traducida del libro [Toby J. Teorey, 113]

Si las tablas no están en estas formas se pueden sufrir anomalías, por ejemplo, al actualizar un registro se pueden modificar datos que no se contemplaron. Lo que se pretende hasta este paso es tener los datos en **4FN**, además de la conservación de dependencias y productos sin pérdida.
Un ejemplo de tablas en la **4FN** se puede ver representado en la figura anterior, donde el atributo **ab_precio**, corresponde a un par de valores en **ab_id** y a otro par de valores en **uso_id**, por lo tanto se cumple que este atributo multidetermina a un par de atributos dentro de la entidad.

Una vez que se tienen los esquemas de tablas en la **1FN, 2FN, 3FN y 4FN** , se tienen que corregir las últimas anomalías con las que uno se puede encontrar, para esto se utiliza la **5FN**.

Definición
[14] **Quinta forma normal (5FN)**
Una tabla está en la **5FN** si ésta no puede tener otra descomposición en pequeñas tablas.

Es decir, una menor descomposición, que al **unir** esta proyección, resulte la tabla original, sin pérdida de registros, la cual recibe el nombre de **desunión**. A este tipo de dependencias se les conoce como **dependencias de unión (DUs)**; una tabla está en la **5FN** si ésta no puede ser descompuesta en dos o más proyecciones.

Una descomposición en tres o más proyecciones es equivalente a **DUs** ; por ejemplo, si una tabla está en la **4FN** con por lo menos una **DF**, se puede descomponer a la **5FN** por medio de su **llave candidata** , recordando que esta llave es una combinación de atributos que no pueden ser super llaves, teniendo en cada tabla resultante una llave candidata y una **no llave** asociada a la **llave candidata**, este ejemplo lo respalda la siguiente afirmación :
La unión **ABC** es igual a las proyecciones resultantes **Aa, Bb, Cc,** esto se puede ver representado por **transitividad y uniones**, como se muestra en la siguiente definición :

Para poder explicar la siguiente definición es necesario tener presente la **figura 3.7** , ya que a partir de ésta, gira la explicación y comprensión de la **5FN**.

Definición
Transitividad y uniones
Sea **Aa, Bb** están en **AB**; **Bb, Cc** están en **BC**; y **Aa, Cc** están en **AC**. En estas uniones se navega para poder unir los atributos que se necesiten, donde **ABC** es una tabla cualquiera. Una vez separada forma tres pequeñas tablas, conservando sus **DFs** con sus llaves candidatas, esto es, **Aa,Bb,Cc** donde se puede navegar a través de sus llaves candidatas para obtener información asociada a los atributos dependientes de estas llaves candidatas y poder obtener la información que se encontraba en la tabla original. En la **figura 3.7** se tiene la tabla **a)**

[14] Definición traducida del libro [Toby J. Teorey, 117]

habilidad_usada, donde se busca la forma de proyectarla en tres tablas derivadas de ésta, siempre y cuando no presente pérdida de información; esto se va a lograr tomando en cuenta las siguientes convenciones para este ejemplo, ya que los atributos de esta tabla tendrán los siguientes **alias** : **em_id = A, pl_numero = B, em_habilidad = C**.

Para la primer desunión, en la tabla b) **habilidad_usada1**, se tiene una **llave candidata A** y una **no llave B**.
Para la segunda desunión, en la tabla c) se tiene que **la llave candidata B** y la **no llave C**, nótese que la **llave candidata B** en la primer desunión tiene el papel de **no llave**, pero en ésta debe ser candidata, ya que por medio de ésta, cuando se **unen las tablas b) y c),** se debe lograr parcialmente la tabla original.
Para la tercer desunión, en la tabla d) , se tiene la **llave candidata A** y la **no llave C**. Nótese que **A** también es candidata en la primer desunión.
Respecto a las desuniones mencionadas anteriormente, se logran fácilmente de la siguiente forma :

Para la tabla b)
Primero, se debe buscar una posible **llave candidata**.
Segundo, se debe buscar qué posibles valores puede tomar esa llave candidata, siempre y cuando no se repitan, y colocarlos en esta tabla.

Para la tabla c)
Se busca la **llave candidata**; en este caso será la **no llave** de la anterior desunión, y se buscan posibles valores que dependen de esta llave, siempre y cuando no se repitan, y se colocarán en esta tabla.

Para la tabla d)
Se busca **la posible llave candidata**; en este caso será **la llave candidata que se identificó en la primer desunión (tabla b)**, de tal forma que se busquen dependencias de la no llave para con la llave candidata, siempre y cuando no se repitan los valores de la no llave. En este ejemplo se ven los valores de las llaves candidatas, además de que se confirma que no se repitan las no llaves.
Una vez que se entendió cómo trabajar las desuniones, a continuación se van a comprender las uniones , que permiten afirmar que la desunión que se hizo efectivamente se realizó de forma correcta.
Para comprobar **se une la tabla b) y c),** de tal manera que se **obtiene la tabla bc).** Para logra esto se tienen los siguientes pasos :
Primero, se identifica que **DFs** tienen las **llaves candidatas** en la **tabla b).**
Segundo, se identifican las **DFs** de las **no llaves** en la **tabla b)** y se buscan sus posibles **DF** en **la tabla c)** y **se coloca en la tabla bc).**
Tercero, este mismo criterio se sigue hasta finalizar con las llaves candidatas de la **tabla b)** y colocándose en la **tabla bc),** como se muestra en la **tabla bc).**

Nótese que aún no se tiene la tabla original (**habilidad_usada**). Para poder llegar a ésta hay que **unir la tabla bc) con la tabla d).**

Para lograr esta unión se parte de la tabla **bc),** aquí se observa cuáles son los valores que dependen de la **llave candidata A** y que están representados en la **no llave** de la tabla d), esto se debe hacer valor por valor. Una vez que se identifican, se van **colocando en la tabla bcd),** nótese que el último valor de la **tabla bc)** se elimina, ya que no se encuentra un valor dependiente de esta llave candidata en la tabla d).
La teoría descrita anteriormente en la **5FN**, se puede ver representada en la siguiente figura :

a) habilidad_usada

A	B	C
em-id	pl-id	hab-id
EM0001	PL0003	HAB001
EM0001	PL0003	HAB002
EM0001	PL0004	HAB001
EM0001	PL0004	HAB003
EM0002	PL0003	HAB001
EM0002	PL0003	HAB002
EM0002	PL0004	HAB001
EM0002	PL0004	HAB002

b) habilidad_usada1

A	B
em-id	pl-id
EM0001	PL0003
EM0001	PL0004
EM0002	PL0003
EM0002	PL0004

c) habilidad_usada2

B	C
pl-id	hab-id
PL0003	HAB001
PL0003	HAB002
PL0004	HAB001
PL0004	HAB002
PL0004	HAB003

d) habilidad_usada3

A	C
em-id	hab-id
EM0001	HAB001
EM0001	HAB002
EM0001	HAB003
EM0002	HAB001
EM0002	HAB002

Si se une b) y c)
se forma la tabla **bc)**

A	B	C
em-id	pl-id	hab-id
EM0001	PL0003	HAB001
EM0001	PL0003	HAB002
EM0001	PL0004	HAB001
EM0001	PL0004	HAB002
EM0001	PL0004	HAB003
EM0002	PL0003	HAB001
EM0002	PL0003	HAB002
EM0002	PL0004	HAB001
EM0002	PL0004	HAB002
EM0002	PL0004	HAB003

Si se une bc) con la tabla d)
se tiene la tabla **bcd)**

A	B	C
em-id	pl-id	hab-id
EM0001	PL0003	HAB001
EM0001	PL0003	HAB002
EM0001	PL0004	HAB001
EM0001	PL0004	HAB003
EM0002	PL0003	HAB001
EM0002	PL0003	HAB002
EM0002	PL0004	HAB001
EM0002	PL0004	HAB002

Figura 3.7 Tablas en la (6FN).
Aplicando Dependencias de Unión.

Resumen

En este capítulo se tienen que reconocer valores únicos en la **1FN**; en la **2FN** se deben identificar **DFs** , esto implica determinar la **PK**. En la **3FN** se deben establecer **DFs** no triviales, donde los atributos involucrados son uno o más atributos simples o una composición de éstos, también se deben **determinar super llaves**, es decir atributos que por su definición y contenido formarán una entidad en particular, asimismo se deberá **identificar llaves candidatas**, es decir una combinación de atributos que no pueden ser super llaves, con las cuales se formarán nuevas tablas; en la **4FN** se evalúan las tablas en la **FNBC**, en la cual se identifican super llaves. Asimismo se asegura que todas las tablas contengan super llaves, además de que estas tablas se deben evaluar por **DFMs**, si una tabla está evaluada en la **FNBC** y por **DFMs** automáticamente está en la **4FN**; en la **5FN** se evalúan las tablas por dependencias de unión y desunión, de tal forma que una tabla no pueda tener una menor descomposición.

Capítulo IV

Ejemplo

Análisis del problema y solución

CAPÍTULO IV
EJEMPLO

En este capítulo se hace referencia a la transformación de la base de datos en tablas, la cual se realiza una vez que se tiene en una forma normal y se obtuvo durante el desarrollo del capítulo III, se puede visualizar en su totalidad en el **apéndice e)** , también se mostrarán las necesidades de información que se presentan diariamente con los usuarios en el área de alimentos y bebidas, con las cuales se ejemplificarán los anteriores capítulos y sus respectivos apéndices, de tal forma que estos requerimientos se abordaron y se les dio solución a través del **modelo-ER** y de los pasos que involucra el ciclo de vida de una **BD**. Este proceso se tomó desde que el producto llega al restaurante por medio de proveedores, hasta que sale el mismo procesado en un alimento de consumo para un cliente en el restaurante.

Para poder involucrarse en este capítulo se debe tener en cuenta que las necesidades de información son con base en el giro de la empresa, venta de alimentos y bebidas; también se debe considerar que este proceso se toma desde que el producto llega al restaurante hasta que sale procesado en un alimento o una bebida.

Teniendo en cuenta lo descrito anteriormente, se presentará la forma como se analizó el problema sin profundizar en los pasos que éste involucró.

IV.1 Análisis del problema y solución

Después de haber realizado la investigación bibliográfica correspondiente, además de entrevistas con los usuarios, se presenta la siguiente descripción de puestos, así como funciones del área de alimentos y bebidas para un restaurante. Por otro lado, se hace énfasis en que a partir de este análisis se identificó cómo está constituido y se maneja un restaurante y qué necesidades tiene en cuanto a información se refiere.
El análisis se logró al realizar los siguientes puntos :

1) **Investigación de bibliografía** relacionada con restaurantes.
2) Después de conocer cómo funciona la empresa, se **identificaron jerarquías,** presentándolas en un diagrama descendente de los puestos dentro de la organización.
3) Una vez teniendo los puestos, se explicaron cómo están **subordinados** unos de otros, así como las funciones que realizan. Estos se pueden ver representados en la **figura 4.1** (puestos del área de alimentos y bebidas)
4) Con estos datos se procedió a **identificar las entidades** dentro de la organización.
5) Una vez identificadas las entidades, se describieron y documentaron para saber que es lo qué contienen, esto se puede ver en el **apéndice c)** .
6) Con las entidades **se identificaron los atributos** pertenecientes a cada una de ellas.
7) En la identificación de atributos se procedió a describirlos y documentarlos para saber qué es lo que contienen, además de la entidad a la que pertenecen. Los pasos 6 y 7 se representan en los **apéndices c) y d)**
8) En esta parte del trabajo se **normalizó la BD**, en el **capítulo III**.

9) Teniendo perfectamente identificadas entidades y atributos, el siguiente paso fue continuar **aplicando el modelo-ER**, que dio solución a necesidades de información que en él se presentan.
10) Este **modelo-ER** muestra un **flujo de información** tremendo, del cual sólo se tomaron los datos necesarios para dar salida a la información requerida.
11) En las relaciones se muestra **el nombre de la relación** (requerimientos de información) y las entidades que se relacionan, así como los atributos que la forman. Además se da una llave primaria o foránea de acceso para esá relación.

En este capítulo se hace énfasis en los puntos 10 y 11, con los cuales se ejemplificará la parte final de este trabajo, ya que por la cantidad de ocurrencias de información de cada uno de los usuarios que trabajan en el área de alimentos y bebidas, sería muy extenso mostrar toda la solución a los requerimientos, por eso sólo se tomarán algunas ocurrencias de los usuarios para ejemplificarlas y darles solución a través de consultas a la **BD**, haciendo énfasis en que este mismo procedimiento se sigue con las demás ocurrencias que se presentan en este capítulo y que por lo extenso de la solución sólo se presentan en el **apéndice f).**
Para la mejor comprensión de este capítulo y por lo mencionado durante este trabajo, respecto a que la manipulación de la **BD** se realizará por medio de **Sybase** , primero se dará una breve introducción sobre manipulación de **BD** , la cual se puede ver en el **apéndice g)** , en el que se muestrá la sintaxis de una determinada cláusula, así como su ejemplo correspondiente, de tal forma que este apéndice sirva para familiarizase con un lenguaje estructurado de consulta, ya que de esta forma será mejor comprendido este capítulo.
Una vez comprendido el **apéndice g)** , se darán los siguientes ejemplos de manipulación que serán la culminación de este trabajo, ya que por medio de éstos se pueden satisfacer las necesidades planteadas por los usuarios en el análisis, tomando como ejemplos algunas de las necesidades de información que tiene el gerente en el área de alimentos y bebidas, en las cuales se muestrán las ocurrencias, así como el **alias** de la tabla donde las consultará, tales como las siguientes :

IV.2 Necesidades de información del gerente de alimentos y bebidas.

Es el responsable ante el gerente, de la adecuada administración del área de alimentos y bebidas. Sus principales funciones son :
1) Supervisa el trabajo realizado por los empleados a su cargo (chef, maitre, contralor de costos, jefe de banquetes, jefe de bares y chef steward). **CO**
2) Junto con el gerente general, el jefe de compras y el almacenista elabora especificaciones **standard de compras** de alimentos y bebidas. **CO**
3) **Adquisición de mercancías** cuyos requisitos no estén en los standares de compras. **CO**
4) Supervisa **máximos y mínimos** de alimentos y bebidas. **AL**
5) Elabora hojas de **costos de recetas standard** de alimentos y bebidas. **RE**
6) Fija los **precios de venta** de alimentos y bebidas. **RE**
7) Elabora y actualiza **menús**. **ME**
8) **Reporte diario de ventas** de alimentos y bebidas. **VE**
9) Autoriza **ventas de promociones y cortesías**. **P**
10) Promueve las buenas relaciones entre el personal a su cargo.

11) Promueve juntas con su equipo de trabajo.
12) Contrata **variedades. CO**
14) Conoce **capacidad del restaurante y fechas de ocupación. EV**
15) Controla **horarios y fechas de eventos. EV**
16) Políticas para **cancelación de eventos. EV**

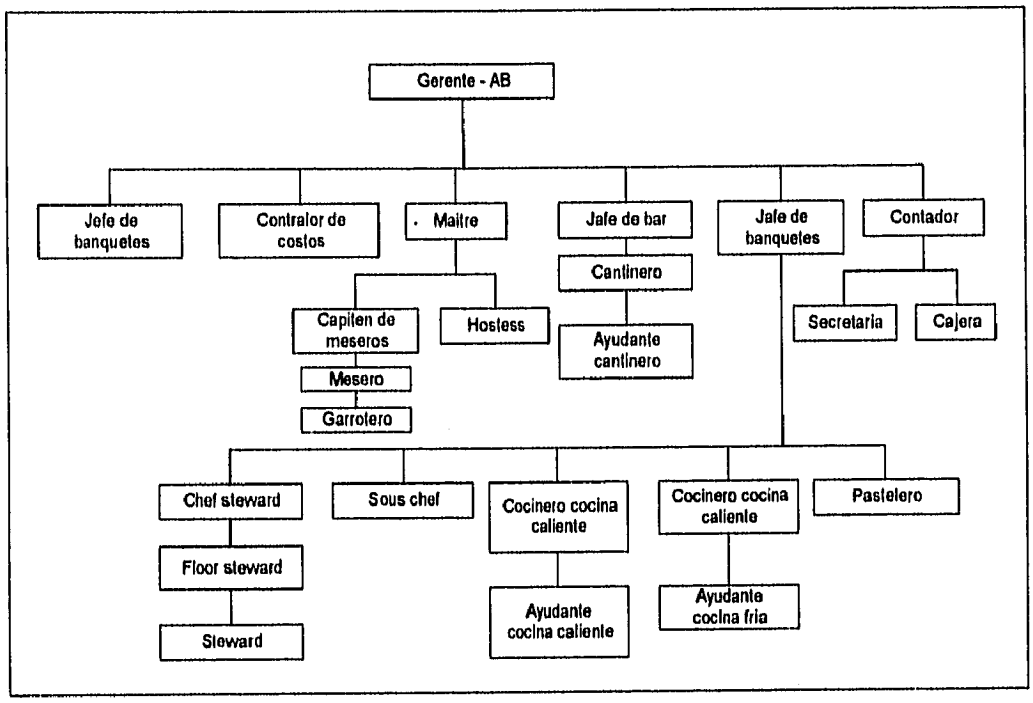


Figura 4.1 Puestos de los empleados dentro del restaurante.

En las siguientes figuras **4.2 a), 4.2 b) y 4.2 c)** muestro consultas a la **BD.**

OCURRENCIAS DE INFORMACIÓN.	REPRESENTACIÓN GRAFÍCA DEL MODELO-ER	CONSULTAS DE OCURRENCIAS EN LAS TABLAS DE LA DB.
EL GERENTE DESEA SABER CUALES SON LOS ALIMENTOS QUE COMUNMENTE ADQUIERE EL DEPARTAMENTO DE COMPRAS Y CUYO COSTO UNITARIO SEA MENOR DE $ 100 Y QUE EL PROVEEDOR SEA ALPURA S.A. DE C.V.	**OCURRENCIA : STANDART_COMPRAS** pr_id (PK) ab_id (FK) pr_nombre pr_cia pr_telefono pr_tentrega PROVEEDORES prd_id (PK) ab_id (FK) prd_nombre prd_cunitario PRODUCTOS	**select** pr_id.pr, prd_id.prd, ab_id pr_nombre, pr_cia, pr_telefono, pr_tentrega, prd_nombre, prd_cunitario **from** proveedores pr, productos prd **where** pr_id.pr = prd_id.prd **and** prd_cunitario < 100 **and** pr_nombre='ALPURA S.A. DE C.V.' **go**
EL GERENTE DESEA SABER QUE DISTRIBUIDORES TIENE DISPONIBLES PARA ABASTECER UN PEDIDO, DADO EL CASO QUE SE NECESITE ALGUN PRODUCTO CON CIERTA URGENCIA.	**OCURRENCIA : ADQUIERE_MERCANC** pr_id (PK) pr_nombre pr_ciudad pr_telefono pr_pago pr_cunitario pr_tentrega PROVEEDORES prd_id (PK) ab_id (FK) prd_nombre PRODUCTOS	**select distinct** pr_id.pr, pr_nombre prd_id.prd, ab_id, pr_ciudad, pr_telefono, pr_pago, pr_cunitario, pr_tentrega **from** proveedores.pr, productos.prd **where** ab_id='AOOOO1' **and** pr_id.pr=prd_id.prd **go**
EL GERENTE DESEA SABER CUALES SON SUS CANTIDADES MAXIMAS Y MINIMAS DE LOS ALIMENTOS Y BEBIDAS, ASI COMO LA EXISTENCIA QUE TIENE ACTUALMENTE EN EL ALMACEN.	**OCURRENCIA : MAXIMOS_MINIMOS** al_id (PK) al_descripcion al_existencias al_cminima al_cmaxima ALMACEN	**select** al_id, al_descripcion, al_existencias, al_cminima, al_cmaxima **from** almacen **go**
EL GERENTE DEBE ELABORAR SEMANALMENTE EL COSTO QUE TIENE REALIZAR LAS RECETAS DE ALIMENTOS Y BEBIDAS.	**OCURRENCIA : COSTO_RECETAS** re_id (PK) re_nombre re_descripcion re_fecha re_ingredientes re_costo RECETAS	**select** re_nombre, re_descripcion, re_ingredientes, **sum**(re_costo) **from** recetas **where** re_fecha= '07-Marzo-96 00:00AM' **and** '13-Marzo-96 23:59PM' **go**

Figura 4.2 a) Consultas de necesidades de información en la BD.

OCURRENCIAS DE INFORMACIÓN.	REPRESENTACIÓN GRAFÍCA DEL MODELO-ER	CONSULTAS DE OCURRENCIAS EN LAS TABLAS DE LA DB.
SUPONGASE QUE EN EL AREA DE ALIMENTOS Y BEBIDAS DESEAN SABER EL PRECIO DE VENTA DE LOS DIFERENTES PLATILLOS.	**OCURRENCIA : PRECION_VENTA** re_id (PK) re_nombre re_descripcion re_ingredientes re_pventa RECETAS	**select distinct** (re_id), re_nombre, re_descripcion, re_ingredientes, re_pventa **from** recetas **go**
EL GERENTE DESEA SABER QUE MENUS TIENE DISPONIBLES PARA CADA UNO DE LOS DIAS DE LA SEMANA.	**OCURRENCIA : MENUS** me_id (PK) me_nombre me_descripcion me_costo men_id (m) men_id (PK) men_nombre MENUS — MENU_SEMANA	**select** distinct me_id, men_id.me, men_id.men me_nombre, me_descripcion, me_costo, men_nombre **from** menus.me, menu_semana.men **where** men_id.me=men_id.men **go**
EN GERENTE INFORMA DIARIAMENTE AL DUEÑO DEL RESTAURANTE DE LAS VENTAS OCURRIDAS DURANTE UN DETERMINADO DIA, O BIEN SEMANALMENTE.	**OCURRENCIA : REPORTE_VENTAS** ve_id (PK) ve_factura ve_fventa ve_total VENTAS	**select** ve_id, ve_factura, ve_fventa, **sum**(ve_total) **from** ventas **where** ve_fventa **between** "01-Enero-96 00:00AM" **and** '01-Enero-96 23:59PM' **go** **create view** reporte_vetas **as** **select** ve_id, ve_factura, ve_fventa, **sum**(ve_total) **from** ventas **where** ve_fventa **between** "01-Enero-96 00:00AM" **and** '01-Enero-96 23:59PM' **go**
EL GERENTE DESEA SABER QUE TIPO DE PROMOCIONES TIENE EL RESTAURANTE DEL 1 AL 15 DE FEBRERO DE 1996, ASI COMO LOS PORCENTAJES DE LAS PROMOCIONES QUE SE OTORGAN A ESTAS.	**OCURRENCIA : PROMOCIONES** p_id (PK) p_nombre p_horario p_finicial p_ffinal p_porcentaje PROMOCIONES	**select** p_id, p_nombre, p_horario, p_finicial, p_ffinal **from** promociones **where** p_finicial= '01-Febrero-96 00:00AM' **and** p_ffinal='15-Febrero-96 23:59PM' **go**

Figura 4.2 b) Consultas de necesidadea de información en la BD.

OCURRENCIAS DE INFORMACIÓN.	REPRESENTACIÓN GRAFÍCA DEL MODELO-ER	CONSULTAS DE OCURRENCIAS EN LAS TABLAS DE LA DB.
SUPONGASE QUE EL GERENTE DESEA SABER QUE PROVEEDOR PUEDE TENER UN MARIACHI DISPONIBLE EL 21 DE MARZO DE 1996, Y UNA VEZ QUE SE TIENE AL PROVEEDOR SE PREGUNTA SI ESTA DISPONIBLE PARA LA FECHA.	**OCURRENCIA : CONTRATA_VARIDAD** va_id (PK) pr_id (FK) va_nombre pr_id (PK) pr_nombre pr_telefono VARIEOADES — PROVEEDORES	**select** va_id, pr_id.va , pr_nombre, pr_telefono **from** variedades va, proveedores pr **where** va_nombre='MARIACHI' **and** pr_id.va = pr_id.pr **go**
SUPONGASE QUE SE DESEA SABER LOS EVENTOS QUE HAY PROGRAMADOS DEL 1 AL 15 DE MAYO DE 1996, ADEMAS DE LA CAPACIDAD QUE TIENEN ESTOS EVENTOS.	**OCURRENCIA : FECHAS_OCUPACION** ev_id (PK) ev1_id (FK) ev_fecha ev_capacidad ev1_id (PK) ev1_nombre EVENTOS — EVENTOS_1	**select** ev_id, ev1_id, ev1_nombre, ev_fecha, ev_capacidad **from** eventos, enenios_1 **where** ev_fecha **between** '01-Marzo-96 00:00AM' **and** '15-Marzo-96 23:59PM' **go**
SUPONGASE QUE SE QUIERE SABER EL NUMERO DE EVENTOS CANCELADOS QUE SE TUVIERON EN EL MES DE MARZO DE 1996 Y CUYO PRECIO FUE MAYOR O IGUAL A $ 10,000.	**OCURRENCIA : EV_CANCELADOS** ev_id (PK) ev_descripcion ev_fecha ev_fcancelado ev_costo EVENTOS	**select** ev_id ev_descripcion, ev_fecha, ev_costo, ev_fcancelado **from** eventos **where** ev_fecha **between** '01-Marzo-96 00:00AM' **and** '31-Marzo-96 23:59PM' **and** ev_costo >= 10,000 **go**

Figura 4.2 c) Consultas de necesidades de información en la BD.

Resumen

En este capítulo se señalaron las necesidades de requerimientos de información que se obtuvieron durante el análisis. Una vez obtenidos, se presentó la solución a éstos, manipulándolos en **Sybase**, de tal forma que cumplió con lo establecido en el objetivo de este trabajo durante el desarrollo del mismo, teniendo su culminación en este capítulo.

También se debe tener en cuenta que estas necesidades de información pueden cambiar con el tiempo, de tal forma que esto implicaría que se presenten otras necesidades de información, pero en este trabajo se muestran bases sólidas para que estas modificaciones, en cuanto a manipulación se refiere, sean mínimas y que no afecten el diseño lógico que fue parte primordial en este trabajo.

Conclusiones

Durante el desarrollo de este trabajo, uno como diseñador se da cuenta de lo valiosa que es la planeación de cualquier tarea que se desee emprender, ya que si no respetamos una metodología, se puede caer en errores tales como omisión de datos importantes para la empresa. Para prevenir esto, se tomó como apoyo la teoría mencionada anteriormente, pero como parte adicional, para poder analizar un problema determinado en su estructura lógica, se tiene una herramienta muy valiosa, la cual es la teoría existente del **modelo-ER**, que no solamente se puede aplicar al diseño de bases de datos, sino a cualquier problema que se desee analizar, claro, realizando una abstracción correspondiente en cuanto a sus ventajas y desventajas.

Durante el desarrollo de los capítulos se consideró la organización como un todo, que poco a poco, aplicando el **modelo-ER**, se analizó en cada una de las partes que componen la misma y de tal forma que los problemas abordados individualmente se van solucionando conforme transcurre el ciclo de vida de la **BD.**

Respecto al **modelo-ER**, es el más utilizado actualmente por los diseñadores de **BD**, ya que se transforma fácilmente en tablas en cualquier **SMBD**, por lo tanto, se debe considerar como una herramienta de vanguardia en el diseño lógico de **BD**.
También, por la experiencia profesional que he tenido hasta este momento, aseguro que este modelo es una herramienta confiable para mantener la integridad y consistencia de una **BD**, ya sea centralizada o distribuida, esto diseñando aplicaciones acordes a lo establecido en el diseño lógico, sin descartar las fallas que pueden ocurrir y que se mencionaron en el capítulo correspondiente.

Además, debido al gran auge que han tenido las **BD** en todas las empresas y dependencias gubernamentales, este campo es uno de tantos en los que profesionales egresados de la **Licenciatura en Matemáticas Aplicadas y Computación** pueden desarrollarse, tomando en cuenta que el valor de la información es muy grande cuando se toma una decisión dentro de una organización. Debido a esto, en este trabajo se plantea la importancia del **diseño lógico** de un problema determinado, ya que a partir de éste será la confiabilidad de los datos que una empresa ponga en nuestras manos y a medida de los resultados que se obtengan permitirá a los profesionales involucrados en esta actividad crecer junto con la empresa.

Apéndices

APÉNDICES

Como parte importante para la mejor comprensión de este trabajo y como complemento, se presentan varios apéndices, que como su nombre lo indica, son parte de este trabajo que añade y subsana omisiones durante el desarrollo del mismo, estos apéndices se muestran en esta parte del trabajo y son una herramienta de referencia importante para la mejor comprensión de las bases de datos y abreviaturas utilizadas alrededor de este tema.

Apéndices

Apéndice a) Abreviaturas y Definiciones
Este apéndice muestra abreviaturas de palabras que se utilizan continuamente, además de algunas definiciones a las cuales se hará referencia una o dos veces durante el desarrollo del mismo.

Apéndice b) Integración del diseño lógico de la base de datos
Este apéndice es parte del capítulo II, donde se presentan las herramientas necesarias para construir el diseño lógico de la base de datos, además de explicar en detalle los pasos por los que atraviesa este diseño lógico y que por la manera como está definido este trabajo, se escogieron ejemplos que mejor se prestaron para la total comprensión del capítulo II, de tal forma que en este apéndice se observa la integración del diseño lógico, omitiendo detalles de cómo se realizó.

Apéndice c) Descripción de la base de datos
Aquí se puede ver una descripción detallada de tablas, atributos y llaves.

Apéndice d) Resumen de tablas de la base de datos
Este apéndice presenta un resumen de tablas, que serán útiles como referencia rápida cuando no se recuerde la estructura de la base de datos.

Apéndice e) Transformación del modelo-ER en tablas
Aquí se muestra la transformación de esquemas de tablas, ya normalizadas, con sus correspondientes atributos identificadores y descriptores, a tablas que realmente contendrá la base de datos, utilizando **Sybase**.

Apéndice f) Requerimientos de información
Este apéndice muestra el total de requerimientos de información, para cada uno de los usuarios en el área de alimentos y bebidas.

Apéndice g) Introducción a SQL-server Sybase
Este apéndice presenta una visión de lo que es un lenguaje estructurado de consulta, el cual servirá para una mejor comprensión del capítulo IV.

Apéndice a)

Abreviaturas y Definiciones

APÉNDICE A
Abreviaturas y definiciones

Aquí se presentan **abreviaturas** de palabras de uso continuo, además de algunas **definiciones,** para la práctica. Cuando se hable por primera vez de una abreviatura, se escribirá la palabra o conjunto de palabras además de su correspondiente abreviatura, con letras mayúsculas obscuras entre paréntesis, las demás veces que se haga mención a esa palabra, únicamente se utilizará la abreviatura correspondiente denotándola con letras mayúsculas obscuras.

El apéndice está organizado de la siguiente forma :
Para las **abreviaturas** se escribe la palabra o conjunto de palabras en español y en inglés, omitiendo aquellas que sean conocidas por su significado de mejor forma en el idioma inglés.
Para las **definiciones** se escribe la palabra, así como su definición correspondiente.

Abreviaturas

Base de datos. **(BD), DB**

Modelo entidad relación. **(modelo-ER), E/R**

Lenguaje estructurado de consulta. **(SQL)**

Software. **(SW)**

Hardware. **(HW)**

Sistema manejador de bases de datos. **(SMBD), DBMS, RDBMS**

Lenguaje de manipulación de datos. **(LMD), DML**

Lenguaje de definición de datos. **(LDD), DDL**

Administrador de la base de datos. **(ABD), DBA**

Query. **(QRY)**

Dependencia funcional. **(DF)**
Dependencias funcionales. **(DFs)**

Dependencia funcional multivaluada. **(DFM)**
Dependencias funcionales multivaluadas. **(DFMs)**

Dependencia de unión. **(DU)**
Dependencias de unión. **(DUs)**

Llave primaria. **(PK)**
Llaves primarias. **(PKs)**

Llave foránea. **(FK)**
Llaves foráneas. **(FK)**

Definiciones

Aplicaciones
Son programas de aplicación (programas escritos y compilados en un determinado lenguaje), se crean por medio de un **LMD,** que es precompilado por un precompilador de **LMD**. Una vez compilados son llamados a procedimientos en lenguaje principal y se crea un código objeto.

Compilador
Software que convierte un programa escrito en un determinado lenguaje, en un lenguaje máquina, es decir, comprensible por el ordenador.

Compilar
Operación que ejecuta un compilador.

Query
Es igual a una consulta de una determinada ocurrencia en la base de datos.

Alias
Palabra o conjunto de letras con los que se identifica una entidad, es decir, un seudónimo de la misma.

Reglas del negocio
Políticas a seguir durante el diseño de la **BD.**

Manipulación
Se entenderá como manipulación al almacenamiento y recuperación de datos (actualización, consulta, inserta y elimina información) o cualquier operación con la **BD,** conservando su consistencia , siendo éste el principal objetivo de un **SMBD**.

Datos e información
Son sinónimos en este trabajo.

Diseño descendente
Es el proceso mediante el cual un problema general se descompone en varios pequeños problemas, de tal forma que se solucione cada uno de ellos, realizando pequeñas tareas que en su conjunto optimizan el problema general planteado en un principio.

Transacción
Operación con datos que lleva a cabo internamente **Sybase.**

Apéndice
Parte que se añade a un libro para ampliarlo y subsanar omisiones.

Término
Palabra o vocablo, es decir, cada uno de los elementos necesarios para una relación gramatical.

Abstracción
Reflexión de las cualidades de algo.

Vista
Es una visión de datos requeridos en una consulta.

Prólogo
Discurso que precede ciertas obras para explicarlas o presentarlas al público.

Implementación
Es cuando los esquemas de las tablas son transformados en tablas y manipulados, utilizando **Sybase**.

Implementar
Ejecutar y poner en práctica la **BD en SMBD.**

Trivial
Es algo obvio o muy simple.

Procedimientos almacenados
Procedimientos escritos en **SQL** y almacenados en el servidor, estos son llamados conforme los requirieran las aplicaciones.

Triggers
Son procedimientos escritos en **SQL**, que se ejecutan automáticamente cuando se actualiza, inserta o se borra información.

Plataformas
Software y Hardware donde se instala **Sybase**.

Apéndice b)

Integración del Diseño Lógico de la Base de Datos

APÉNDICE B
Integración del diseño lógico de la base de datos

Este apéndice es parte del capítulo II, donde se presentan las herramientas necesarias para construir el diseño lógico de la base de datos, además de explicar en detalle los pasos por los que atraviesa este diseño lógico y que por la manera como está definido el trabajo, se tomaron ejemplos que mejor se prestaron para la total comprensión del capítulo II, de tal forma que en este apéndice se presenta la integración del diseño lógico, omitiendo detalles de cómo se realizó.
Para la mejor comprensión de estos modelos se utilizará la siguiente notación, la cual se presentó en el capítulo II.
Una **línea perpendicular (I)** entre la relación y la entidad, indica que la entidad es mandatoria, por lo tanto, forzosamente debe existir. Su conectividad mínima es uno.
Un **cero (0)** entre la entidad y la relación, indica que la entidad es opcional, es decir, su existencia es opcional; su conectividad mínima es cero.
Si no se encuentra una línea perpendicular entre la entidad y la relación, su existencia es indefinida, es decir, puede o no existir; su conectividad mínima es uno.
Para los valores de conectividad se entenderá lo siguiente :
N, M, P serán igual a dos o más ocurrencias y 1 es igual a uno.

Estas pueden tener los tres tipos, dependiendo de la ocurrencia de información.

Un ejemplo se tiene en la siguiente figura :

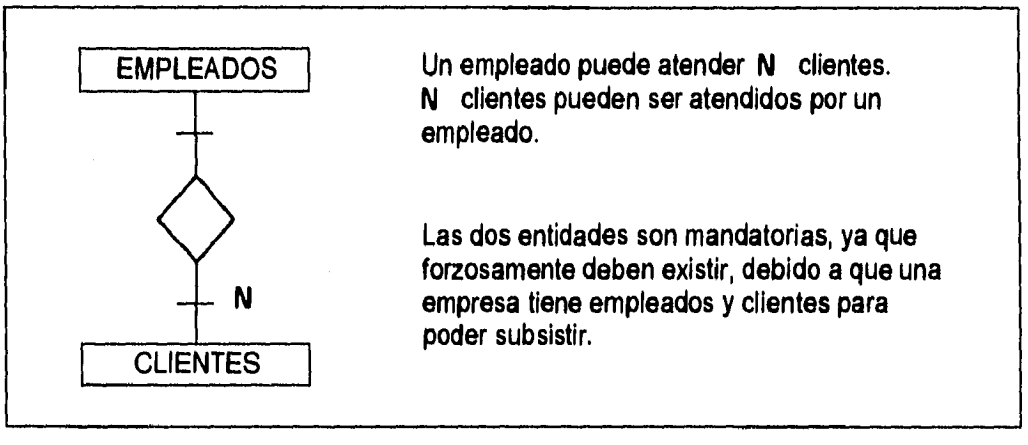


Figura b.1 Interpretación a la conectividad de empleados y clientes.

A continuación se presentan algunas de las posibles interpretaciones a las cardinalidades que se pueden ver en la integración al diseño lógico :

Empleados

N empleados están dentro de una nómina y una nómina contiene N empleados.
N departamentos están dentro de una nómina y una nómina contiene N departamentos.
N puestos están dentro de una nómina y una nómina contiene N puestos posibles para los empleados.
Un empleado maneja un departamento y un departamento contiene N empleados.
Un empleado está casado con sólo un empleado.
Un empleado puede manejar N empleados.
Un empleado tiene un horario en un turno determinado y N horarios se asignan a N empleados.
Un empleado puede tener N modales y N modales pueden ser propios de un empleado.
N capacitaciones pueden ser dadas a N empleados y N empleados pueden tener N capacitaciones.
N empleados pueden tener N sugerencias y N sugerencias son dadas por N empleados.
Un empleado es encargado de escribir la bitácora y la bitácora es escrita por un empleado en un turno determinado.
Un platillo es preparado por N empleados y N empleados preparan N platillos.

Alimentos

N alimentos pueden ser devueltos al almacén y las devoluciones son opcionales al almacén.
N alimentos están almacenados en un almacén y un almacén contiene N alimentos.
N proveedores surten N productos y N productos son surtidos por N proveedores.
N alimentos son surtidos por un proveedor y N proveedores surten N alimentos.
N alimentos son vendidos en una venta y una venta tiene N alimentos.
Una venta tiene uno o más clientes y N ventas pueden tener N clientes.
Una venta puede ser debido a un evento y un evento involucra una venta.
Una compra involucra uno o más alimentos y uno o más alimentos involucran una compra.
N proveedores pueden estar involucrados en una venta y una compra puede involucrar varios proveedores.
Una compra puede ser debido a un evento y un evento puede involucrar una compra.
N lugares de uso tienen los alimentos en el almacén.
Un almacén tiene productos con N lugares de uso.
N alimentos tienen un almacén y un almacén tiene N alimentos.
N proveedores surten un alimento en el almacén y un almacén tienen N alimentos.
Se tienen N recetas para N alimentos y con N alimentos se prepara una receta.

Requisición

Una requisición es necesaria para N alimentos y N alimentos necesitan forzosamente una requisición.
N alimentos pueden tener N pruebas de rendimiento y N pruebas pueden ser aplicadas a N alimentos.
Una comanda es necesaria para la preparación y entrega de un alimento.
N alimentos necesitan una comanda para su preparación y entrega.

Clientes
N clientes pueden o no tener N reservaciones y N reservaciones son opcionales para N clientes.
N clientes pueden tener N sugerencias y N sugerencias son dadas por N clientes.

Objetos
Un objeto puede ser olvidado varias veces y varios objetos pueden ser olvidados una vez.

Servicios
Un servicio es dado N veces y N servicios pueden ser dados una sola vez.

Eventos
Una relación pública puede provocar uno o más eventos y uno o más eventos pueden ser consecuencia de una buena relación pública.
Una relación pública puede o no involucrar una promoción y una promoción puede o no estar contenida en una relación pública.
Una venta no involucra necesariamente una promoción y una promoción no es necesaria en una venta.
Un alimento no necesariamente debe estar promocionado y las promociones son opcionales para los alimentos de poco movimiento.
Un evento no necesariamente involucra una promoción y una promoción es opcional para un evento y conservar los clientes.
N alimentos son requeridos para un evento y un evento necesita forzosamente de N alimentos.
N accesorios son necesarios para un evento y un evento necesita forzosamente de N accesorios.
Un evento es opcional por lo tanto, dependiendo de la existencia de éste necesita o no accesorios.

Accesorios
N accesorios son necesarios para un evento y un evento necesita forzosamente de N accesorios.
N accesorios son utilizados en uno o más lugares de uso y N lugares de uso pueden utilizar N accesorios.

Estadísticas
N estadisticas son necesarias de N clientes y N clientes tienen N estadísticas.
Una nómina tiene N estadísticas y N estadísticas abarcan una nómina.
N compras tienen N estadísticas y N estadísticas abarcan una compra.
N ventas tienen N estadísticas y N estadísticas abarcan varias ventas.
N estadisticas generan N reportes y N reportes son generados por N estadísticas.
Nota :
Se pueden tener estadísticas de todas las entidades contenidas en la **BD**, dependiendo de lo que se quiera evaluar.

Contabilidad
Una contabilidad abarca N empleados y un empleado tiene sólo una contabilidad.
Una contabilidad abarca una empresa y una empresa tiene sólo una contabilidad.

Una vez que se tiene presente la interpretación de algunas de las cardinalidades de la integración del diseño lógico , se presentan en las siguientes figuras :

Aquí se omitirá el rombo que representa la relación entre cada una de éstas debido que resulta más práctico trabajar de esta forma.

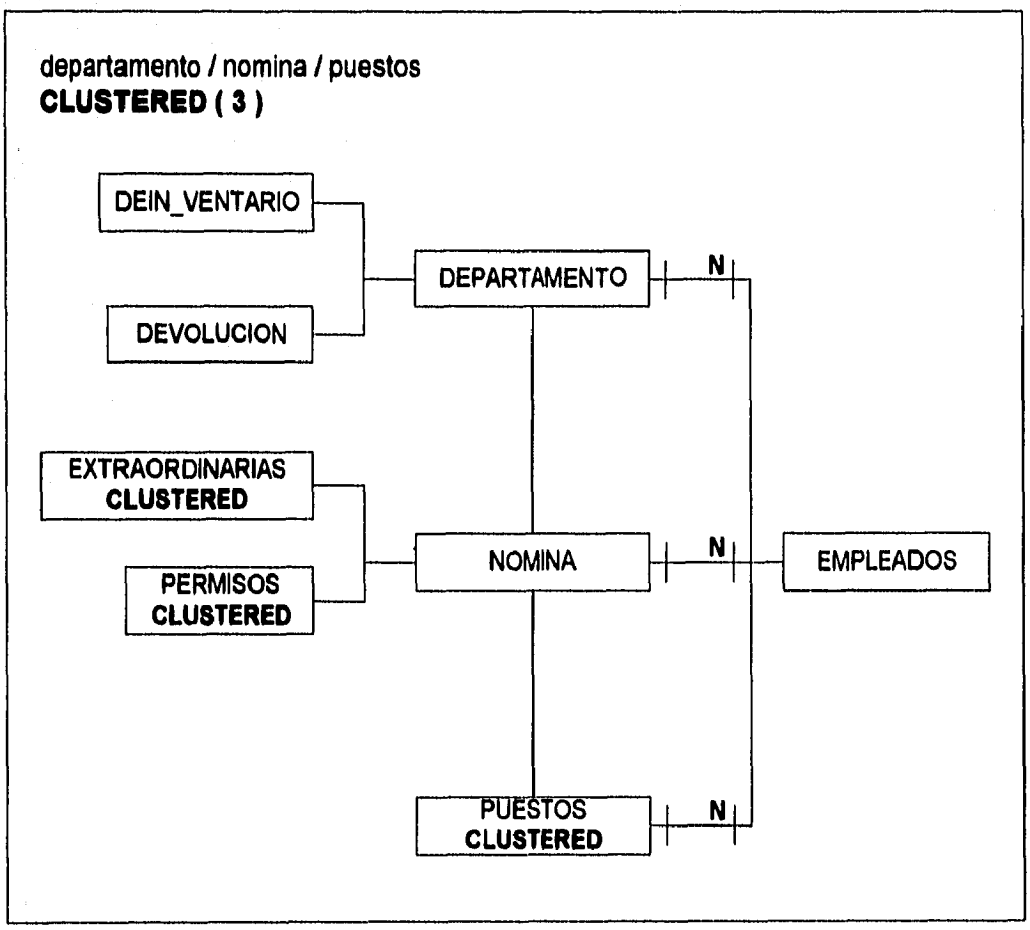


Figura b.2 Integración del diseño lógico.

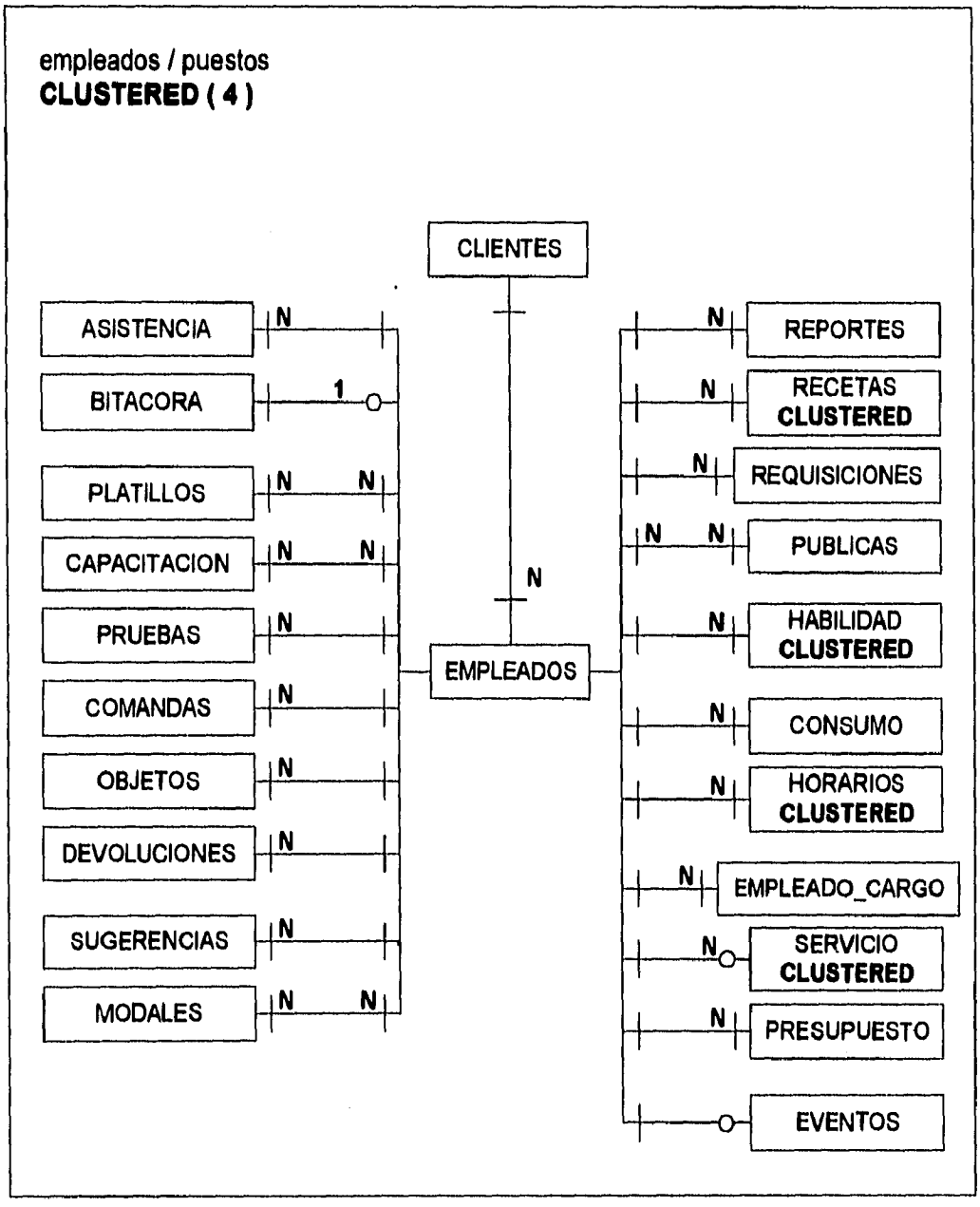


Figura b.3 Integración del diseño lógico.

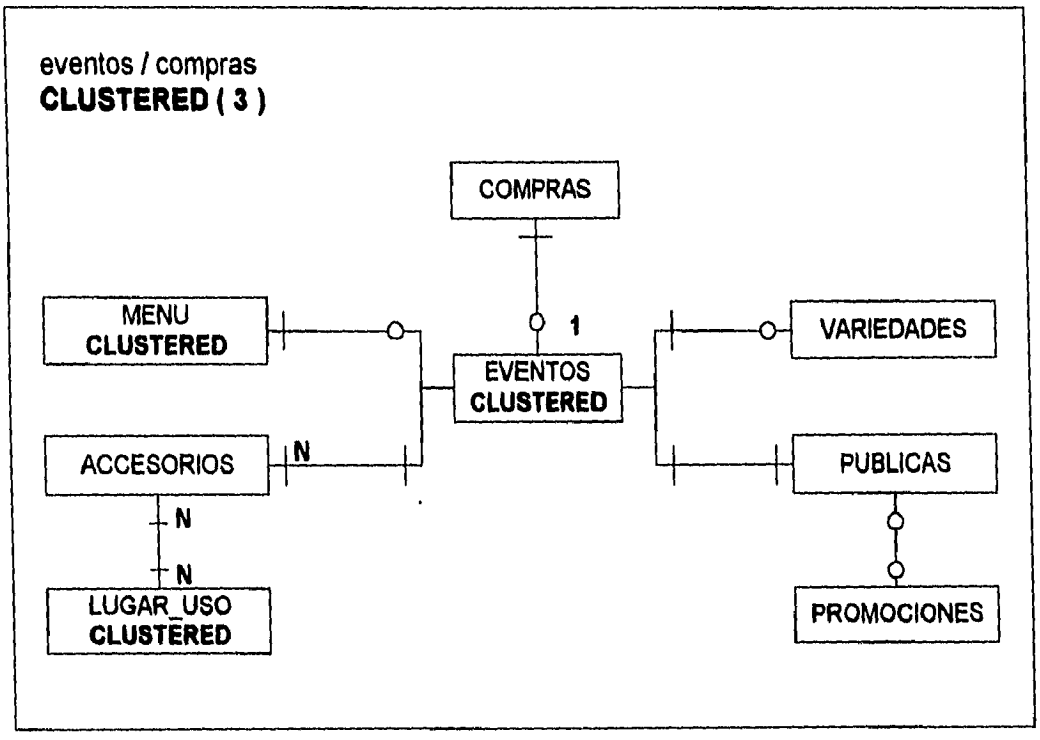


Figura b.4 Integración del diseño lógico.

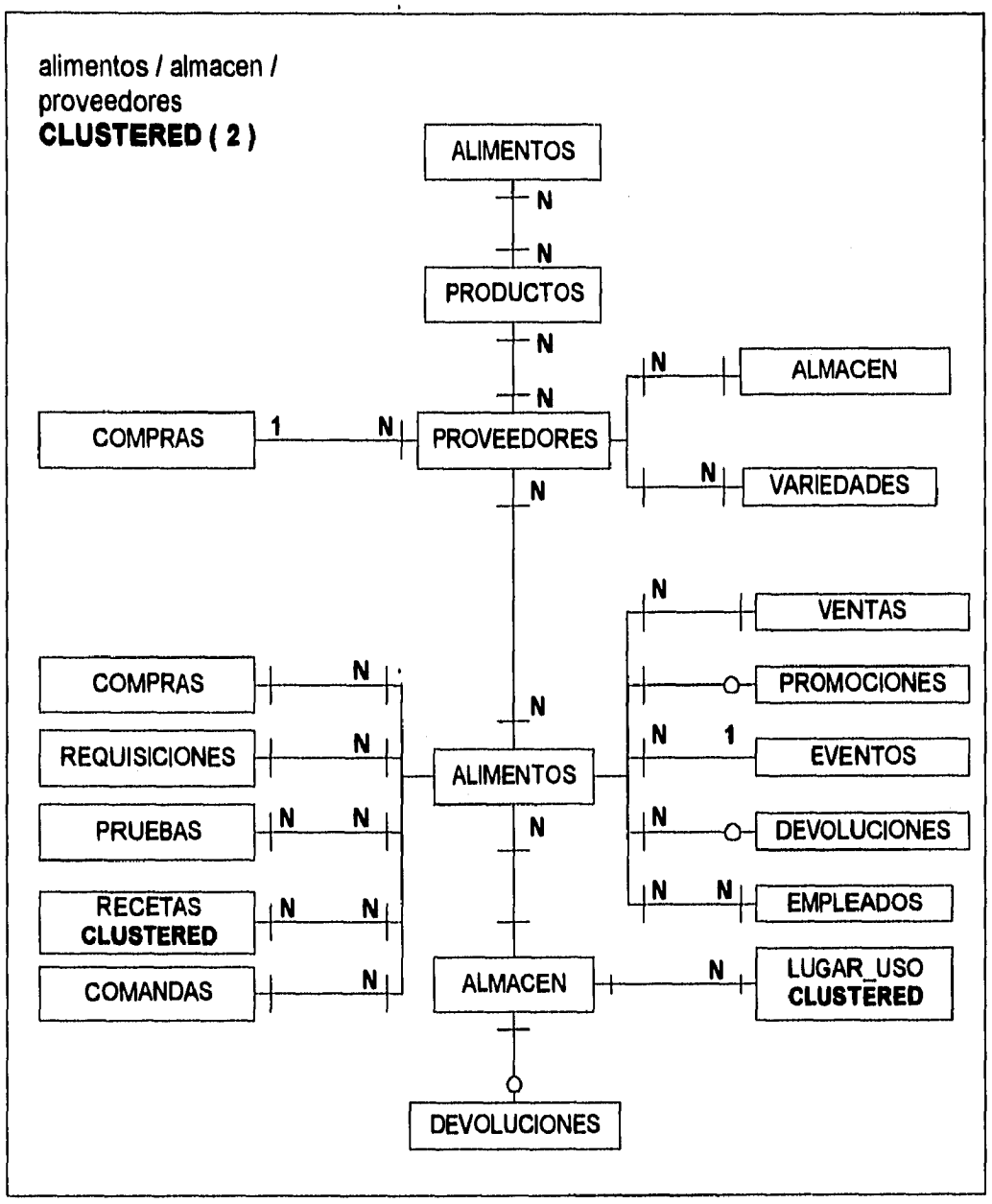


Figura b.5 Integración del diseño lógico.

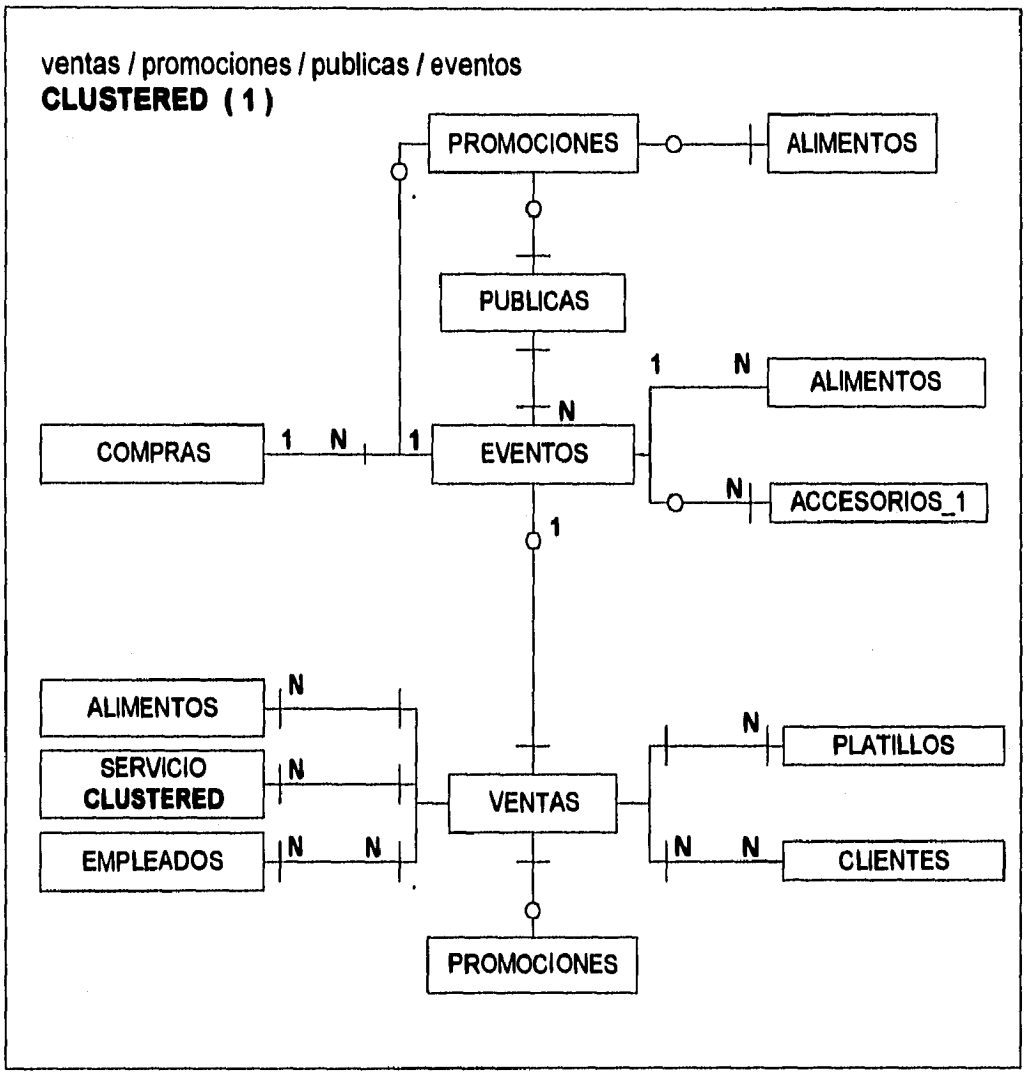


Figura b.6 Integración del diseño lógico.

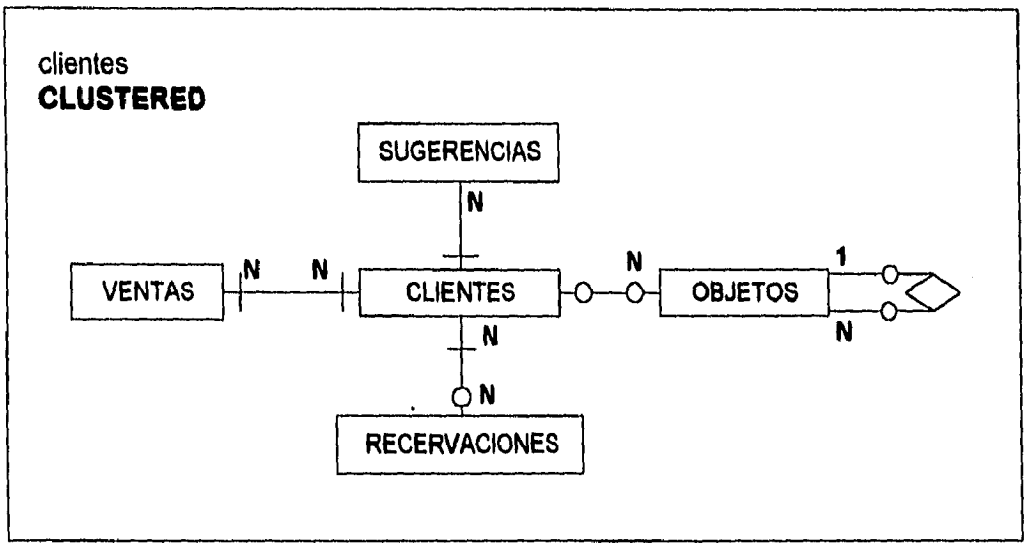


Figura b.7 Integración del diseño lógico.

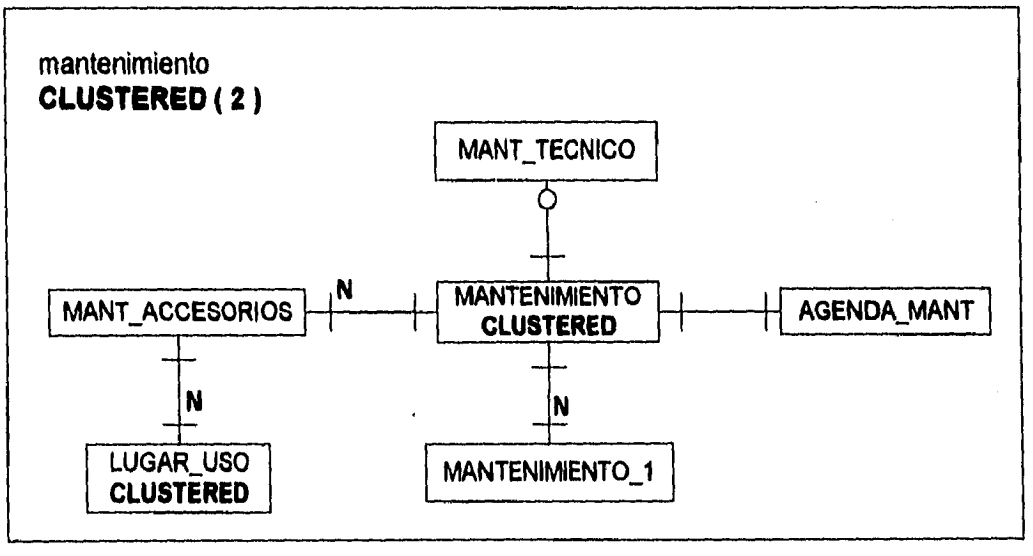


Figura b.8 Integración del diseño lógico.

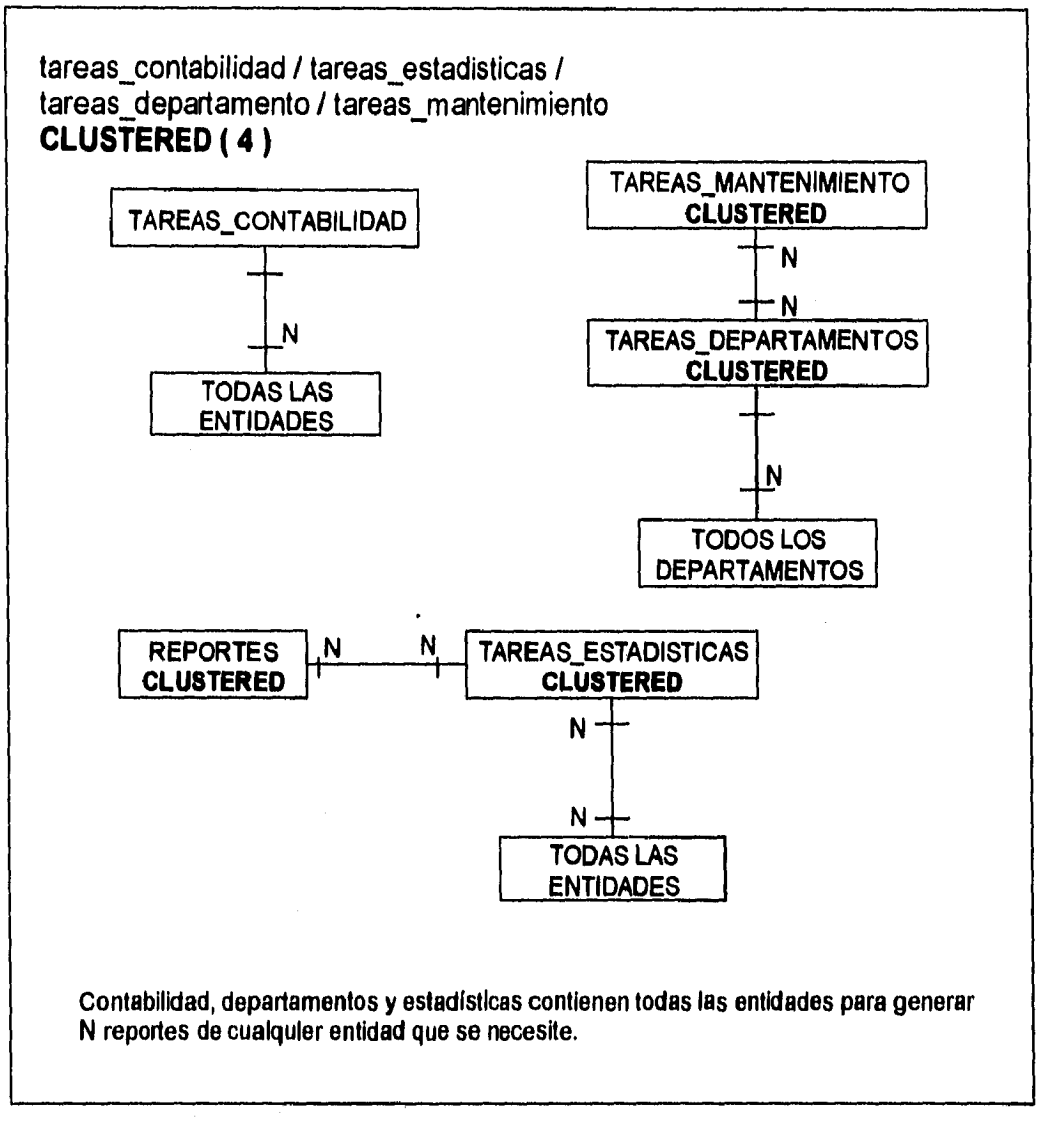


Figura b.9 Integración del diseño lógico.

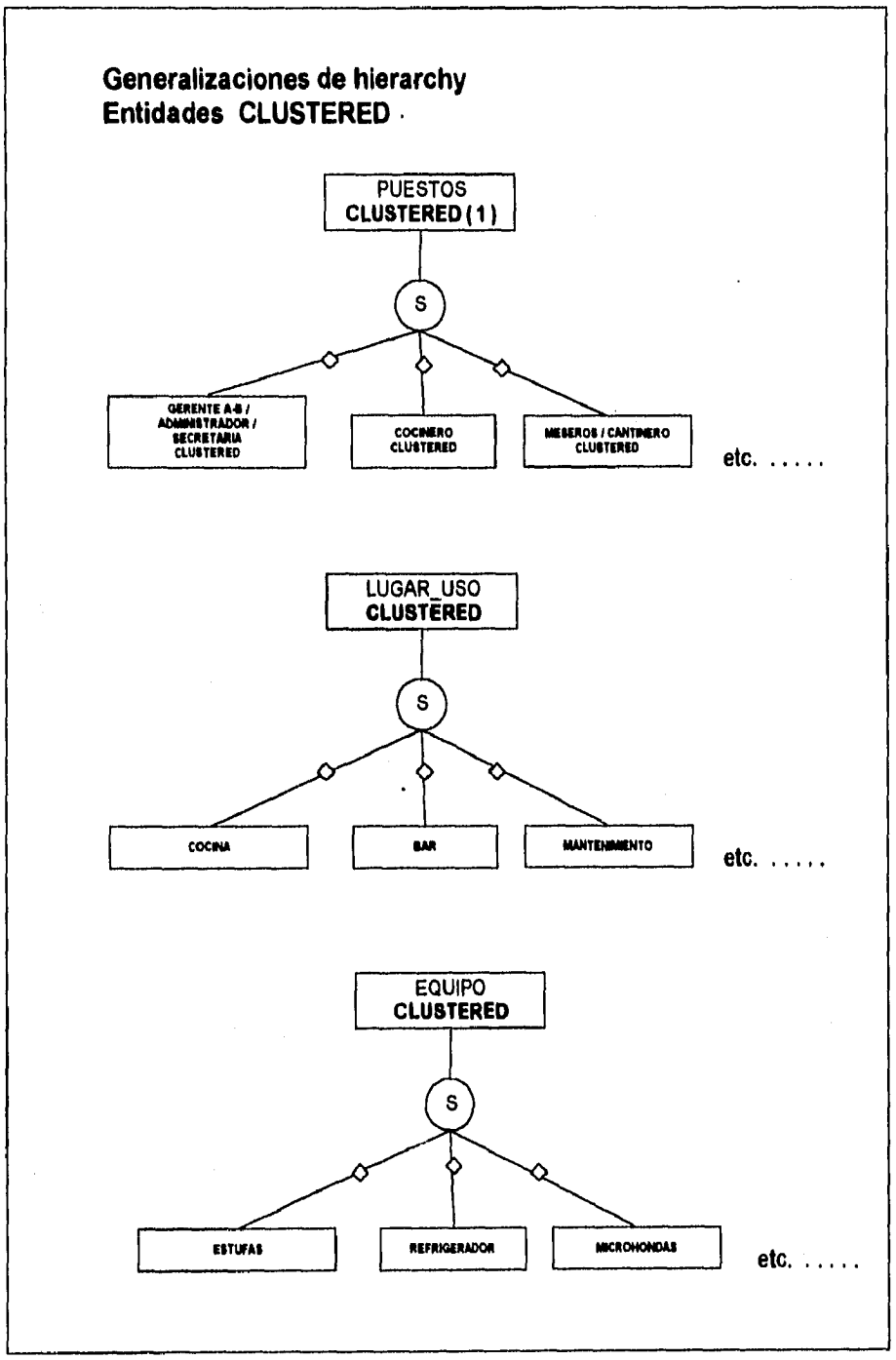


Figura b.10 Integración del diseño lógico.

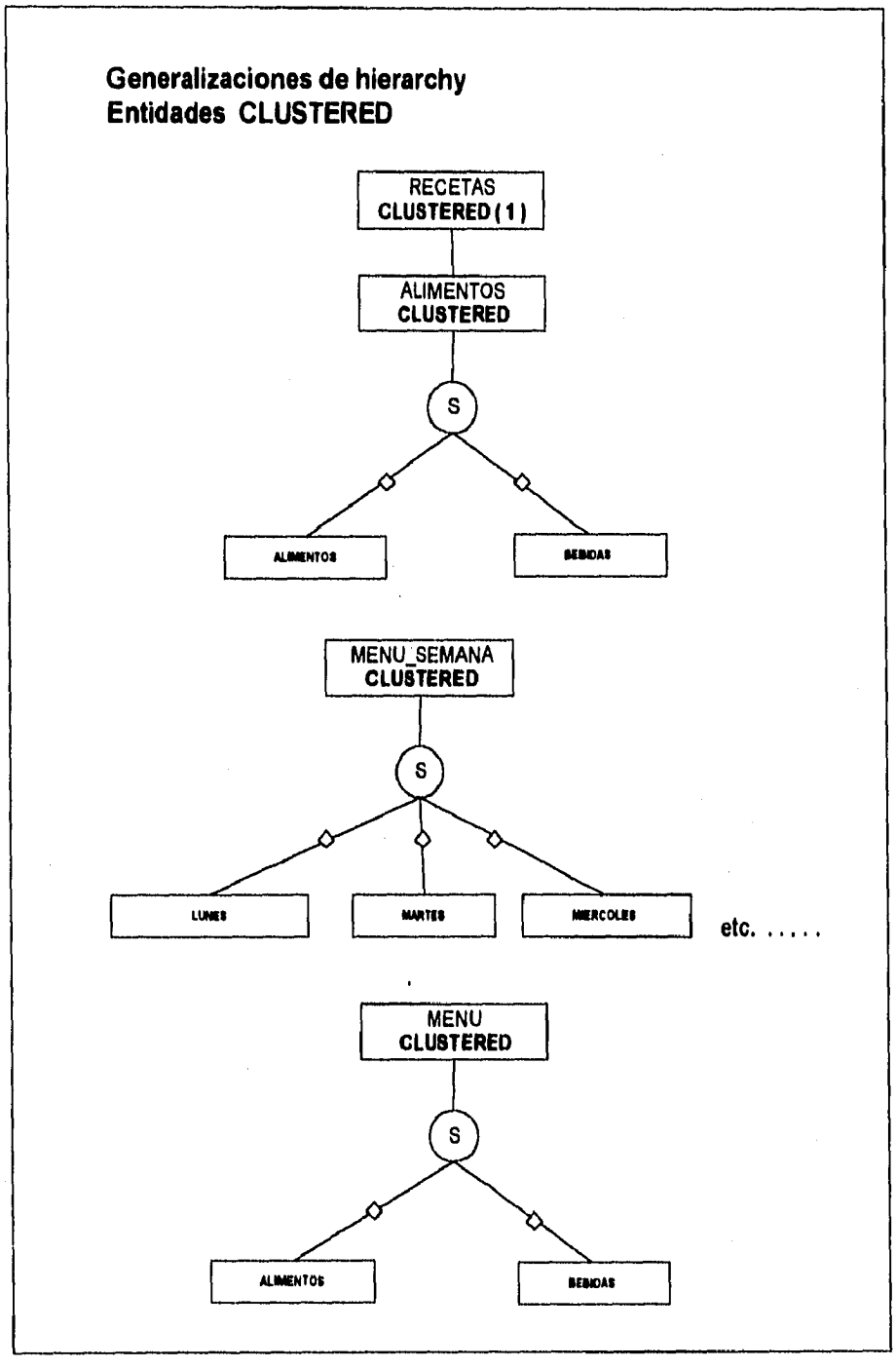


Figura b.11 Integración del diseño lógico.

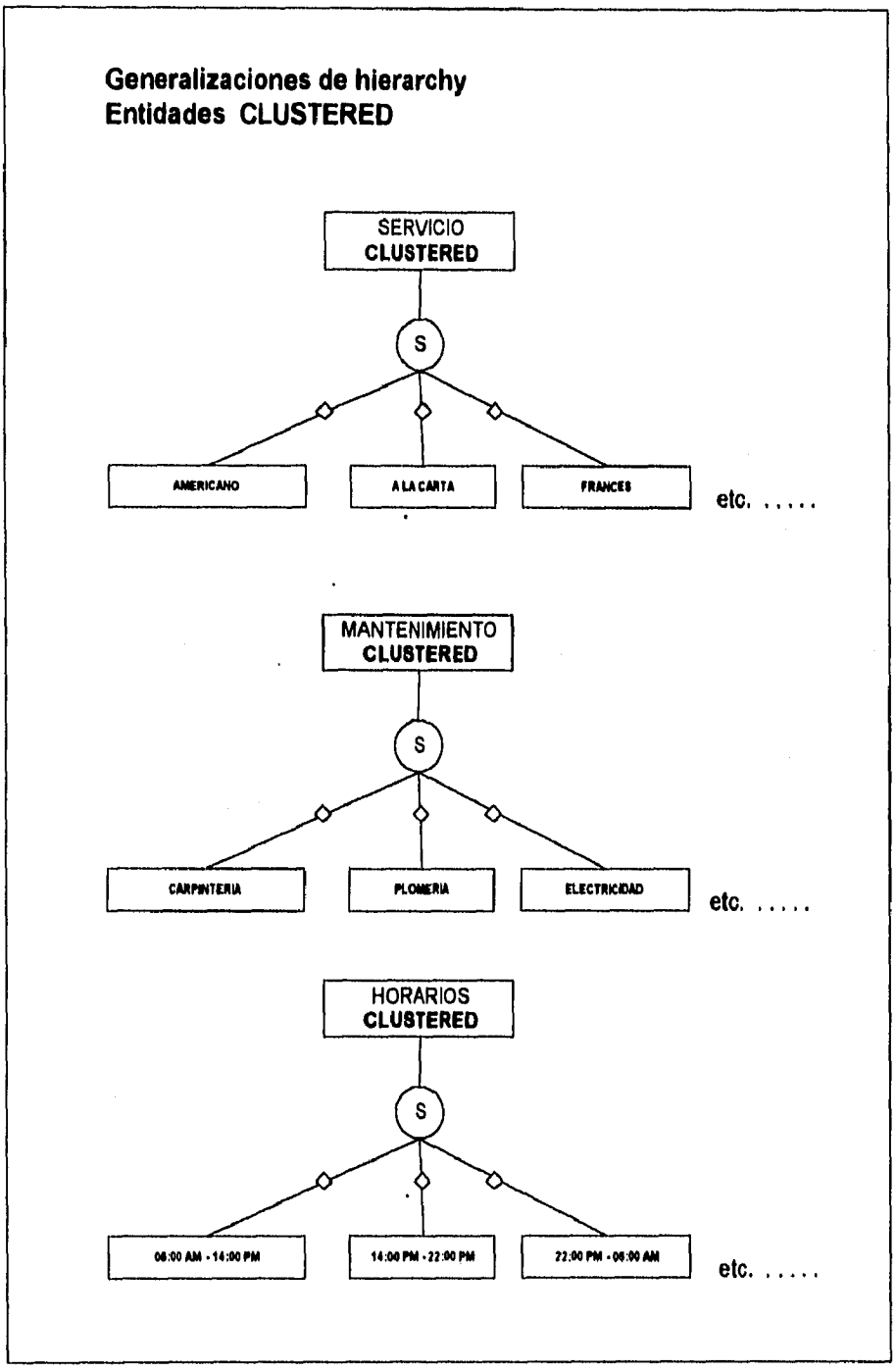


Figura b.12 Integración del diseño lógico.

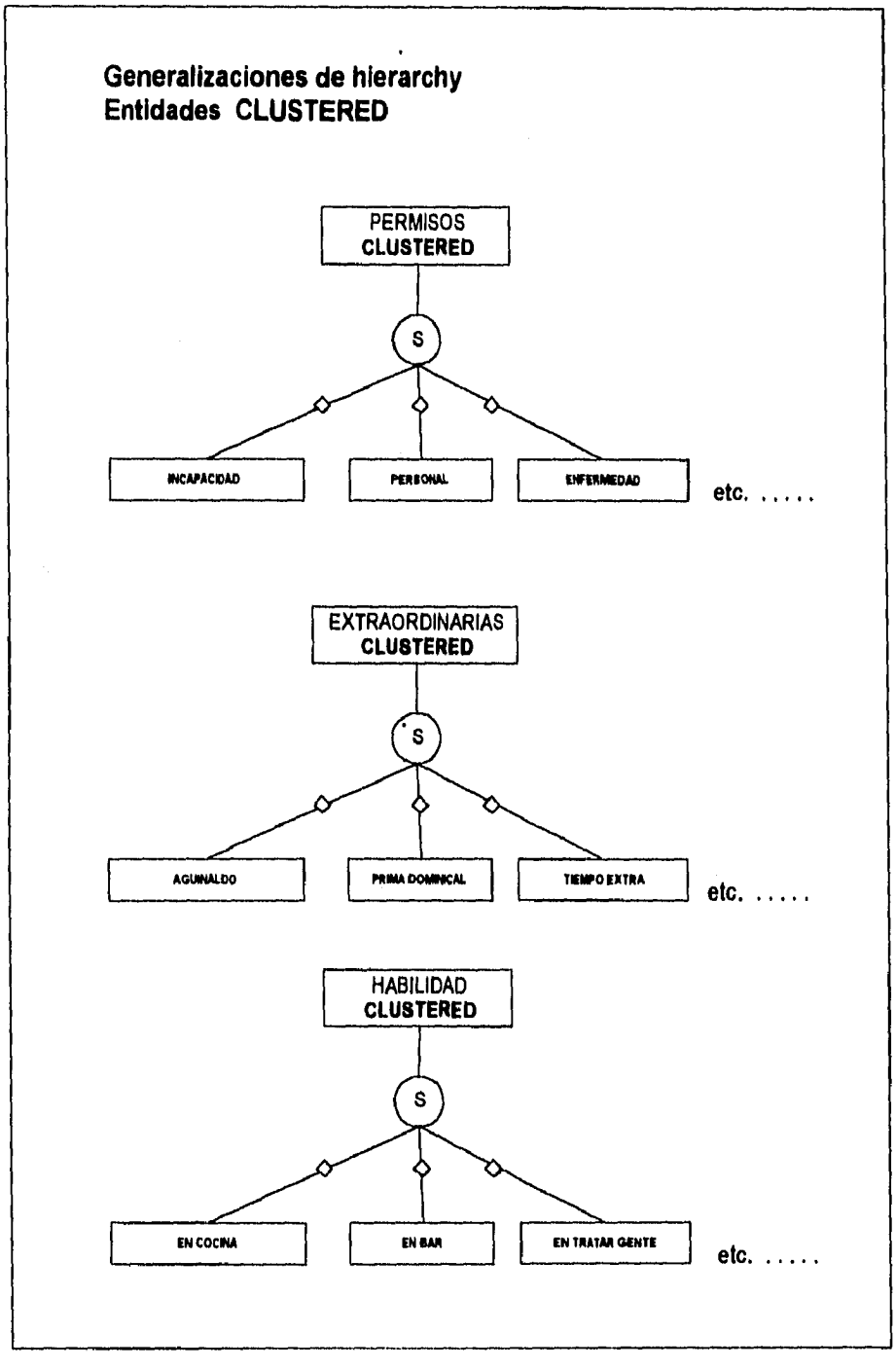


Figura b.13 Integración del diseño lógico.

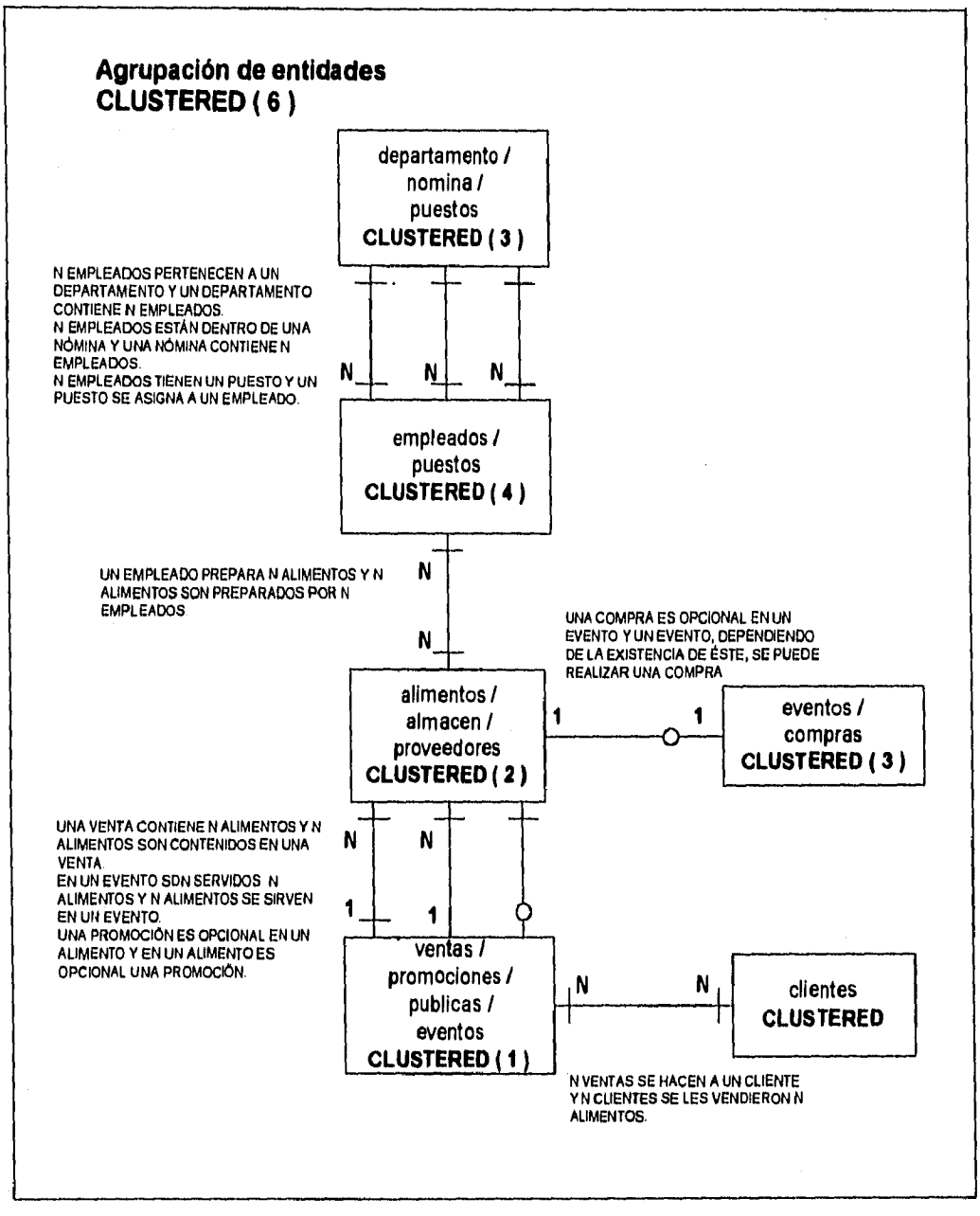


Figura b.14 Integración del diseño

En las anteriores figuras se puede ver la integración del diseño lógico, por medio del cual se solucionarán los requerimientos de información planteados por los usuarios en el análisis.

Apéndice c)

Descripción de la Base de Datos

APÉNDICE C

Descripción de la base de datos

Descripción corta	Alias	Nombre	tipo	Requ	Descripción larga
Clientes	**CL**	**clientes**	**tabla**		**CATALOGO DE CLIENTES**
clave del cliente (PK)		cl_id	char (6)	SR	identificador único del cliente
clave empleado (FK)		em_id	char (6)	SR	identificador único del empleado que atendió al cliente
cliente distinguido		cl_dorado	bit	SR	identifica si el cliente es distinguido (0 / 1) 1 distinguido
nombre		cl_nombre	varchar (40)	SR	nombre o razón social del cliente
dirección		cl_direccion	varchar (40)	SR	dirección del cliente
ciudad		cl_ciudad	varchar (20)	SR	clave de la ciudad
telefono		cl_telefono	varchar (20)	SR	teléfono del cliente
antigüedad		cl_antiguedad	varchar (20)	SR	antigüedad del cliente
frecuencia		cl_frecuencia	varchar (20)	SR	frecuencia de consumo del cliente
profesión		cl_profesion	varchar (20)	SR	profesión del cliente
RFC		cl_rfc	varchar (13)	SR	RFC del cliente
alergia		cl_alergia	varchar (50)	SR	si el cliente padece alguna alergia con algún alimento
dieta		cl_dieta	varchar (50)	SR	si el cliente tiene alguna dieta en especial
fecha ingreso		cl_fingreso	datetime	SR	fecha en que se dio de alta al cliente
fecha baja		cl_fbaja	datetime	SR	fecha en la que se dio de baja al cliente
Empleados	**EM**	**empleados**	**tabla**	Requ	**CATALOGO DE EMPLEADOS**
clave del empleado (PK)		em_id (PK)	char (6)	SR	identificador único del empleado
clave del dpto. (FK)		de_id	char (6)	SR	identificador único del departamento donde labora el emp
clave de rel. pub. (FK)		pu_id	char (6)	SR	identificador único del tipo de relación publica utiliza el emp
clave de habilidad (FK)		hab_id (m)	char (6)	SR	identificador único de la habilidad que tiene el empleado
empleado depende (FK)		emc_id (m)	char (6)	SR	identificador único del empleado jefe de otro empleado
clave de consumo (FK)		cons_id	char (6)	SR	identificador único del consumo de alimentos del emp
clave del modal (FK)		mo_id	char (6)	SR	identificador único del modal del empleado
clave del servicio (FK)		se_id (m)	char (6)	SR	identificador único del servicio que da al empleado
clave del esposo		em_esposo_id	char (6)	SR	identificador único del esposo (a) , de un empleado
nombre		em_nombre	varchar (40)	SR	nombre del empleado
dirección		em_direccion	varchar (40)	SR	dirección donde vive el empleado
ciudad		em_ciudad	varchar (20)	SR	ciudad del empleado
teléfono		em_telefono	varchar (20)	SR	teléfono del empleado
RFC		em_rfc	varchar (13)	SR	RFC del empleado
experiencia		em_experiencia	varchar (20)	SR	experiencia del empleado
pasaporte		em_pasaporte	varchar (15)	SR	pasaporte del empleado
escolaridad		em_escolaridad	varchar (20)	SR	escolaridad del empleado
licencia		em_licencia	varchar (9)	SR	numero licencia del empleado
idiomas		em_idiomas	varchar (50)	SR	idiomas que habla el empleado
motivación		em_motivacion	varchar (50)	SR	tipo de motivación para el empleado
presentación		em_presentacion	varchar (50)	SR	presentación de empleado al trabajo
estado civil		em_edocivil	varchar (10)	SR	estado civil del empleado
Empleado a cargo	**EMP**	**empleado_cargo**	**tabla**	Requ	**CATALOGO DE EMPLEADOS SUBORDINADOS**
clave del jefe (PK)		emc_id (m)	char (6)	SR	identificador único del empleado (jefe de otro empleado)
clave del subord. (FK)		em_id	char (6)	SR	identificador único de un empleado subordinado
Habilidad	**HAB**	**habilidad**	**tabla**	Requ	**CATALOGO DE HABILIDADES DEL EMPLEADO**
clave de la hab (PK)		hab_id (m)	char (6)	SR	identificador único de la habilidad del empleado
clave del empleado (FK)		em_id	char (6)	SR	identificador único del empleado que tiene habilidades
nombre de habilidad		hab_nombre	varchar (40)	SR	nombre de la habilidad que tiene el empleado
Consumo	**CONS**	**consumo**	**tabla**	Requ	**CATALOGO DEL CONSUMO DE AB DEL EMP.**
clave del consumo (PK)		cons_id	char (6)	SR	identificador único del consumo de alimentos del emp
clave del empleado (FK)		em_id	char (6)	SR	identificador único del empleado que consume alimentos
clave de la receta (FK)		re_id (m)	char (6)	SR	identificador único de la receta que consume el empleado
Alimentos	**AB**	**alimentos**	**tabla**	Requ	**CATALOGO DE ALIMENTOS Y BEBIDAS**
clave alimento (PK) (FK)		ab_id	char (6)	SR	identificador único del alimento o bebida
clave del proveedor (FK)		pr_id	char (6)	SR	identificador único del proveedor del alimento o bebida
lugar de uso (FK)		uso_id (m)	char (3)	SR	lugar de uso del alimento o bebida
nombre		ab_nombre	varchar (40)	SR	nombre del alimento o bebida (carne de pollo, ras, etc.)
descripción		ab_descripcion	varchar (40)	SR	descripción del alimento o bebida
existencias		ab_existencias	int	SR	existencias del alimento o bebida
fecha de ingreso		ab_fingreso	datetime	SR	fecha de ingreso del alimento o bebida

fecha de caducidad		ab_fcaducidad	datetime	SR	fecha de caducidad del alimento o bebida
año		ab_ano	datetime	SR	año de producción del alimento o bebida
fecha de producción		ab_fproduc	datetime	SR	fecha preparación del alimento o bebida
precio venta		ab_precio	money	SR	precio venta del alimento o bebida
temperatura		ab_temperatura	varchar (20)	SR	temperatura para conservar el alimento
Proveedores	**PR**	**proveedores**	**tabla**	Reque	**CATALOGO DE PROVEEDORES**
clave del proveedor (PK)		pr_id	char (6)	SR	identificador único del proveedor (un pr distribuye N ab.)
clave del alimento (FK)		ab_id	char (6)	SR	identificador único del alimento
compañia		pr_cia	varchar (40)	SR	compañia que representa el proveedor
nombre		pr_nombre	varchar (40)	SR	nombre o razón social del proveedor
dirección		pr_direccion	varchar (40)	SR	dirección del proveedor
ciudad		pr_ciudad	varchar (20)	SR	ciudad en la que radica el proveedor
telefono		pr_telefono	varchar (20)	SR	telefono del proveedor
existencia		pr_existencia	varchar (5)	SR	existencia
pago		pr_pago	varchar (20)	SR	forma de pago al proveedor
tiempo de entrega		pr_tentrega	varchar (30)	SR	tiempo de entrega del producto por parte del proveedor
RFC		pr_rfc	varchar (13)	SR	RFC del proveedor (para entregarlos a Hacienda)
retraso		pr_retraso	varchar (30)	SR	entregas retrasadas
condición		pr_condición	varchar (10)	SR	estado de los alimentos (bueno mal estado)
Productos	**PRD**	**productos**	**tabla**	Reque	**CATALOGO DE PRODUCTOS**
clave del producto (PK)		prd_id	char (6)	SR	identificador único del proveedor
clave del alimento (FK)		ab_id	char (6)	SR	identificador único del alimento o bebida
nombre del alimento		prd_nombre	varchar (40)	SR	nombre del alimento o bebida
costo		prd_cunitario	money	SR	costo unitario del AB
Ventas	**VE**	**ventas**	**tabla**	Reque	**CATALOGO DE VENTAS**
clave de la venta (PK)		ve_id	char (6)	SR	identificador único y consecutivo de la venta
clave del alimento (FK)		ab_id	char (6)	SR	identificador único del alimento o bebida
clave del cliente (FK)		cl_id	char (6)	SR	identificador único del cliente
clave del evento (FK)		ev_id	char (6)	SR	identificador único del evento
clave del empleado (FK)		em_id	char (6)	SR	identificador único del empleado que realizo la venta
clave del platillo (FK)		pl_id	char (6)	SR	identificador único para obtener el nombre del platillo
clave del servicio (FK)		se_id (m)	char (6)	SR	identificador único del tipo de servicio
factura		ve_factura	varchar (10)	SR	numero de factura al realizar la venta
condiciones		ve_condiciones	varchar (10)	SR	condiciones de venta (todas las ventas son en efectivo)
cantidad		ve_cvendida	int	SR	cantidad que se vendió
fecha		ve_fventa	datetime	SR	fecha y hora de la vanta (estadisticas)
quejas		ve_quejas	varchar (30)	SR	quejas al servicio
total		ve_total	money	SR	total a pegar por el clienta
precio de venta		ve_precio	money	SR	precio de venta del ab (precio unitario X 500 %)
Compras	**CO**	**compras**	**tabla**	Reque	**CATALOGO DE COMPRAS**
clave de la compra (PK)		co_id	char (6)	SR	identificador único y consecutino de la compra
clave del proveedor (FK)		pr_id	char (6)	SR	identificador único del proveedor
clave del alimento (FK)		ab_id	char (6)	SR	identificador único del alimento o bebida
cleva del evento (FK)		ev_id	char (6)	SR	identificador único del evento
descripción		co_dascripcion	varchar (40)	SR	descripción de la compra
condiciones		co_condiciones	varchar (10)	SR	condicionas de la compra (contado o efectivo)
garantia		co_garantia	varchar (20)	SR	garantía de lo comprado
fecha de la compra		co_fcompra	datetime	SR	fecha de la compra
calided		co_calidad	varchar (20)	SR	calidad del producto
precio competitivo		co_competencia	money	SR	precio de compatencia
total a pagar		co_total	money	SR	total a pagar en le compra
compras diarias		co_diarias	verchar (200)	SR	lista de compras diarias
Relaciones publicas	**PU**	**publicas**	**tabla**	Reque	**CATALOGO DE RELACIONES PUBLICAS**
clave de la relación (PK)		pu_id	char (6)	SR	identificador único del tipo de relación
clave de la promoción (FK)		p_id	char (6)	SR	identificador único del tipo de promoción
clave del evento (PK)		ev_id	char (6)	SR	identificador único del evanto
folletos		pu_folletos	varchar (30)	SR	publicidad en folletos para un evento
tarjetas		pu_tarjetas	varchar (30)	SR	tarjetas de invitación
personales		pu_invitecionper	varchar (30)	SR	invitaciones personales
regalos		pu_regalos	varchar (30)	SR	regalos a los clientes
notas		pu_notas	varchar (30)	SR	notas referentes a una determinada publicidad
agradecimientos		pu_agradecimvanto	varchar (30)	SR	agradecimientos a clientes

cortesíes		pu_cortecias	varchar (30)	SR	cortesías qua se otorgan
Promociones	**P**	**promociones**	**tabla**	Reque	**CATALOGO DE PROMOCIONES**
clave de la promo (PK)		p_id	char (6)	SR	identificador único de la promoción
clave de la venta (FK)		ve_id	char (6)	SR	identificador de la venta
clave del alimento (PK)		ab_id	char (6)	SR	identificador del alimento o bebida promocionado
nombre		p_nombre	varchar (40)	SR	nombre de la promoción
horario		p_horario	varchar (20)	SR	horario de la promoción
porcentaje		p_porcentaje	tinyint	SR	porcentaje de descuento
cortesía		p_cortesia	varchar (20)	SR	cortesía de un consumo extra para el cliente
fecha inicio		p_finicial	datetime	SR	fecha de inicio de la promoción
fecha final		p_ffinal	datatime	SR	fecha final de la promoción
Eventos	**EV**	**eventos**	**tabla**	Reque	**CATALOGO DE EVENTOS**
clave del evento (PK)		ev_id	char (6)	SR	identificador único y consecutivo del evento
clave del alimento (FK)		ab_id	char (6)	SR	identificador único del alimento o bebida
clave de accesorios (FK)		ac_id	char (6)	SR	identificador único de los accesorios a utilizar en el evento
clave de la promo (FK)		p_id	char (6)	SR	identificador único de la promoción para el evento
clave de la descrip. (FK)		ev1_id (m)	char (6)	SR	identificador único que para la descripción del evento
clave del menú (FK)		me_id	char (6)	SR	identificador único que proporciona el menú del evento
clave de la variedad (FK)		va_id	char (6)	SR	identificador único del tipo de variedad para el evento
clave quien monto (FK)		em_id	char (6)	SR	identificador único del empleado que monto el evento
capacidad		ev_capacidad	varchar (40)	SR	capacidad del evento
fecha y hora		ev_fecha	datetime	SR	fecha y hora del evento
decoración		ev_decoracion	varchar (60)	SR	tipo de decoración para el evento (servilletas, globos)
música		ev_musica	varchar (60)	SR	tipo de música para el evento
costo		ev_costo	money	SR	costo del evento
días de ocupación		ev_ocupacion	varchar (60)	SR	días en los que tenemos eventos
cancelado		ev_cancelado	bit	SR	(0/ 1) 1 si el evento se cancelo
fecha cancelado		ev_fcancelado	datetime	SR	fecha en que se cancelo el evento
condiciones		ev_condiciones	varchar (20)	SR	condiciones en que se desarrolla el evento
Eventos_1	**EV1**	**eventos_1**	**tabla**	Reque	**CATALOGO NOMBRE DE EVENTOS**
clave del nombre (PK)		ev1_id (m)	char (6)	SR	identificador único del nombre del evento
nombre		ev1_nombre	varchar (40)	SR	nombre del evento (boda, bautizo, etc, .)
variedades	**VA**	**variedades**	**tabla**	Reque	**CATALOGO DE VARIEDADES**
clave de la varied (PK)		va_id	char (6)	SR	identificador único de la variedad
clave del prov. (FK)		pr_id	char (6)	SR	identificador único del proveedor de la variedad
nombre		va_nombre	varchar (40)	SR	nombre de la variedad
Lugar de uso	**USO**	**lugar_uso**	**tabla**	Reque	**LUGAR DE USO DE UN ARTICULO, AB**
clave uso (PK)		uso_id (m)	char (3)	SR	
lugar de uso		uso_nombre	varchar (40)	SR	nombre del lugar da uso cocina
Accesorios	**AC**	**Accesorios**	**tabla**	Reque	**Tabla de accesorios**
clave del accesorio (PK)		ac_id	char (6)	SR	identificador único del accesorio
lugar de uso (FK)		uso_id (m)	char (3)	SR	identificador único del lugar de uso del accesorio
accesorios evento		ac_evento	varchar (100)	SR	accesorios para el evento
nombre del accesorio		ac_nombre	varchar (40)	SR	nombre del accesorio (copas, manteles, flores)
cantidad mínima		ac_cminima	int	SR	cantidad mínima en existencia
cantidad máxima		ac_cmaxima	int	SR	cantidad máxima
cantidad existente		ac_existencia	int	SR	cantidad existente del accesorio
bajas		ac_bajas	int	SR	cantidad dada de baja
accesorios_1	**AC1**	**accesorios_1**	**tabla**	Reque	**TABLA DE ACC. PARA EVENTOS**
acc en evento (PK) (PK)		ac1_id (m)	char (6)	SR	identificador único del accesorio para el evento
clave del evento (FK)		ev_id	char (6)	SR	identificador único del evento
total		ac1_total	int	SR	total de accesorios para el avento
Equipo	**EQ**	**equipo**	**tabla**	Reque	**TABLA DE INVENTARIO DE EQUIPO**
clave del equipo (PK)		eq_id (m)	char (6)	SR	identificador único del equipo
clave del uso (FK)		uso_id (m)	char (3)	SR	identificador único del lugar de uso
nombre		eq_nombre	varchar (40)	SR	nombre del equipo (estufa, refrigerador, platos, etc)

descripcion		eq_descripcion	varchar (40)	SR	descripcion del equipo
cantidad mínima		eq_cminima	int	SR	cantidad mínima del equipo
cantidad máxima		eq_cmaxima	int	SR	cantidad máxima del equipo
existencias		eq_existencias	int	SR	existancias
fecha de compra		eq_fcompra	datetime	SR	fecha de compra del equipo
facha de caducidad		eq_fcaducidad	datetime	SR	fecha de caducidad del aquipo
bajas		eq_bajas	varchar (10)	SR	bajas de equipo por perdida o por desuso

Almacen	**AL**	**almacen**	**tabla**	Reque	**CATALOGO DE ALMACEN**
clave ab (PK) (FK)		al id	char (6)	SR	identificador único del producto en el almacen
clave del proveedor (FK)		pr id	char (6)	SR	identificador único del proveedor del producto
clave de la devoluc (FK)		dev id	char (6)	SR	identificador único de la devolución
lugar de uso (FK)		uso id (m)	char (3)	SR	lugar de uso del producto
tipo de alimento		tipo id (m)	char (2)	SR	tipo de alimento (percedero o no perecedero)
descripción		al descripcion	varchar (40)	SR	descripción del producto
existencias		al existencias	int	SR	existencias del producto
fecha de ingreso		al fingreso	datetime	SR	fecha de ingreso producto
fecha de salida		al fsalida	datetime	SR	fecha de salida del producto
fecha de caducidad		al fcaducidad	datetime	SR	facha de caducidad del producto
fecha de producción		al fproduccion	datetime	SR	fecha de producción del alimento o bebida
cantidad mínima		al cminima	int	SR	cantidad mínima del producto
cantidad máxima		al cmaxima	int	SR	cantidad máxima del producto
clave del tipo		tipo id (m)	char (2)	SR	identificador único dal tipo de alimento
tipo de alimento		tipo nombre	varchar (15)		tipo de alimento (perocedero o no perecedero)

Recetas	**RE**	**recetas**	**tabla**	Reque	**CATALOGO DE RECETAS**
clave de la receta (PK)		re id (m)	char (6)	SR	identificador único de la receta
clave en el almacen (FK)		al id	char (6)	SR	identificador único del alimento o bebida que identifica
clave del empleado (FK)		em id	char (6)	SR	identificador único del empleado que elaboro la receta
clave lugar de uso (FK)		uso id (m)	char (3)	SR	identificador único del lugar de uso
nombre de la receta		re nombre	varchar (40)	SR	nombre de la receta
descripcion de la receta		re descripcion	varchar (40)	SR	descripcion de la receta (modo de preparar)
precio venta		re pventa	money	SR	precio venta del platillo
ingredientes de la receta		re ingredientes	varchar (100)	SR	ingredientes de la receta
cantidad de ingredientes		ra cantidad	int	SR	cantidad de ingredientes
fecha de caducidad		re fcaducidad	datetime	SR	fecha de caducidad de la receta
fecha de elaboración		re fecha	datetime	SR	fecha en que se elaboro la receta
costo		re costo	money	SR	costo que tiene elaborar una receta (arbitrariamente)

Menús	**ME**	**menu**	**tabla**	Reque	**CATALOGO DE MENÚS**
clave del menú (PK)		me id	char (6)	SR	identificador único del menú
clave día semana (FK)		men id (m)	char (6)	SR	identificador único del día semana sirve ese menú
clave quien eleboro (FK)		em id	char (6)	SR	identificador único del empleado que elaboro el menu
nombre		me nombre	varchar (40)	SR	nombre del menú
descripcion		me descripcion	varchar (100)	SR	descripcion del contenido del menú
costo		me costo	money	SR	costo del menú

Menú semana	**MEN**	**menu semana**	**tabla**	Reque	**CATALOGO DE DÍAS SEMANA PARA EL MENÚ**
clave día semana (PK)		men id (m)	char (6)	SR	identificador único del día semana en que se sirve el menú
día de la semana		men dia	varchar (10)	SR	nombre del día de la semana en que se sirve el menú

Departamento	**DE**	**departamento**	**tabla**	Reque	**CATALOGO DE DEPARTAMENTOS**
clave del dpto. (PK)		de id	char (6)	SR	identificador único del departamento
numero de empleados		de noempleados	varchar (10)	SR	numero de empleados en el departamento
nombre		de nombre	varchar (20)	SR	nombre del departamento
costo		de costo	money	SR	costo del departamento
música		de musica	varchar (40)	SR	música ambiental en el departamento
horario		de horario	varchar (10)	SR	horario an el que trabaja este departamento

Inventario dpto.	**DEIN**	**dein ventario**	**tabla**	Reque	**CATALOGO DE INVENTARIO EN DPTOS.**
invent. físico (PK) (FK)		dein id	char (6)	SR	identificador único del departamento
inventario físico		dein fisico	varchar (100)	SR	inventario físico an el departamento
cantidad máxima		dein cmaxima	int	SR	cantidad máxima en el departamento
cantidad mínima		dein cminimo	int	SR	cantidada mínima en el departamento

reservaciones	**RES**	**reservaciones**	**tabla**	Reque	**CATALOGO DE RESERVACIONES**
clave reservación (PK)		res id	char (6)	SR	identificador único de la reservación
clave del cliente (FK)		cl id	char (6)	SR	identificador único del cliente que reserva
frecuencia de reserva		res frecervaciones	varchar (30)	SR	frecuencia de reservaciones del cliente

cancelada		res_cancelada	bit	SR	si la reservación se cancela o no (0 / 1) 1 = cancelado
Horarios	**HO**	**horarios**	**tabla**	Reque	**CATALOGO DE HORARIOS**
clave del horario (PK)		ho_id (m)	char (6)	SR	identificador único del horario
clave del empleado (FK)		em_id	char (6)	SR	identificador único para el empleado
clave horario dpto. (FK)		de_id	char (6)	SR	identificador único del dpto. para conocer su horario
descanso		ho_descanso	varchar (20)	SR	días de descanso del empleado
vacaciones		ho_vacaciones	varchar (50)	SR	periodo de vacaciones del empleado
permisos dados al emp.		ho_permisos	varchar (50)	SR	permisos dados al empleado
aseo del empleado		ho_aseo	varchar (50)	SR	aseo del empleado
disciplina del empleado		ho_disciplina	varchar (50)	SR	conducta del empleado
Asistencia	**AS**	**asistencia**	**tabla**	Reque	**CATALOGO DE ASISTENCIA DEL EMP**
empleado asist (PK) (FK)		em_id	char (6)	SR	identificador único del empleado para ver su asistencia
asistencia		as_asistencia	bit	SR	asistencia del empleado 0 / 1 | 1 = falta
fecha de asistencia		as_fasistencia	datetima	SR	fecha de asistencia
Requisiciones	**REQ**	**requisiciones**	**tabla**	Reque	**CATALOGO DE REQUISICIONES (AL)**
clave de requisición (PK)		req_id	char (6)	SR	identificador único y consecutivo de la requisición
clave del alimento (FK)		ab_id	char (6)	SR	identificador único del alimento o bebida
clave del empleado (PK)		em_id	char (6)	SR	clave del empleado que hace la requisición
descripcion		req_descripcion	varchar (60)	SR	descripcion de los artículos
cantidad		req_cantidad	varchar (100)	SR	cantidad de cada artículos
total		req_total	int	SR	total de artículos solicitados
Pruebas	**PRU**	**pruebas**	**tabla**	Reque	**CATALOGO DE PRUEBAS**
clave de prueba (PK)		pru_id	char (6)	SR	identificador único de la prueba
clave del alimento (FK)		ab_id	char (6)	SR	identificador único del alimento al que se aplica la prueba
clave del empleado (FK)		em_id	char (6)	SR	identificador único del empleado que hizo la prueba
rendimiento kilo		pru_rend_kilo	varchar (30)	SR	rendimiento por kilo
rendimiento litro		pru_rend_litro	varchar (30)	SR	rendimiento por litro
condiciones		pru_condiciones	varchar (30)	SR	condiciones de la prueba
tiempo		pru_tiempo	varchar (30)	SR	tiempo de prueba
papel donde se toman las ordenes					
Comandas	**COM**	**comandas**	**tabla**	Reque	**CATALOGO DE COMANDAS**
clave de comanda (PK)		com_id	char (6)	SR	identificador único de la comanda y numero consecutivo
clave del alimento (FK)		ab_id	char (6)	SR	identificador único para conocer el nombre del ab
clave del alimento (FK)		al_id	char (6)	SR	identificador único para conocer la descripcion del ab
quien estregó (FK)		com_ent_em_id	char (6)	SR	empleado que entrego comanda
quien autorizo (FK)		com_aut_em_id	char (6)	SR	empleado que autorizo la comanda
cantidad del alimento		com_ab_cantidad	int	SR	cantidad del alimento o bebida
Objetos	**OB**	**objetos**	**tabla**	Reque	**CATALOGO DE OBJETOS OLVIDADOS**
clave del objeto (PK)		ob_id	char (6)	SR	identificador único del objeto
lugar de uso (FK)		uso_id (m)	char (3)	SR	identificador único del lugar donde se olvido el objeto
clave del empleado (FK)		em_id	char (6)	SR	identificador único del empleado que entrego el objeto
descripcion		ob_descripcion	varchar (40)	SR	descripcion del objeto
fecha de ingreso		ob_fingreso	datetime	SR	fecha de ingreso del objeto al departamento
fecha de devolución		ob_fdevolucion	datetime	SR	fecha de devolución del objeto a su propietario
como se identifica		ob_identifica	varchar (40)	SR	con que se identifico el propietario del objeto
tipo de servicio que brinda el rest.					
Servicio	**SE**	**servicio**	**tabla**	Reque	**CATALOGO DE SERVICIOS**
clave del servicio (PK)		se_id (m)	char (6)	SR	identificador único del servicio
nombre del servicio		se_nombre	varchar (20)	SR	nombre del servicio
descripcion del servicio		se_descripcion	varchar (40)	SR	descripcion del servicio
modales que debe tener el empleado					
Modales	**MO**	**modales**	**tabla**	Reque	**CATALOGO DE MODALES**
clave del modal (PK)		mo_id	char (6)	SR	identificador único del comportamiento del empleado
descripcion		mo_descripcion	varchar (30)	SR	descripcion de la manera de comportarce
Capacitación	**CA**	**capacitacion**	**tabla**	Reque	**CATALOGO DE CAPACITACION**
clave de capac. (PK)		ca_id	char (6)	SR	identificador único del tipo de capacitacion

clave del empleado (FK)		em id	char (6)	SR	identificador único del empleado
tipo		ca_tipo	varchar (30)	SR	tipo de capacitacion
fecha		ca_fecha	datetime	SR	fecha de capacitacion
Devolución	**DEV**	**devolucion**	**tabla**	Reque	**CATALOGO DE DEVOLUCIONES**
clave de devolución (PK)		dev id	char (6)	SR	identificador único y consecutivode la dev. al almacen
clave del alimento (FK)		ab id	char (6)	SR	identificador único del alimento o bebida devuelto
clave del empleado (FK)		em id	char (6)	SR	identificador único del empleado que hace la devolución
clave del departa. (FK)		de id	char (6)	SR	identificador único del departamento que hace la devoluc.
descripcion		dev_descripcion	varchar (30)	SR	descripcion del articulo devuelto
cantidad		dev_cantidad	varchar (30)	SR	cantidad devuelta
Sugerencias	**SU**	**sugerencias**	**tabla**	Reque	**CATALOGO DE SUGERENCIAS**
clave de sugerencia (PK)		su id	char (6)	SR	identificador único de la sugerencia
clave del cliente (FK)		cl id	char (6)	SR	identificador único del cliente que hace la sugerencia
clave del empleado (FK)		em id	char (6)	SR	identificador único del empleado que hace la sugerencia
descripcion		su_descripcion	varchar (60)	SR	descripción de la sugerencia
Bitácora	**BI**	**bitacora**	**tabla**	Reque	**CATALOGO DE BITACORA**
clave en la bitacora (PK)		bi id	char (6)	SR	identificador único y consecutivo de la bitácora
clave del empleado (FK)		em id	char (6)	SR	identificador único del emp que redacta la bitácora
descripcion		bi_descripcion	varchar (100)	SR	descripción de la nota
fecha		bi_fecha	datetime	SR	fecha de la nota
Platillos	**PL**	**platillos**	**tabla**	Reque	**CATALOGO DE PLATILLOS**
clave del platillo (PK)		pl id	char (6)	SR	identificador único y numero consecutivo del platillo
clave del empleado (FK)		em id	char (6)	SR	identificador único del empleado que preparo el platillo
nombre		pl_nombre	varchar (30)	SR	nombre del platillo
Puestos	**PUE**	**puestos**	**tabla**	Reque	**CATALOGO DE PUESTOS**
clave puesto (PK)		pue id (m)	char (6)	SR	identificador único del puesto
clave del empleado (FK)		em id	char (6)	SR	identificador único del empleado en un determinado puesto
Puestos1	**PUE1**	**puestos_1**	**tabla**	Reque	**CATALOGO DE NOMBRE DE LOS PUESTOS**
puesto (PK) (FK)		pue1 id (m)	char (6)	SR	Identificador único del nombre del puesto
nombre		pue1_nombre	varchar (30)	SR	nombre del puesto
descripción		pue1_descripción	varchar (50)	SR	descripcion del puesto
Perdidas	**PER**	**perdidas**	**tabla**	Reque	**CATALOGO DE PERDIDAS**
clave de perdida (PK)		per id (m)	char (6)	SR	identificador único de la perdida
clave del alimento (FK)		ab id	char (6)	SR	identificador único del alimento
cantidad		per_cantidad	varchar (20)	SR	cantidad del alimento
costo		per_costo	money	SR	costo de la perdida
Nomina	**NO**	**nomina**	**tabla**	Reque	**CATALOGO DE NOMINA**
clave en la nomina (PK)		no id	char (6)	SR	identificador único de la nomina
clave del empleado (FK)		em id	char (6)	SR	identificador único del empleado
clave del dpto. (FK)		de id	char (6)	SR	clave del departamento donde labora el empleado
clave del puesto (FK)		pue id (m)	char (6)	SR	clave del puesto que tiene dentro de la empresa
clave del permiso (FK)		perm id (m)	char (6)	SR	clave del permiso que se le dio al empleado
clave de gratificación (FK)		ext id (m)	char (6)	SR	gratificaciones extraordinarias
		percepciones			
sueldo		no_sueldo	money	SR	sueldo del empleado
tiempo extra		no_textra	varchar (10)	SR	tiempo extra que se da a los empleados
otras		no_otraspercep	money	SR	otras percepciones
indebidas		no_indebidas	money	SR	deducciones indebidas
		deducciones			
ISPT		no_ispt	money	SR	impuesto sobre el producto trabajado
cuota IMSS		no_imss	money	SR	cuota que se descuenta al empleado IMSS
faltas		no_faltas	int	SR	descuentos por faltas
INFONAVIT		no_infonavit	money	SR	descuentos por crédito en INFONAVIT
deducciones		no_otrasdeduc	money	SR	otras deducciones
indebidas		no_percepindeb	money	SR	percepciones indebidas
sindicato		no_sindicato	money	SR	descuento a pagar al sindicato

vida		no_seguro	money	SR	seguro de vida
préstamo		no_prestamo	money	SR	préstamo que otorga la empresa al empleado
mantenimiento		no_mantinfonavit	money	SR	mantenimiento infonavit
pensión		no_pension	money	SR	pensión alimenticia
total percepciones		no_totalpercep	money	SR	total percepciones
total deducciones		no_totaldeduc	money	SR	total deducciones
total a pagar		no_total	money	SR	total a pagar al empleado
horas		no_horas	varchar (10)	SR	horas por cubrir
ausencias		no_ausencias	int	SR	ausencias por enfermedad
sueldo acumulado		no_sueldoacum	money	SR	sueldo acumulado por empleado
antigüedad		no_antiguedad	varchar (10)	SR	antigüedad del empleado
suspenciones		no_suspencion	varchar (10)	SR	si el empleado a sido suspendido en el trabajo
días		no_dias	int	SR	días trabajados
folio		no_folio	varchar (10)	SR	numero de folio del recibo de nomina
fecha		no_fecha	datetime	SR	fecha de pago
periodo		no_periodo	varchar (40)	SR	periodo de pago 16-31
quincena		no_quincena	varchar (40)	SR	número de quincena que se paga
Extraordinarias	**EXT**	**extraordinarias**	**tabla**	Requie	**CATALOGO DE GRATIF. EXTRAORD.**
clave compensación (PK)		ext_id (m)	char (6)	SR	identificador único de la gratificación extraordinaria
nombre de la gratif. (FK)		ext_nombre	varchar (30)	SR	gratificación (comisiones, prima dominical, aguinaldo,)
Permisos	**PERM**	**permisos**	**tabla**	Requie	**CATALOGO DE PERMISOS**
clave del permiso (PK)		perm_id (m)	char (6)	SR	identificador único del permiso que se da al empleado
clave del empleado (FK)		em_id	char (6)	SR	identificador único del empleado al que se le dio el perm
descripción		perm_descripcion	varchar (30)	SR	descripción de la causa del permiso
Nota :					
De estadistica se obtendrán todos los reportes por lo tanto, debe contener todos					
los identificadores de las tablas.					
Estadísticas	**ES**	**estadistica**	**tabla**	Requie	**CATALOGO DE ESTADÍSTICAS**
clave de acceso		cl_id	char (6)	SR	identificador único para tener datos de clientes
clave de acceso		em_id	char (6)	SR	identificador único para tener datos de empleados
clave de acceso		emp_id	char (6)	SR	identificador único para tener datos de empleados jefes
clave de acceso		ab_id	char (6)	SR	identificador único para tener datos de alimentos y bebidas
clave de acceso		pr_id	char (6)	SR	identificador único para tener datos de proveedores
clave de acceso		prd_id	char (6)	SR	identificador único para tener datos de productos
clave de acceso		ve_id	char (6)	SR	identificador único para tener datos de ventas
clave de acceso		co_id	char (6)	SR	identificador único para tener datos de compras
clave de acceso		pu_id	char (6)	SR	identificador único para tener datos de relaciones publicas
clave de acceso		p_id	char (6)	SR	identificador único para tener datos de promociones
clave de acceso		ev_id	char (6)	SR	identificador único para tener datos de eventos
clave de acceso		ev1_id (m)	char (6)	SR	identificador único para tener datos de nombres de eventos
clave de acceso		va_id	char (6)	SR	identificador único para tener datos de variedades
clave de acceso		uso_id (m)	char (3)	SR	identificador único para tener datos de lugares de uso
clave de acceso		ac_id	char (6)	SR	identificador único para tener datos de accesorios
clave de acceso		ac1_id (m)	char (6)	SR	identificador único para tener datos de acc. para un evento
clave de acceso		eq_id (m)	char (6)	SR	identificador único para tener datos de equipo
clave de acceso		al_id	char (6)	SR	identificador único para tener datos del almacen
clave de acceso		re_id (m)	char (6)	SR	identificador único para tener datos de recetas
clave de acceso		me_id	char (6)	SR	identificador único para tener datos de menús
clave de acceso		men_id (m)	char (6)	SR	identificador único para tener datos del menú para la sem.
clave de acceso		de_id	char (6)	SR	identificador único para tener datos de departamentos
clave de acceso		dein_id	char (6)	SR	identificador único para tener datos de inventarios físicos
clave de acceso		rep_id	char (6)	SR	identificador único para tener datos de reportes
clave de acceso		re1_id (m)	char (6)	SR	identificador único para tener datos de nombres de rep.
clave de acceso		res_id	char (6)	SR	identificador único para tener datos de reservaciones
clave de acceso		ho_id (m)	char (6)	SR	identificador único para tener datos de horarios
clave de acceso		req_id	char (6)	SR	identificador único para tener datos de requisiciones
clave de acceso		pru_id	char (6)	SR	identificador único para tener datos de pruebas de ab
clave de acceso		com_id	char (6)	SR	identificador único para tener datos de comandas
clave de acceso		ob_id	char (6)	SR	identificador único para tener datos de objetos olvidados
clave de acceso		mo_id	char (6)	SR	identificador único para tener datos de modales
clave de acceso		ca_id	char (6)	SR	identificador único para tener datos de capacitación
clave de acceso		dav_id	char (6)	SR	identificador único para tener datos de devoluciones
clave de acceso		su_id	char (6)	SR	identificador único para tener datos de sugerencias
clave de acceso		bi_id	char (6)	SR	identificador único para tener datos de la bitacora
clave de acceso		pl_id	char (6)	SR	identificador único para tener datos de platillos
clave de acceso		pue_id (m)	char (6)	SR	identificador único para tener datos de puestos
clave de acceso		pue1_id (m)	char (6)	SR	identificador único para tener datos de nombres de puestos
clave de acceso		per_id (m)	char (6)	SR	identificador único para tener datos de permisos a emp

clave de acceso		hab_id (m)	char (6)	SR	identificador único para tener datos de habilidades de emp
clave de acceso		per_id	char (6)	SR	identificador único para tener datos de permisos a emp
clave de acceso		ext_id (m)	char (6)	SR	identificador único para tener datos de gratif. extraord.
clave de acceso		as_id	char (6)	SR	identificador único para tener datos de aseo del empleado
Nota :					
Contabilidad al igual que estadísticas tiene acceso a todas las entidades para poder obtener					
información y poder elaborar los procesos propios de esta área.					
Contabilidad	**CON**	**contabilidad**	**tabla**	Reque	**CATALOGO DE CONTABILIDAD**
Presupuesto	**PRE**	**presupuesto**	**tabla**	Reque	**CATALOGO DE PRESUPUESTO**
clave del presup. (PK)		pre_id	char (6)	SR	identificador único del presupuesto
clave del empleado (FK)		em_id	char (6)	SR	identificador único del empleado que elaboro el presup.
clave del dpto. (FK)		de_id	char (6)	SR	identificador único del dpto. al que se asigna el presup.
nombre presupuesto		pre_nombre	varchar (30)	SR	nombre del presupuesto
fecha elaboro		pre_fecha	datetime	SR	fecha en que se elaboro el presupuesto
fecha inicial		pre_finicial	datetime	SR	fecha inicial del presupuesto
fecha final		pre_ffinal	datetime	SR	fecha final del presupuesto
desviación presupuesto		pre_desviacion	tinyint	SR	desviaciones del presupuesto
Reportes	**REP**	**reportes**	**tabla**	Reque	**CATALOGO DE REPORTES**
clave del reporte (PK)		re_id (m)	char (6)	SR	identificador único del reporte
clave del empleado (FK)		em_id	char (6)	SR	identificador del empleado que emite el reporte
Reporte1	**RE1**	**reporte1**	**tabla**	Reque	**CATALOGO DE NOMBRES DE REPORTES**
clave nom. rep. (PK) (FK)		rep1_id	char (6)	SR	identificador único del nombre del reporte
nombre del reporte		rep1_nombre	varchar (30)	SR	nombre del reporte
Nota : los posibles reportes y estadísticas que se pueden consultar en el restaurante son los siguientes :					
ingresos		es_ingresos	varchar (30)	SR	ingresos
egresos		es_egresos	varchar (30)	SR	egresos
venta mensual		es_mensual	varchar (30)	SR	venta mensual
venta semanal		es_semanal	varchar (30)	SR	venta semanal
venta diaria		es_diaria	varchar (30)	SR	venta diaria
ventas por mesero		es_mesero	varchar (30)	SR	ventas por mesero
mejor mes		es_mes	varchar (30)	SR	mes en el que se vendió mas
peor mes		es_menos	varchar (30)	SR	mes el que se vendió menos
hora del día de > venta		es_hora	varchar (30)	SR	hora del día de mayor venta
alimento mas		es_ab_mas	varchar (30)	SR	alimento mas vendido
alimento menos		es_ab_mas	varchar (30)	SR	alimento menos vendida
promoción		es_promocion	varchar (30)	SR	promoción
total de eventos		es_eventos	varchar (30)	SR	no. de eventos en un periodo de tiempo
artículos vendidos		es_ve_consecutivo	varchar (30)	SR	conteo de artículos vendidos
Mantenimiento	**MA**	**mantenimiento**	**tabla**	Reque	**CATALOGO DE MANTENIMIENTO**
clave mant. (PK)		ma_id	char (6)	SR	identificador único de la tarea de mantenimiento
clave tipo mant. (FK)		me1_id (m)	char (6)	SR	identificador único del tipo de mantenimiento
descripción de la tarea		ma_descripcion	varchar (50)	SR	nombre del mantenimiento que se lleva a cabo
Técnico mant	**MAT**	**mant_tecnico**	**tabla**	Reque	**CATALOGO DE TECNICOS DE MANTENIMIENTO**
clave tecnico (PK)		mat_id	char (6)	SR	identificador único del técnico que brinda el mantenimiento
nombre del técnico		mat_nombre	varchar (40)	SR	nombre del técnico
dirección		mat_direccion	varchar (40)	SR	dirección del técnico
teléfono		mat_telefono	varchar (40)	SR	teléfono del técnico
Agenda mant	**AGM**	**agenda_mant**	**tabla**	Reque	**CATALOGO AGENDA DE MANTENIMIENTO**
clave agenda (PK)		agm_id	char (6)	SR	identificador único en la agenda de mantenimiento
clave empleado (FK)		em_id	char (6)	SR	identificador único del empleado que dio el mantenimiento
clave tecnico (FK)		mat_id	char (6)	SR	identificador único del técnico que brindo el servicio
clave dpto. (FK)		de_id	char (6)	SR	identificador único del depto. donde se brinda el servicio
fecha de la falla		agm_ffalla	datetime	SR	fecha de la falla
fecha del servicio		agm_fservicio	datetime	SR	fecha del servicio (día de mantenimiento)
fecha del proximo serv.		agm_fproxservicio	datetime	SR	fecha del proximo servicio en el respectivo departamento

Tipo mantenimiento	**MA1**	**mantenimiento_1**	**tabla**	Requs	**CATALOGO DE TIPOS DE MANTENIMIENTO**
clave tipo mant. (PK)		ma1_id (m)	char (6)	SR	identificador único del tipo de mantenimiento
nombre		ma1_tipo	char (6)	SR	nombre que recibe el tipo de mantenimiento (refrig, electr)
Accesorios mantan	**MAC**	**ment_accesorios**	**tabla**	Requs	**CATALOGO DE ACCESORIOS PARA MANT.**
clave acc, mant. (PK)		mac_id	char (6)	SR	identificador único del accesorio de mantenimiento
lugar de uso (FK)		uso_id (m)	char (3)	SR	identificador único del lugar de uso dol accesorio
nombre del accesorio		mac_nombre	varchar (40)	SR	nombre del accesorio

Apéndice d)

Resumen de Tablas de la Base de Datos

APÉNDICE D
Resumen de las tablas de la base de datos

clientes	tabla
cl_id	char (6)
em_id	char (6)
cl_dorado	bit
cl_nombre	varchar (40)
cl_direccion	varchar (40)
cl_ciudad	varchar (20)
cl_telefono	varchar (20)
cl_antiguedad	varchar (20)
cl_frecuencia	varchar (20)
cl_profesion	varchar (20)
cl_rfc	varchar (13)
cl_alergia	varchar (50)
cl_dieta	varchar (50)
cl_fingreso	datetime
cl_fbaja	datetime

empleados	tabla
em_id (PK)	char (6)
de_id	char (6)
pu_id	char (6)
hab_id (m)	char (6)
emc_id (m)	char (6)
cons_id	char (6)
mo_id	char (6)
se_id (m)	char (6)
em_esposo_id	char (6)
em_nombre	varchar (40)
em_direccion	varchar (40)
em_ciudad	varchar (20)
em_telefono	varchar (20)
em_rfc	varchar (13)
em_experiencia	varchar (20)
em_pasaporte	varchar (15)
em_escolaridad	varchar (20)
em_licencia	varchar (9)
em_idiomas	varchar (50)
em_motivacion	varchar (50)
em_presentacion	varchar (50)
em_edocivil	varchar (10)

empleado_cargo	tabla
emc_id (m)	char (6)
em_id	char (6)

habilidad	tabla
hab_id (m)	char (6)
em_id	char (6)
hab_nombre	varchar (40)

consumo	tabla
cons_id	char (6)
em_id	char (6)
re_id (m)	char (6)

alimentos	tabla
ab_id	char (6)
al_id	char (6)
pr_id	char (6)
uso_id (m)	char (3)
ab_nombre	varchar (40)
ab_descripcion	varchar (40)
ab_existencias	int
ab_fingrese	datetime
ab_fcaducidad	datetime
ab_ano	datetime
eb_fproduc	datetime
ab_precio	money
ab_temperatura	varchar (20)

proveedores	tabla
pr_id	char (6)
ab_id	char (6)
pr_cia	varchar (40)
pr_nombre	varchar (40)
pr_direccion	varchar (40)
pr_ciudad	varchar (20)
pr_telefono	varchar (20)
pr_existencia	varchar (5)
pr_pago	varchar (20)
pr_tentrega	varchar (30)
pr_rfc	varchar (13)
pr_retraso	varchar (30)
pr_condición	varchar (10)

productos	tabla
prd_id	char (6)
ab_id	char (6)
prd_nombre	varchar (40)
prd_cunitario	money

ventas	tabla
ve_id	char (6)
ab_id	char (6)
cl_id	char (6)
ev_id	char (6)
em_id	char (6)
pl_id	char (6)
se_id (m)	char (6)
ve_factura	varchar (10)
ve_condiciones	varchar (10)
ve_cvendida	int
ve_fventa	datetime
ve_quejas	varchar (30)
ve_total	money
ve_precio	money

compras	tabla
co_id	char (6)
pr_id	char (6)
ab_id	char (6)
ev_id	char (6)
co_descripcion	varchar (40)
co_condiciones	varchar (10)
co_garantia	varchar (20)
co_fcompra	datetime
co_calidad	varchar (20)
co_competencia	money
co_total	money
co_dianas	varchar (200)

publicas	tabla
pu_id	char (6)
p_id	char (6)
ev_id	char (6)
pu_folletos	varchar (30)
pu_tarjetas	varchar (30)
pu_invitacionper	varchar (30)
pu_regalos	varchar (30)
pu_notas	varchar (30)
pu_agradecimiento	varchar (30)
pu_cortecias	varchar (30)

promociones	tabla
p_id	char (6)
ve_id	char (6)
ab_id	char (6)
p_nombre	varchar (40)
p_horario	varchar (20)
p_porcentaje	tinyint
p_cortesia	varchar (20)
p_finicial	datetime
p_ffinal	datetime

eventos	tabla
ev_id	char (6)
ab_id	char (6)
ac_id	char (6)
p_id	char (6)
ev1_id (m)	char (6)
me_id	char (6)
va_id	char (6)
em_id	char (6)
ev_capacidad	varchar (40)
ev_fecha	datetime
ev_decoracion	varchar (60)
ev_musica	varchar (60)
ev_costo	money
ev_ocupacion	varchar (60)
ov_cancelado	bit
ev_fcancelado	datetime
ev_condiciones	varchar (20)

eventos_1	tabla
ev1_id (m)	char (6)
ev1_nombre	varchar (40)

variedades	tabla
va_id	char (6)
pr_id	char (6)
va_nombre	varchar (40)

lugar_uso	tabla
uso_id (m)	char (3)
uso_nombre	varchar (40)

Accesorios	tabla
ac_id	char (6)
uso_id (m)	char (3)
ac_evento	varchar (100)
ac_nombre	varchar (40)
ac_cminima	int
ac_cmaxima	int
ac_existencia	int
ac_bajas	int

accesorios_1	tabla
ac1_id (m)	char (6)
ev_id	char (6)
ac1_total	int

equipo	tabla
eq_id (m)	char (6)
uso_id (m)	char (3)
eq_nombre	varchar (40)
eq_descripcion	varchar (40)
eq_cminima	int
eq_cmaxima	int
eq_existencias	int
eq_fcompra	datetime
eq_fcaducidad	datetime
eq_bajas	varchar (10)

almacen	tabla
al_id	char (6)
ab_id	char (6)
pr_id	char (6)
dev_id	char (6)
uso_id (m)	char (3)
tipo_id (m)	char (2)
al_descripcion	varchar (40)
al_existencias	int
al_fingreso	datetime
al_fsalida	datetime
al_fcaducidad	datetime
al_fproduccion	datetime
al_cminima	int
al_cmaxima	int

tipo_alimento	tabla
tipo_id (m)	char (2)
tipo_nombre	varchar (15)

recetas	tabla
re_id (m)	char (6)
al_id	char (6)
em_id	char (6)
uso_id (m)	char (3)
re_nombre	varchar (40)
re_descripcion	varchar (40)
re_pventa	money
re_ingredientes	varchar (100)
re_cantidad	int
re_fcaducidad	datetime
re_fecha	datetime
re_costo	money

menu	tabla
me_id	char (6)
men_id (m)	char (6)
em_id	char (6)
me_nombre	varchar (40)
me_descripcion	varchar (100)
me_costo	money

menu_semana	tabla
men_id (m)	char (6)
men_dia	varchar (10)

departamento	tabla
de_id	char (5)
de_noempleados	varchar (10)
de_nombre	varchar (20)
de_costo	money
de_musica	varchar (40)
do_horario	varchar (10)

dein_ventario	tabla
dein_id	char (6)
dein_fisico	varchar (100)
dein_cmaxima	int
dein_cminimo	int

reservaciones	tabla
res_id	char (6)
cl_id	char (6)
res_frecervaciones	varchar (30)
res_cancelada	bit

horarios	tabla
ho_id (m)	char (6)
em_id	char (6)
de_id	char (6)
ho_descanso	varchar (20)
ho_vacaciones	varchar (50)
ho_permisos	varchar (50)
ho_aseo	varchar (50)
ho_disciplina	varchar (50)
asistencia	**tabla**
em_id	char (6)
as_asistencia	bit
as_fasistencia	datetime
requisiciones	**tabla**
req_id	char (6)
ab_id	char (6)
em_id	char (6)
req_descripcion	varchar (60)
req_cantidad	varchar (100)
req_total	int
pruebas	**tabla**
pru_id	char (6)
ab_id	char (6)
em_id	char (6)
pru_rend_kilo	varchar (30)
pru_rend_litro	varchar (30)
pru_condiciones	varchar (30)
pru_tiempo	varchar (30)
comandas	**tabla**
com_id	char (6)
ab_id	char (6)
al_id	char (6)
com_ent_em_id	char (6)
com_aut_em_id	char (6)
com_ab_cantidad	int
objetos	**tabla**
ob_id	char (6)
uso_id (m)	char (3)
em_id	char (6)
ob_descripcion	varchar (40)
ob_fingreso	datetime
ob_fdevolucion	datetime
ob_identifica	varchar (40)
servicio	**tabla**
se_id (m)	char (6)
se_nombre	varchar (20)
se_descripcion	varchar (40)
modales	**tabla**
mo_id	char (6)
mo_descripcion	varchar (30)
capacitacion	**tabla**
ca_id	char (6)
em_id	char (6)
ca_tipo	varchar (30)
ca_fecha	datetime
devolucion	**tabla**
dev_id	char (6)
ab_id	char (6)
em_id	char (6)
de_id	char (6)
dev_descripcion	varchar (30)
dev_cantidad	varchar (30)

sugerencias	tabla
su_id	char (6)
cl_id	char (6)
em_id	char (6)
su_descripcion	varchar (60)
bitacora	**tabla**
bi_id	char (6)
em_id	char (6)
bi_descripcion	varchar (100)
bi_fecha	datetime
plantillas	**tabla**
pl_id	char (6)
em_id	char (6)
pl_nombre	varchar (30)
puestos	**tabla**
pue_id (m)	char (6)
em_id	char (6)
puestos_1	**tabla**
pue1_id (m)	char (6)
pue1_nombre	varchar (30)
pue1_descripción	varchar (50)
perdidas	**tabla**
per_id (m)	char (6)
ab_id	char (6)
per_cantidad	varchar (20)
per_costo	money
nomina	**tabla**
no_id	char (6)
em_id	char (6)
de_id	char (6)
pue_id (m)	char (6)
perm_id (m)	char (6)
ext_id (m)	char (6)
percepciones	
no_sueldo	money
no_textra	varchar (10)
no_otraspercep	money
no_indebidas	money
deducciones	
no_ispt	money
no_imss	money
no_faltas	int
no_infonavit	money
no_otrasdeduc	money
no_percepindeb	money
no_sindicato	money
no_seguro	money
no_prestamo	money
no_mantinfonavit	money
no_pension	money
no_totalpercep	money
no_totaldeduc	money
no_total	money
no_horas	varchar (10)
no_ausencias	int
no_sueldoacum	money
no_antiguedad	varchar (10)
no_suspencion	varchar (10)
no_dias	int
no_folio	varchar (10)
no_fecha	datetime
no_periodo	varchar (40)
no_quincena	varchar (40)

extraordinarias	tabla
ext_id (m)	char (6)
ext_nombre	varchar (30)
permisos	**tabla**
perm_id (m)	char (6)
em_id	char (6)
perm_descripcion	varchar (30)
estadistica	**tabla**
cl_id	char (6)
em_id	char (6)
emp_id	char (6)
ab_id	char (6)
pr_id	char (6)
prd_id	char (6)
va_id	char (6)
co_id	char (6)
pu_id	char (6)
p_id	char (6)
ev_id	char (6)
ev1_id (m)	char (6)
va_id	char (6)
uso_id (m)	char (3)
ac_id	char (6)
ac1_id (m)	char (6)
eq_id (m)	char (6)
al_id	char (6)
re_id (m)	char (6)
me_id	char (6)
man_id (m)	char (6)
de_id	char (6)
dein_id	char (6)
rep_id	char (6)
re1_id (m)	char (6)
tes_id	char (6)
ho_id (m)	char (6)
req_id	char (6)
pru_id	char (6)
com_id	char (6)
ob_id	char (6)
mo_id	char (6)
ca_id	char (6)
dev_id	char (6)
su_id	char (6)
bi_id	char (6)
pl_id	char (6)
pue_id (m)	char (6)
pue1_id (m)	char (6)
per_id (m)	char (6)
hab_id (m)	char (6)
per_id	char (6)
ext_id (m)	char (6)
as_id	char (6)
presupuesto	**tabla**
pre_id	char (6)
em_id	char (6)
de_id	char (6)
pre_nombre	varchar (30)
pre_fecha	datetime
pre_finicial	datetime
pre_ffinal	datetime
pre_desviacion	tinyint
reportes	**tabla**
re_id (m)	char (6)
em_id	char (6)
reporte1	**tabla**
rep1_id	char (6)
rep1_nombre	varchar (30)

mantenimiento	tabla
ma_id	char (6)
mat_id (m)	char (6)
ma_descripcion	varchar (50)
mant_tecnico	**tabla**
mat_id	char (6)
mat_nombre	varchar (40)
mat_direccion	varchar (40)
mat_telefono	varchar (40)
agenda_mant	**tabla**
agm_id	char (6)
em_id	char (6)
mat_id	char (6)
de_id	char (6)
agm_ffalla	datetime
agm_fservicio	datetime
agm_fproxservicio	datetime
mantenimiento_1	**tabla**
ma1_id (m)	char (6)
ma1_tipo	char (6)
mant_accesorios	**tabla**
mac_id	char (6)
uso_id (m)	char (3)
mac_nombre	varchar (40)

Apéndice e)

Transformación del Modelo-ER en Tablas

APÉNDICE E
Transformación del modelo-ER en tablas

Para iniciar este apéndice es necesario primero dar la sintaxis de cómo crear los dispositivos donde estará almacenada físicamente la **BD**, para esto se deberá de iniciar el disco donde contendrá las siguientes propiedades : nombre lógico, físico, número del identificador virtual y el tamaño en **Kb** del dispositivo, tal como se muestra a continuación :

Sintaxis para crear los dispositivos :

```
DISK INIT
NAME = "device_name",
PHYSNAME ="physicalname",
VDEDNO = virtual_device_number,
SIZE = number_of_bloks
```

Ejemplo :

```
disk init
name = "cabana_db",
physname ="d:\sybdata\cabanadat.dat",
vdevno=1,
size =51200
go

disk init
name = "cabana_log",
physname ="d:\syblog\cabanalog.dat",
vdevno=2,
size =5120
go
```

Una vez que se crean los dispositivos, el primero para la **BD** y el segundo para su bitacora de transacciones (parte física donde se lleva a cabo la manipulación de la **BD**), se debe crear la **BD**, para esto se presenta la sintaxis de cómo crear una **BD**, en seguida se tendrá que conectar a la **BD** creada para poder transformar el **modelo-ER** en tablas.

Sintaxis para crear una BD :

```
CREATE DATABASE database_name
[ ON { DEFAULT / database_device } [ = size ]
[ , database_device [ = size ] ] .....]
[ LOG ON database_device [ = size ]
[ , database_device [ = size  ] ] .....]
```

Ejemplo :

```
create database CABANA
on cabana_db = 100
log on cabana_log = 10
go
```

Para interpretar la anterior sintaxis se debe tomar en cuenta que **database_name** es el nombre de la nueva **BD** , **on** es una llave para especificar la localización de la **BD**, en el dispositivo que se almacenará, **size** es el espacio permitido para almacenar información en la **BD**, se calcula en megabytes. Por default **Sybase** da un tamaño de 2 megabytes, pero este tamaño lo puede alterar el **ABD**, usando el **sp_configure** después **reconfigure** y reinicializando **Sybase** para que tome el nuevo tamaño por default.

Una **BD** puede ocupar diferentes dispositivos, dependiendo del tamaño de la **BD**.

Para poder crear la nueva BD es necesario primero entrar a la BD master, contenida dentro de Sybase, una vez dentro de ésta, lo que se hace con la anterior sintaxis es crear una copia de la BD model contenida en Sybase. Esta BD es una estructura de una BD ya creada por Sybase, que se modifica y pone en operación nuevas propiedades de la BD.

Después de haber creado la base de datos es necesario cambiar las opciones de la base de datos, esto se hace con el procedimiento almacenado **sp_dboption** como se muestra a continuación.

```
sp_dboption CABANA,"select",true
sp_dboption CABANA,"trunc",true
checkpoint
go
```

Con los anteriores comandos lo que se hace es poder accesar la **BD**. Para que estos cambios tengan efecto sobre la **BD** se debe ejecutar el comando **checkpoint** .
El siguiente paso es adicionar el **login y el alias** para la **BD**, de la siguiente forma :

```
sp_addlogin CABANA, CABANA, CABANA
sp_addalias CABANA, dbo
go
```

Donde la primer **CABANA es el login,** la segunda **CABANA es el password,** y la tercer **CABANA es la DB;** para el caso del **alias, CABANA es la BD** que se quiere accesar **y dbo** es el propietario de la **BD.**

Como último paso se debe usar la **BD** que se ha creado para adicionar el **login y el alias**, como en **MASTER**.

```
use CABANA
sp_addlogin CABANA, CABANA, CABANA
sp_addalias CABANA, dbo
checkpoint
go
```

Para que todos estos cambios tomen efecto se debe ejecutar el comando **checkpoint** como se mostró anteriormente.

Después de haber creado la **BD**, es necesario conectarse a esta **BD, como el usuario CABANA** y correr el siguiente texto que permitirá crear las tablas que contendrá la **BD**. Se pone en uso la **BD** de la siguiente forma :

```
# isql -UCABANA -PCABANA -SCABANA

1 > use CABANA
2 > go
```

```
drop table clientes
go

create table clientes
( cl_id           char    ( 6 ) not null,
em_id             char    ( 6 )not null,
cl_dorado         bit           not null,
cl_nombre         varchar ( 40 ) not null,
cl_direccion      varchar ( 40 ) not null,
cl_ciudad         varchar ( 20 ) not null,
cl_telefono       varchar ( 20 ) not null,
cl_antiguedad     varchar ( 20 ) not null,
cl_frecuencia     varchar ( 20 ) not null,
cl_profesion      varchar ( 20 ) not null,
cl_rfc            varchar ( 13 ) not null,
cl_alergia        varchar ( 50 ) not null,
cl_dieta          varchar ( 50 ) not null,
cl_fingreso       datetime       not null,
cl_fbaja          datetime       not null,
go
alter table clientes add primary key ( cl_id )
go
alter table clientes add foreing key ( em_id )
go


drop table empleados
go

create table empleados
( em_id           char ( 6 )   not null,
de_id             char ( 6 )   not null,
pu_id             char ( 6 )   not null,
hab_id            char ( 6 )   not null,
emc_id            char ( 6 )   not null,
cons_id           char ( 6 )   not null,
mo_id             char ( 6 )   not null,
se_id             char ( 6 )   not null,
em_esposo_id      char ( 6 )   not null,
em_nombre         varchar ( 40 ) not null,
em_direccion      varchar ( 40 ) not null,
em_ciudad         varchar ( 20 ) not null,
em_telefono       varchar ( 20 ) not null,
em_rfc            varchar ( 13 ) not null,
em_experiencia    varchar ( 20 ) not null,
em_pasaporte      varchar ( 15 ) not null,
em_escolaridad    varchar ( 20 ) not null,
em_licencia       varchar ( 9  ) not null,
em_idiomas        varchar ( 50 ) not null,
em_motivacion     varchar ( 50 ) not null,
em_presentacion   varchar ( 50 ) not null,
em_edocivil       varchar ( 10 ) not null )
go
```

```
drop table empleado
go
create table empleado_cargo
( emc_id          char ( 6 )   not null,
em_id             char ( 6 )   not null )
go
drop table habilidad
go

create table habilidad
( hab_id          char ( 6 )   not null,
em_id             char ( 6 )   not null,
hab_nombre        varchar ( 40 ) not null )
go

drop table consumo
go

create table consumo
( cons_id         char ( 6 )   not null,
em_id             char ( 6 )   not null,
re_id             char ( 6 )   not null )
go

drop table alimentos
go

create table alimentos
( ab_id           char ( 6 )   not null,
pr_id             char ( 6 )   not null,
uso_id            char ( 3 )   not null,
ab_nombre         varchar ( 40 ) not null,
ab_descripcion    varchar ( 40 ) not null,
ab_existencias    int          not null,
ab_fingreso       datetime     not null,
ab_fcaducidad     datetime     not null,
ab_ano            datetime     not null,
ab_fproduc        datetime     not null,
ab_precio         money        not null,
ab_temperatura    varchar ( 20 ) not null )
go

drop table proveedores
go

create table proveedores
( pr_id             char ( 6 )   not null,
ab_id             char ( 6 )     not null,
pr_[illegible]a   varchar ( 40 ) not null,
pr_nombre         varchar ( 40 ) not null,
pr_direccion      varchar ( 40 ) not null,
pr_ciudad         varchar ( 20 ) not null,
pr_telefono       varchar ( 20 ) not null,
pr_existencia     varchar ( 5  ) not null,
pr_pago           varchar ( 20 ) not null,
```

```
pr_tentrega       varchar ( 30 ) not null,
pr_rfc            varchar ( 13 ) not null,
pr_retraso        varchar ( 30 ) not null,
pr_condición      varchar ( 10 ) not null )
go

drop table productos
go

create table productos
( prd_id          char ( 6 )   not null,
ab_id             char ( 6 )   not null,
prd_nombre        varchar ( 40 ) not null,
prd_cunitario     money          not null )
go

drop table ventas
go

create table ventas
( ve_id           char ( 6 )   not null,
ab_id             char ( 6 )   not null,
cl_id             char ( 6 )   not null,
ev_id             char ( 6 )   not null,
em_id             char ( 6 )   not null,
pl_id             char ( 6 )   not null,
se_id            char ( 6 )   not null,
ve_factura        varchar ( 10 ) not null,
ve_condiciones    varchar ( 10 ) not null,
ve_cvendida       int            not null,
ve_fventa         datetime       not null,
ve_quejas         varchar ( 30 ) not null,
ve_total          money          not null,
ve_precio         money          not null )
go

drop table compras
go

create table compras
( co_id           char ( 6 )   not null,
pr_id             char ( 6 )   not null,
ab_id             char ( 6 )   not null,
ev_id             char ( 6 )   not null,
co_descripcion    varchar ( 40 ) not null,
co_condiciones    varchar ( 10 ) not null,
co_garantia       varchar ( 20 ) not null,
co_fcompra        datetime       not null,
co_calidad        varchar ( 20 ) not null,
co_competencia    money          not null,
co_total          money          not null,
co_diarias        varchar ( 200 ) not null )
go
drop table publicas
go
```

```
create table publicas
( pu_id            char ( 6 )   not null,
p_id               char ( 6 )   not null,
ev_id              char ( 6 )   not null,
pu_folletos        varchar ( 30 ) not null,
pu_tarjetas        varchar ( 30 ) not null,
pu_invitacionper   varchar ( 30 ) not null,
pu_regalos         varchar ( 30 ) not null,
pu_notas           varchar ( 30 ) not null,
pu_agradecimiento  varchar ( 30 ) not null,
pu_cortecias       varchar ( 30 ) not null )
go

drop table promociones
go

create table promociones
( p_id             char ( 6 )   not null,
ve_id              char ( 6 )   not null,
ab_id              char ( 6 )   not null,
p_nombre           varchar ( 40 ) not null,
p_horario          varchar ( 20 ) not null,
p_porcentaje       tinyint      not null,
p_cortesia         varchar ( 20 ) not null,
p_finicial         datetime     not null,
p_ffinal           datetime     not null )
go

drop table eventos
go

create table eventos
( ev_id            char ( 6 )   not null,
ab_id              char ( 6 )   not null,
ac_id              char ( 6 )   not null,
p_id               char ( 6 )   not null,
evl_id             char ( 6 )   not null,
me_id              char ( 6 )   not null,
va_id              char ( 6 )   not null,
em_id              char ( 6 )   not null,
ev_capacidad       varchar ( 40 ) not null,
ev_fecha           datetime     not null,
ev_decoracion      varchar ( 60 ) not null,
ev_musica          varchar ( 60 ) not null,
ev_costo           money        not null,
ev_ocupacion       varchar ( 60 ) not null,
ev_cancelado       bit          not null,
ev_fcancelado      datetime     not null,
ev_condiciones     varchar ( 20 ) not null )
go

drop table eventos_1
go
```

```
create table eventos_1
( ev1_id          char ( 6 )     not null,
ev1_nombre        varchar ( 40 ) not null )
go

drop table variedades
go

create table variedades
( va_id           char ( 6 )   not null,
pr_id             char ( 6 )   not null,
va_nombre         varchar ( 40 ) not null )
go

drop table lugar_uso
go

create table lugar_uso
( uso_id          char ( 3 )   not null,
uso_nombre        varchar ( 40 ) not null )
go

drop table accesorios
go

create table accesorios
( ac_id           char ( 6 )   not null,
uso_id            char ( 3 )   not null,
ac_evento         varchar ( 100 ) not null,
ac_nombre         varchar ( 40 ) not null,
ac_cminima        int          not null,
ac_cmaxima        int          not null,
ac_existencia     int          not null,
ac_bajas          int          not null )
go

drop table accesorios_1
go

create table accesorios_1
( ac1_id          char ( 6 )   not null,
ev_id             char ( 6 )   not null,
ac1_total         int          not null,
go

drop table equipo
go

create table equipo
( eq_id           char ( 6 )   not null,
uso_id            char ( 3 )   not null,
eq_nombre         varchar ( 40 ) not null,
eq_descripcion    varchar ( 40 ) not null,
eq_cminima        int          not null,
eq_cmaxima        int          not null,
```

```
eq_existencias    int           not null,
eq_fcompra        datetime      not null,
eq_fcaducidad     datetime      not null,
eq_bajas          varchar ( 10 ) not null )
go
drop table almacen
go

create table almacen
( al_id           char ( 6 )    not null,
pr_id             char ( 6 )    not null,
dev_id            char ( 6 )    not null,
uso_id            char ( 3 )    not null,
tipo_id           char ( 2 )    not null,
al_descripcion    varchar ( 40 ) not null,
al_existencias    int           not null,
al_fingreso       datetime      not null,
al_fsalida        datetime      not null,
al_fcaducidad     datetime      not null,
al_fproduccion    datetime      not null,
al_cminima        int           not null,
al_cmaxima        int           not null )
go

drop table tipo
go

create table tipo
( tipo_id         char ( 2)     not null,
tipo_nombre       varchar ( 15 ) not null )
go

drop table recetas
go

create table recetas
( re_id           char ( 6 )    not null,
al_id             char ( 6 )    not null,
em_id             char ( 6 )    not null,
uso_id            char ( 3 )    not null,
re_nombre         varchar ( 40 ) not null,
re_descripcion    varchar ( 40 ) not null,
re_pventa         money         not null,
re_ingredientes   varchar ( 100 ) not null,
re_cantidad       int           not null,
re_fcaducidad     datetime      not null,
re_fecha          datetime      not null,
re_costo          money         not null )
go

drop table menu
go
create table menu
( me_id           char ( 6 )    not null,
men_id            char ( 6 )    not null,
```

```
em_id              char ( 6 )   not null,
me_nombre          varchar ( 40 ) not null,
me_descripcion     varchar ( 100) not null,
me_costo           money          not null )
go
drop table menu_semana
go

create table menu_semana
( men_id             char ( 6 )   not null,
men_dia            varchar ( 10 ) not null )
go

drop table departamento
go

create table departamento
( de_id            char ( 6 )   not null,
de_cempleados     varchar ( 10 ) not null,
de_nombre          varchar ( 20 ) not null,
de_costo           money          not null,
de_musica          varchar ( 40 ) not null,
de_horario         varchar ( 10 ) not null )
go

drop table dein_ventario
go

create table dein_ventario
( dein_id          char ( 6 )   not null,
dein_fisico        varchar ( 100 ) not null,
dein_cmaxima       int            not null,
dein_cminimo       int            not null )
go

drop table reservaciones
go

create table reservaciones
( res_id           char ( 6 )   not null,
cl_id              char ( 6 )   not null,
res_frecervaciones varchar ( 30 ) not null,
res_cancelada      bit            not null,
horarios           tabla          not null,
ho_id              char ( 6 )   not null,
em_id              char ( 6 )   not null,
de_id              char ( 6 )   not null,
ho_descanso        varchar ( 20 ) not null,
ho_vacaciones      varchar ( 50 ) not null,
ho_permisos        varchar ( 50 ) not null,
ho_aseo            varchar ( 50 ) not null,
ho_disciplina      varchar ( 50 ) not null )
go
drop table asientos
go
```

```
create table asistencia
( em_id              char ( 6 )   not null,
as_asistencia     bit          not null,
as_fasistencia    datetime     not null )
go

drop table requisiciones
go

create table requisiciones
( req_id          char ( 6 )   not null,
ab_id             char ( 6 )   not null,
em_id             char ( 6 )   not null,
req_descripcion   varchar ( 60 ) not null,
req_cantidad      varchar ( 100 ) not null,
req_total         int          not null )
go

drop table pruebas
go

create table pruebas
( pru_id          char ( 6 )   not null,
ab_id             char ( 6 )   not null,
em_id             char ( 6 )   not null,
pru_rend_kilo     varchar ( 30 ) not null,
pru_rend_litro    varchar ( 30 ) not null,
pru_condiciones   varchar ( 30 ) not null,
pru_tiempo        varchar ( 30 ) not null )
go

drop table comandas
go

create table comandas
( com_id          char ( 6 ) . not null,
ab_id             char ( 6 )   not null,
al_id             char ( 6 )   not null,
com_ent_em_id     char ( 6 )   not null,
com_aut_em_id     char ( 6 )   not null,
com_ab_cantidad   int          not null )
go

drop table objetos
go

create table objetos
( ob_id           char ( 6 )   not null,
uso_id            char ( 3 )   not null,
em_id             char ( 6 )   not null,
ob_descripcion    varchar ( 40 ) not null,
ob_fingreso       datetime     not null,
ob_fdevolucion    datetime     not null,
ob_identifica     varchar ( 40 ) not null )
go
```

```
drop table servicios
go

create table servicio
( se_id           char ( 6 )   not null,
se_nombre         varchar ( 20 ) not null,
se_descripcion    varchar ( 40 ) not null )
go

drop table ser modales
go

create table modales
( mo_id           char ( 6 )   not null,
mo_descripcion    varchar ( 30 ) not null )
go

drop table capacitacion
go

create table capacitacion
( ca_id           char ( 6 )   not null,
em_id             char ( 6 )   not null,
ca_tipo           varchar ( 30 ) not null,
ca_fecha          datetime     not null )
go

drop table devolucion
go

create table devolucion
( dev_id          char ( 6 ) not null,
ab_id             char ( 6 )   not null,
em_id             char ( 6 )   not null,
de_id             char ( 6 )  not null,
dev_descripcion   varchar ( 30 ) not null,
dev_cantidad      varchar ( 30 ) not null )
go

drop table sugerencias
go

create table sugerencias
( su_id           char ( 6 )   not null,
cl_id             char ( 6 )   not null,
em_id             char ( 6 )   not null,
su_descripcion    varchar ( 60 ) not null )
go

drop table bitacora
go
```

```
create table bitacora
( bi_id           char ( 6 )   not null,
em_id             char ( 6 )   not null,
bi_descripcion    varchar ( 100 ) not null,
bi_fecha          datetime     not null )
go

drop table platillos
go

create table platillos
( pl_id           char ( 6 )   not null,
em_id             char ( 6 )   not null,
pl_nombre         varchar ( 30 ) not null )
go

drop table puestos
go

create table puestos
( pue_id          char ( 6 )   not null,
em_id             char ( 6 )   not null )
go

drop table puestos_1
go

create table puestos_1
( pue1_id         char ( 6 )   not null,
pue1_nombre       varchar ( 30 ) not null,
pue1_descripción  varchar ( 50 ) not null )
go

drop table perdidas
go

create table perdidas
( per_id          char ( 6 )     not null,
ab_id             char ( 6 )     not null,
per_cantidad      varchar ( 20 ) not null,
per_costo         money          not null )
go

drop table nomina
go

create table nomina
( no_id           char ( 6 )   not null,
em_id             char ( 6 )   not null,
de_id             char ( 6 )   not null,
pue_id            char ( 6 )   not null,
perm_id           char ( 6 )   not null,
ext_id            char ( 6 )   not null,
no_sueldo         money        not null,
no_textra         varchar ( 10 ) not null,
```

```
no_otraspercep    money          not null,
no_indebidas      money          not null,
no_ispt           money          not null,
no_imss           money          not null,
no_faltas         int            not null,
no_infonavit      money          not null,
no_otrasdeduc     money          not null,
no_percepindeb    money          not null,
no_sindicato      money          not null,
no_seguro         money          not null,
no_prestamo       money          not null,
no_mantinfonavit  money          not null,
no_pension        money          not null,
no_totalpercep    money          not null,
no_totaldeduc     money          not null,
no_total          money          not null,
no_horas          varchar ( 10 ) not null,
no_ausencias      int              not null,
no_sueldoacum     money            not null,
no_antiguedad     varchar ( 10 ) not null,
no_suspencion     varchar ( 10 ) not null,
no_dias           int              not null,
no_folio          varchar ( 10 ) not null,
no_fecha          datetime         not null,
no_periodo        varchar ( 40 ) not null,
no_quincena       varchar ( 40 ) not null )
go

drop table extraordinarias
go

create table extraordinarias
( ext_id          char ( 6 )     not null,
ext_nombre        varchar ( 30 ) not null )
go

drop table permisos
go

create table permisos
( perm_id           char ( 6 )   not null,
em_id             char ( 6 )   not null,
perm_descripcion  varchar ( 30 ) not null )
go

drop table estadisticas
go

create table estadistica
( cl_id           char ( 6 )   not null,
em_id             char ( 6 )   not null,
emp_id            char ( 6 )   not null,
ab_id             char ( 6 )   not null,
pr_id             char ( 6 )   not null,
prd_id            char ( 6 )   not null,
```

```
ve_id               char ( 6 )   not null,
co_id               char ( 6 )   not null,
pu_id               char ( 6 )   not null,
p_id                char ( 6 )   not null,
ev_id               char ( 6 )   not null,
ev1_id              char ( 6 )   not null,
va_id               char ( 6 )   not null,
uso_id              char ( 3 )   not null,
ac_id               char ( 6 )   not null,
ac1_id              char ( 6 )   not null,
eq_id               char ( 6 )   not null,
al_id               char ( 6 )   not null,
re_id               char ( 6 )   not null,
me_id               char ( 6 )   not null,
men_id              char ( 6 )   not null,
de_id               char ( 6 )   not null,
dein_id             char ( 6 )   not null,
rep_id              char ( 6 )   not null,
re1_id              char ( 6 )   not null,
res_id              char ( 6 )   not null,
ho_id               char ( 6 )   not null,
req_id              char ( 6 )   not null,
pru_id              char ( 6 )   not null,
com_id              char ( 6 )   not null,
ob_id               char ( 6 )   not null,
mo_id               char ( 6 )   not null,
ca_id               char ( 6 )   not null,
dev_id              char ( 6 )   not null,
su_id               char ( 6 )   not null,
bi_id               char ( 6 )   not null,
pl_id               char ( 6 )   not null,
pue_id              char ( 6 )   not null,
pue1_id             char ( 6 )   not null,
per_id              char ( 6 )   not null,
hab_id              char ( 6 )   not null,
per_id              char ( 6 )   not null,
ext_id              char ( 6 )   not null,
as_id               char ( 6 )   not null )
go

drop table presupuestos
go

create table presupuesto
( pre_id            char ( 6 )   not null,
em_id               char ( 6 )   not null,
de_id               char ( 6 )   not null,
pre_nombre          varchar ( 30 ) not null,
pre_fecha           datetime     not null,
pre_finicial        datetime     not null,
pre_ffinal          datetime     not null,
pre_desviacion      tinyint      not null )
go
```

```
drop table reportes
go

create table reportes
( re_id           char ( 6 )   not null,
em_id             char ( 6 )   not null )
go

drop table reportes1
go

create table reporte1
( rep1_id           char ( 6 )   not null,
rep1_nombre       varchar ( 30 ) not null )
go

drop table mantenimiento
go

create table mantenimiento
( ma_id           char ( 6 )   not null,
ma1_id ( m )      char ( 6 )   not null,
ma_descripcion    varchar ( 50 ) not null )
go

drop table mant_tecnico
go

create table mant_tecnico
( mat_id          char ( 6 )   not null,
mat_nombre        varchar ( 40 ) not null,
mat_direccion     varchar ( 40 ) not null,
mat_telefono      varchar ( 40 ) not null )
go

drop table agenda_mant
go

create table agenda_mant
( agm_id            char ( 6 )   not null,
em_id             char ( 6 )   not null,
mat_id            char ( 6 )   not null,
de_id             char ( 6 )   not null,
agm_ffalla        datetime     not null,
agm_fservicio     datetime     not null,
agm_fproxservicio datetime     not null )
go
drop table mantenimiento_1
go

create table mantenimiento_1
( ma1_id ( m )    char ( 6 )   not null,
ma1_tipo          char ( 6 )   not null )
go
```

```
drop table mant_accesorios
go

create table mant_accesorios
( mac_id          char ( 6 )   not null,
uso_id            char ( 3 )   not null,
mac_nombre        varchar ( 40 ) not null )
go
```

A continuación se presentan algunos comandos de **Transact - SQL** que sirven para darnos una idea de cómo se manipula una **DB** en **Sybase**.

Aquí se presenta cómo alterar la estructura de una tabla y se agrega una columna a la tabla clientes :

```
alter table clientes
add cl_premio varchar(20)
check cl_premio not in (" CONSUMO GRATIS")
go
```

Se crea un índice lógico en la columna cl_nombre, de tal forma que los nombres de los clientes se presentarán ordenados :

```
create index clienteindex
on clientes (cl_nombre)
go
```

Se crea un índice fisico en la columna cl_nombre :

```
create clustered index clientesindex1
on clientes (cl_nombre)
go
```

Se crea un defaul para cuando no se capture un valor en la columna cl_profesion se inserte esta automáticamente :

```
create default clientesdefault1 clientes
on cl_profesion  as "LICENCIATURA"
go
```

Se crea una regla para que la ciudad siempre sea **estado de mexico**, cuando se inserte un empleado :

```
create rule cl_ciudadrule
as  &cl_ciudad in "ESTADO DE MEXICO"
go
```

Se crean triggers para que despliegen un mensage cada vez que ocurra un evento de inserción, actualización o baja de algun cliente :

create trigger t1 on clientes
for update
as prit "CLIENTE DADO DE ALTA"
go

create trigger t2 on clientes
for delete
as prit "CLIENTE DADO DE BAJA"
go

create trigger t3 on clientes
for update
as prit "CLIENTE MODIFICADO"
go

Se crea un grupo de usuarios dentro de la **DB** cabana
sp_addgrp meseros

Se adiciona un usuario **mary** dentro del grupo meseros
sp_adduser mary,meseros

Se adiciona login y password a mary en la **DB** cabana
sp_addlogin mary,mary,cabana

Se dan permisos de insertar, actualizar o borrar un cliente al usuario mary :
grant insert, delete, update on clientes to mary

Se dan todos los permisos al grupo de los meseros para que manipulen la información de los clientes :
grant all on clientes to meseros

La sintaxis para este apéndice fue tomada del curso
[SISTEM 10 FAST TRACT TO SQL - SERVER]
SYBASE, INC,. USA 1993

Apéndice f)

Requerimientos de información

APÉNDICE F
Requerimientos de información

En este apéndice se observarán los requerimientos de información para los usuarios en las áreas del restaurante tales como **área 1), alimentos y bebidas; área 2), mantenimiento, y área 3), contabilidad.**

Esto se presenta en este apéndice debido a lo extenso de las necesidades y por espacio y didáctica del capítulo IV se considera que es mejor mostrarlo aquí.

En estos requerimientos se profundiza más en el **área de alimentos y bebidas**, ya que a partir de éstos giran todos los movimientos dentro del restaurante, debido a que el **área de contabilidad** toma la mayoría de los datos de esta área para elaborar todos los informes, además, el **área de mantenimiento** se encarga del buen funcionamiento de las instalaciones del restaurante y de las que están contempladas dentro de esta área.
Para los siguientes ejemplos mostrará la información de la siguiente forma :
Primero se observará una descripción del puesto, seguido de sus funciones, de las cuales surgirán las necesidades de información, resaltándolas con letras obscuras. Seguido de esto, muestra el **alias** de la entidad o entidades a partir de las cuales se explicará información o servirán para establecer una relación con otras entidades.

Área 1)
Alimentos y bebidas (AB)

Gerente de alimentos y bebidas.
Es el responsable ante el gerente de la adecuada administración del área de alimentos y bebidas.
Sus principales funciones son :
1) Supervisa el trabajo realizado por los empleados a su cargo (chef, maitre, contralor de costos, jefe de banquetes, jefe de bares y chef steward).
2) Junto con el gerente general, el jefe de compras y el almacenista, elaborar especificaciones **standard de compras** de alimento y bebidas. **CO**
3) **Adquisición de mercancías** cuyos requisitos no estén en los standares de compras. **CO**
4) Supervisar **máximos y mínimos** de alimentos y bebidas. **AL**
5) Elaborar hojas de **costos de recetas standard** de alimentos y bebidas. **RE**
6) Fijar los **precios de venta** de alimentos y bebidas. **RE**
7) Elaborar y actualiza **menús**. **ME**
8) **Reporte diario de ventas** de alimentos y bebidas. **ES**
9) Autorizar **ventas de promociones y cortesías**. **P**
10) Elaborar **presupuesto de gastos** de alimentos y bebidas. **PRE**
11) Revisar mensualmente los **estados de resultados** y presenta un informe. **ES**
12) Vigilar los **sistemas de trabajo**.
13) Supervisar **normas de sanidad**.
14) Revisar **programas de promoción**. **P**

15) Revisar **materiales y equipo. AC, EQ**
16) Promueve las buenas relaciones entre el personal a su cargo.
17) Promueve juntas con su equipo de trabajo.
18) Contratar **variedades. CO**

Jefe de banquetes
Depende del gerente de alimentos y bebidas, sus principales necesidades de información son:
1) Organizar **banquetes. EV**
2) Conocer **capacidad del restaurante. EV**
3) Controlar **horarios y fechas de eventos. EV**
4) Políticas para **propinas en eventos.**
5) Políticas para **cancelación de eventos. EV**
6) Promueve **eventos especiales. EV**
7) Manual **informativo de banquetes. EV**
8) Presupuestos **anuales de banquetes. EV**
9) Atiende a **clientes potenciales. PU**
10) Ventas de **eventos especiales. VE**
11) Realiza **cotizaciones para eventos. EV**
12) Elabora **contratos para eventos. EV**
13) Elabora **publicidad para eventos. EV**
14) Recoge **sugerencias de clientes. SU**
15) Supervisa pagos de cuenta.
16) Elabora **cartas de agradecimiento** para clientes. **PU**
17) Llama a **clientes con reservaciones** tentativas. **RES**
18) Elaborar **reporte de ingresos y egresos. ES**
19) Elabora **reporte mensual de productividad. ES**
20) Programa eventos especiales (fiesta de fin de año, 16 de septiembre, etc.) **EV.**
21) Contrata servicios externos (ballet, música, flores) **PRD.**
22) Informa la **productividad de banquetes. ES**

Contralor de costos de alimentos y bebidas.
Sus actividades son :
1) Elabora análisis diario de **costos de alimentos y bebidas. ES**
2) Elabora **standard de nómina. NO**
3) Revisa **desviaciones de presupuestos. PRE**
4) Revisa **standares de nómina. NO**
5) Realiza especificaciones de **standares de compra. CO**
6) Revisa **proceso de compra** (proveedores y pedidos). **CO**
7) Verifica **inventario de artículos perecederos. AL**
8) Reporta **artículos de poco movimiento. AL**
9) Revisa **fecha de entrada** de productos en el almacén. **AL**
10) Supervisa **horarios. HO**
11) Revisa temperaturas de **refrigeración de alimentos y bebidas. AB**

12) Verifica **costos unitarios** en facturas. **PRD**
13) Verifica sellos en facturas.
14) Verifica etiquetas contra factura.
15) Prepara **inventario del almacén**. (mensualmente) **AL**
16) Establece **horarios de almacén**. **HO**
17) **Verifica requisiciones al almacén**. **REQ**
18) Elabora **hoja diaria de compras**. **CO**
19) Calcula **facturas**, **requisiciones**, registro de **inventarios de bebidas**, **análisis de costo departamental**. **VE, AL, REQ, PRE**
20) **Consumo promedio** por empleado. **CONS**
21) Prepara **pruebas de rendimiento** en carnes y bebidas (en artículos crudos y cocidos).**PRU**
22) Prepara **recetas standard**. **RE**
23) **Costea menús**. **ME**
24) Conteo estadístico de **artículos vendidos**. **ES**
25) Prepara **ventas potenciales**. **PU**
26) Listas de **porciones standar**. **RE**
27) **Costeo de recetas**. **RE**
28) **Costea cada artículos** en menús. **ME**
29) Trimestralmente **Actualiza el menú. ME**
30) Mensualmente verifica costo **compra de alimento o bebida contra costo venta**. **CO, VE**
31) Reporta mensualmente **pérdidas y ganancias**. **VE**
32) Prepara **costo de bebidas**. **CO** ,
33) Revisa cálculo de **precios unitarios**. **PRD**
34) Revisa reportes de **precios por botella**. **PRD**
35) **Inventarios físicos** en bares. **DEIN**
36) Analiza ventas de **refrescos y jugos**. **VE**
37) Revisa **inventarios físicos** en el almacén. **DEIN**
38) Determina **diferencias** entre inventario físico y lo que hay en documentos.
39) Controla **porciones** en artículos de cocina. **RE**
40) Verifica **ingresos a la caja**. **ES**
41) Tiene registro de **ventas de vinos**. **VE**
42) **Costea comida** para empleados. **CONS**
43) Actualiza **cambios en precios de verduras**. **AL, CO**

Maitre
Es el responsable ante el gerente de alimentos y bebidas del servicio a la mesa.
1) Selecciona a su personal y asigna tareas. **EM**
2) Programa **turnos de trabajo**. **HO**
3) Autoriza **tiempo extra**. **EM, NO**
4) Autoriza **permisos al personal**. **PER**
5) Elabora **reportes de asistencia**. **AS**
6) Supervisa **aseo y disciplina**. **HO**
7) Evalúa a su personal.

8) Supervisa uso de **materiales y equipo** en el restaurante. **AC, EQ**
9) Capacita a su personal en :
Presentación personal.
Normas de cortesia.
Trabajo de equipo.
Sistema para tomar órdenes.
Ventas en restaurante.
Formas de escribir órdenes.
Acarreo de charola.
Formas de servir.
Formas de retirar platos.
Prevención de accidentes.
Estaciones de servicio.
Presentación de cuentas.
Tráfico en la cocina.
Montajes de mesa.
Tipos de cubiertos.
Tipos de loza y cristalería.
Flameado.
Arreglos de mesa.
Montaje de bufetes.
Servicio de bufetes.
Métodos para catar vinos.
Formas de abrir las botellas de vino.
Formas de almacenar y servir vinos.
Formas de cortar la carne.
Doblado de servilletas.
Formas de hacer figuras de hielo.
Método de deshuesado.
Explicación de platillos en el menú. **ME**
10) Supervisa calidad en **menús**. **ME**
11) Establece formas para **revisión de equipo**.
12) Conoce y aplica **tipos de servicio** (americano, francés, ruso, bufete, banquete). **SE**
13) Atención personal.
14) Requisita **personal eventual** para eventos especiales. **CO**
15) Elabora **menús**. **ME**
16) Verifica pronósticos de **ocupación**. **EV**
17) Reporta **requisiciones** de mercancia. **RE**
18) Solicita servicio de mantenimiento. **MA**
19) Evalúa calidad de vinos.
20) Analiza la venta promedio por cliente. **ES**

Capitán de meseros.
Es el responsable ante el maitre del buen funcionamiento del restaurante y bar, sus obligaciones son :

1) Supervisión de :
Aseo general del lugar.
Material y equipo completo. **EQ**
Montaje correcto.
Mantenimiento general del lugar. **MA**
Música ambiental. DE
Personal completo (meseros, ayudantes, cajero, cocinero). **DE**
Limpieza del personal y gafetes puestos.
Comandas. COM
Suficientes **accesorios** limpios. **AC**
Personal enterado de la **especialidad del día. ME**
Asignación de mesas a meseros.
Reservación de mesas.
2) Es el responsable de la supervisión del cumplimiento de todos los procedimientos y normas establecidas por la empresa en el restaurante.
3) Recibe al cliente con cortesía.
4) Toma **orden del cliente**. **COM**
5) Conoce **ingredientes de platillos. RE**
6) Entrega de **comandas**. **COM**
7) Supervisa el servicio al cliente.
8) Supervisa **porciones y presentación** de platillos. **RE**
9) Supervisa **cuentas y comandas**. **COM, VE**
10) Supervisa a la cajera.
11) Despide a los clientes y los **invita a regresar. PU**
12) Mantienen el espíritu de equipo entre su personal.
13) Mantienen comunicación constante.
14) Entrena al personal de nuevo ingreso.
15) Da **capacitación** y adiestramiento. **CA**
16) Cuida que los empleados no formen grupos.
17) Evita llamar la atención en presencia de los clientes.
18) Hace cambios para el mejor funcionamiento del personal.
19) Hace **requisiciones** del material faltante. **REQ**
20) Recibe **quejas de los clientes** y busca solución. **SU**
21) Da atención especial a clientes importantes.
22) Presenta cuenta al cliente en eventos especiales.
23) Asiste a juntas.
24) Toma lista de asistencia.
25) Asigna turnos de trabajo. **HO**
26) Distribuye días de descanso. **HO**
27) Programa vacaciones. **HO**
28) Aplica **suspenciones**. **EM**
29) Autoriza **tiempo extra**. **NO**
30) Supervisa material completo al final de la jornada.
31) En caso de accidentes de trabajo, supervisa la solución.
32) Conoce los diferentes tipos de servicio. **SER**

33) Revisa presupuestos y gastos del restaurante. **ES**
34) Revisa que los objetos olvidados sean mandados al dpto. de objetos olvidados. **OB**

Hostess
Sus obligaciones son :
1) Supervisa, junto con el capitán de meseros, el montaje del restaurante o bar.
2) Controla las **reservaciones**. **RES**
3) Da la bienvenida a los clientes.
4) Distribuye a la clientela.
5) Supervisa calidad del mesero.
6) Interviene en los problemas que puedan surgir entre cliente y mesero.
7) Evalúa la calidad del servicio, invitando a los clientes a regresar. **SU**
8) Debe conocer el trabajo del capitán de meseros y el garrotero para poderlos ayudar.
9) Cuando se encuentra lleno el restaurante, no rechaza al cliente , sino que lo invita a pasar al bar.
10) Debe tener excelente presentación y dominar el idioma inglés.

Mesero
Sus obligaciones son :
1) Una persona limpia (cuerpo y uniforme). **EM**
2) Conocimientos en **primeros auxilios**. **EM**
3) Conoce terminología en alimentos y bebidas.
4) Conoce uso del material. **EQ**
5) Limpieza de mesas, candeleros, lámparas, etc.
6) Correcto montaje de mesas.
7) Surtir estaciones de servicio (salsas, azúcar, cubiertos).
8) Ayuda a las Hostess en el acomodo de clientes en las mesas.
9) Debe ser amable y cortés.
10) Sirve agua en las copas.
11) Conoce **platillos del menú**. **ME**
12) Conoce **aperitivos**, entremeses. **RE**
13) Conoce clientes que tienen **dieta o alguna alergia** en especial. **CL**
14) Conoce **lista de vinos**. **AB**
15) Toma órdenes de vinos.
16) Conoce **temperaturas** para servir vinos. **AB**
17) Conoce formas de abrir y servir vinos.
18) Espera aprobación del cliente para servir más vino.
19) Inspecciona qué platillos sean según lo solicitado.
20) Sirve alimentos a la mesa.
21) Conoce los diferentes **tipos de servicio**. **SE**
22) Sirve bufete cuando un cliente lo solicita.
23) Flamea alimentos.
24) Conoce técnicas de trinchado.

25) Conoce técnicas de rebanado y deshuesado.
26) Reemplaza ceniceros.
27) Solicita talón para la cuenta del cliente.
28) Presenta al cliente **la cuenta** para su pago. **VE**
29) No toca dinero directamente con las manos.
30) Paga al cajero la cuenta.
31) Observa que los clientes no **olviden ningún objeto. OB**
32) Participa en la distribución de la propina.
33) Entrena al personal de nuevo ingreso.
34) Reporta **comentarios de clientes. SU**
35) Sirve bebidas en los cocteles.
36) Conoce y aplica procedimientos en accidentes.
38) Es recomendable que el mesero sea :
Amable con los clientes.
No hacer grupos.
Evitar conversaciones personales con otro empleado.
No gritar. **MO**
No discutir con nadie. **MO**
No secarse la cara con las servilletas. **MO**
Si una servilleta cae al suelo, reemplazarla por otra. **MO**
No fumar dentro del área de trabajo. **MO**
No hablar con majaderías. **MO**
No comer durante el servicio. **MO**
Llevar el menú en las manos. **MO**
No colocar las manos en los bolsillos. **MO**
Caminar rápido, no correr. **MO**
Ser amable con los niños. **MO**
No contar la propina frente al cliente. **MO**
Seguir el sistema de rotación al tomar las órdenes. **MO**
Ser amable con el cliente. **MO**
Notificar si por alguna razón se ausenta del trabajo. **MO**
39) Entrega **comandas** para salida de alimentos. **COM**
40) Verifica **comandas** con autorización del cajero. **COM**
41) Canta órdenes. **COM**
42) Reúne **comandas** que ya fueron surtidas al contralor de costos. **COM**

Ayudante de mesero (garrotero)
Sus obligaciones específicas son :
1) Montaje de mesas.
2) Levanta " el muerto ".
3) Coloca en las mesas lo indispensable (salsas, limones, galletas).
4) Sirve mantequilla a los clientes.
5) Llena copas de agua .
6) Ayuda al mesero a traer comida de la cocina.

7) Prepara café.
8) Conoce cómo transportar charolas.
9) Evita desperdicios separando mantequilla y salsa para ser aprovechadas.
10) Conoce las obligaciones del mesero.

Jefe del bar
Es el responsable ante el gerente de alimentos y bebidas de la operación del bar.
Sus obligaciones específicas son :
1) Selecciona, evalúa y **capacita** a los cantineros. **CA**
2) Supervisa el trabajo de los cantineros.
Servicio general.
Aseo en preparación.
Porciones en bebidas.
Entrega de bebidas en comanda.
3) Asigna **turnos de trabajo**. **HO**
4) Control de **costos y especificaciones standar de compras**. **CO**
5) Establece **tipos de vinos en bebidas compuestas**. **RE**
6) Elabora **recetas standar de bebidas**. **RE**
7) Hace cálculo de porciones en cada botella. **PRU**
8) Contribuye en la elaboración de precios de bebidas para banquetes.
9) Verifica **máximos y mínimos** de bebidas en el bar. **DEIN**
10) Hace **devoluciones** al almacén de vinos y sobrantes de eventos. **DEV**
11) Analiza con el gerente de alimentos y bebidas los **estados de resultados** del bar.
(costo, gasto y utilidad), y adopta medidas para mejorar los resultados. **ES**
12) Autoriza **descuentos y cortesías**. **P**
13) Autoriza las cuentas que se envían para su cobro.

Cantinero
1) Conoce los tipos de **bebidas usados** (vinos, licores, cervezas, refrescos, jugos , etc.). **AB**
2) Conoce los **tipos de botanas**. **AB**
3) Conoce el almacenamiento de vinos. **AL**
4) Conoce la preparación de **bebidas compuestas**. **RE**
5) Conoce **cristalería para servir la botana**. **EQ**
6) Levanta **inventarios diarios** y solicita faltante al almacén. **DEIN**
7) Llena requisiciones al almacén. **REQ**
8) **Prepara botanas**. **RE**
9) Toma órdenes directas de los clientes.
10) Elabora informe de **control de botellas**. **AL**
11) Al cerrar el bar es el responsable de que los artículos queden protegidos contra robo.
12) Distribuye y evalúa tareas de su ayudante.

Ayudante del cantinero
Recibe órdenes del cantinero y sus obligaciones principales son :

1) Ayudar a **levantar inventarios. DEIN**
2) Llevar **requisiciones** al almacén. **REQ**
3) Recibir artículos del almacén para llevarlos al bar.
4) Verificar hielo suficiente en el bar. **DEIN**
5) Cortar fruta y decorar bebidas.
6) Proveer cristalería al cantinero. **EQ**
7) Recoger basura de botellas.
8) Llenar refrigeradores con material faltante.
9) Preparar botanas.
10) Lavar cristalería en el bar.
11) Conocer el trabajo del cantinero, por si se ofrece cubrirlo.

Chef

Es responsable ante el gerente de alimentos y bebidas del adecuado funcionamiento de las cocina, de la adecuada preparación de los alimentos dentro de las normas de calidad y del buen servicio a bajo costo, sus funciones específicas son :

1) Coordina al personal de la cocina en los siguientes aspectos :
Entrevista a aspirantes de puestos de cocina.
Selecciona al personal.
Capacita al personal. **CA**
Supervisa el trabajo.
Elabora horarios de trabajo. **HO**
Programa días de descanso. **HO**
Solicita personal eventual. **CO**
Hace juntas periódicas.
Escucha comentarios y sugerencias. **SU**
Mantiene disciplina y aseo.
Elabora descripciones de puestos.
Asigna tareas a su personal.
Mantiene el espíritu de equipo.
2) Diariamente levanta inventario de artículos de consumo inmediato. **DEIN**
3) Elabora **requisiciones**. **RE**
4) Supervisa calidad de alimentos.
5) Elabora hojas de costos de **recetas standar**. **RE**
6) Participa en la **elaboración de menús**. **ME**
7) Conoce **variaciones de costos** en materia prima. **CO**
8) Reduce gastos de nómina. **NO**
9) Inspecciona porciones , guarniciones, limpieza y decoración de platillos.
10) Supervisa aplicación de normas sanitarias. (checar normas sanitarias)
11) Establece máximos y mínimos de productos. **AL**
12) Analiza **costos reales** de cocina.
13) Elabora **menús para empleados**. **ME**
14) Supervisa calidad de alimentos y bebidas. **AB**

Souschef

Es el responsable de las actividades que se llevan a cabo en la cocina durante un periodo determinado, sus funciones son :

1) Dirige, supervisa y controla la producción de platillos. **ME**
Limpieza y mantenimiento de equipo. **MA**
Control de materia prima.
2) Supervisa **calidad y control** de alimentos. **AB**
3) Asigna y supervisa tareas del personal.
4) Prepara platillos especiales. **ME**
5) Supervisa salida de alimentos de la cocina.
6) Requisita materia prima.
7) Analiza los resultados de operación.
8) Elabora presupuestos.

Cocinero (cocina caliente)

Es responsable ante el chef de la preparación de alimentos en la cocina; sus funciones son :

1) Conocer **tiempos de cocimiento**. **PRU**
2) Conocer **técnicas para asar y hornear**. **RE**
3) Conocer **recetas standar**. **RE**
4) Conocer cómo evaluar alimentos en mal estado.
5) Aplicar técnicas para almacenar alimentos.
6) Evitar el desperdicio en sobrantes para elaborar platillos.
7) Es responsable del equipo y material a su cargo. **EQ**
8) Verifica material necesario.
9) Supervisa el trabajo de su ayudante.
10) Conoce técnicas para decorar alimentos.
11) Requisita materia prima al almacén. **AL**
12) Revisa y aprovecha sobrantes del día anterior.
13) Elabora salsas. **RE**

Ayudante cocina caliente

Asesora al cocinero en la elaboración de alimentos; sus obligaciones específicas son :

1) Enciende estufas y hornos.
2) Prepara **guarniciones**. **RE**
3) Elabora **botanas** calientes. **RE**
4) Elabora **salsas** y las coloca en su lugar. **RE**
5) Ayuda a **preparar sopas y especialidades**. **RE**
6) Ayuda a preparar **alimentos en baño maría**. **RE**
7) Mantiene limpio el lugar de trabajo.
8) Sustituye al cocinero cuando se ausenta.
9) Ayuda al montaje de alimentos.
10) Controla y **surte carne** al cocinero. **AL**
11) Corta la carne según el menú.

12) Limpia y muele carne.
13) Conoce procedimientos para almacenar carne.
14) Rebana jamón, tocino, etc.
15) Prepara carne para menudo, consomé, etc.
16) Deshuesa pollo y pescado.
17) Proporciona mariscos.
18) Prepara **brochetas**. **RE**
19) Elabora **carne adobada**. **RE**
20) Llena **requisiciones** para carne. **REQ**

Cocinero cocina fría
Sus obligaciones principales son :
1) Conocer **recetas standar**. **RE**
2) Supervisar a su ayudante.
3) Preparar **alimentos fríos**. **RE**
4) Preparar **ensaladas**. **RE**
5) Preparar **cocteles**. **RE**
6) Elaborar **salsas**. **RE**
7) Elaborar **sandwich**. **RE**
8) Revisar cámara fría (refrigerador).
9) Elaborar órdenes de **carnes frías**. **RE**
10) Preparar **frutas frescas, frutas en almíbar**, etc. **RE**
11) **Requisitar materia prima** al almacén. **REQ**
12) Aplicar técnicas de almacenamiento de alimentos.

Ayudante de la cocina fría
1) Prepara **guarniciones**. **RE**
2) Prepara **ensaladas**. **RE**
3) Elabora **salsas y mayonesa**. **RE**
4) Rebana carne.
5) Limpia legumbres y fruta.
6) Elabora **flanes y gelatina**. **RE**
7) Ayuda a decorar el bufete.
8) Revisa existencia de alimentos. **AL**
9) Sustituye al cocinero cuando éste se ausenta.
10) Proporciona los siguientes alimentos :
jugos, frutas en almíbar, galletas, helados, leche, cremas, jarabe natural, té helado, refrescos.
11) Prepara **bebidas frías**. **RE**
12) Mantiene limpia su zona de trabajo.
13) Llena requisiciones al almacén. **REQ**

Pastelero
Sus principales obligaciones son :
1) **Panadería, pastelería, confitería, heladería. RE**
2) Elabora **postres. RE**
3) Calcula **precios de postres. PRU, RE**
4) Elabora **flanes y gelatinas. RE**

Chief steward
Sus obligaciones específicas son :
1) Verifica limpieza de cocina.
2) Coordina montaje y desmontaje de eventos.
3) Verifica limpieza de equipo en las diferentes áreas.
4) Checa preparación de café.
5) **Suministra** agua y hielo. **AL, REQ**
6) Realiza **pedidos de material** de limpieza. **REQ**
7) Verifica plan de limpieza.
8) Controla el **almacén. AL**
9) Supervisa **inventarios** mensuales. **AL**
10) Verifica el desalojo de la basura.
11) Solicita **reparación** de equipo. **MA**
12) Controla pérdidas de **equipo. EQ**
13) En cuanto a su personal, sus obligaciones son :
Entrevistar a aspirantes.
Capacitar al personal. **CA**
Evaluar.
Elaborar **horarios** de trabajo. **HO**
Programar **días de descanso y vacaciones. HO**
Aplicar suspenciones. **EM**
Solicitar **personal eventual. CO**
Supervisar **disciplina y aseo. EM**
Asignar tareas al personal.
Elaborar reportes para el departamento de personal. **REP**
Realizar juntas con su **personal.**
Controlar su **nómina. NO**
Autorizar **tiempo extra. EM**
14) Colaborar en la **elaboración del presupuesto. VE**
15) Supervisar **mantenimiento** al inmueble. **MA**
16) Elaborar **requisiciones de compra** de material. **REQ, CO**

Floor steward
Es el responsable ante el chef steward de la operación del departamento durante un turno determinado. Sus funciones son :
1) Supervisar la asistencia del personal.

2) Asignar y supervisar tareas de personas a su cargo.
3) Verificar pendientes de turno anterior.
4) Verificar programación de **eventos especiales**. **EV**
5) Supervisar el buen funcionamiento del equipo.
6) Hacer **requisiciones** por faltantes. **REQ**
7) Supervisar orden y **limpieza del almacén**. **MA, AL**
8) Es el responsable del montaje y desmontaje de bufetes.

Steward
Sus obligaciones son :
1) Limpieza general de utensilios del área de cocina (cerrador), durante la noche.
2) Atiende **peticiones de materias primas** y las suministra. **REQ, AL**

Área 2)
Mantenimiento
En esta área, debido a que es muy costoso tener un técnico por cada una de las actividades de mantenimiento dentro de restaurante, se tomó la decisión de contratar a una persona con el mayor número de conocimientos en cuidado y conservación de la instalaciones, siendo éstas : refrigeración, aire acondicionado, electricidad, plomería, albañilería, pintura, carpintería y jardinería. Contratar un técnico especializado dado el caso que el empleado encargado del mantenimiento no tuviera los conocimientos para solucionar un problema o llevar a cabo una tarea especifica.

Área 3)
Contabilidad
Esta área es de vital importancia dentro de la empresa, debido a que ésta elabora y presenta estados financieros, tales como ingresos y egresos, presupuestos e informes estadísticos necesarios, también se encarga de llevar todos los movimientos que se realizan con empleados, tales como la nómina, afiliación al seguro social etc.
A continuación se presenta algunas de las necesidades de información :

1) **Informe diario** de ventas. **VE**
2) Preparar **mensualmente reportes. RE**
3) **Semanalmente la nomina. NO**
4) Mensualmente estados financieros. **VE, CO**
5) **Presupuestos**. **CO**
6) Revisar **periódicamente el almacén**. **AL**
7) Elaborar **balance general**. **CO, VE**
8) Elaborar **declaración de impuestos**. **CO**
9) Recibe **facturas de proveedores**. **PR**
10) Elaborar **cheques para proveedores**. **CO**
11) Llevar reporte de **asistencia de empleados**. **NO**
12) **Calcula sueldos y salarios**. **NO**

Como se mencionó en el inicio de este trabajo, cada uno de los departamentos que pertenecen al restaurante tienen actividades específicas que requieren muchos procesos internos propios de estas áreas, tal es el caso para la contabilidad del restaurante, por ello, sólo se presentan algunas de las operaciones que éste realiza, pero se dejan bases sólidas en el diseño lógico para que de aquí se tomen los datos requeridos, para tomar una decisión importante dentro de la empresa.

Apéndice g)

Introducción a SQL-Server Sybase

APÉNDICE G
Introducción a SQL-server Sybase

En este apéndice se observan algunas de las cláusulas más utilizadas cuando se consulta una **BD**, tomando para ejemplificarlas algunas ocurrencias que se presentan diariamente en el restaurante.

Cuando un diseñador se encuentra desarrollando el análisis de un determinado problema, se encuentra con diferentes términos que se relacionan en su significado, además estos términos tienen que ver con la etapa de desarrollo del mismo, ya que dependiendo de ésta el diseñador se refiere a ellos de la siguiente forma :

Analogías en términos utilizados

SQL	Modelo-ER	Representación física
tabla	entidad	archivo
columna	atributo	campo
renglon	instancia	registro
llave primaria (PK)	llave primaria (PK)	
llave foránea (FK)	llave foránea (FK)	

La anterior figura se interpreta a continuación :
En **SQL** una **tabla** es una parte única de la estructura lógica, una **columna** describe parte de la tabla, un **renglón** es un registro de dicha tabla, en el **modelo-ER**, (ya se describió en el capítulo II), **en la representación en disco**, un **archivo** va a ser la representación física de la **entidad** y contendrá la misma información que la entidad que represente; un **campo** es la representación física de un **atributo** y contendrá la misma información; un **registro** es la representación física de una instancia y contendrá información en un momentos dado.

Para los ejemplos que se muestran a continuación se debe tener en cuenta la analogía de términos mostrada anteriormente, los cuales se mostrarán de la siguiente forma :

Primero se muestra el nombre de la cláusula, tomando en cuenta que ésta puede constar de varias; segundo, se da una breve explicación de lo que se puede hacer con la cláusula; tercero se presenta la sintaxis de la cláusula, así como un ejemplo acorde con la sintaxis mostrada y que solucione la ocurrencia que se esté presentando.

En ocasiones no se presentará la sintaxis de la cláusula , ya que esta representación puede resultar obvia por los anteriores ejemplos, debido a que lo único que se agrega es una cláusula más.
Durante los ejemplos se utilizará continuamente la palabra cláusula, por ésta se entenderá un conjunto de palabras que expresan un pensamiento y que son ejecutadas por comandos en **Sybase**.
Una vez convenido lo descrito anteriormente, se inicia la descripción de las cláusulas :

1) Cláusula select / from / where

Esta cláusula es un conjunto de palabras que expresan un pensamiento, donde **select** sirve para seleccionar todas o sólo las columnas necesarias para satisfacer una ocurrencia determinada; **from** indica de qué tabla o tablas se extrae la información; **where** permite obtener datos que cumplan con ciertas características.

Sintaxis :	**Ejemplo :**
select select_list from table_list	**select * from clientes** **go**
select [distinct] select_list from table_list	**select distinct em_id from ventas** **go**
select select_list from table_list **where search_conditions**	**select * from clientes** **where cl_profesion = ' LICENCIATURA '** **go**

En el anterior ejemplo, el comodín * indica que se desea extraer todas las columnas de la tabla clientes.

La cláusula **distinct** indica que no permitirá valores repetidos en la columna que esté evaluando, es este caso en **em_id** que corresponde a la columna donde se almacenan las claves de los empleados que realizaron una determinada venta; también se puede ver que se extrae información de la tabla clientes, siempre y cuando cumplan la condición de que hayan estudiado una licenciatura.

En los ejemplos anteriores se observa al final de cada consulta, la cláusula " **go** " , lo que indica es la ejecución de la consulta; esta misma cláusula sirve para todos los ejemplos.

2) Condiciones de la cláusula where

Las condiciones de la cláusula **where** pueden ser las siguientes :

a) Operadores de comparación (=, <, >, >=, <=, <>, !=, !>, !<)	**select * from almacen** **where al_cminima >= 33**
b) Rangos (between and not between)	**go**
c) Patrones a seguir (like and not like),'CA%'	**select * from empleados**
d) % cualquier cadena de ceros o más caracteres, 'CA'	**where de_id like in ('A%')**
[_] cualquier carácter simple dentro de este rango, '70_'	**go**
[^] cualquier carácter que no este especificado en el rango	**select * from clientes**
'[A-D]' rango, '[A-D]%' primer carácter y siguientes caracteres	**where em_nombre = '[A-B] % '**
	go
e) Valores indefinidos (is null and is not null)	**select * from ventas** **where ve_total is null**
f) Mientras esté en (in and not in)	
g) Combinaciones (and, or)	

Conectando condiciones
Cuando se une más de un operador lógico se hace utilizando los siguientes conectores **(not, and, or).**

```
select * from eventos
where ev_descripcion = 'ANIVERSARIO DE BODAS '
and ev_fecha = getdate()
```

En los anteriores ejemplos se observa que se extraen todos los alimentos y bebidas cuya cantidad existente sea mayor o igual a 33 unidades.
En el siguiente se desea extraer toda la información de los empleados cuya área de trabajo (departamento) sea la cocina, ya que ésta tiene como primer caracter del identificador la letra ' A ' , y el comodín ' % ' , indica que se traerán todos los registros cuya primer letra sea ' A ' sin importar los caracteres siguientes.
En el siguiente ejemplo se observa que se desea visualizar a todos los clientes cuyo nombre inicie con la letra ' A o B ' sin importar los siguientes caracteres.
En el siguiente ejemplo se observa que se desea visualizar todas aquellas ventas que aún no se han consumado, ya que la cláusula is null indica que esta columna no contiene datos, es decir, no se ha registrado la venta en su totalidad.
También se observa que un conector de condiciones **and** sirve para unir dos ocurrencias, en este caso se va a visualizar toda la información de la tabla eventos cuya descripción sea ' ANIVERSARIO DE BODAS ' y cuya fecha sea el día de hoy, representada por la función agregada **getdate().** Respecto a estas funciones, como su nombre lo indica, se agregan a ciertas ocurrencias para una mejor manipulación, recordando que estas funciones ya están definidas en **Sybase** y sólo se tiene que hacer mención a ellas para obtener el resultado deseado.

3) Renombrando columnas
Ahora se muestra la forma como se renombran columnas en la **BD**, esto se hace cuando se va a desplegar información, pero debido a la abstracción del nombre de las columnas para usuarios que no están familiarizados con la **BD**, les es difícil comprender el significado de las mismas. Para evitar la interpretación errónea de la información presentada, se renombran las columnas con un conjunto de caracteres que expresen explícitamente la columna a la que se refiere.

Sintaxis :

```
select  new_column_name = original_column_name
select  'new_column_name' = original_column_name


select  original_column_name  new_column_name
```

Ejemplo :

```
select  numero_de_cta = cl_id,
nombre_del_cliente = cl_nombre  from
clientes
where cl_alergia is not null
go
select  cl_rfc  registro_federal
from clientes
go
```

De los ejemplos anteriores se extrae el identificador de un cliente, así como su nombre, si no se renombran las columnas, la información aparecerá de esta forma :

cl_id	**cl_nombre**
CL0001	CARMEN ALCANTARA

Lo que se hace es renombrar las columnas para que se entienda que **cl_id** representa a clave del cliente y **cl_nombre** la columna donde se almacena el nombre del cliente y se muestra de la siguiente forma :

numero_de_cta	**nombre_del_cliente**
CL0002	CARMEN OLIVERA

4) Adicionando comentarios

También se pueden adicionar comentarios a la cláusula **select,** de tal forma que al igual que los anteriores ejemplos tome sentido la información que se extrae para los usuarios que la solicitan.

Ejemplo :

```
select 'el nombre del platillo es ', re_nombre from recetas
where re_id like in ('A0%')
go
```

Del ejemplo anterior se adiciona el comentario **' el nombre del platillo es '** a todos los platillos que se elaboran en el restaurante.

5) Operadores aritméticos

También **Sybase** proporciona operadores aritméticos, con los cuales se pueden hacer infinidad de operaciones.

Operador :	**Ejemplo :**
' + ' adición	**select em_sueldo + em_sueldoacum from empleados**
' - ' sustracción	**go**
' * ' multiplicación	**select em_sueldo - em_sueldoacum from empleados**
' / ' división	**go**
' % ' residuo	

Del ejemplo anterior se observa que se desea saber cuánto ha ganado un empleado hasta el día de hoy, para lo cual se suman **em_sueldo** más **em_sueldoacum** que representa el total de percepciones que ha recibido el empleado. Estos operadores pueden ser usados en cualquier columna numérica.

6) Valores nulos

Algo que se debe tomar en cuenta siempre son los valores nulos, ya que por medio de esta cláusula se pueden conocer infinidad de ocurrencias alrededor de éstos.

Ejemplos :

```
select ab_nombre, ab_descripcion from alimentos
where ab_temperatura is null
go
```

```
select ab_nombre, ab_descripcion from alimentos
where ab_temperatura is not null
go
select ab_nombre, ab_descripcion from alimentos
where ab_fcaducidad >= getdate()
and ab_fproduccion is null
go
```

Del ejemplo anterior se interpreta que se visualizará el nombre y la descripción de algunos alimentos, los cuales no necesitan temperatura para su conservación en buen estado, caso contrario ocurre en el segundo ejemplo, donde se muestran alimentos que sí necesitan cierta temperatura para su conservación, en el tercer caso se muestran alimentos cuya fecha de caducidad sea mayor o igual al día de hoy, y aún no se han producido.

7) Cláusula select / where / order by

Esta cláusula permite seleccionar y ordenar como uno lo desee, para una mejor visión de la **BD**, ya sea ascendente o descendente.

Sintaxis :

```
select | distinct | select_list from tabla
[ where search_conditions ]
[ order by { column I expression }
 | asc I desc ] [,...]]
```

Ejemplo :

```
select * from compras
where co_fcompra between '01-febrero-96 00:00AM'
and '01-febrero-96 23:59PM'
order by co_descripcion
go
```

En el ejemplo anterior se visualizan todas las compras efectuadas el día 1 de febrero de 1996 y se ordenan por la descripción de la compra en orden ascendente.

8) Funciones agregadas

También se debe tener en cuenta todas las bondades que dan las funciones agregadas, ya que sólo hay que hacer mención de éstas donde se necesite, para que realicen su función.

Funciones agregadas	**Ejemplo :**
count(*) regresa el número de registros seleccionados	**select count(*) from ventas** **go**
count(col_name) regresa el número de valores no nulos en la columna	**select count(ve_factura) from ventas** **where ve_factura=1** **go**
max(col_name) regresa el valor más grande de la columna	**select max(re_pventa) from recetas** **go**
min(col_name) regresa el valor mínimo de la columna	**select min(re_pventa) from recetas** **go**
sum(col_name) regresa el total de valores de la columna	**select sum(ve_total) from ventas** **go**
avg(col_name) regresa el promedio de valores en la columna la función **avg** , no toma en cuenta valores nulos	**select avg(ve_pventa) from recetas** **go**

```
select min(prd_cunitario),
max(prd_cunitario) from productos
go
```

Sintaxis para los anteriores ejemplos :

```
select aggregate_name ([distinct]expression)
from table_name
[ where ..... conditions ]
go

select [distinct] select_list
[ from table [, ... ] ]
[ where search_conditions ]
[ group by [ all ] aggregate-free_expression [ .... ] ]
[ having search_conditions ]
[ asc I desc ] [, .... ] ]
go
```

De los anteriores ejemplos, en el primero la función **count(*)** regresa el número de registros de la tabla ventas. En el segundo, la función **count(ve_factura)** regresa el número de ventas que se han realizado con una factura a petición del cliente.
En el tercer caso, **max(re_pventa)** regresa el precio más alto de un alimento que se vende en el restaurante, el caso contrario para la función **min(re_pventa);** para la función **sum(ve_total)** regresa la suma de todas las ventas en el restaurante (esto es el dinero que ha entrado al restaurante por ventas), en el siguiente caso se visualizará el promedio de los precios que se tienen en los alimentos; el cuarto caso es igual al tercero.

9) Funciones agregadas con la cláusula distinct

Ahora voy a mostrar una bondad que otorga la cláusula **distinct**, ya que como mencioné anteriormente no permite registros duplicados, es opcional en las funciones **sum, avg y count (col_name)** ; no se permite con **min, max, count(*)** , y se usa con nombre de columnas, no con expresiones.

Ejemplo :

```
select ve_id.ve, avg(distinct re_pventa) from recetas, ventas
where ve_id.ve = 'EM0001'
go
select ve_id.ve, avg(re_pventa) from recetas, ventas
where em_id='EM0002'
go
select count( distinct em_id ) from recetas
go
```

De los anteriores ejemplos, en el primero se obtiene el promedio de aquellos precios de venta, siempre y cuando no sean duplicados en la tabla recetas y los haya elaborado el empleado EM0001; en el segundo ejemplo se obtiene el promedio del precio de venta de platillos, siempre y cuando los haya elaborado el empleado EM0002. En el tercer caso se desea saber qué empleados elaboran las recetas, en este caso hay empleados que preparan más de una receta; para esto se agrega la cláusula **distinct** , para no repetirlos.

10) Tratando valores nulos
Otra bondad de **Sybase** es que permite sustituir valores nulos por valores reales no nulos.

Sintaxis :
is null (substituye un valor real por un valor nulo)
is null (colum_which_might_be_null, value_to_use_if_null)

Ejemplo :
```
select avg ( is null (re_pventa, $ 25,00)) from recetas
```

En el ejemplo anterior se sustituye el precio de venta nulo por un valor real que en este caso es 25 pesos.

11) La cláusula select / group by
Esta cláusula permite tratar grupos de registros y realizar operaciones sobre éstos, teniendo en cuenta que estas operaciones se realizan de acuerdo al grupo que pertenezcan, para esto se deben tomar en cuenta las siguientes convenciones :

a) Se usa con funciones agregadas o en una lista seleccionada o con expresiones que no contengan funciones agregadas.
b) Las funciones agregadas serán calculadas.
c) Todos los valores nulos en un grupo serán calculados y tratados como un grupo.
d) **group by** despliega las columnas y expresiones en el **select** que no estén en la lista de funciones agregadas.

Ejemplo :
```
select re_quien, precio promedio de venta = avg(re_pventa)
from recetas
group by em_id
go
```

En el anterior ejemplo se obtendrá el promedio del precio de venta para cada uno de los empleados que elaboran las recetas, este promedio será individual para cada uno de los empleados.

12) La cláusula select / group by / where / having
Restringe aún más el desplegado de los resultados, ya que estos deben tener una condición extra dada por la cláusula **having**.

Condiciones :
a) **where** elimina los registros que no cumplan la condición antes de que se agrupen y no acepta una función agregada.

c) **having** restringe el desplegado de los archivos seleccionados.
d) Amplía las condiciones después de haber sido formada.

Ejemplo :
```
select ve_id, sum(ve_total) from ventas
where ab_id='AB0001'
go

select avg(re_pventa), re_nombre
from recetas
group by re_nombre
having avg( re_pventa) > 12,00
go
```

En el ejemplo anterior se obtiene primero la suma del total de ventas que se han realizado, cuya clave del alimento sea AB0001 y que tiene por descripción ' PLATO DEL DIA' ; en el segundo se obtiene el promedio de los precios de venta de las recetas, y los ordena por el nombre de la receta, además sólo despliega aquellas recetas cuyos promedios de precios de venta sean mayores que 12 pesos.

13) Joins

Una vez que se comprendió lo anterior, se presenta una bondad de gran utilidad para navegar por la **BD** , a la cual se le conoce como **joins** (uniones o relaciones). Estos joins permiten unir la información de más de dos tablas, cuyas condiciones necesariamente son evaluadas en las tablas involucradas para su consulta a través de su **PK** y **FK,** respectivamente. Para lograr esto se presentan los pasos que se deben seguir para su óptima ejecución.

Pasos :

a) Decidir qué columnas se necesitan ver.

b) Usar un diagrama de conectividad.

c) Seguir las líneas en el esquema para encontrar qué se necesita para conectar las tablas.

Para lograr esto se debe tomar también en cuenta que es una operación multitablas , no acepta valores nulos , que las columnas con el mismo nombre en diferentes columnas deben ser precedidas por la tabla y que los nombres de columnas no necesitan ser igual para los joins.

Sintaxis :

```
select [ table. ] column_name, [ ... ] from {table }
{ , table } [ , .... ]
[ where search_conditions ]
```

Ejemplo :

```
select cl_nombre, clientes.em_id, em_nombre from clientes, empleados
where clientes.em_id = empleados.em_id
go
select alimentos.ab_id, ab_nombre, ab_existencias
from alimentos, provecedores
where alimentos.ab_id = provecedores.ab_id
go
select ab_id, ab_nombre, ab_existencias, pr_condicion
from alimentos, provecedores
where alimentos.ab_id = provecedores.ab_id
go
select alimentos.al_id, prd_cunitario, ab_existencias,
total _a_pagar = ab_existencias * prd_cunitario from alimentos, productos
where alimentos.ab_id = productos.ab_id
order by ab_existencias
go
select alimentos.al_id, sum(al_counitario), ab_existencias,
total _a_pagar = ab_existencias * prd_cunitario
from alimentos, productos
where alimentos.ab_id = productos.ab_id
group by ab_existencias
go
```

Del ejemplo anterior se desea saber qué empleados trabajan atendiendo a clientes, para esto es necesario comparar los identificadores **em_id** en las tablas **clientes y empleados** , para lograr esto basta colocar el nombre de una de las tablas, anteponiendo al identificador que se desea evaluar en la cláusula select, y colocando las tablas de las cuales se va a extraer la información en la cláusula **from** , una vez que se tiene el identificador a evaluar en la cláusula **where**, se evalúa en ambas tablas el identificador, anteponiendo el nombre de la tabla separada por un punto del identificador a evaluar.
Este mismo criterio se sigue para los demás ejemplos, teniendo estos operadores aritméticos y algunas cláusulas que ya se explicaron anteriormente.

14) Alias

Como parte adicional y para facilitar los **joins**, se pueden definir **alias**, que como se explicó anteriormente, sirve para identificar una entidad de otra con otro nombre, en este caso será un nombre más corto que facilite la sintaxis al unir las tablas, de tal forma que estos alias se especifican en la cláusula **from,** donde se definen las tablas a consultar, para ello basta con colocar el alias con el que se conocerá a la tabla en consulta, separada por un espacio en blanco.

Sintaxis :

```
select select_list
from table_name alias_name1, table_name alias_name2
where  alias_name1.column_name = alias_name2_column_name
```

Ejemplo :

```
select cl_nombre, clientes.em_id, em_nombre
from  clientes cl, empleados em
where cl.em_id = em.em_id
go
select alimentos.ab_id, ab_nombre, ab_existencias
from alimentos al, proveedores pr
where al.ab_id = pr.ab_id
go
select  ab_id, ab_nombre, ab_existencias, pr_condicion
from alimentos al, proveedores pr
where al.ab_id = pr.ab_id
go
select  alimentos.al_id, prd_cunitario,  ab_existencias
total _a_pagar = ab_existencias * prd_cunitario
from alimentos ab, productos prd
where ab.ab_id = prd.ab_id
order by ab_existencias
go
select  alimentos.al_id, sum(prd_cunitario), ab_existencias
total _a_pagar = ab_existencias * prd_cunitario
from alimentos ab, productos prd
where al.ab_id = prd.ab_id
group by  ab_existencias,
go
```

Respecto a los anteriores ejemplos, se presentaron en los **joins**, pero aquí están utilizando sus respectivos **alias**.

15) Subqueries (subconsultas)

Otro método alternativo para consultas de ocurrencias que se deben hacer en las tablas, son los subqueries, los cuales se utilizan con la cláusula **select** y se pueden usar como parte de expresiones al actualizar, insertar o borrar, se usan debido a que algunas ocasiones son más fáciles de entender que las cláusulas joins y para mejorar la estructura de programas en lugar de los **joins**.

Estos subqueries constan de varias restricciones tales como :
Sólo las columnas seleccionadas serán mostradas, sólo se puede incluir una columna en la lista del **subquery** (más de una columna no es permitida), además de que las columnas que se comparan en el **subquery** deben ser del mismo tipo y longitud que los del **subquery** , es decir, deben ser compatibles, ya que si se relacionan columnas de diferente tipo se producen errores al momento de su ejecución, también hay que tener en cuenta que la cláusula **distinct** no puede ser usada en **subqueries** que incluyan la cláusula **group by**.
También se debe tomar en cuenta para que los subqueries tomen más de un valor, se debe usar la cláusula **in**, además de que éstos se pueden usar con operadores de comparación y tener presente que un **subquery** puede tener más **subqueries anidados**.

Los subqueries sólo regresan un valor.

En los siguientes ejemplos se mostró.lo descrito anteriormente.

Sintaxis :

```
select select_list
[ from {table } [, ..... ] ]
[ where search_conditions ] =
( select subquery_select_list
[ from { table } ] , ... [
] where search_conditions ]
[ group by aggregate_free_expressions [ , ... ] ]
[ having search_conditions ] )

[ group by aggregate_free_expressions [ , ... ] ]
] having search_conditions ]
[ order by { { table }.column I select_list_number I
expression } ]asc I desc [ ] , ... ] ]

[ compute row_aggregate ( column ) [ , ... ]
] by column ] , ... ] ]
```

Ejemplos :

Usando dos queries :

```
select em_id from empleados
where em_name = 'ZORAYDA GONZALEZ ALVAREZ '
go
select max(ve_total) from ventas
where em_id = 'EM0001' and ve_fecha
between '11-Oct-96 00:00AM' and '11-Oct-96 23:59PM'
go
```

Usando join :

```
select em_nombre, ve_total from empleados, ventas
where empleados.em_id = ventas.em_id
and em_name = 'ZORAYDA GONZALEZ ALVAREZ'
between '11-Oct-96 00:00AM' and '11-Oct-96 23:59PM'
go
```

Usando subqueries :

```
select ve_total from ventas
where em_id =
        ( select em_id from empleados
        where em_nombre = 'ZORAYDA GONZALEZ ALVAREZ' )
between '11-Oct-96 00:00AM' and '11-Oct-96 23:59PM'
go
```

En los anteriores ejemplos se tiene, para el primer caso se desea saber el nombre del empleado que realizó la mayor venta el día '11 de octubre de 1996', para esto, si se consulta la tabla ventas no regresará el nombre de empleado, lo que se obtiene es la clave con que se identifica al empleado, pero este no es el dato que se desea obtener, para lograr esto se tienen que consultar ambas tablas.
En el primer ejemplo se realizó la consulta con dos queries , en el primer query consultó la clave con que se identifica al empleado ' ZORAYDA GONZALEZ ALVAREZ', una vez que se obtuvo la clave con que se identifica al empleado , la consulto en ventas con la condición que me interesa, en este caso la mayor venta para el día indicado; en el segundo ejemplo se realiza la misma consulta, utilizando un join; en el tercer ejemplo se realiza la consulta, utilizando un subquery.

En los subqueries se pueden usar los operadores de comparación con la siguiente sintaxis :

Sintaxis :

```
select select_list
[ from { table } [ ,... ] ]
[ where expressions { = | != | < | > | <= | >= | } [ any | all ]
( subquery )
```

Ejemplo :

```
select distinct re_nombre, re_pventa from recetas
where re_pventa > 20,00
and prd_id =
        ( select pr_id from proveedores
        where pr_nombre = 'COMERCIAL S.A. DE C.V.' )
go
```

Para terminar esta introducción se presenta la sintaxis de las cláusulas que sirven para actualizar, borrar y suprimir datos en la **BD**.

16) Cláusula delete

Esta cláusula sirve para borrar uno o varios registros que cumplan con ciertas condiciones.

Sintaxis :

```
delete [from] table_name
  where current of cursor_name
```

Ejemplo :

```
delete clientes
where cl_id='CL0010'
go
```

En este caso lo que borra es el cliente, cuya clave es CL0010.

17) Cláusula update

Esta cláusula se utiliza cuando se quiere modificar el contenido de uno o varios registros que cumplan con ciertas condiciones.

Sintaxis :

```
update { table_name | view_name}
  set column_name1 = { expression1 | NULL
  | (select_statement ) }
  [, column_name2 = { expression2 | NULL
  |   (select_statement) } ...
  from table_name, ], table_name ...]
  [where search_conditions]
```

Ejemplo :

```
Update clientes set
set cl_nombre='MOISES ALVAREZ RODRIGUEZ'
where cl_id='CL0013'
go
```

En el ejemplo anterior, tal vez cuando se dio de alta el empleado, se capturó mal el nombre del cliente, debido a esto es necesario modificarlo y capturar el nombre correcto, el cual es MOISES ALVAREZ RODRIGUEZ y cuya clave del empleado es CL00013.

18) Cláusula insert

Esta cláusula sirve para dar de alta uno o varios registros.

Sintaxis :

```
insert table_name
values ( constant1, constant2, ..... )

insert into table_name
values ( constant1, constant2, ..... )
```

Ejemplo :

```
insert clientes
values ('CL0023','JESUS ALVAREZ' )
go
insert into clientes ( CL_ID, CL_NOMBRE )
values ('CL0024','CARMEN OLIVERA')
go
```

En los ejemplos anteriores se realizó la inserción de dos clientes con sus claves y nombres respectivos, para lo cual, en la primer inserción no se presentan los nombres de las columnas donde se van a insertar los datos. En el segundo ejemplo se presentan las columnas donde se van a realizar las inserciones; para ambos casos es importante capturar los datos en el orden correcto, ya que si no se hace de esta forma **Sybase** puede mostrar errores.

Una vez que se presentó esta **introducción a SQL** en este apéndice, se han dado las bases suficientes para poder manipular la **BD** y poder comprender mejor el capítulo IV.

Esta introducción, como su nombre lo dice, es sólo una visión de lo que se puede hacer al manipular una **BD** con **Sybase**, ya que esta visión **Sybase** sirve para poder desarrollar aplicaciones que ayuden a mantener la consistencia e integridad de la **BD**, cuando se actualice, borre o se lleve a cabo alguna alta a la base de datos de algún registro en particular, ya que **Sybase** es un manejador tan poderoso, que una visión total de lo que es **Sybase** sería tema no para uno, sino para varios trabajos de manipulación de **BD**.

La sintaxis descrita en este apéndice se tomó tal como se muestra en las bibliografías siguientes :

[SYBASE,INC,.]
SISTEM 10 FAST TRACT TO SQL SERVER
SYBASE, INC, USA 1993

[SYBASE,INC,.]
SISTEM 10 INTRODUCTION TO SQL
STUDEN GUIDE
SYBASE, INC, USA 1993