Universidad Nacional del Santa

Facultad de Ingeniería

EAP de Ingeniería de Sistemas e Informática

MANUAL

BASE DE DATOS

- - - Uso Interno - - -

Ing. Hugo Caselli Gismondi

Chimbote

8va Edición – 2010

V8.0

INDICE

Introducción

El presente documento tiene como objetivo fundamental servir como guía didáctica para la asignatura de Base de Datos para los alumnos del VII ciclo de la Escuela Académico Profesional de Ingeniería de Sistemas e Informática de la Universidad Nacional del Santa.

El contenido de este material de lectura corresponde al silabo oficial del curso, y está dividido de acuerdo a los temas que se consideran fundamentales para los objetivos académicos propios para el conocimiento del Profesional de Ingeniería de Sistemas. Como son El modelado de datos, el diseño conceptual de bases de datos, Lenguaje de consulta SQL, protección, seguridad de los datos y alternativas en la distribución de datos.

Es necesario anotar que por si solo este material de lectura no es suficiente para su total comprensión, sino que tiene que ir acompañada de una explicación detallada brindada por el profesor del curso.

PRIMERA UNIDAD

SEMANA 01: INTRODUCCIÓN A LOS SISTEMAS DE BASES DE DATOS MODELADO DE DATOS

INTRODUCCIÓN A LOS SISTEMAS DE BASES DE DATOS

1. OBJETIVOS DEL DISEÑO DE ALMACENAMIENTOS DE DATOS

Algunas personas consideran el almacenamiento de los datos como la esencia del sistema de información. El impacto de la estructura de datos sobre la estructura de programa y la complejidad procedimental, hace que el diseño de datos tenga una gran influencia en la calidad del software.

Independientemente de las técnicas de diseño usadas, bs datos bien diseñados pueden conducir a una mejor estructura de programa, a una modularidad efectiva y a una complejidad procedimental reducida.

Los objetivos generales a lograr en el diseño de la organización de almacenamientos de datos son:

1) **Disponibilidad de datos:** Los datos deben estar disponibles para cuando el usuario desee usarlos. Sólo se debe negar el acceso a datos reservados o confidenciales por razones legales.
2) **Accesibilidad**: Los datos deben ser fáciles de acceder por los usuarios. Si no se pueden acceder fácilmente, no serán utilizados (y eventualmente pueden ser "inventados"). No deben reflejar las necesidades de un sector particular, o serán difíciles de acceder para los otros.
3) **Integridad de datos**: Los datos deben ser precisos (cada valor almacenado debe estar dentro de un rango aceptable respecto del valor real) y consistentes (no deben reflejar realidades distintas ya sea en el tiempo (las Cuentas a Pagar no cierran contra las Facturas Pendientes) como en los datos relacionados (un Gerente de Área no puede ser un empleado de categoría mínima). El conjunto de datos debe representar fielmente a la organización.
4) **Almacenamiento eficiente.** Eficiencia tanto en el sentido de ocupar el mínimo espacio de almacenamiento (evitando la redundancia), como en el de evitar el desperdicio de las capacidades, eligiendo una adecuada estructura de almacenamiento según en tamaño de los datos individuales.
5) **Actualización y recuperación eficiente**. Las estructuras de datos deben facilitar el acceder rápidamente a ellos según, se lo requieran los procedimientos a realizar. Como el logro estos dos últimos objetivos implica, muchas veces, el tomar medidas contrarias, es necesario lograr una solución de compromiso.
6) **Recuperación dirigida de la información**: la información obtenida de los datos almacenados debe contar con un formato útil que facilite la administración, la planificación, el control y la toma de decisiones.

2. ARCHIVOS CONVENCIONALES Y BASES DE DATOS

En un sistema de información basado en computadora se cuenta con dos enfoques para el almacenamiento de los datos.

El primer método consiste en almacenar los datos en archivos individuales, exclusivos para una aplicación en particular. La figura 1 ilustra una organización con diversos sistemas de información que utilizan archivos convencionales independientes. Existen tres archivos individuales: ARCHIVO-VENTAS, el cual contiene la información de los datos históricos, ACTIVIDADES-ACTUALES, el cual se actualizan con frecuencia y ARCHIVO-PERSONAL, que contiene las direcciones, los títulos y datos similares. Observe que NUM y NOMBRE existen en ambos archivos. Además del esfuerzo adicional requerido para introducir el

NOMBRE hasta tres veces, el cambio de nombre (por ejemplo, por un cambio en el estado civil) requeriría de la actualización de tres archivos individuales,

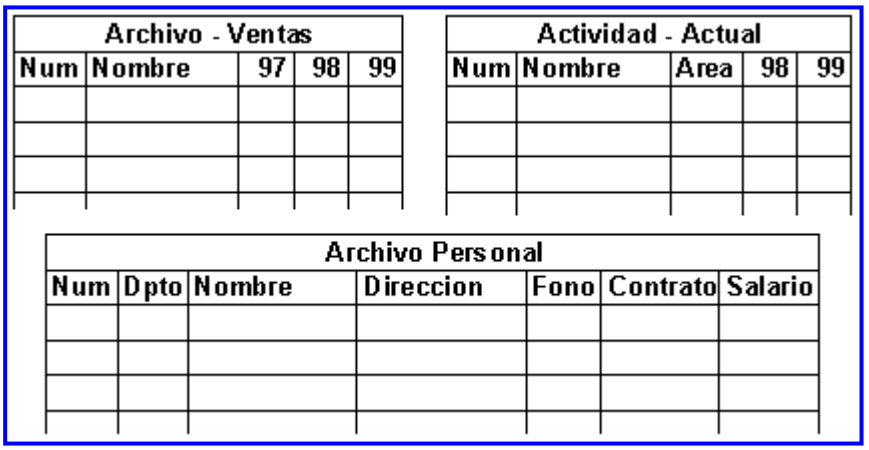


FIGURA 1.- El uso de archivos separados con frecuencia implica que los mismos datos se almacenen en más de un lugar.

El segundo método para el almacenamiento de datos en un sistema basado en computadora, involucra la elaboración de una base de datos.
Una base de datos es un almacenamiento de datos formalmente definido, controlado centralmente para intentar servir a múltiples y diferentes aplicaciones.

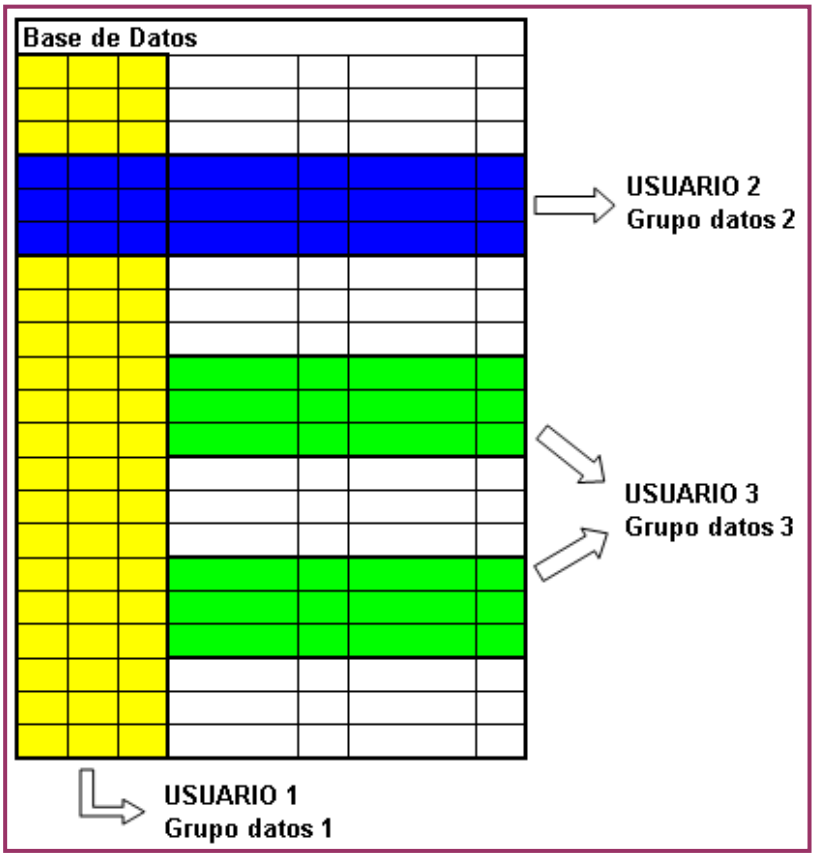


FIGURA 2.- El enfoque de base de datos permite que diferentes usuarios compartan la misma base de datos, teniendo acceso a diferentes grupos de datos.

La *figura 2* muestra cómo los diferentes usuarios de distintos departamentos dentro de una organización comparten la misma base de datos. Un usuario puede seleccionar una parte de la base de datos, como se muestra en el conjunto de datos 1, o ciertos renglones, como en el conjunto de datos 2.
El conjunto de datos 3 contiene columnas selectas y utiliza ciertos renglones para calcular los totales.
Estos conjuntos de datos se superponen y a diferencia de los archivos convencionales (donde serían redundantes), solamente se almacenarían una vez en la base de datos.

2.1. Los Archivos Convencionales

Los archivos convencionales, sin duda alguna, permanecen como una manera práctica de almacenar los datos de ciertas aplicaciones. Un archivo convencional se puede diseñar y elaborar de manera rápida, reduciendo los problemas de disponibilidad de datos y de seguridad. Cuando el diseño de los archivos se realiza de manera cuidadosa, toda la información necesaria queda incluida y se reduce el riesgo de omitir datos de manera accidental.
Normalmente se tiene la posibilidad de tener en servicio archivos separados, según las aplicaciones que los utilicen. Cuando se requieran nuevas aplicaciones y el tiempo de desarrollo es una consideración importante, el analista de sistemas no puede involucrarse en el problema de rehacer el almacenamiento de los datos sólo con el fin de lograr un enfoque de base de datos. La solución obvia en función de las restricciones de tiempo consiste en limitar el alcance del proyecto, diseñando otro archivo separado para la nueva aplicación.
La velocidad de procesamiento es otra ventaja para el uso de archivos. Hay posibilidad de elegir una técnica óptima para el procesamiento de los archivos de una sencilla aplicación, pero llega a ser imposible alcanzar un diseño óptimo para tareas muy variadas. En consecuencia, si el analista de sistemas se enfrenta al diseño de un sistema para una aplicación específica, cuando la eficiencia del procesamiento es de gran relevancia, el mejor enfoque puede ser el diseño de un archivo individual para tal propósito.
El uso de archivos individuales tiene diversas consecuencias. Veamos el siguiente ejemplo para ilustrarlo:
Considérese parte de una empresa bancaria de ahorros que guarda la información sobre todos los clientes y sus cuentas de ahorro. El sistema tiene diversos programas de aplicación que permiten al usuario manejar los archivos, incluyendo:

- Un programa para hacer cargos o abonos a una cuenta.
- Un programa para añadir una nueva cuenta.
- Un programa para obtener el saldo de una cuenta.
- Un programa para generar estados mensuales.

Estos programas de aplicación los han escrito programadores de sistemas en respuesta a las necesidades de la organización bancaria. Según surge la necesidad, se añaden nuevos programas de aplicación al sistema. Por ejemplo, supóngase que una nueva ley del Gobierno permite al banco de ahorros ofrecer cuentas de cheques. Como resultado, se crean nuevos archivos permanentes que contienen información acerca de todas las cuentas de cheques que mantiene el banco, y puede que sea necesario escribir nuevos programas de aplicación. Así, según pasa el tiempo, se añaden más archivos y programas de aplicación al sistema.
El típico sistema de procesamiento de archivos descrito arriba está apoyado por un sistema operativo convencional. Los registros permanentes se almacenan en varios archivos, y se escribe un número de diferentes

programas de aplicación para extraer registros de y añadir registros a archivos apropiados. Este enfoque tiene un número de desventajas importante:

- **Falta de potencial para evolucionar:** Con frecuencia, los archivos se diseñan con base en las necesidades inmediatas. Cuando llega a ser importante la consulta del sistema mediante una combinación de ciertos atributos, tales atributos pudieran encontrarse en archivos separados, o simplemente no existir. El rediseño de archivos, implica a menudo que los programas que los accesan deban escribirse nuevamente de manera acorde. Esto implica un incremento en el tiempo de programación para el archivo, para el desarrollo y mantenimiento de programas.
- **Redundancia e inconsistencia de los datos**. Puesto que los archivos y los programas de aplicación son creados por distintos programadores durante un largo periodo de tiempo, es probable que los archivos tengan diferentes formatos y los datos pueden estar duplicados en varios sitios (archivos). Por ejemplo, la dirección y el número de teléfono de un cliente determinado pueden aparecer en un archivo que consta de registros de cuentas de ahorros y en un archivo que consta de registros de cuentas de cheques. Esta redundancia aumenta los costes de almacenamiento y acceso. Además, puede llevar a inconsistencias de los datos, esto es, las diversas copias de los mismos datos no concuerdan entre sí. Por ejemplo, una dirección cambiada de un cliente puede estar reflejada en los registros de cuentas de ahorros pero en ningún sitio más del sistema. Resultados de la inconsistencia de los datos.
- **Dificultad para tener acceso a los datos**. Supóngase que uno de los gerentes del banco necesita averiguar los nombres de todos los clientes que viven dentro la ciudad con código postal 7833. El gerente pide al departamento de procesamiento de datos que genere la lista correspondiente. Puesto que esta solicitud no fue prevista cuando se diseñó el sistema original, no hay ningún programa de aplicación a mano que la satisfaga. Existe, sin embargo, un programa de aplicación para generar la lista de todos los clientes. El gerente tiene ahora dos elecciones: O bien tomar la lista de clientes y extraer la información necesaria manualmente, o pedir al departamento de procesamiento de datos que ponga a un programador de aplicaciones a escribir el programa necesario. Obviamente, ninguna de las dos alternativas es satisfactoria. Supóngase que se escribe un programa semejante y que, varios días después, el mismo gerente necesita arreglar esa lista para incluir solo aquellos clientes con un saldo de 10000 dólares o más. Como se esperaba, no existe un programa que genere esa lista. De nuevo, el gerente tiene las dos opciones anteriores, ninguna de las cuales es satisfactoria. Lo que se trata de probar aquí es que esos entornos convencionales de procesamiento de archivos no permiten recuperar los datos de una forma conveniente y eficiente, Deben desarrollarse sistemas de recuperación de datos para uso general.
- **Aislamiento de los datos**: Puesto que los datos están repartidos en varios archivos, y éstos pueden tener diferentes formatos, es difícil escribir nuevos programas de aplicación para obtener los datos apropiados.
- **Anomalías del acceso concurrente**: Para mejorar el funcionamiento global del sistema y obtener un tiempo de respuesta más rápido, muchos sistemas permiten que múltiples usuarios actualicen los datos simultáneamente En un entorno así, la interacción de actualizaciones concurrentes puede dar por resultado datos inconsistentes. Considérese

una cuenta bancaria A, con 500 dólares. Si dos clientes retiran fondos (digamos 50 y 100 dólares, respectivamente) de la cuenta A casi al mismo tiempo, el resultado de las ejecuciones concurrentes puede dejar la cuenta en un estado incorrecto (o inconsistente). En particular, la cuenta puede contener 450 o 400 dólares, en vez de 350 dólares Para prevenir esta posibilidad, debe mantenerse alguna forma de supervisión en el sistema. Puesto que se puede acceder a los datos por medio de diversos programas de aplicación diferentes que no han sido previamente coordinados, esta supervisión es muy difícil de proporcionar.

- **Problemas de seguridad**: No todos los usuarios del sistema de base de datos deben poder acceder a todos los datos. Por ejemplo, en un sistema bancario, el personal de las nóminas solo necesita ver la parte de la base de datos que tiene información acerca de los distintos empleados del banco. No necesitan acceder a información sobre las cuentas de los clientes. Puesto que los programas de aplicación se añaden al sistema de una forma específica, es difícil implantar tales restricciones de seguridad.
- **Problemas de integridad**: La integración de los datos llega a convertirse en una causa de preocupación, ya que los cambios en un archivo, requerirán también la modificación de ciertos datos en otros archivos. Aquellos archivos utilizados esporádicamente bien pueden quedar ignorados durante la fase de la actualización. Por otra parte, los valores de datos almacenados en la base de dalos deben satisfacer ciertos tipos de restricciones de consistencia. Por ejemplo, el saldo de una cuenta bancaria nunca debe caer por debajo de una cantidad prescrita (digamos, 25 dólares). Estas restricciones se hacen cumplir en el sistema añadiendo códigos apropiados en los diversos programas de aplicación. Sin embargo, cuando se añaden restricciones nuevas, es difícil cambiar los programas para hacerlos cumplir. El problema se complica aún más cuando las restricciones implican varios elementos de información de distintos archivos.

Estas dificultades, entre otras, han fomentado el desarrollo de sistemas de gestión de bases de datos.

2.2. LOS SISTEMAS DE BASE DE DATOS

Las bases de datos no son meramente una colección de archivos. Más bien, una base de datos es una fuente central de datos significativos, los cuales son compartidos por numerosos usuarios para diversas aplicaciones. La esencia de una base de datos es el Sistema Administrador de la Base de Datos (DBMS: Database Management System), el cual permite la creación, modificación y actualización de la base de datos, la recuperación de los datos y la emisión de reportes.

A la persona responsable de asegurar que la base de datos satisfaga los objetivos programados se le denomina administrador de la base de datos.

Los objetivos de eficiencia de la base de datos son:

1) Asegurar que los datos puedan ser compartidos por los usuarios, para una variedad de aplicaciones.
2) Lograr que el mantenimiento de los datos sea preciso y consistente.
3) Garantizar que todos los datos requeridos para las aplicaciones presentes y futuras se encuentren siempre disponibles.
4) Permitir que la base de datos evolucione y se adapte a las necesidades crecientes de los usuarios.
5) Permitir que los usuarios desarrollen su propia visión de los datos, sin preocuparse por la manera en que los datos se encuentren almacenados físicamente.

La anterior lista de objetivos nos advierte sobre las ventajas y desventajas del enfoque de la base de datos.
La primera ventaja: el compartir los datos, significa que éstos deben almacenarse una sola vez. Esto a su vez apoya que se mantenga la integridad de los datos, ya que el cambio de los datos se realizará de manera más sencilla y confiable si éstos aparecen una vez y no en varios archivos. Cuando un usuario necesite un dato particular, la base de datos con un buen diseño, debería anticipar tal necesidad. En consecuencia, los datos tendrán mayor probabilidad de encontrarse disponibles en una base de datos que en un sistema de archivos convencionales. Una base de datos con un buen diseño también llega a ser más flexible que dos archivos separados, esto significa que una base de datos llega a evolucionar conforme se modifican las necesidades de los usuarios y de sus aplicaciones.

Finalmente, el enfoque de base de datos, tiene la ventaja de permitir que los usuarios expongan sus puntos de vista sobre los datos, sin necesidad de preocuparse de la estructura presente de la base de datos o de su ubicación física,
La primera desventaja del enfoque de base de dato es que todos los datos se almacenan en un solo lugar; y en consecuencia, los datos son más vulnerables a accidentes y requerirán de un respaldo completo. Esto en la actualidad ha sido mejorado con técnicas avanzadas de bases de datos (Bases de datos replicadas y distribuidas). Existe el riesgo de que quien administra la base de datos se convierta en el único privilegiado o habilitado para estar cerca de los datos y los procedimientos burocráticos requeridos para modificar o para actualizar la base de datos pueden llegar a ser insuperables. Otras desventajas para la administración de los datos como recurso se presentan al intentar satisfacer dos objetivos de eficiencia:

1) Reducción del tiempo requerido para insertar, actualizar, eliminar y recuperar los datos en tiempos tolerables.
2) Mantenimiento del costo del almacenamiento de datos en una cantidad razonable.

Es necesario recordar que una base de datos no puede optimizar la recuperación de los datos para una aplicación en especial, ya que deberá compartirse con numerosos usuarios y con varias aplicaciones. Además, se puede llegar a requerir de cierto software adicional para la DBMS y, ocasionalmente, también se necesitará una computadora de mayor capacidad. El enfoque de la base de datos es un concepto que se vuelve cada vez más relevante. El uso de base de datos relaciónales en computadoras personales indica el grado de difusión que este concepto ha alcanzado entre los usuarios. Con este enfoque, los usuarios toman una parte de la base de datos central y cargan sus computadoras personales. Luego estas pequeñas bases de datos se utilizan para emitir reportes o contestar consultas específicas del usuario final.

Resumen

Sistemas de Almacenamiento de Datos
- Archivos Convencionales
  - Inconvenientes
    - Falta de potencial para evolucionar
    - Redundancia e inconsistencia de datos
    - Dificultades de Acceso
    - Problemas de Concurrencia

- Aislamiento de los datos
- Seguridad
- Integridad
- Casos en que conviene
  - Aplicaciones ya existentes
  - Cuestiones de Performance

Bases de Datos

- Objetivos
  - Compartir información
    - Múltiples usuarios
    - Múltiples aplicaciones
  - Mantenimiento de datos preciso y consistente
  - Disponibilidad de los datos
  - Flexibilidad para evolucionar según las necesidades
  - Independencia del almacenamiento físico
- Inconvenientes
  - Almacenamiento centralizado (tradicionalmente)
  - Dependencia burocrática del DBA
  - Tiempo de actualización
  - No se optimiza el acceso para una aplicación especial

3. ORGANIZACIÓN DE SISTEMAS DE BASE DE DATOS

Una base de datos, a diferencia de un archivo, es compartida por muchos usuarios, y naturalmente cada usuario verá los datos de manera diferente (nos referimos a la forma en que un usuario concibe y describe los datos desde una presentación de usuario). Estas presentaciones se examinan en el modelo lógico global de la base de datos, que eventualmente deberá desarrollarse. Finalmente, el modelo lógico de la base de datos debe transformarse en el correspondiente diseño físico de la base de datos. El diseño físico considera la forma del almacenamiento de los datos y de sus interrelaciones, así como la mecánica del acceso.

Un sistema de gestión de base de datos (DBMS) consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a esos datos. La colección de datos, normalmente denominada base de datos, contiene información acerca de una empresa determinada. El objetivo primordial de un DBMS es proporcionar un entorno que sea a la vez conveniente y eficiente para ser utilizado al extraer y almacenar información de la base de datos.

Los sistemas de bases de datos están diseñados para gestionar grandes bloques de información. La gestión de datos implica tanto la definición de estructuras para el almacenamiento de información como la provisión de mecanismos para la gestión de la información. Además, los sistemas de bases de datos deben mantener la seguridad de la información almacenada, pese a caídas del sistema o intentos de accesos no autorizados. Si los datos van a ser compartidos por varios usuarios, el sistema debe evitar posibles resultados anómalos.

La importancia de la información en la mayoría de las organizaciones, y por tanto el valor de la base de datos, ha llevado al desarrollo de una gran cantidad de conceptos y técnicas para la gestión eficiente de los datos.

ABSTRACCIÓN DE DATOS

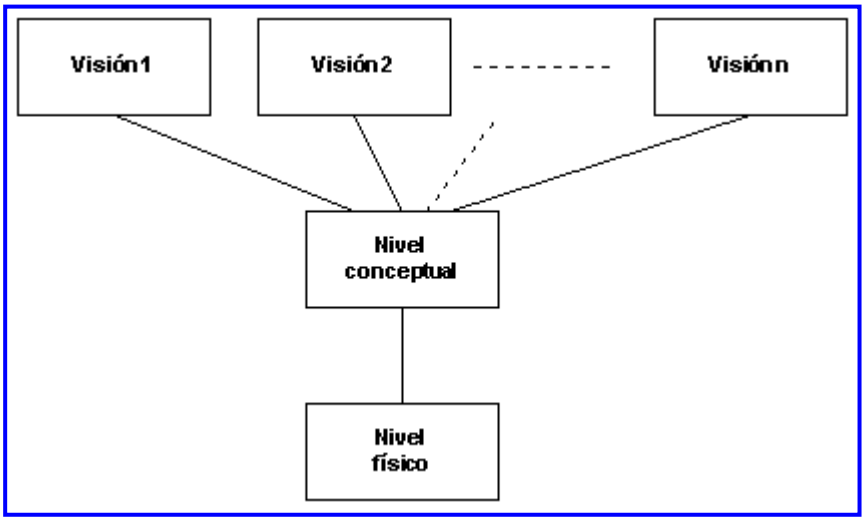
Un objetivo importante de un sistema de bases de datos es proporcionar a los usuarios una visión abstracta de los datos. Es decir, el sistema esconde ciertos detalles de cómo se almacenan y mantienen los datos. Sin embargo, para que el sistema sea manejable, los datos se deben extraer eficientemente. Este

requerimiento ha llevado al diseño de estructuras de datos complejas para la representación de datos en la base de datos. Puesto que muchos usuarios de sistemas de bases de datos no tienen experiencia en computadoras, se les esconde la complejidad a través de diversos niveles de abstracción para simplificar su interacción con el sistema.

Nivel Físico: El nivel más bajo de abstracción describe como se almacenan realmente los datos. En el nivel físico, se describen en detalle las estructuras de datos complejas de bajo nivel.

Nivel Conceptual: El siguiente nivel de abstracción describe qué datos son realmente almacenados en la base de datos y las relaciones que existen entre los datos. Aquí se describe la base de datos completa en términos de un número pequeño de estructuras relativamente sencillas las que pueden implicar estructuras complejas en el nivel físico. El nivel conceptual lo usan los administradores de base de datos, quienes deben decidir qué información se va a guardar en la base de datos.

Nivel de Visión: El nivel más alto de abstracción describe sólo partes de la base de datos completa. Muchos usuarios de la base de datos, no necesitan conocer toda la información. En cambio, dichos usuarios sólo necesitan conocer una parte de la base de datos. El sistema puede proporcionar múltiples visiones para la misma base de datos.



INDEPENDENCIA DE DATOS

Anteriormente definimos tres niveles de abstracción en los que puede verse la base de datos. La capacidad de modificar una definición de un esquema en un nivel sin afectar la definición de un esquema en el nivel superior siguiente se llama independencia de datos. Hay dos niveles de independencia de datos:

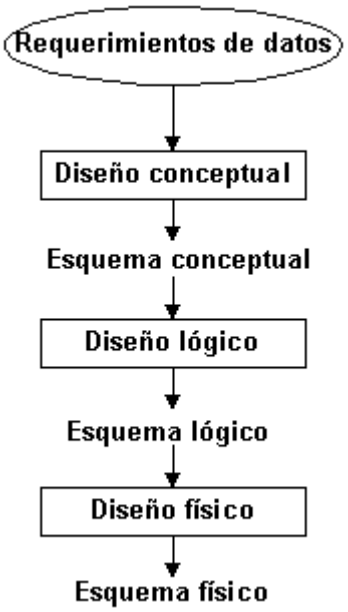
- **Independencia física de datos**: es la capacidad de modificar el esquema físico sin provocar que se vuelvan a escribir los programas de aplicación. En algunas ocasiones son necesarias las modificaciones en el nivel físico para mejorar el funcionamiento.
- **Independencia lógica de datos**: es la capacidad de modificar el esquema conceptual sin provocar que se vuelvan a escribir los programas de aplicación. Las modificaciones en el nivel conceptual son necesarias siempre que se altera la estructura lógica de la base de datos.

La independencia lógica de datos es más difícil de lograr que la independencia física de datos, ya que los programas de aplicación son fuertemente dependientes de la estructura lógica de los datos a los que acceden.

El concepto de independencia de datos es similar en muchos aspectos al concepto de tipos abstractos de datos en los lenguajes de programación modernos. Ambos ocultan detalles de implementación a los usuarios. Esto permite a los usuarios que se concentren en la estructura general en vez de los detalles de implementación de bajo nivel.

4. ENFOQUE ORIENTADO A LOS DATOS PARA EL DISEÑO DE SISTEMAS

Con este enfoque, primero se diseña la base de datos, luego las aplicaciones que las usan. Este método se desarrolló en la década de 1970, con el establecimiento de la tecnología de bases de datos. El gráfico siguiente muestra las etapas previstas en un Enfoque Orientado a los Datos:



Los objetivos de cada etapa son las siguientes:

Diseño conceptual. El propósito es describir el contenido de la información de la base de datos, mas que las estructuras de almacenamiento que se necesitan para manejar esta información. Esta fase parte de la especificación de requerimientos y su resultado es un esquema conceptual. El esquema conceptual es una descripción de alto nivel de la estructura de la base de datos, independiente del software del SMBD que se use para manipularla. Es importante recalcar que los esquemas conceptuales se describen usando el modelo conceptual.

Diseño lógico. Tiene como fin obtener el esquema lógico, que es una descripción de la estructura de la base de datos que puede procesar el software del SMBD. Para especificar el esquema lógico se usa el modelo lógico (jerárquico, de redes, o relacional, que es actualmente el mas usado). El diseño lógico depende del modelo de datos usado por el SMBD y no del SMBD utilizado (el diseño lógico se realiza de la misma forma para todos los SMBD relacionales porque todos utilizan el modelo relacional).

Diseño físico. El objetivo es obtener el esquema físico, el cual es una descripción de la implantación de una base de datos en la memoria secundaria, describe las estructuras de almacenamiento y los métodos usados para tener acceso efectivo a los datos. Hay una retroalimentación entre el diseño físico y lógico, porque las decisiones tomadas durante el diseño físico para mejorar el rendimiento, pueden afectar la estructura del esquema lógico.

Una vez completo el diseño físico de la base de datos, la base de datos se crea y se carga, y puede ser probada. Las aplicaciones que usan las bases de datos pueden especificarse, implantarse y probarse completamente. De este modo, la base de datos se vuelve paulatinamente operacional.

5. CONCEPTOS SOBRE EL MODELADO DE DATOS

MODELOS DE DATOS.

Un modelo de datos es una serie de conceptos que pueden utilizarse para describir un conjunto de datos y operaciones para manipular los mismos.
Los modelos de datos se describen, por lo regular, mediante representaciones lingüísticas y gráficas; es decir, puede definirse una sintaxis y puede desarrollarse una notación gráfica, como partes de un modelo de datos (Batini).

Existen dos tipos de modelos de datos:

- **Modelos Conceptuales**.
  Son instrumentos para representar la realidad a un alto nivel de abstracción. Con su uso se puede construir una descripción de la realidad, fácil de entender e interpretar. Se caracterizan por el hecho de que proporcionan capacidad de estructuración bastante flexible y permiten especificar restricciones de datos explícitamente. Hay modelos diferentes, y es probable que aparezcan más. Algunos de los más extensamente conocidos son:
  - El modelo entidad-relación.
  - El modelo orientado a objetos.
  - El modelo binario.
  - El modelo semántico de datos.
  - El modelo infológico.
  - El modelo funcional de datos.

- **Modelos Lógicos.**
  Apoyados por los sistemas de manejo de base de datos (SMBD). Describen los datos procesables en un computador. Los tres modelos de datos más ampliamente aceptados son los modelos relacional, de red y jerárquico.
  El modelo relacional, que ha ganado aceptación por encima de los otros dos en años recientes. Los modelos de red y jerárquico, son todavía usados en un gran número de bases de datos más antiguas, pero están cayendo rápidamente en desuso. Debajo presentamos una breve visión de cada modelo.

  Modelo relacional

  El modelo relacional representa los datos y las relaciones entre los datos mediante una colección de tablas, cada una de las cuales tiene un número de columnas con nombres únicos.

  Modelo de red

  Los datos en el modelo de red se representan mediante colecciones de registros (en el sentido de la palabra en Pascal o PL/1) y las relaciones entre los datos se representan mediante enlaces, los cuales pueden verse como punteros. Los registros en la base de datos se organizan como colecciones de grafos arbitrarios.

  Modelo jerárquico

  El modelo jerárquico es similar al modelo de red en el sentido de que los datos y las relaciones entre los datos se representan mediante registros y enlaces, respectivamente. Se diferencia del modelo de red en que los registros están organizados como colecciones de árboles en vez de grafos arbitrarios.

Diferencias entre los modelos

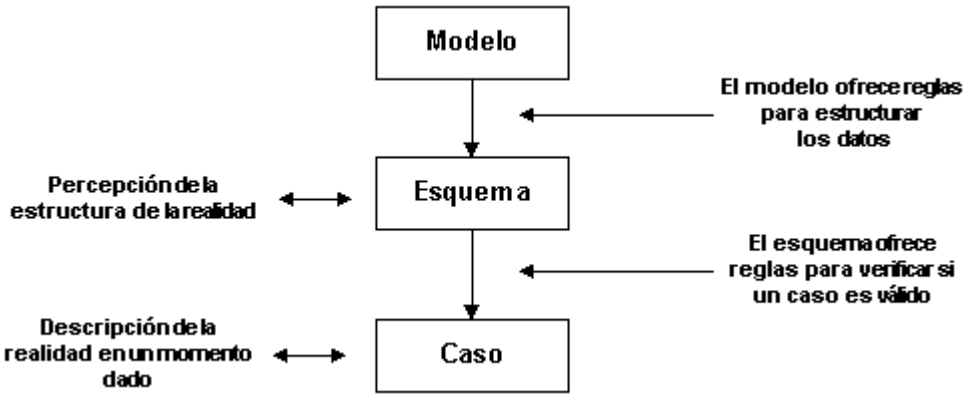
Los modelos relaciónales se diferencian de los modelos de red y jerárquico en que no usan punteros o enlaces. En cambio, el modelo relacional conecta registros mediante los valores que éstos contienen. Esta libertad del uso de punteros permite que se defina una base matemática formal.

Estos modelos tienen una correspondencia sencilla con la estructura física de las bases de datos.

Un **Esquema** es una representación de una parte específica de la realidad, creada usando un determinado modelo de datos. Es un conjunto estático de representaciones lingüísticas o gráficas, que describen la estructura de datos de interés, como por ejemplo los de una organización. La calidad de los esquemas resultantes depende no solo de la habilidad del diseñador, sino también de las características del modelo de datos seleccionado.

Un **Caso de un esquema** es una colección de datos dinámica, variable en el tiempo, que se ajusta a la estructura de datos que define el esquema. Cada esquema puede tener múltiples casos. El estado de la base de datos en un momento determinado, corresponde a uno de esos casos. La evolución de la base de daros puede verse como la transición de un caso a otro, originada por alguna operación de modificación de datos.

El diagrama siguiente muestra la correspondencia entre modelo, esquema y caso:



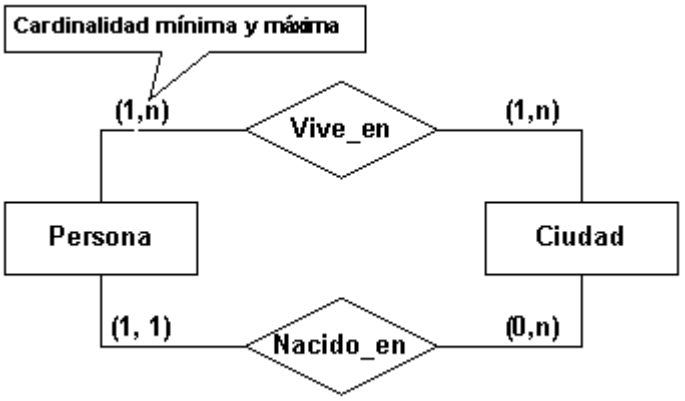
Una forma de ver la diferencia entre esquema y caso es considerar que el primero denota propiedades estructurales de los datos, y el segundo denota una asignación de valores a los datos.

SEMANA 02: MODELO ENTIDAD-RELACIÓN

El modelo de datos entidad-relación (E-R) se basa en la percepción de un mundo real que consiste en un conjunto de objetos básicos llamados entidades y relaciones. Se desarrolló para facilitar el diseño de bases de datos permitiendo la especificación de un esquema empresarial. Este esquema representa la estructura lógica global de la base de datos. Es el modelo de datos más ampliamente usado para el diseño conceptual de bases de datos. El modelo entidad-relación fue propuesto inicialmente por Peter Chen en 1976 y ha sido estudiado por varios autores. El MER se ve principalmente como una herramienta de diseño.
Originalmente, el modelo ER incluía solo los conceptos de entidad, relación y atributos; más tarde, otros conceptos, como los atributos compuestos y las jerarquías de generalización, se agregaron como componentes del modelo ER mejorado.

ELEMENTOS BÁSICOS DEL MODELO ER

Los conceptos básicos previstos por el modelo ER son entidades, relaciones y atributos. El siguiente gráfico muestra una parte de un esquema ER, que representa las entidades PERSONA y CIUDAD y las relaciones NACIDO_EN y VIVE_EN:



A continuación se introducen los elementos del modelo ER:

ENTIDADES

Las entidades representan clases de objetos. Una entidad es un objeto que es distinguible de otros objetos por medio de un conjunto específico de atributos. Una entidad puede ser concreta, tal como una persona o un libro, o puede ser abstracta, como un día festivo o un concepto. PERSONA, EMPLEADO y CIUDAD, son ejemplos de entidades para una base de datos de Personal. Las entidades se representan gráficamente por medio de rectángulos.

RELACIONES

Las relaciones representan agregaciones de dos o más entidades. Un ejemplo de la relación binaria en el esquema mostrado arriba es NACIDO_EN, que relaciona PERSONA y CIUDAD de nacimiento. Mas adelante al analizar la cardinalidad, veremos que es posible obtener varios tipos de relaciones, como son: las relaciones n-arias (una relación entre más de 2 entidades) y las relaciones recursivas (una relación de una entidad consigo misma).

ATRIBUTOS

Los atributos representan las propiedades básicas de las entidades o relaciones. Toda información extensiva es portada por los atributos. Para cada atributo hay un conjunto de valores permitidos llamados dominio de ese atributo. Formalmente, un atributo es una función que asigna un conjunto de entidades a un dominio. Así,

cada entidad se describe por medio de un conjunto de pares (atributo, valor del dato), un par para cada atributo del conjunto de entidades. Por ejemplo, los atributos de PERSONA podrían ser: Nombre, Apellido, DNI y Profesión.

ATRIBUTO O ENTIDAD

Existen situaciones en las que un atributo puede modelarse como una entidad distinta. Considérese el conjunto de entidades empleado con atributos nombre-empleado y número-teléfono. Se puede argumentar fácilmente que un teléfono es una entidad en sí misma con atributos número-teléfono y ubicación (la oficina donde está colocado el teléfono). Si tomamos este punto de vista, el conjunto de entidades empleado debe definirse como sigue:

- El conjunto de entidades empleado con atributo nombre-empleado.
- El conjunto de entidades teléfono con atributo número-teléfono y situación.
- El conjunto de relaciones EmpITelef, que indica la asociación entre los empleados y los teléfonos que tienen.

Surge entonces una pregunta natural: ¿Qué constituye un atributo, y que constituye una entidad? Desafortunadamente, no hay una respuesta sencilla. La distinción depende principalmente de la estructura del dominio de problema que se está modelando y de la semántica asociada con el atributo en cuestión.

CARDÍNALIDAD.

Además de entidades y relaciones, el modelo E-R representa ciertas restricciones a las que deben ajustarse los contenidos de una base de datos. Una restricción importante es la de cardinalidad.

Se distingue entre cardinalidad mínima y cardinalidad máxima.

La **cardinalidad máxima** es utilizada para especificar cuantas veces una entidad está asociada, como máximo, a través de un conjunto de relaciones. Se consideran dos cardinalidades máximas:

- **1:** indica que una entidad está asociada como máximo a una entidad a través de la relación.
- **n:** indica que una entidad puede estar asociada a muchas entidades a través de la relación.

La cardinalidad mínima es utilizada para especificar cuantas veces una entidad está asociada, como mínimo, a través de un conjunto de relaciones. Se consideran dos cardinalidades mínimas:

- **0:** indica que una entidad no está obligatoriamente asociada a través de la relación (participación opcional).
- **1:** indica que una entidad debe estar asociada al menos una vez a través de la relación (participación obligatoria).

Así, en el gráfico anterior:

- La cardinalidad de la entidad PERSONA en la relación VIVE_EN es (1, n), que quiere decir que una persona (cualquiera de ellas) vive en, como mínimo 1 ciudad y como máximo en n (muchas) ciudades.
- La cardinalidad de la entidad CIUDAD en la relación VIVE_EN es (1, n), significa que en una ciudad (cualquiera de ellas) viven, como mínimo 1 persona y como máximo n (muchas) personas.
- La cardinalidad de la entidad PERSONA en la relación NACIDO_EN es (1, 1), que quiere decir que una persona nació en, como mínimo 1 ciudad y como máximo en 1 ciudad (es decir en una sola ciudad).
- La cardinalidad de la entidad CIUDAD en la relación NACIDO_EN es (0, n), que quiere decir que en una ciudad nacieron, como mínimo 0 personas y como máximo n (muchas) personas.

En realidad podemos analizar dos tipos distintos de cardinalidad de asignación: cardinalidad en atributos y cardinalidad en relaciones.

Cardinalidad en Atributos

En general puede considerarse que cada entidad de un conjunto, tiene asociada exactamente un valor de cada atributo. Sin embargo esto no es obligatorio.
Pueden especificarse cantidad mínima y máxima de valores de un atributo asociados a una entidad.
La *cardinalidad mínima* indica el número mínimo de valores para el atributo:

- Si la card-min = 0 indica que el atributo es opcional y puede no estar especificado en algunos casos.
- Si la card-min = 1 indica que el atributo es obligatorio y al menos un valor debe especificarse para cada instancia de la entidad.

Ej.: card-min (PERSONA, Nombre) = 1, card-min (PERSONA, Teléfono) = 0
La *cardinalidad máxima* indica el número máximo de valores para el atributo. Si la card-max = 1 indica que el atributo es monovalente. Si la card-max > 1 indica que el atributo es polivalente.
Ej.: card-max (PERSONA, DNÍ) = 1, card-max (PERSONA, Teléfono) = n

Podemos expresar la cardinalidad de atributos como el par (card-min, card-max).
Ej.: card (PERSONA, Nombre) = (U), card (PERSONA, Teléfono) = (0, n)

Como disciplina de modelado, es aconsejable iniciar el proceso restringiéndose a atributos con exactamente un valor (1,1) por los siguientes motivos:

- Los atributos multi-valorados (card-max n) son difíciles de implementar en SGBD relacionales, obligando a procesos de normalización.
- Los atributos opcionales (card-min 0) generalmente indican clases especiales de entidades.

Cardinalidad en Relaciones

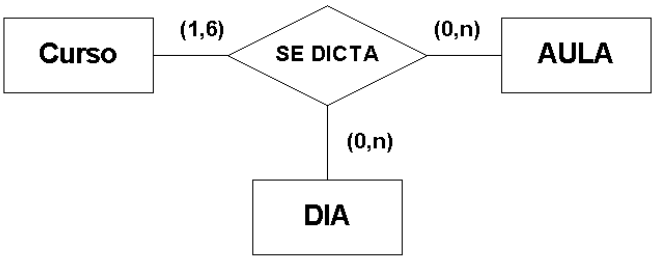
Expresan el número de entidades con las que puede asociarse otra entidad mediante un conjunto de relaciones. En esta sección nos concentraremos sólo en conjuntos binarios de relaciones. Más adelante trataremos con conjuntos de relaciones n-arios (n > 2).
Para un conjunto binario de relaciones R entre los conjuntos de entidades A y B, la cardinalidad de asignación debe ser una de las siguientes:

- **Una a Una (1,1):** Una entidad en A está asociada a lo sumo con una entidad de B, y una entidad en B está asociada a lo sumo con una entidad en A.

- **Una a Muchas (1, n):** Una entidad en A está asociada con un número cualquiera de entidades en B. Una entidad en B, sin embargo puede estar asociada a lo sumo con una entidad en A.
- **Muchas a Una (n, 1);** Una entidad en A está asociada a lo sumo con una entidad en B. Una entidad en B, sin embargo, puede estar asociada con un número cualquiera de entidades en A.
- **Muchas a muchas (n, n):** Una entidad en A está asociada con un número cualquiera de entidades en B, y una entidad en B está asociada con un número cualquiera de entidades en A.

La cardinalidad adecuada para un conjunto de relaciones determinado obviamente es dependiente del dominio de problema que el conjunto de relaciones está modelando.

Las relaciones n-arias conectan tres o más entidades.

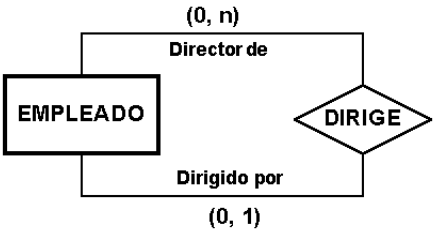


El gráfico muestra una parte de un esquema ER, que representa las entidades CURSO, AULA y DÍA y la relación ternaria SE DICTA.

La cardinalidad mínima y máxima en este caso se interpreta así:

- La cardinalidad de la entidad CURSO en la relación SE DICTA es (1, 6), que quiere decir que un curso debe tener de 1 a 6 sesiones de clase.
- La cardinalidad de la entidad AULA en la relación SE DICTA es (0, n), significa que en un aula, se dictan como mínimo 0 sesiones y como máximo n sesiones.
- La cardinalidad de la entidad DÍA en la relación SE DICTA es (0, n), significa que en un día, se dictan como mínimo 0 sesiones y como máximo n sesiones.

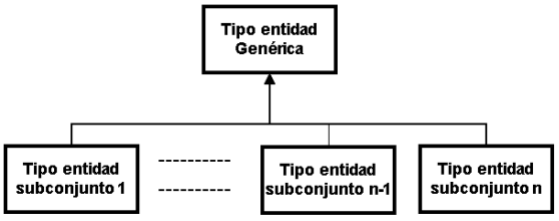
Las relaciones que conectan a una entidad consigo misma, se conocen como **relaciones recursivas**.



El gráfico muestra la relación recursiva DIRIGE.
En este caso, es necesario distinguir los papeles que la entidad EMPLEADO cumple en la relación. En la parte superior cumple el rol de Director (Un empleado es director de 0 o más empleados). En la línea inferior cumple el rol de subordinado (Un empleado es subordinado de 0 o 1 empleado).

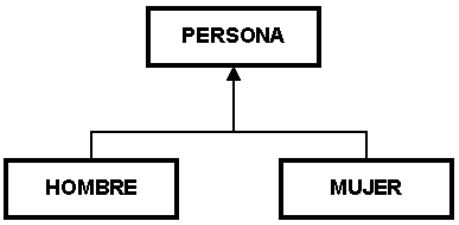
JERARQUÍAS DE GENERALIZACIÓN

Una entidad E es una generalización de un grupo de entidades El, E2,..., En (entidades subconjunto), si cada objeto de la clase El, E2,..., En, es también un objeto de la clase E (entidad genérica).
Lo opuesto a la generalización es la especialización.
Todas las propiedades (atributos, relaciones o generalizaciones) de la entidad genérica serán heredadas por las entidades subconjunto.



Las entidades subconjunto se justifican porque, o tienen atributos exclusivos respecto a las otras entidades subconjuntos, o tienen relaciones exclusivas con otras entidades.

En el gráfico adjunto, la entidad PERSONA es una generalización de las entidades HOMBRE y MUJER.



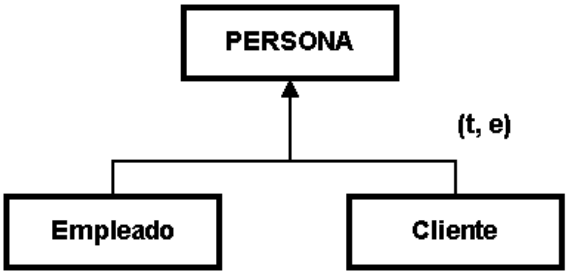
Cobertura de las Generalizaciones

Las jerarquías de generalización presentan la propiedad de cobertura. La cobertura puede ser parcial o total y exclusiva o superpuesta.

La cobertura parcial o total permite especificar una restricción entre la entidad genérica y sus entidades subconjunto; Si todos los elementos de la entidad genérica pertenecen a alguna de sus entidades subconjunto la cobertura es total, sino, cuando hay elementos de la entidad genérica que no pertenecen a ninguna entidad subconjunto, la cobertura es parcial. La cobertura exclusiva o superpuesta permite especificar una restricción entre las entidades subconjunto; Si los elementos que pertenecen a una entidad subconjunto pueden pertenecer también a otra entidad subconjunto la cobertura es superpuesta, sino es una cobertura exclusiva.

Ejemplo: Consideremos el caso de un banco cualquiera y diferentes políticas respecto a las personas a considerar, en su calidad de empleados y clientes. Se verá como es afectada la cobertura según el caso.

- ***Cobertura total y exclusiva:*** Todas las personas están clasificadas como empleados o clientes (cobertura total); Sí una persona se clasifica como empleado, no puede clasificarse como cliente y al contrario ocurre lo mismo (cobertura exclusiva).



- ***Cobertura total y superpuesta:*** Todas las personas son empleados o clientes del banco (cobertura total), permitiéndose que un empleado sea a su vez cliente (cobertura superpuesta).
  El esquema es similar al anterior solo que en vez de poner la indicación (t, e) se pondrá (t, s).
- ***Cobertura parcial y exclusiva:*** No todas las personas están clasificadas como empleados o clientes (cobertura parcial); Si una persona se clasifica como empleado, no puede clasificarse como cliente y al contrario ocurre lo mismo (cobertura exclusiva). El esquema sigue siendo el mismo, pero con la indicación (p, e).
- ***Cobertura parcial y superpuesta:*** No todas las personas están clasificadas como empleados o clientes (cobertura parcial); Si una persona se clasifica como empleado también puede clasificarse como cliente (cobertura superpuesta). El esquema llevaría la indicación (p, s).

IDENTIFICADORES

Un identificador de una entidad E es un grupo de atributos o de entidades relacionadas con E, que tienen la propiedad de determinar en forma única todos los casos de E. Los identificadores se llaman también claves primarias o claves candidatas. Los identificadores se clasifican básicamente en simples y compuestos, según que consistan de un solo atributo o cuando se componen de más de dos atributos, respectivamente.

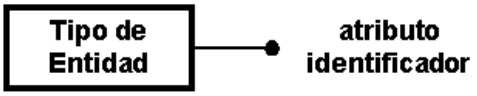
PERSONA
DNI
Nombre
Apellido
Profesión

En el gráfico adjunto se muestra la entidad PERSONA, con sus tres atributos: DNI, Nombre y Profesión. El atributo DNI es el identificador o clave primaria. No es recomendable considerar el identificador Profesión compuesto (Nombre, Apellido), ya que se puede dar el caso de homonimia, violando la unicidad del identificador.
También es común representar a los atributos fuera del rectángulo de la entidad, mediante líneas conectadas al rectángulo y al extremo un pequeño círculo con el nombre del atributo.

Se puede formalizar la definición de identificador así:

Un atributo I, posiblemente compuesto, de una entidad E, es un identificador de E, si y sólo si satisface las siguientes 2 propiedades independientes del tiempo.

a. Unicidad. En cualquier momento dado, no existen dos elementos en E con el mismo valor de I.
b. Minimalidad. Si I es compuesto, no será posible eliminar ningún atributo componente de I sin destruir la propiedad de unicidad.

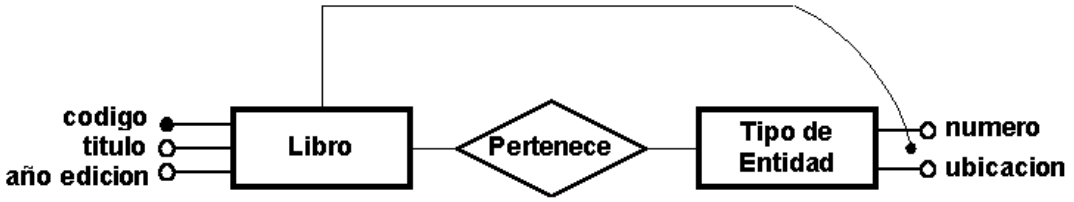


Ejemplo: en Perú, para una entidad Persona, el identificador puede ser DNI.

Clasificación de los tipos de entidad según sus identificadores

- Entidad Fuerte: Tipo de entidad con identificador interno. Por ejemplo, más adelante se muestra la entidad LIBRO.
- Entidad Débil: Tipo de entidad con identificador externo o mixto. Como ejemplo, se tiene mas adelante la entidad EJEMPLAR.

Ejemplo: En una biblioteca, se tiene para cada libro uno o más ejemplares. Cada LIBRO tiene un código único, mientras que para diferenciar a un ejemplar de otro, se le agrega un correlativo. Para esta situación, se puede tener un esquema como el siguiente, donde el tipo de entidad LIBRO tiene como atributos código, título y año edición, mientras que EJEMPLAR tiene como atributos número de ejemplar y ubicación.



El identificador de LIBRO es el atributo Código, mientras que el identificador de la entidad débil EJEMPLAR es la composición del atributo externo Código y el atributo Número.
Así, se tienen las entidades:
Libro: (Código. Título, Año edición)

Ejemplar: (Código (FK). Número. Ubicación)

Donde FK indica que Código es un atributo externo, en este caso de la entidad fuerte LIBRO. El identificador de Ejemplar se dice que es mixto, porque tiene un atributo interno y otro externo.

CLAVES DE UNA RELACIÓN (DE UNA TABLA).

En el modelo relacional se habla de claves en vez de identificadores, existiendo comúnmente las siguientes definiciones:

- **Clave candidata** de una relación (tabla) es un conjunto no vacío de atributos que identifican unívoca y mínimamente cada tupla (registro).
- **Clave primaria:** Es aquella clave candidata que el usuario elegirá, por consideraciones ajenas al modelo relacional, para identificar las tuplas de la relación.
- **Clave alternativas:** Son aquellas claves candidatas que no han sido escogidas como claves primarias.
- **Clave ajena o foránea:** de una relación R2 es un conjunto no vacío de atributos cuyos valores han de coincidir con los valores de la clave primaria de una relación R1 (R1 y R2 no son necesariamente distintas). Notar que la clave ajena y la correspondiente clave primaria han de estar definidas sobre los mismos dominios.

Practica de Laboratorio

Ejercicio

De una compañía consultora de ingeniería que analiza la construcción y estado de casas y otros edificios e instalaciones, se tiene la información siguiente:

- ✓ Algunos empleados son ingenieros.
- ✓ Cada autobús debe estar asignado a un ingeniero, pero no todos los ingenieros tienen autobús.
- ✓ Los ingenieros proporcionan servicios a los clientes.
- ✓ Un ingeniero puede no proporcionar servicio a algún cliente, o de igual manera atender a varios clientes.
- ✓ Un servicio debe ser proporcionado por determinado ingeniero, y ese servicio únicamente lo puede realizar ese ingeniero en particular.
- ✓ Los clientes tiene muchos servicios, y un servicio puede ser solicitado por muchos clientes.
- ✓ Un cliente debe haber comprado por lo menos un servicio, pero un servicio no necesariamente tiene que haber tenido clientes.
- ✓ Debemos conservar la tarifa que se le cobro a un cliente por determinado servicio, es decir la cantidad que un cliente en particular paga por un servicio determinado.
- ✓ Algunas veces unos clientes envían a otros, esto significa que recomiendan el servicio.
- ✓ Un cliente puede recomendar a uno o a varios clientes más. Un cliente puede o no haber sido recomendado por otro cliente, pero un cliente puede ser recomendado sólo por otro cliente.
- ✓ Un ingeniero se capacita constantemente, y cuando concluye cada curso y ha obtenido la aprobación necesaria le extienden un certificado. Un ingeniero puede obtener varias certificaciones.
- ✓ Un ingeniero para ejercer la profesión debe estar habilitado por el colegio profesional respectivo.

Elabore el diagrama entidad relación respectivo.

SEMANA 03: CASOS DE MODELO ENTIDAD-RELACIÓN

Problemas Propuestos Diseño Conceptual

1. Los requerimientos que se muestra a continuación describe la información de un gabinete de ingenieros que realizan proyectos de instalaciones eléctricas industriales. Las empresas que desean los servicios del gabinete contactan con el departamento de atención al cliente, que abre una ficha de proyecto, asignándole un número que lo identificará en adelante. En esta ficha se registran los datos de la empresa y se deposita en la bandeja de nuevos proyectos del ingeniero jefe. Todas las mañanas, el ingeniero jefe revisa los nuevos proyectos, asignando a cada uno el ingeniero que considera adecuado, al tiempo que se lo comunica a éste personalmente y lo anota en la ficha. El ingeniero asignado visita la empresa y, en función de las necesidades del cliente, elabora un presupuesto que adjunta a la ficha del proyecto. En este presupuesto figuran las descripciones de las tareas a realizar, el presupuesto para cada tarea y el importe total. Cada tarea tiene fijado un importe base que es siempre el mismo, independientemente del proyecto. Cuando el presupuesto se envía a la empresa, ésta puede aceptarlo o no, por lo que habrá proyectos aceptados y no aceptados. Cuando un proyecto es aceptado, el ingeniero jefe decide la fecha de inicio y le asigna los operarios necesarios de cada especialidad, comprobando que no estén ocupados en otro proyecto. Toda esta información también se registra en la ficha del proyecto. Periódicamente, para los proyectos de larga duración, el ingeniero asignado debe informar del grado de ejecución del proyecto. Una vez finalizados los trabajos de un proyecto, el ingeniero asignado lo comunica al ingeniero jefe que procede a anotarlo en la ficha del proyecto y la envía al departamento de contabilidad para que proceda a gestionar el cobro. Diseñe la BD correspondiente empleando la estrategia Descendente

2. La empresa ABC ensambla diversos modelos de productos eléctricos. Se requiere diseñar la base de datos que registre la composición de cada modelo en términos de partes, componentes y piezas, además de los posibles proveedores (en caso de tenerlos) y los precios ofrecidos por estos proveedores. Cada modelo está compuesto de partes (a su vez cada parte puede estar en varios modelos), y cada parte está dividida en componentes. Los componentes están constituidos por piezas ó por otros componentes. ABC ensambla todos sus modelos y partes, y la mayoría de sus componentes; Sin embargo, todas las piezas y algunos componentes (los que no ensambla) pueden ser comprados a varios proveedores, siendo necesario registrar los precios ofrecidos por cada proveedor.
   Realizar el esquema entidad-relación, indicando detalladamente atributos, claves primarias y candidatas.

3. Considerando lo siguiente:
   En un instituto de investigación se tienen proyectos y computadoras disponibles para los usuarios (posibles participantes en los proyectos).
   Hay 2 tipos de usuarios: investigadores y aprendices. Cada proyecto es coordinado por un investigador y puede tener como miembros a varios usuarios. Las computadoras tienen horarios (hay una entidad Horario con atributos fecha y hora de inicio, y es una entidad débil con respecto a Computadora). Los investigadores registran las infracciones de los aprendices en una entidad denominada Infracción (entidad débil

respecto a Aprendiz). Los usuarios pueden participar en varios proyectos y deben registrar las reservas de horarios (de computadoras) por su trabajo en un determinado proyecto. Los aprendices forman grupos que son asesorados por un investigador (independientemente de los proyectos). Dibujar el esquema entidad-relación.

4. Se requiere una base de datos que contenga el origen de los alumnos de la Universidad. Se debe registrar: datos personales, colegios donde estudió, tipo de colegio, dirección del colegio, teléfonos, director actual, nota en los 3 últimos años de secundaria y puntaje de ingreso a la universidad. Dibujar el esquema entidad-relación.

5. En una Biblioteca se tienen en forma simplificada lo siguiente: cada libro tiene varios temas, y varios libros pueden abarcar el mismo tema. Hay dos tipos de usuarios: internos y externos. Los usuarios internos pueden llevar los libros, prestados a domicilio o a la sala de lectura. Los usuarios externos no pueden llevar libros a domicilio. Con el fin de tener estadísticas para efectuar nuevas compras, se registran los préstamos de cada libro e inclusive los pedidos no atendidos (esto es, un libro pedido que no estaba disponible). Dibujar el esquema entidad-relación.

6. Obtener el esquema conceptual para el siguiente caso:
   Un club de esparcimiento tiene diversos socios. Se debe controlar que cada socio pague sus cuotas mensuales de afiliación. Además, cuando los socios visitan el Club, se le brindan servicios que son registrados y, cuyos costos, son luego cargados a las cuentas de cada socio consumidor. Estos gastos de los socios son pagados por partes, por lo que se debe controlar también los pagos parciales.
   Para determinar el gasto de cada socio por los servicios recibidos en cada visita, se tiene una tarifa definida por cada servicio. Dibujar el esquema entidad-relación.

7. En un Centro Comercial se requiere tener estadísticas históricas de precios de productos (tanto precio de compra como de venta).
   Cada producto tiene distintos proveedores, sin embargo, los precios ofrecidos por cada uno de ellos han ido variando en el tiempo. Es necesario registrar para cada producto, corno han cambiado en el tiempo los precios de cada proveedor, señalando la fecha de vigencia de dichos precios. Asimismo, se requieren las estadísticas de variación de los precios de venta al público a través del tiempo. Dibujar el esquema entidad-relación.

8. Represente en un esquema E-R de unas Elecciones similares a las últimas pasadas. En este caso se registra el voto de cada elector (por quien votó cada elector), considerarlo así, a pesar de que en Perú no se sabe por quien votó cada uno de ellos. El elector solo puede votar por un presidente, y hasta por dos congresistas, pero los congresistas deben ser del mismo grupo, aunque pueden ser diferentes de grupo diferente al del Presidente.

9. Diseñar un esquema conceptual para el caso de unos programadores que usan determinadas computadoras en determinados proyectos. Un mismo programador puede estar en varios proyectos a la vez. Las computadoras se pueden compartir para varios proyectos. Dibujar el esquema entidad-relación.

10. El club de Gisela, imparte clases de baile de salón y ofrece lecciones privadas y en grupo; cobra S/. 20.00 la hora por estudiante (o pareja) cuando se trata de una lección

privada, y S/.5.00 la hora por estudiante en el caso de una lección en grupo. Las lecciones privadas se imparten a cualquier hora, desde el medio día hasta las 22:00 horas, seis días a la semana. Las lecciones en grupo se imparten sólo en las tardes.

El club de bailes emplea dos tipos de instructores: asalariados de tiempo completo, e instructores de medio tiempo. La remuneración de los instructores de tiempo completo consiste en una cantidad fija por semana y a los de tiempo parcial se les paga cierta cantidad por una tarde, o por impartir sólo una clase.

Además de las lecciones, el club patrocina a la semana dos presentaciones de baile social con música grabada. La cuota de admisión es de S/.3.00 por persona. El baile de los viernes por la noche es el más popular y en promedio se reúnen 80 personas; el baile del domingo por la noche atrae aproximadamente a 30. El propósito de los bailes es proporcionar a los estudiantes un lugar para que practiquen sus habilidades. No se sirven alimentos ni bebidas.

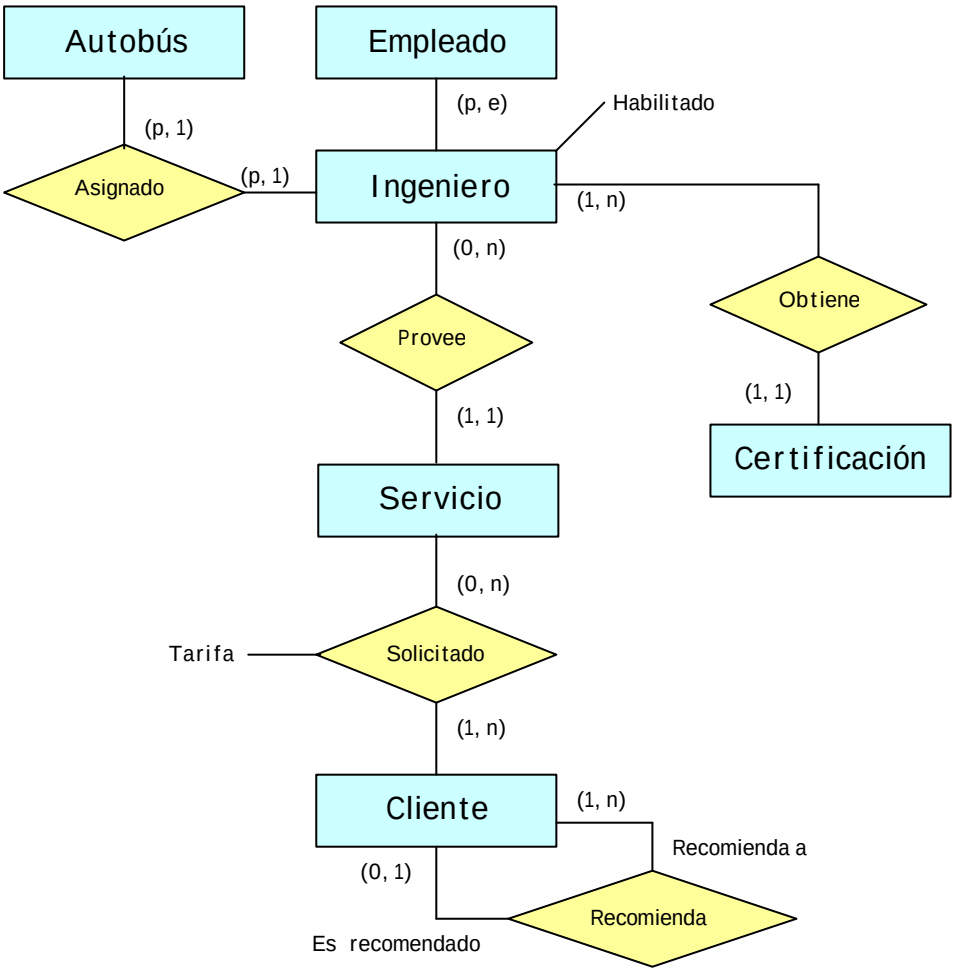
Al club de Gisela le gustaría desarrollar un sistema de información para dar seguimiento a los estudiantes y a las clases que éstos han tomado. Al administrador también le gustaría saber cuántas lecciones y de que tipo ha impartido cada maestro, y poder calcular el costo promedio por lección de cada uno de sus instructores.

11. Ingrese al sitio Web de un vendedor de libros, por ejemplo *Amazon* (www.amazon.com). Use el sitio Web para determinar los tres mejores libros de XML (Extended Markup Language). Cuando visite el sitio Web piense en la estructura de una posible base de datos de libros, autores, temas y temas afines.

    Desarrolle un diagrama entidad-relación de una base de datos de libros para este sitio Web. Muestre todas las entidades y relaciones y cuando menos dos o tres atributos por entidad. Indique las cardinalidades mínimas y máximas para ambos lados de cada relación. Las posibles entidades son: TITULO, AUTOR, EDITORIAL, COPIA Y TEMA. Por supuesto, hay muchas más entidades posibles. Modele cualquier atributo multivalor. Use subtipos donde sea apropiado. Para evitar que el diagrama se expanda demasiado, suponga que solo se dará seguimiento a los libros. Además, limite su diseño a las necesidades de alguien que esta buscando libros que quiere comprar. No considere pedido del cliente, entrega del pedido, orden de compra y otros procesos de negocios.

Practica de Laboratorio

Se tiene el siguiente diagrama E-R:



Modelo conceptual de datos

En el diseño de una base de datos, el proceso de diseño comienza normalmente en el plano conceptual, donde no es necesario estudiar los detalles del desarrollo físico real.

Un CMD representa la estructura lógica de una base de datos, que es independiente de cualquier software o estructura de almacenamiento de datos. Un modelo conceptual a menudo contiene objetos de datos que aún no se han aplicado en la base de datos física. Ofrece una representación formal de los datos necesarios para gestionar una empresa o una actividad empresarial.

Modelo lógico

El modelo lógico le permite diseñar la estructura de base de datos y realizar algunas acciones de normalización de base de datos.

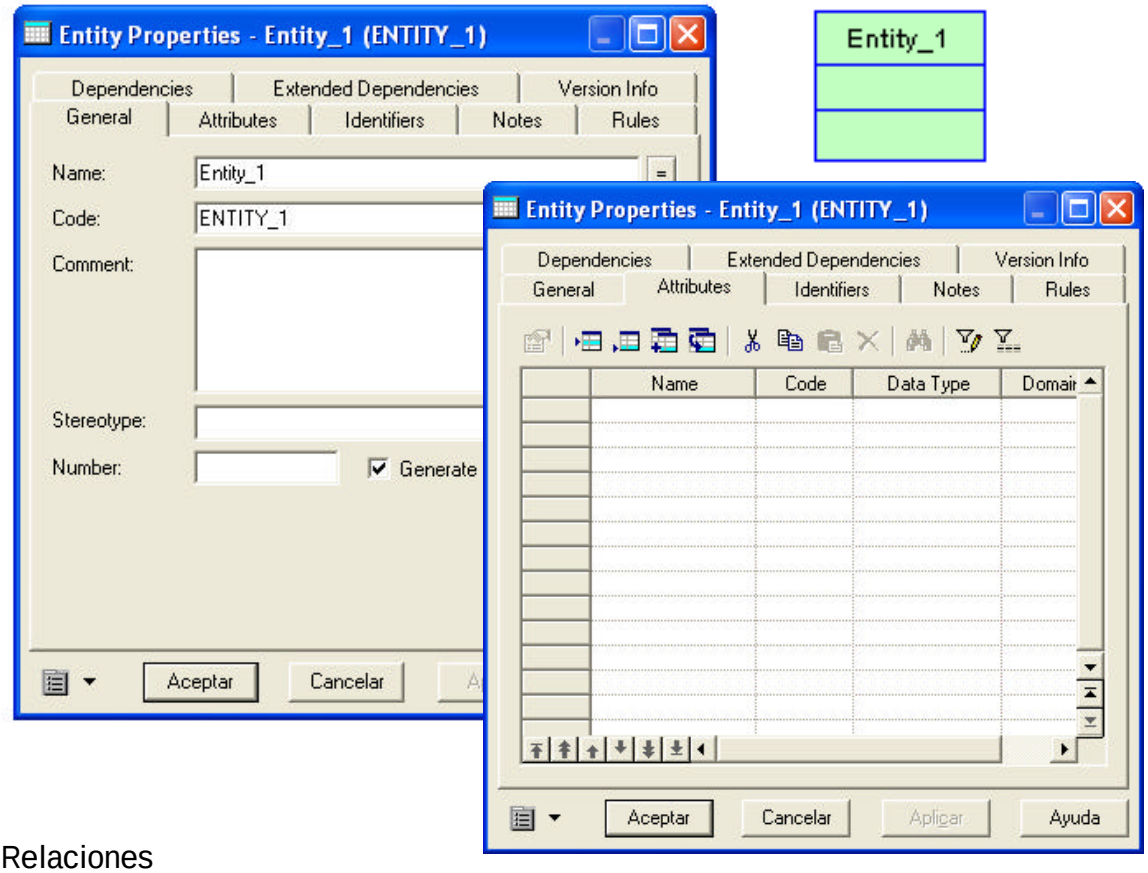
En PowerDesigner, se quiere diseñar un modelo lógico utilizando un PDM con el <Modelo lógico> del DBMS. Este PDM es un modelo físico con los objetos, DBMS sin opciones físicas específicas y generación de capacidades.

En el proceso de diseño de bases de datos, usted usa un modelo lógico como un paso intermedio entre lo físico y lo conceptual del diseño:
Comience con un CMD que contenga entidades, atributos, relaciones, dominios, datos y reglas de negocio.
Generar un modelo lógico (PDM con el <Modelo Lógico> del DBMS). Crear índices y FK especificar nombres de columna y otras características comunes.
Generar un conjunto de PDMS, cada una dirigida a una determinada aplicación DBMS.
Si es necesario, puede desarrollar el CMD en varios pasos a partir de un modelo de alto nivel a un nivel bajo de CMD.
Este proceso de diseño le permite mantener todo coherente en un gran esfuerzo de desarrollo.

Objetos en un CDM

1. Entidad

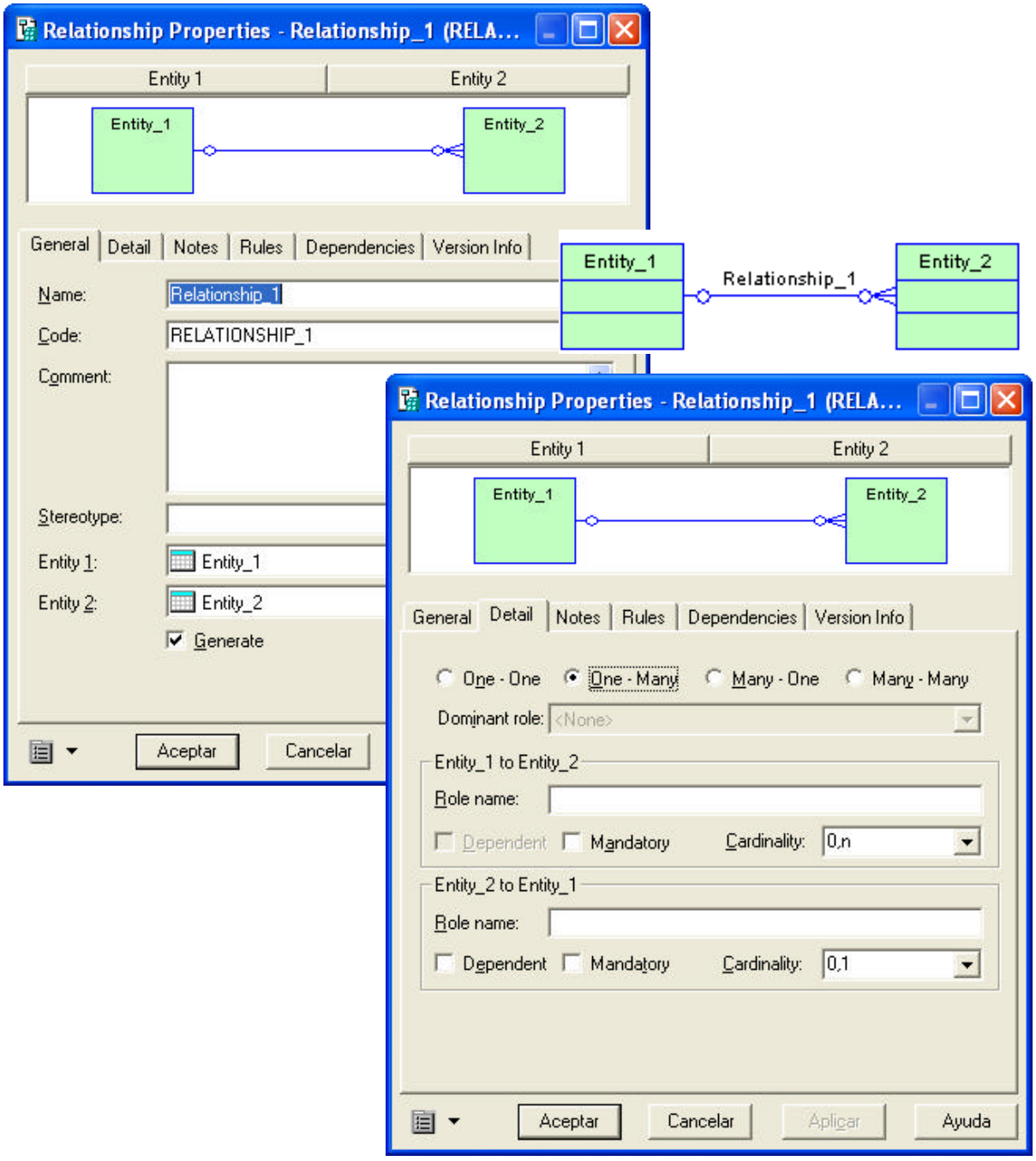
Una entidad representa un objeto definido dentro del sistema de información sobre la que desea almacenar información. Por ejemplo, en un modelo concerniente a empleados y divisiones, las entidades son los empleados y la división.
Una ocurrencia de una entidad es un elemento perteneciente a la entidad. Por ejemplo, el empleado Martín es una ocurrencia de la entidad Empleado.



2. Relaciones

Una relación es un enlace entre entidades. Por ejemplo, en un CMD que gestiona recursos humanos, la relación miembro vincula a las entidades de empleados y equipo, ya que los empleados pueden ser miembros de los equipos. Esta relación expresa que cada empleado trabaja en un equipo y que cada equipo tiene empleados.

Una ocurrencia de una relación corresponde a un caso de cada una de las dos entidades que participan en la relación. Por ejemplo, el empleado Martín trabajando en el equipo de marketing es una ocurrencia de la relación miembro.



- *Cardinalidad*
  Cardinalidad indica el número de instancias (ninguno, uno, o muchos) de una entidad en relación a otra entidad. Puede seleccionar los siguientes valores de cardinalidad

Cardinalidad	Símbolo	Descripción
Uno a uno	<1..1>	Un ejemplo de la primera entidad puede corresponder a una sola instancia de la segunda entidad
Uno a muchos	<1..n>	Un ejemplo de la primera entidad puede corresponder con mas de una sola instancia de la segunda entidad
Muchos a uno	<n..1>	Más de un ejemplo de la primera entidad puede corresponder a la misma instancia de la segunda entidad
Muchos a mucho	<n..m>	Más de una instancia de la primera entidad puede corresponder a más de una instancia de la segunda entidad

Termination point	Existence	Cardinality	Description
	Mandatory	One	Must exist one and only one
	Mandatory	Many	Must exist one or more
	Optional	One	May exist one, or none
	Optional	Many	May exist one or more, or none

3. Herencia

La herencia le permite definir una entidad como un caso especial de una entidad de carácter más general. Las entidades que participan en una herencia tienen muchas características similares, pero sin embargo son diferentes. La entidad general que se conoce como una entidad supertipo (o padre) y contiene todas las características comunes. El caso especial de entidad que se conoce como un subtipo (o hijo) de las entidades y contiene todas las características particulares.

Entre las entidades, también es posible definir una herencia enlace. En una herencia enlace, uno o más entidades subtipos (o hijo) a heredar, en el plano físico, la totalidad o una parte de la entidad atribuye a cargo de una entidad supertipo (o padre).

E/R y Notación Merise	Descripción
	Estándar
	Mutuamente exclusiva
	Herencia completa
	Herencia completa y mutuamente exclusiva

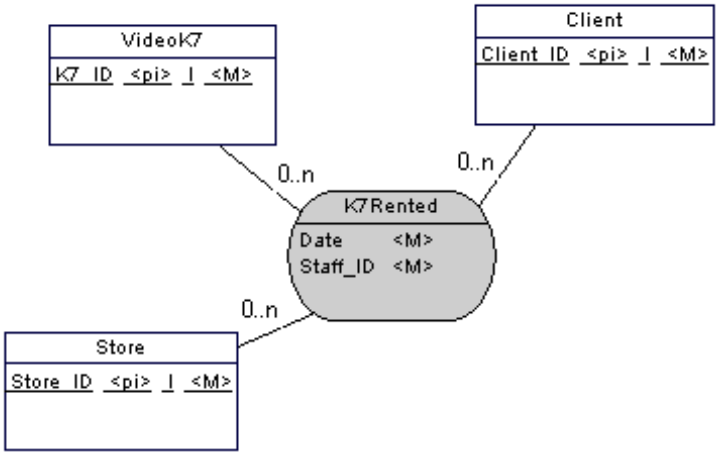
4. Asociación

Una asociación es una relación entre las entidades. En la metodología de modelado Merise una asociación se usa para conectar varias entidades que cada uno representa claramente definidos los objetos, pero están unidos por un evento, que puede no ser tan claramente representado por otra entidad.

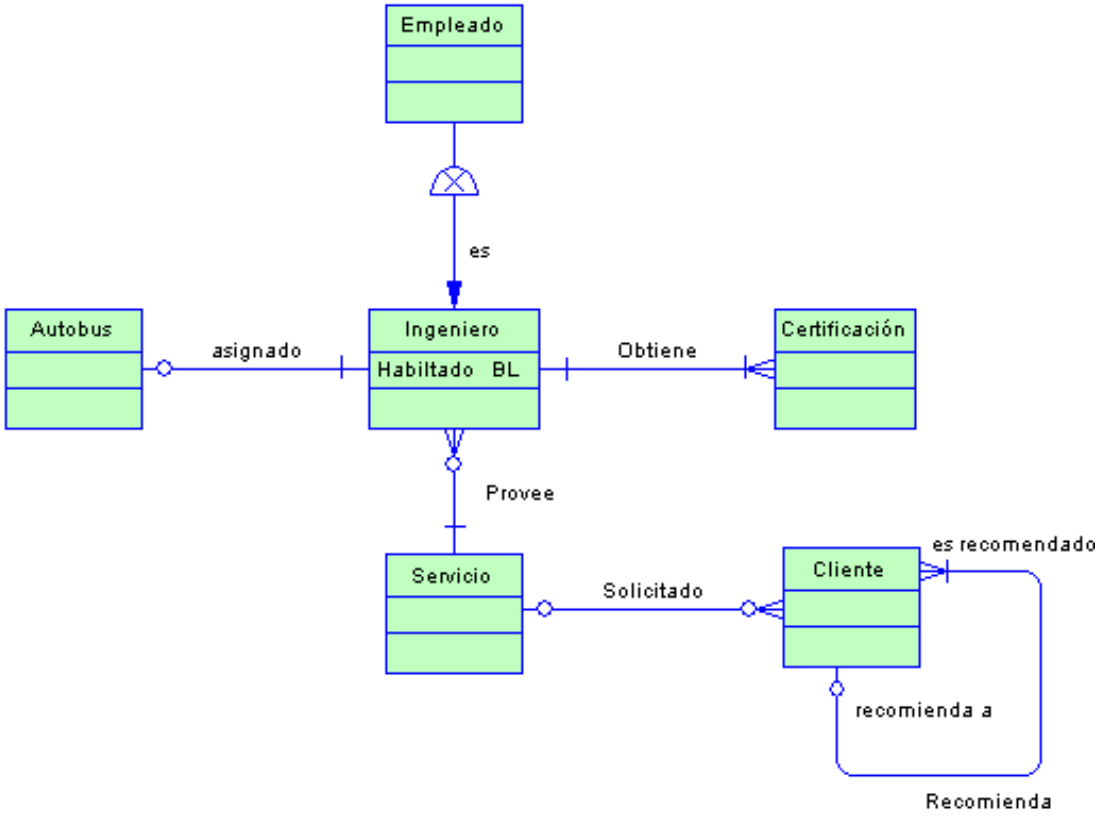
Cada instancia de una asociación corresponde a una instancia de cada entidad vinculada a la asociación.

Ejemplo:

Tres entidades VIDEOK7, CLIENTE, y ALMACEN, contiene video, cliente y almacén de información. Ellos están vinculados por una asociación que representa a una cinta de vídeo de alquiler (K7RENTAL). La asociación K7RENTAL también contiene los atributos fecha y STAFF_ID, que dan la fecha del alquiler, y la identidad del funcionario que alquila la cinta de vídeo.



Practica.- Observar su corrección



SEMANA 04: DISEÑO CONCEPTUAL DE BASE DE DATOS

29. DISEÑO CONCEPTUAL

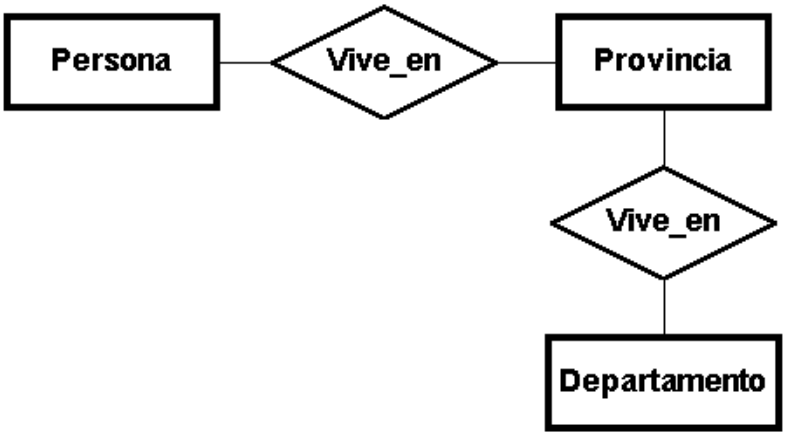
29.1. ELEMENTOS USADOS EN EL DISEÑO CONCEPTUAL.

El diseño de un esquema conceptual es el resultado de un análisis complejo de los requerimientos del usuario. El esquema conceptual se desarrolla gradualmente en un proceso iterativo, empezando con una versión preliminar del esquema y efectuando una serie de transformaciones de esquemas, que finalmente, producen la versión definitiva. Estas transformaciones se pueden restringir a un conjunto limitado, denominado transformaciones primitivas, es decir, transformaciones con una estructura simple que no se pueden descomponer en otras más simples.

PRIMITIVAS DEL DISEÑO CONCEPTUAL

Las transformaciones de esquemas se realizan tomando en cuenta que, el esquema resultante debe heredar todas las conexiones lógicas definidas para los conceptos del esquema inicial.

(a) Esquema inicial

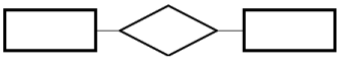
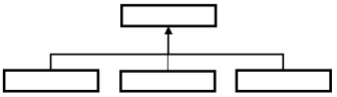
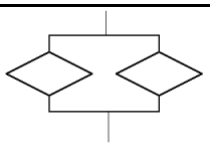
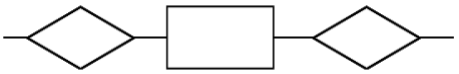
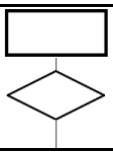
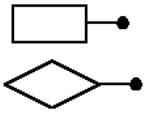


(b) Esquema resultante

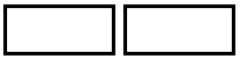
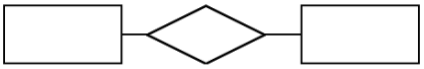
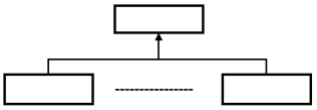
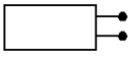
Ejemplo de transformaciones de esquemas

Como se puede apreciar en el ejemplo, la entidad LUGAR se ha transformado en dos entidades relacionadas: PROVINCIA y DEPARTAMENTO, y se mantiene la relación VIVE EN, heredando, de esta manera, la conexión lógica inicial.

PRIMITIVAS DESCENDENTES. Las primitivas descendentes se caracterizan porque por cada concepto del esquema inicial, se obtiene un nuevo conjunto de conceptos, que describen los conceptos iniciales en un nivel de abstracción mas bajo. Las conexiones lógicas deben ser heredadas por un solo concepto del esquema resultante.

PRIMITIVA	ESQUEMA INICIAL	ESQUEMA RESULTANTE
T1: Entidad ? Entidades relacionadas		
T2: Entidad ? Generalización		
T3: Entidad ? Entidades no relacionadas		
T4: Relación ? Relaciones paralelas		
T5: Relación ? Entidad con relaciones		
T6: Desarrollo de atributos		

PRIMITIVAS ASCENDENTES. Las primitivas ascendentes introducen nuevos conceptos y propiedades que no aparecían en versiones anteriores del esquema. Estas primitivas se usan en el diseño de un esquema, siempre que se descubren rasgos del dominio de aplicación que no fueron captados en ningún nivel de abstracción en la versión anterior del esquema. Las primitivas ascendentes se aplican, así mismo, cuando se fusionan esquemas diferentes para formar un esquema global más amplio (integración).

PRIMITIVA	ESQUEMA INICIAL	ESQUEMA FINAL
B1: Generación de Entidades		
T2: Entidades ? Vinculos		
T3: Entidades ? Generalización		
T4: Atributos? Agregación de atributos		

Clasificación de las primitivas ascendentes

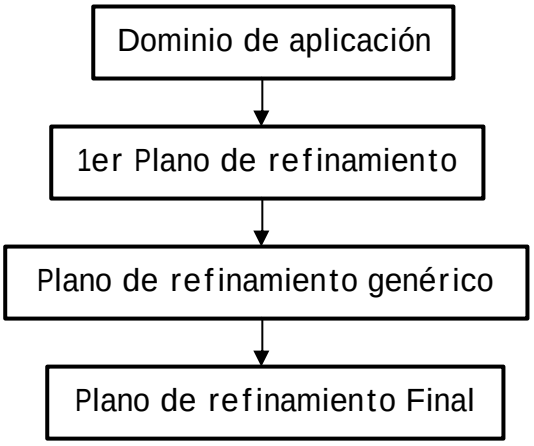
29.2. ESTRATEGIAS PARA EL DISEÑO DE ESQUEMAS

Para desarrollar el esquema conceptual, se sigue una estrategia, la cual, mediante iteraciones, y partiendo de un esquema inicial, logra sucesivamente esquemas más refinados. Se distinguen cuatro estrategias para el diseño de Esquemas:

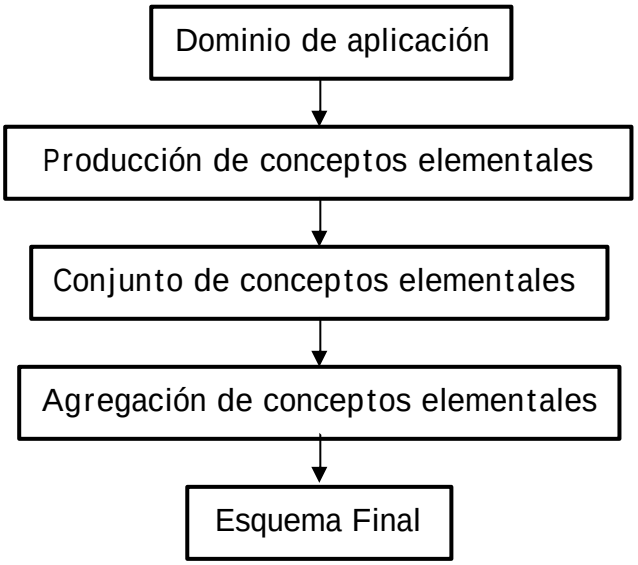
Estrategia descendente, ascendente, centrifuga y mixta. Cada una se caracteriza por el uso de tipos particulares de primitivas.

A continuación, explica brevemente como es que opera cada estrategia:

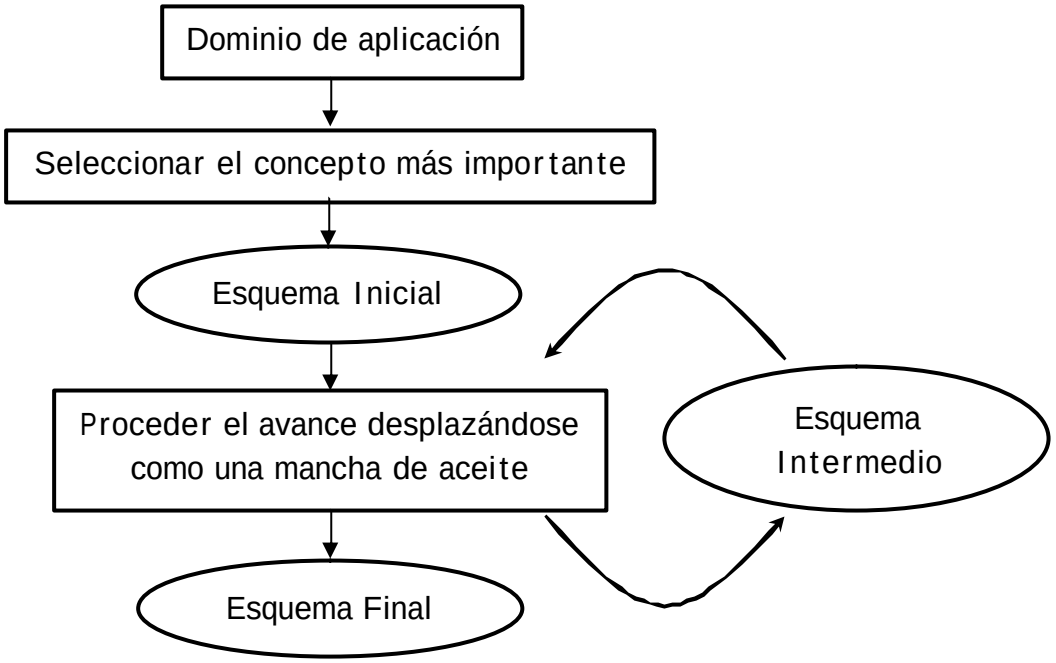
A. **Estrategia descendente**. Se obtiene un esquema aplicando solo las primitivas de refinamiento descendente; cada primitiva introduce nuevos detalles en el esquema. El proceso termina cuando están representados todos los requerimientos.



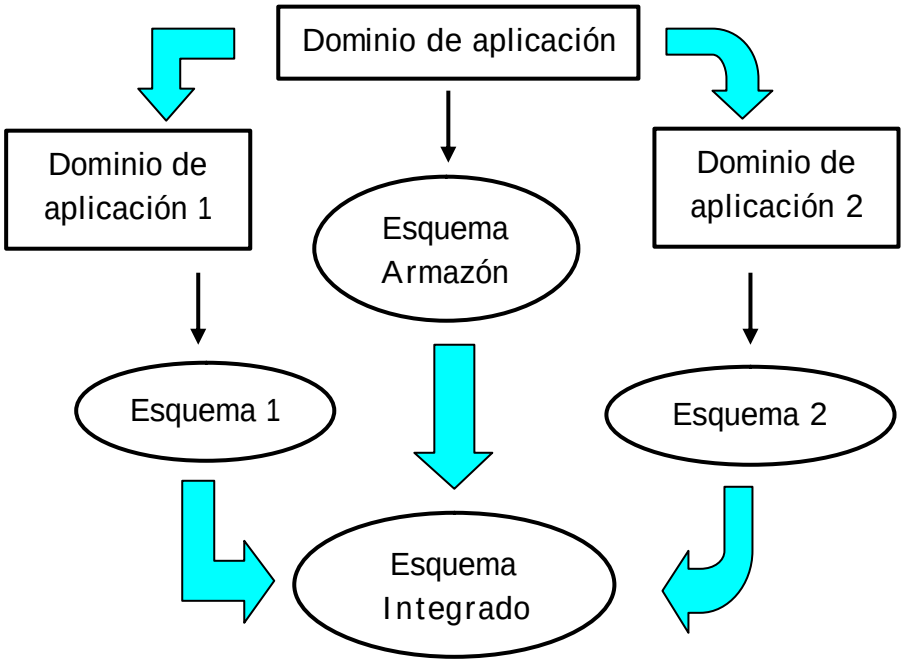
B. **Estrategia ascendente**. Se obtiene un esquema aplicando solo las primitivas de refinamiento ascendente, partiendo de conceptos elementales y construyendo conceptos más complejos a partir de ellos; los requerimientos se descomponen, se conceptualizan de manera independiente y, finalmente, se fusionan en un esquema global.



C. **Estrategia centrifuga**. Es un caso especial de estrategia ascendente. Primero se fijan los conceptos más importantes o evidentes, y luego se procede con un movimiento similar al de una mancha de aceite, seleccionando primero los conceptos más importantes o más cercanos al concepto inicial, y navegando después hacia los más distantes.



D. **Estrategia mixta**. Aprovecha tanto la estrategia ascendente como la descendente, al permitir particiones controladas de los requerimientos.



29.3. CUALIDADES DE UN BUEN ESQUEMA DE BASE DE DATOS

El esquema de una base de datos debe validarse antes de convertirse en un producto estable del diseño. Este proceso de validación se realiza revisando varias cualidades que debería poseer un buen esquema, estas cualidades se examinan a continuación:

- Complección esquema es completo cuando representa todas las características pertinentes del dominio de la aplicación. La complección puede, en principio, comprobarse:

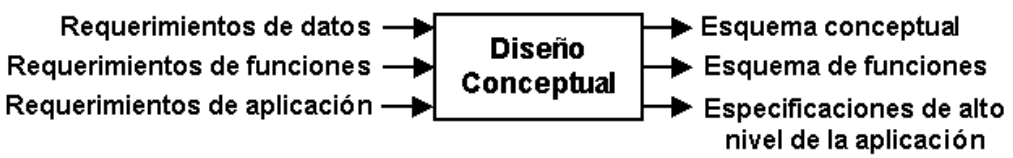
- Mirando con detalle todos los requerimientos del dominio de la aplicación y verificando que cada uno de ellos esté representado en algún lugar del esquema.
- Revisando el esquema para ver que cada concepto se mencione en los requerimientos (en este caso, se dice que los requerimientos están completos respecto al esquema).

- **Corrección.** Un esquema es correcto cuando usa con propiedad los conceptos del modelo ER. Se pueden distinguir dos tipos de corrección, sintáctica y semántica. Un esquema es sintácticamente correcto cuando los conceptos se definen con propiedad en el esquema; por ejemplo, los subconjuntos y las generalizaciones se definen entre entidades pero no entre relaciones. Un esquema es semánticamente correcto cuando los conceptos (entidades, relaciones, etc.) se usan de acuerdo con sus definiciones.
- **Minimalidad**. Un esquema es mínimo cuando cada aspecto de los requerimientos aparece sólo una vez en el esquema. También se puede decir que un esquema es mínimo si no se puede borrar del esquema un concepto sin perder alguna información. Si el esquema no es mínimo, entonces tiene elementos redundantes.
- **Expresividad**. Un esquema es expresivo cuando representa los requerimientos de una forma natural y se puede entender con facilidad, sin necesidad de explicaciones adicionales.
- **Legibilidad**. Esta es una propiedad del diagrama que representa gráficamente al esquema. Un diagrama tiene buena legibilidad cuando representa ciertos criterios estéticos que hacen el diagrama elegante.
- **Autoexplicación**. Un esquema se explica a sí mismo cuando pueden representarse un gran número de propiedades usando solamente el modelo conceptual, sin otras anotaciones adicionales (por ejemplo, anotaciones en lenguaje natural).
- **Extensibilidad**. Un esquema es extensible si ofrece facilidad para ampliar los conceptos que representa. Un esquema se adapta fácilmente a requerimientos cambiantes cuando puede descomponerse en partes (módulos o vistas), a fin de aplicar los cambios dentro de cada parte.
- **Normalidad**. El concepto de normalidad viene de la teoría de la normalización, asociada al modelo relacional. Las formas normales (primera, segunda, tercera, cuarta y una variación de la tercera forma normal, llamada forma normal de Boyce-Codd), pretenden mantener la estructura lógica de los datos en una forma normal limpia, "purificada", mitigando los problemas de anomalías de inserción, borrado y actualización que ocasionan trabajo innecesario, porque deben aplicarse los mismos cambios a varios casos de datos, así como el problema de la pérdida accidental de datos o la dificultad de representación de determinados hechos.
  Cuando más alta sea la forma normal, mayor será la pureza de la definición del Esquema Relacional.

29.4. METODOLOGÍA PARA EL DISEÑO CONCEPTUAL

Las primitivas y las estrategias son los pilares para el desarrollo de las metodologías de diseño conceptual. Las entradas y salidas del diseño conceptual se indican en el siguiente diagrama;

La metodología propuesta por Batini/Ceri/Navathe, consiste de las siguientes actividades:



Análisis de requerimientos. Se estudian minuciosamente los requerimientos para empezar a producir un esquema conceptual.
Conceptualización inicial. El objetivo es realizar una primera selección de los conceptos que se van a representar en el esquema conceptual. En este nivel se crea un conjunto preliminar de abstracciones, buenas candidatas para ser representadas como entidades, relaciones o generalizaciones. El esquema producido es, en gran medida, incompleto, porque representa sólo algunos aspectos de los requerimientos.
Conceptualización incremental. Es la actividad central del diseño conceptual. Usando las estrategias generales, el esquema preliminar se refina para dar lugar a un esquema conceptual final.
Integración. Es una actividad típica de las estrategias mixtas y ascendentes. Implica la fusión de varios esquemas y la producción de una nueva representación global de todos ellos.
Reestructuración. Es una suspensión del proceso de diseño conceptual y prestar la atención a medir y mejorar la calidad del producto obtenido. Este proceso de validación se realiza revisando las cualidades deseables de un buen esquema, como son: compleción, corrección, minimalidad, expresividad, legibilidad, autoexplicación, extensibiíidad y normalidad, las cuales fueron explicadas anteriormente.

29.5. OBSERVACIÓN IMPORTANTE SOBRE LA NORMALIZACIÓN

En el modelo relacional, la normalización de relaciones es un proceso de aplicación de transformaciones progresivas para lograr la forma normal deseada. El proceso está guiado por las dependencias funcionales.
La metodología con un enfoque orientado a datos, estudiada en este trabajo, tiene una visión muy especial respecto a la normalización:
Se considera a la ***normalización como una herramienta para validar la calidad del esquema, más que como un método para diseñar el esquema***.
Esto es discutible porque varios enfoques proponen a la normalización como el único método aceptable para diseñar bases de datos. Estos enfoques sugieren el uso del propio modelo relacional como modelo de diseño y producen como resultado una serie de relaciones en una forma normal dada.
Se destaca que el modelo ER y los elementos de diseño (primitivas, estrategias y actividades) propuestos para el diseño conceptual, ***tienden a producir con naturalidad esquemas normalizados.***
Finalmente, es oportuno indicar que los casos que violan la normalización, se deben generalmente, a un mal diseño del esquema conceptual. Esto concuerda con la visión de considerar a la normalización como herramienta de validación.

ENUNCIADO DE CASO DE APLICACIÓN

Para una base de datos de un censo (que representa varias propiedades de las personas y lugares geográficos), se tiene los siguientes requerimientos:

- ✓ En esta base de datos demográfica se consideran las siguientes propiedades de las personas: nombre, apellido, sexo, estatura, edad, lugar de nacimiento, lugar de residencia, años de residencia. Situación militar de los hombres, apellido de soltera de las mujeres.
- ✓ Los lugares pueden ser países o ciudades nacionales. Cada uno tiene nombre y número de habitantes (que representa la población total en el caso de los países), se debe considerar los nombres de las regiones nacionales.

SEMANA 05: CASOS DE DISEÑO CONCEPTUAL

30. DISEÑO CONCEPTUAL DE UN SISTEMA

La metodología propuesta en la sección 7.3 define los pasos para el Diseño Conceptual, según Batini. El objetivo de esta fase es crear el Esquema Conceptual Inicial, para ello se trabaja en secuencia, con una serie de esquemas.

Hay varias formas de expresar los requerimientos de los usuarios: los hay desde enunciados en lenguaje natural, hasta formularios predeterminados y esquemas de datos. La primera forma (lenguaje natural) es la mas general, por ello se hará énfasis en un método que trabaja con este tipo de descripción inicial.

Considerando que se parte de los requerimientos del usuario, expresados en lenguaje natural, los pasos a seguir para el Diseño Conceptual se resumen así:

DISEÑO CONCEPTUAL A PARTIR DE LOS REQUERIMIENTOS EN LENGUAJE NATURAL

1. Análisis de los requerimientos.
   - 1.1. Se analizan los requerimientos en lenguaje natural y se filtran las ambigüedades para obtener los REQUERIMIENTOS DE LA BASE DE DATOS. Algunas recomendaciones para obtener un enunciado más estructurado son:
     - Los términos deben seguir un nivel de abstracción apropiado. Por ejemplo es diferente hablar de lugar y de ciudad, asimismo de lapso y de número de años.
     - No usar casos cuando se requieren conceptos más generales. Por ejemplo un electrónico puede pedir el stock de chips, el término chip no es un concepto, sino que es un caso del concepto correcto: componente
     - Verificar los sinónimos y homónimos.
     - Utilizar un glosario de términos.
   - 1.2. Dividir los enunciados en conjuntos homogéneos, para obtener los grupos principales sobre conceptos candidatos a ser entidades.
2. Diseño Inicial. El objetivo es obtener un Esquema E-R Armazón Inicial
   - 2.1. Obtener los esquemas E-R parciales, según los grupos de enunciados
   - 2.2. Fusionar (integrar) los esquemas parciales para obtener el Esquema Armazón Inicial
3. Diseño de Esquemas. Para cada concepto del Esquema Armazón, aplicar:
   - 3.1. Refinamientos descendentes
   - 3.2. Refinamientos ascendentes
   - 3.3. Refinamientos centrífugos

Con estos pasos se obtiene el Esquema Armazón refinado llamado también el Esquema Conceptual.

A continuación se desarrolla un caso para mostrar los pasos anteriores:

31. CASO: DISEÑO CONCEPTUAL DE UN SISTEMA ACADÉMICO

El presente es el caso de una universidad que requiere diseñar su sistema académico el cual soporta a varias Escuelas (carreras) y con la posibilidad de que cada Escuela tenga varios planes de estudio. Se muestra el resultado de cada paso indicado anteriormente:

1. ANÁLISIS DE LOS REQUERIMIENTOS

1.1. Filtrar las ambigüedades:

En el análisis de requerimientos, cuestionarios, entrevistas y como consecuencia de la observación directa, y luego de aplicar las

recomendaciones indicadas anteriormente se obtuvo el siguiente enunciado, en forma estandarizada:

REQUERIMIENTOS DE LA BASE DE DATOS

En la Base de datos del Sistema Académico se tienen varias Escuelas, cada una de ellas posee una o más currículos. Cada currículo contiene un grupo de cursos. Para el curso se representa su nombre, número de créditos, horas de teoría, prácticas y total de horas, el ciclo y el tipo del curso. Para los alumnos, se requieren sus datos personales como su apellido, nombre, sexo, dirección, teléfono; Además, se tiene su información académica resumida, como su Escuela, el currículo que sigue, el semestre de ingreso, su promedio ponderado total, el número de créditos aprobados y el estado académico del alumno (activo, inactivo o egresado).

Cada semestre se eligen de entre todos los cursos y currículos, los cursos a dictar, obteniéndose los cursos-sección programados. De esta manera, el profesor encargado del curso recibe un listado de los alumnos asistentes a cada curso-sección programado.

Asimismo, los cursos-sección programados, se dictan en una serie de aulas disponibles, en horas definidas y dirigidos por uno o varios profesores por curso (cada profesor dicta en horas fijas).

La Base de datos registra las notas de los alumnos al llevar los cursos en forma regular cada semestre. También hay notas que se obtienen por Convalidación (aplicable en caso de traslado externo o de segunda especialización), emitiéndose para ello un documento de convalidación que contiene la nota para varios cursos.

De la misma manera, cada Escuela tiene una Plana Docente, la cual es renovada cada semestre, y es elegida de entre todo el grupo de profesores de la Universidad. Un profesor puede dictar en varias Escuelas a la vez, sin embargo, tiene una sola Escuela titular.

El alumno se matricula mediante una ficha en los cursos-sección programados, siguiendo un control de pre-requisitos y de la cantidad de créditos máxima a llevar, de acuerdo a su promedio ponderado. Esta matrícula es realizada y verificada por un Asesor, acotándose que cada Escuela tiene un grupo de Asesores a disposición de los alumnos.

Por otro lado, se requiere tener las Tablas de Equivalencias entre los cursos de diferentes currículos, existiendo en este caso, equivalencias simples. Las Tablas de Equivalencias permitirán el cambio de currículo del alumno. Finalmente, cada curso tiene cero o varios pre-requisitos (cursos del mismo currículo que el curso titular).

1.2. División de los enunciados en conjuntos homogéneos.

Aquí se considera el agrupamiento de los enunciados en conjuntos homogéneos, a fin de reconocer los conceptos y facilitar el detalle durante el diseño. De esta manera, distinguimos los siguientes grupos de enunciados:

a) Enunciados sobre los currículos y cursos:
   En la Base de datos del Sistema Académico se tienen varias Escuelas, cada una de ellas posee una o más currículos. Cada currículo contiene un grupo de cursos. Para el curso se representa su nombre, número de créditos, horas de teoría, prácticas y total de horas, el ciclo y el tipo del curso.
b) Enunciados sobre los alumnos:
   Para los alumnos se requieren sus datos personales como su apellido, nombre, sexo, dirección y teléfono; Además, se tiene su

información académica resumida, como su Escuela, el currículo que sigue, el semestre de ingreso, su promedio ponderado total, el número de créditos aprobados y el estado académico del alumno (activo, inactivo o egresado).

c) Enunciados sobre la programación de los cursos:
Cada semestre se elige de entre todos los cursos y currículos, los cursos a dictar, obteniéndose los cursos-sección programados. De esta manera, el profesor encargado del curso recibe un listado de los alumnos asistentes a cada curso-sección programado, Asimismo, los cursos-sección programados, se dictan en una serie de aulas disponibles, en horas definidas y dirigidos por uno o varios profesores por curso (cada profesor dicta en horas fijas).

d) Enunciados sobre las notas:
La Base de datos registra las notas de los alumnos por llevar los cursos en forma regular cada semestre. También hay notas obtenidas por Convalidación (aplicable en caso de traslado externo o de segunda especialización), emitiéndose para ello un documento de convalidación que contiene la nota para varios cursos.

e) Enunciados sobre los profesores:
De la misma manera, cada Escuela tiene una Plana Docente, la cual es renovada cada semestre, y es elegida de entre todo el grupo de profesores de la Universidad. Un profesor puede dictar en varias Escuelas a la vez, sin embargo, tiene una sola Escuela titular.

f) Enunciados sobre la matrícula:
El alumno se matricula mediante una ficha en los cursos-sección programados, siguiendo un control de pre-requisitos y de la cantidad de créditos máxima a llevar, de acuerdo a su promedio ponderado. Esta matrícula es realizada y verificada por un Asesor, acotándose que cada Escuela tiene un grupo de Asesores a disposición de los alumnos.

g) Enunciados sobre equivalencias de cursos y pre-requisitos:
Por otro lado, se requiere tener las Tablas de Equivalencias entre los cursos de diferentes currículos, existiendo en este caso, equivalencias simples. Las Tablas de Equivalencias permitirán el cambio de currículo del alumno. Finalmente, cada curso tiene cero o varios pre-requisitos (cursos del mismo currículo que el curso titular).

2. DISEÑO INICIAL

Aquí se obtendrá un esquema armazón inicial. Los conceptos que aparecen en este esquema son los más evidentes mencionados en los requerimientos. Para obtener el esquema armazón, nos servimos del agrupamiento por conceptos realizado en el paso anterior (conceptos que son buenos candidatos a convertirse en entidades del esquema armazón).

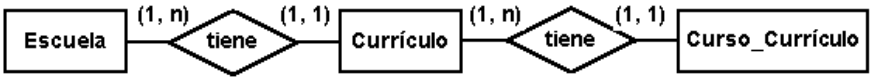
Podemos distinguir, inicialmente, las siguientes posibles entidades:

Escuela	*Currículo*	*Curso-currículo*
Equivalencias	*Pre-requisito*	*Curso-sección programado*
Asesor	*Profesor*	*Alumno*
Nota	*Ficha de Matrícula*	*Aula*
Sesión de clases	*Hora*	

2.1. Esquemas parciales según los grupos de enunciados:

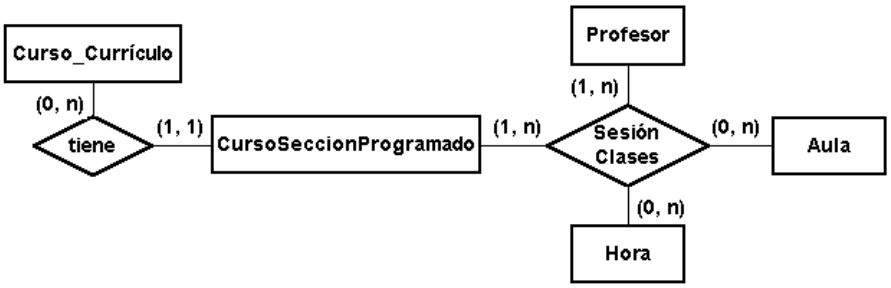
A un alto nivel, y todavía sin precisar atributos, se obtuvieron los siguientes esquemas parciales:

a) *Enunciados sobre los currículos y cursos:*
Existen varias Escuelas, cada una con varios currículos y cada currículo incluye varios cursos. El esquema parcial sería:

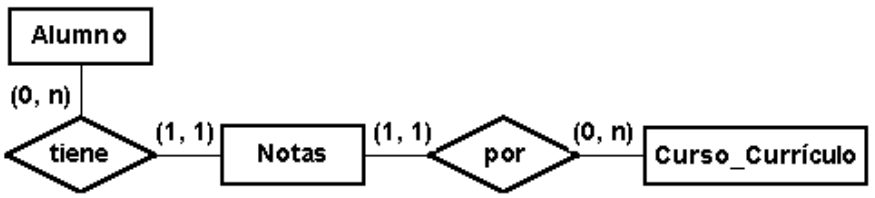


b) *Enunciados sobre los alumnos:*
Básicamente, hay que resaltar la dependencia de un alumno a un currículo, esto implica su pertenencia a una Escuela.

c) *Enunciados sobre la programación de los cursos:*
Un CURSO-SECCIÓN PROGRAMADO, es un curso que se dicta en un semestre determinado, pueden haber varias secciones del mismo curso.
El horario de las sesiones de clases esta definido en la relación cuaternaria SESIÓNCLASES. La entidad HORA es un listado de horarios, así cada hora de cada día, tendrá un código de hora.
Finalmente, es importante destacar que Profesor se relaciona con SESIÓNCLASES y no directamente con el Curso, pues puede suceder que un curso tenga varios profesores y es necesario especificar en que horas del curso dicta cada profesor.

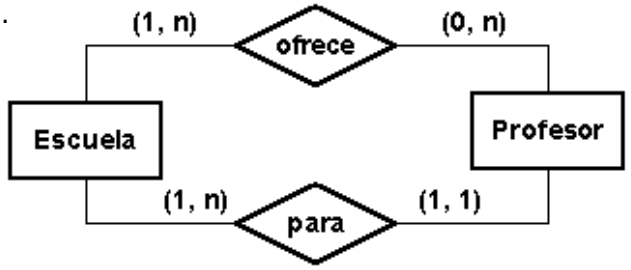


d) *Enunciados sobre las notas:*
Por el momento se tendrá de manera simplificada, una entidad NOTAS, más adelante se especificará el tipo de nota.

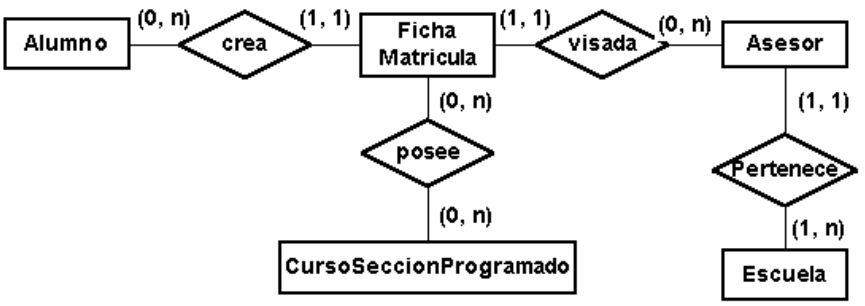


e) *Enunciados sobre los profesores:*
Aquí se enfoca solo las relaciones del PROFESOR con la ESCUELA. La relación OFRECE se refiere a los profesores que dictarán por un Escuela en un semestre determinado, en cambio la relación PARA indica a que Escuela pertenece el Profesor (su Escuela Titular), como se ve, en una Escuela se pueden dictar clases de profesores de diferentes Escuelas (la relación OFRECE indica que un profesor puede dictar en muchas Escuelas y que en

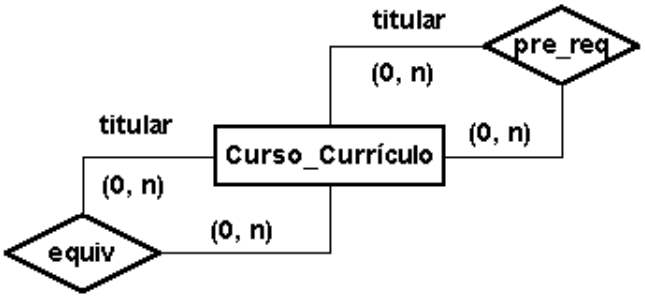
estas, dictan muchos profesores, no necesariamente de la misma Escuela).



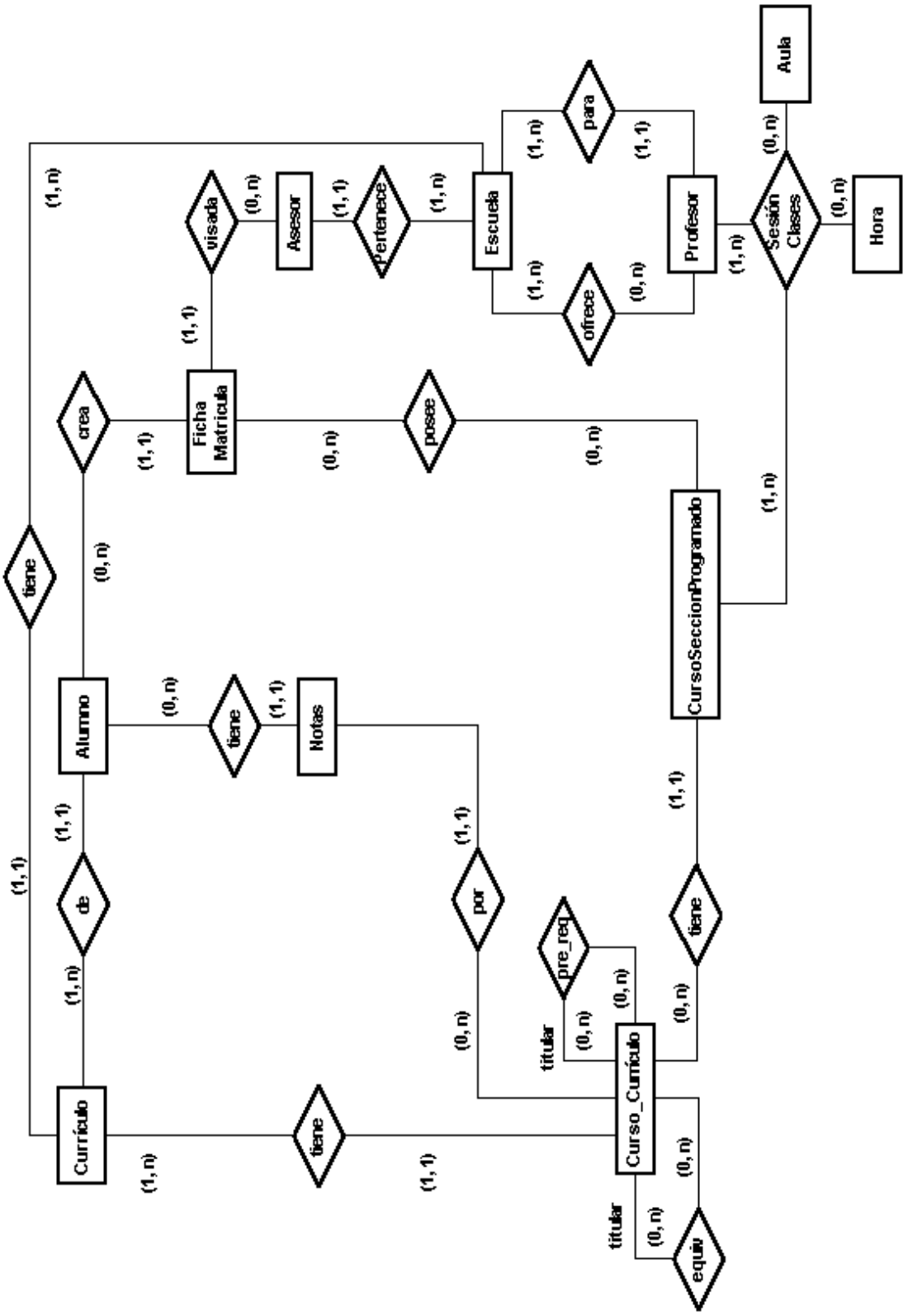
f) Enunciados sobre la matrícula:
Los alumnos se matriculan en los Cursos-sección programados, por intermedio de una ficha de matrícula, la cual es visada por un Asesor de la Escuela.



g) Enunciados sobre equivalencias de cursos y pre-requisitos:
Las tablas de Equivalencias quedan modeladas como relaciones recursivas, por ejemplo las equivalencias simples, representada como EQUIV_A, relaciona un curso (el titular) con su equivalente, notamos que un curso puede tener muchos equivalentes, a su vez un curso puede ser el equivalente de uno a varios cursos (uno a la vez). El esquema conceptual parcial siguiente muestra como es que se implementa el enunciado usando las relaciones recursivas.



Como se ve en el esquema, de la misma forma, se modelan los pre-requisitos, un curso titular tiene cero o más pre-requisitos, a su vez un curso puede ser el pre-requisito de cero o muchos cursos.
Al fusionar los esquemas parciales anteriores, se obtiene el Esquema Armazón Inicial, el cual se muestra en el gráfico siguiente.



Esquema Armazón Inicial

3. DISEÑO DE ESQUEMAS

Aquí se refina y extiende el esquema a fin de representar todas las características expresadas en los requerimientos.

El análisis se concentra en el Esquema Armazón Inicial, verificando si se puede refinar con el uso de los refinamientos (primitivas) descendentes, ascendentes o centrífugos.

3.1. Refinamientos descendentes

La entidad NOTAS se puede refinar en subconjuntos a fin de ser más específicos, estos subconjuntos corresponden a los 2 tipos de notas y son: Las obtenidas por convalidaciones, representadas en la entidad CONVALIDACIONES y las de los cursos llevados en cada semestre, representadas en la entidad REGULAR.

3.2. Refinamientos ascendentes

De manera similar al surgir la entidad REGULAR, que son las notas del semestre actual, es necesario indicar a que curso del semestre actual se refieren, por lo que surge la relación DE, entre las entidades REGULAR y CURSOSECCIONPROG.

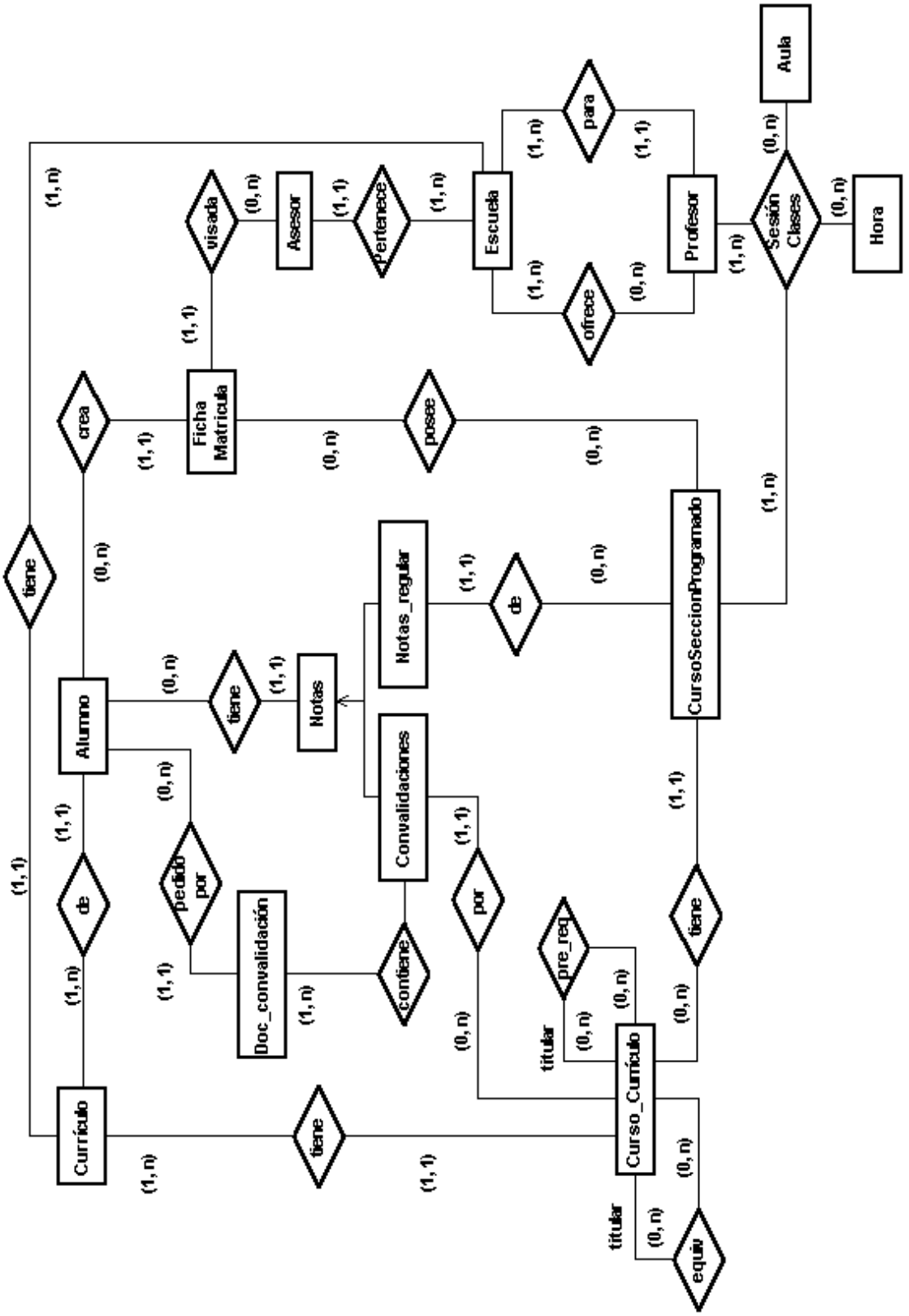
3.3. Refinamientos centrífugos

Al surgir la entidad que representa a las notas de CONVALIDACIONES, se ve la necesidad de referenciarlas al documento que les dio origen, en este caso, a la resolución de convalidación, la cual contiene varias notas de un solo alumno, surge así la entidad DOC_CONVALIDAC, la cual inclusive, trae consigo las relación con la entidad ALUMNO (relación PEDIDOPOR) y la relación con la entidad CONVALIDACIONES (relación CONTIENE).

Producto de estos refinamientos se obtiene el Esquema Armazón Refinado al cual también se le denomina Esquema Conceptual, (o Esquema Conceptual Inicial) y se muestra a continuación:

3.4. Refinamientos Mixtos

Emplea las estrategias descendentes y ascendentes permitiendo dividir en forma controlada los requerimientos. Partiendo de n dominios de aplicación y utilizando un esquema armazón, es posible llegar a un esquema final integrado.



Esquema Conceptual

Para el proyecto de asignatura (de la empresa que hallan elegido), acompañar el CDM en PowerDesigner, que se corresponda con su esquema final.

32. Problemas Propuestos Diseño Conceptual

Estrategias de diseño

Instrucciones.- Resolver con los grupos ya conformados y presentar, el día del examen.

1. Una compañía de seguros de automóviles desea informatizar la gestión de todas sus operaciones, para lo cual quiere crear una base de datos que recoja el funcionamiento de su negocio y que se refleja en los siguientes supuestos semánticos:
   a. El elemento fundamental de información es la póliza, la cuál se identifica mediante un número único, tiene un tipo de seguro (a todo riesgo, a terceros, etc.), un importe de cobertura máxima y un estatus (alta, baja, suspensión, etc.). La póliza pertenece a un único cliente (un cliente puede tener más de una póliza diferente) y referencia a un único vehículo y cada vehículo sólo puede tener una póliza.
   b. Los clientes se referencian mediante su NIF, además se quiere guardar su nombre y apellidos, su teléfono, fecha de nacimiento, fecha de obtención del permiso de conducir y su dirección completa (calle, número, ciudad, código postal y provincia).
   c. De los vehículos es importante conocer su número de chasis, su matrícula, la marca, el modelo, la potencia, el año de fabricación y el color. Además un vehículo puede tener una serie de extras (alarma, auto radio, etc.)
   d. Una póliza puede tener una serie personas autorizadas, de las cuales se quiere tener su NIF, nombre y apellidos, fecha de nacimiento y relación con el cliente. Un autorizado sólo tendrá relación con un único cliente.
   e. Cuando se produce un siniestro, se crea un parte de accidente (identificado por un número de siniestro) donde se recoge la información del siniestro: datos de la póliza del cliente, datos del conductor (sólo puede ser el cliente, o alguien autorizado), fecha del siniestro, datos del taller donde se va a reparar el vehículo y fecha e importe de la reparación. Si el accidente es contra otro vehículo no se guardan ninguna información del vehículo contrario, si es de la misma compañía el cliente ya dará su propio parte de accidente. Diseñe la BD correspondiente empleando la estrategia Descendente.

2. Se desea mantener una base de datos para una cadena de farmacias distribuida en diferentes ciudades. Cada farmacia tiene sus empleados propios y un farmacéutico. Por cada ciudad existe un único farmacéutico; esto es, si en una ciudad hubiera más de una farmacia, el mismo farmacéutico estaría afectado a todas las farmacias de esa ciudad. Cada farmacia tiene a su vez su stock de medicamentos. El mismo se mantiene por medicamento y presentación. Los medicamentos se organizan según la o las monodrogas que lo componen, su presentación (por ejemplo ampollas de 5 unidades, jarabe de 100ml, inyecciones por 10 unidades, pomada 60gr, etc.), el laboratorio que lo comercializa, y su acción terapéutica (analgésico, antibiótico, etc.). Por cada medicamento se mantiene su precio y la cantidad en existencia del mismo. El sistema deberá permitir consultar la base de datos de diferentes alternativas para medicamentos compuestos por una monodroga, medicamentos de un laboratorio, presentaciones de un medicamento, entre otras. Diseñe la BD correspondiente empleando la estrategia Ascendente.

3. Los requerimientos que se muestra a continuación describe la información de un gabinete de ingenieros que realizan proyectos de instalaciones eléctricas industriales. Las empresas que desean los servicios del gabinete contactan con el departamento de atención al cliente, que abre una ficha de proyecto, asignándole un número que lo identificará en adelante. En esta ficha se registran los datos de la empresa y se deposita en la bandeja de nuevos proyectos del ingeniero jefe. Todas las mañanas, el ingeniero jefe revisa los nuevos proyectos, asignando a cada uno el ingeniero que considera adecuado, al tiempo que se lo comunica a éste personalmente y lo anota en la ficha. El ingeniero asignado visita la empresa y, en función de las necesidades del cliente, elabora un presupuesto que adjunta a la ficha del proyecto. En este presupuesto figuran las descripciones de las tareas a realizar, el presupuesto para cada tarea y el importe total. Cada tarea tiene fijado un importe base que es siempre el mismo, independientemente del proyecto. Cuando el presupuesto se envía a la empresa, ésta puede aceptarlo o no, por lo que habrá proyectos

aceptados y no aceptados. Cuando un proyecto es aceptado, el ingeniero jefe decide la fecha de inicio y le asigna los operarios necesarios de cada especialidad, comprobando que no estén ocupados en otro proyecto. Toda esta información también se registra en la ficha del proyecto. Periódicamente, para los proyectos de larga duración, el ingeniero asignado debe informar del grado de ejecución del proyecto. Una vez finalizados los trabajos de un proyecto, el ingeniero asignado lo comunica al ingeniero jefe que procede a anotarlo en la ficha del proyecto y la envía al departamento de contabilidad para que proceda a gestionar el cobro. Diseñe la BD correspondiente empleando la estrategia Centrifuga

4. Los requerimientos que se muestra a continuación describen la información que mantiene un gimnasio sobre las clases que imparte, sus socios y sus monitores. Las clases se imparten en las distintas salas del gimnasio. Cada sala tiene un número, una ubicación dentro del gimnasio, es de un tipo (cardio, general, muscular) y tiene un número de metros cuadrados. Hay salas que tienen aparatos y salas que no. Los aparatos tienen un código, una descripción y una indicación de su estado de conservación. Algunos de ellos están asignados a una sala de forma permanente. De las clases que se imparten se tiene un código, el tipo de clase (step, aerobic, spinning, etc.), el día de la semana en que se imparte cada clase y la hora. Estas clases las imparten monitores, de los que se tienen sus datos personales (DNI, nombre, teléfono), titulación (si la tienen), la experiencia profesional y su preparación como monitores, es decir, qué tipos de clases pueden impartir (step, aerobic, spinning, etc.) y desde qué año. Quienes reciben las clases son los socios, de los que se tiene su número, los datos personales (nombre, teléfono, dirección), su profesión y la cuenta bancaria a través de la que pagan las mensualidades del gimnasio. Además, el gimnasio posee pistas de squash. Cada pista tiene un número (distinto del de cualquiera de las salas), una ubicación dentro del gimnasio y una indicación sobre su estado de conservación. Estas pistas pueden ser reservadas por los socios. Cada reserva será para una fecha y una hora determinada. Para poder llevar a cabo estudios sobre la utilización de las pistas, se quiere mantener información histórica de todas las reservas realizadas. Diseñe la BD correspondiente empleando la estrategia Mixta.

5. Se requiere diseñar la base de datos de un sistema de administración hotelera. Se tienen aproximadamente 200 habitaciones de diferentes capacidades y tarifas. Hay varios tipos de habitaciones, desde individuales hasta habitaciones para 4 personas. Se disponen además de 15 camareras, las cuales deben realizar la limpieza de las habitaciones. Se requiere controlar la disponibilidad de habitaciones, las identidades de las personas que ocupan cada habitación, y la programación de la limpieza de las habitaciones, a cargo de las camareras. Elaborar el esquema entidad-relación, de la base de datos del sistema descrito, indicando atributos y claves primarias. Emplear la estrategia descendente. (Sugerencia: considerar la entidad alquiler)

6. Se requiere controlar la historia clínica de los clientes inscritos en una Empresa que brinda seguro médico por medio de una serie de Clínicas y Farmacias a disposición de los clientes. Cada historia clínica registra las atenciones que ha tenido cada paciente. En cada atención se debe indicar el Doctor que lo atendió (un Doctor puede trabajar en varias clínicas), la Clínica donde se atendió, el diagnóstico, y los medicamentos recetados. De los medicamentos, solo algunos son cubiertos por el seguro, por lo que se debe indicar desde que farmacia del seguro se hizo la entrega de cada medicamento. Además de los medicamentos, en cada atención es posible que se incluyan otros servicios, como por ejemplo, toma de radiografías, rayos X, etc. Emplear la estrategia centrífuga

SEGUNDA UNIDAD

SEMANA 07: EL MODELO RELACIONAL

33. Modelo relacional

En 1970, el modo en que se veían las bases de datos cambió por completo cuando E. F. Codd introdujo el modelo relacional. En aquellos momentos, el enfoque existente para la estructura de las bases de datos utilizaba punteros físicos (direcciones de disco) para relacionar registros de distintos ficheros. Si, por ejemplo, se quería relacionar un registro con un registro, se debía añadir al registro un campo conteniendo la dirección en disco del registro. Este campo añadido, un puntero físico, siempre señalaría desde el registro al registro. Codd demostró que estas bases de datos limitaban en gran medida los tipos de operaciones que los usuarios podían realizar sobre los datos. Además, estas bases de datos eran muy vulnerables a cambios en el entorno físico. Si se añadían los controladores de un nuevo disco al sistema y los datos se movían de una localización física a otra, se requería una conversión de los ficheros de datos. Estos sistemas se basaban en el modelo de red y el modelo jerárquico, los dos modelos lógicos que constituyeron la primera generación de los SGBD.

El modelo relacional representa la segunda generación de los SGBD. En él, todos los datos están estructurados a nivel lógico como tablas formadas por filas y columnas, aunque a nivel físico pueden tener una estructura completamente distinta. Un punto fuerte del modelo relacional es la sencillez de su estructura lógica. Pero detrás de esa simple estructura hay un fundamento teórico importante del que carecen los SGBD de la primera generación, lo que constituye otro punto a su favor.

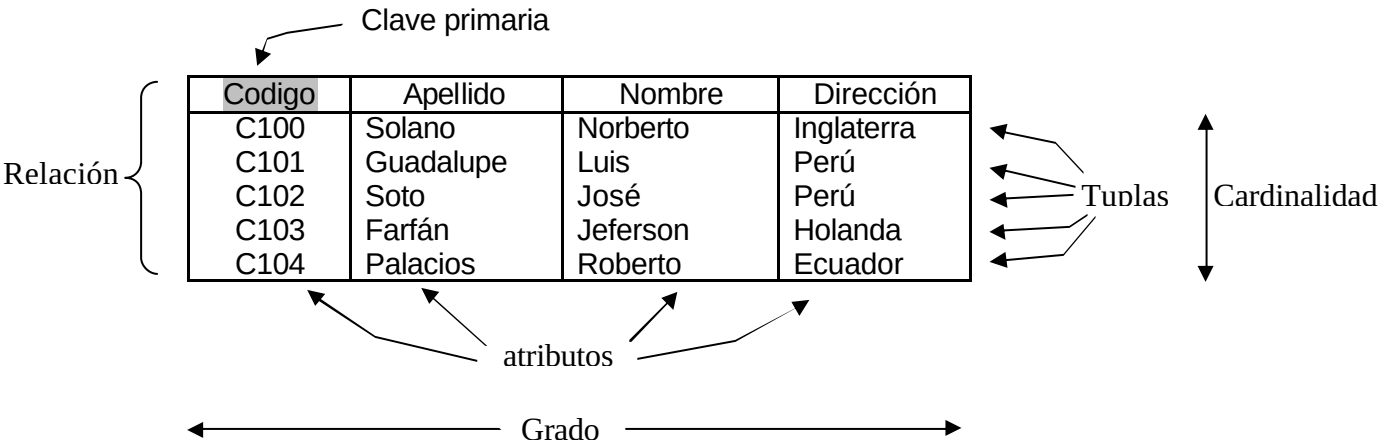
Dada la popularidad del modelo relacional, muchos sistemas de la primera generación se han modificado para proporcionar una interfaz de usuario relacional, con independencia del modelo lógico que soportan (de red o jerárquico).

El modelo relacional, como todo modelo de datos, tiene que ver con tres aspectos de los datos:

a. Estructura de datos.
b. Integridad de datos.
c. Manipulación de los datos.

1. Estructura de datos relacional

Los términos estructurales más importantes se muestran a continuación:



a) Relaciones

El modelo relacional se basa en el concepto matemático de relación, que gráficamente se representa mediante una tabla. Codd, que era un experto matemático, utilizó una terminología perteneciente a las matemáticas, en concreto de la teoría de conjuntos y de la lógica de predicados.

Una relación es una tabla con columnas y filas. Un SGBD sólo necesita que el usuario pueda percibir la base de datos como un conjunto de tablas. Esta percepción sólo se aplica a la estructura lógica de la base de datos (en el nivel externo y conceptual de la arquitectura de tres niveles ANSI-SPARC). No se aplica a la estructura física de la base de datos, que se puede implementar con distintas estructuras de almacenamiento.

b) ***Un atributo*** es el nombre de una columna de una relación. En el modelo relacional, las relaciones se utilizan para almacenar información sobre los objetos que se representan en la base de datos. Una relación se representa gráficamente como una tabla bidimensional en la que las filas corresponden a registros individuales y las columnas corresponden a los campos o atributos de esos registros. Los atributos pueden aparecer en la relación en cualquier orden.

c) ***Un dominio*** es el conjunto de valores legales de uno o varios atributos. Los dominios constituyen una poderosa característica del modelo relacional. Cada atributo de una base de datos relacional se define sobre un dominio, pudiendo haber varios atributos definidos sobre el mismo dominio. El dominio responde a la idea del tipo de dato.

El concepto de dominio es importante porque permite que el usuario defina, en un lugar común, el significado y la fuente de los valores que los atributos pueden tomar. Esto hace que haya más información disponible para el sistema cuando éste va a ejecutar una operación relacional, de modo que las operaciones que son semánticamente incorrectas, se pueden evitar. Por ejemplo, no tiene sentido comparar el nombre de una calle con un número de teléfono, aunque los dos atributos sean cadenas de caracteres. Sin embargo, el importe mensual del alquiler de un inmueble no estará definido sobre el mismo dominio que el número de meses que dura el alquiler, pero sí tiene sentido multiplicar los valores de ambos dominios para averiguar el importe total al que asciende el alquiler. Los SGBD relacionales no ofrecen un soporte completo de los dominios ya que su implementación es extremadamente compleja.

d) ***Una tupla*** es una fila de una relación. Los elementos de una relación son las tuplas o filas de la tabla. En la relación FUTBOLISTA, cada tupla tiene seis valores, uno para cada atributo. Las tuplas de una relación no siguen ningún orden.

e) ***El grado*** de una relación es el número de atributos que contiene. El grado de una relación no cambia con frecuencia.

f) ***La cardinalidad*** de una relación es el número de tuplas que contiene. Ya que en las relaciones se van insertando y borrando tuplas a menudo, la cardinalidad de las mismas varía constantemente.

Una base de datos relacional es un conjunto de relaciones normalizadas.

g) Propiedades de las relaciones

Las relaciones tienen las siguientes características:

- Cada tupla es distinta de las demás: no hay tuplas duplicadas.
- El orden de las tuplas no importa: las tuplas no están ordenadas.
- El orden de los atributos no importa: los atributos no están ordenados.
- Los valores de los atributos son atómicos: en cada tupla, cada atributo toma un solo valor. Se dice que las relaciones están normalizadas.

- Cada relación tiene un nombre y éste es distinto del nombre de todas las demás.
- No hay dos atributos que se llamen igual.

h) Tipos de relaciones

En un SGBD relacional pueden existir varios tipos de relaciones, aunque no todos manejan todos los tipos.

- ✓ Relaciones base. Son relaciones reales que tienen nombre y forman parte directa de la base de datos almacenada (son autónomas).
- ✓ Vistas. También denominadas relaciones virtuales, son relaciones con nombre y derivadas: se representan mediante su definición en términos de otras relaciones con nombre, no poseen datos almacenados propios.
- ✓ Instantáneas. Son relaciones con nombre y derivadas. Pero a diferencia de las vistas, son reales, no virtuales: están representadas no sólo por su definición en términos de otras relaciones con nombre, sino también por sus propios datos almacenados. Son relaciones de sólo de lectura y se refrescan periódicamente.
- ✓ Resultados de consultas. Son las relaciones resultantes de alguna consulta especificada. Pueden o no tener nombre y no persisten en la base de datos.
- ✓ Resultados intermedios. Son las relaciones que contienen los resultados de las subconsultas. Normalmente no tienen nombre y tampoco persisten en la base de datos.
- ✓ Resultados temporales. Son relaciones con nombre, similares a las relaciones base o a las instantáneas, pero la diferencia es que se destruyen automáticamente en algún momento apropiado.

i) Claves

Ya que en una relación no hay tuplas repetidas, éstas se pueden distinguir unas de otras, es decir, se pueden identificar de modo único. La forma de identificarlas es mediante los valores de sus atributos.

Una superclave es un atributo o un conjunto de atributos que identifican de modo único las tuplas de una relación.

Una clave candidata es una superclave en la que ninguno de sus subconjuntos es una superclave de la relación. El atributo o conjunto de atributos de la relación es una clave candidata para si y sólo si satisface las siguientes propiedades:

- ✓ Unicidad: nunca hay dos tuplas en la relación con el mismo valor de Irreductibilidad

- ✓ Minimalidad: ningún subconjunto de tiene la propiedad de unicidad, es decir, no se pueden eliminar componentes de sin destruir la unicidad.

Cuando una clave candidata está formada por más de un atributo, se dice que es una clave compuesta. Una relación puede tener varias claves candidatas.

Para identificar las claves candidatas de una relación no hay que fijarse en un estado o instancia de la base de datos. El hecho de que en un momento dado no haya duplicados para un atributo o conjunto de atributos, no garantiza que los duplicados no sean posibles. Sin embargo, la presencia de duplicados en un estado de la base de datos sí es útil para demostrar que cierta combinación de atributos no es una clave candidata. El único modo de identificar las claves candidatas es conociendo el significado real de los atributos, ya que esto permite saber si es posible que aparezcan duplicados. Sólo usando esta información semántica se puede saber con certeza si un conjunto de atributos forman una clave candidata.

La clave primaria de una relación es aquella clave candidata que se escoge para identificar sus tuplas de modo único. Ya que una relación no tiene tuplas duplicadas, siempre hay una clave candidata y, por lo tanto, la relación siempre tiene clave primaria. En el peor caso, la clave primaria estará formada por todos los atributos de la relación, pero normalmente habrá un pequeño subconjunto de los atributos que haga esta función.

Las claves candidatas que no son escogidas como clave primaria son denominadas claves alternativas.

Una clave ajena es un atributo o un conjunto de atributos de una relación cuyos valores coinciden con los valores de la clave primaria de alguna otra relación

SEMANA 08: RESTRICCIONES DE INTEGRIDAD

EL termino **integridad** se refiere a la exactitud o corrección de los datos en la base de datos. Cuando los contenidos de una base de datos se modifican con sentencias INSERT, DELETE o UPDATE, la integridad de los datos almacenados puede perderse de muchas maneras diferentes.
Pueden añadirse datos no válidos a la base de datos, tales como un pedido que especifica un producto no existente.
Pueden modificarse datos existentes tomando un valor incorrecto, como por ejemplo si se reasigna un vendedor a una oficina no existente.
Los cambios en la base de datos pueden perderse debido a un error del sistema o a un fallo en el suministro de energía.
Los cambios pueden ser aplicados parcialmente, como por ejemplo si se añade un pedido de un producto sin ajustar la cantidad disponible para vender.
Una de las funciones importantes de un DBMS relacional es preservar la integridad de sus datos almacenados en la mayor medida posible.

34. DEFINICIÓN DE RESTRICCIONES DE INTEGRIDAD

Base de Datos Relacionales.- una base de datos relacional es una base de datos percibida por el usuario como una colección de relaciones normalizadas de diversos grados que varía con el tiempo.

Las reglas de integridad relacional, se ajustan precisamente al Modelo relacional, la cual incluye dos reglas generales de integridad, las cuales se refieren a las claves primarias y a las claves foráneas.

Claves Primarias.- es el identificador único para una relación (tabla). Se debe tener en cuenta que una clave cualquiera debe cumplir con dos propiedades mínimas:

a. Unicidad.- En cualquier momento dado, no existen dos tuplas (registros) en la relación con el mismo valor de clave primaria
b. Minimimalidad.- Si la clave primaria es compuesta, no será posible eliminar ningún componente de la clave primaria sin destruir la propiedad de unicidad.

Claves Ajenas.- es un atributo (quizá compuesto) de una relación determinada, supongamos R2, cuyos valores deben concordar con los de la clave primaria de alguna relación R1 (R1 y R2 no necesariamente deben ser distintos).

a) Regla de Integridad de las Entidades

"Ningún componente de la clave primaria de una relación base puede aceptar nulos"

b) Regla de integridad referencial (para Claves Ajenas)

"La base de datos no debe contener valores de clave ajena sin concordancia"

Aquí se hace énfasis, principalmente, en la integridad referencial, como un sistema de reglas usadas para garantizar que las relaciones entre los registros de tablas relacionadas sean válidas, y que no se eliminen ni modifiquen accidentalmente datos relacionados.

Se puede establecer la integridad referencial cuando se cumplen todas las condiciones siguientes:

- El campo coincidente de la tabla principal, es una clave primaria o una clave candidata, es decir sus valores deben ser únicos en toda la tabla.
- Los atributos relacionados: el llamado clave ajena, en la tabla relacionada; y la llamada clave primaria, en la tabla principal, ambos tienen el mismo tipo de datos.

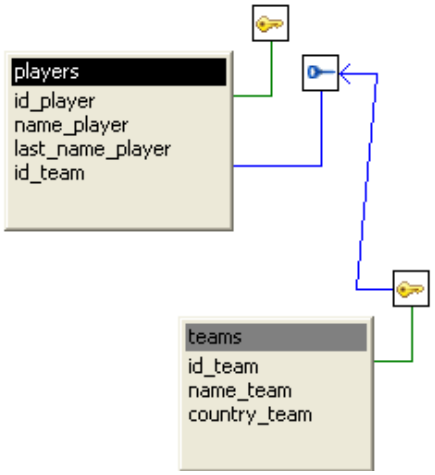
Cuando se habla de integridad referencial, se pueden considerar las siguientes reglas:

a) No se puede introducir un valor de clave ajena de la tabla relacionada que no exista en la clave principal de la tabla principal. No obstante, se puede introducir un valor Nulo en la clave ajena, especificando que los registros no están relacionados.
b) No puede cambiar un valor de clave principal en la tabla principal si ese registro tiene registros relacionados. *Por ejemplo*, no puede cambiar la clave de un currículo en la tabla CURRICULA, si existen alumnos relacionados a ese currículo en la tabla ALUMNO.
c) No se puede eliminar un registro de una tabla principal si existen registros coincidentes en una tabla relacionada. *Por ejemplo*, no se puede eliminar un registro de Alumnos de la tabla ALUMNO si existen notas de convalidaciones de ese alumno en la tabla CONVALIDACIONES.

Para hacer práctica la aplicación de la integridad referencial, consideramos 4 alternativas:

1. Exigir integridad referencial. Verifica que se cumplan las tres reglas mencionadas anteriormente.
2. Actualizar en cascada. Verifica que se cumplan las reglas a y c, obviando la regla b. De esta forma, si se cambia un valor de la clave principal, entonces, automáticamente se cambian los valores en la clave ajena de la tabla relacionada. Por ejemplo, al cambiar la clave de un currículo en la tabla CURRICULA, se cambia automáticamente el campo Cod_Curr en los alumnos relacionados de la tabla ALUMNO.
3. Eliminar en cascada. Verifica que se cumplan las reglas a y b, obviando la regla c. Así, si se elimina un registro en la tabla principal, entonces, automáticamente se eliminan los registros coincidentes en la tabla relacionada. Por ejemplo, al eliminar un curso en la tabla CURSO_CURRICÜLA, se eliminan automáticamente los registros relacionados que indican cuales son sus pre-requisitos, en la tabla PRE-REQUISITO.
4. Actualizar y eliminar en cascada. Verifica que se cumpla la regla a, obviando las reglas b y c. Así, si se actualiza o elimina un registro en la tabla principal, entonces automáticamente, se actualizan o eliminan los registros coincidentes en la tabla relacionada.

35.Práctica de Laboratorio.- Reglas de Integridad Referencial



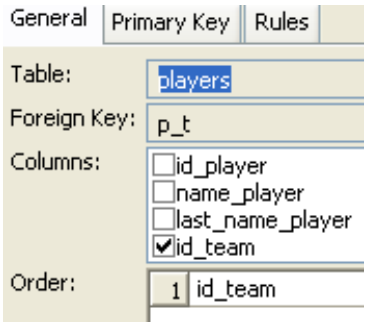
Columns (pract_ri) - players

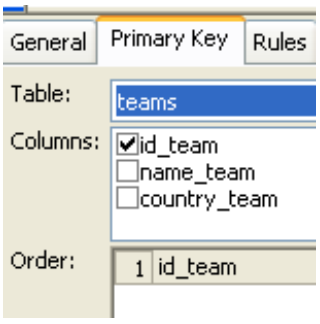
Column Name	Data Type	Width	Dec	Null	Default
id_player	char	5		No	autoincrement
name_player	char	10		No	(None)
last_name_player	char	10		No	(None)
id_team	char	5		Yes	(None)

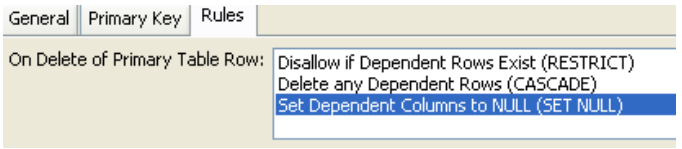
Columns (pract_ri) - teams

Column Name	Data Type	Width	Dec	Null	Default
id_team	char	5		No	(None)
name_team	char	10		No	(None)
country_team	char	10		No	(None)

Propiedades PK-FK







Results

Id Player	Name Player	Last Name Player	Id Team
0100	Nolberto	Solano	0100
0200	Claudio	Pizarro	0200
0300	Paolo	Guerrero	0200
0400	Jefferson	Farfan	0300

Results

Id Team	Name Team	Country Team
0100	NewCasttle	Inglaterra
0200	Bayern	Alemania
0300	PSV	Holanda

Utilizando Adaptative Server Anuwhere 9.0 de Sybase en el entorno de PowerBuilder 10.5, crear las tablas correspondientes al siguiente esquema de base de datos relacional, relacionando la gestión de los préstamos de una biblioteca:

Libro (asignatura, autor, titulo, editor, clase)
Usuario (carnet, nombre, direccion)
Clase (clase, tiempo_de_prestamo)
Prestamo (asignatura, carnet, fecha_inicio, fecha_fin)

Con las siguientes claves ajenas:
Libro.clase ? Clase
Prestamo.asignatura ? Libro
Prestamo.carnet ? Usuario

Utilice las restricciones de integridad referencial: **RESTRICT**, **CASCADE**, **SET NULL** adecuadamente.

Poblar las tablas con algunas líneas de datos.

Probar el funcionamiento de las restricciones empleadas.

- Agregar a la tabla Usuario el campo Fecha_Ingreso (que sea obligatorio)
- Agregar a la tabla Libro el campo Prestado (que sea obligatorio), y asignarle por defecto el valor 1.
- Al finalizar borrar las tablas creadas.

SEMANA 09: MEJORA DE LA CALIDAD DE LOS ESQUEMAS DE BASES DE DATOS. NORMALIZACIÓN

36. TEORÍA DE LA NORMALIZACIÓN

Cuando se diseña una base de datos mediante el modelo relacional, tenemos distintas alternativas para obtener diferentes esquemas relaciónales (estructuras de tablas), y no todos son equivalentes, ya que algunos van a representar la realidad mejor que otros.

Es necesario conocer qué propiedades debe tener un esquema relacional para representar adecuadamente una realidad y cuáles son los problemas que se pueden derivar de un diseño inadecuado.

La Teoría de la Normalización es un método que se aplica en el diseño de bases de datos relacionales. Las formas normales (primera, segunda, tercera, cuarta, de Boyce Codd) pretenden mantener la estructura lógica de los datos en una forma limpia, "purificada", mitigando los problemas de anomalías de inserción, borrado y actualización que ocasionan trabajo innecesario (porque deben aplicarse los mismos cambios en varios lugares), así como el problema de pérdida accidental de datos, o la dificultad de representación de determinados hechos.

Algunos problemas que se pueden presentar son:

- Incapacidad para almacenar ciertos hechos
- Redundancias y por tanto, posibilidad de incoherencias
- Ambigüedades
- Pérdida de información
- Pérdida de dependencias funcionales, es decir, ciertas restricciones de integridad que dan lugar a interdependencias entre los datos.
- Aparición en la BD de estados no válidos, es decir, anomalías de inserción, borrado y modificación.

Así, el esquema relacional obtenido debe ser analizado para comprobar que no presenta los problemas anteriores. Analicemos la siguiente tabla: ESCRIBE, donde se guardan los datos de autores y libros.

AUTOR	NACIONALIDAD	COD_LIBRO	TITULO	EDITORIAL	AÑO
Date, C.	Norteamericana	98987	Database	Addison	1990
Date, C.	Norteamericana	97777	SQL	Addison, W.	1986
Date, C.	Norteamericana	98967	Guide for DB	Addison, W.	1988
Codd.E.	Norteamericana	7890	Relational DB	Addison. W.	1990
Gardarin	Francesa	12345	Basi Dati	Paraninfo	1986
Gardarin	Francesa	67890	Comp BD	Eyrolles	1984
Valduriez	Francesa	67890	Comp BD	Eyrolles	1984
Kim.W.	Norteamericana	11223	BDOO	ACM	1989
Lochovsky	Canadiense	11223	BDOO	ACM	1989

Algunos problemas con esta tabla (relación) son:

- Redundancia, ya que la nacionalidad del autor se repite por cada ocurrencia del mismo. Lo mismo sucede cuando un libro tiene más de un autor, se repite la editorial y el año de publicación.
- Anomalías de modificación, es fácil cambiar el nombre de una editorial en una tupla (fila) sin modificar el resto de las que corresponden al mismo libro, lo que da lugar a incoherencias.

- Anomalías de inserción, ya que si queremos ingresar información de algún autor, del que no hubiera ningún libro en la base dalos, no sería posible, porque el cod_libro es parte de la clave primaria de la relación. La inserción de un libro, que tiene dos autores obliga a insertar dos tuplas en la relación.
- Anomalías de borrado, ya que si queremos eliminar un cierto libro, perderíamos los datos de su autor y viceversa.

En los casos anteriores, se deja en manos del usuario manejar la integridad de la base de datos.
Los problemas descritos suceden pues no se cumple un hecho básico de todo diseño:

"hechos distintos, deben almacenarse en objetos distintos"

Una forma de evitar estos problemas consiste en seguir la metodología propuesta en el curso, es decir, un riguroso diseño conceptual y un traspaso de éste al modelo relacional. Sin embargo, ante posibles dudas respecto a que un esquema relacional esté correcto, aplicaremos a dicho esquema un método formal de análisis, que permita analizar errores y generar esquemas correctos. Esta es la Teoría de la Normalización.

Para el ejemplo anterior, el siguiente conjunto de relaciones no presenta estos problemas:
LIBRO (cod libro, titulo, editorial, año)
AUTOR (nombre, nacionalidad)
ESCRIBE (cod libro, nombre)

NOTA: En realidad, siguiendo un adecuado modelamiento conceptual se tendría el siguiente esquema:

Y a partir de este esquema se generan las 3 tablas indicadas, es decir, indirectamente se han evitado los problemas.

La normalización introduce una técnica formal para diseñar bases de datos relaciónales, y permite mecanizar parte del proceso al disponer de algoritmos de normalización.
Una observación importante, es que las anomalías antes descritas se producen en procesos de actualización y no en procesos de consulta.
La normalización penaliza las consultas, al disminuir la eficiencia (las hace más lentas), ya que aumenta el número de relaciones (tablas) presentes en la base de datos, por lo que una determinada consulta puede llevar consigo el acceso a varias tablas.

37. NOCIÓN INTUITIVA DE LAS FORMAS NORMALES

La Normalización tiene como objetivo obtener esquemas relacionales que cumplan determinadas condiciones, a través de las formas normales.

- ✓ **PRIMERA FORMA NORMAL (1FN)**, fue introducida por Codd, en su primer trabajo. Es una restricción inherente al modelo relacional por lo que su cumplimiento es obligatorio. Consiste en la prohibición de que en una relación (tabla) existan grupos repetitivos, es decir, un atributo no puede tomar más de un valor del dominio.

- ✓ **SEGUNDA FORMA NORMAL (2FN)**, también introducida por Codd. Una relación está en 2FN, si además de estar en 1FN, todos los atributos que no forman parte de ninguna clave candidata constituyen información acerca de la clave completa, y no sólo de una parte de la clave.
  Para la relación:
  PRESTAMO (num_socio, nombre_socio, cod_libro, fec_prest, editorial, país),
  Cuyas claves candidatas son:
  (num_socio, cod_libro) y
  (nombre_socio, cod_libro)
  Se puede observar que ciertos atributos que no forman parte de las claves candidatas, tal como editorial, constituye información acerca del libro, pero no acerca de la clave completa. Luego, la relación préstamo no se encuentra en **2FN**.
  La solución es descomponer esta relación en las siguientes:
  PRESTAMO1(num_socio, nombre_socio, codJibro, fec_prest)
  LIBRO (cod_libro, editorial, país)

  En la relación **PRESTAMO1**, el único atributo que no forma parte de las claves candidatas es fec_prest, y constituye información acerca de la clave completa, así, la tabla está en **2FN**.
  En la relación **LIBRO**, la clave es cod_libro, y los dos atributos: editorial y país constituyen información de la clave completa, por lo tanto, está en **2FN**.
  Observación: Una relación cuya clave está formada por un único atributo, esta en 2 FN.

- ✓ **TERCERA FORMA NORMAL (3FN)**, propuesta por Codd. Una relación está en **3FN**, si además de estar en **2FN**, los atributos que no forman parte de ninguna clave candidata constituyen información sólo de la(s) clave(s) y no de otros atributos.
  En la relación **PRESTAMO1**, el atributo fec_prest constituye información de las claves, y ya que no existen más atributos, entonces, está en 3FN.
  En la relación **LIBRO**, el atributo país constituye información acerca de la editorial que publica el libro (es decir de un atributo que no es clave), por lo que no está en 3FN.
  La solución es descomponer la relación LIBRO en:
  LIBR01 (cod_libro, editorial)
  EDITORIAL (editorial, país),
  Que están en 3FN, ya que todo atributo no clave constituye información sólo acerca de la clave.

- ✓ **FORMA NORMAL DE BOYCCE Y CODD (FNBC)**. La relación **PRESTAMO1**, que está en **3FN**, todavía presenta anomalías, ya que num_socio y nombre_socio, se repiten innecesariamente por cada cod_libro. Una relación está en **FNBC** si y solo si, las claves candidatas son los únicos descriptores sobre los que se facilita información por cualquier otro atributo.
  En la relación **PRESTAMO1**, num_socio es información acerca de nombre_socio y viceversa. Ninguno de estos atributos son clave (aunque formen parte de la clave), entonces no se cumple con la **FNBC**. Para solucionarlo descomponemos la relación en:
  SOCIO (num_socio, nombre_socio)
  PRESTAMO2 (num_socio, cod_libro, fec_prest), que si están en **FNBC**.

Hasta ahora nuestro esquema relacional está compuesto por las siguientes relaciones en FNBC:
LIBRO1 (cod_libro. editorial)
EDITORIAL (editorial, país)
SOCIO (num_socio, nombre_socio)
PRESTAMO2(num_socio, codJibro, fec_prest)

NOTA: Con un adecuado modelamiento conceptual se tendría el siguiente esquema:

Mas adelante en el curso, se verá que a partir de este esquema se obtienen (as mismas tablas indicadas. Es decir, un correcto modelamiento asegura, según la metodología a seguir en el curso, obtener tablas normalizadas.

De una manera más formal, la Teoría de la Normalización se basa en restricciones definidas sobre los atributos de una relación, que son conocidas como dependencias. Existen varios tipos de dependencias:

- Funcionales, relacionadas con la 2FN y 3FN y FNBC
- Multivaluadas, relacionadas con la 4FN
- De proyección o combinación, relacionadas con la 5FN.

38. DEPENDENCIAS FUNCIONALES

Sea el esquema de relación R definido sobre el conjunto de atributos A y sean X e Y subconjuntos de A llamados descriptores. Se dice que Y depende funcionalmente de X o que X determina o implica a Y, que se representa por X ? Y, si solo si, cada valor de X tiene asociado en todo momento un único valor de Y.

Ejemplo: cod_libro ? titulo, el código del libro determina el titulo. El cod_libro es el implicante y titulo es el implicado. Siempre el implicado es un hecho (una información) acerca del implicante.

- **OBS1:** La afirmación de que cod_libro determina titulo NO significa que a partir de cod_libro podamos conocer el titulo. Es decir, para un esquema R, si tenemos la dependencia funcional X ? Y, dado un valor de X no podemos en general conocer el valor de Y. Solo nos limitaremos a firmar que para dos tuplas de cualquier extensión de R que tengan el mismo valor de X, el valor de Y también será igual en ambas.
- **OBS2:** Las dependencias son predicados o restricciones sobre cualquier extensión válida del esquema de relación, por lo que observar una determinada extensión (datos) no puede llevamos a afirmar la existencia de una dependencia funcional.

22.1. Dependencia funcional completa

Si el descriptor X es compuesto, es decir, X(X1, X2), se dice que Y tiene dependencia funcional completa de X, si depende funcionalmente de X, pero no depende de ningún subconjunto del mismo, esto es:
X? Y
No: X1? Y
No: X2? Y.
Se representa por $X \Rightarrow Y$.

Ejemplo: La tabla PUBLICA (articulo, revista, numero, pagina), que representa la pagina inicial en la que comienza un articulo en una revista. Un mismo artículo puede aparecer publicado en distintas revistas y en cada una de ellas, en páginas distintas y una revista publica varios artículos, se tiene:
Articulo, revista, numero ? página
No: articulo ? página
No: revista ? pagina
No: numero ? página
Por lo tanto: articulo, revista, numero $\Rightarrow$ página

22.2. Dependencia funcional trivial

Una dependencia funcional X ? Y es trivial si Y es un subconjunto de X ($Y \subset X$). *Ejemplos:* cod_libro ? cod_libro y
Articulo, revista ? revista.

22.3. Dependencia funcional elemental

Solo las dependencias elementales son útiles en la normalización. Una dependencia funcional X ? Y es elemental si Y es un atributo único, no incluido en X y no existe X' incluido en X tal que X' ? Y, es decir, la dependencia funcional elemental es una dependencia funcional completa no trivial, en la que el implicado es un atributo único.

22.4. Dependencia funcional transitiva

Sea la relación R(X, Y, Z), en la que existen las siguientes dependencias funcionales:
X ? Y y Y ? Z, se dice que Z tiene dependencia transitiva respecto a X, a través de Y, ya que podemos inferir: X ? Z
Ejemplo: LIBRO_ED (codlibro, editorial, país)
La dependencia entre codlibro y país es transitiva, a través de editorial. Intuitivamente se interpreta como que PAÍS es una información del libro, pero indirectamente, ya que es una información de EDITORIAL y esta a su vez de LIBRO.

39. DEFINICIÓN FORMAL DE LA TRES PRIMERAS FORMAS NORMALES

39.1. Primera Forma Normal: equivalente a la definición dada antes

39.2. Segunda Forma Normal

- Está en 1FN
- Cada atributo no principal tiene dependencia funcional completa respecto de cada una de las claves.

  Ejemplo: La relación PUBLICA2 (artículo, revista, número, página, editorial) que refleja en qué numero de qué revista se publica un artículo, en qué pagina comienza y cuál es la editorial. Tenemos las siguientes dependencias:
  Articulo, revista, numero ? página
  Revista ? editorial
  Clave:(articulo, revista, numero)
  Esta relación no está en 2FN, ya que editorial depende de la revista y tiene redundancia, pues se repite la editorial para cada artículo que se publica en una revista.

39.3. Tercera Forma Normal

- Está en **2FN**
- Ningún atributo no principal depende transitivamente de ninguna clave de la relación

Ejercicio:

R (estudiante, nro_matricula, curso, centro, profesor, texto),

Con las restricciones:

- Un estudiante puede estar matriculado en varios cursos
- Un estudiante tiene un numero de matrícula distinto para cada curso en el que está matriculado
- Un curso se imparte en un solo centro
- El número de matrícula identifica al centro en el que se imparte el curso y al curso mismo
- Un curso es impartido por un solo profesor, pero un profesor puede impartir varios cursos
- Un curso se apoya en distintos textos y un mismo texto puede servir de soporte a varios cursos.

Reducir el esquema anterior a un conjunto equivalente de relaciones en **FNBC**

40. ORGANIZACIÓN DE RELACIONES

- Estructuración (consideraciones lógicas)
  - Normalización
  - Particionamiento horizontal
- Reestructuración (consideraciones físicas)
  - Desnormalización
  - Particionamiento (horizontal, vertical)

Particionamiento Horizontal de relaciones

Esta estructuración permite eliminar valores nulos, debido en general a no haberse detectado los subtipos de una entidad o haberlas reunido en una sola entidad.

Por ejemplo, en DOCUMENTO (cod-doc, titulo, idioma, editorial), que almacena datos de libros y artículos. El atributo editorial es inaplicable a artículo, podríamos descomponer la relación en:

LIBRO (cod-doc, titulo, idioma, editorial)
ARTICULO (cod-doc, titulo, idioma)

La relación origen pasa por selección a una que tiene todos los atributos de la original, pero contiene valores conocidos junto con otra a través de una selección que contiene solo los atributos no nulos, eliminando el atributo de la relación origen que tenía nulos. La construcción de la relación original se realiza por medio de la unión relacional, después de añadir los atributos para que sean compatibles en la unión.

Desnormalizacion y Particionamiento

Son métodos o formas de organizar las relaciones, teniendo en cuenta razones de tipo físico:

- Tasa de actualizaciones versus consultas
- Las veces que se accede conjuntamente a los atributos
- El tamaño en bytes de los atributos
- Tipos de proceso (en línea-batch)
- Prioridad de los procesos
- Tamaño de las tablas.

41. RESUMEN NORMALIZACION

La normalización de datos es un procedimiento que asegura que un modelo de datos se ajusta a ciertos estándares definidos para minimizar la redundancia de datos, proporcionar flexibilidad para soportar requisitos funcionales y permitir que el modelo se diseñe sobre una amplia variedad de diseños alternativos de bases de datos. Para ello emplearemos los conceptos de formas normales.

Dependencia Funcional (DF).- Dada una relación R, el atributo Y de R depende funcionalmente del atributo X de R. (R.X ^ R.Y). R.X determina funcionalmente a R.Y, si y sólo si: si un valor Y en R está asociado a cada valor de X en R. (En cualquier momento dado).

Peligros y Anomalías en el Diseño de Bases de datos Relaciónales

Entre las propiedades indeseables que un mal diseño puede tener están:

- ✓ Repetición de la información
- ✓ Incapacidad para representar cierta información.
- ✓ Pérdida de información. Lo que puede generar las siguientes anomalías: de inserción, de eliminación, de actualización.

El proceso de normalización es una progresiva detección y eliminación de esas dependencias indeseadas.

PRIMERA FORMA NORMAL (1FN)

Una relación está en la 1ra. forma normal sí y sólo si todos los dominios simples y subyacentes contienen sólo valores atómicos. Lo que significa que debemos eliminar loa atributos repetidos o grupos de atributos. Esta 1ra. Forma normal la logra automáticamente cualquier relación plana, sin entradas anidadas, esto significa que todos los valores de sus columnas son simples.

SEGUNDA FORMA NORMAL (2FN)

Una relación está en 2NF, si y sólo si está en 1NF y todos los atributos no clave dependen funcionalmente por completo de la clave primaria. Entonces cuando una relación está en 1NF pero no en 2NF siempre podrá reducirse a un conjunto equivalente de relaciones en 2NF, para que suceda esto su clave primaria deberá ser compuesta, de tal manera que se pueda reducir.

TERCERA FORMA NORMAL (3FN)

Una relación está en 3NF, si y sólo si está en 2NF y todos los atributos no clave dependen funcionalmente de manera no transitiva de la clave primaria. (Es decir no tienen ninguna dependencia funcional transitiva).

FORMA NORMAL BOYCE/CODD (BCNF)

La tercera forma normal no maneja de manera satisfactoria el caso de una relación en la cual:

a) Hay varia claves candidatas
b) Esas claves candidatas son compuestas
c) Las claves candidatas se traslapan (es decir pueden tener por lo menos un atributo en común).

No siempre se dan casos de combinación de las condiciones a, b y c, por lo cual la BCNF se reduce a la simple 3NF.

Una relación está en BCNF si y sólo si todo atributo del cual depende funcionalmente por completo algún otro atributo es una clave candidata.

PEDIDO N° 12345

Compro Barato S. A.

Fecha: 13/06/2005

Proveedor N° 7865

Nombre del Proveedor: "Tepresto Hermanos"

Dirección del Proveedor: Av. Los Tiburones s/n

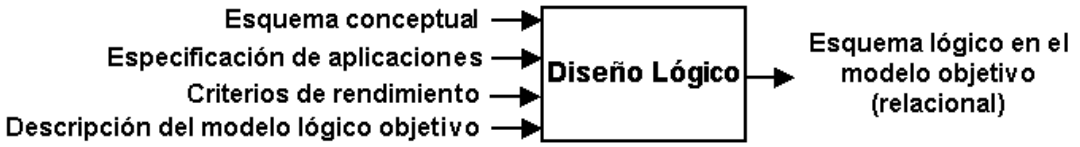
Deseamos que envíen:

Número Producto	Descripción	Precio Unitario	Cantidad	Sub Total
1324	Reproductor DVD LG	$55.00		$110.00
2143	Horno Microondas Miray	$125.00		$375.00
3241	Equipo Stereo AIWA	$380.00		$380.00
			Importe:	$865.00
			IGV:	$164.35
			TOTAL:	$1029.35

SEMANA 10: DISEÑO LÓGICO EN EL MODELO RELACIONAL. CASOS

42. DISEÑO LÓGICO (RELACIONAL)

El objetivo del diseño lógico es convertir el esquema conceptual a un esquema lógico ajustado a un sistema de gestión de base de datos específico que se tenga disponible. Las entradas y salidas del diseño lógico se muestran en el diagrama:



Los criterios de rendimiento son aquellos que miden la eficiencia de la implantación, como podrían ser el tiempo de respuesta, el almacenamiento ocupado por la base de datos, utilización de CPU, etc.

Esquema Relacional. Consiste en un conjunto de definiciones de relaciones (tablas), en el cual cada relación tiene una clave primaria.

Es necesario precisar que el término **relación**, en el modelo relacional, se refiere, en una forma sencilla, a las tablas de la base de datos (formalmente: es un. subconjunto del producto cartesiano de una lista de dominios); Por otro lado, el mismo término en el modelo ER, se refiere a la correspondencia entre dos o más entidades; Sin embargo, el contexto en que se use el término evita los malentendidos.

Al aplicar las transformaciones propias del diseño lógico, las cuales se describen más adelante, las relaciones producidas corresponden, a entidades o a relaciones, y además, mantienen la misma forma normal.

26.1. ESTRATEGIAS PARA GENERAR ESQUEMAS RELACIÓNALES A PARTIR DE ESQUEMAS ER. (CORRESPONDENCIA DEL ESQUEMA ER AL MODELO RELACIONAL)

El mapeamiento de un esquema ER a uno en el Modelo relacional (MR) trae asociadas toma de decisiones y algunas perdidas de semántica; Esto es debido a que el modelo relacional, al ser lógico, es menos expresivo. Las estrategias indicadas a continuación persiguen convertir un esquema ER en un conjunto de entidades y relaciones, tales que su correspondencia con el modelo relacional sea sencillo.

A modo de guía para la transformación de cada elemento de un esquema ER, se presentan las directrices siguientes (las cuales se pueden considerar como una metodología para el diseño lógico, pero aquí las vamos a considerar como herramientas que sustentan la metodología para el diseño lógico, que se indica más adelante):

Con el fin de facilitar la aplicación de las reglas de correspondencia, se considera una fase inicial de correspondencia entre esquemas.

Esta correspondencia, llamada "correspondencia preparatoria", consiste en:

1) Eliminación de Jerarquías de generalización.
2) Eliminación de identificadores externos.
3) Eliminación de atributos compuestos y polivalentes del esquema.

Una vez terminada la correspondencia preparatoria, seguimos ¡os siguientes pasos:

4) Transformación de cada entidad del esquema en una relación.
5) Transformación de Relaciones de uno a uno.

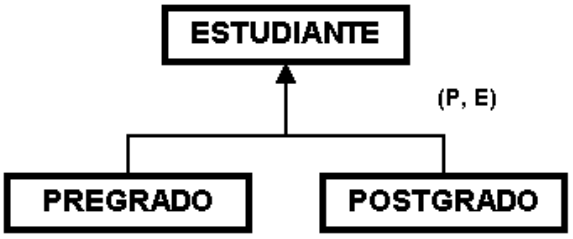
6) Transformación de Relaciones de uno a muchos.
7) Transformación de Relaciones de muchos a muchos.
8) Relaciones n_arias y recursivas.

A continuación, se mostrará en forma más detallada en que consiste cada uno de los pasos:

1) **Eliminación de Jerarquías de generalización**. La eliminación de las jerarquías de generalización se realiza para mantener en el esquema solo entidades y relaciones, lo cual es requisito para aplicar directamente las reglas de correspondencia de las etapas siguientes. Para eliminar una jerarquía generalización, se tienen las alternativas de: modelarla como una sola entidad súper conjunto, como un grupo de entidades subconjunto, o mediante relaciones; Al eliminar la generalización, se debe tomar en cuenta que las relaciones existentes antes de la eliminación, deben conservarse o transformarse de manera consistente, en el esquema resultante.
La eliminación de las generalizaciones causa una pérdida de semántica, en mayor o menor grado, dependiendo de la estrategia elegida. Un criterio importante para elegir entre una y otra alternativa es considerar el tipo de cobertura de la generalización.

Ahora, describimos las tres alternativas de donde elegir:

1.1. **Integrar la jerarquía de generalización en una sola entidad**, uniendo los atributos de las subentidades y añadiendo estos atributos a los de la superentidad. Esto se permite si la distinción entre las subentidades no es significativa desde el punto de vista de una aplicación. Más aún, se añade un atributo discriminante para indicar el caso al cual pertenece la entidad en consideración. Esta alternativa es aplicable a todas las coberturas, y tiene el problema que a veces hay que manejar demasiados valores nulos, y que las operaciones, que sólo actuaban sobre una subentidad, tendrán que buscar ahora los casos específicos dentro del conjunto completo de casos.



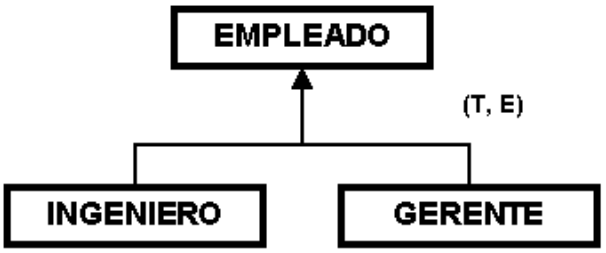
ESTUDIANTE con a tributos: número-matricula, nombre
PREGRADO con atributo: carrera
POSTGRADO con atributo: título-tesis

Esquema resultante:

ESTUDIANTE: número-matricula, nombre, carrera, título-tesis, tipo

Además, se debe verificar la exclusividad: CHECK ((tipo = 'Postgrado' AND carrera IS NULL AND titulo-tesis IS NOT NULL) OR (tipo = ´pregrado' AND carrera IS NOT NULL AND titulo-tesis IS NULL))

1.2. **Eliminar la superentidad reteniendo las subentidades**. Aquí los atributos heredados deben propagarse entre las subentidades. Esta alternativa no es práctica para generalizaciones superpuestas o parciales; sólo lo es para jerarquías totales y exclusivas. Además, si el número de atributos de la superentidad (comunes a toda las subentidades) es excesivo, su duplicación en el esquema de cada subentidad no se justifica.



EMPLEADO con atributos: LE, nombre
INGENIERO con atributo: especialidad
GERENTE con atributo: Numero _ supervisados

Esquema resultante

INGENIERO: LE, nombre, especialidad
GERENTE: LE, nombre, Numero _ supervisados

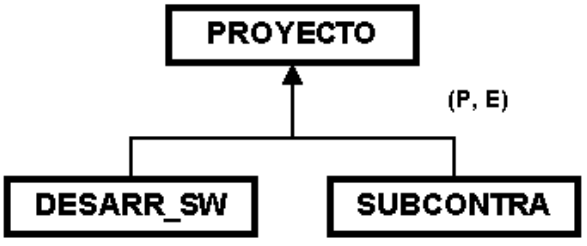
1.3. **Retener todas las entidades y establecer explícitamente las interrelaciones entre la superentidad y las subentidades.** Esta alternativa se puede considerar como la más general de las tres, ya que siempre es posible.

Las desventajas de este enfoque son:

Que el esquema resultante se complica, además de que hay una redundancia inherente al representar cada eslabón ES-UN en la jerarquía original a través de una relación explícita.

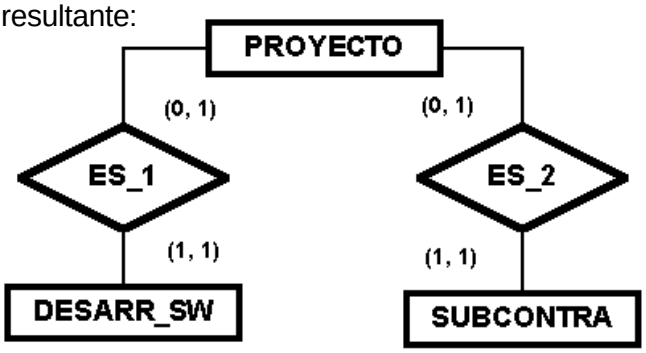
Las ventajas, por otra parte, son:

Que modela todos los casos, lo que la hace más flexible ante cambios de requerimientos, y es conveniente si la mayoría de las operaciones son estrictamente locales respecto a la superentidad o a una de las subentidades.



PROYECTO con atributos: Número-provecto, nombre-proyecto
DESARR-SW con atributo: Numero _ módulos
SUBCONTRA con atributo: contratista-principal

Esquema resultante:



Sin cambios en las entidades resultantes.

2) **Eliminación de identificadores externos**. Los identificadores externos se transforman en identificadores internos. La migración de atributos es de la entidad fuerte a la entidad débil.

3) **Eliminación de atributos compuestos y polivalentes**. El modelo relacional, en su forma básica, sólo contiene atributos simples, monovalentes, por ello los atributos compuestos y polivalentes se deben modelar en términos de atributos simples monovalentes, para ello se siguen los siguientes pasos:
   a) Eliminar el atributo compuesto considerando todos sus componentes como atributos individuales.
   b) Eliminar los componentes individuales y considerar el atributo compuesto entero como un solo atributo.

4) **Transformación de Entidades**. Se transforma cada entidad del esquema en una relación. Los atributos y la clave primaria de la entidad se convierten en atributos y clave primaria de la relación.

5) **Transformación de Relaciones de uno a uno**. Consideramos dos posibilidades:

   Integración en una relación. Esta opción tiene sentido cuando la participación de las dos entidades en la relación es total (cardinalidad (1,1)), existen dos casos;
   a. Si las dos entidades tienen la misma clave primaria, entonces se integran en una relación combinando los atributos, e incluyendo la clave primaria sólo una vez.
   b. Cuando las dos entidades tienen diferentes claves primarías, se integran en una relación y se escoge una de las dos claves primarias como clave para la relación.

   Definición de una relación aparte. Esta opción se usa cuando una de las dos entidades tiene una participación parcial, es decir su cardinalidad es (0,1).
   a. Cuando solo una de las entidades tiene participación parcial (0,1), entonces, se crea una relación adicional (serían 3 relaciones) usando las claves primarias de ambas entidades; Ambas claves primarias son claves candidatas de la nueva relación. Otra posibilidad es integrar las relaciones, escogiendo como clave primaria la clave de la entidad que tenía participación parcial

b. Sí las dos entidades tienen participación parcial, en este caso, para evitar valores nulos y representar tanto las entidades como la relación, se crea una nueva relación a partir de las claves de las entidades, y ambas claves son claves candidatas.

6) Transformación de Relaciones de uno a muchos.

a. Si una entidad tiene una participación obligatoria (1,1), entonces, la clave primaria de la entidad del lado (0, n) migra hacia la otra entidad, como atributo simple. Si la relación tiene atributos, estos también migran. De este modo, desaparece la relación (el rombo) entre ambas entidades.

b. Si una entidad tiene una participación parcial (0,1), entonces, son posibles dos soluciones:

I. Usar la solución anterior (6a), con el problema de que habrían valores nulos.

II. Crear una nueva relación con las claves primarias de las entidades, cuya clave primaria será la clave de la entidad del lado (0,1).

7) Transformación de Relaciones de muchos a muchos.

La solución no depende de la cardinalidad mínima, se crea una nueva relación teniendo como clave a la combinación de las claves primarias de las dos entidades.

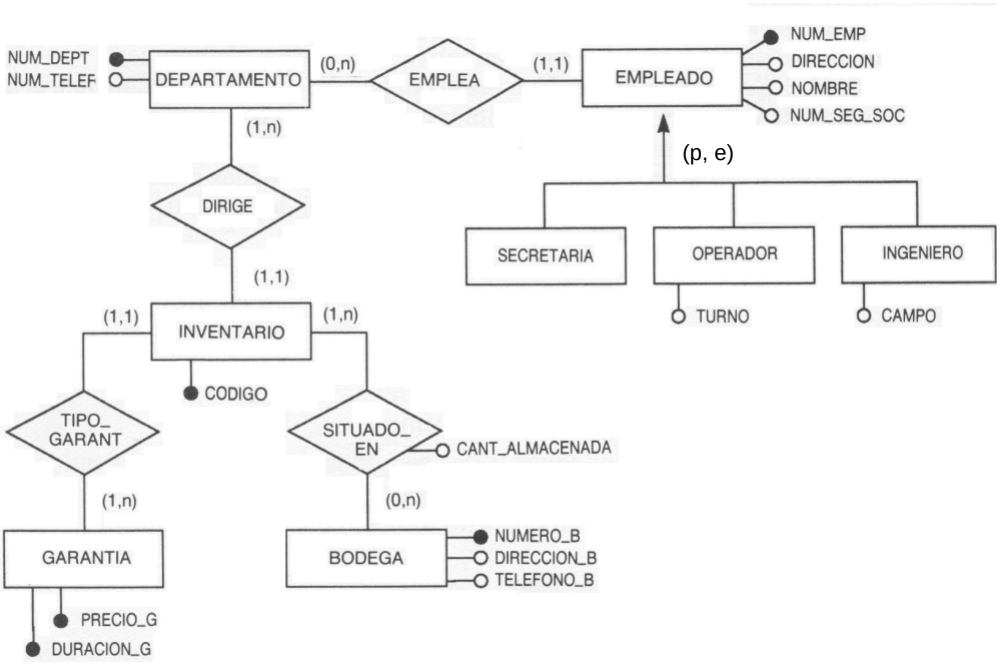
8) Relaciones n_arias y recursivas.

En el caso de relaciones n-arías, se crea una nueva relación que hereda todos los identificadores de las n entidades. En algunos casos, la clave obtenida de esta manera es mínima (la clave mínima es aquella que no contiene ninguna dependencia funcional entre los atributos), pero en otros casos no, y entonces, basta considerar como clave compuesta sólo a las claves de algunas de las entidades; La dependencia funcional juega un papel decisivo aquí.

En las relaciones recursivas, se crea una nueva relación que incluye dos veces la clave primaria de la entidad, pero ambas claves difieren en nombre, de acuerdo al papel que cumplen en la relación recursiva.

Ejercicio

Efectuar la transformación del modelo
E-R, necesarios para llegar al esquema lógico correspondiente.



43. EJERCICIOS NORMALIZACIÓN
TRANSFORMACIÓN DE ESQUEMAS ER a MODELO RELACIONAL

INSTRUCCIONES.- La presentación del trabajo se realiza con los grupos pre establecidos para todo el curso. Entrega el día del examen.

1. Se tiene el siguiente esquema:
   R(estudiante, num_matricula, curso, pabellón, profesor, texto)

   Con las restricciones:
   - ✓ Un estudiante puede estar matriculado en varios cursos
   - ✓ Un estudiante tiene un número de matricula distinto para cada curso en el que está matriculado
   - ✓ Un curso se imparte en un solo pabellón
   - ✓ El número de matricula identifica el pabellón en el que se imparte el curso y el curso mismo
   - ✓ Un curso es impartido por un solo profesor, pero un profesor puede impartir varios cursos
   - ✓ Un curso se apoya en distintos textos, y un mismo texto puede servir de soporte a varios cursos.

   Reducir el esquema anterior a un conjunto equivalente de esquemas entidad/relación, normalizando en forma adecuada.

2. Una BD debe contener información concerniente a Agentes de ventas, Áreas de ventas y Productos. Cada Agente es responsable de las ventas en una o más Áreas, cada Área tiene uno o más Agentes como responsables de las ventas de ellas. Del mismo modo, cada Agente es responsable de la venta de uno o más Productos y cada producto tiene uno o más Agentes responsables de su venta. Todos los productos se venden en todas la Áreas, pero no hay dos Agentes que vendan el mismo producto en la misma Área. Cada Agente vende el mismo conjunto de Productos en todas las Áreas en las que opera.

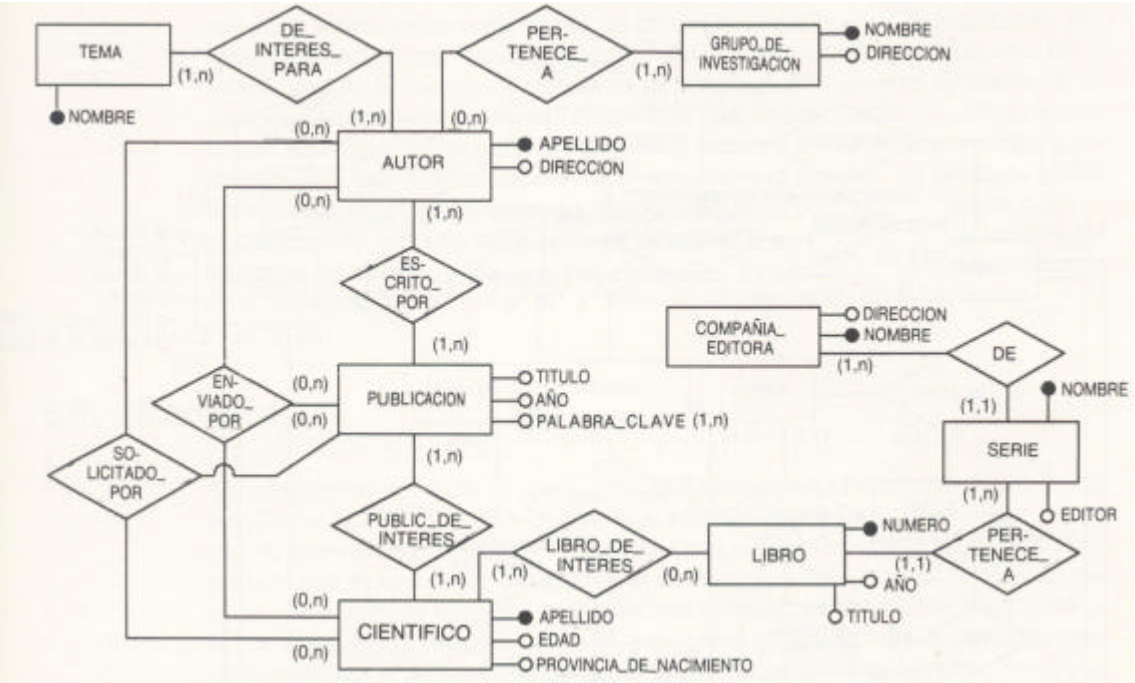
   Diseñe un conjunto de relaciones debidamente normalizadas.

3. El proceso inverso de diseño lógico se denomina Retroingeniería. Muchas veces es posible deducir aproximadamente un esquema conceptual que da origen al esquema lógico actual.
   En un deporte se tienen clubes, cada club pertenece a una sola liga departamental y cada liga departamental contiene varios clubes.
   Considerando las siguientes tablas:
   SOCIO (DNIsoc, nombresoc, fechaingreso)
   PERTENECE (DNIsoc, codclub)
   CLUB (codclub, presidenteclub, ligadepartamento, presidenteliga)

   a. Indicar en que forma normal se encuentran las tablas mostradas. Justifique
   b. Deducir el esquema conceptual que dio origen al esquema mostrado.
   c. Rediseñar adecuadamente la base de datos (esquema conceptual y lógico), añadiendo la consideración de que un socio puede ser a lo mucho presidente de una liga y a lo mucho presidente de un club.

4. Un sistema de control de pedidos de materiales utiliza una sola tabla (un pedido puede incluir varios materiales):
   PEDIDO (codpedido, codmaterial, nombrematerial, densidadmaterial, cantidad, fecha)
   a. Indique claramente los problemas de esta tabla, incluyendo los tipos de anomalías, si las hubiera (2 puntos)
   b. ¿En que forma normal se encuentra la tabla?. Justifique su respuesta (1 puntos)
   c. Rediseñe adecuadamente la base de Datos (esquema conceptual y lógico)

5. Desarrolle el diseño lógico. Obtenga los esquemas relacionales en el siguiente diagrama entidad/relación. Solo haga suposiciones coherentes para la mejora del esquema.



TERCERA UNIDAD

SEMANA 12: SQL (STRUCTURED QUERY LANGUAGE)

El Lenguaje de Consulta Estructurado (Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Reúne características del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar información de interés de una base de datos de una forma sencilla.

Orígenes.- Los orígenes del SQL están ligados a los de las bases de datos relaciónales. En 1970 Codd propone el modelo relacional y asociado a este un sublenguaje de acceso a los datos basado en el cálculo de predicados. Basándose en estas ideas los laboratorios de IBM definen el lenguaje SEQUEL (Structured English QUEry Language) que más tarde sería ampliamente implementado por el SGBD experimental System R, desarrollado en 1977 también por IBM. Sin embargo, fue Oracle quien lo introdujo por primera vez en 1979 en un programa comercial.

El SEQUEL terminaría siendo el predecesor de SQL, siendo este una versión evolucionada del primero. El SQL pasa a ser el lenguaje por excelencia de los diversos SGBD relacionales surgidos en los años siguientes y es por fin estandarizado en 1986 por el ANSI, dando lugar a la primera versión estándar de este lenguaje, el SQL-86 o SQL1. Al año siguiente este estándar es también adoptado por la ISO.

Sin embargo este primer estándar no cubre todas las necesidades de los desarrolladores e incluye funcionalidades de definición de almacenamiento que se consideraron suprimir. Así que en 1992 se lanza un nuevo estándar ampliado y revisado del SQL llamado SQL-92 o SQL2.

En la actualidad el SQL es el estándar de la inmensa mayoría de los SGBD comerciales. Y, aunque la diversidad de añadidos particulares que incluyen las distintas implementaciones comerciales del lenguaje es amplia, el soporte al estándar SQL-92 es general y muy amplio.

El ANSI SQL sufrió varias revisiones y agregados a lo largo del tiempo:

Año	Nombre	Alias	Comentarios
1986	SQL-86	SQL-87	Primera publicación hecha por ANSI. Confirmada por ISO en 1987
1989	SQL-89		Revisión menor
1992	SQL-92	SQL2	Revisión menor
1999	SQL:1999	SQL3	Se agregan expresiones regulares, consultas recursivas (para relaciones jerárquicas), triggers y algunas características orientadas a objetos.
2003	SQL:2003		Introduce algunas características de XML, cambio en las funciones, estatización de las columnas auto numéricas

El SQL proporciona una rica funcionalidad más allá de la simple consulta (o recuperación) de datos. Asume el papel de lenguaje de definición de datos (DDL - Data Definition Language), lenguaje de manipulación de datos (DML – Data Manipulation Language). Además permite la concesión y denegación de permisos, la implementación de restricciones de integridad y controles de transacción, y la alteración de esquemas.

Sentencias SQL Adaptive Server Anywhere (ASA)

SELECT	Recuperación de data
INSERT UPDATE DELETE	Lenguaje de manipulación de datos (DML)
CREATE ALTER DROP TRUNCATE	Lenguaje de definición de datos (DDL)
COMMIT ROLLBACK ROLLBACK SAVEPOINT	Control de transacciones
GRANT REVOKE	Lenguaje de control de datos (DCL)

Palabras Clave (Keywords)

Cada sentencia SQL contiene une o mas palabras clave. SQL (ASA) no es case sensitive en las palabras clave.

Identificadores

Los identificadores son nombres de objetos en la base de datos, tales como user IDs, tablas, y columnas. Tienen una longitud máxima de 128 bytes. Ellos deben estar encerrados entre comillas dobles o corchetes si cualquiera de las siguientes condiciones son ciertas:

- El identificador contiene espacios.
- El primer carácter del identificador no es un carácter alfabético (como se define luego)
- El identificador contiene una palabra reservada.
- El identificador contiene otros caracteres no alfabéticos y dígitos.

Caracteres alfabéticos incluyen el alfabeto, así como el carácter de subrayado (_), el signo (@), el signo de numero (#), y el signo dólar ($).

Los siguientes caracteres no están permitidos en identificadores:

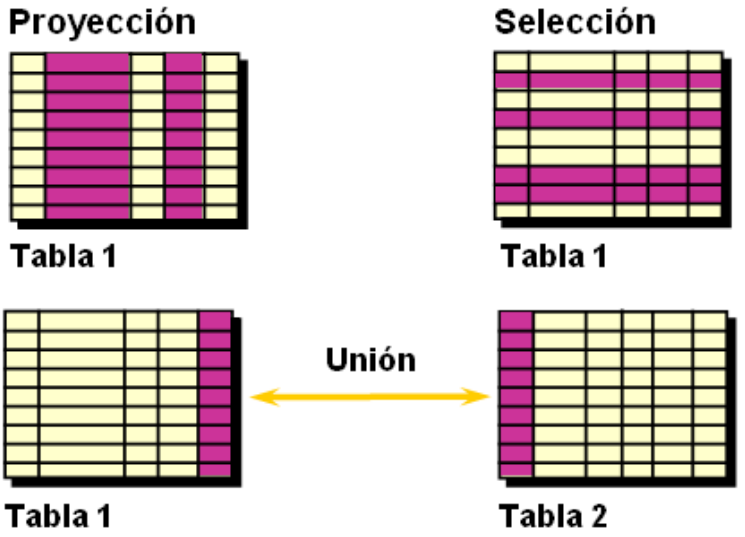
- Comillas
- Caracteres de Control
- Doble backslashes

Ejemplos.- Los siguientes son identificadores validos:

Surname, "Surname", [Surname], SomeBigName, "Client Number"

Capacidades de las Sentencias SQL SELECT

La sentencia SELECT recupera datos de una base de datos y los devuelve en forma de resultados de la consulta.



Sentencias SELECT Básicas

Sintaxis:

```
SELECT [ ALL | DISTINCT ] select-list
 [ FROM table-expression ]
```

donde:

- Cláusula SELECT especifica las columnas que se han de mostrar (select-list).
- Cláusula FROM especifica la tabla que contiene las columnas.
- ALL selecciona todas las columnas utilizando el asterisco (*)
- DISTINCT suprime duplicados.

Ejemplos:

- ✓ Selección de Todas las Columnas

```
SELECT *  FROM   departments;
```

- ✓ Selección de Columnas Específicas

```
SELECT department_id, location_id FROM   departments;
```

Operadores Aritmeticos

- **Adición.-** expresión + expresión. Sí cualquiera de las expresiones es NULL, el resultado es NULL.
- **Substracción.-** expresión - expresión. Sí cualquiera de las expresiones es NULL, el resultado es NULL.
- **Negación.-** – expresión. Sí la expresión es NULL, el resultado es NULL.
- **Multiplicación.-** expresión * expresión. Sí cualquiera de las expresiones es NULL, el resultado es NULL.
- **División.-** expresión / expresión. If Sí cualquiera de las expresiones es NULL o sí la segunda expresión es 0, el resultado es NULL.
- **Modulo.-** expresión % expresión. Encuentra el entero sobrante después de una división. Por ejemplo, 21 % 11 = 10.

Ejemplo:

- ✓ SELECT last_name, salary, salary + 300 FROM   employees;

Prioridad de los Operadores

- La multiplicación y la división tienen prioridad sobre la suma y la resta.
- Los operadores de idéntica prioridad se evalúan de izquierda a derecha.
- Los paréntesis se utilizan para forzar evaluaciones prioritarias y para clarificar sentencias.

Ejemplos:

- ✓ SELECT last_name, salary, 12*salary+100 FROM   employees;

- ✓ Uso de parentesis

```
SELECT last_name, salary, 12*(salary+100) FROM   employees;
```

Definición de un Alias de Columna

Un alias de columna:

- Cambia el nombre de una cabecera de columna.
- Resulta útil con cálculos.
- Se sitúa inmediatamente detrás del nombre de la columna, también puede existir la palabra clave opcional AS entre el nombre de la columna y el alias.
- Requiere comillas dobles si contiene espacios, caracteres especiales o si es sensible a mayúsculas/minúsculas.

Operador de Concatenación

Un operador de concatenación:

- Concatena columnas o cadenas de caracteres a otras columnas.
- Está representado por dos barras verticales (||).
- Crea una columna resultante que es una expresión de caracteres.
- expresión + expresión.- Alternativa de concatenación de cadenas. Cuando use el operador + de concatenación, ud. Debe asegurarse que los operando son datos que soporten la concatenación. Ejemplos:

SELECT 123 + 456 = 579; SELECT '123' + '456' = 123456

Cadenas de Caracteres Literales

- Un literal es un carácter, un número o una fecha incluida en la lista SELECT.
- Los valores literales de caracteres y fecha se deben escribir entre comillas simples.
- Cada cadena de caracteres tiene una salida para cada fila devuelta.

Filas Duplicadas

La visualización por defecto de las consultas son todas las filas, incluidas las filas duplicadas.

Elimine filas duplicadas mediante la palabra clave DISTINCT de la cláusula SELECT.

Limitación de las Filas Seleccionadas

- Restrinja las filas devueltas utilizando la cláusula WHERE.

```
SELECT  *|{[DISTINCT] column|expression [alias],...}
FROM    table
[WHERE  condition(s)];
```

- La cláusula WHERE sigue a la cláusula FROM.

Cadenas de Caracteres y Fechas

- Las cadenas de caracteres y los valores de fechas se escriben entre comillas simples.
- Los valores de caracteres son sensibles a mayúsculas/minúsculas y los de fecha, al formato.
- El formato de fecha por defecto es DD-MON-RR.

Operadores de Comparación

Operador	Descripción
=	Igual a
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor igual que
!=	No igual a
<>	No igual a
!>	No más grande que
!<	No menor que

Los operadores de comparación se utilizan en condiciones que comparan una expresión con otro valor o expresión. Se usan en la cláusula WHERE con el siguiente formato:

Sintaxis

... WHERE *expr operator value*

Por Ejemplo ... WHERE hire_date='01-JAN-95'; ... WHERE salary>=6000

Todas las cadenas de comparación son case no sensitive a menos que la base de datos fuese creada como case sensitive. Por defecto, base de datos Adaptive Server Anywhere son creadas como case no sensitive, mientras base de datos Adaptive Server Enterprise son creadas como case sensitive.

Otros Operadores de Comparación

Operador	Significado
BETWEEN ...AND...	Entre dos valores (ambos inclusive)
IN(set)	Coincide con cualquiera de una lista de valores
LIKE	Coincide con un patrón de caracteres. Las condiciones de búsqueda pueden contener caracteres literales o números: % indica cero o muchos caracteres. _ indica un carácter.
IS NULL	Es un valor nulo

Operadores Lógicos

Operador	Significado
AND	Devuelve TRUE si las dos condiciones componentes son verdaderas
OR	Devuelve TRUE si alguna de las condiciones componentes es verdadera
NOT	Devuelve TRUE si la condición es falsa

Tabla de Verdad de AND

La siguiente tabla muestra los resultados de combinar dos expresiones con AND:

AND	**TRUE**	**FALSE**	**NULL**
TRUE	TRUE	*FALSE*	*NULL*
FALSE	*FALSE*	FALSE	*FALSE*
NULL	*NULL*	*FALSE*	NULL

Tabla de Verdad de OR

La siguiente tabla muestra los resultados de combinar dos expresiones con OR:

OR	**TRUE**	**FALSE**	**NULL**
TRUE	TRUE	*TRUE*	*TRUE*
FALSE	*TRUE*	FALSE	*NULL*
NULL	*TRUE*	*NULL*	NULL

Tabla de Verdad de NOT

La siguiente tabla muestra el resultado de aplicar el operador NOT a una condición:

NOT	**TRUE**	**FALSE**	**NULL**
	FALSE	TRUE	NULL

Reglas de Prioridad

Las reglas de prioridad determinan el orden en el que se evalúan y se calculan las expresiones. La tabla enumera el orden de prioridad por defecto. Puede sustituir el orden por defecto escribiendo entre paréntesis las expresiones que desee calcular en primer lugar.

Orden de Evaluación	Operador
1	Operadores aritméticos
2	Operador de concatenación
3	Condiciones de comparación
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	condición lógica NOT
7	condición lógica AND
8	condición lógica OR

Cláusula ORDER BY

Ordene filas con la cláusula ORDER BY

- ASC: orden ascendente, por defecto
- DESC: orden descendente

La cláusula ORDER BY aparece en último lugar en la sentencia SELECT.

```
SELECT   *|{[DISTINCT] column|expression [alias],...}
FROM     table
[WHERE   condition(s)];
ORDER BY order-by-expression [ASC|DESC];
```

- **Ordenación por Defecto de Datos**
  El orden por defecto es ascendente:
  - Los valores numéricos se muestran con los valores más bajos primero; por ejemplo, 1 a 999.
  - Los valores de fecha se muestran con la fecha anterior primero; por ejemplo, 01-ENE-92 antes de 01-ENE-95.
  - Los valores de caracteres se muestran en orden alfabético; por ejemplo, *A* en primer lugar y *Z* en último.
  - Los valores nulos se muestran los últimos para las secuencias ascendentes y los primeros para las descendentes.
- **Reversión al Orden por Defecto**
  Para revertir el orden en el que se muestran las filas, especifique la palabra clave DESC después del nombre de columna en la cláusula ORDER BY.

- **Ordenación según Alias de Columna**
  Puede utilizar un alias de columna en la cláusula ORDER BY.

- **Ordenación según Múltiples Columnas**
  Puede ordenar el resultado de la consulta según más de una columna. El límite de ordenación es el número de columnas de la tabla dada.
  En la cláusula ORDER BY, especifique las columnas y separe los nombres de columna mediante comas. Si desea revertir el orden de una columna, especifique

DESC después de su nombre. También puede ordenar según columnas no incluidas en la cláusula SELECT.

Funciones de una Sola Fila

Las funciones de una sola fila:

- Manipulan elementos de datos.
- Aceptan argumentos y devuelven un valor.
- Actúan sobre cada fila devuelta.
- Devuelven un resultado por fila.
- Pueden modificar el tipo de dato.
- Se pueden anidar.
- Aceptan argumentos que pueden ser una columna o una expresión.

Sintaxis:

function_name [(arg1, arg2,...)]

- Funciones de caracteres: Aceptan entradas de caracteres y pueden devolver valores de caracteres y numéricos.
- Funciones numéricas: Aceptan entradas numéricas y devuelven valores numéricos.
- Funciones de fecha: Operan sobre valores del tipo de dato DATE. (Todas las funciones de fecha devuelven un valor del tipo de dato DATE excepto la función MONTHS_BETWEEN, que devuelve un número).
- Funciones de conversión: Convierten un valor de un tipo de dato en otro.
- Funciones generales

✓ **Funciones de Cadenas**

Mayúsculas/Minúsculas

LOWER (string expression) LCASE(string expression)	Convierte todo los caracteres en una cadena en minúsculas.
UPPER (string-expression) UCASE(string expression)	Convierte todo los caracteres en una cadena en mayúsculas

Manipulación de Cadenas

{ SUBSTRING | SUBSTR }(string-expression, start [, length])	Retorna una subcadena de una cadena. **string-expression** es la cadena del cual una subcadena será retornada. **start** posición de inicio de la subcadena a retornar. **length** la longitud de la subcadena a retornar en caracteres.
LENGTH (string-expression)	Retorna el número de caracteres en la cadena especificada.
INSERTSTR (integer-expression, string-expression-1, string-expression-2)	Inserta una cadena dentro de otra cadena en una posición especificada. **integer-expression** posición después del cual la cadena será insertada. Use cero para insertar una cadena al inicio.
STRING (string-expression [, ...])	Concatena une o más cadenas dentro de una cadena larga.
REPLACE (original-string, search-string, replace-string)	Reemplaza todas las ocurrencias de una subcadena con otra subcadena.

✓ **Funciones Numéricas**

Funciones numéricas ejecutan operaciones matemáticas sobre tipos de datos numéricos o retornan información numérica.

- ROUND (numeric-expression, integer-expression).- Redondea el valor a los decimales especificados.
  ROUND(45.926, 2) 45.93
- TRUNCNUM (numeric-expression, integer-expression).- Trunca el valor a los decimales especificados.
  TRUNC (45.926, 2) 45.92
- MOD (dividendo, divisor).- Devuelve el resto de la división.
  MOD (1600, 300) 100
- ABS (numeric-expression).- Calcula el valor absoluto de n.
  ABS (-16) 16
- CEILING (numeric-expression).- Calcula el menor número entero mayor o igual que n.
  CEILING (16.7) 17
- FLOOR (numeric-expression) Calcula el mayor número entero menor o igual que n.
  FLOOR (16.7) 16
- POWER (numeric-expression-1, numeric-expression-2).- Devuelve ne1 elevado a la ne2 potencia, ne2 debe ser entero
  POWER (3,2) 9

✓ **Trabajando con Fechas y Tiempo**

- La fecha (date) es un tipo de dato estándar que es un calendario de fecha en el formato: yyyy-mm-dd donde:
  yyyy es el año (un número de 1000 a 3000)
  mm es el mes (un número de 01 a 12)
  dd es el día (un número de 01 a 31)

  Por ejemplo:
  1990-01-31 es January 31, 1990.

  El formato de visualización de fecha por defecto es:
  DD/MM/YYYY = 24/05/1999

  CURRENT DATE retorna el año, mes y día actuales (del sistema).

```
SELECT  CURRENT DATE;
```

- La hora (time) es un tipo de dato estándar que es una hora en formato de 24-horas: hh:mm: ss.ffffff donde:
  hh es la hora (un número 00 a 23)
  mm son los minutos (un número 00 a 59)
  ss son los segundos (un número 00 a 59)
  ffffff son los microsegundos (un número 000000 a 999999)
  Colons (:) y periodo (.) son requeridos. Blancos no son permitidos.

  Por ejemplo:
  23:59:59.999999 es un microsegundo antes de medianoche.

  PowerBuilder e InfoMaker usan el formato de tiempo de 24-horas:
  hh:mm:ss:ffffff (hours:minutes:seconds:microseconds)

  CURRENT TIME retorna la hora, los minutos, segundos y fracciones de segundo actuales (del sistema).

```
SELECT  CURRENT TIME;
```

Aritmética con Fechas

- Sume o reste un número a/de una fecha para producir un valor de fecha.
- Reste dos fechas para buscar el número de días entre ellas.
- Sume horas a una fecha dividiendo el número de horas por 24.

Ejemplo:

```
SELECT last_name, (CURRENT DATE - hire_date)/7 AS WEEKS
FROM   employees
WHERE department_id = 90;
```

Funciones de Fecha

- DATE (expression).- Convierte la expresión en una fecha, y elimina cualquier hora, minuto o segundo contenido.
  Ejemplo:
  ```
  SELECT (CURRENT DATE - DATE ('1985-09-21 21:20:53'))/365 AS "Años";
  ```

- MONTH (date-expression)
  Retorna un número de 1 a 12 correspondiente al mes de la fecha dada

- MONTHS ([datetime-expression,] datetime-expression)
  MONTHS (datetime-expression, integer-expression)
  Número de meses entre dos fechas

- NOW (*).- Retorna el año, mes, día, hora, minuto, segundo y fracción de un segundo actual. La fracción de segundo. La precisión es limitada por la precisión del reloj del sistema.
  La información que la función NOW retorna es equivalente a la información retornada por la función GETDATE y el valor especial CURRENT TIMESTAMP.

- TODAY (*).- retorna la fecha actual.

- YEAR (datetime-expression).- retorna el año de una fecha dada.
  Ejemplo:
  ```
  SELECT YEAR ('2001-09-12');        retorna 2001.
  ```

- DOW (date-expression).- Retorna un número de 1 a 7 representando el día de la semana de la fecha, Domingo = 1, Lunes = 2, y así.

29. **Semana 12.- Practica SQL (Básico)**

Instrucciones.- Con la base de datos creada y poblada con datos anteriormente, resuelva las siguientes preguntas:

a) Escriba una consulta para mostrar la fecha actual. Etiquete la columna como Fecha.
b) Para cada empleado, visualice su número, apellido, salario y salario incrementado en el 15 % y expresado como número entero. Etiquete la columna como Nuevo Salario.
c) Modifique la consulta 2 para agregar una columna que reste el salario antiguo del nuevo. Etiquete la columna como Incremento.
d) Escriba una consulta que muestre los apellidos de los empleados con la primera letra en mayúsculas y todas las demás en minúsculas, así como la longitud de los nombres, para todos los empleados cuyos nombres comienzan por J, A o M. Asigne a cada columna la etiqueta correspondiente. Ordene los resultados según los apellidos de los empleados.
e) Para cada empleado, muestre su apellido y calcule el número de meses entre el día de hoy y la fecha de contratación. Etiquete la columna como MESES_TRABAJADOS. Ordene los resultados según el número de meses trabajados. Redondee el número de meses hacia arriba hasta el número entero más próximo.
f) Escriba una consulta que produzca lo siguiente para cada empleado: <apellido del empleado > gana <salario> mensual pero quiere <3 veces su salario>. Etiquete la columna como Salario Soñado.
g) Muestre el apellido de cada empleado, así como la fecha de contratación y la fecha de revisión de salario, que es el primer lunes después de cada seis meses de servicio. Etiquete la columna REVISION. Formatee las fechas para que aparezca el nombre del día.
h) Escriba una consulta para visualizar el apellido del empleado, el número y el nombre de departamento para todos los empleados.
i) Cree un listado único de todos los cargos que haya en el departamento 80. Incluya la ubicación del departamento en el resultado.
j) Escriba una consulta para mostrar el apellido del empleado, el nombre de departamento, el identificador de ubicación y la ciudad de todos los empleados que perciben comisión.
k) Visualice el apellido del empleado y el nombre de departamento para todos los empleados que tengan una a (minúscula) en el apellido.
l) Cree una consulta en la que pueda visualizar el nombre, el cargo, el nombre de departamento, el salario y el grado de todos los empleados.

SEMANA 13: SQL (2DA SESIÓN)

Función Condicional

- **IFNULL (** *expression-1*, *expression-2* [, *expression-3*] **)**
  Si la primera expresión es NULL, el valor de la segunda expresión se devuelve. Si la primera expresión no es NULL, el valor de la tercera expresión se devuelve. Si la primera expresión no es NULL y no hay tercera expresión, se devuelve NULL.

Obtención de Datos de Varias Tablas

✓ **Productos Cartesianos**

- Un producto Cartesiano se forma cuando:
  - Una condición de unión está omitida.
  - Una condición de unión no es válida.
  - Todas las filas de la primera tabla se unen a todas las filas de la segunda tabla.
- Para evitar un producto Cartesiano, incluya siempre una condición de unión válida en una cláusula WHERE.

✓ **Unión de Tablas**

Utilice una unión para consultar datos de más de una tabla.

Sintaxis:

```
SELECT   table1.column, table2.column
FROM     table1, table2
WHERE    table1.column1 = table2.column2;
```

- Escriba la condición de unión en la cláusula WHERE.
- Escriba en el nombre de columna el nombre de tabla como prefijo si aparece el mismo nombre de columna en más de una tabla.

- Unión de igualdad
  Cuando se corresponden la clave primaria y la clave foránea de dos tablas.

- Cualificación de Nombres de Columna Ambiguos
  - Utilice prefijos de tabla para cualificar nombres de columna que estén en varias tablas.
  - Mejore el rendimiento utilizando prefijos de tabla.
  - Distinga las columnas que tengan nombres idénticos pero que residan en tablas diferentes utilizando alias de columna.

- Unión de Más de Dos Tablas
  Para unir n tablas, necesita un mínimo de n-1 condiciones de unión. Por ejemplo, para unir tres tablas, se requiere un mínimo de dos uniones.

- Uniones de No Igualdad
  Ejemplo:
  El salario de la tabla EMPLOYEES debe estar entre el salario menor y el mayor de la tabla JOB_GRADES.

- Autouniones
  A veces, es necesario unir una tabla consigo misma.
  Ejemplo:
  Buscar el nombre del director de cada empleado

- Uniones Externas
  Si una fila no satisface una condición de unión, no aparecerá en el resultado de la búsqueda. Por ejemplo, en la condición de unión de igualdad de las tablas EMPLOYEES y DEPARTMENTS, el empleado Grant no aparece porque no hay identificador de departamento registrado para él en la tabla EMPLOYEES. En lugar de ver 20 empleados en el juego de resultados, ve 19.

- Creación de Uniones Cruzadas
  - La cláusula **CROSS JOIN** produce varios productos entre dos tablas.
  - Es lo mismo que un producto Cartesiano entre las dos tablas.

- Creación de Uniones Naturales
  - La cláusula NATURAL JOIN se basa en todas las columnas de las dos tablas que tienen el mismo nombre.
  - Selecciona filas de las dos tablas que tienen los mismos valores en todas las columnas coincidentes.
  - Si las columnas que tienen el mismo nombre tienen distintos tipos de dato, se devuelve un error.

- Recuperación de Registros con la Cláusula ON
  - La condición de unión para la unión natural es básicamente una unión de igualdad de todas las columnas con el mismo nombre.
  - Para especificar condiciones arbitrarias o especificar columnas para unir, se utiliza la cláusula ON.
  - La condición de unión se separa de otras condiciones de búsqueda.
  - La cláusula ON facilita la comprensión del código.

- Unión de Tres Tablas

- Uniones INNER y OUTER
  - En SQL: 1999, la unión de dos tablas que devuelve solamente las filas coincidentes es una unión interna.
  - Una unión entre dos tablas que devuelve los resultados de la unión interna así como las tablas izquierdas (o derecha) de filas no coincidentes es una unión externa izquierda (o derecha).
  - Una unión entre dos tablas que devuelve los resultados de una unión interna así como los de una unión izquierda y derecha es una unión externa completa.

Aplicación de Condiciones Adicionales
Puede aplicar condiciones adicionales en la cláusula WHERE. En el ejemplo mostrado se realiza una unión en las tablas EMPLOYEES y DEPARTMENTS y, además, solamente se visualizan los empleados con un identificador de director igual a 149.

✓ **Utilizando Funciones de Grupo**
Las funciones de grupo operan sobre juegos de filas para proporcionar un resultado por grupo.

- Tipos de Funciones de Grupo
  - AVG
  - COUNT
  - MAX
  - MIN

- STDDEV
- SUM
- VARIANCE

o Sintaxis de las Funciones de Grupo

```
SELECT      [column,] group_function(column), ...
FROM        table
[WHERE      condition]
[GROUP BY   column]
[ORDER BY   column];
```

o Uso de la Función COUNT
COUNT(*) devuelve el número de filas de una tabla.

- COUNT(*expr*) devuelve el número de filas con valores no nulos para *expr*.
- Visualice el número de valores de departamento de la tabla EMPLOYEES, excluyendo los valores nulos.

- Funciones de Grupo y Valores Nulos
  Las funciones de grupo ignoran los valores nulos de la columna.

✓ **Creación de Grupos de Datos**
A veces se necesita dividir la tabla de información en grupos más pequeños. Esto se puede realizar utilizando la cláusula GROUP BY.

o Sintaxis de la Cláusula Group By

```
SELECT      column, group_function(column)
FROM        table
[WHERE      condition]
[GROUP BY   group_by_expression]
[ORDER BY   column];
```

- Si incluye una función de grupo en una cláusula SELECT, no puede seleccionar también resultados individuales, *a menos que* la columna individual aparezca en la cláusula GROUP BY. Recibirá un mensaje de error si no incluye la lista de columnas en la cláusula GROUP BY.
- Si utiliza una cláusula WHERE, puede excluir filas antes de dividirlas en grupos.
- Debe incluir las *columnas* en la cláusula GROUP BY.
- No se puede utilizar un alias de columna en la cláusula GROUP BY.
- Por defecto, las filas se ordenan en orden ascendente de las columnas incluidas en la lista GROUP BY. Puede sustituir este orden por defecto utilizando la cláusula ORDER BY.
  Ejemplo:
  Todas las columnas de la lista SELECT que no estén en las funciones de grupo deben estar en la cláusula GROUP BY.

  La columna GROUP BY no tiene que estar en la lista SELECT.

  Uso de la Cláusula GROUP BY en Varias Columnas.- Puede devolver resultados resumidos para grupos y subgrupos incluyendo en una lista más de una columna GROUP BY. Puede determinar el orden por defecto de los resultados por el orden de las columnas en la cláusula GROUP BY.

Consultas no Válidas Utilizando Funciones de Grupo.- Toda columna o expresión de la lista SELECT que no sea una función agregada debe estar en la cláusula GROUP BY.

- No se puede utilizar la cláusula WHERE para restringir grupos.
- Utilice la cláusula HAVING para restringir grupos.
- No se pueden utilizar funciones de grupo en la cláusula WHERE.

Exclusión de Resultados de Grupo: **La Cláusula HAVING**
Utilice la cláusula HAVING para restringir grupos:
- Las filas se agrupan.
- Se aplica la función de grupo.
- Se muestran los grupos que coinciden con la cláusula HAVING.

Sintaxis:

```
SELECT      column, group_function
FROM        table
[WHERE      condition]
[GROUP BY   group_by_expression]
[HAVING     group_condition]
[ORDER BY   column];
```

Anidamiento de Funciones de Grupo.- Las funciones de grupo se pueden anidar hasta una profundidad de dos.

✓ **Subconsultas**

Las subconsultas son sentencias SELECT embebidas en cláusulas de otras sentencias SELECT. Con ellas puede crear sentencias potentes a partir de sentencias simples. Esto puede resultar muy útil si necesita seleccionar filas de una tabla con una condición que depende de los datos de la propia tabla.
Puede colocar la subconsulta en diversas cláusulas SQL como, por ejemplo:
- La cláusula WHERE
- La cláusula HAVING
- La cláusula FROM

¿Quién tiene un salario mayor que el de Abel?
Consulta Principal:
¿Qué empleados tienen salarios mayores que el de Abel?
Subconsulta
¿Cuál es el salario de Abel?

o Sintaxis de la Cláusula Group By

```
SELECT      select_list
FROM        table
WHERE       expr operator
                (SELECT     select_list
                FROM        table);
```

En la sintaxis:
operator incluye una condición de comparación como >, = o IN

Nota: Las condiciones de comparación son de dos clases: operadores de una sola fila (>, =, >=, <, <>, <=) y operadores de varias filas (IN, ANY, ALL).

Con frecuencia se hace referencia a la subconsulta como sentencia SELECT anidada, sub-SELECT o SELECT interna. La subconsulta se suele ejecutar primero y su resultado se utiliza para completar la condición de consulta de la consulta principal o externa.

- Escriba las subconsultas entre paréntesis.
- Coloque las subconsultas a la derecha de la condición de comparación.
- La cláusula ORDER BY de la subconsulta no es necesaria salvo que esté realizando un análisis N principal.
- Utilice operadores de una sola fila con subconsultas de una sola fila y operadores de varias filas con subconsultas de varias filas.

o Tipos de Subconsultas
  - Subconsultas de una sola fila: Consultas que devuelven una sola fila a partir de la sentencia SELECT interna.
  - Subconsultas de varias filas: Consultas que devuelven más de una fila a partir de la sentencia SELECT interna.

o Subconsultas de una Sola Fila
  - Devuelven una sola fila
  - Utilizan operadores de comparación de una sola fila (=, >, >= , <, <=,<>)

o La Cláusula HAVING con Subconsultas
  - Se ejecuta en primer lugar las subconsultas.
  - Se devuelve resultados a la cláusula HAVING de la consulta principal.

o ¿Qué Es Incorrecto en la siguiente Sentencia?

```
SELECT employee_id, last_name
FROM   employees
WHERE  salary =
            (SELECT    MIN(salary)
             FROM     employees
             GROUP BY department_id);
```

o Subconsultas de Varias Filas
  - Devuelven más de una fila
  - Utilizan operadores de comparación de varias filas (IN, ANY, ALL)

BASE DE DATOS

30. Semana 13.- Practica SQL (Intermedio)

Instrucciones.- Con la base de datos creada y poblada con datos anteriormente, resuelva las siguientes preguntas:

a) Diga si la siguiente sentencia SELECT se ejecuta satisfactoriamente:
```
SELECT last_name, job_id, salary AS Sal
FROM   employees;
```
(Verdadero/Falso)

b) Hay cuatro errores de codificación, identifíquelos:
```
SELECT    employee_id, last_name
sal x 12  ANNUAL SALARY
FROM      employees;
```

c) Muestre el apellido concatenado con el job ID, separado por una comma y espacio, y nombre la columna Employee and Title.

d) Muestre el apellido y el salario para todos los empleados cuyos salarios no están en el rango de $5,000 y $12,000.

e) Muestre el apellido, job ID, y fecha de inicio de los empleados contratados entre el 20 de Febrero de 1998 y el 1 de Mayo de 1998. Ordene ascendentemente por fecha de inicio.

f) Muestre el apellido de todos los empleados donde la tercera letra de su nombre es una a.

g) Muestre el apellido, job, y salario para todos los empleados cuyo job es representante de ventas o empleado de almacén y cuyos salarios no son iguales a $2,500, $3,500, o $7,000.

h) Crear un query que muestre los apellidos de los empleados y la cantidad de comisión. Si un empleado no gana comisión, poner la etiqueta “Sin Comisión” en la columna COMM.

i) Escriba un query para mostrar el apellido, nombre de departamento, location ID, y ciudad de todos los empleados quienes ganan una comisión.

j) Muestre los apellidos y los códigos de los empleados con los apellidos y códigos de sus managers. Etiquete las columnas Employee, Emp#, Manager, and Mgr#, respectivamente.

k) Modifique la pregunta anterior para mostrar todos los empleados incluyendo a King, quien no tiene manager. Ordene los resultados por el código del empleado.

l) Crear un query para mostrar el nombre y fecha de contratación de cualquier empleado contratado después que el empleado Davies.

m) Muestre el mas alto, bajo, suma, y promedio de salario de todos los empleados. Etiquete las columnas Máximo, Mínimo, Suma, y Promedio, respectivamente. Redondee sus resultados al entero más cercano.

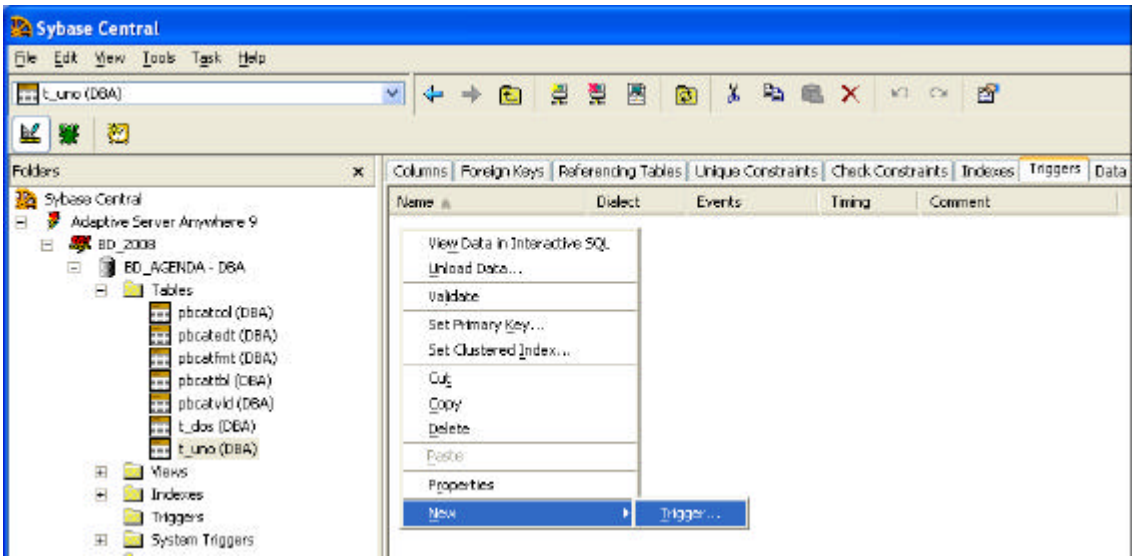
n) Modifique el Query anterior para mostrar el Máximo, Mínimo, Suma y Promedio de salarios para cada tipo de job.

o) Escriba un query para mostrar el número de gente con el mismo job.

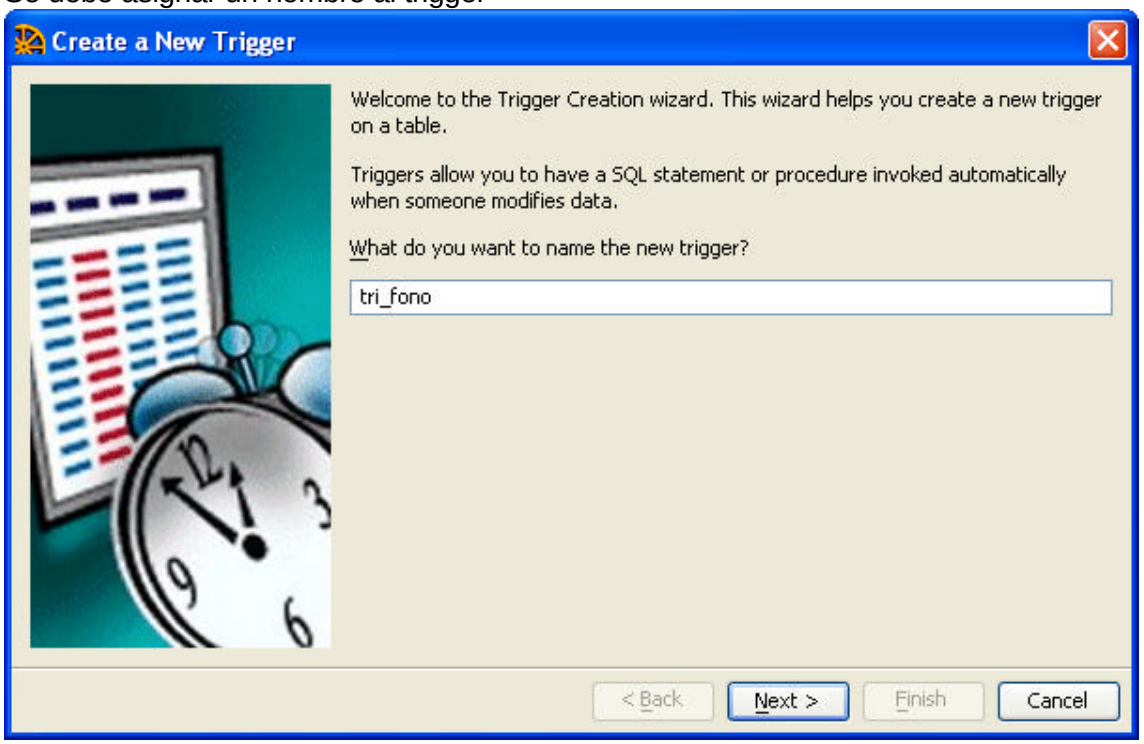
SEMANA 14: Lenguaje de: manipulación de datos y de definición de datos

CREANDO TRIGGERS CON SYBASE CENTRAL

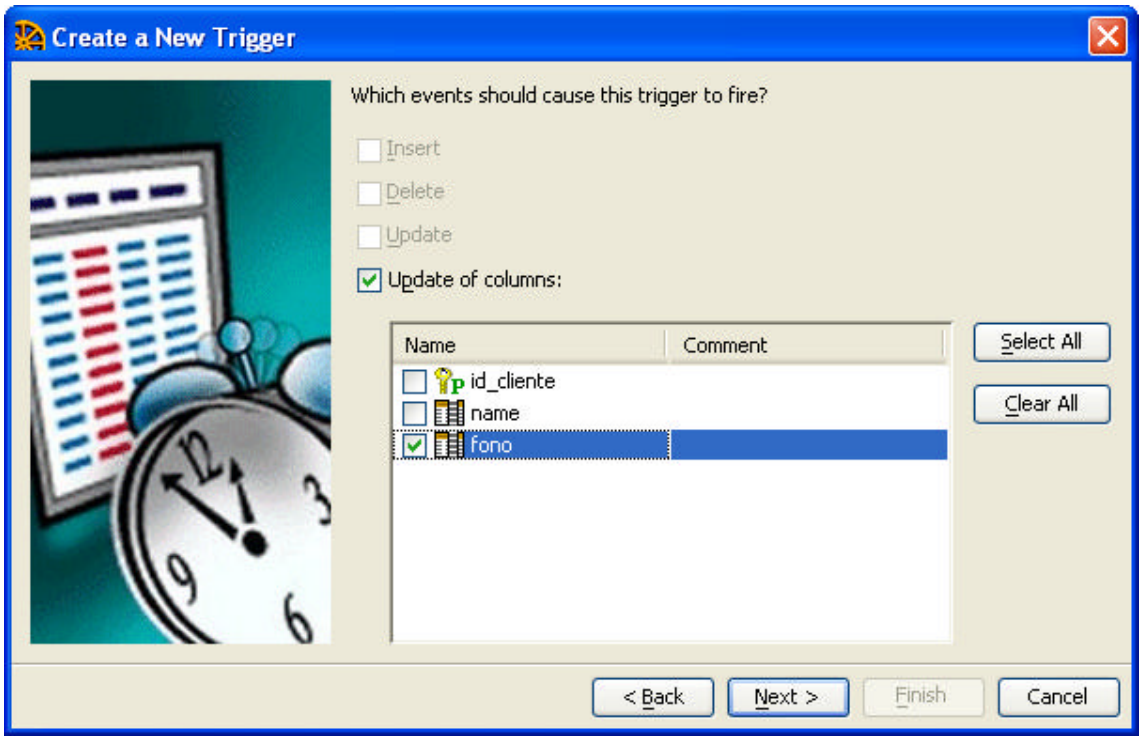
Después de conectar la base de datos "BD_Agenda" con Sybase Central, tenemos dos pequeñas tablas demo, t_uno y la t_dos, deseamos actualizar el campo teléfono de la última tabla, cuando se modifique el valor del teléfono sobre la tabla maestra.
Sobre la tabla en la cual se piensa emplear un disparador, se debe activar el tab (pestaña) Triggers, hacer click derecho y en el desplegable elegir **New** y en seguida **Trigger,** como se muestra:



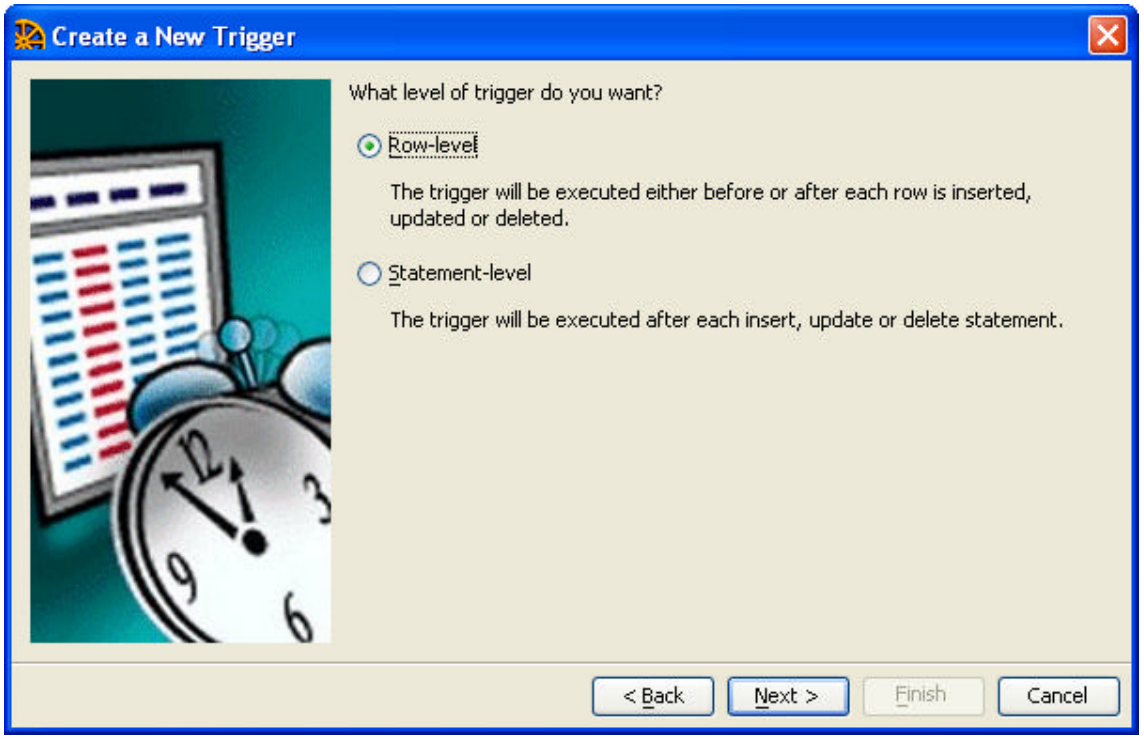
Se debe asignar un nombre al trigger



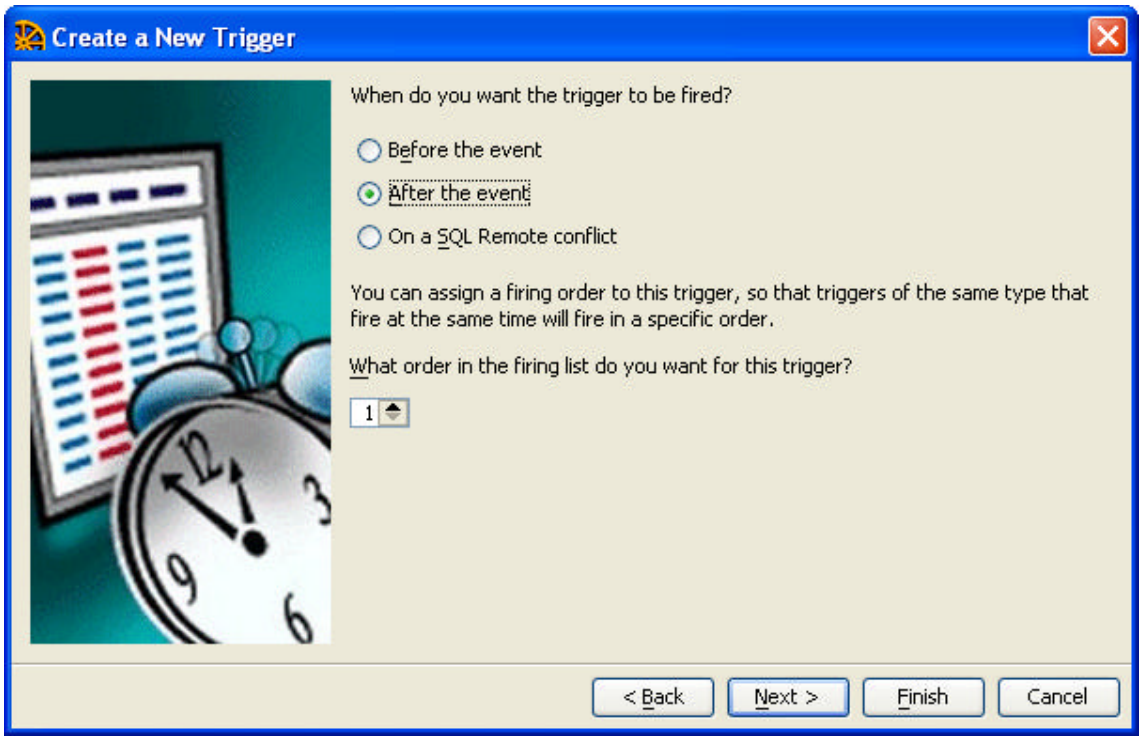
Seleccionamos el evento que disparará el trigger



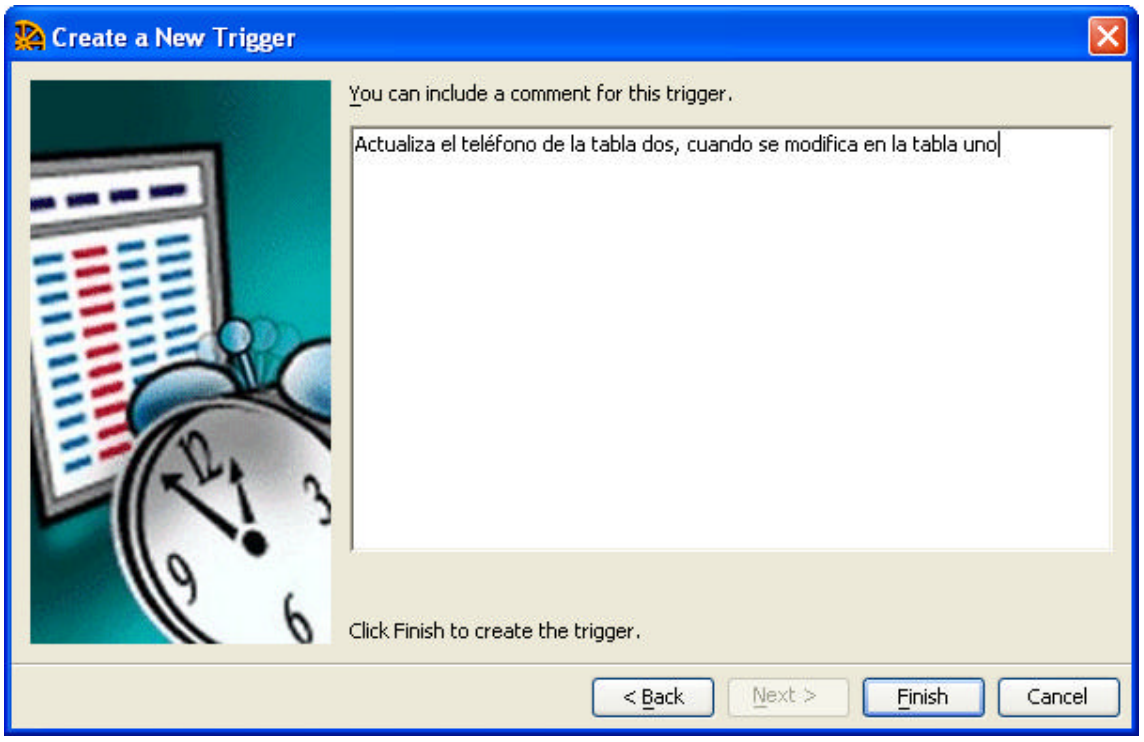
Luego se debe seleccionar el nivel de disparo del trigger, siendo Row Level trigger (disparador a nivel de fila) definido para ejecutarse antes o después de Insert, Update o Delete. Statement Level trigger (Disparador a nivel de sentencia), se ejecuta después de una sentencia. Para el ejemplo trabajaremos a nivel de fila.



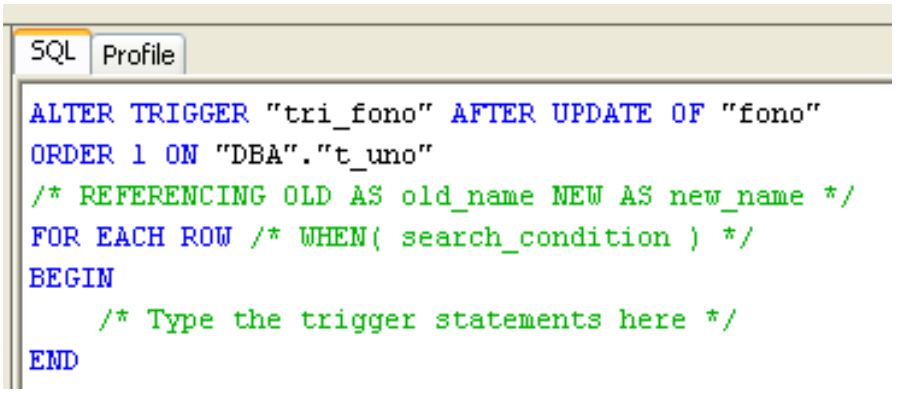
A continuación se debe seleccionar el momento de ejecución del trigger, antes o después de un evento o cuando se sucede un conflicto. Así mismo si es necesario indicar el orden de ejecución si existieran más de un trigger sobre el mismo evento.



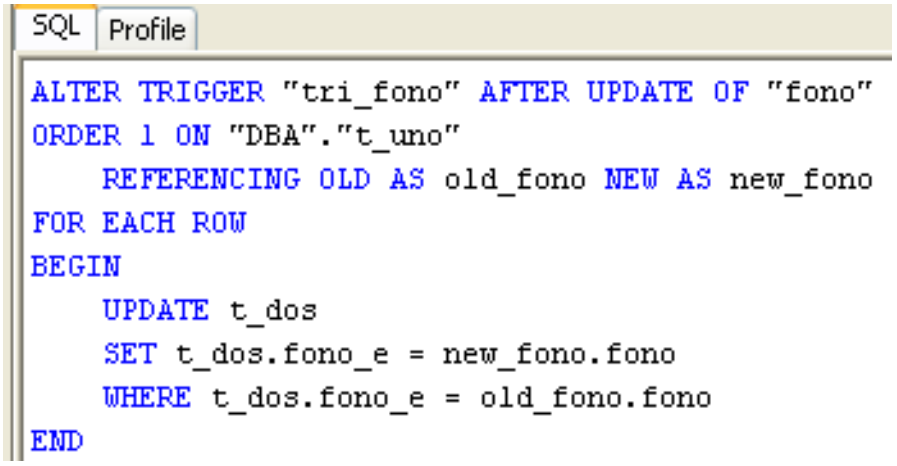
Podemos incluir un comentario que documente el nuevo trigger, y enseguida hacemos click en **Finalizar**



A continuación se nos muestra la Sintaxis básica del trigger, sobre la cual debemos agregar y/o modificar de acuerdo a nuestros requerimientos



Para el caso del ejemplo inicial, a continuación se muestran las modificaciones necesarias y adecuadas



Después de almacenar el trigger debe probarlo desde el cliente.

SEMANA 15

Base de datos Cliente/ Servidor y Base de Datos Distribuidas

31. Base de datos Cliente/ Servidor

Esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa -el servidor- que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

Una disposición muy común son los sistemas multicapa en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes computadoras aumentando así el grado de distribución del sistema.

La arquitectura cliente-servidor sustituye a la arquitectura monolítica en la que no hay distribución, tanto a nivel físico como a nivel lógico.

Características

Características de un cliente

En la arquitectura C/S el remitente de una solicitud es conocido como cliente. Sus características son:

- Es quien inicia solicitudes o peticiones, tienen por tanto un papel activo en la comunicación (dispositivo maestro o amo).
- Espera y recibe las respuestas del servidor.
- Por lo general, puede conectarse a varios servidores a la vez.
- Normalmente interactúa directamente con los usuarios finales mediante una interfaz gráfica de usuario.

Características de un servidor

En los sistemas C/S el receptor de la solicitud enviada por cliente se conoce como servidor. Sus características son:

- Al iniciarse esperan a que lleguen las solicitudes de los clientes, desempeñan entonces un papel pasivo en la comunicación (dispositivo esclavo).
- Tras la recepción de una solicitud, la procesan y luego envían la respuesta al cliente.
- Por lo general, aceptan conexiones desde un gran número de clientes (en ciertos casos el número máximo de peticiones puede estar limitado).
- No es frecuente que interactúen directamente con los usuarios finales.

32. Base de datos distribuidas

En un sistema de base de datos distribuida, los datos se almacenan en varios computadores. Los computadores de un sistema distribuido se comunican entre sí a través de diversos medios de comunicación, tales como cables de alta velocidad o líneas telefónicas. No comparten la memoria principal ni el reloj.
Los procesadores de un sistema distribuido pueden variar en cuanto su tamaño y función. Pueden incluir microcomputadores pequeños, estaciones de trabajo y sistemas de computadores grandes de aplicación general. Estos procesadores reciben diferentes nombres, tales como localidades, nodos o computadores.
La diferencia principal entre los sistemas de base de datos centralizados y distribuidos es que, en los primeros, los datos residen en una sola localidad, mientras que, en los últimos, se encuentran en varias localidades.

Estructura de Base de Datos Distribuidas

Un sistema distribuido de base de datos consiste en un conjunto de localidades, cada una de las cuales mantiene un sistema de base de datos local. Cada localidad puede procesar transacciones locales, o bien transacciones globales entre varias localidades, requiriendo para ello comunicación entre ellas.
Las localidades pueden conectarse físicamente de diversas formas, las principales son: Red totalmente conectada, prácticamente conectada, con estructura de árbol, de estrella, de anillo.

Las diferencias principales entre estas configuraciones son:
Coste de instalación: El coste de conectar físicamente las localidades del sistema
Coste de comunicación: El coste en tiempo y dinero que implica enviar un mensaje desde la localidad A a la B.
Fiabilidad: La frecuencia con que falla una línea de comunicación o una localidad.
Disponibilidad: La posibilidad de acceder a información a pesar de fallos en algunas localidades o líneas de comunicación.

Ventajas de la distribución de datos

- La principal ventaja de los sistemas distribuidos es la capacidad de compartir y acceder a la información de una forma fiable y eficaz.
- Utilización compartida de los datos y distribución del control
- La ventaja principal de compartir los datos por medio de la distribución es que cada localidad pueda controlar hasta cierto punto los datos almacenados localmente. En un sistema centralizado, el administrador de base de datos de la localidad central controla la base de datos. En un sistema distribuido existe un administrador global de la base de datos que se encarga de todo el sistema. Parte de esta responsabilidad se delega al administrador de base de datos de cada localidad. Dependiendo del diseño del sistema distribuido, cada administrador local podrá tener un grado de autonomía diferente, que se conoce como autonomía local. La posibilidad de contar con autonomía local es en muchos casos una ventaja importante de las bases de datos distribuidas.

Fiabilidad y disponibilidad

Si se produce un fallo en una localidad de un sistema distribuido, es posible que las demás localidades puedan seguir trabajando. En particular, si los datos se repiten en varias localidades, una transacción que requiere un dato específico puede encontrarlo en más de una localidad. Así, el fallo de una localidad no implica necesariamente la desactivación del sistema.

El sistema debe detectar cuando falla una localidad y tomar las medidas necesarias para recuperarse del fallo. El sistema no debe seguir utilizando la localidad que falló. Por último, cuando se recupere o repare esta localidad, debe contarse con mecanismos para reintegrarla al sistema con el mínimo de complicaciones.
La disponibilidad es fundamental para los sistemas de bases de datos que se utilizan en aplicaciones de tiempo real. Por ejemplo, si una línea aérea no puede tener acceso a la información, es posible que pierda clientes a favor de la competencia.
Agilización del procesamiento de consultas
Si una consulta comprende datos de varias localidades, puede ser posible dividir la consulta en varias subconsultas que se ejecuten en paralelo en distintas localidades. Sin embargo, en un sistema distribuido no se comparte la memoria principal, así que no todas las estrategias de intersección se pueden aplicar en estos sistemas. En los casos en que hay repetición de los datos, el sistema puede pasar la consulta a las localidades más ligeras de carga.

Desventajas de la distribución de los datos

- La desventaja principal de los sistemas distribuidos es la mayor complejidad que se requiere para garantizar una coordinación adecuada entre localidades.
  El aumento de la complejidad se refleja en: Coste del desarrollo de software: es más difícil estructura un sistema de bases de datos distribuidos y por tanto su coste es menor
- Mayor posibilidad de errores: puesto que las localidades del sistema distribuido operan en paralelo, es más difícil garantizar que los algoritmos sean correctos.
- Mayor tiempo extra de procesamiento: el intercambio de mensajes y los cálculos adicionales son una forma de tiempo extra que no existe en los sistemas centralizados.

REFERENCIAS BIBLIOGRAFICAS

1. **Date** C.J. *Introducción a los Sistemas de Bases de Datos*. Ed. Addison-Wesley Iberoamericana S.A., USA. 2001.
   Código Bibliot. UNS: 005.74 D28

2. **Batini** Carlo, **Ceri** Stefano, **Navathe** Shamkant B. *Diseño Conceptual de Bases de Datos*. Ed. Addison-Wesley / Diaz de Santos. USA.2000.
   Código Biblio. UNS: 005.74 B25

3. **Korth** Henrio F. - **Silberschatz** Abraham. *Fundamentos de Bases de Datos*. 3ra Edición. Ed.McGraw-Hill, Inc. España. 2001.

4. **Kroenke David.** *Procesamiento de Base de Datos.* Ed. Pearson Educación. Mexico. 8va Edición. 2003.

5. **Post** Gerald V. *Sistemas de Administración de Bases de Datos.* 3ra Edición. Ed.McGraw-Hill, Inc. España. 2006.

6. **Guide User PowerDesigner 11.0. Sybase.**

7. **Guide User SQL Anywhere Sybase 9.0**

Enlaces

Base de datos Cliente Servidor
http://es.wikipedia.org/wiki/Cliente-servidor

Base de datos Distribuidas
http://html.rincondelvago.com/bases-de-datos-distribuidas_1.html