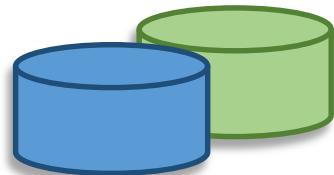


MANUAL DE BASE DE DATOS.

Diseño Conceptual y Lógico



Mag. Hugo Esteban Caselli Gismondi

Primera Edición

© HCG-2019

Tabla de contenidos

INTRODUCCIÓN	3
Capítulo I.- Introducción a los Sistemas de Bases de Datos ..	4
Capítulo II.- Modelo entidad-relación	17
Capítulo III.- Modelo entidad-relación extendido	26
Capítulo IV.- Casos Modelo entidad-relación	40
Capítulo V.- Diseño Conceptual de Bases de Datos	56
Capítulo VI.- Caso de Diseño Conceptual de Bases de Datos ..	74
Capítulo VII.- El Modelo Relacional	84
Capítulo VIII.- Mejorando la calidad de esquemas de BD	129
Referencias Bibliográficas	132

INTRODUCCIÓN

Alo largo de los últimos años, hemos visto como las bases de datos de las organizaciones iniciaron con guardar sus datos en archivos simples, generando doble trabajo o más, ya sea por redundancia, inconsistencias de los datos, bajos niveles de seguridad, falta de independencia de los datos, etc., en el largo camino, con los aportes de E.F. Codd, Peter Chen, entre otros, esto se fue modificando, primero en entender que las teorías eran válidas, y que esto luego se validó de manera natural en el tiempo, cuando ellas pudieron ser implementadas en las herramientas gestoras de bases de datos, que de momento aún son las más populares en el mercado, como son las bases de datos relacionales y en su soporte el modelo entidad relación, que los analistas utilizan para poder diseñar esas bases de datos.

Hemos considerado en el presente Manual una breve guía en estos enfoques de modelado de datos, partiendo desde el Diseño conceptual que involucra el modelo entidad relación que inicio con Peter Chen, pero que se ha visto enriquecido en el camino y ha sido bastante mejorado de tal manera que podamos tener resultados de representación de los sistemas del mundo real con una precisión semántica mucho mayor, que se derivan en poder utilizar estrategias de diseño, como ayudas metodológicas que nos evitan los tanteos innecesarios y nos vuelven más ordenados y que la elicitation de requerimientos se plasmen de manera sintáctica y semánticamente correcta.

Avanzando en una segunda parte en donde podremos realizar la revisión de nuestros diseños conceptuales a través de la normalización, que es un paso antes de poder realizar la implementación de nuestro modelo en la herramienta gestora de base de datos de nuestra elección. De forma paralela, podemos, a partir de nuestro diseño conceptual realizar su transformación con una serie de pautas ordenadas, de tal manera que, a la hora de guardar la data, no se tengan alteraciones semánticas y podamos recuperar la data en el sentido idéntico en el cual el usuario lo requirió, sin ningún tipo de anomalías, esto significa que, si detectamos alguna, de manera dinámica debemos proceder a sus mejoras continuas.

En el mismo sentido, no debemos perder de vista los continuos cambios tecnológicos, pues ya se vislumbran nuevas maneras de darle tratamiento a las grandes cantidades de datos que generamos.

Capítulo I

INTRODUCCIÓN A LOS SISTEMA DE BASES DE DATOS

a) Dato e información

Partiremos con la definición de dato, proporcionada por la Real Academia Española [1] quienes nos dicen que es “*Información sobre algo concreto que permite su conocimiento exacto o sirve para deducir las consecuencias derivadas de un hecho*”, naturalmente que ellos ya introdujeron la palabra información para poder definirla, que luego para nosotros tiene una connotación distinta que nos evite las confusiones, por lo que, si realmente iniciáramos con su origen etimológico, [2] encontramos que dato proviene del latín datum, que significa dado, proporcionado, que es la manera adoptada por [3] también por [4] de igual manera [5] quienes coinciden en decir que los datos son hechos dados, y podemos ejemplificarlo con un número, con un nombre, pero no tienen mayor significado si no es asociado a un contexto en particular.

En consecuencia, una posible acepción de información desde el punto de vista de las bases de datos y los sistemas de información, sería: el significado que los datos tienen de manera contextualizada para un usuario en particular. En extensión, podríamos aseverar que dato significaría: la fracción mínima de información en un contexto particular. Por ejemplo: un número tal como 59 no significa nada a menos que lo contextualicemos y digamos que significa peso y a partir de ahí podemos obtener información que sea relevante para una gestión de pacientes en un nosocomio, podemos obtener promedios de pesos, de ser necesario acumulados de pesos, frecuencias de pesos, etc.

b) Archivos convencionales

Teniendo en cuenta el significado de dato como parte constitutiva de la información, veremos que, con la era industrial, las empresas crecieron, y su vez la data que manejaban crecieron en la misma magnitud, y las formas de conservarla en más de los casos se

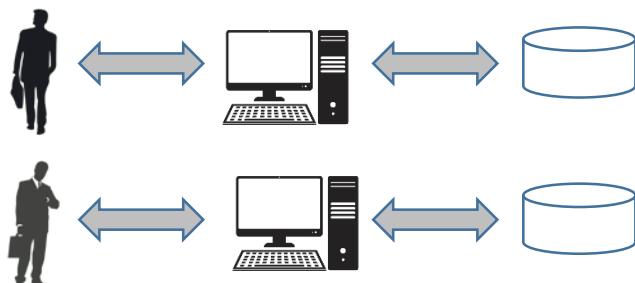
realizaba en papel, miles de clientes, miles de papeles, conservar tantos datos era complicado, buscar algún dato se tornaba también complicado, avanzando sobre la mitad del siglo XX, la aparición de las computadoras, y su forma rápida de masificarse, se introdujeron nuevas maneras de conservar la data de una organización, entre ellos en los inicios resultó en el manejo de lo que ahora se conoce como archivos convencionales, los cuales tuvieron en su momento sus ventajas, pero que a la larga sus desventajas luego, los hicieron más caóticos que la solución que proporcionaban.

Desventajas de los archivos convencionales

De acuerdo con [6] y con [7] las desventajas más saltantes se enumeran a continuación:

- ❖ Datos separados y/o aislados

En cualquier negocio, por más pequeño que sea, se tendrá a los clientes como eje y se conservan datos de estos y hay una persona encargada de las altas de los clientes, por otro lado, están las persona que se encargan de las ventas, quienes deben relacionar las ventas que ejecutan día a día con esos clientes, al tener cada quien su propia computadora y su propio archivo, la data relevante, está aislada de manera natural.



Cada quién es dueño de sus datos

- ❖ Datos duplicados e inconsistencia de la data

Cuando se trabaja con archivos convencionales quién realiza las altas de clientes conserva sus nombres, dirección, situación jurídica, correo, teléfono de contacto, etc. y probablemente quien maneje los datos de las ventas haga lo propio, además de los datos de la venta, también se ve obligado a tener referencias de a quién le está vendiendo y si la venta es a crédito con

mayor razón. Al generarse está redundancia, se incrementa la inconsistencia de la data, por cuanto cada usuario desde su visión, tendrá su propia manera de almacenar los datos, por ejemplo, siendo la misma persona, pero en aplicación distintas dentro de la misma organización, ventas conserva a Juan C. Pérez, y en contabilidad Juan Carlos Pérez Pérez

❖ Dependencia del programa de aplicación

En los albores del uso del computador de manera casi masiva, el procesamiento de estos archivos de almacenamiento tenía un vínculo directo con la forma física de almacenamiento, cuyo código de programación debía formar parte de la aplicación, en ese sentido al haber N usuarios, habrá N aplicaciones, por consiguiente, N almacenamientos, de ocurrir un cambio en un campo importante en particular, el cual debiera ser incluido o excluido de esas N aplicaciones, dado que el lenguaje de programación pudo ser COBOL, habría que realizar esos N mantenimientos, que pudieron entorpecer el normal desarrollo del trabajo.

❖ Archivos incompatibles

El crecimiento de la organización, hace necesario el tener que procesar la data que crece también, los desarrolladores nos proporcionan soluciones, pero muchas veces ocurría que, con lenguajes de programación diferentes, hemos mencionado que la aplicación del ejemplo anterior está desarrollada en COBOL, pero también tenemos aplicaciones desarrolladas en Lenguaje C, estos lenguajes tienen formatos de código distintos, lo que hace que sus productos sean incompatibles, lo que implicaría una tarea adicional que es transformar los datos de un lenguaje al otro para que podamos resolver un procesamiento adecuado y correcto con respecto a nuestros clientes.

❖ Dificultad de proporcionar adecuada vista de usuario

Si el Gerente de la empresa, desea ver toda la data del cliente que incluya sus ventas y probablemente sus créditos, esto sería muy dificultoso pues la data está separada en diversas máquinas y en diversas aplicaciones.

❖ Acceso concurrente

Pueden ocurrir anomalías, cuando más de un usuario quiere acceder a la data a la vez, por ejemplo, un cliente quiere retirar S/. 1000.00 de su cuenta, que algunos instantes previos le fue abonado por un familiar, pero solo se ve su cuenta vacía. Esto ocurre con agencias descentralizadas que tiene que actualizar su data la cerrar la noche y no ven las transacciones actualizadas al instante.

❖ Problemas de seguridad

Todos los usuarios del sistema no deben acceder a toda la data de la organización, por ejemplo, el personal de logística de una organización financiera no tiene por qué ver la data de los créditos de los clientes

Archivo - Ventas					
Num	Nombre	2016	2017	2018	

Archivo - Actividad Actual					
Num	Nombre	Área	2016	2017	

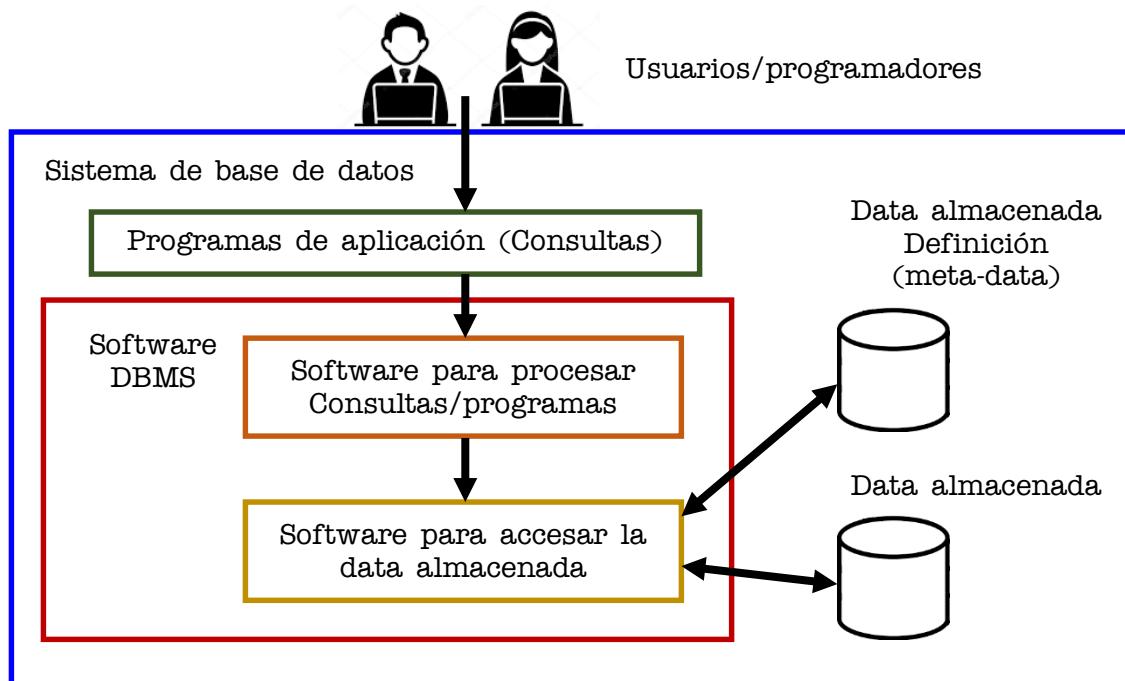
Archivo - Personal						
Num	Dpto	Nombre	Dirección	Celular	Salario	

El uso de archivos separados, implica que los mismos datos se almacenen en más de un lugar.

c) Base de datos y Sistemas de base de datos

Para [3] y [4] una base de datos es una colección de datos relacionados en un formato estándar.

De igual forma para [3], [4] y [5] un Sistema de base de datos es un sistema computarizado para guardar datos, dicho software permite crear y dar mantenimiento a la base de datos.



Entorno de un Sistema de base de datos, de Adaptado de [3]

Por lo tanto, las bases de datos no son una simple colección de archivos, son una fuente central de datos significativos para un determinado contexto, que son compartidos por múltiples usuarios sobre diversas aplicaciones.

La persona responsable de asegurar que la base de datos satisfaga los objetivos esperados, es el **administrador de base de datos**. Estos objetivos de eficiencia son:

- Asegurar que los datos, puedan ser compartidos por los usuarios.
- Realizar el mantenimiento de los datos
- Garantizar que los datos requeridos estén disponibles para las aplicaciones presentes y futuras
- Permitir que la base de datos evolucione y se adapte a las necesidades de los usuarios.

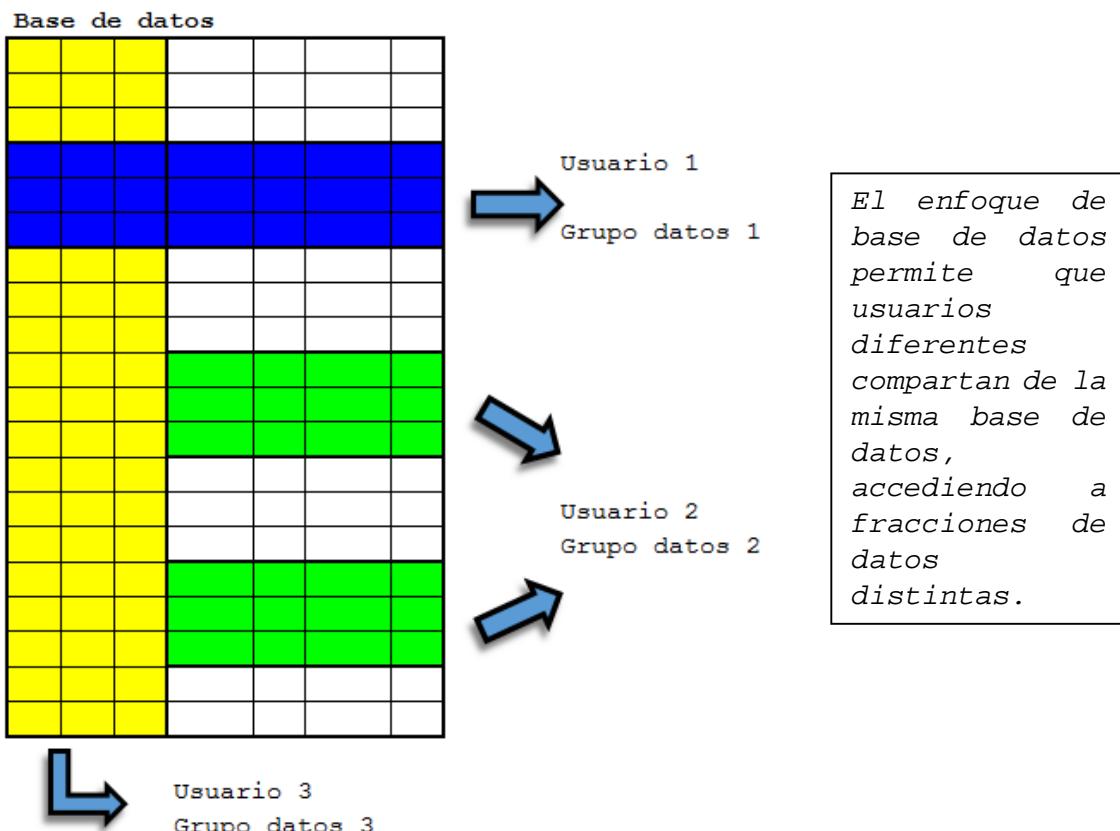
- e) Permitir que cada usuario desarrolle su propia visión de los datos, sin preocuparse de como los datos se almacenan físicamente.

Ventajas y desventajas del enfoque de base de datos

De acuerdo con [3] enumeramos las siguientes ventajas:

- a) Disminuir la redundancia
- b) Restricción de accesos no autorizados
- c) Almacenamiento persistente para los objetos del programa
- d) Proporciona estructuras de almacenamiento para procesamiento eficaz de consultas
- e) Copias de seguridad y recuperación
- f) Proporcionar interfaces de usuario
- g) Representación de relaciones complejas entre los datos
- h) Implementación de restricciones de integridad
- i) Inferencia y acciones usando reglas.

Entre las desventajas, se puede mencionar que, al almacenar los datos en un solo lugar, estos son más vulnerables a accidentes y precisan de un respaldo completo, aun cuando esto se ha mejorado notablemente con bases de datos replicadas y distribuidas. Riesgo latente de quien administra la base de datos es que sea el único privilegiado o habilitado de realizar los mantenimientos a la base de datos, pudiendo crearse barreras burocráticas insuperables.



d) Arquitectura de los Sistemas de Base de datos

La arquitectura de los Sistemas de base de datos propuesto por ANSI/X3/SPARC en 1975 [8], proporciona tres niveles de abstracción, los cuales ocultan su complejidad inherente y permiten a los usuarios tener la visión adecuada de acuerdo con sus necesidades particulares. Estos niveles de abstracción son los siguientes:

✓ **Nivel Físico**

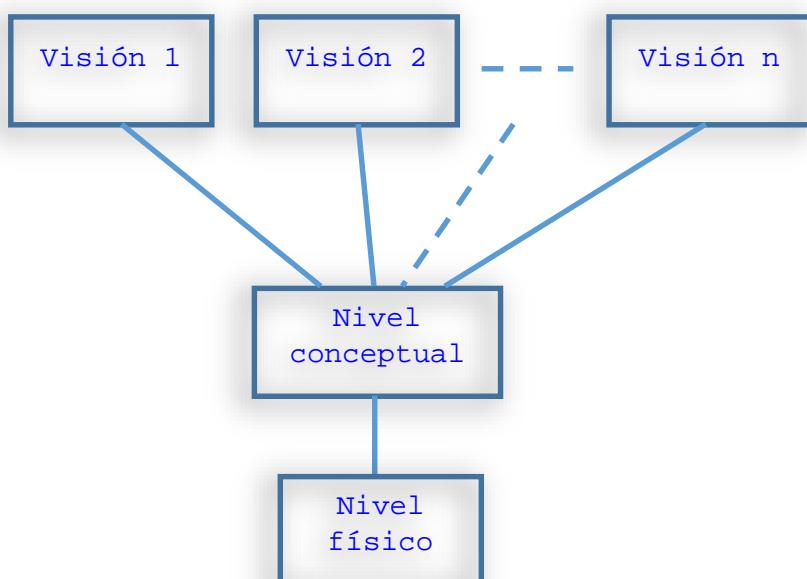
Este nivel describe como se almacenan los datos es decir se describe en detalle las estructuras complejas de bajo nivel, por eso se le nomina como el nivel más bajo.

✓ **Nivel Conceptual**

Es el que describe que datos se almacenaran en la base de datos y las relaciones que existen entre estos datos. Esta descripción se realiza a través de estructuras relativamente sencillas que pueden tener una contraparte de estructuras complejas a nivel físico. Este nivel es de los analistas/diseñadores y/o administradores de base de datos, quienes tiene la función de determinar cómo se debe guardar la data en la base de datos.

✓ **Nivel de Visión**

Es el nivel más alto, es el nivel del usuario, en donde se describen solo partes de la base de datos, dado que no todos los usuarios necesitan conocer toda la data de la organización. Por lo tanto, el sistema debe ser capaz de proporcionar tantas visiones como usuarios diferentes se tengan.



e) La Independencia de datos

Sobre la base de la abstracción previa, como parte de la arquitectura y como una ventaja adicional del enfoque de base de datos, es que podemos modificar una definición en un esquema sin que se afecte el nivel inmediato, a esto se llama independencia de datos [4] y [5], con dos niveles de independencia:

- Independencia física de datos

Se entiende como 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

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. De cualquier manera, la independencia de datos oculta detalles de implementación a los usuarios, por lo que ellos se concentraran en la estructura general y no en los detalles de bajo nivel.

f) Enfoque orientado a los datos para el Diseño de Sistemas

Con este enfoque, primero se diseña la base de datos [9], 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. Este enfoque se descompone en diseño conceptual, lógico y físico.

- Diseño conceptual

El propósito es describir el contenido de la data de la base de datos, más que las estructuras de almacenamiento que se necesitan para manejar esta data. 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 SGBD 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 SGBD.

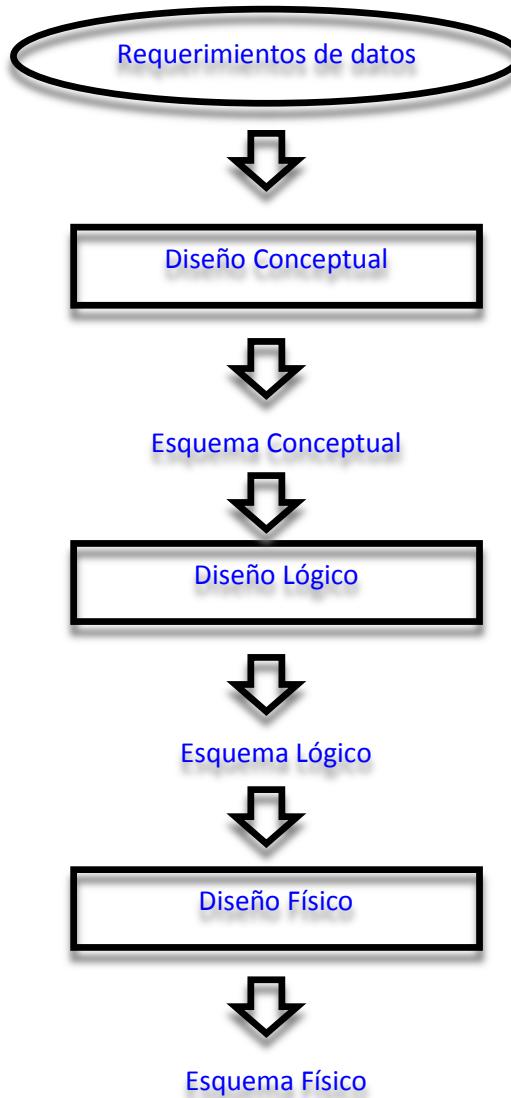
Para especificar el esquema lógico se usa el modelo lógico (jerárquico, de redes, o relacional, que es actualmente el más usado). El diseño lógico depende del modelo de datos usado por el SGBD y no del SGBD utilizado (el diseño lógico se realiza de la misma forma para todos los SGBD relacionales porque 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.

El gráfico siguiente muestra las etapas previstas en un Enfoque Orientado a los Datos:



Enfoque orientado a los datos para el Diseño de Sistemas

Fuente: Adaptado de [9]

g) Modelos de datos

Un modelo de datos es de acuerdo con [9] y con [3] una serie de conceptos que pueden utilizarse para describir un conjunto de datos, sus relaciones, restricciones 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.

Categorizaremos 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. Existen diferentes modelos, y es probable que aparezcan más.

Algunos de los más extensamente conocidos son:

- Modelo entidad-relación.
- Modelo orientado a objetos.
- Modelo binario.
- Modelo semántico de datos.
- Modelo infológico.
- 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 el modelo relacional, de red y jerárquico.

El modelo relacional, que ha ganado aceptación por encima de los otros dos hasta la actualidad. Los modelos de red y jerárquico, son parte de la historia y los mencionamos de manera breve para entender la evolución a las bases de actuales y proyectarnos hacia los nuevos estándares, aun cuando [5] los ha declarado obsoletos. A continuación, 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 relationales 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.

Los modelos de manera genérica están conformados por esquemas y a su vez los esquemas responden a una instantánea de la realidad visto como caso.

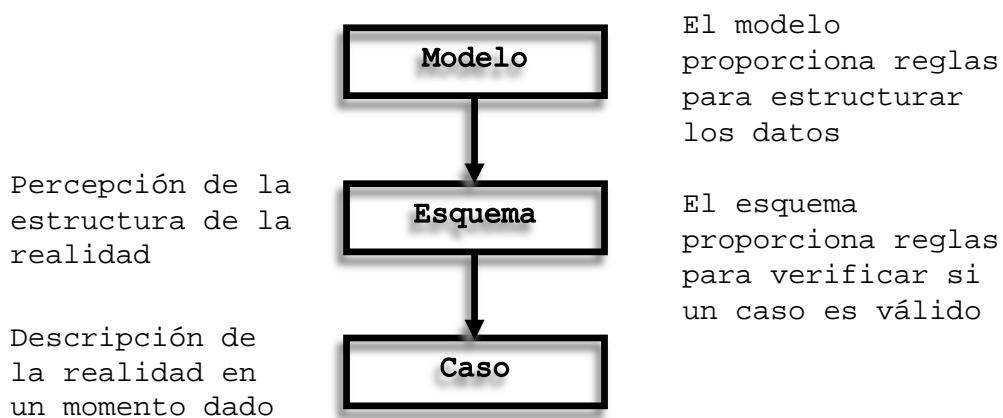
Esquema

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.

Caso

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 datos puede verse como la transición de un caso a otro, originada por alguna operación de modificación de datos.



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.

Capítulo II

MODELO ENTIDAD-RELACIÓN

El modelo entidad-relación (MER) fue propuesto por Peter Chen en 1976, como una herramienta para el diseño de base de datos, sobre vistas naturales del mundo real como las entidades, relaciones y atributos [10], más adelante fue enriquecido con la participación de otros autores, quienes agregaron otros conceptos como atributos compuestos, jerarquías de generalización, lo que se conoce como MER extendido o mejorado [9], [3].

a) Elementos Básicos del Modelo E-R

Como ya se mencionó, los conceptos básicos para el MER son las entidades, relaciones y atributos los cuales se revisan a continuación:

➤ Entidad

[8] lo define como una persona, lugar, cosa, concepto o evento real o abstracto, de interés para la empresa. Una entidad es un “objeto” el cual puede ser identificado claramente y se puede distinguir de otro, 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, un concepto o un evento. ESTUDIANTE, DOCENTE, ESCUELA, son ejemplos de entidades para una base de datos de una Universidad. Las entidades se representan gráficamente con un rectángulo.



ESCUELA

➤ Relación

Las relaciones representan agregaciones de dos o más entidades. En el caso de relacionar dos entidades se denomina relación binaria, un ejemplo de la relación binaria sería PERTENECE_A entre las entidades DOCENTE y DEPARTAMENTO. Cuando más de dos entidades son las que se relacionan, se denominaran

relaciones n-arias y si una entidad se relaciona consigo misma, se denominará relación recursiva. Gráficamente una relación se representa con un rombo, como se ve en el ejemplo siguiente:



➤ Atributo

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 DOCENTE podrían ser: Nombre, Apellido, DNI y Profesión



Atributos compuestos

Son grupos de atributos simples que se complementan entre ellos y tienen un significado común pero independientes entre ellos, estos pueden formar una jerarquía. Se representa con un ovalo que contiene el nombre integrador.



Atributos almacenados y derivados

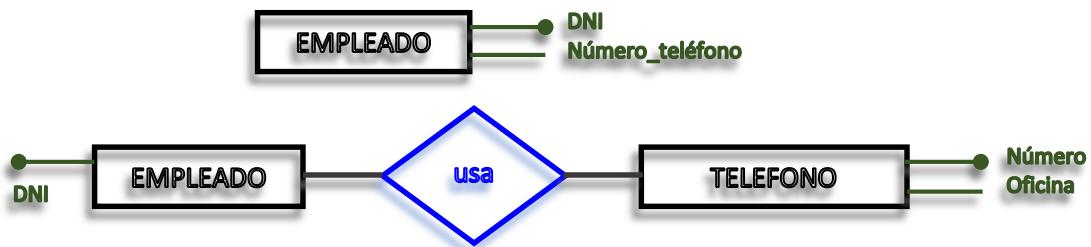
Los atributos almacenados son los denominados atributos base u origen, por ejemplo, fecha_de_credito.

Los atributos derivados son los que surgen a partir de un atributo almacenado o entidades relacionadas. A partir de la fecha de crédito, se puede obtener los días de mora, si acaso el cliente no paga puntualmente sus cuotas. Pero no vale la pena tener un atributo que cambia dinámicamente en el tiempo. De igual manera puede tener un atributo Número de alumnos matriculados a Base de Datos (Num_matr_BD) cuando ese valor se puede calcular con SQL.

b) Atributo o Entidad

En muchos casos, el recién iniciado encuentra la dificultad de decidir si debe modelar una determinada característica del dominio del problema del mundo real, ¿Como entidad o como atributo? Cuestión que no tiene una respuesta sencilla, y básicamente dependerá de la habilidad adquirida por el modelador de poder ubicarse en el contexto adecuado y poder proporcionar una solución que represente semánticamente los requerimientos y necesidades de guardar datos de los usuarios.

Caso típico es el del empleado que cuenta con un número de teléfono, pero resulta que dicho número es utilizado por varios empleados dentro de la misma oficina, para efectos de diseño conceptual, se puede modelar de dos maneras:



c) Cardinalidad

Peter Chen, mapeo inicialmente el indicador del número de entidades en cada conjunto de entidades a las cuales les está permitido un conjunto de relaciones tales como 1:1, 1:n y n:m [10]. En definitiva, la Cardinalidad se puede definir como el número mínimo y máximo de ocurrencias de un tipo de entidad con respecto a otra entidad que participa en la relación [11]. A continuación,

abordaremos la cardinalidad tanto en las relaciones como en los atributos.

Cardinalidad en las relaciones

➤ Cardinalidad mínima

Se utiliza para especificar si la participación de una entidad en la relación es obligatoria u opcional, donde:

0: Indica que una entidad no está obligatoriamente asociada a través de la relación, por lo tanto, su participación es opcional.

1: Indica que una entidad debe estar asociada a través de la relación con participación obligatoria.

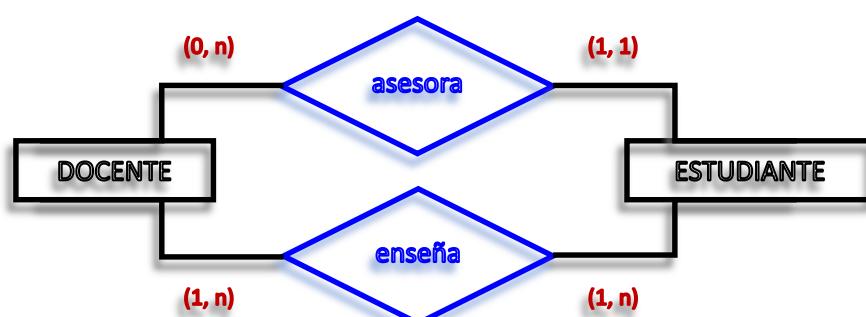
➤ Cardinalidad Máxima

Se utiliza 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.



En el esquema entidad relación anterior, se entiende:

- La cardinalidad de la entidad DOCENTE, en la relación ASESORA, es (0, n), significa que un docente de forma mínima 0 no está obligado a asesorar a un estudiante, como máximo puede asesorar a n estudiantes.
- La cardinalidad de la entidad ESTUDIANTE, en la relación ASESORA, es (1, 1), significa que un estudiante de forma mínima 1 tiene la obligación a tener un asesor docente, como máximo 1 puede ser asesorado por un docente.

- La cardinalidad de la entidad DOCENTE, en la relación ENSEÑA, es (1, n), significa que un docente de forma mínima 1 está obligado a enseñar a un estudiante, como máximo n puede enseñar a n estudiantes.
- La cardinalidad de la entidad ESTUDIANTE, en la relación ENSEÑA, es (1, n), significa que un estudiante de forma mínima 1 tiene la obligación a ser enseñado por un docente, como máximo n puede ser enseñado por n docentes.

Sobre la base de la cardinalidad máxima, inicialmente trabajaremos con conjuntos binarios de relaciones, más adelante trataremos con conjuntos de relaciones n-arias (más de dos entidades). Entonces para un conjunto binario de relaciones R, entre el conjunto de entidades A y B, se entenderán que la relación entre ellos en función de su cardinalidad máxima será:

- **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.

Se debe entender que la cardinalidad adecuada para un conjunto de relaciones determinado, obviamente es dependiente del dominio de problema que el conjunto de relaciones está modelando.

d) 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 clasifican básicamente en simples y compuestos, según que consistan de un solo atributo o cuando se componen de más de dos atributos, respectivamente.

Formalizamos la definición de identificador como sigue:

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.

- i. Unicidad. En cualquier momento dado, no existen dos elementos en E con el mismo valor de I .
 - ii. Minimalidad. Si I es compuesto, no será posible eliminar ningún atributo componente de I sin destruir la propiedad de unicidad.
- Los identificadores se llaman también claves primarias (si es el elegido) o claves candidatas (a ser elegidos de manera alterna).

Caso 1



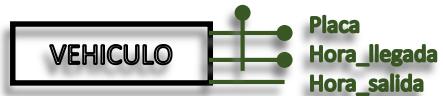
En el gráfico se muestra la entidad PERSONA, con los siguientes atributos: DNI, Nombre, Apellido y Profesión. El atributo DNI es el identificador o clave primaria. No es recomendable considerar el atributo Nombre como identificador, tampoco Nombre y Apellido de manera compuesta, ya que se puede dar el caso de homonimia, violando la unicidad del identificador.

Como se puede notar, el identificador DNI, se le representa fuera del rectángulo de la entidad, mediante una línea conectada al rectángulo y al extremo final un pequeño círculo con el nombre del

atributo clave primaria, el resto de atributos se enlazan a través de una línea simple.

Caso 2

En el caso siguiente:



Se ha definido una clave primaria compuesta $\text{PK}(\underline{\text{placa}}, \underline{\text{Hora_llegada}})$, esto puede ocurrir en una playa de estacionamiento, quienes ofrecen el parqueo vehicular por horas. No se utiliza en solitario el atributo placa como identificador (PK) por cuanto puede ocurrir que el mismo vehículo pueda retornar varias veces en un día, por lo que el diferenciador para cada registro sería la hora de llegada.

EJEMPLO 1

Se requiere una base de datos que contenga el origen de los estudiantes de la Universidad. Para ello 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.

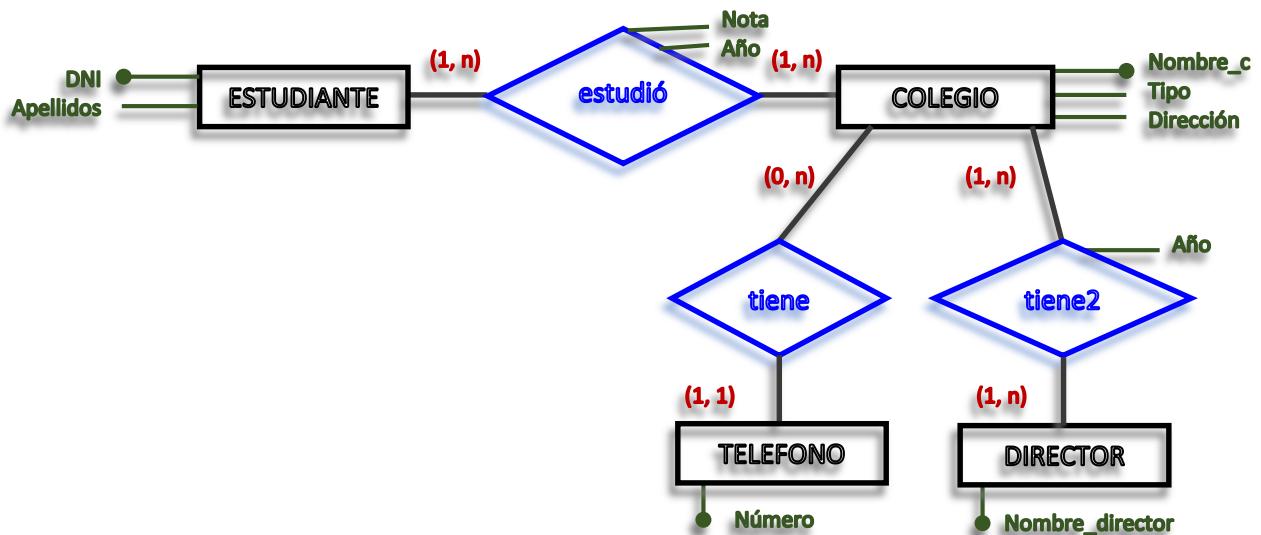
Solución

En el análisis inicial del caso, se debe separar a priori las posibles entidades, relaciones y atributos, luego hacia el final poder fijar las restricciones que salgan de la data recogida, las que se traducen en la cardinalidad entre las relaciones de las entidades.

Entidades	Relaciones	Atributos
<ul style="list-style-type: none"> ✓ Estudiante ✓ Colegio 	<ul style="list-style-type: none"> ✓ Estudió ✓ Tiene (teléfono) 	<ul style="list-style-type: none"> ✓ Datos personales <i>¿?</i> ✓ Nombre_colegio <i>¿?</i> ✓ Tipo colegio ✓ Dirección_colegio ✓ Teléfonos <i>¿?</i> ✓ Director_actual <i>¿?</i> ✓ Nota 1 ✓ Nota 2 ✓ Nota 3 ✓ Puntaje_ingreso

En esta lluvia de ideas inicial, hay interrogantes por diversos motivos, y tiene que ver con la calidad de los datos de los requerimientos obtenidos en la elicitación de los mismos, ¿Qué tan clara fue la obtención de esa data?

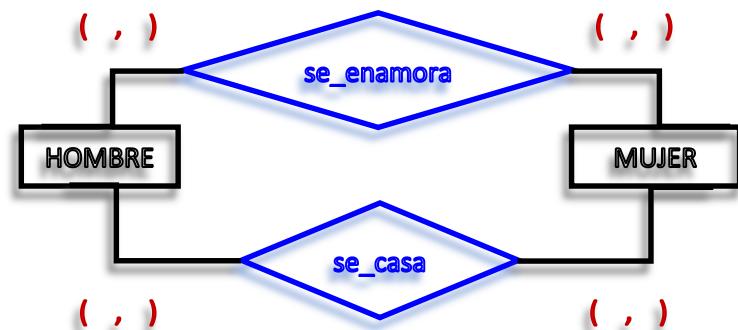
En primer lugar, Datos personales, no se puede representar con un solo atributo los datos personales (nombre, apellidos, DNI, fecha de nacimiento, email, etc.) por conveniencia y por no hacer muy extensa la solución nos quedaremos con DNI y apellidos. En cuanto a Nombre_colegio, también lo asumiremos como una suposición válida, a lo cual agregaremos que los nombres de colegios son únicos, de tal manera que se pueda utilizar como identificador, pero se debe tener en cuenta que no se debe suponer datos, se debe revisar la validez de los datos recogidos. Para efectos didácticos el identificador será una única suposición valida si acaso no aparece explícitamente en el enunciado del caso. Teléfonos, ha sido considerado como atributo, pero no puede haber un campo que guarde más de un dato, pues más adelante colisionará con el modelo relacional, por ello lo consideremos como entidad. De igual manera se ha considerado Director_actual como atributo de colegio, pero hay una particularidad porque nos están solicitando que se recuerde el director actual y esto puede significar que cada año, puede haber un Director distinto, esto ameritaría tener la entidad Director con los añadidos que se incluirán en el diagrama de solución. En cuanto a las notas han sido considerado como 3 atributos, pero se ha omitido el requerimiento en los últimos 3 años.



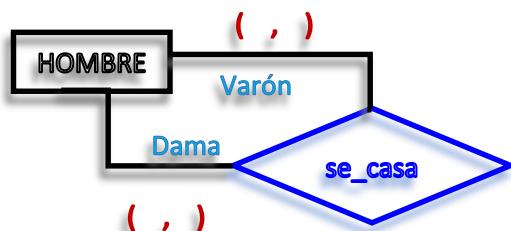
EJERCICIOS

1. Revise la cardinalidad en los siguientes casos

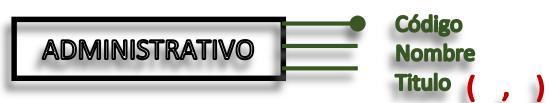
a.



b.



c.



d.



Capítulo III

MODELO ENTIDAD-RELACIÓN EXTENDIDO

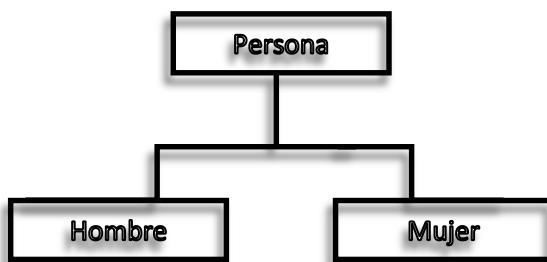
3.1. Jerarquías de generalización

Como parte del MER extendido, las jerarquías de generalización se utilizan para agrupar tipos de entidades parecidas y resaltar sus diferencias, las cuales se resumen con la aplicación de la herencia de atributos, desde el nivel más alto al más bajo.

Por lo tanto, una entidad E es una generalización de un conjunto de entidades E_1, E_2, \dots, E_n (entidades subconjunto), siempre y cuando cada objeto de tipo E_1, E_2, \dots, E_n , es también un objeto del tipo E (entidad genérica) [9].



Todas las propiedades (atributos, relaciones o generalizaciones) de la entidad genérica serán heredadas por las entidades subconjunto. Las entidades subconjunto se justifican porque tienen atributos exclusivos respecto a las otras entidades subconjunto, o porque tienen relaciones exclusivas con otras entidades.



En el gráfico, la entidad PERSONA es una generalización de las entidades Hombre y Mujer, donde ambos subconjuntos tienen características similares y diferenciadoras.

Cobertura de las Generalizaciones

Las jerarquías de generalización presentan ciertas correspondencias entre la entidad genérica y sus entidades, así como entre las sub entidades. Las coberturas pueden ser:

- **Cobertura total o parcial**

Esta cobertura 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.

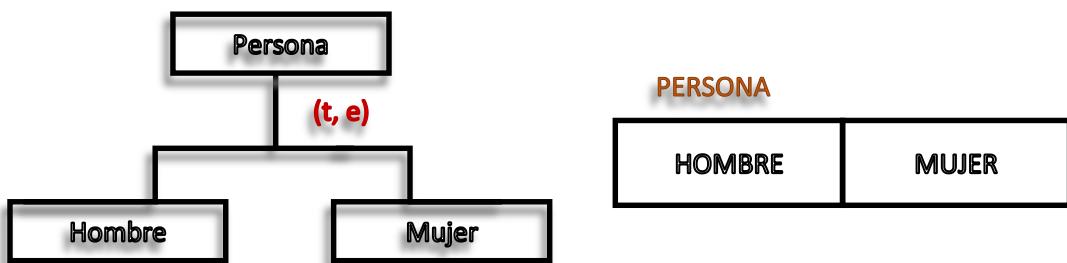
- **Cobertura exclusiva o superpuesta**

Este tipo de cobertura especifica 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.

Debemos verificar que existe ambos tipos de cobertura en una generalización, de arriba abajo entre la súper entidad y sus sub entidades (Cobertura total o parcial), de lado a lado entre las sub entidades (Cobertura exclusiva o superpuesta).

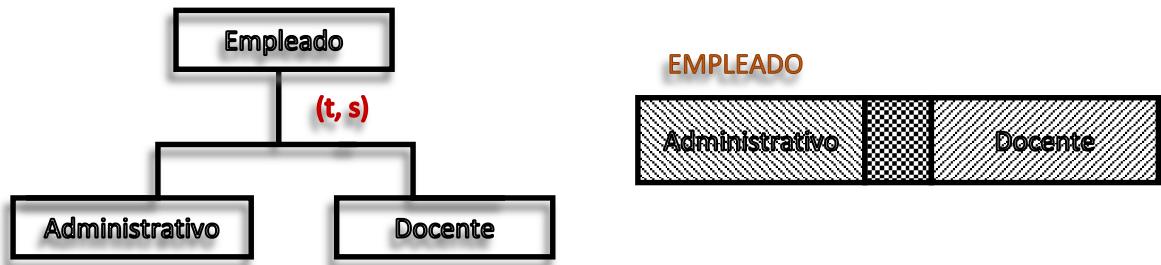
Veamos los siguientes ejemplos:

- **Cobertura total y exclusiva**



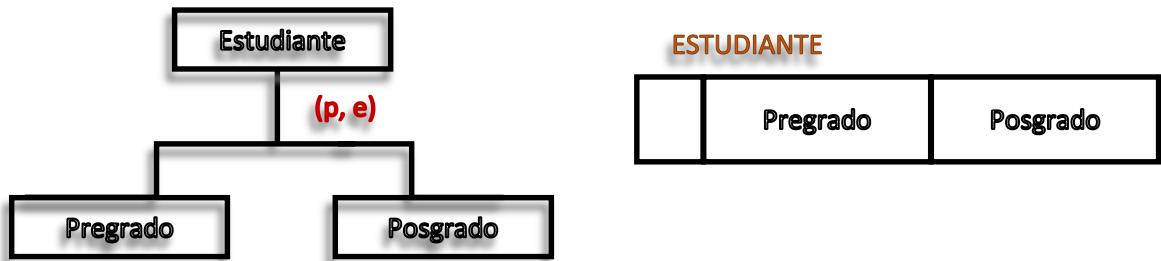
Todas las personas son Hombre y Mujer (total), pero un hombre no puede ser mujer y viceversa (exclusiva).

- Cobertura total y superpuesta



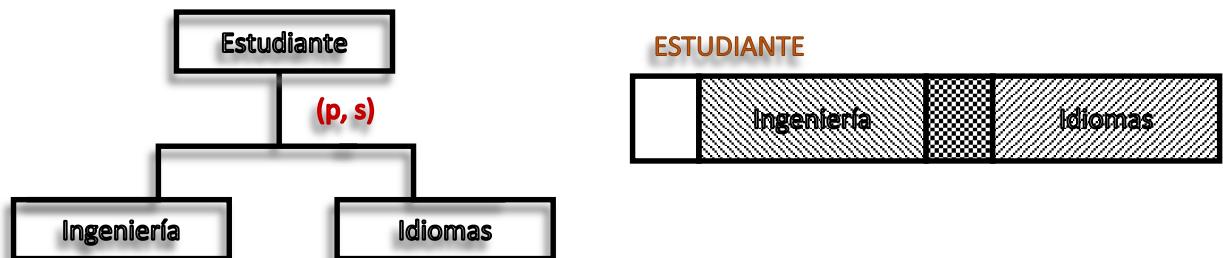
Todos los empleados son administrativos o docentes, pudiendo haber empleados desempeñando ambas funciones (superpuesta)

- Cobertura parcial y exclusiva



Algunos estudiantes son de pregrado y otros de posgrado, pudiendo haber más tipos de estudiantes (cobertura parcial), pero que no son contemplados en el modelo y no existen estudiantes en ambas situaciones (exclusiva)

- Cobertura parcial y superpuesta



Algunos estudiantes son de Ingeniería y otros de idiomas, pudiendo haber más tipos de estudiantes (cobertura parcial), pero que no son contemplados en el modelo y si existen estudiantes en ambas situaciones (superpuesta)

3.2. Cardinalidad en los atributos

Se pudiera considerar que cualquier entidad de una base de datos particular deba tener asociado un solo valor para cada atributo, entendiéndose que luego más adelante el modelo relacional nos reclama que debe existir un solo valor para cada intersección de fila columna correspondiente a un campo o atributo de dicha base de datos. Para efectos de diseño conceptual esto no es obligatorio, y podemos tener un entendimiento rápido de las necesidades de almacenar información de los usuarios en nuestro esquema de alto nivel. De igual manera que con las relaciones, se puede especificar una cantidad mínima y una máxima, de valores de un atributo, que está asociado a una entidad.

➤ **Cardinalidad mínima**

Significa el número mínimo de valores para el atributo en el sentido de opcional u obligatorio:

- 0:** Indica que el valor del atributo es opcional y puede no estar especificado en algunos casos, no es obligatorio tener un valor.
- 1:** Indica que el atributo es obligatorio y al menos un valor debe especificarse para cada instancia de la entidad.

➤ **Cardinalidad Máxima**

Se utiliza para especificar el número máximo de valores para el atributo. Se consideran dos cardinalidades máximas:

- 1:** El atributo es monovalente.
- n:** El atributo es polivalente



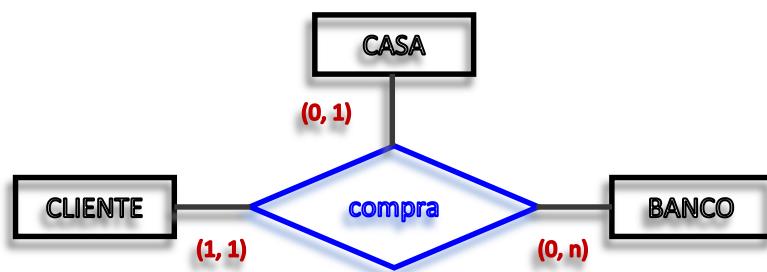
En este ejemplo, la cardinalidad mínima 0 indica que no es obligatorio que algún empleado tenga teléfono, por otro lado, la cardinalidad máxima dice que los empleados pueden tener n muchos teléfonos registrados a su nombre.

Como disciplina, se aconseja al modelador iniciar sus esquemas, ciñéndose a restringir los atributos con exactamente un valor (1,1) por los siguientes motivos:

- Los atributos multi-valorados (cardinalidad máxima n) para su implementación en los SGBD relacionales deben pasar por procesos de normalización.
- Los atributos opcionales (cardinalidad mínima 0) generalmente indican clases especiales de entidades y generan almacenamiento de valores nulos.

3.3. Relaciones n-arias

Estas relaciones son las que conectan tres o más entidades. La lectura de las restricciones de la asociación de las tres entidades vía la cardinalidad se debe dar en el contexto de la relación de las entidades participantes.

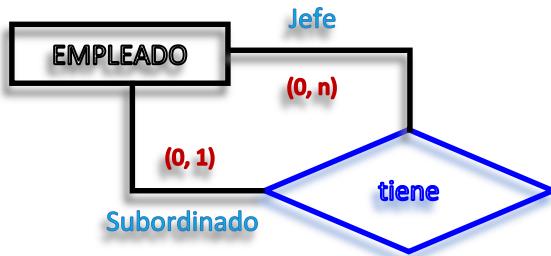


- ❖ En este caso el cliente en el contexto de la compra de una casa a través del banco, lo tiene que realizar atendiendo las reglas del negocio, si el cliente está registrado en esta base de datos la cardinalidad mínima 1 dice que es obligatorio que realice una compra de una casa a través un banco y la cardinalidad máxima 1 que en este entorno solo puede comprar una casa con un banco aliado.
- ❖ La casa de esta lista (base de datos) no es obligatorio que la compre algún cliente-banco por la cardinalidad mínima 0, la cardinalidad máxima n dice que la casa debe ser comprada como máximo una sola vez por una pareja cliente-banco.
- ❖ Un banco no está obligado a participar de la compra por parte de un cliente y una casa en particular por la cardinalidad mínima 0, en cambio la cardinalidad máxima n indica que un

banco puede participar en muchas compras de binomios casa-cliente.

3.4. Relaciones Recursivas

Son las relaciones que conectan una entidad consigo misma.



En el gráfico tenemos la relación recursiva DIRIGE, que conecta a la entidad empleado consigo misma.

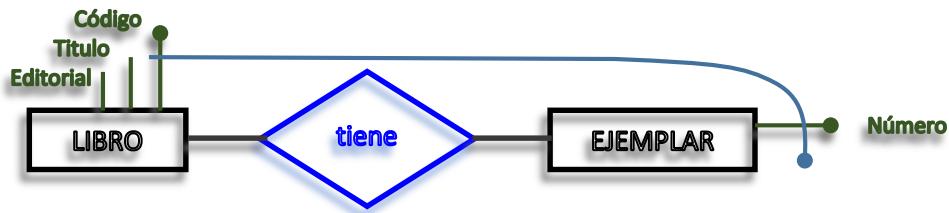
En este tipo de relaciones es necesario tener en cuenta el rol que desempeña la entidad en cada uno de los sentidos de la relación, para de esta manera poder tener una correcta lectura de la cardinalidad de la relación.

En el caso del empleado en el rol de Jefe, la cardinalidad mínima 0 nos dice que no es obligatorio que un empleado tenga subordinados, en tanto la cardinalidad máxima nos dice que el empleado en su rol de Jefe puede tener n (muchos) subordinados. Para el caso del empleado en su rol de subordinado, la cardinalidad mínima 0 indica que no es obligatorio que un empleado tenga Jefe, la cardinalidad máxima 1 nos indica que el empleado en su rol de subordinado debe tener como máximo un Jefe.

3.5. Clasificación de los tipos de entidad según sus identificadores

Existen dos clases de entidades: fuertes y débiles [11].

- **Entidad Fuerte:** Aquella que existen por sí misma, es un tipo de entidad con identificador interno. Ejemplos, la entidad LIBRO, ESCUELA, EQUIPO.
- **Entidad Débil:** Es aquella cuya existencia depende de otro tipo de entidad, es un tipo de entidad con identificador externo o mixto. Como ejemplo, se tiene la entidad EJEMPLAR con respecto a la entidad LIBRO.



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:

Donde FK (Foreign Key, clave foránea) indica que Código es un atributo externo, en este caso de la entidad fuerte LIBRO. El identificador de Ejemplar se dice que es compuesto, porque tiene un atributo interno y otro externo. Por conveniencia las claves primarias han sido puestas en negrita y subrayado para poder diferenciarlos de los atributos comunes. PK (Primary Key, clave primaria).

Ejemplo 1

Se requiere se diseñe el modelo entidad-relación relativo al Campeonato Nacional de Fútbol La Liga 1, de acuerdo con los siguientes requerimientos:

- ✓ Un jugador pertenece a un único equipo y no hay dos jugadores con el mismo nombre. Se recuerda su DNI, nombres, apellidos, puestos en los que se desempeña.
 - ✓ Un jugador puede desempeñarse en diversos puestos dentro del campo, pero en un determinado partido solo puede jugar en un solo puesto.
 - ✓ En cada partido intervienen 11 jugadores, cada uno en un puesto distinto.
 - ✓ Cada partido tiene asignado cuatro árbitros, uno principal y dos jueces de línea y un cuarto hombre como asistente de campo. De ellos se sabe su carnet de árbitro, DNI, nombres y apellidos, celular.

- ✓ Un árbitro puede desempeñar una función en un partido y otra en otro partido.
- ✓ En cada partido participan dos equipos y se registran los goles marcados por cada equipo.
- ✓ Cada partido se juega en un estadio. Del cual se sabe el nombre, ubicación y aforo.
- ✓ Todo estadio pertenece a un equipo y no hay ningún equipo que no tenga estadio.

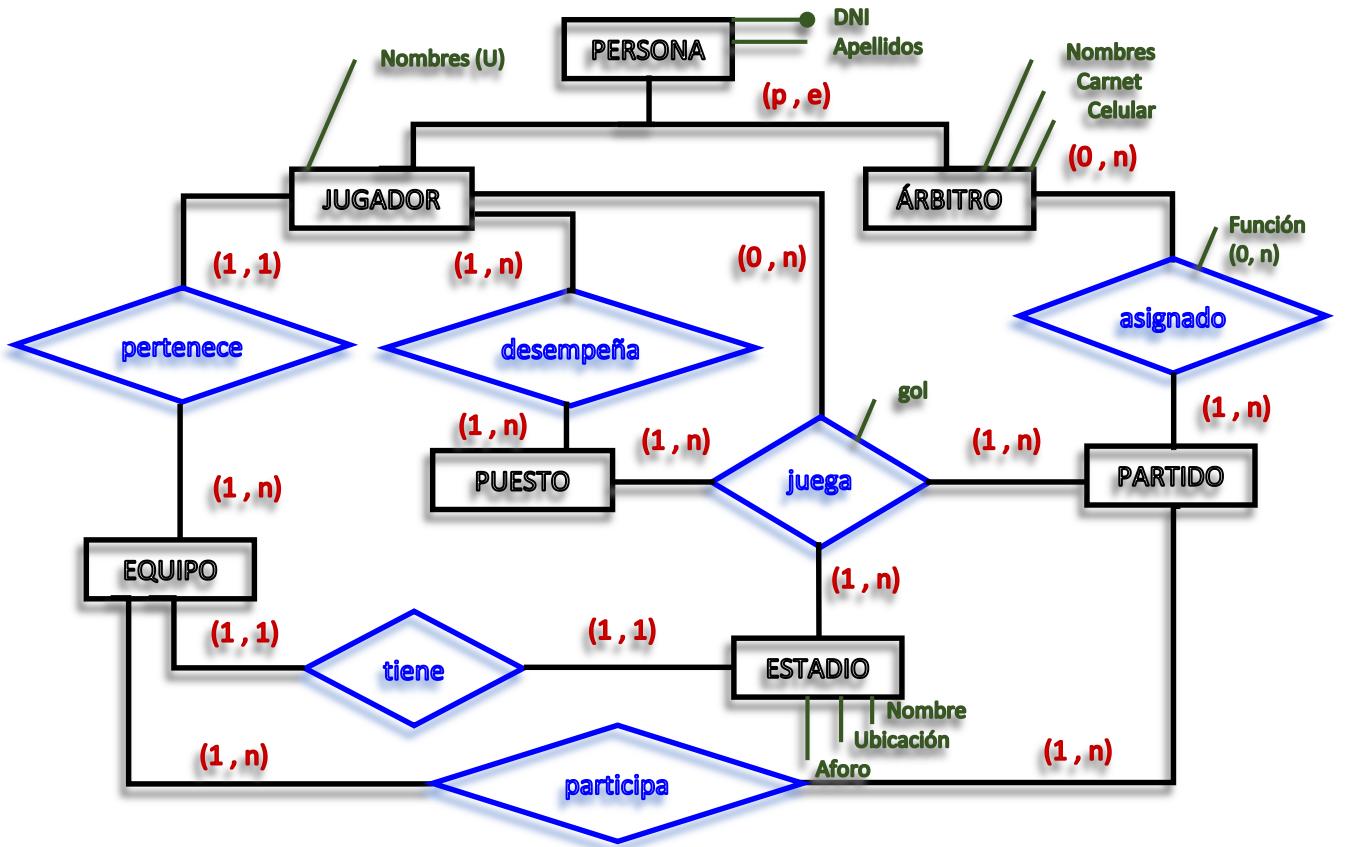
Solución

Separamos entidades, relaciones y atributos, para comprender el contexto del requerimiento

Entidades	Relaciones	Atributos
<ul style="list-style-type: none"> ✓ Jugador ✓ Equipo ✓ Partido ✓ Arbitro ✓ Estadio 	<ul style="list-style-type: none"> ✓ pertenece (jug-equipo) ✓ desempeña (jug-pues) ✓ asignado (arb-part) ✓ participan (part-equipo) ✓ se juega (part-estadio) ✓ pertenece (est-equipo) 	<ul style="list-style-type: none"> ✓ DNI_j ✓ Nombres_j ✓ Apellidos_j ✓ Puesto_j <i>¿?</i> ✓ Carnet_a ✓ DNI_a ✓ Nombres_a ✓ Apellidos_a ✓ Nombre_e ✓ Ubicación ✓ Aforo ✓ Nombre_e

En esta primera revisión se identificaron 5 entidades que explícitamente se muestran en el enunciado, de igual manera se detectan 6 relaciones, de preferencia se debe indicar entre paréntesis que entidades intervienen en la relación, para no confundir relaciones similares, que luego se pueden diferenciar en el esquema; finalmente los atributos, en este caso es conveniente diferenciar los atributos que se parecen añadiéndole al menos un carácter como sufijo para poder identificar la entidad. También notamos que se ha considerado puesto como atributo, pero también forma parte de una relación como entidad, algo que tenemos que resolver.

Preliminarmente, dado que hay 2 tipos de personas: jugador y árbitro, con atributos muy similares, amerita que sean considerados como una jerarquía de generalización, iniciamos con ello:



Hemos tratado de plasmar en el MER resultante los requerimientos del enunciado que se encuentran explícitamente mencionados, en vista que no nos han proporcionado los atributos de algunas entidades, la mejor suposición que podemos hacer es agregarle una PK, para completar los demás atributos debemos volver y reforzar la elicitation de los datos con el usuario, para trabajar sobre la base de la necesidad de guardar datos de ellos y no sobre las suposiciones del analista. En el caso del atributo nombre de la generalización el requisito es que no hay dos jugadores con el mismo nombre, por lo tanto, se les ha considerado por separado, donde el nombre del jugador es considerado con una restricción Único (UNIQUE) a nivel de base de datos.

En base al diagrama resultante, planteamos las siguientes interrogantes necesarias, sí con este diseño:

¿Realmente podemos asegurar que se han plasmado todos los requerimientos de los usuarios?

¿Se consigue restringir la intervención de 11 jugadores en cada partido?

¿Podremos asegurar que solo se asignan cuatro árbitros en cada partido?

¿Podremos restringir la participación de 2 equipos en cada partido?
Revise la cardinalidad de todas las relaciones y verifique si se ajusta al requerimiento del usuario.

Ejemplo 2

El siguiente caso del campeonato de ajedrez es una adaptación de [12], resulta que nuestra Liga Distrital, ha sido encargada por la Federación Nacional de Ajedrez, para organizar los campeonatos nacionales, con sede en nuestra localidad, para ello será necesario mantener una base de datos que permita gestionar participantes, alojamiento y las partidas. Para ello se ha recogido la siguiente data:

En los campeonatos participan jugadores y árbitros, de quienes debemos conservar su número de afiliación, nombre, dirección, teléfono, y campeonatos en los que ha participado (tanto para el jugador como para el árbitro). De los jugadores además se debe conservar su puntuación ELO. Los árbitros no deben participar como jugadores.

Cada Liga distrital de Ajedrez envían a cada campeonato un conjunto de jugadores y árbitros, aun cuando no todas las ligas participan. Todo jugador y arbitro es enviado por una única liga. Aun cuando una liga distrital puede ser representado por participantes de una liga vecina.

Cada liga se identifica por un número correlativo según su orden alfabético, además se debe guardar su nombre, el número de clubes de ajedrez que las integran.

Cada partida se identifica por un numero correlativo, juegan dos jugadores y es arbitrada por un solo árbitro. Se requiere registrar las partidas que juega cada jugador y el color con el que juega

(blancas o negras). Se debe tener en cuenta que un árbitro no puede arbitrar a jugadores de su misma liga de origen.

Todo participante, participa al menos en una partida.

Los jugadores y los árbitros se alojan en uno de los hoteles donde se desarrollan las partidas, se debe recordar en que hotel y en que fechas se ha alojado cada uno de los participantes. Los participantes pueden no permanecer en nuestra localidad durante todo el campeonato, sino acudir cuando tiene que jugar alguna partida alojándose en el mismo hotel o en distinto hotel. De los hoteles se conoce el nombre, dirección y el número de teléfono.

El campeonato se desarrolla a lo largo de una serie de jornadas (año, mes y día) y cada partida ocurre en una de las jornadas, aunque puede que no haya partidas en todas las jornadas.

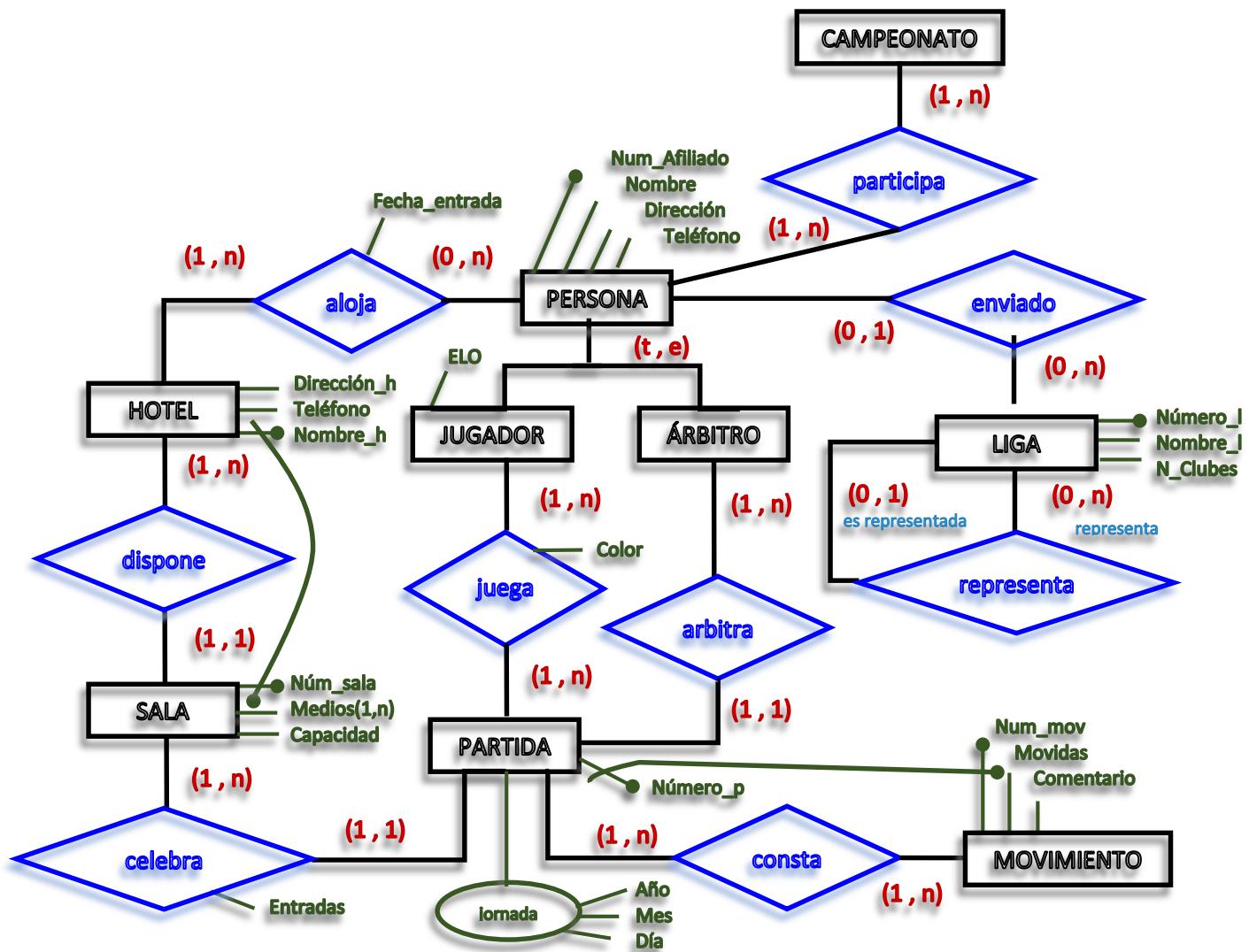
Cada partida se celebra en una de las salas de las que disponen los hoteles, se debe mantener el número de entradas vendidas en la sala para cada partida. De cada sala se conoce su capacidad y medios de los que dispone (TV, radio, video, internet ...) para facilitar la transmisión de los encuentros. Una sala puede disponer de varios medios distintos.

De cada partida se pretende registrar todos los movimientos que la componen, la identificación del movimiento se establece en base a un número de orden dentro de cada partida: para cada movimiento se guarda la jugada (5 posiciones) y un breve comentario realizado por un experto.

Solución

Entendiendo el contexto del requerimiento:

Entidades	Relaciones	Atributos
<ul style="list-style-type: none"> ✓ Jugador ✓ Arbitro ✓ Partida ✓ Movimiento ✓ Hotel ✓ Sala ✓ Campeonato ✓ Liga 	<ul style="list-style-type: none"> ✓ participa (pers-camp) ✓ aloja (pers-hotel) ✓ arbitra (arb-part) ✓ dispone (hotel-sala) ✓ juega (jug-part) ✓ enviado(pers-liga) ✓ representa (liga-liga) 	<ul style="list-style-type: none"> ✓ Num_afiliado ✓ Nombre ✓ Dirección ✓ Dirección_h ✓ nombre_h ✓ Teléfono ✓ Número_1 ✓ Nombre_1 ✓ N_clubes ✓ Num_sala ✓ Medios



Del requerimiento se detecta que la entidad sala es una entidad débil con respecto a la entidad hotel, de igual manera la entidad movimiento es una entidad débil con respecto a la entidad partida, en cualquiera de ambos casos, las entidades débiles, no existirían por si solas. Es posible también a nivel de diseño conceptual utilizar atributos compuesto que representan a un conjunto de ellos, luego más adelante debemos y podemos resolver esta jerarquía de atributos para concordar con el modelo relacional con el almacenaje de valores atómicos. Y también se detecta una relación recursiva, cuando nos indican que una liga puede representar a otra liga.

Quedan algunas interrogantes:

¿A nivel de base de datos podremos controlar que un árbitro no arbitre un parido de un coterráneo?

¿Será necesario una fecha de salida en la relación aloja?

Ejercicios

1. Nuestra Universidad está interesada en tener una base de datos de las investigaciones que nuestros docentes tienen vigentes a partir de la fecha, para ello nos proporcionan sus requerimientos para una solución inmediata:
 - Del proyecto de investigación debemos conservar el nombre, objetivos, línea de investigación, etc.)
 - Los investigadores imprescindiblemente deben contar con su código ORCID, nombre, domicilio, líneas de investigación en las cuales de desempeña.
 - Si el proyecto es ganador de un concurso externo, es necesario saber quién convocó el concurso (FONDECYT, PNIPA, etc.) y si hubo una entidad colaboradora es necesario su nombre, su razón social y giro del negocio.
 - El Vicerrectorado de Investigación, sabe que un proyecto puede formar parte de otro más complejo.
 - Un investigador puede trabajar en varios proyectos a la vez, y en cada uno de ellos desempeñar una función distinta (investigador principal, integrante, consultor, asesor, etc.)
 - Solo se permite que cada proyecto tenga un investigador principal. Y un investigador principal no puede tener esta misma función en otro proyecto a la vez.
 - Las entidades colaboradoras pueden ser de dos tipos: Patrocinadoras (las que aportan la contrapartida de financiamiento de un concurso) y las de colaboración científica (si acaso aportan científicos externos a nuestra universidad).
2. El instituto de investigación de nuestra Universidad tiene proyecto y computadoras disponibles para los usuarios que son de dos tipos: Investigadores e investigadores junior. Cada proyecto es liderado por un investigador y tener varios usuarios como miembros. Las computadoras tienen horarios, y debemos recordar la fecha y hora de inicio. Los investigadores registran las asistencias de los investigadores junior. Los usuarios pueden participar en varios proyectos y deben registrar las reservas de horarios de las computadoras por su trabajo en un determinado proyecto. Los

investigadores junior forma grupos de investigación (semilleros) que son asesorados por un investigador independientemente de los proyectos. Diseñe utilizando el modelo entidad-relación.

3. 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, como 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. Diseñar la base de datos utilizando el modelo entidad-relación.
4. Utilizando el modelo Entidad-relación, diseñe la base de datos de las Elecciones nacionales para presidente y congresistas, 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 quién 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.

Capítulo IV

CASOS MODELO ENTIDAD-RELACIÓN

1.1 Control de redundancias

En los esquemas entidad-relación, pueden ocurrir redundancias no advertidas por el diseñador, y que pueden generar inconsistencias. De acuerdo con [13] un elemento de un esquema es redundante si acaso este puede ser eliminado sin pérdida de semántica.

Se reconoce dos formas de redundancia en el MER, en los atributos (derivados) y en las relaciones.

Los atributos derivados lo hemos visto en capítulo anterior, en este nos centraremos en las redundancias en las relaciones.

Relaciones redundantes

Ocurren cuando la eliminación de dicha relación no significa perdida de semántica, dado que es posible obtener los mismos resultados de asociación por medio de otras relaciones presentes en el esquema de solución.

Podemos anotar una condición de ocurrencia de relación redundante, aunque no única, que esta forme parte de un ciclo, por lo que es necesario verificar el esquema MER, siempre que aparecen estos ciclos.

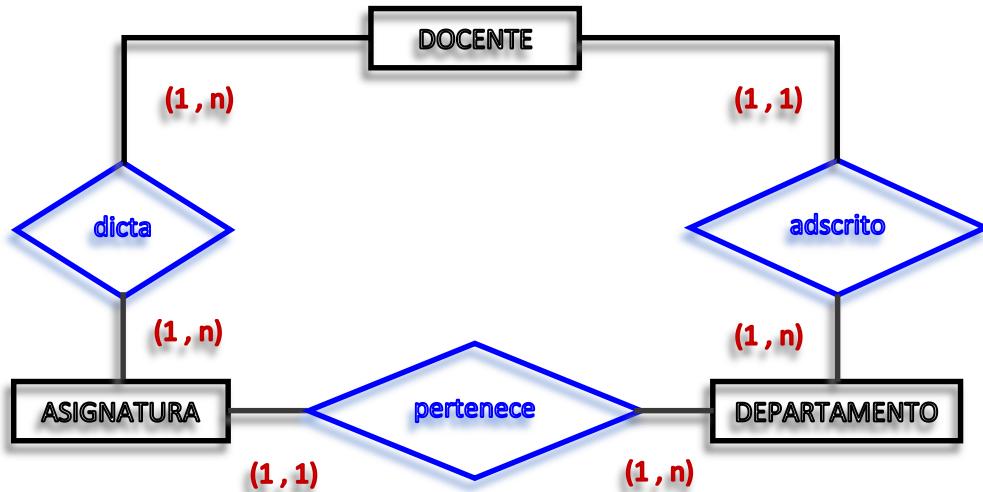
Empecemos analizando el siguiente esquema:



Diga si podremos decir que es una relación redundante.

Como sabemos una de las condiciones es que la relación deba estar insertada en un ciclo y en el esquema que se muestra no hay ningún ciclo, por lo tanto, no hay redundancia.

Veamos el siguiente diagrama que si contiene un ciclo:



¿En este esquema hay alguna relación redundante?

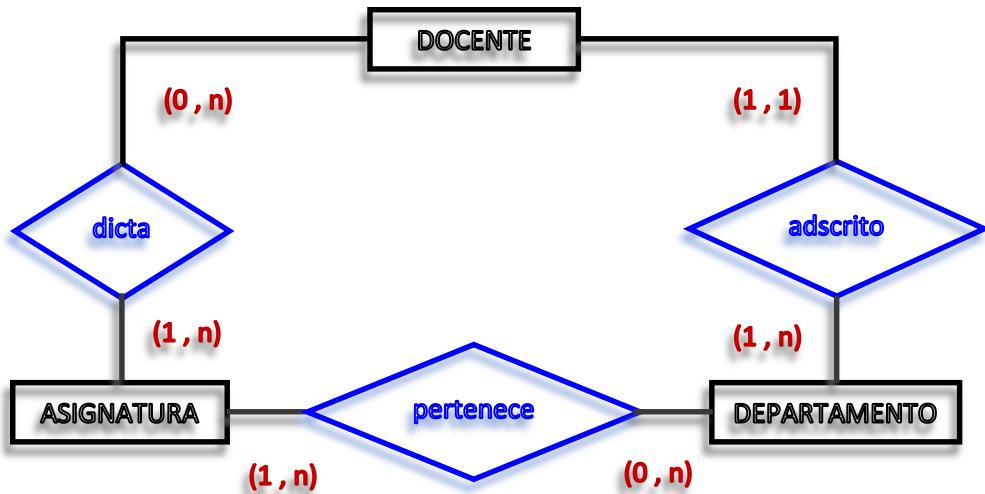
Esto va a depender del contexto de los requisitos, y además de existir un ciclo, debemos revisar en adición las cardinalidades, los tipos de entidades, los atributos y la semántica a reflejar del mundo real.

1º Partiremos del supuesto que los docentes dictan asignaturas a las Escuelas Profesionales, que están registradas (pertenecen) en el departamento al que él está adscrito. Que en el diagrama se evidencia rápidamente.

Por otro lado, si sabemos que asignaturas están registradas (pertenecen) en el departamento, y que profesores dictan tales asignaturas, entonces podremos saber que docentes están adscritos a un determinado departamento. Dada esta circunstancia podremos asegurar que la relación “adscrito” es redundante y su eliminación no producirá perdida de información, es decir podremos obtener el resultado esperado, sin necesidad de agregar una nueva interrelación.

2º Si acaso modificamos el requerimiento, es decir cambiamos la semántica asociada al esquema entidad-relación, como sigue: un departamento puede que no atienda cursos de las Escuelas Profesionales, además un mismo curso puede estar registrado a

más de un departamento, y probablemente haya docentes que no dicten ningún curso. El esquema E-R sería como el que sigue:



Si seguimos con la lógica del ejemplo anterior, probablemente no logremos saber que docentes están adscritos al Departamento, por cuanto, sino es obligatorio que un docente dicte una asignatura, no formara parte de la lista de consulta, de igual manera, en la medida que algún departamento no tenga registrado cursos de las Escuelas Profesionales, se pierde vínculo de relación con asignaturas y con el docente que lo pudiera dictar, generando listas de docentes adscritos a los departamentos con probablemente faltantes. Dado que la relación “adscrito” no se puede deducir de las relaciones que forman parte del ciclo, podemos asegurar que no es redundante.

La relación “dicta”, tampoco es redundante, ya que un curso de las Escuelas Profesionales puede ser dictado por diversos departamentos, que como sabemos pertenecen muchos docentes, por lo que no se puede saber que profesor en concreto dicta una determinada asignatura.

Asimismo, la relación “adscrito” tampoco es redundante, ya que una asignatura dictada por un docente no tiene necesariamente que pertenecer al departamento al que está adscrito el docente, hay departamentos que no tienen cursos registrados (pertenecen) y los docentes pueden colaborar en cursos que pertenecen a otros departamentos distintos a los de ellos.

Es necesario hacer hincapié, en que es posible que la relación pueda ser deducida a partir de las otras relaciones que conforman el ciclo dentro del esquema, no se podrá eliminar si acaso esta relación contiene atributos, que no se pueden derivar a las entidades de la relación.

Remarcamos, para eliminar relaciones redundantes es necesario que exista un ciclo, pero además debe revisarse las cardinalidades mínimas y máximas, la semántica de representación de las relaciones dentro del esquema y comprobar la existencia de atributos.

Resumen

Para eliminar una relación redundante, debe ocurrir:

- Que exista un ciclo
- Que las relaciones que conforman el ciclo sean equivalentes y conserven la semántica de los requerimientos.
- Que se puedan asociar la ocurrencia de las dos entidades que estaban relacionadas, luego de haberse eliminado la relación (redundante).
- Que la relación no contenga atributos, o que estos puedan ser transferidos a las entidades vecinas a fin de no perder semántica.

PowerDesigner

Modelamiento de datos

El modelo de datos es una representación de la información consumida y producida por un sistema, que le permite analizar los objetos de datos presentes en el sistema y las relaciones entre ellos. PowerDesigner [14] proporciona modelos de datos conceptuales, lógicos y físicos que le permiten analizar y modelar un sistema en todos los niveles de abstracción.

Modelo conceptual de datos

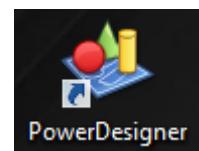
Un modelo de datos conceptual o del inglés “conceptual data model” (CDM) ayuda a analizar la estructura conceptual de un sistema de información, para identificar las entidades principales que se representarán, sus atributos y las relaciones entre ellas. Un CDM es más abstracto que un modelo de datos lógico (LDM) o físico (PDM).

Un CDM permite:

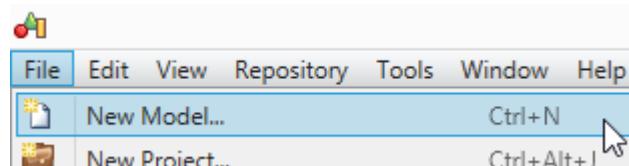
- ✓ Representar la organización de los datos en un formato gráfico para crear Diagramas entidad-relación (DER).
- ✓ Verificar la validez del diseño de los datos.
- ✓ Generar un modelo de datos lógicos o del inglés “Logical Data Model” (LDM), un modelo de datos físicos o “Physical Data Model” (PDM) o un modelo orientado a objetos (OOM), el cual especifica una representación de objeto del CDM utilizando el estándar UML.

Creando un Modelo Conceptual de Datos

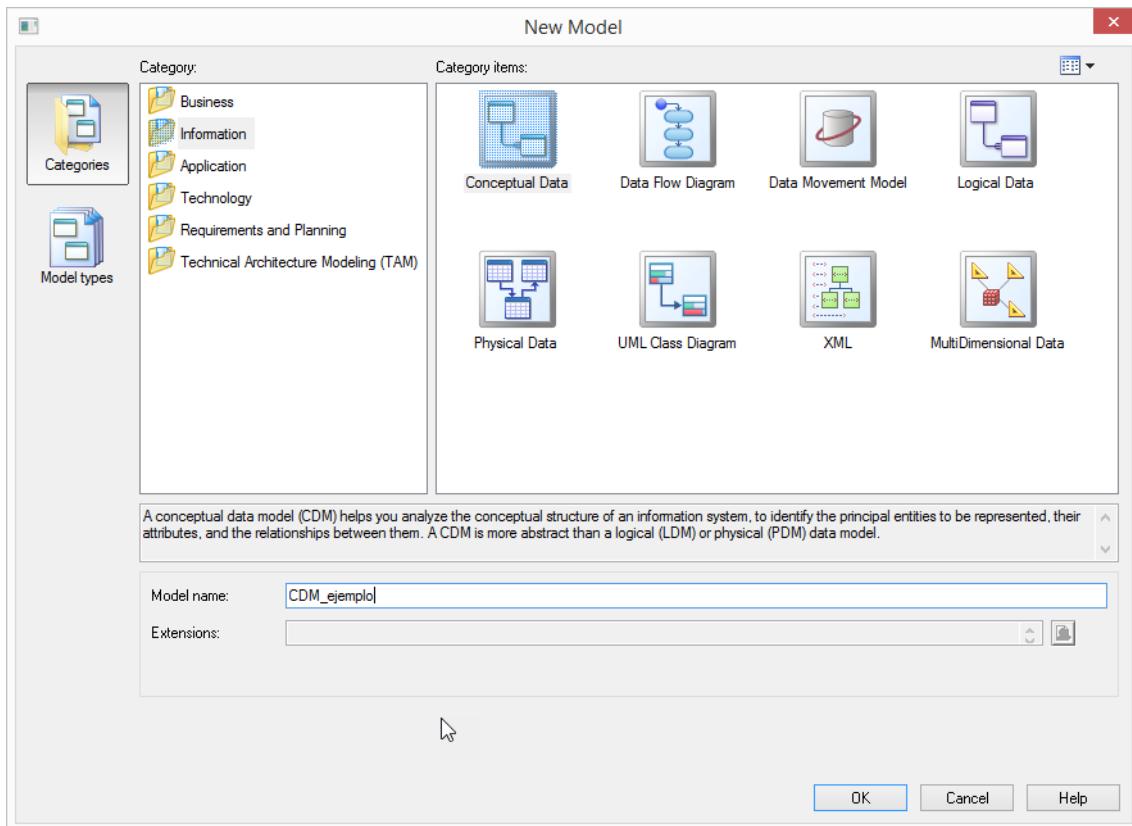
Luego de ejecutar el acceso directo a PowerDesigner



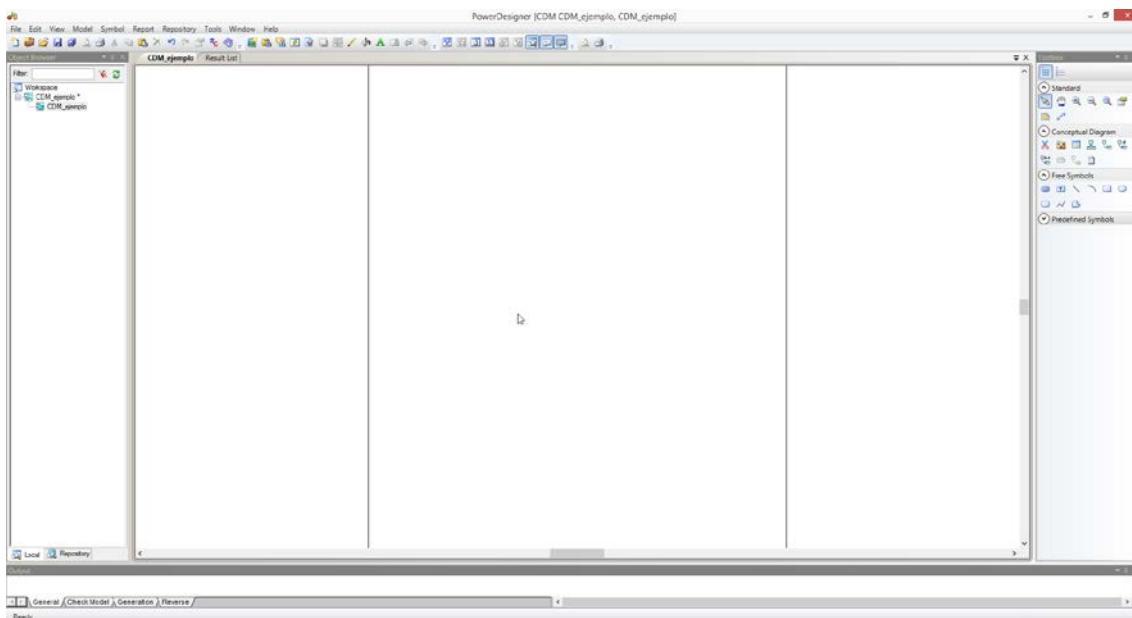
Creamos un Nuevo Modelo



En la caja de dialogo, elegimos Conceptual Data



Y podemos darle nombre al modelo: CDM_ejemplo, luego de aceptar con OK, tendremos acceso al editor de diagramas:

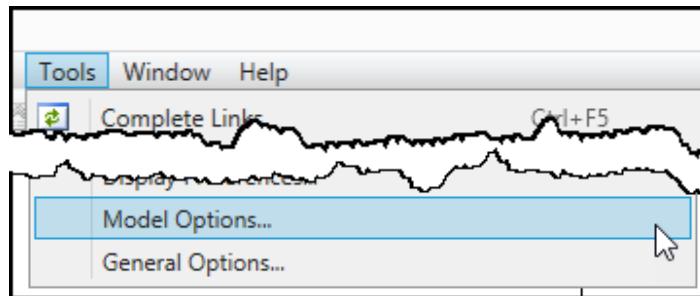


En la parte superior el menú principal, en la columna izquierda el buscador de objetos (Object browser), en la parte central, el editor de

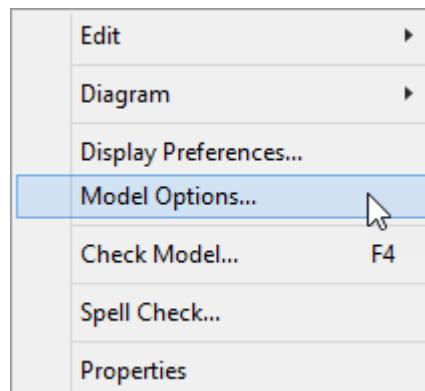
diagramas propiamente dicho, el cual puede sumar pestañas de diagramas y en la columna izquierda la caja de herramientas (toolbox).

Configurando las opciones del modelo

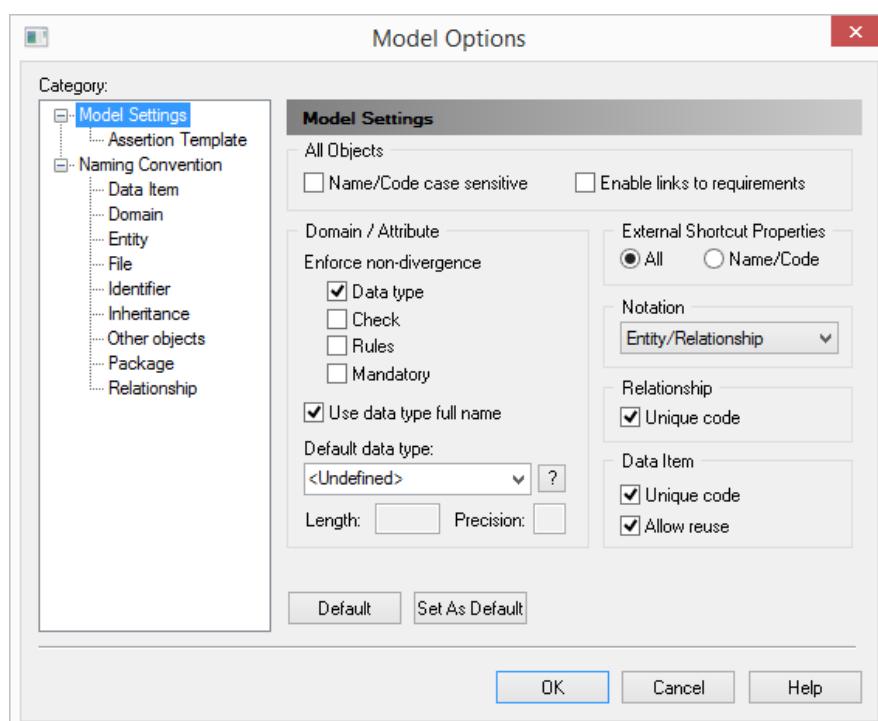
Se puede configurar las opciones del modelo seleccionando del menú principal Tools » Model Options:



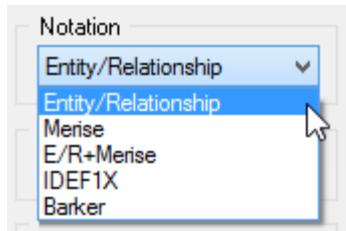
O haciendo click derecho en el fondo del diagrama:



Enseguida se presentan las opciones de configuración del Modelo:

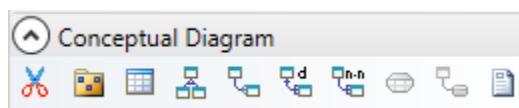


De las cuales en primer lugar remarcaremos la opción de notación:

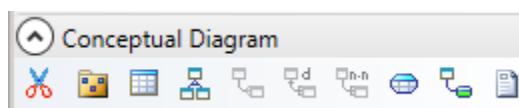


De cuyo menú desplegable se puede elegir entre las siguientes notaciones:

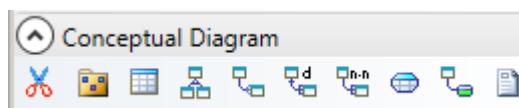
- **Entidad / Relación** [Por defecto].- La notación de entidad / relación conecta entidades con enlaces que representan una de las cuatro relaciones entre ellas. Estas relaciones tienen propiedades que se aplican a ambas entidades involucradas en la relación.



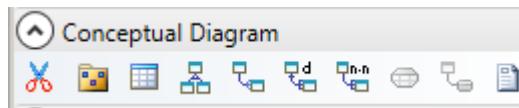
- **Merise**.- usa asociaciones en lugar de relaciones.



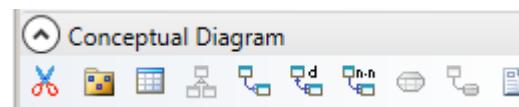
- **E / R + Merise**.- tanto la entidad / relación como Merise se usan en el mismo modelo.



- **IDEF1X**.- notación de modelado de datos para relaciones y entidades. En esta notación, cada conjunto de símbolos de relación describe una combinación de la opcionalidad y la cardinalidad de la entidad a su lado.



- **Barker**.- las herencias se representan colocando las entidades secundarias dentro del símbolo de la entidad principal, y las relaciones se dibujan en dos partes, cada una reflejando la multiplicidad del rol de la entidad asociada.



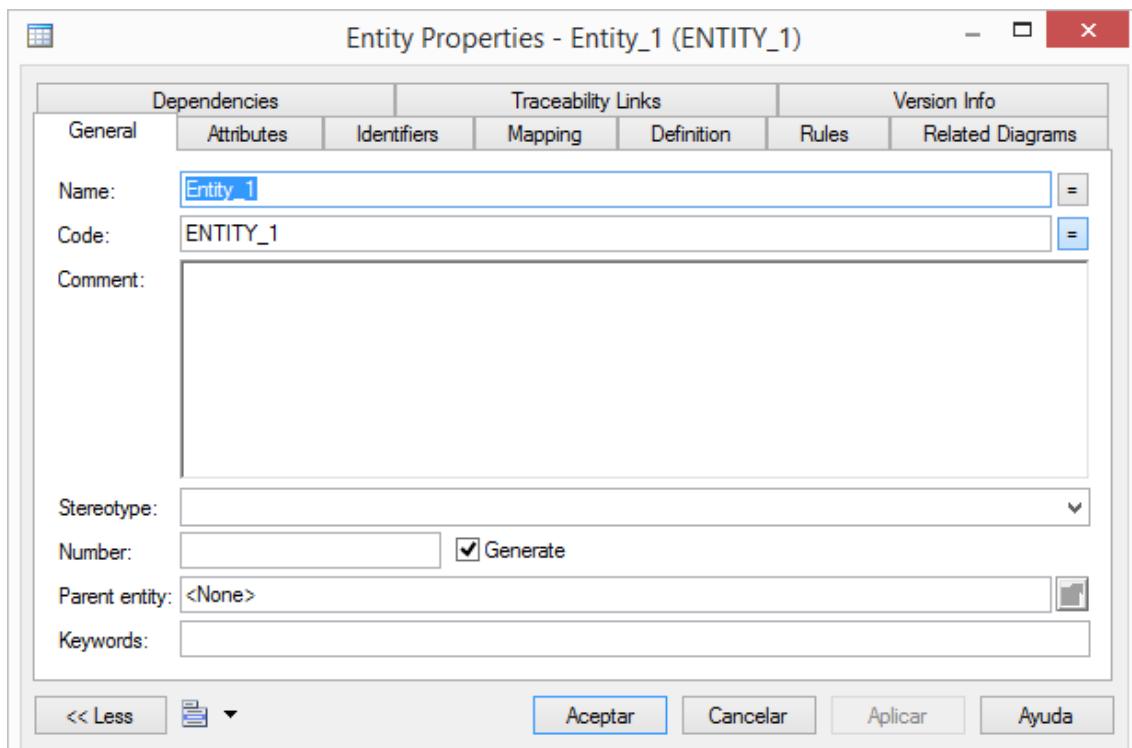
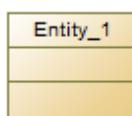
Los símbolos de las notaciones significan:

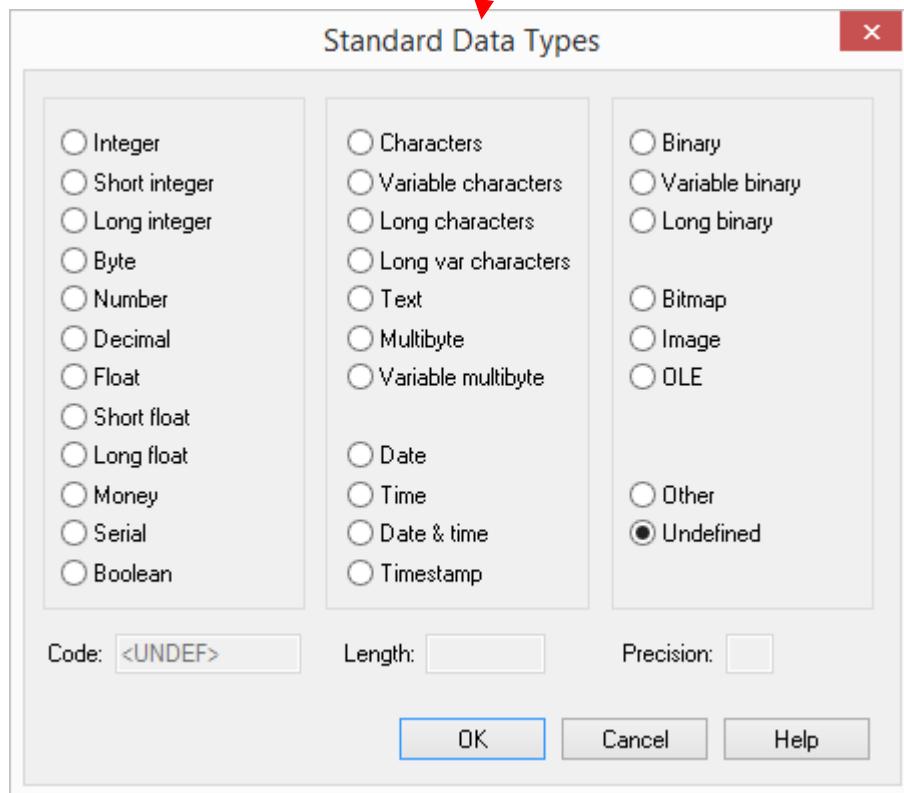
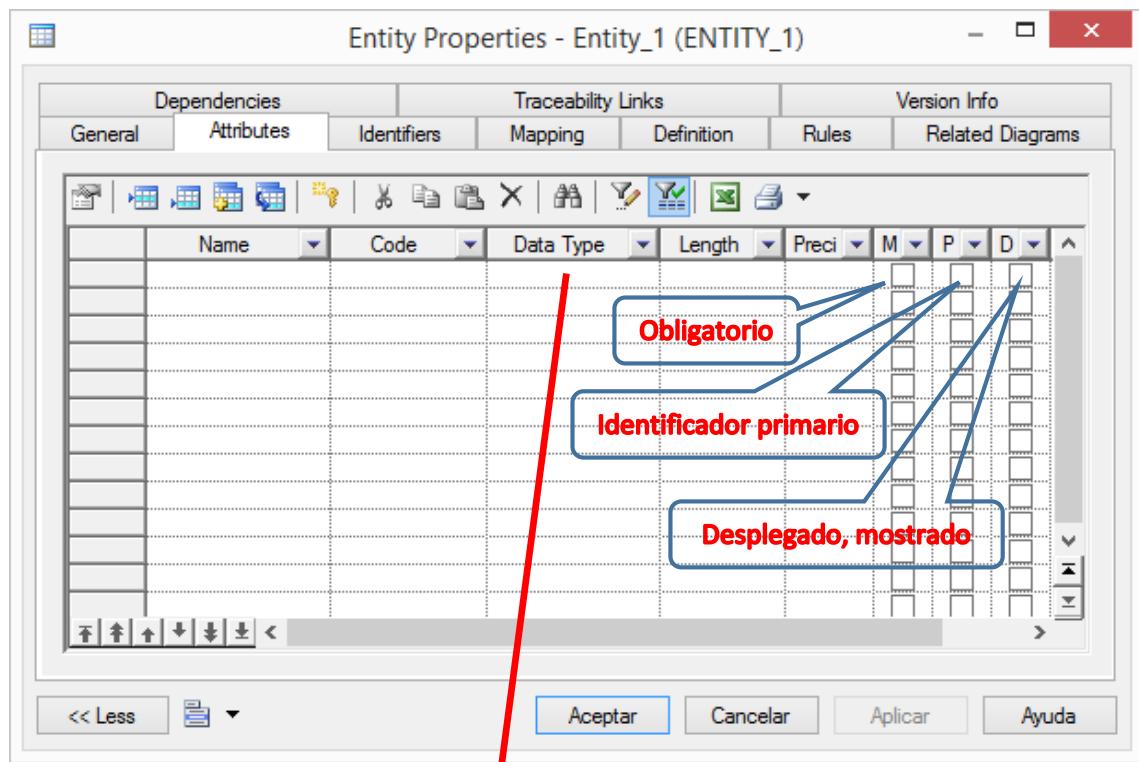
	Entidad: persona, lugar, cosa o concepto que es de interés para la empresa.
	Relación de Uno-muchos, dependiente Uno-Muchos, Muchos-Muchos - Una conexión entre entidades (metodología de modelado ER).
	Herencia: una relación que define una entidad como un caso especial de una entidad más general
	Asociación: una conexión o asociación entre entidades (metodología de modelado Merise).
	Enlace de asociación: un enlace que conecta una asociación con una entidad

Objetos en un CMD

1. Entidad

Persona, lugar, cosa o concepto que es de interés para la empresa. Representa un objeto definido dentro de la organización sobre el que se desea guardar sus datos para interés del sistema de información. Por ejemplo: entidades dentro de la empresa son el Empleado y los Departamentos. Una ocurrencia de una entidad es un elemento perteneciente a la entidad, por ejemplo, Martín es una ocurrencia de la entidad empleado.





2. Relaciones

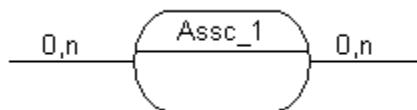
Una relación es un enlace entre entidades. Por ejemplo, en un modelo de gestión de una empresa, entre las entidades Empleado y Equipo puede existir una relación “Miembro” que los vincula y expresa que cada empleado trabaja en un equipo (es miembro de) y cada equipo tiene empleados. Por ejemplo, el empleado Martín trabaja en el equipo de marketing se entenderá como una ocurrencia de la relación “Miembro”.

En PowerDesigner (PD), las relaciones son utilizadas para enlazar entidades en notación ER, Barker e IDEF1X, mientras Merise utiliza asociaciones. Aun cuando PD permite combinarlas o utilizarlas independientemente en el mismo modelo.

Notación para las relaciones en ER

Relación y cardinalidad	Notación gráfica
Uno a Uno (1 .. 1)	
Muchos a Uno (n .. 1)	
Muchos a muchos (n .. n)	
Obligatorio (mandatory)	
Dependiente (dependent)	

Para el caso de Merise, la notación es:



Cardinalidad

Permite especificar la naturaleza de la relación entre una entidad y otra, indica el número de instancias (ninguno, uno o muchos), tal como se ve en el cuadro anterior.

Convenciones de la notación:

Punto terminación	Existencia	Cardinalidad	Descripción
	Obligatorio	Uno	Debe existir uno y solo uno.
	Obligatorio	Muchos	Debe existir uno o más.
	Opcional	Uno	Puede existir uno o ninguno.
	Opcional	Muchos	Puede existir uno o más, o ninguno.
	Obligatorio	Uno	Dependiente con respecto a uno del otro.
	Obligatorio	Muchos	Dependiente con respecto a muchos del otro.

Herencia

Define una entidad como un caso especial de una entidad más general. La entidad general o supertipo (o principal) contiene todas las características comunes, y la entidad de subtipo (o secundaria) contiene solo las características particulares. Las entidades que participan en una herencia tienen muchas características similares, pero sin embargo son diferentes.

No hay un objeto de herencia separado en la notación de Barker, ya que las herencias se representan colocando un símbolo de entidad encima de otro. Las herencias de Barker siempre son completas y se excluyen mutuamente, y el supertipo enumera sus subtipos en la pestaña Subtipos.

Por ejemplo, la entidad de Cuenta representa todas las cuentas bancarias en la base de datos. Hay dos subtipos: cuentas corrientes y cuentas de ahorro.

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

Asociación

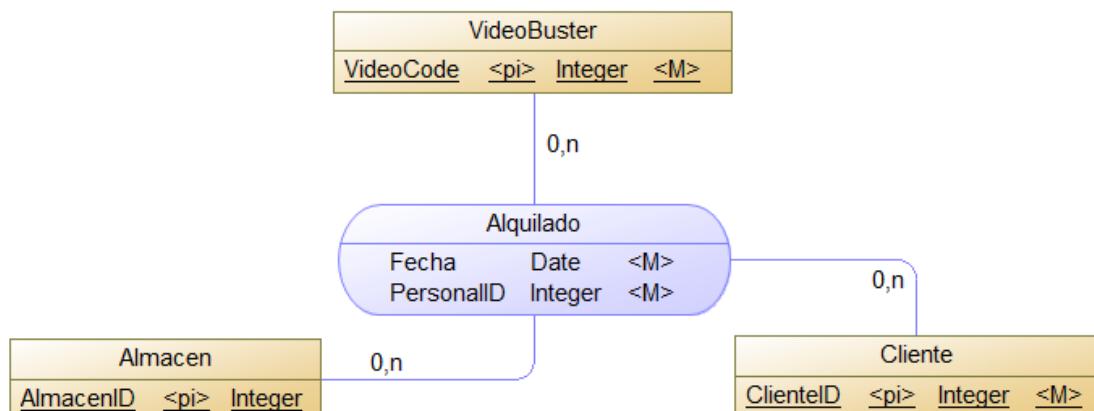
Es una relación entre las entidades. En la metodología de modelado Merise, una asociación se usa para conectar varias entidades, cada una de las cuales representa objetos claramente definidos, pero están vinculadas por un evento, que puede no estar tan claramente representado por otra entidad.

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

Cuando genera un PDM a partir de un CDM, las asociaciones se generan como tablas o referencias.

Ejemplo:

Tres entidades VIDEOBUSTER, CLIENTE, y ALMACEN, contiene datos de videos, clientes y almacén. Ellos están vinculados por una asociación que representa a una cinta de video de alquiler (ALQUILADO). La asociación ALQUILADO también contiene los atributos Fecha y PersonalID, que dan la fecha del alquiler, y la identidad del empleado que alquila la cinta de video.

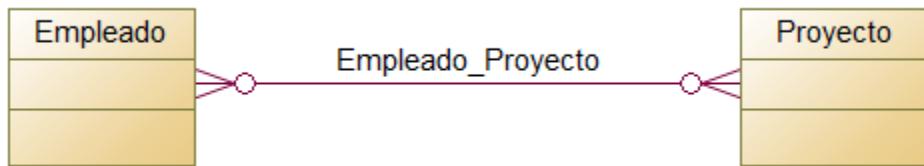


Ejemplos

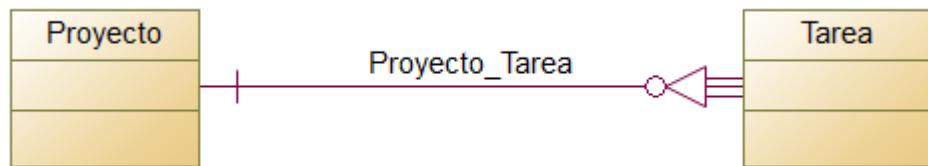
- a) Relación de 1 a muchos: vincula una instancia de la primera entidad a varias instancias de la segunda entidad. En este ejemplo, cada departamento puede contener muchos empleados y cada empleado puede pertenecer a un departamento:



- b) Relación de muchos a muchos: vincula varias instancias de la primera entidad a varias instancias de la segunda entidad. En este ejemplo, cada empleado puede trabajar en múltiples proyectos y cada proyecto puede tener múltiples empleados trabajando en ello:



- c) Relación dependiente de uno a muchos: vincula una instancia de la primera entidad a varias instancias de la segunda entidad y especifica que cada una de las muchas entidades secundarias depende de la primera entidad y está parcialmente identificada. En este ejemplo, cada proyecto puede contener múltiples tareas y cada tarea debe pertenecer a un proyecto:

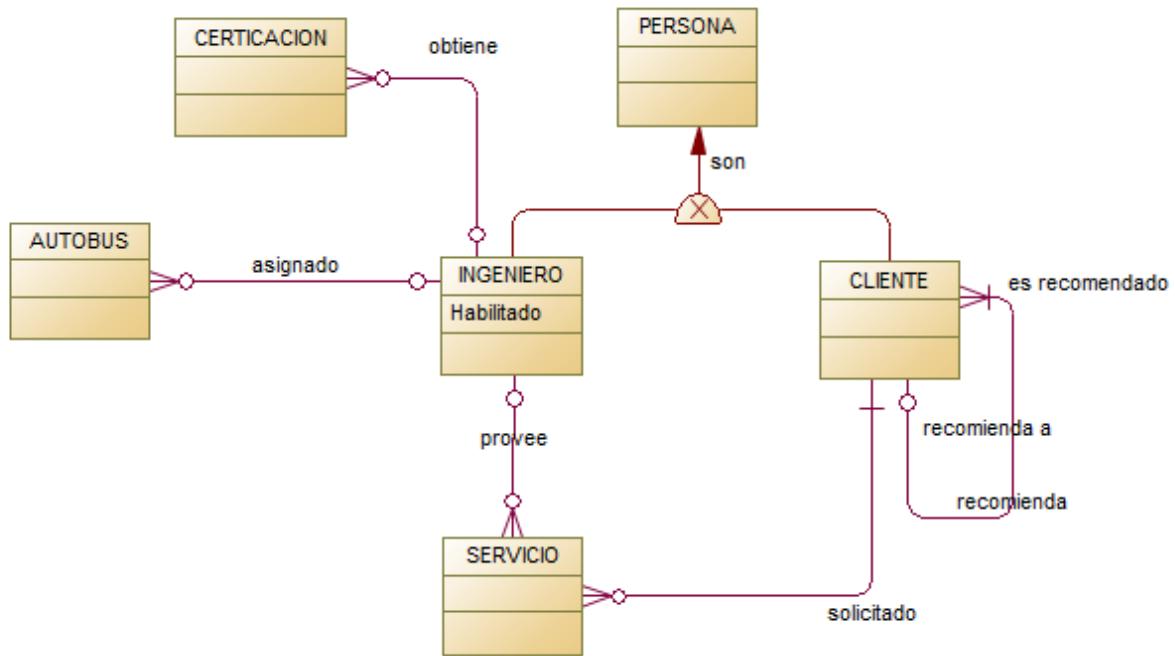


- d) Se tiene el siguiente diagrama ER



Práctica

Observar su corrección



EJERCICIO

- 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 base datos utilizando el MER y elabore su CDM en PowerDesigner.

Capítulo V

DISEÑO CONCEPTUAL DE BASES DE DATOS

El diseño de un esquema conceptual de base de datos es el resultado de un análisis complejo de los requerimientos del usuario [9]. 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. Las primitivas de diseño conceptual se clasifican como descendentes y ascendentes las cuales se adecuan diversas estrategias de diseño, una estrategia de diseño completamente descendente permite refinar los conceptos abstractos y convertirlos en concretos, la estrategia ascendente en sentido inverso de los concreto a lo abstracto.

Primitivas del diseño conceptual

Para el diseño de datos utilizaremos un conjunto de primitivas de refinamiento que utilizaremos para efectuar transformaciones, que aplicadas a un esquema inicial tendrá como producto un esquema resultante.

Se reconocen dos tipos de primitivas: descendentes y ascendentes.

a) Primitivas descendentes

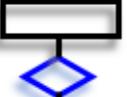
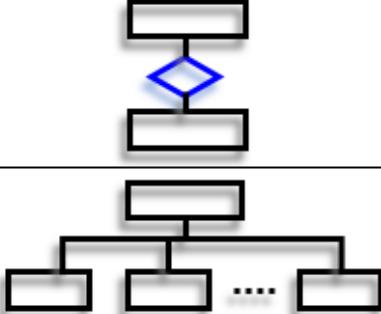
Estas logran refinamientos puros, aplicados a un concepto simple en el esquema inicial, produciendo una descripción más detallada del concepto en el esquema resultante.

Poseen las siguientes propiedades:

- ✓ Tienen estructura simple, inician con un concepto único y resultan en conjunto de conceptos.
- ✓ Todos los nombres se refinan, consiguiendo nuevos nombres que describen el inicial con un nivel de abstracción más bajo.

- ✓ Las conexiones lógicas deben ser heredados por un solo concepto en el esquema resultante.

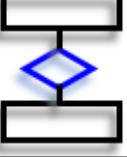
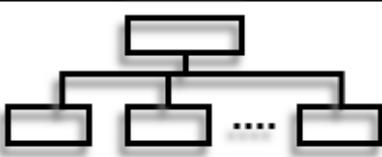
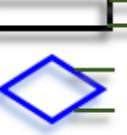
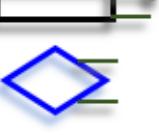
A continuación, presentamos la clasificación de las primitivas descendentes:

Nº	Primitiva	Esquema Inicial	Esquema Resultante
D-1	Entidad → Entidades relacionadas		
D-2	Entidad → Generalización		
D-3	Entidad → Entidades no relacionadas		
D-4	Relación → Relaciones paralelas		
D-5	Relación → Entidad con relaciones		
D-6	Desarrollo de atributos		
D-7	Desarrollo de atributos compuestos		
D-8	Refinamiento de atributos		

b) Primitivas ascendentes

Estas primitivas permiten introducir conceptos nuevos 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).

Clasificación de las primitivas ascendentes:

Nº	Primitiva	Esquema Inicial	Esquema Resultante
A-1	Generación de entidad		
A-2	Generación de relación		
A-3	Generación de Generalización		
A-4	Aggregación de atributos		
A-5	Aggregación de atributo compuesto		

Estrategias para el diseño de esquemas

En capítulos anteriores, hemos presentado el Modelo Entidad-Relación para el diseño de esquemas de base de datos, de manera bastante libre, que probablemente en el azar de la representación, dejamos sin querer semántica no representada, porque no tenemos un orden, trabajamos de manera intuitiva. Para poder obtener un esquema ER que cumpla con las cualidades de un buen esquema, es necesario que empleemos alguna estrategia que nos dé el orden necesario para poder representar todos los requerimientos de la organización.

Entonces para lograr un esquema conceptual, debemos seguir una estrategia, la cual, mediante iteraciones, y partiendo de un esquema inicial, logra sucesivamente esquemas más refinados y obtener un esquema final que cumpla con los requerimientos en su totalidad.

Se distinguen cuatro estrategias para el diseño de Esquemas: Estrategia descendente, ascendente, centrífuga y mixta. Cada una se caracteriza por el uso de tipos particulares de primitivas.

Enseguida se 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. Para ello es necesario partir del dominio de aplicación e ir refinando hasta llegar a un esquema final.



Ejemplo

Se requiere diseñar la base de datos de una empresa hotelera, la cual tiene 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, quienes deben realizar la limpieza de las habitaciones, de quienes se conoce su DNI y nombres. Se requiere controlar la disponibilidad de las habitaciones, las identidades de los huéspedes que ocupan dichas habitaciones (DNI, pasaporte o carnet de extranjería, nombres), y las fechas de cuando entraron y cuando salieron, y quien es el que cancela el alquiler de dicha habitación, también es necesario conocer la programación de la limpieza de las habitaciones a cargo de las camareras. Utilice la estrategia descendente y presente los esquemas entidad-relación correspondientes (adaptado de web).

Solución

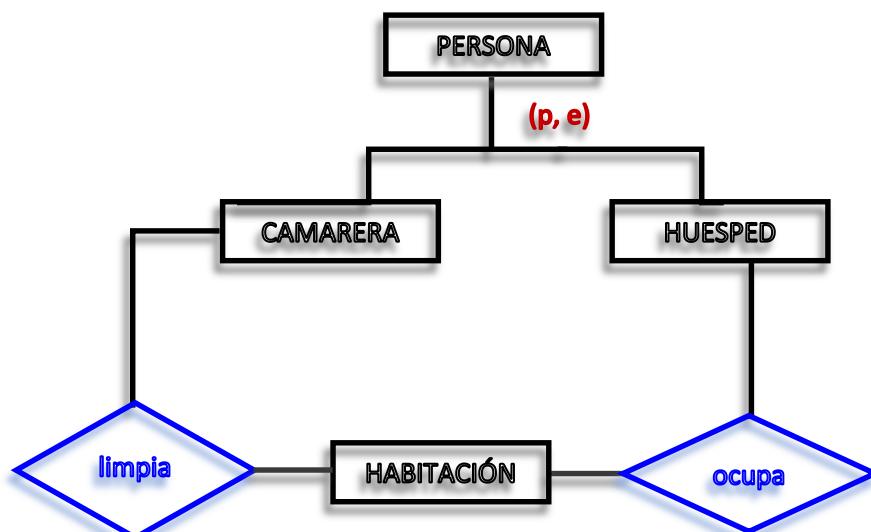
1^{er} Refinamiento



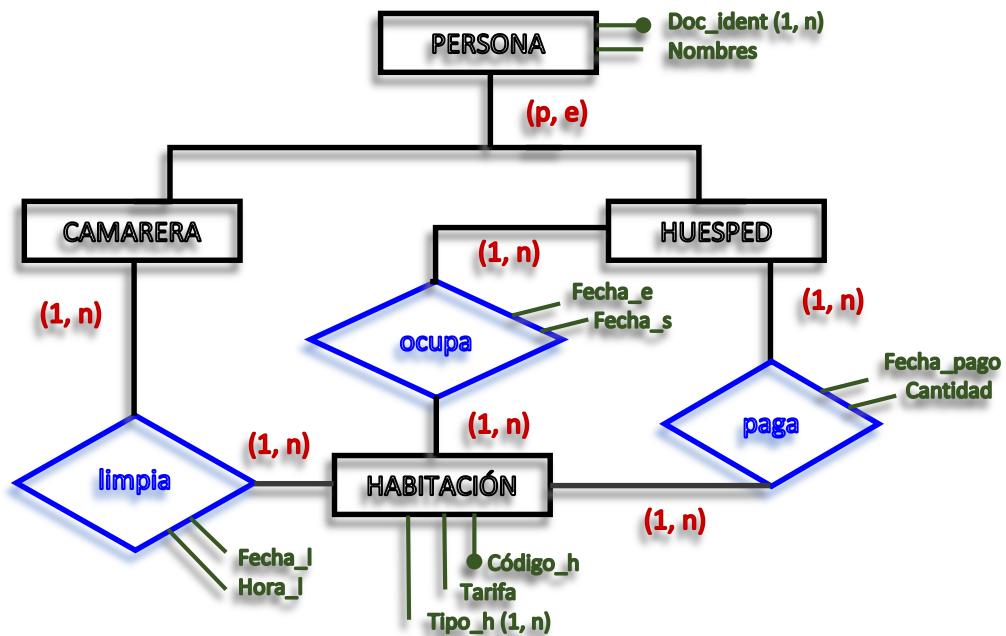
2^{do} Refinamiento



3^{er} Refinamiento



Esquema Final



B. Estrategia ascendente

Con esta estrategia, 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.



Ejemplo

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. (Ejemplo adaptado de [9])

Solución

1° Dominio de aplicación: Base de datos demográfica

En este requerimiento en particular se nos brinda dos posibles sinónimos: “base de datos de un censo”, elegimos uno. En muchos requerimientos tendremos que abstraer el dominio de aplicación por cuenta propia, con el entendimiento del contexto de la organización expresados en los datos que ellos disponen y no necesariamente nos indiquen el dominio de aplicación.

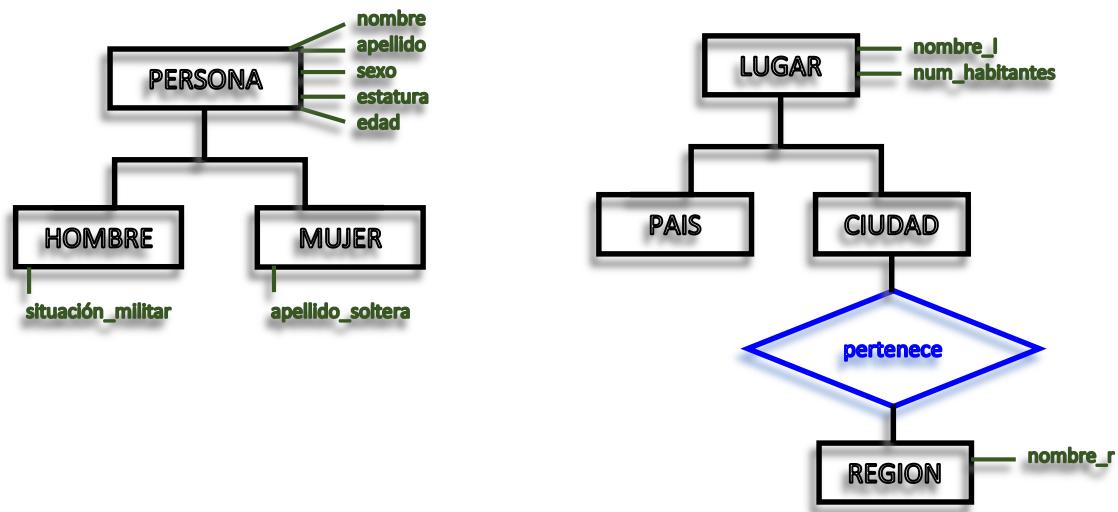
2° Primer Esquema (Sin estructuras, solo atributos)

- nombre	- lugar nacimiento (?)	- nombre l
- apellido	- lugar residencia (?)	- num habitantes
- sexo	- años residencia	- nombre región
- estatura	- situación militar h	
- edad	- apellido soltera m	

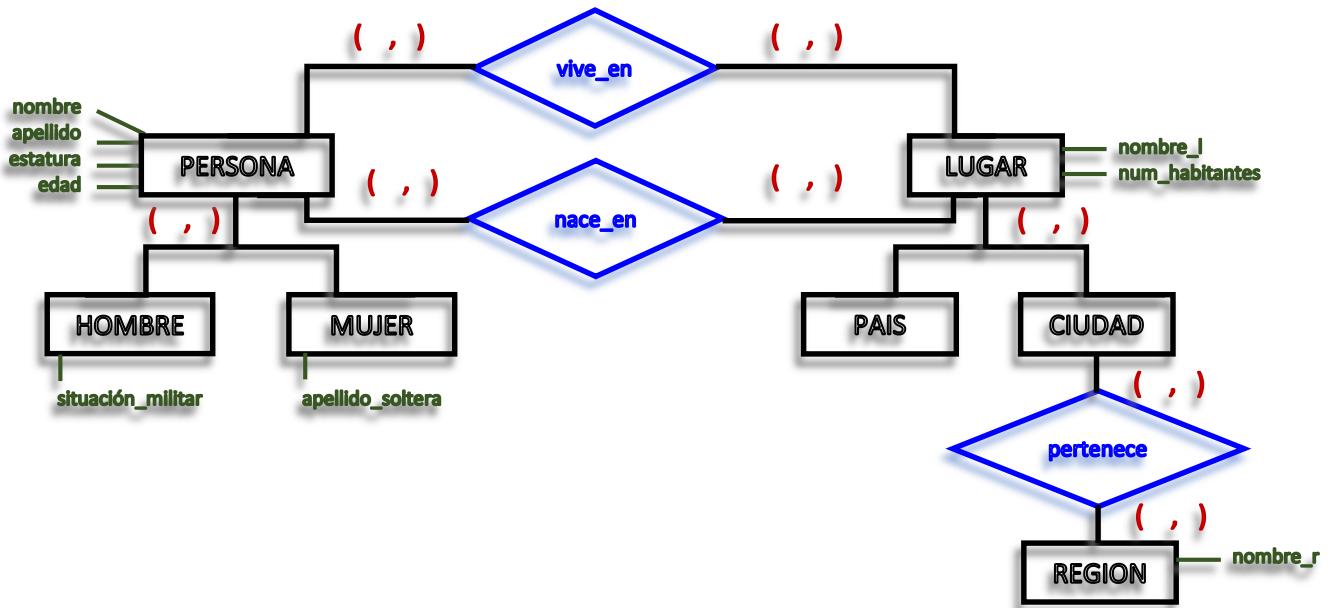
3° Segundo Esquema (Abstracción sobre los atributos:)



4° Tercer Esquema (Abstracción de generalización:)

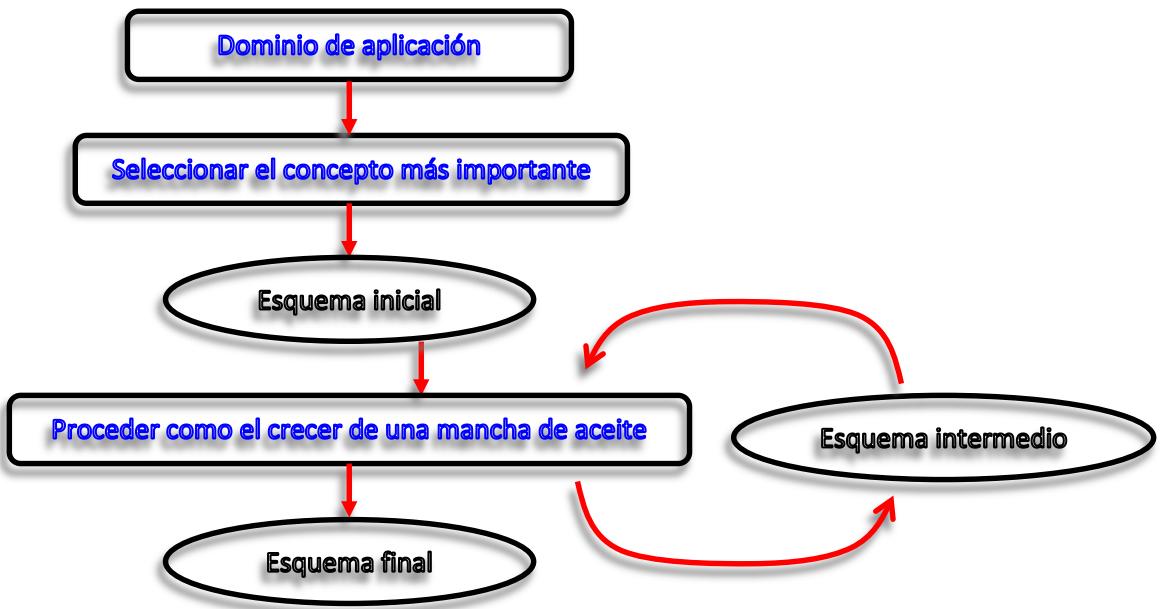


5° Esquema final (adición de relaciones, cardinalidad, cobertura, identificadores)



C. Estrategia centrífuga

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.



Ejemplo

Se quiere diseñar una base de datos para facilitar la gestión de Torneo de Tenis Grand Slam, de acuerdo con la siguiente:

La base de datos debe almacenar todos los encuentros que se han desarrollado desde que existe el torneo, así como las siguientes características de estos.

Descripción:

El Grand Slam se compone de cuatro torneos anuales que se celebran en Gran Bretaña (Campeonato de Wimbledon), Estados Unidos (Abierto de Estados Unidos), Francia (Torneo de Roland Garros) y Australia (Abierto de Australia.)

En cada país se pueden desarrollar en distintos lugares (p. ej., en EE. UU. Puede desarrollarse en Forest Hill o en Flushing Meadows).

Cada partido tiene asociado un premio de consolación para el perdedor que dependerá de la fase en que se encuentre el torneo (p. ej., el perdedor de octavos de final puede ganar 5.000 dólares). El ganador de la final recibirá el premio correspondiente al torneo.

Cada torneo tiene cinco modalidades: Individual masculino, individual femenino, dobles masculino, dobles femenino y dobles mixtos.

También hay que tener en cuenta la nacionalidad de un jugador, de forma que este puede ser apátrida o tener varias nacionalidades.

Resultados a considerar:

Los datos almacenados en la BD deben dar respuesta a las siguientes preguntas:

1. Dado un año y un torneo, composición y resultado de los partidos.
2. Lista de árbitros que participaron en el torneo.
3. Ganancias percibidas en premios por un jugador a lo largo del torneo.
4. Lista de entrenadores que han entrenado a un jugador a lo largo del torneo y fechas en las que lo hizo.

Ejemplos de acceso a la base de datos.

1. Connors gano Gerulaitis en Roland Garros en 1979 en cuartos de final en individuales masculinos por 6-3 4-6/7-5 6-0.
2. El señor Wilkinson arbitró ese partido.
3. Alemania ha ganado dos veces las individuales masculinas de Wimbledon. Borg ha ganado \$2.000.000 de dólares a lo largo de su participación en el Grand Slam.
4. El ganador de Roland Garros de 1987 ganó 20.000 dólares.
5. Noah ha jugado cuatro veces en dobles mixtos con Mandlikova.

Diseñe un modelo de datos entidad/relación completa, empleando la estrategia centrífuga (adaptado de web).

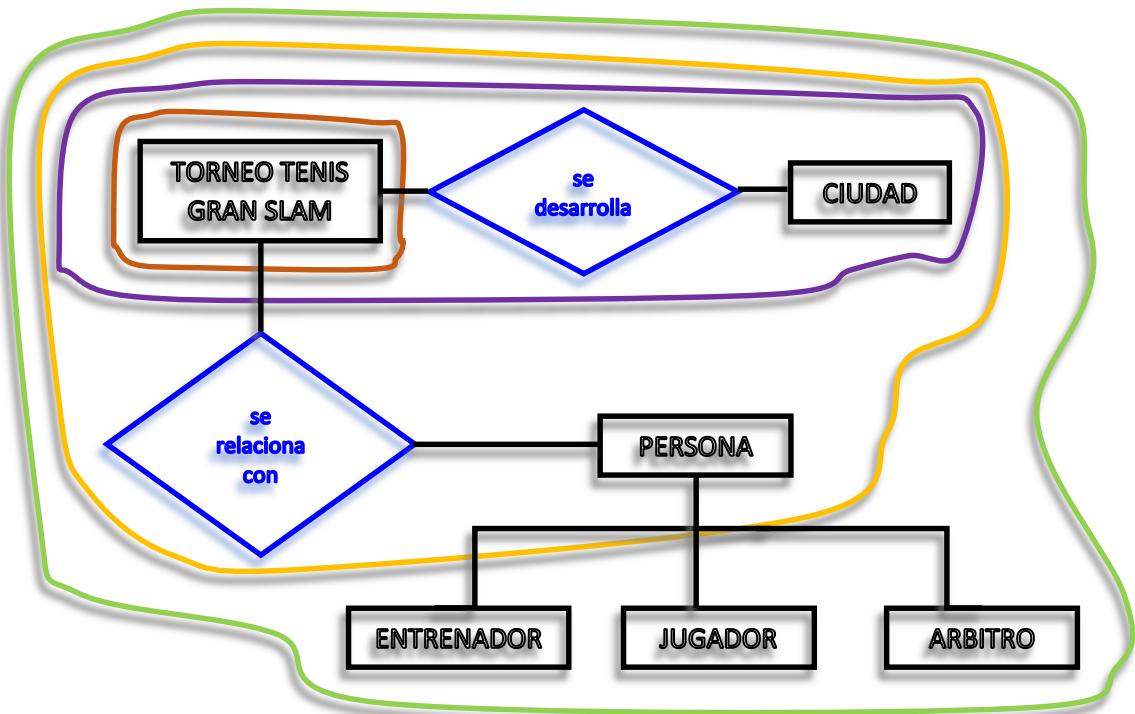
Solución

a) Esquema Inicial (desde el dominio de aplicación)

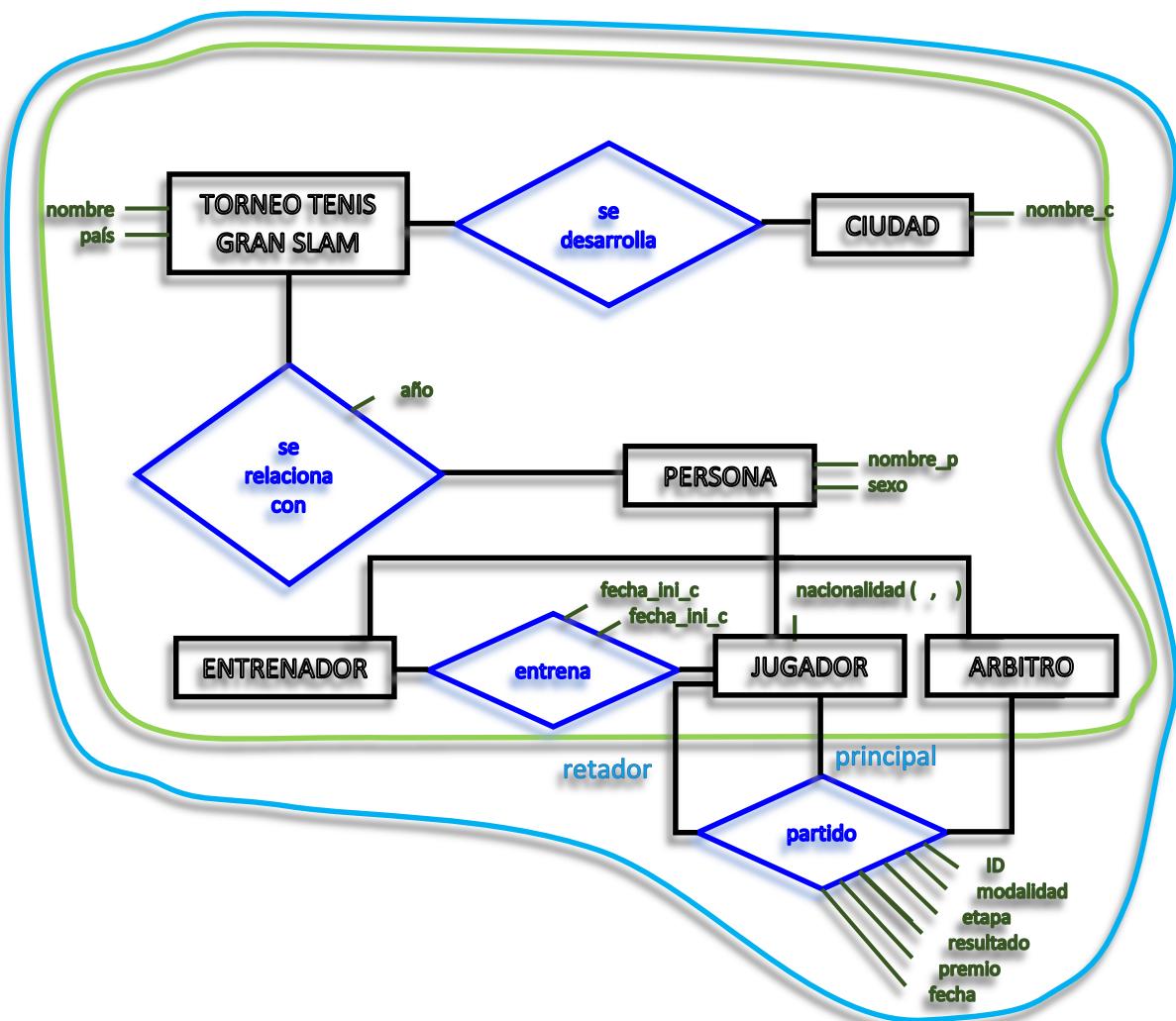
El concepto más importante:



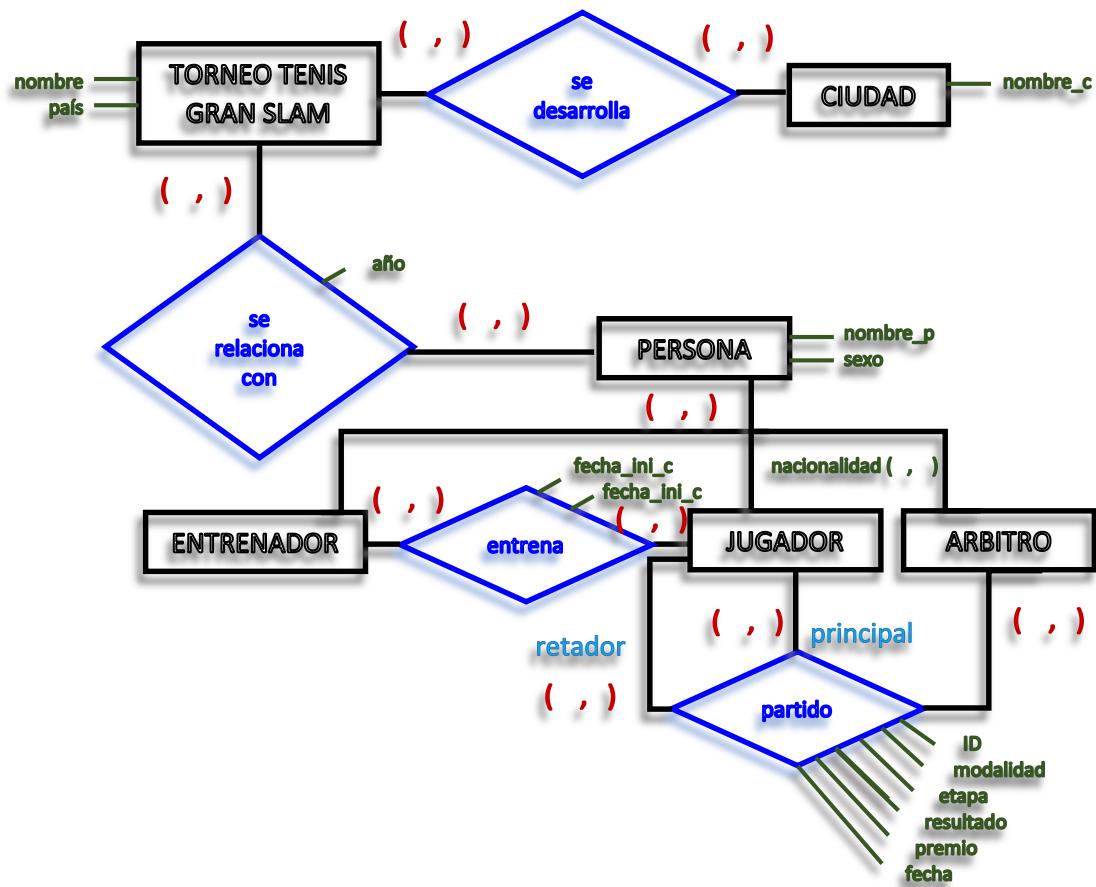
b) Esquema Intermedio 1



c) Esquema Intermedio 2



d) Esquema Final



D. Estrategia mixta

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



Ejemplo

Se trata de diseñar una base de datos para la gestión del personal de una entidad bancaria determinada que dispone de muchos empleados y de una amplia red de agencias. La siguiente descripción resume los requisitos de los usuarios de la futura base de datos:

- a. Los empleados se identifican por un código de empleado, y también deseamos conocer su DNI, su NSS, su nombre y su apellido. Será importante registrar su ciudad de residencia, considerando que hay ciudades donde no reside ningún empleado.
- b. Interesa saber en qué ciudades están ubicadas las diversas agencias de la entidad bancaria. Estas agencias bancarias se identifican por la ciudad donde están y por un nombre que permite distinguir las agencias de una misma ciudad. Se quiere tener constancia del número de habitantes de las ciudades, así como de la dirección y el número de teléfono de las agencias. Se debe considerar que la base de datos también incluye ciudades donde no hay ninguna agencia.
- c. Un empleado, en un momento determinado, trabaja en una sola agencia, lo cual no impide que pueda ser trasladado a otra o, incluso, que vuelva a trabajar en una agencia donde ya había trabajado anteriormente. Se quiere tener constancia del historial del paso de los empleados por las agencias.
- d. Los empleados pueden tener títulos académicos (aunque no todos los tienen). Se quiere saber qué títulos tienen los empleados.
- e. Cada empleado tiene una categoría laboral determinada (auxiliar, oficial de segunda, oficial de primera, etc.). A cada categoría le corresponde un sueldo base determinado y un precio por hora extra también determinado. Se quiere tener constancia de la categoría actual de cada empleado, y del sueldo base y el precio de la hora extra de cada categoría.
- f. Algunos empleados (no todos) están afiliados a alguna central sindical. Se ha llegado al pacto de descontar de la nómina mensual la cuota sindical a los afiliados a cada central. Esta cuota es única para todos los afiliados a una central determinada. Es necesario

almacenar las afiliaciones a una central de los empleados y las cuotas correspondientes a las diferentes centrales sindicales.

g. Hay dos tipos de empleados diferentes:

- o Los que tienen contrato fijo, cuya antigüedad queremos conocer.
- o Los que tienen contrato temporal, de los cuales nos interesa saber las fechas de inicio y finalización de su último contrato.

Si un empleado temporal pasa a ser fijo, se le asigna un nuevo código de empleado; consideraremos que un empleado fijo nunca pasa a ser temporal. Todo lo que se ha indicado hasta ahora (traslados, categorías, afiliación sindical, etc.) es aplicable tanto a empleados fijos como a temporales.

h. Los empleados fijos tienen la posibilidad de pedir diferentes tipos preestablecidos de préstamos (por matrimonio, por adquisición de vivienda, por estudios, etc.), que pueden ser concedidos o no. En principio, no hay ninguna limitación a la hora de pedir varios préstamos a la vez, siempre que no se pida más de uno del mismo tipo al mismo tiempo. Se quiere registrar los préstamos pedidos por los empleados, y hacer constar si han sido concedidos o no. Cada tipo de préstamo tiene establecidas diferentes condiciones; de estas condiciones, en particular, nos interesaría saber el tipo de interés y el periodo de vigencia del préstamo.

Diseñe un modelo de datos para estas entidades y atributos empleando la estrategia Mixta (adaptado de web).

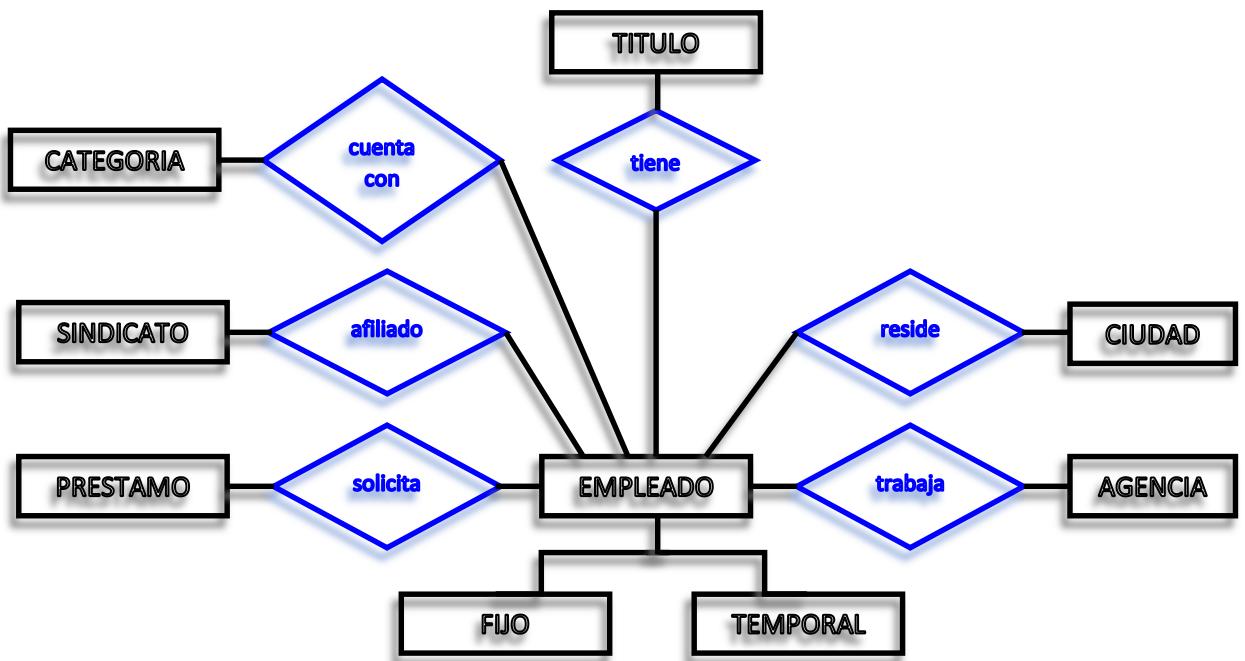
Solución

Dominio de aplicación:

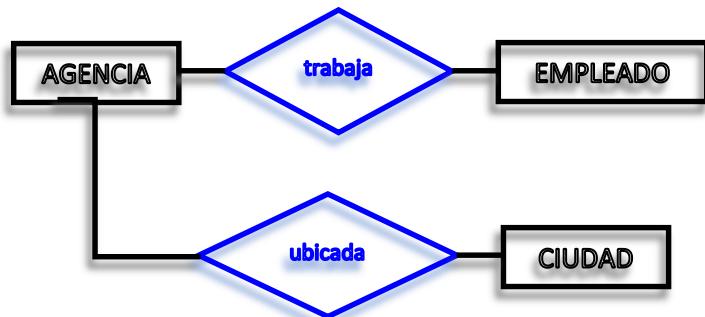
a) Esquema armazón



b) Esquema Empleado

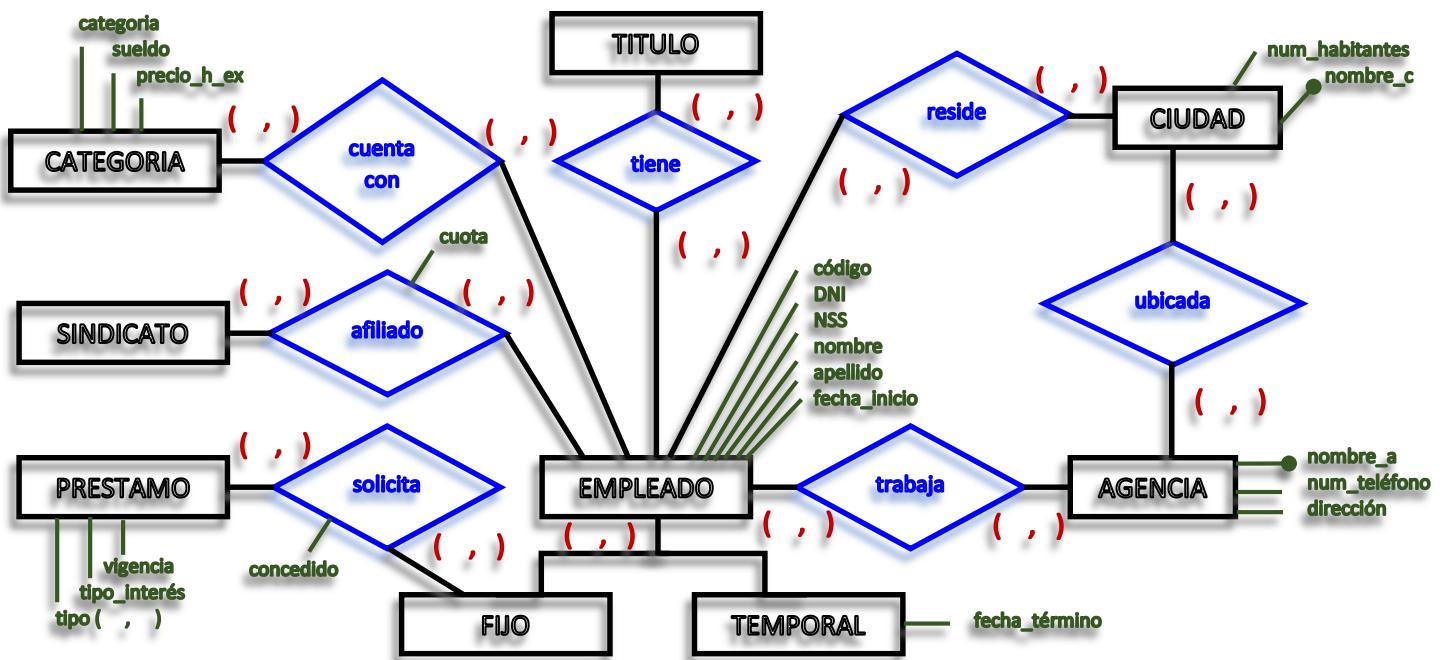


c) Esquema Agencia



d) Esquema Integrado (final)

**** en la página siguiente ****



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:

- **Complejidad.** El esquema es completo cuando representa todas las características pertinentes del dominio de la aplicación. La complejidad 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.

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 en la metodología propuesta por Batini/Ceri/Navathe, consiste de las siguientes actividades que se muestran el diagrama:



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, extensibilidad y normalidad, las cuales fueron explicadas anteriormente.

Capítulo VI

CASO DE DISEÑO CONCEPTUAL DE BASES DE DATOS

6.1 Diseño conceptual a partir de los requerimientos en lenguaje natural

Hay varias formas de obtener los requerimientos de los usuarios: partiendo de los expresados en lenguaje natural, dependiendo de la organización de sus formularios predeterminados y probablemente de esquemas de datos previos.

La primera forma (lenguaje natural) es la más 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 metodológicamente se pueden resumir así:

a) Análisis de los requerimientos

- ❖ Es necesario filtrar los requerimientos obtenidos por lenguaje natural, de tal manera que se filtren las ambigüedades para obtener los requerimientos de la base de datos, de manera clara, y de común entendimiento para los diversos usuarios de la misma. 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 de tiempo y de número de años.
 - ✓ No usar casos cuando se requieren conceptos más generales. Por ejemplo, un usuario de una empresa electrónica puede pedir el inventario de chips, el término chip no es un concepto, sino que es un caso del concepto correcto: componente. Definitivamente que debemos tener cuidado del contexto en el cual queremos implementar la base de datos y por consiguiente más adelante un sistema de información

- ✓ Se debe verificar la no ocurrencia de sinónimos y homónimos, pues pueden confundir a los usuarios.
- ✓ Es bastante recomendable utilizar un glosario de términos, que más adelante será parte de su diccionario de datos
- ❖ Para cualquier estrategia de diseño, es altamente recomendable dividir los enunciados de los requerimientos en conjuntos homogéneos, que permiten obtener el conjunto de características sobre conceptos susceptibles de convertirse en entidades e inclusive relaciones.

b) Diseño Inicial

Dependiendo de la estrategia a utilizar, el diseño inicial puede variar, en estrategia descendente el esquema inicial es el concepto más representativo, al igual que la estrategia centrífuga; en cambio la estrategia ascendente parte de la lluvia de características sin estructura; la estrategia mixta parte de la relación de dos o más conceptos. Es resumen el objetivo es tener un esquema entidad-relación armazón inicial.

c) Diseño de esquemas

- ❖ Luego del esquema inicial debemos continuar con el desarrollo de esquemas parciales, sobre la base del conjunto de enunciados del requerimiento de BD, para ello debemos aplicar refinamientos con primitivas descendentes y ascendentes.
- ❖ De ser necesario se integran o fusionan los esquemas parciales para obtener el esquema final

De esta manera conseguiremos obtener el esquema conceptual de la base de datos que cumpla con las cualidades de un buen esquema de base de datos.

6.2 CASO: Diseño conceptual de una Base de Datos Académica

El siguiente caso, ocurre en una Universidad, que requiere diseñar la base de datos académica que está conformada por varias Escuelas Profesionales (carreras o programas de estudio) las cuales en el tiempo han implementado diversos planes curriculares. A continuación, el detalle metodológico utilizado para obtener el esquema conceptual de la base de datos para este requerimiento.

a) Análisis de los requerimientos

Recogido los datos, tanto en lenguaje natural, cuestionarios, entrevistas, así como la observación directa, se obtuvo lo siguiente:

- ✓ El requerimiento de la base de datos

Para esta Base de Datos Académica, la Universidad oferta 20 Escuelas Profesionales, cada una de ellas posee uno o más currículos. Cada plan curricular contiene entre 50 y 70 asignaturas. Cada 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 Profesional, 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 elige de entre todos los cursos y currículos, los cursos a dictar, obteniéndose las clases programadas. De esta manera, el profesor encargado del curso recibe un listado de los alumnos asistentes a cada clase programada.

Asimismo, las clases programadas, 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), emitíendose 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 adscritos a los Departamentos académicos que dependen de cada Facultad. Un docente puede dictar en varias Escuelas Profesionales a la vez, sin embargo, está adscrito a un solo departamento académico, las Escuelas Profesionales están agrupadas por Facultades.

El estudiante se matricula de manera presencial u on-line en las asignaturas programadas, 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 convalidaciones permitirán el cambio de plan curricular del alumno. Finalmente, cada asignatura tiene cero o varios pre-requisitos (cursos del mismo currículo que el curso titular).

b) **Filtrando ambigüedades**

Luego de revisar el requerimiento anterior se detecta que se está utilizando indistintamente:

Alumno y estudiante, profesor y docente, plan curricular y currículo, curso y asignatura, escuela y escuela profesional, tabla de equivalencias y tabla de convalidaciones.

Teniendo en cuenta la normativa interna de la institución se ha seleccionado los términos utilizados en dichas normativas y son

los que se han subrayado. Quedando el enunciado del requerimiento estandarizado de la siguiente manera:

- ✓ Requerimiento de la base de datos filtrado las ambigüedades

Para esta Base de Datos Académica, la Universidad oferta 20 Escuelas Profesionales, cada una de ellas posee uno o más planes curriculares. Cada plan curricular contiene entre 50 y 70 asignaturas. Cada asignatura se representa por su nombre, número de créditos, horas de teoría, prácticas y total de horas, el ciclo y el tipo de asignatura. Para los estudiantes, se requieren sus datos personales como su apellido, nombre, sexo, dirección, teléfono; Además, se tiene su información académica resumida, como su Escuela Profesional, el plan curricular 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 elige de entre todas las asignaturas y planes curriculares, las asignaturas a dictar, obteniéndose las clases programadas. De esta manera, el docente encargado de la asignatura recibe un listado de los estudiantes asistentes a cada clase programada.

Asimismo, las clases programadas, se dictan en una serie de aulas disponibles, en horas definidas y dirigidos por uno o varios docentes por asignatura (cada docente dicta en horas fijas).

La Base de datos registra las notas de los estudiantes al llevar las asignaturas 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 Profesional tiene una Plana Docente, la cual es renovada cada semestre, y es elegida de entre todo el grupo de docentes adscritos a los Departamentos académicos que dependen de cada Facultad. Un docente puede dictar en varias Escuelas Profesionales a

la vez, sin embargo, está adscrito a un solo departamento académico, las Escuelas Profesionales están agrupadas por Facultades.

El estudiante se matricula de manera presencial u on-line en las asignaturas programadas, 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 Profesional tiene un grupo de Asesores a disposición de los estudiantes.

Por otro lado, se requiere tener las Tablas de Convalidaciones entre las asignaturas de los diferentes planes curriculares, existiendo en este caso, equivalencias simples. Las Tablas de convalidaciones permitirán el cambio de plan curricular del estudiante. Finalmente, cada asignatura tiene cero o varios pre-requisitos (asignaturas del mismo plan curricular que la asignatura titular).

c) División de los enunciados del requerimiento en conjuntos homogéneos

Este caso amerita utilizar la Estrategia mixta, de tal manera que es susceptible de ser desglosado en enunciados del requerimiento en conjuntos homogéneos, de tal manera que nos facilite reconocer los conceptos y sus relaciones a fin de poder detallarlos en el diseño.

i. Enunciados sobre los estudiantes

Para los estudiantes se requieren sus datos personales como su apellido, nombre, sexo, dirección y teléfono; Además, se tiene su información académica resumida, como su Escuela Profesional, el plan curricular 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).

El estudiante se matricula de manera presencial u on-line en las asignaturas programadas, 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 Profesional tiene un grupo de Asesores a disposición de los estudiantes.

La Base de datos registra las notas de los estudiantes por llevar las asignaturas 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.

ii. Enunciados sobre los planes curriculares

En la Base de datos Académica se tienen varias Escuelas Profesionales, cada una de ellas posee una o más planes curriculares. Cada plan curricular contiene un grupo de asignaturas. Cada asignatura se representa con su nombre, número de créditos, horas de teoría, prácticas y total de horas, el ciclo y el tipo de asignatura.

Por otro lado, se requiere tener las Tablas de Convalidaciones entre las asignaturas de diferentes planes curriculares, existiendo en este caso, equivalencias simples. Las Tablas de convalidaciones permitirán el cambio de plan curricular del estudiante. Finalmente, cada asignatura tiene cero o varios prerequisitos (asignaturas del mismo plan curricular que la asignatura titular).

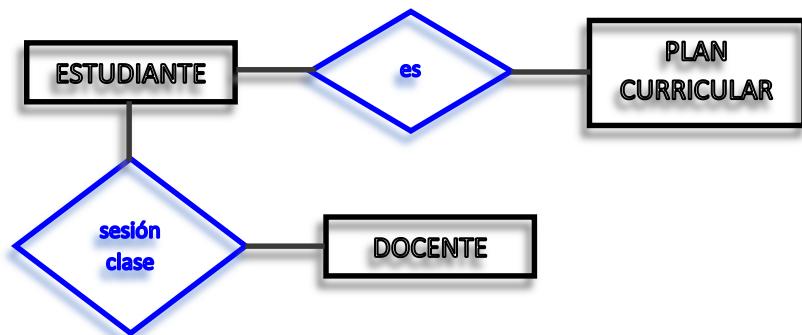
iii. Enunciados sobre los docentes

De la misma manera, cada Escuela Profesional tiene una Plana Docente, la cual es renovada cada semestre, y es elegida de entre todo el grupo de docentes adscritos a los Departamentos académicos que dependen de cada Facultad. Un docente puede dictar en varias Escuelas Profesionales a la vez, sin embargo, está adscrito a un solo departamento

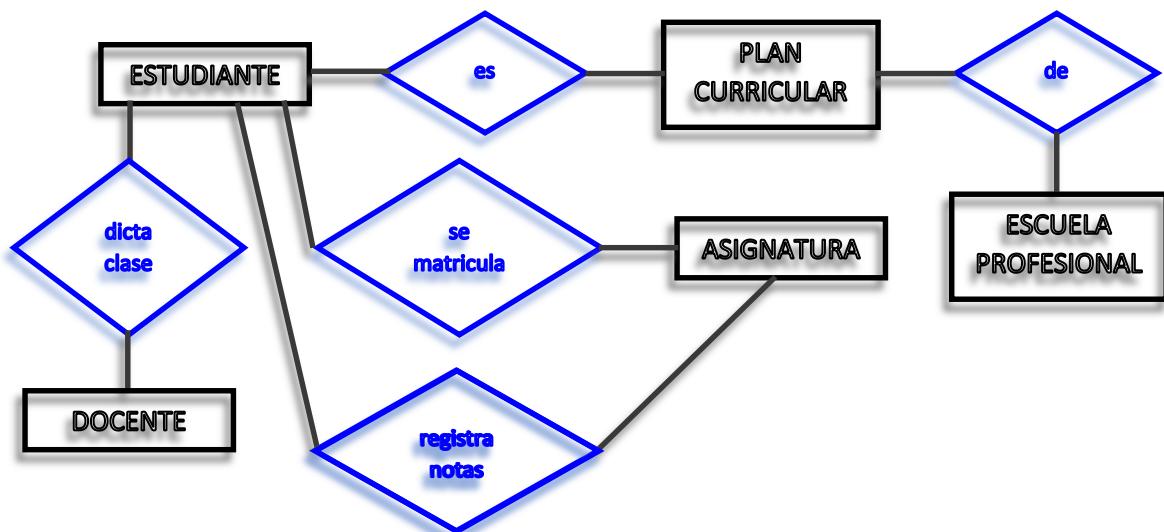
académico, las Escuelas Profesionales están agrupadas por Facultades.

Cada semestre se elige de entre todos las asignaturas y planes curriculares, las asignaturas a dictar, obteniéndose las clases programadas. De esta manera, el docente encargado de la asignatura recibe un listado de los estudiantes asistentes a cada clase programada. Asimismo, las clases programadas, se dictan en una serie de aulas disponibles, en horas definidas y dirigidos por uno o varios docentes por asignatura (cada docente dicta en horas fijas).

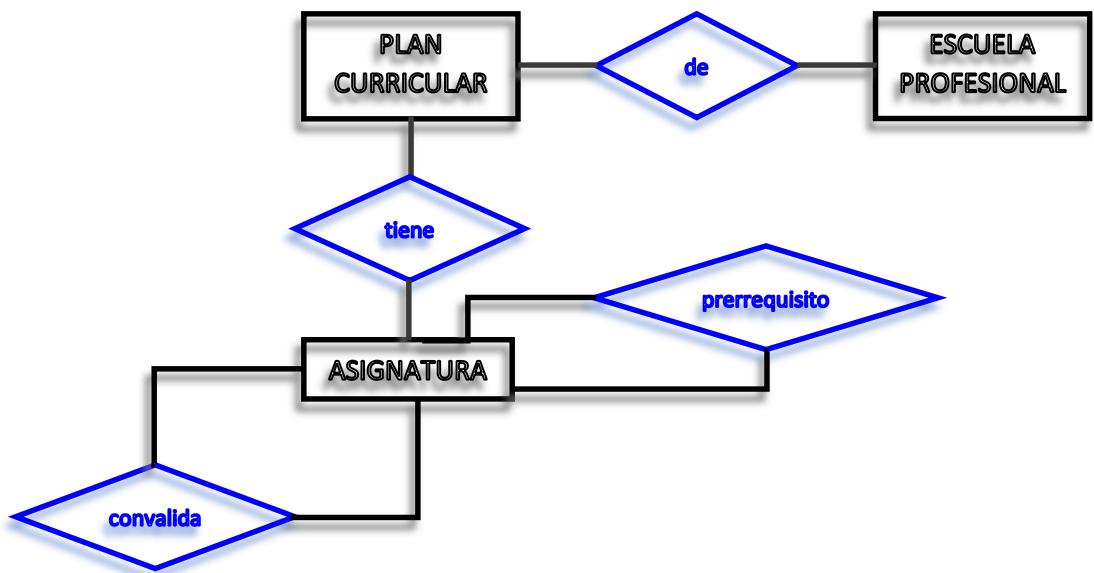
d) Esquema armazón inicial



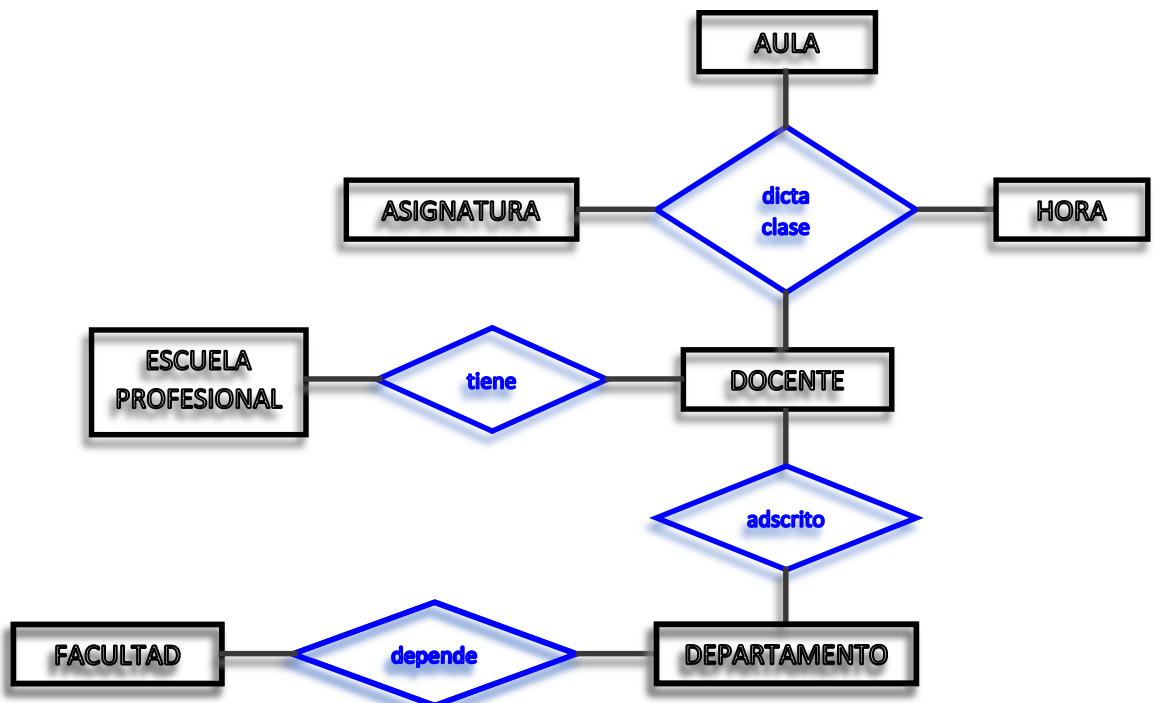
e) Esquema Estudiante



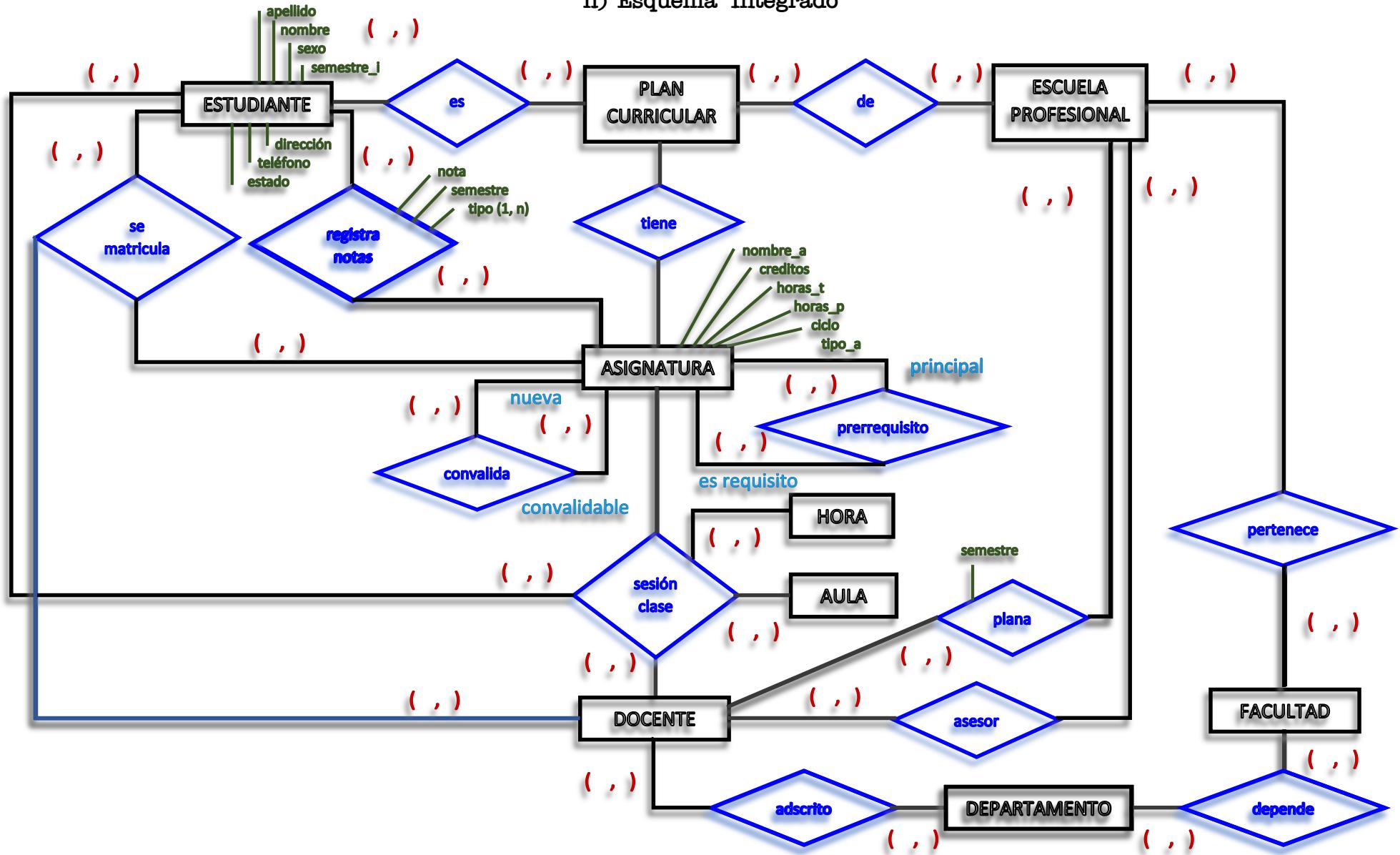
f) Esquema plan curricular



g) Esquema docente



h) Esquema Integrado



Capítulo VII

EL MODELO RELACIONAL

7.1. Modelo Relacional

Edgar Frank Codd, matemático inglés, a fines de 1968 descubre que la ciencia matemática puede ser utilizada para otorgar principios sólidos a la administración de base de datos, y en 1970 cuando era investigador de IBM [5], introdujo el modelo relacional [15], entonces el modo en que se veían las bases de datos cambió por completo.

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 contenido 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). Las primeras implementaciones comerciales del modelo relacional se dan a inicios de los 80's [3] en el siglo pasado, siendo pioneros ORACLE DBMS, SQL/DS sistema de IBM, a partir de ellos se han generado diversas implementaciones comerciales de las bases de datos relacionales.

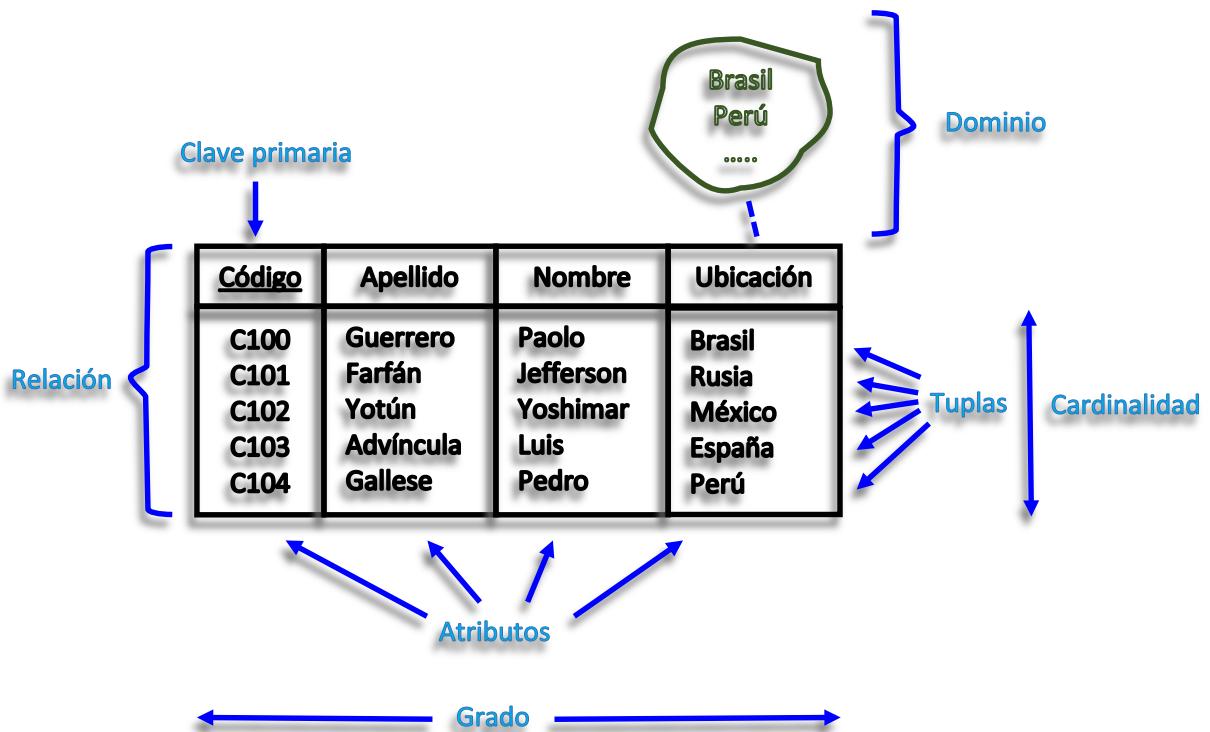
El modelo relacional, como modelo de datos, tiene que ver con tres aspectos de los datos:

- a. Estructura de los datos
 - b. Integridad de los datos, y
 - c. Manipulación de los datos

Los cuales presentaremos en adelante.

7.2. Estructura de datos Relacional

Los términos estructurales del modelo relacional presentados en [15], son adaptados de [5] y se muestran en la siguiente figura:



a) Relación

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. La relación de la figura previa se le puede llamar FUTBOLISTA.

b) 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 y se representan gráficamente como una tabla bidimensional en la que las filas corresponden a registros individuales y las columnas corresponden a los campos o atributos de esas tuplas (registros). Los atributos pueden aparecer en la relación en cualquier orden. En la figura previa, tenemos 4 atributos: código, apellido, nombre y dirección.

c) 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) 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 cuatro valores, uno para cada atributo. Las tuplas de una relación no siguen ningún orden.

e) Grado

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

f) Cardinalidad

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 K de la relación R es una clave candidata para sí y sólo si satisface las siguientes propiedades:

1º Unicidad

Nunca hay dos tuplas en la relación R con el mismo valor de K

2º Minimalidad

Ningún subconjunto de K tiene la propiedad de unicidad, es decir, no se pueden eliminar componentes de K (si este es compuesto) sin destruir la unicidad, significa que es irreductible.

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 forma una clave candidata.

La clave primaria (Primary Key) 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 o foránea (foreign key) 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.

7.3. Integridad de datos

El término 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.

Entonces, sobre la base de los conceptos de clave primaria con sus propiedades de unicidad y Minimalidad, así como el de clave foránea, es que el modelo relacional plantea dos reglas básicas de integridad:

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 las tuplas (registros) de las relaciones (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 puede tener en consideración las siguientes reglas:

1. 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.

2. No se 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 CURRICULO, si existen alumnos relacionados a ese currículo en la tabla ESTUDIANTE.
3. 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 estudiante de la tabla ESTUDIANTE si existen notas de convalidaciones de ese estudiante en la tabla CONVALIDACIONES.

Para hacer práctica la aplicación de la integridad referencial, consideremos las 4 alternativas siguientes:

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 CURRICULO, se cambia automáticamente el campo Code_Curr en los alumnos relacionados de la tabla ESTUDIANTE.
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_CURRICULO, se eliminan automáticamente los registros relacionados que indican cuáles 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.

7.4. Reglas de Integridad Referencial en SYBASE

Antes haremos una introducción a la herramienta.

7.4.1. SYBASE Adaptive Server Anywhere (ASA)

Es un sistema administrador de bases de datos relacionales (RDBMS) incluye gestión de transacciones, integridad referencial, procedimientos almacenados (store procedures) Java y SQL, triggers (disparadores), entre otras funcionalidades [16].

Partes de un sistema de base de datos

Un RDBMS es un sistema para almacenar y recuperar data, en la cual la data está organizada en tablas interrelacionadas

- **Database (base de datos)**

Una base de datos es una colección de tablas que están relacionadas por claves primarias y foráneas. Las tablas mantienen la información en la base de datos. Las tablas y las claves juntas definen la estructura de la base de datos. Un sistema de gestión de bases de datos accede a esta información. Una base de datos de SQL Anywhere es un archivo, generalmente con una extensión de .db.

- **Database Server**

El servidor de base de datos gestiona la base de datos. Todos los accesos a la base de datos ocurren a través del servidor de base de datos.

El servidor de bases de datos permite el acceso a las bases de datos desde las aplicaciones cliente y procesa los comandos de manera segura y eficiente. Una base de datos solo puede tener un servidor administrándolo a la vez. Sin embargo, un servidor de bases de datos SQL Anywhere puede administrar muchas bases de datos al mismo tiempo.

Archivos de base de datos

- **El archivo: database**

Este archivo contiene la información de la base de datos.
Normalmente tiene la extensión .db.

- **El transaction log**

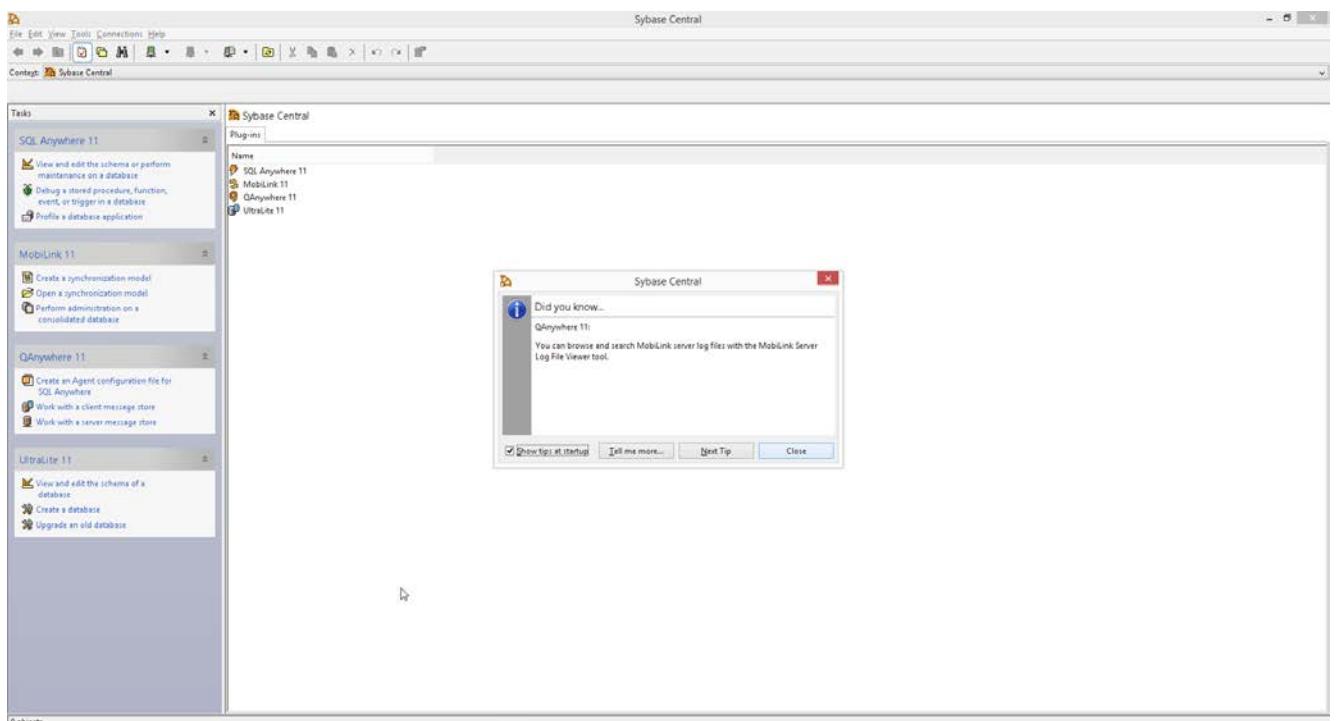
Este archivo contiene un registro de los cambios realizados en la base de datos, y es necesario para la recuperación y sincronización. Normalmente tiene la extensión .log.

- **El archivo temporal (temporary)**

El servidor de la base de datos utiliza el archivo temporal para contener la información necesaria durante una sesión de base de datos. El servidor de la base de datos elimina este archivo una vez que la base de datos se apaga, incluso si el servidor sigue corriendo. Este archivo tiene un nombre generado por el servidor con la extensión .tmp.

7.4.2. Creando una Base de datos con SYBASE Central

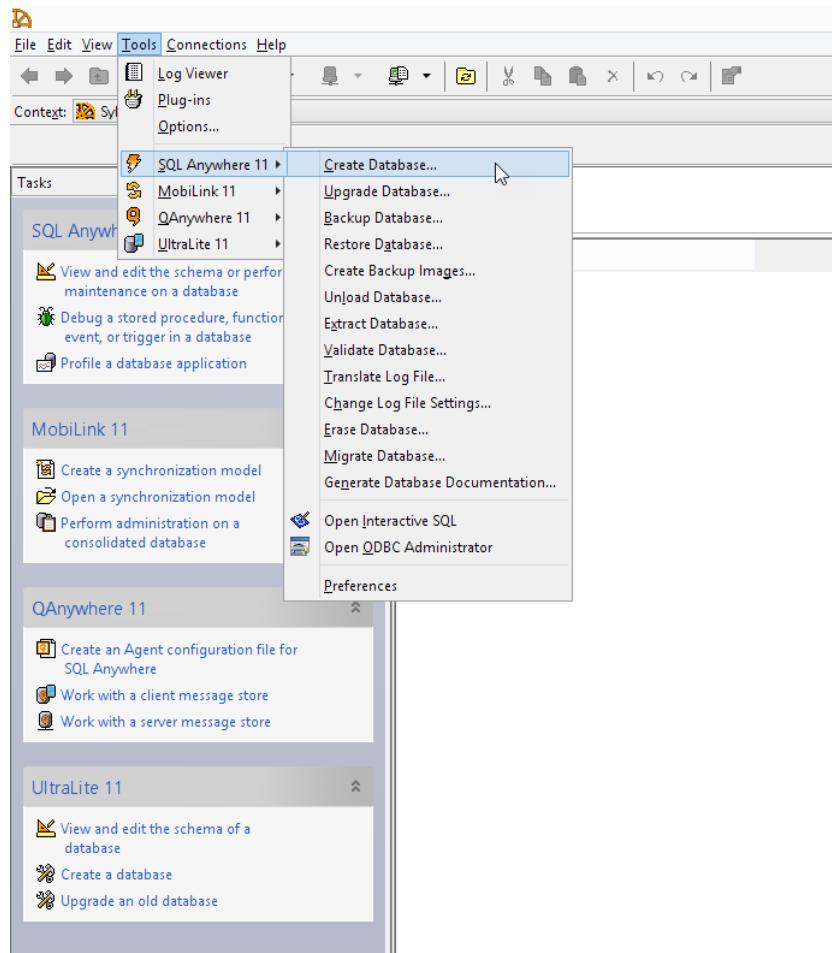
Ejecutamos SYBASE Central



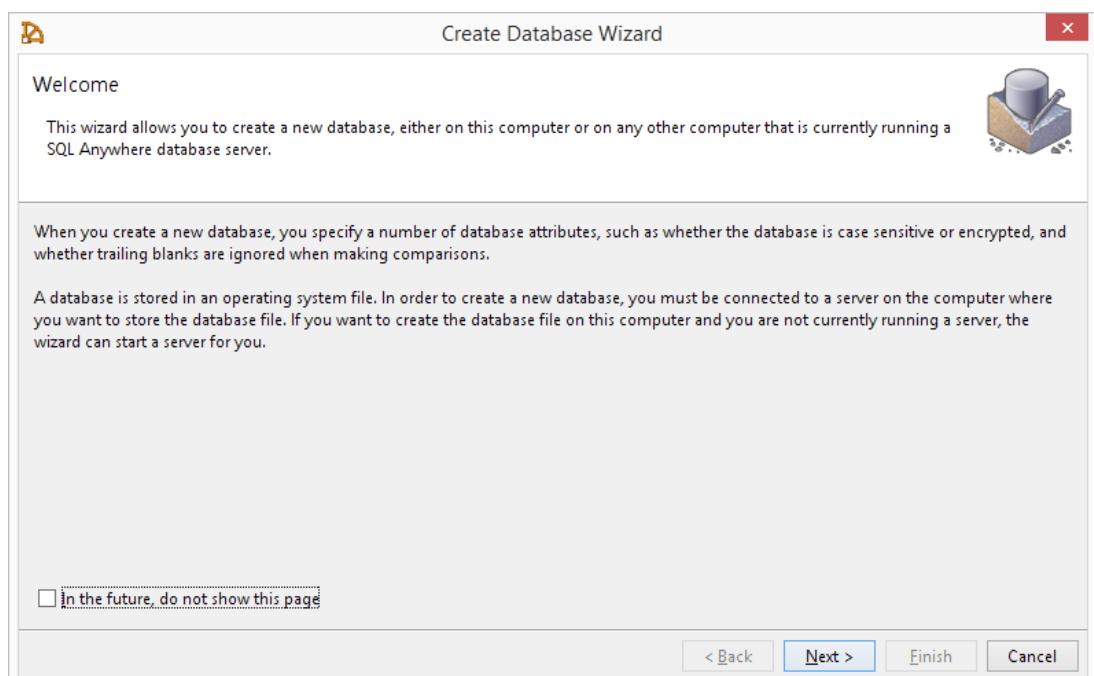
Cerramos la ventana de tips (close)

Y en el menú principal elegimos Tools - SQL Anywhere 11

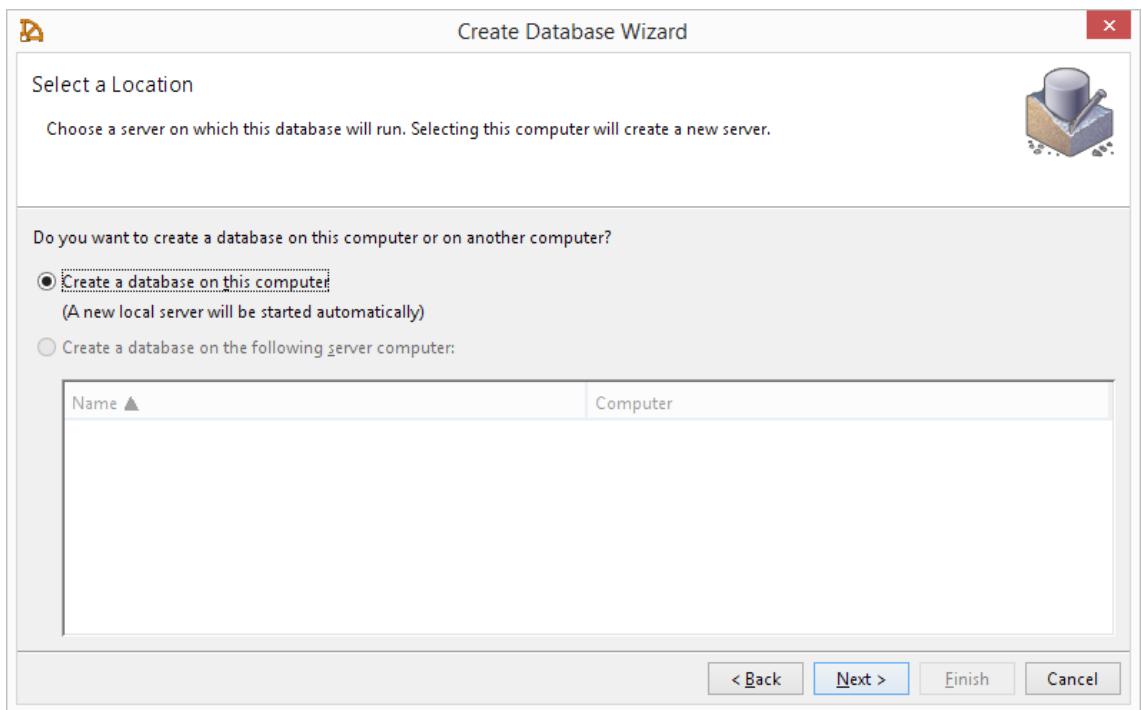
- Create database



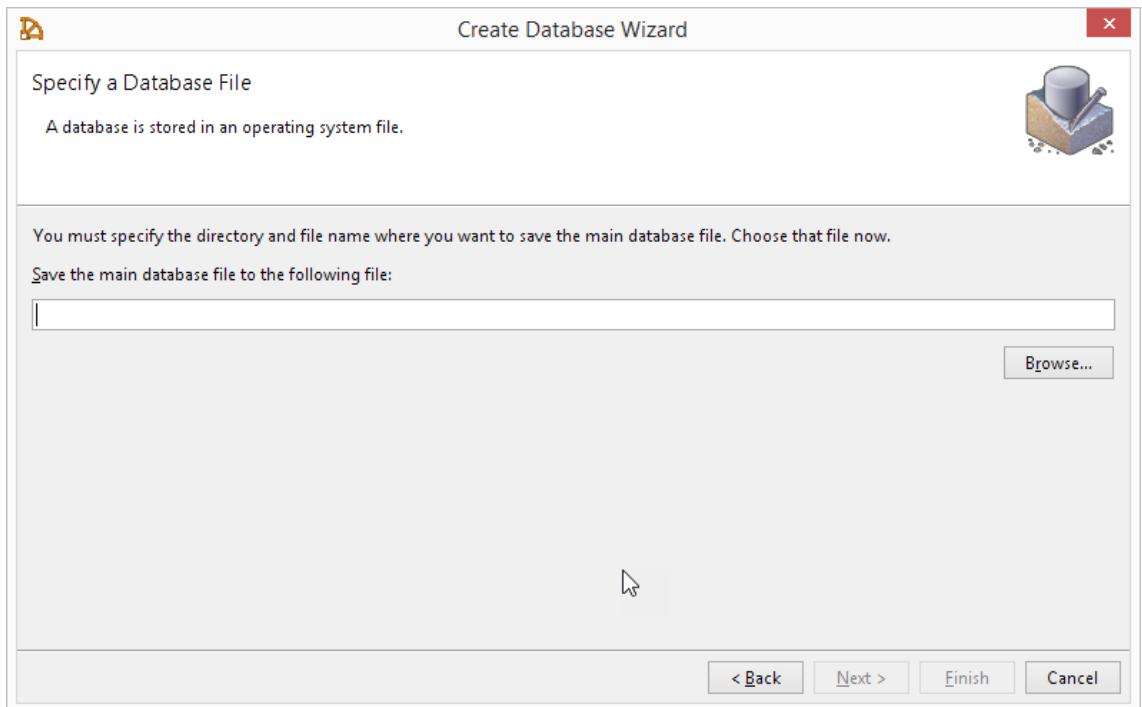
Se nos presentará el wizard que nos ayudará a crear nuestra primera base de datos desde el administrador SYBASE Central.



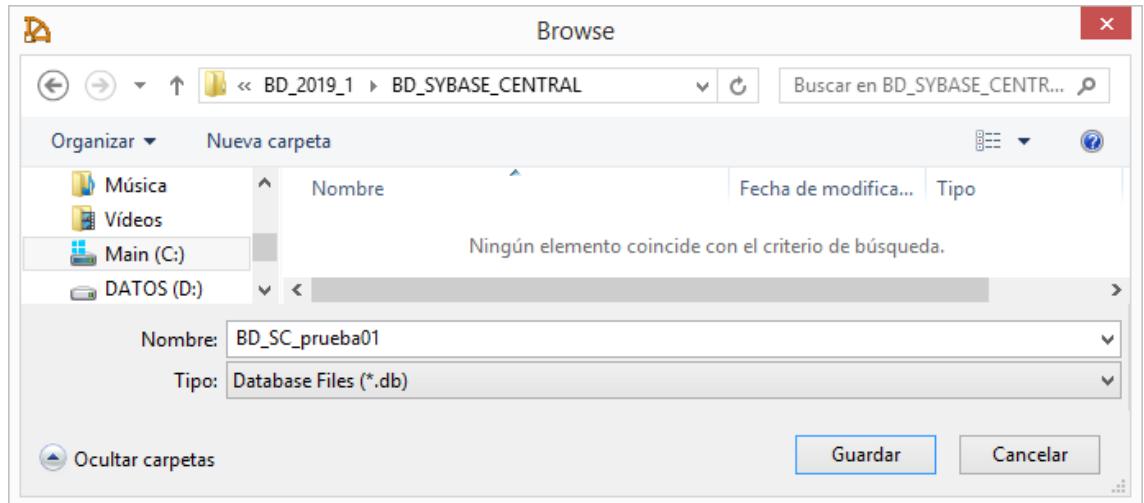
Hacemos click en siguiente (next)



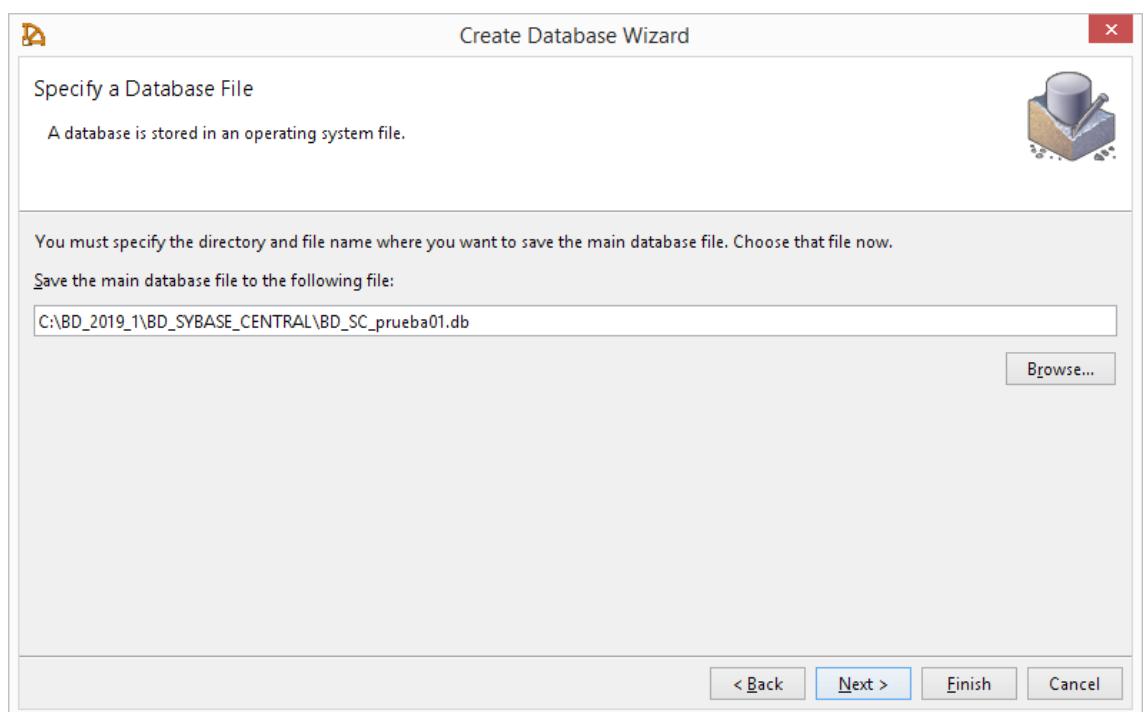
Elegimos la opción por defecto: Crear una base de datos en este computador y hacemos click en siguiente (next)



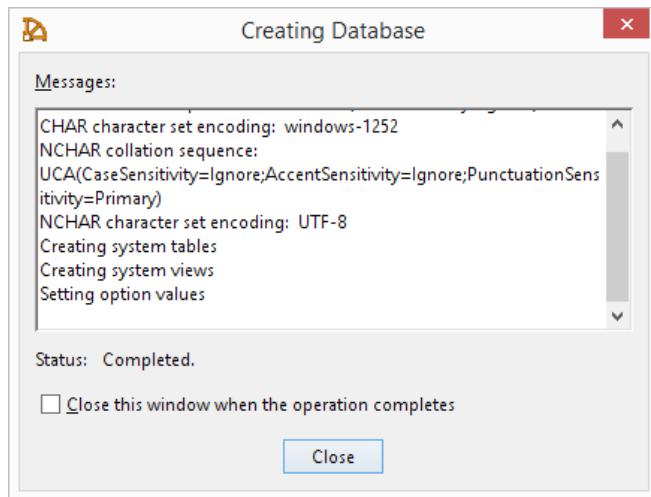
Aquí especificamos el archivo de base de datos, con el browse elegimos el directorio y el nombre de archivo que se quiere guardar como archivo principal de la base de datos.



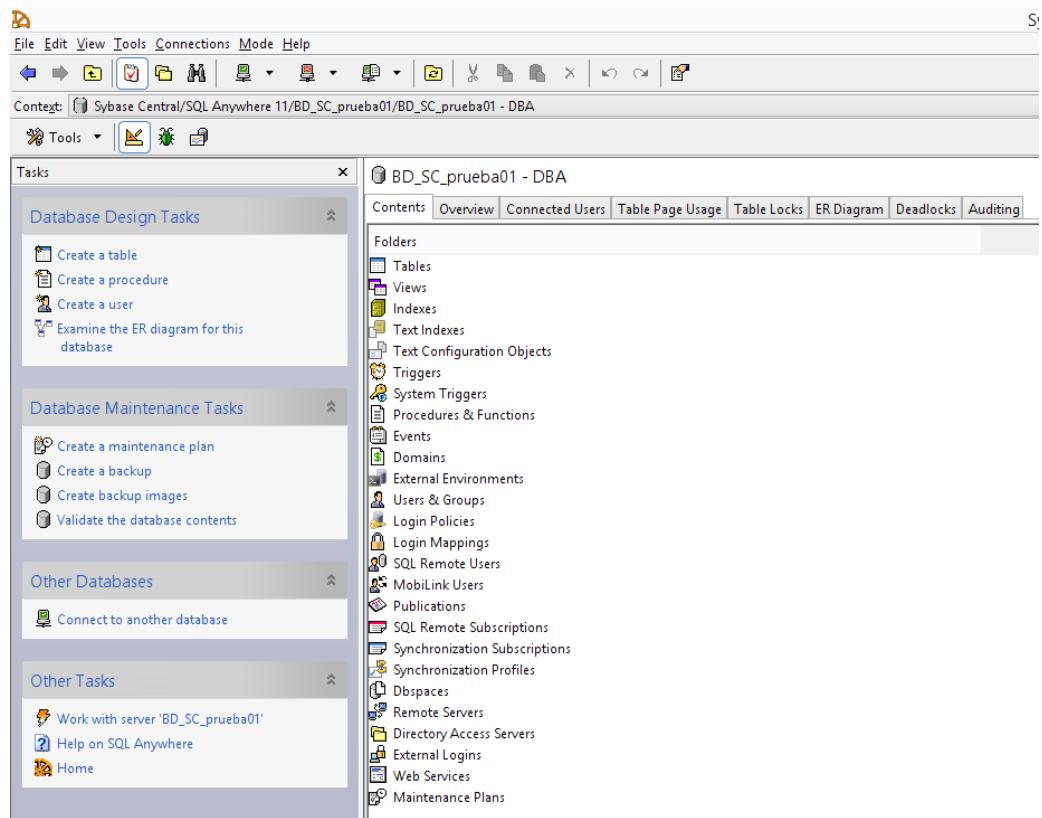
Luego de elegir el disco duro (C: en nuestro caso), la ruta de carpetas: BD_2019_1\BD_SYBASE_CENTRAL donde se guardará nuestra base de datos, le asignamos un nombre: BD_SC_prueba01 y le damos click a guardar.



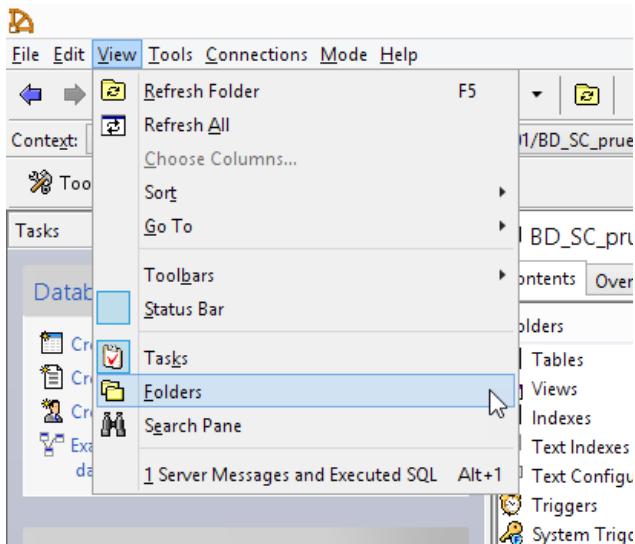
Quedando a la vista la ruta completa y el nombre del archivo database, podemos continuar con las siguientes pantallas de configuración con (next) de hacerlo dejar por defecto y dar siguiente cada vez. En esta oportunidad le damos click a finalizar (finish)



Aparece una caja de mensajes, luego de ver los mensajes de configuración por defecto, el Status se lee completado (completed), entonces cerramos (close)



Se ha creado la base de datos: BD_SC_prueba01 con 26 objetos empezando por las Tablas (Tables) y finalizando con planes de mantenimiento (Maintenance Plans). Notar que el panel de tareas (Tasks) se ha adecuado al cambio. Podemos cambiar de panel izquierdo en el menú principal View\Folders



Modificándose el panel izquierdo con una vista tipo explorador mostrando los 26 folders que contiene la base de datos recién creada.

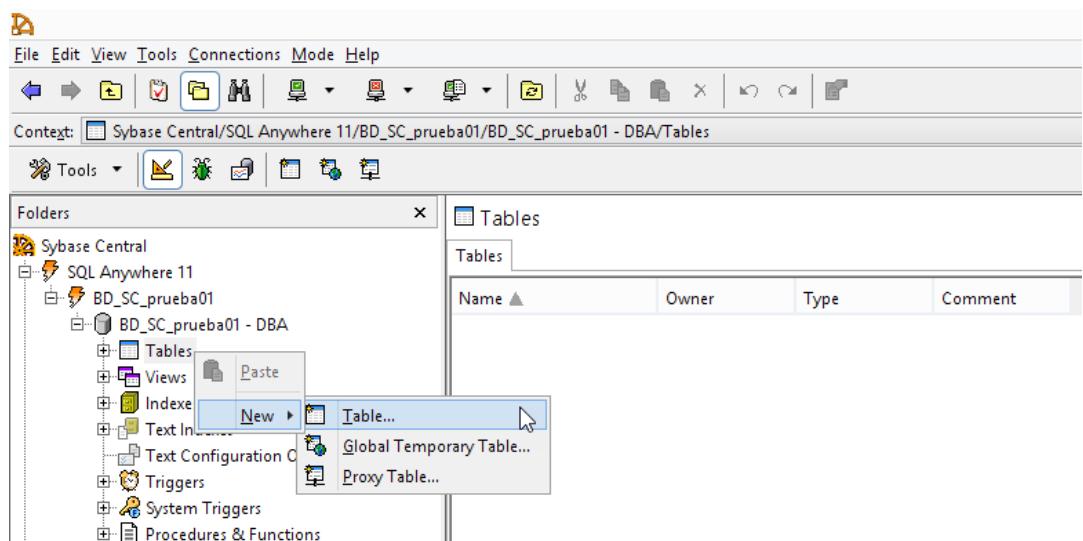
A screenshot of the Sybase Central application window with the 'Folders' tree view expanded. The tree view on the left shows the following structure under 'BD_SC_prueba01 - DBA':

- Tables
- Views
- Indexes
- Text Indexes
- Text Configuration Objects
- Triggers
- System Triggers
- Procedures & Functions
- Events
- Domains
- External Environments
- Users & Groups
- Login Policies
- Login Mappings
- SQL Remote Users
- MobiLink Users
- Publications
- SQL Remote Subscriptions
- Synchronization Subscriptions
- Synchronization Profiles
- Dbspaces
- Remote Servers
- Directory Access Servers
- External Logins
- Web Services
- Maintenance Plans

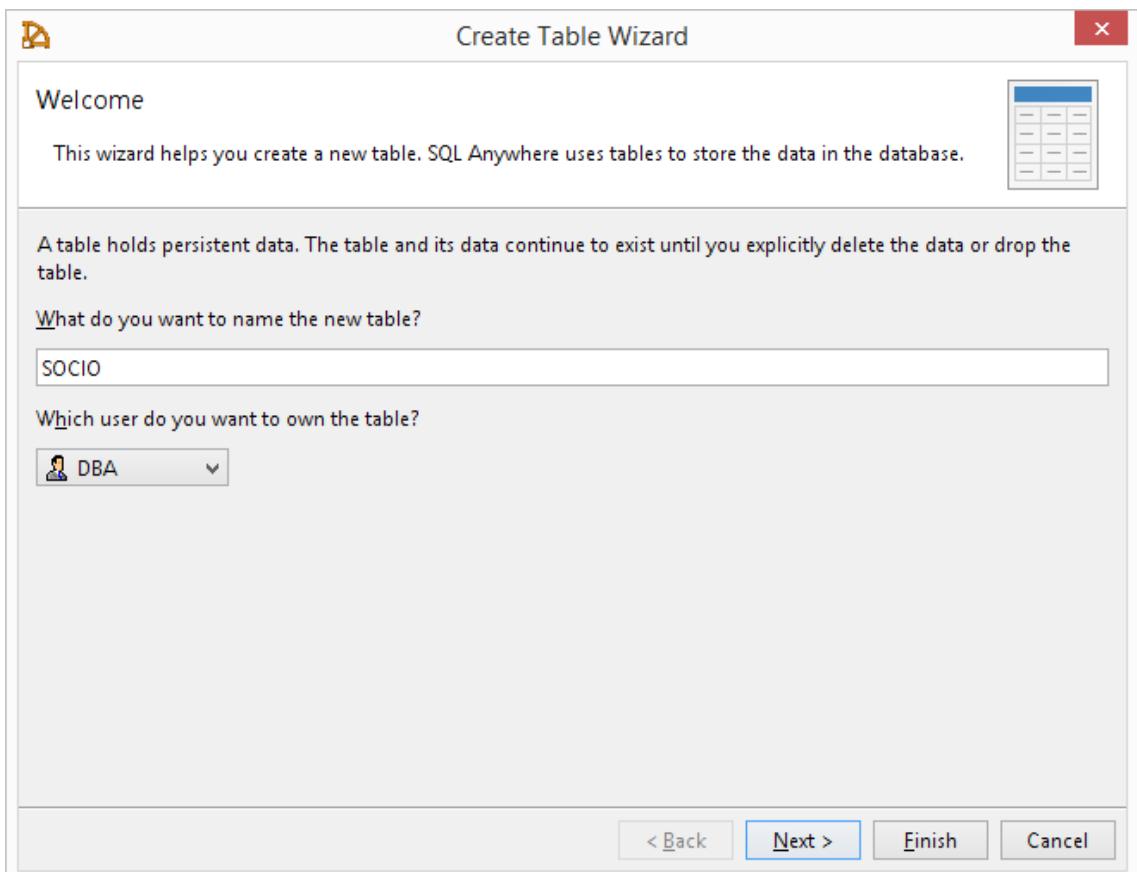
The 'Tables' node is currently selected. The main pane on the right displays a list of objects corresponding to the selected 'Tables' folder, including:

- BD_SC_prueba01 - DBA
- Folders
- Tables
- Views
- Indexes
- Text Indexes
- Text Configuration Objects
- Triggers
- System Triggers
- Procedures & Functions
- Events
- Domains
- External Environments
- Users & Groups
- Login Policies
- Login Mappings
- SQL Remote Users
- MobiLink Users
- Publications
- SQL Remote Subscriptions
- Synchronization Subscriptions
- Synchronization Profiles
- Dbspaces
- Remote Servers
- Directory Access Servers
- External Logins
- Web Services
- Maintenance Plans

Enseguida crearemos una tabla, para ello haremos click derecho sobre el folder tablas (Tables), como sigue:



Enseguida le damos nombre a nuestra tabla



Si le damos siguiente (next) tendremos opciones de configuración personalizadas, en esta oportunidad terminaremos (click en finish) y tendremos acceso a completar los atributos de la tabla

SOCIO (DBA)

Columns

	PKey	Name	ID	Obj. ID	Data Type	Size	Scale	Compressed	Nulls	Uniq.	Value	Comment
1	<input checked="" type="checkbox"/>	codigo			char	3		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
2	<input type="checkbox"/>	nombre			char	10		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
3	<input type="checkbox"/>	email			char	20		<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
4	<input type="checkbox"/>	code_c			char	3		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		

Cuando le damos guardar



los datos ID y Obj.ID se autocompletan por defecto

SOCIO (DBA)

Columns Constraints Referencing Constraints Indexes Text Indexes Triggers Dependent Views Data

	PKey	Name	ID	Obj. ID	Data Type	Size	Scale	Compressed	Nulls	Uniq.	Value	Comment
1	<input checked="" type="checkbox"/>	codigo	1	2956	char	3		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
2	<input type="checkbox"/>	nombre	2	2957	char	10		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
3	<input type="checkbox"/>	email	3	2958	char	20		<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
4	<input type="checkbox"/>	code_c	4	2959	char	3		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		

Procedemos de igual manera para crear la tabla CLUB

CLUB (DBA)

Columns Constraints Referencing Constraints Indexes Text Indexes Triggers Dependent Views Data

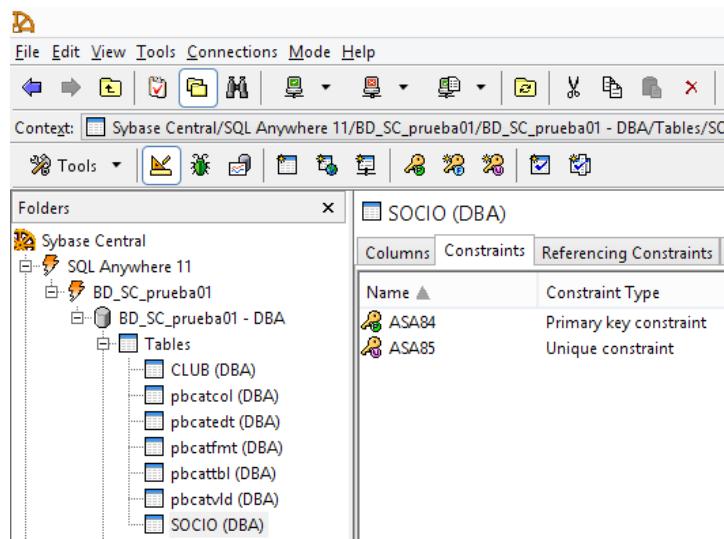
	PKey	Name	ID	Obj. ID	Data Type	Size	Scale	Compressed	Nulls	Uniq.	Value	Comment
1	<input checked="" type="checkbox"/>	code_c	1	2951	char	3		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
2	<input type="checkbox"/>	name_c	2	2952	char	5		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

Dado que para cada tabla hemos definido su primary key (PKey), vamos a proceder a crear la foreign key entre ellas

Creando un Foreign Key con SYBASE Central

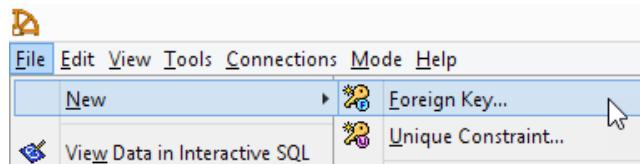
Puede crear una relación de clave externa entre tablas. Una relación de clave externa actúa como una restricción; para las nuevas filas insertadas en la tabla secundaria, el servidor de base de datos comprueba si el valor que está insertando en la columna de clave externa coincide con un valor en la clave principal de la tabla principal. Para ello debe tener autorización del DBA o ser el propietario de la tabla.

Para crear el FK con SYBASE Central, primero debemos seleccionar la tabla a la cual se le quiere crear o eliminar el FK y en el panel derecho se debe seleccionar la pestaña constraints

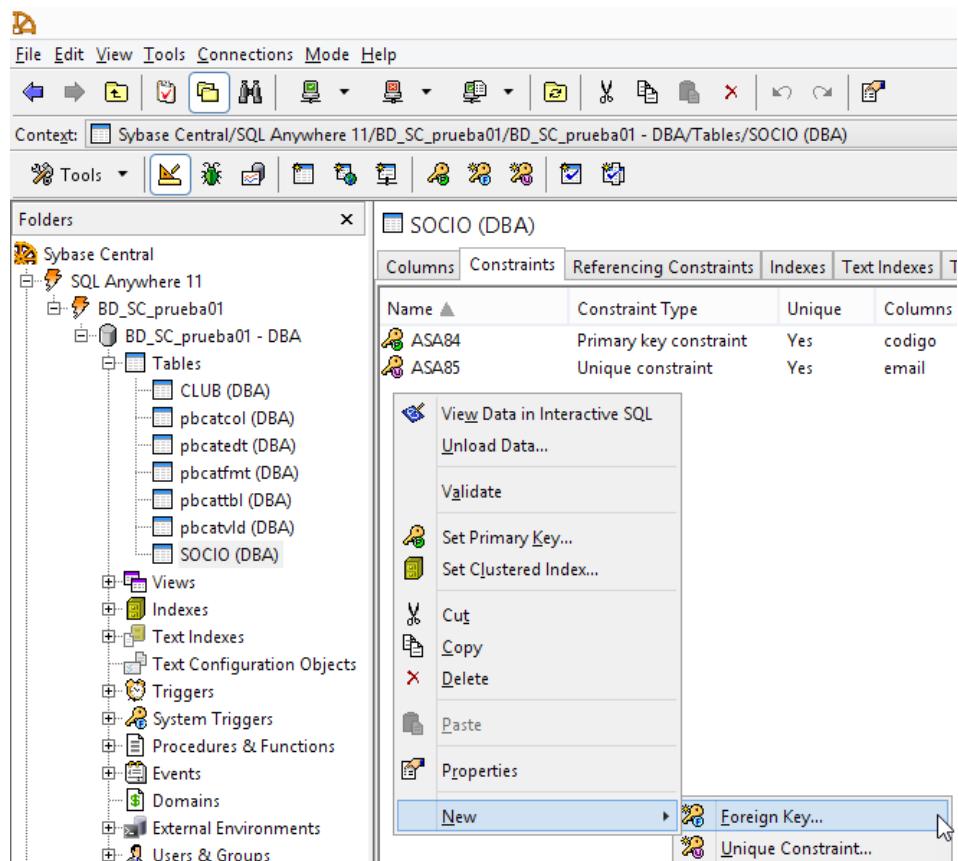


Luego tiene dos maneras:

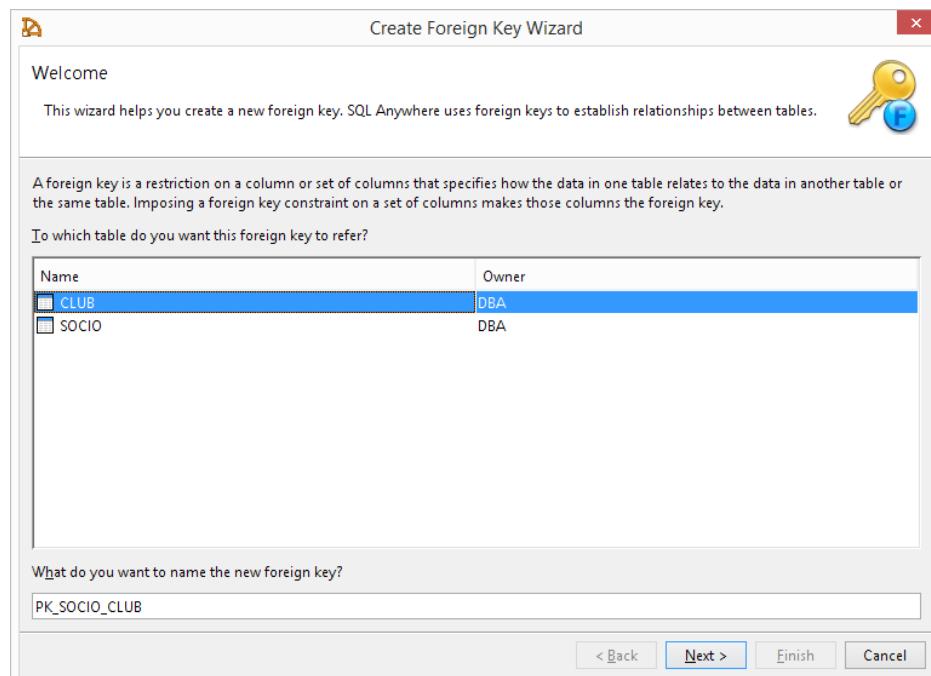
- ✓ Por el menú principal



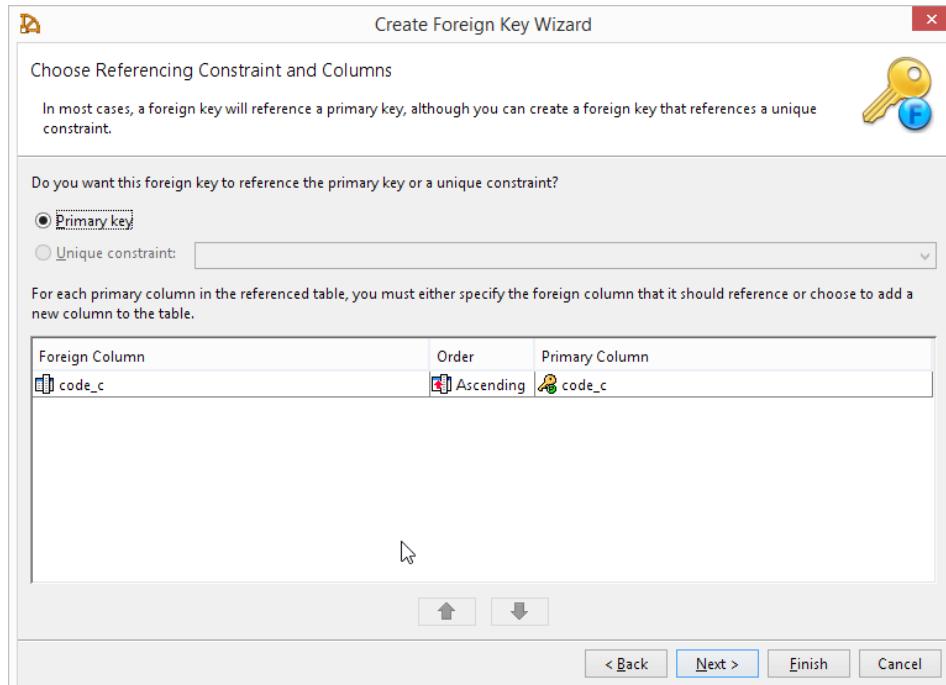
- ✓ Haciendo click derecho en el panel constraints



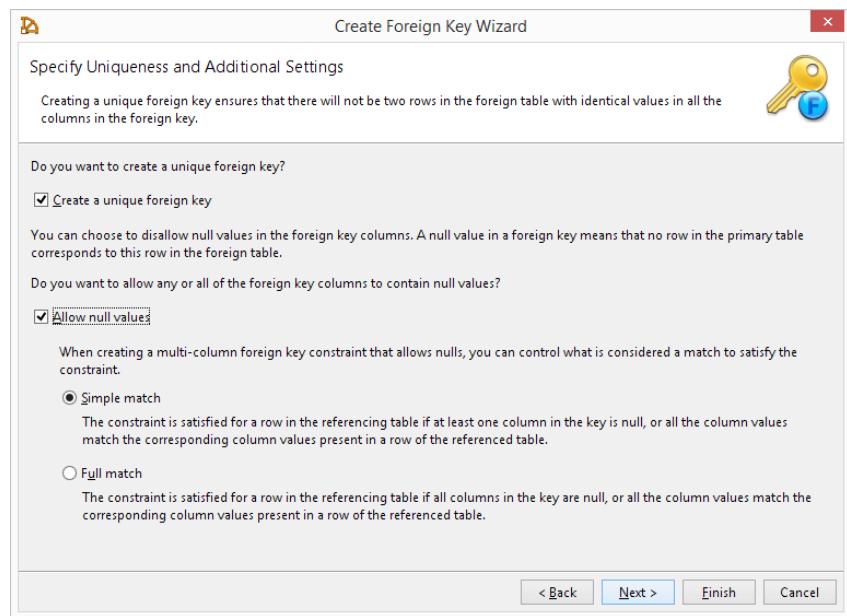
De cualquiera de las maneras llegamos al Wizard



Seleccionamos la tabla de referencia en nuestro caso (CLUB) y le damos un nombre a la clave externa (foreign key), click en seguir (next)



Aquí elegimos la referencia de la restricción y la columna, proseguimos (next)



Aquí indicamos la creación de una única FK y si se permitirán valores nulos, dejar por defecto, proseguir (next)

Create Foreign Key Wizard

Specify Index Clustering

You can make this foreign key's underlying index a clustered index, which causes the rows in the table to be stored in approximately the same order as they appear in the foreign key.

Create a clustered foreign key

Create Foreign Key Wizard

Specify a Comment

Comments help to organize the database and make it easier to administer.

What would you like the comment to be for this foreign key?

Click Finish to create the foreign key.

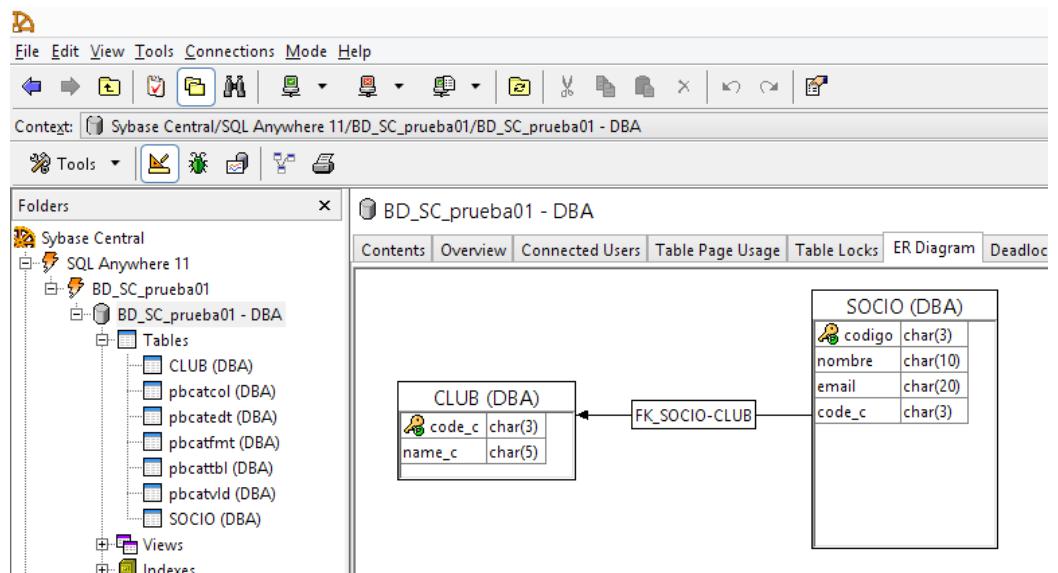
< Back Next > **Finish** Cancel

Especificar índice, dejar por defecto (next), en comentario agregar o dejar por defecto y finalizar (finish)

Name	Constraint Type	Unique	Columns
ASA84	Primary key constraint	Yes	codigo
ASA85	Unique constraint	Yes	email
FK_SOCIO-CLUB	Foreign key constraint	Yes	code_c

Se crea la FK y se muestra en la pestaña constraints.

Adicionalmente, en el panel izquierdo folders, seleccionamos la BD, y en el panel izquierdo seleccionamos la pestaña ER diagram y se podrá visualizar gráficamente la FK creada.



Enseguida, poblamos con algunos datos las tablas

code_c	name_c
123	Juan Pérez

En el panel derecho, click derecho y Add row

The screenshot shows the Sybase Central interface. On the left, there's a tree view of databases: 'SQL Anywhere 11' has 'BD_SC_prueba01' expanded, showing 'Tables' which contains 'CLUB (DBA)'. On the right, a table editor window is open with three rows and two columns. The columns are labeled 'code_c' and 'name_c'. The data is as follows:

code_c	name_c
11	U
22	AL

7.4.3. Conectando BD desde el Cliente con ODBC

Habiendo creado la base de datos desde el Administrador con SYBASE Central, tenemos la necesidad de conectar la base de datos para poder desarrollar el Sistema de información o aplicación que le permita a los usuarios acceder a la data guardada, para ello,

Desde el lado cliente, ejecutamos el IDE PowerBuilder



Y seleccionamos database

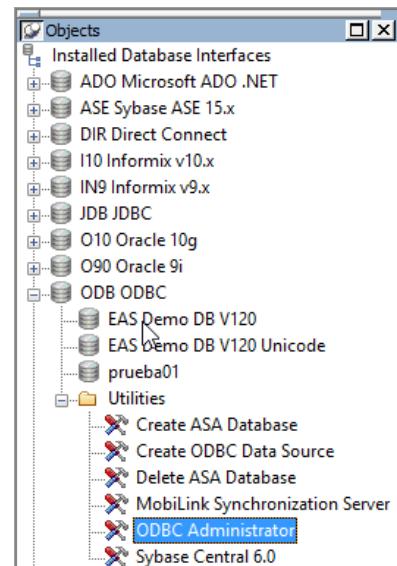


Creando DNS (Data Source Name)

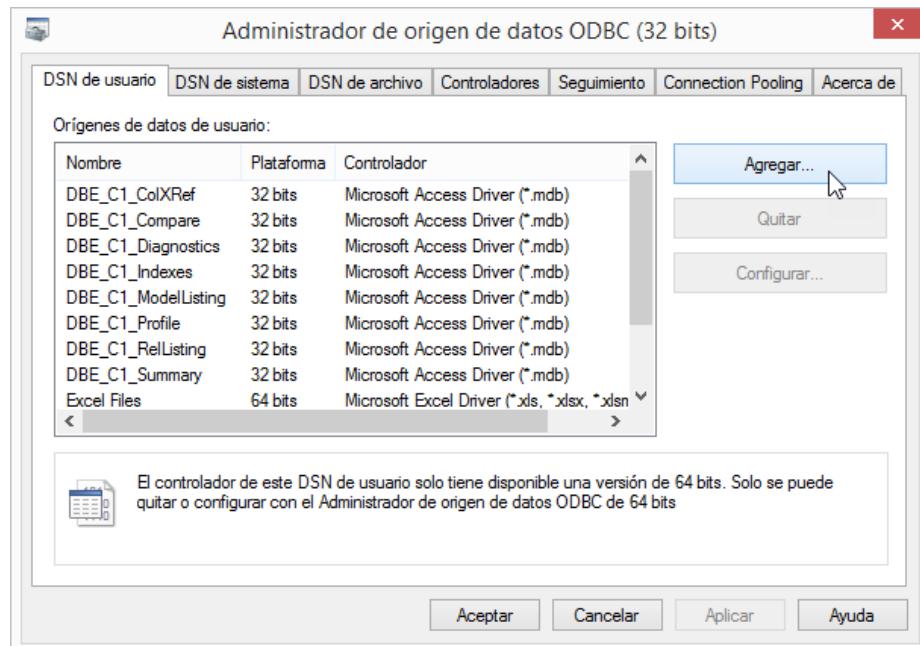
Para conectar la base de datos necesitamos de un DNS o nombre del origen de datos, que permita establecer la conexión con la fuente de datos ODBC accedida por un proveedor de un gestor de base de datos en particular

Seleccionamos el objeto ODB ODBC, desplegamos utilities

y Doble click en ODBC Administrator

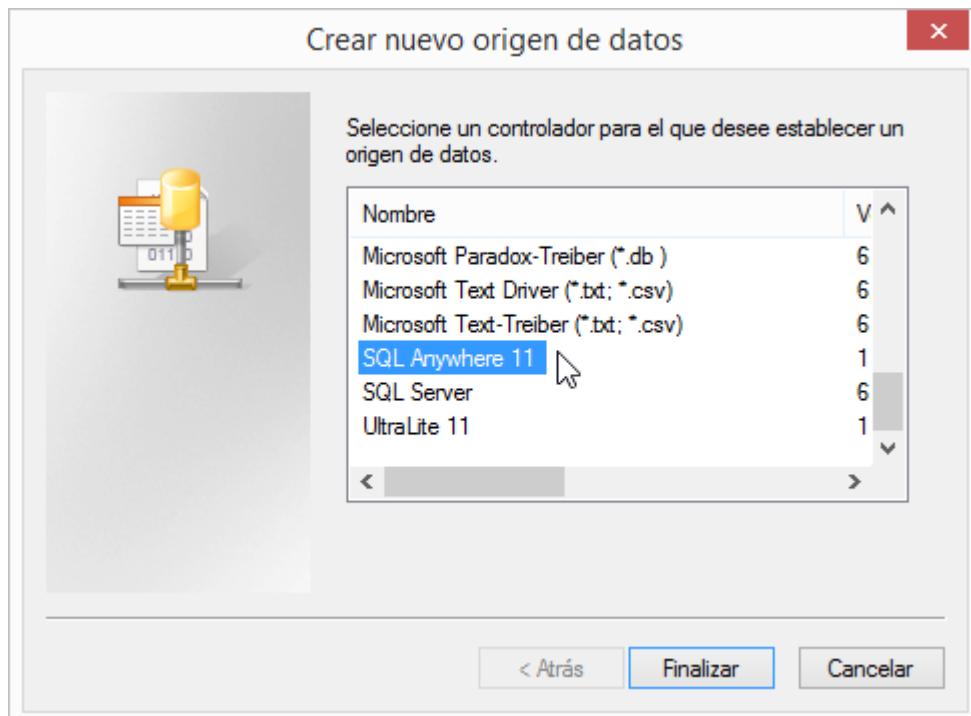


Tenemos acceso a la ventana Administrador de origen de datos ODBC

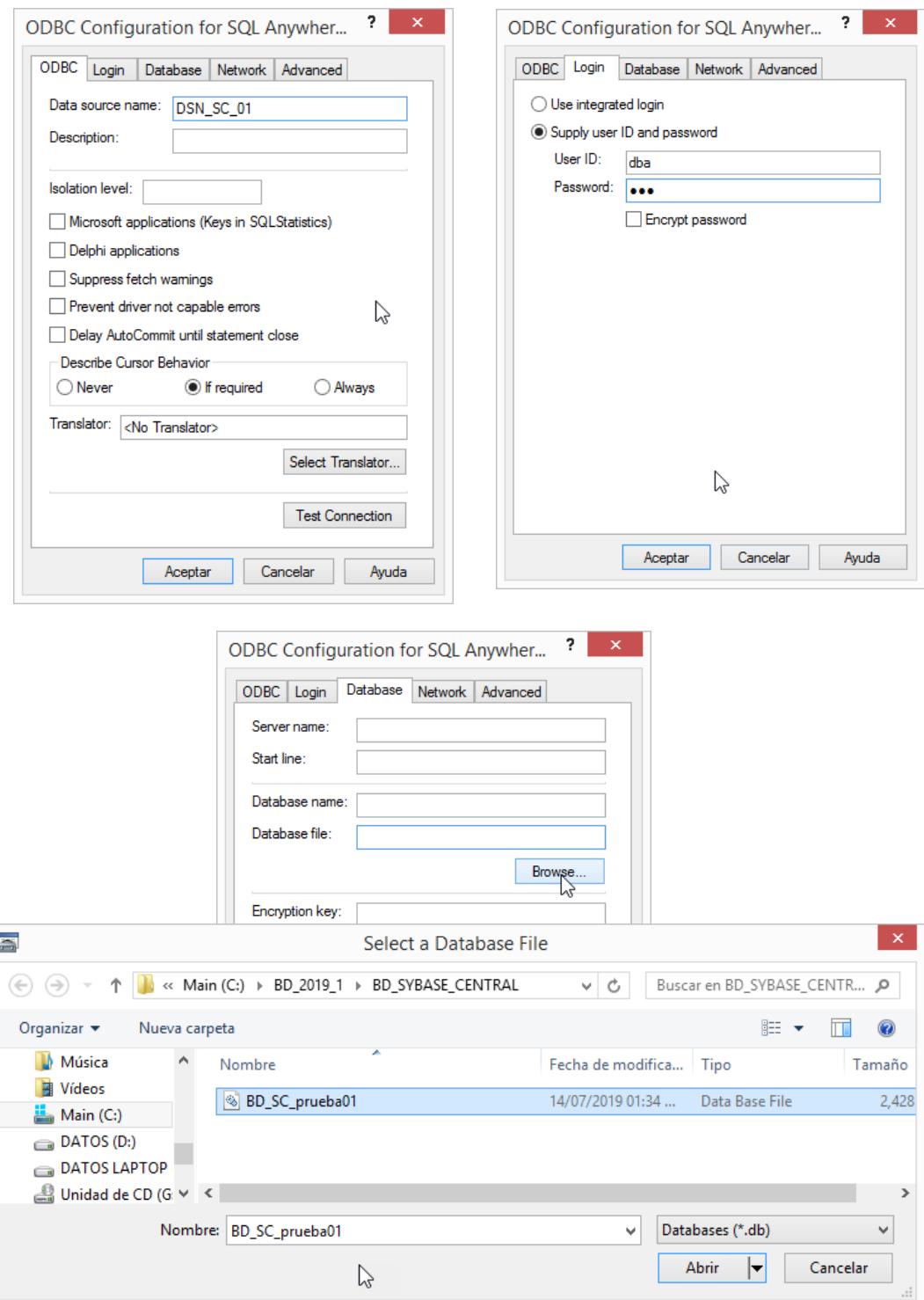


Para crear un nuevo DSN, damos click en agregar

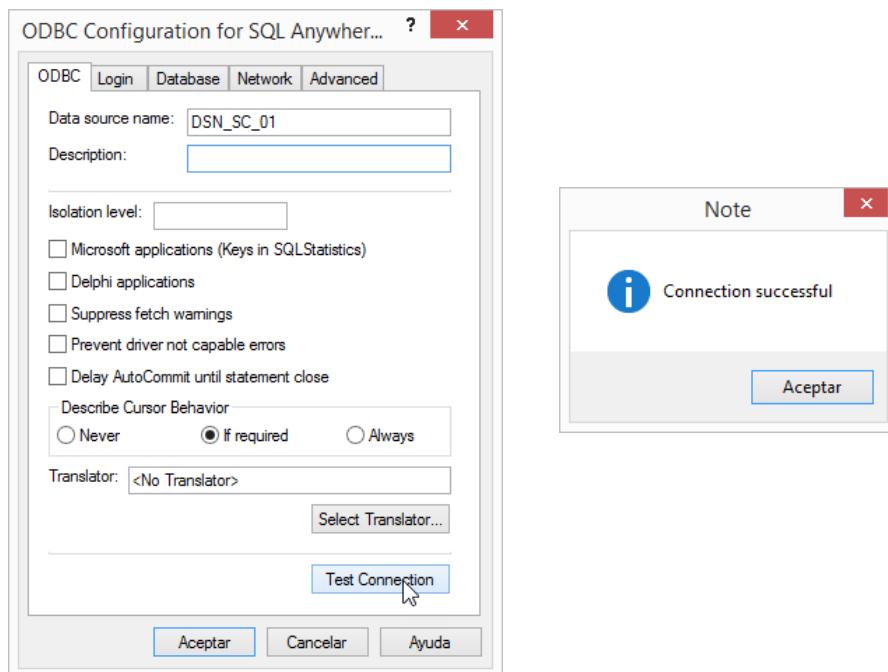
Aquí seleccionamos el controlador de acuerdo con el motor de base de datos que se esté utilizando, en nuestro caso es SQL Anywhere 11 y le damos finalizar.



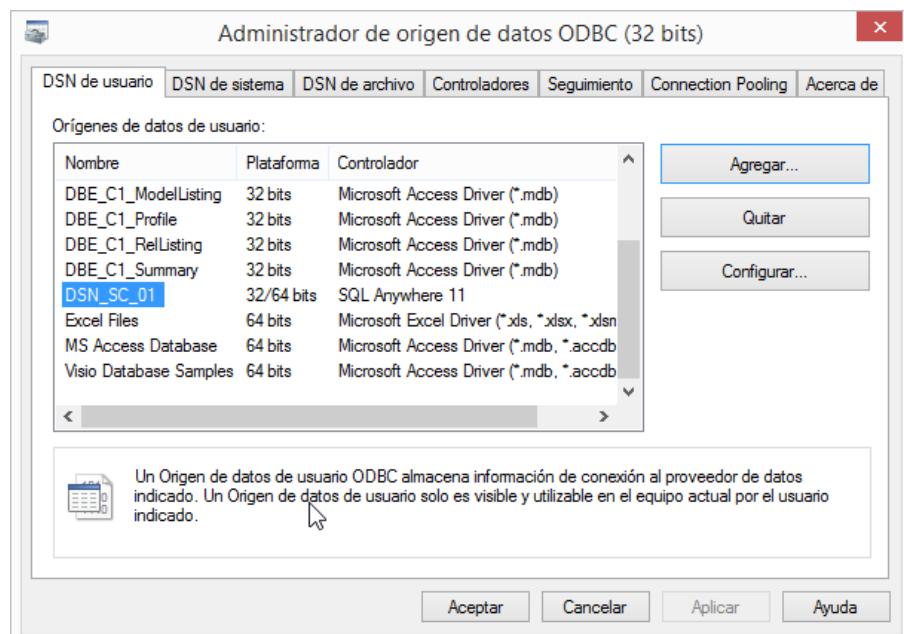
En la siguiente ventana, en la primera pestaña ODBC, le damos nombre al DSN, como DSN_SC_01 (data source name_ sybase central_01), para saber que es nuestro primer origen de datos de una base de datos creada desde sybase central, en la segunda pestaña login indicamos el UserId como dba y el password por defecto sql y en la pestaña Database seleccionamos el archivo base de datos (.db) que hemos creado con Sybase central.



Previamente, podemos efectuar un test de conexión, para verificar que estamos accediendo correctamente al origen de datos, en la pestaña ODBC click en Test Connection, de ser correcto, nos debe devolver el mensaje de conexión exitosa (Connection successful)



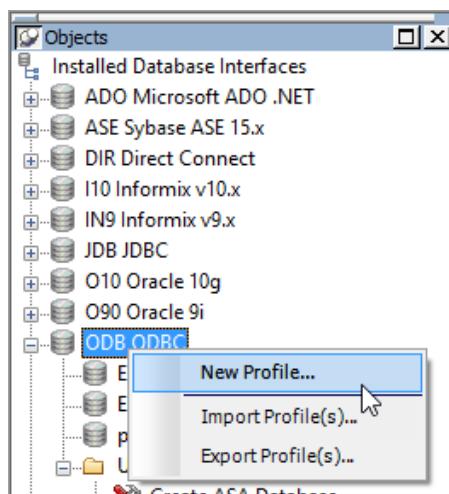
Luego de verificar la conexión, verificamos también que el origen de datos de usuario se ha creado. Aceptar.



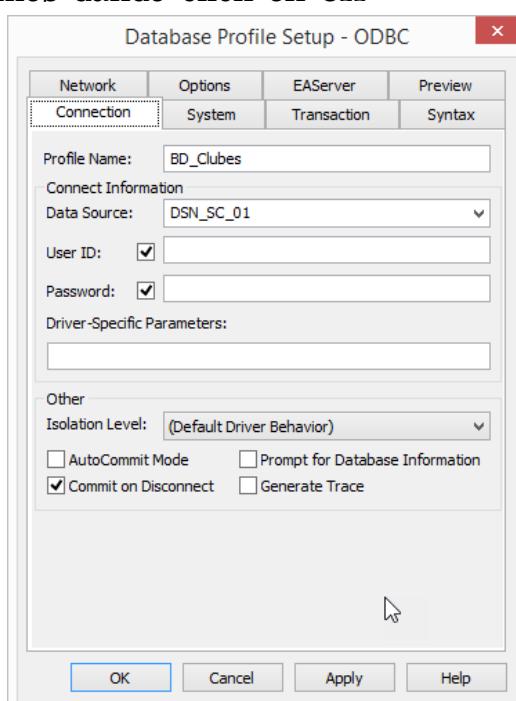
Creando un Perfil

Teniendo una fuente de datos creada, se pueden tener a diversos usuarios accediendo a la base de datos a través de un perfil personal el cual utiliza diferentes parámetros de conexión para acceder a los mismos datos (id de usuario y password distintos, por ejemplo).

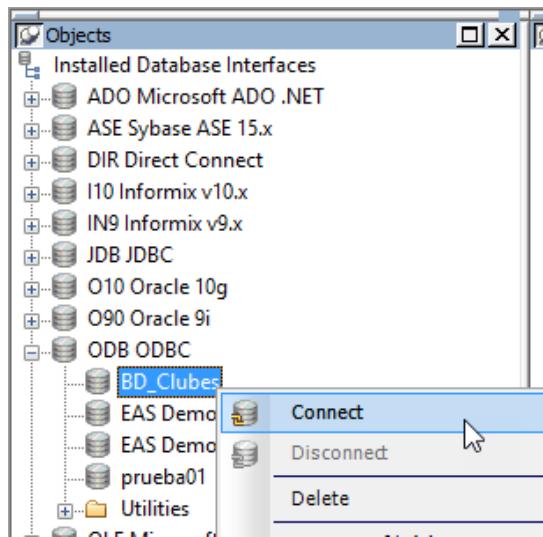
En el IDE PowerBuilder, en el panel de Objetos, seleccionamos el objeto ODB ODBC, le damos click derecho y seleccionamos New Profile



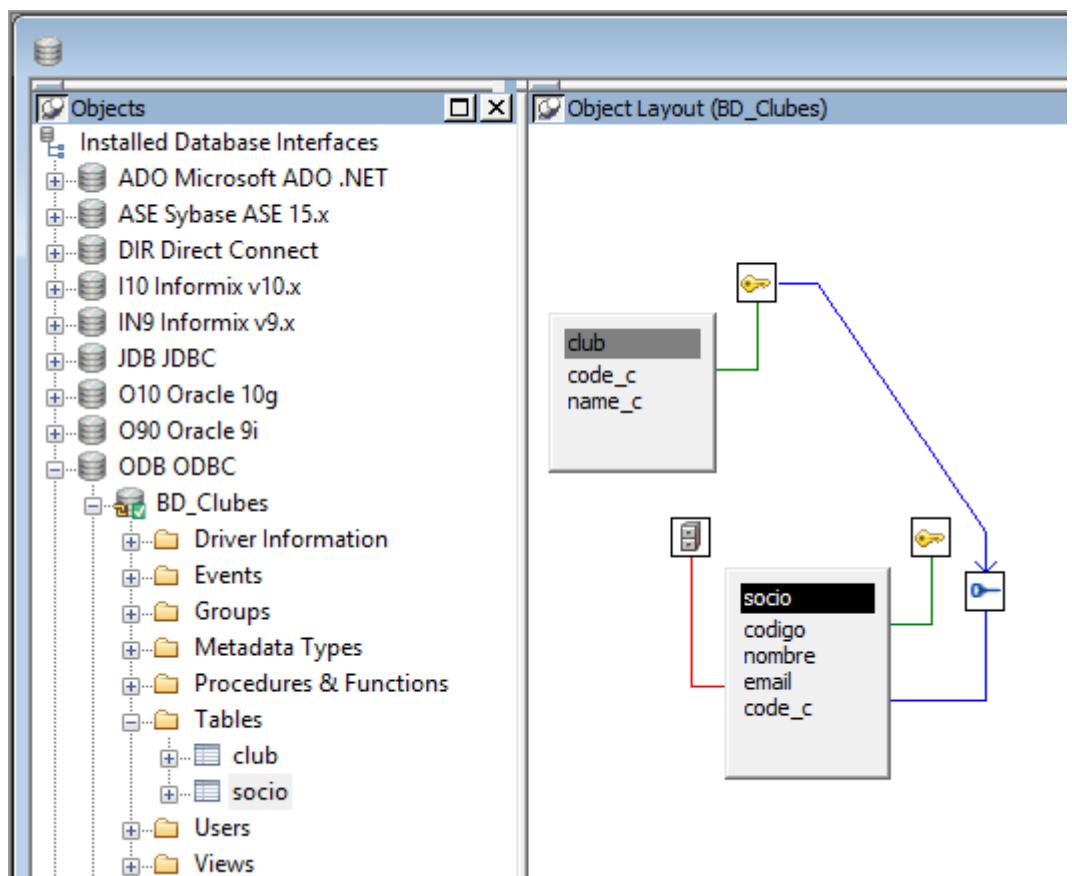
Le damos un nombre al perfil, en nuestro caso: BD_Clubes, seleccionamos el origen de datos previamente creado y por defecto dejamos los User Id y Password y demás campos, proseguimos dando click en OK



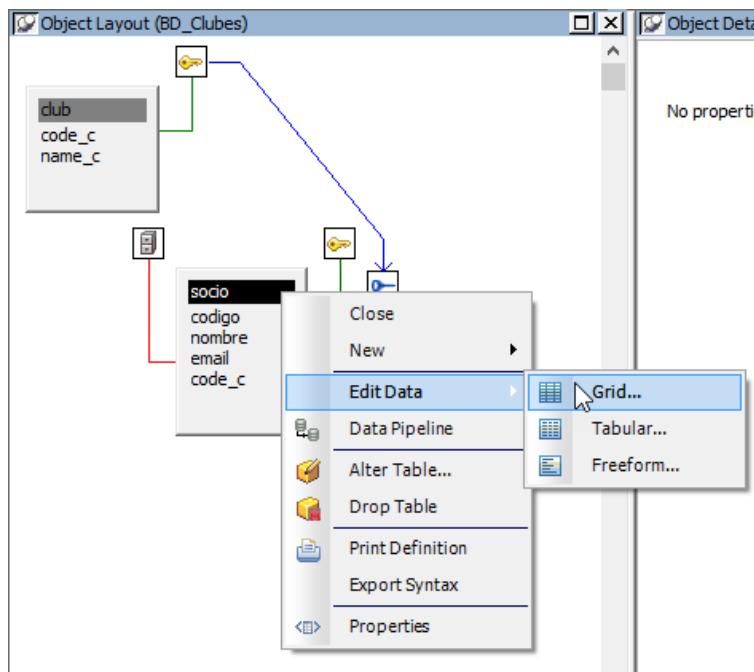
Una vez creado el perfil de acceso: BD_Clubes a la Base de datos: BD_SC_prueba01, creado con Sybase Central, podemos proceder a conectarla, haciendo click derecho en BD_Clubes y Connect



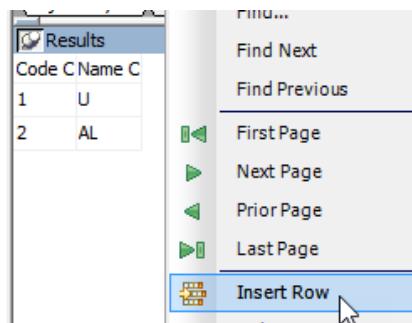
La base de datos está conectada y podemos trabajar con ella, en el panel derecho se ha desplegado gráficamente las tablas que se habían creado desde SYBASE Central.



Desde el cliente, podemos ingresar más datos, clik derecho en una tabla, luego seleccionar Edit Data, tipo Grid



En el panel de resultados, click derecho Insert Row

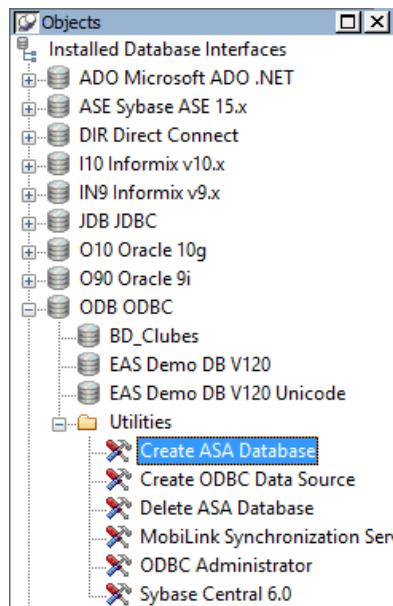


Insertamos más datos en la tabla CLUB y guardamos los cambios con Save Changes

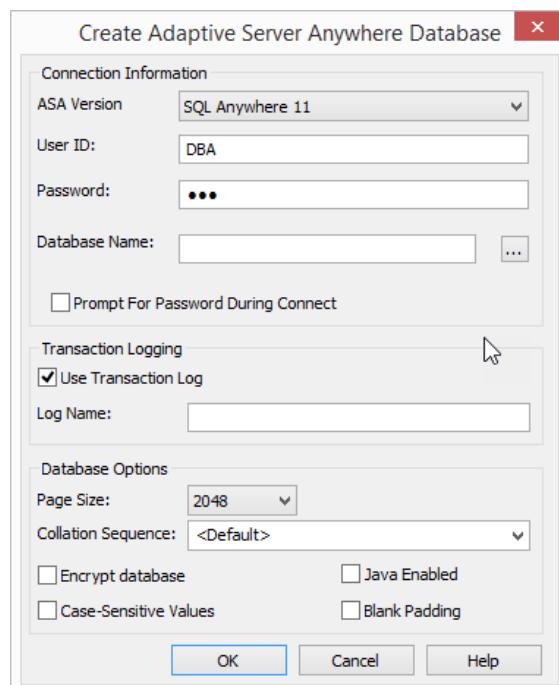
Lo mismo con la tabla SOCIO

7.4.4. Creando una Base de datos con ODBC

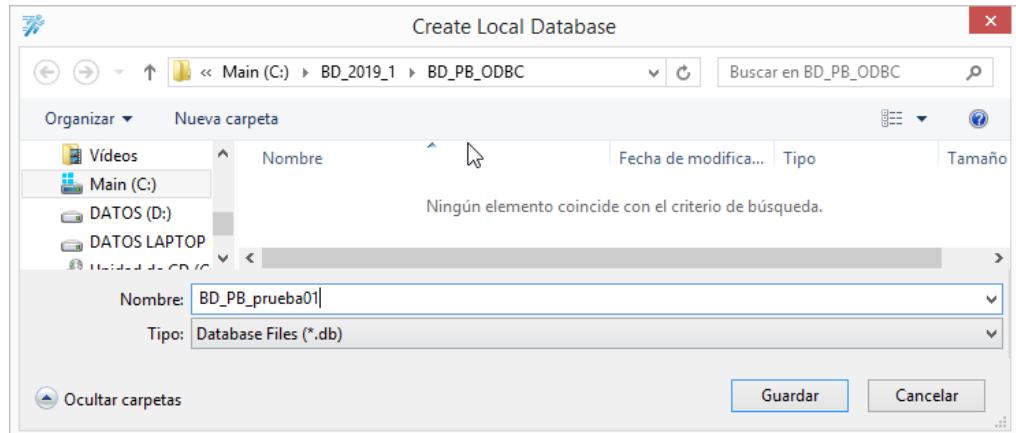
Desde el cliente, vía ODBC podemos crear una base de datos, y desde el IDE PowerBuilder es más simple y sencillo



Desde el panel de objetos del IDE PowerBuilder, seleccionamos el objeto ODB ODBC, desplegamos utilities y seleccionamos: Create ASA Database para ello hacemos doble click y tenemos:

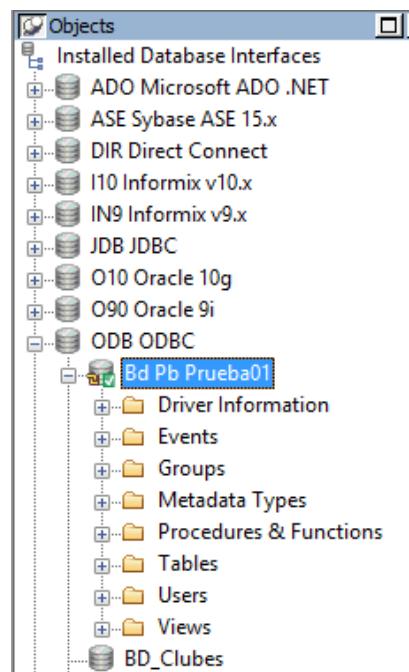
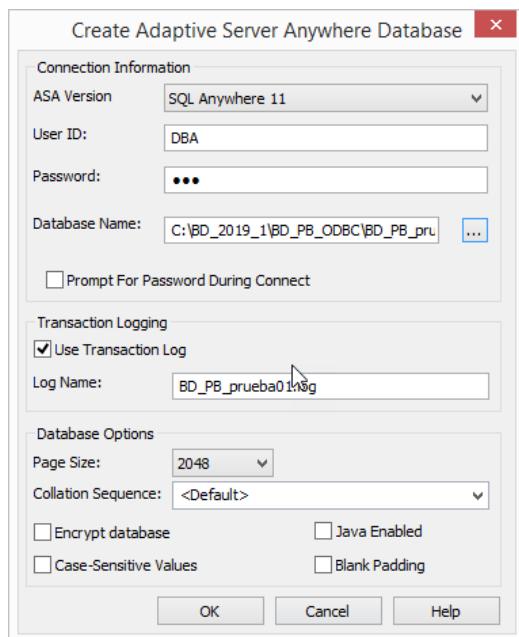


En esta ventana, debemos dar nombre a la base de dato (Database name), hacemos click en el botón para acceder al browser del explorador de archivos.



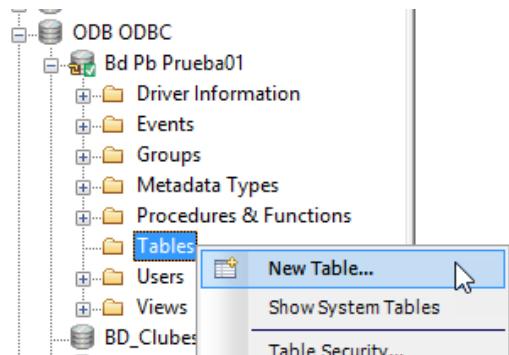
Seleccionamos ruta (disco y carpeta) y le asignamos un nombre a nuestra BD, enseguida click en guardar

Reconocida la ruta y el nombre de la BD, se crea por defecto nombre en espejo del archivo log, por defecto el User ID: es sql y el password: sql



Aceptamos haciendo click en OK.
Y tendremos la BD creada

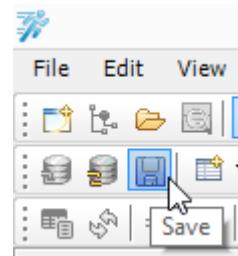
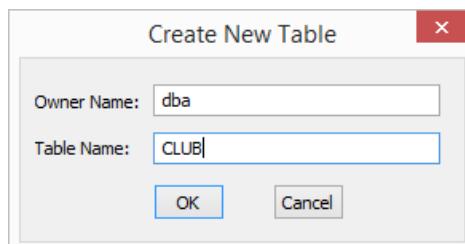
Enseguida creamos las tablas, click derecho en la carpeta Tables de la BD_PB_prueba01 y elegimos New Table



En el panel inferior, declarar cada columna de nuestra nueva tabla CLUB, como sigue:

Column Name	Data Type	Width	Dec	Null	Default
code_c	char	3		No	(None)
name_c	char	10		No	(None)

Para conservar la estructura, la debemos grabar con la opción Save, enseguida nos solicitará que le asignemos un nombre:

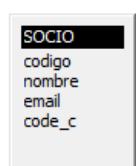


Siguiendo los pasos anteriores, creamos también la Tabla SOCIO, con las siguientes columnas, y grabamos

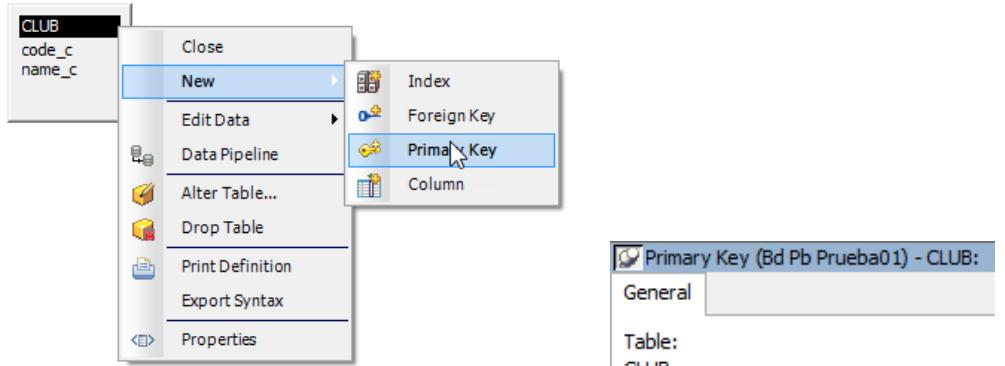
Column Name	Data Type	Width	Dec	Null	Default
codigo	char	3		No	(None)
nombre	char	10		No	(None)
email	char	15		No	(None)



Quedando así:

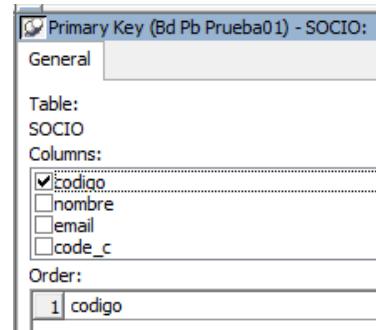


Enseguida crearemos las claves primarias, empezamos con la tabla CLUB, click derecho, New, Primary Key

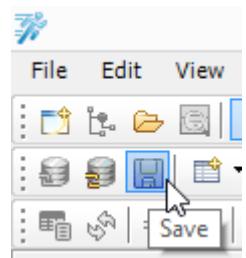


Elegimos el campo o atributo que servirá de PK

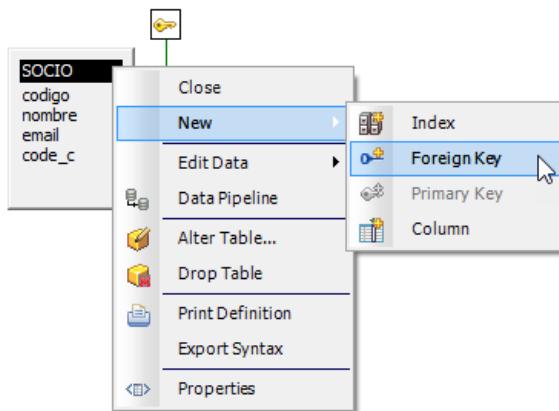
Procedemos de igual manera para la tabla socio



En cada caso debemos grabar para conservar el cambio de la estructura



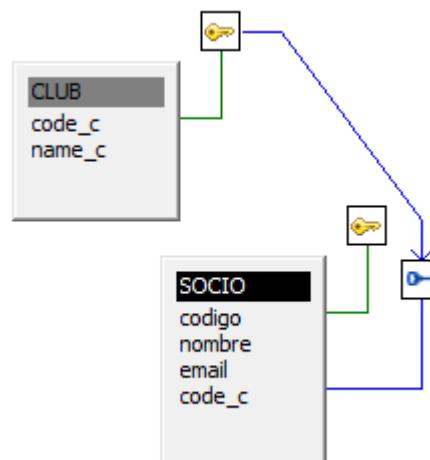
Creando la clave externa entre la tabla SOCIO y la tabla CLUB, para ello hacemos click derecho en la Tabla SOCIO, New, Foreign Key



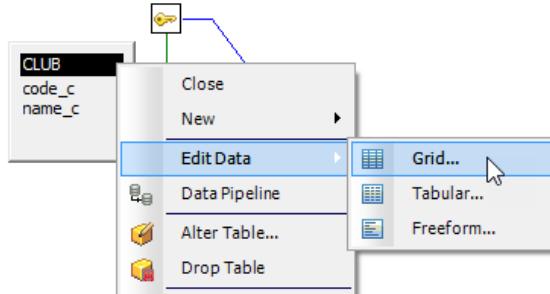
En la pestaña General, seleccionamos el nombre de la Tabla: SOCIO, le damos nombre a la Foreign Key: FK_SOCIO_CLUB y elegimos el atributo FK que se referenciará a una PK de otra tabla, que precisamente se elige en la pestaña Primary Key, en nuestro caso, Table: CLUB y columns: code_c, en la tercera pestaña Rules, dejamos la cinta azul por defecto en la primera regla.

Grabamos

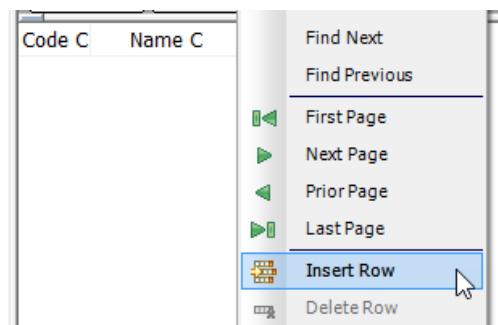
Quedando la Foreign Key creada y gráficamente tendría la siguiente vista:



Creada la Foreign Key, procedemos a poblar las tablas con datos, hacemos click derecho en la tabla CLUB, Edit Data, y seleccionamos Grid



En la cuadricula (grid), click derecho, Insert Row



Insertamos algunos datos de prueba para la tabla CLUB

Code C	Name C
1	U
2	AL
3	SC
4	JG

De igual manera procedemos para la tabla SOCIO

Codigo	Nombre	Email	Code C
1	Juan	juan@uns	1
2	Luis	luis@uns	2
3	Jose	jose@uns	3
4	Ivan	ivan@uns	4
5	Adan	adan@uns	1
6	Saul	saul@uns	2

7.4.5. Aplicando las restricciones de integridad en Base de datos

Creada la base de datos, sus tablas, primary keys, foreign keys y poblada con datos cada tabla, procederemos a verificar las restricciones de integridad del modelo relacional propuesto por E.F. Codd.

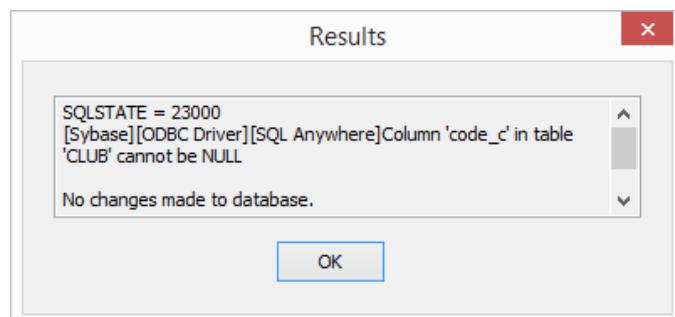
a) Regla de Integridad de las Entidades

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

Por ejemplo, si intentamos guardar un equipo más: César Vallejo (CV), y le damos salvar cambios

Code C	Name C
1	U
2	AL
3	SC
4	JG
5	CV

Nos devolverá el siguiente mensaje:



Que dice, la columna code_c en la tabla CLUB no puede aceptar nulos, porque no le hemos asignado valor a la columna code_c que es la clave primaria.

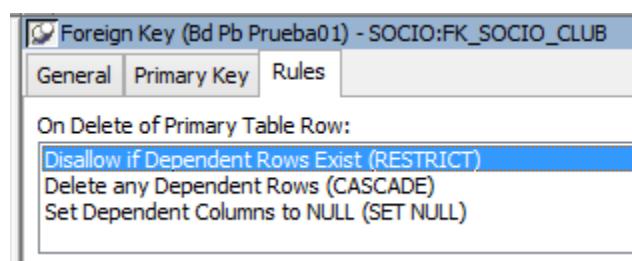
Verificándose la implementación de dicha restricción de integridad en la herramienta gestora de base de datos SYBASE SQL Adaptative Server Anywhere

b) Regla de integridad referencial (para Claves Ajenas)

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

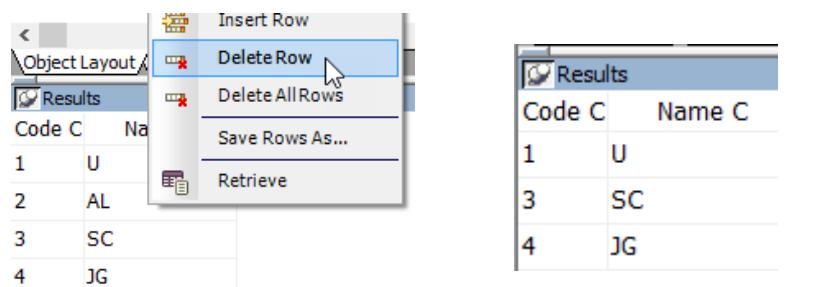
Esta regla es la que viene por defecto cuando creamos una Foreign Key

1. Disallow if dependent rows Exist (Restrict)



Se verifica, cuando queremos eliminar una clave primaria en la tabla CLUB, la cual tiene referencias (hijos) en la tabla SOCIOS.

Por ejemplo, eliminar el club Alianza Lima (AL) como sigue:



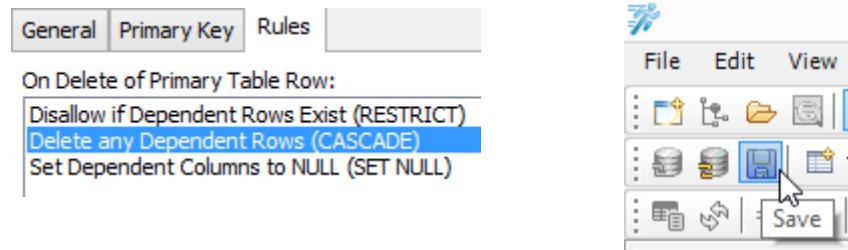
Cuando intentamos salvar los cambios, tendremos el siguiente mensaje:



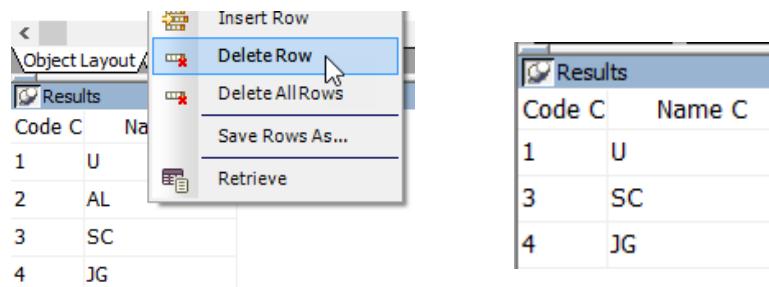
La clave primaria de fila en la tabla CLUB es referenciada por la clave foránea FK_SOCIO_CLUB en la tabla socio.

2. Delete any dependent rows (Cascade)

Probamos cambiando a la segunda regla, en la foreign key definida previamente y guardamos la nueva regla.



Eliminamos el club Alianza Lima (AL), click derecho, Delete Row:



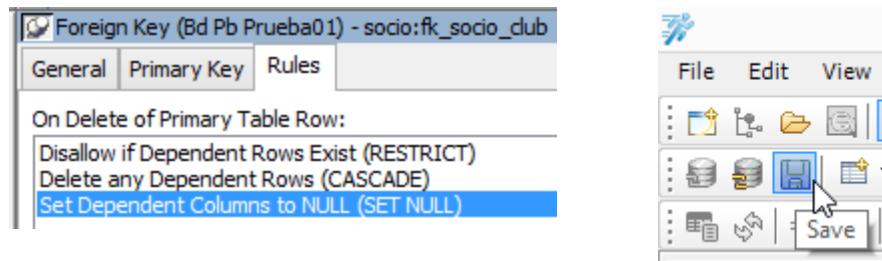
Salvamos los cambios



Y los registros, que tenían como clave foránea al club Alianza Lima en la tabla SOCIO han sido eliminados en cascada, junto con el club Alianza Lima (AL) en la tabla CLUB

3. Set dependent columns to NULL (SET NULL)

Finalmente, cambiamos a la tercera regla, guardamos



Procedemos como el anterior y eliminamos al club la U de la tabla CLUB, y salvamos los cambios

Code C	Name C
3	SC
4	JG

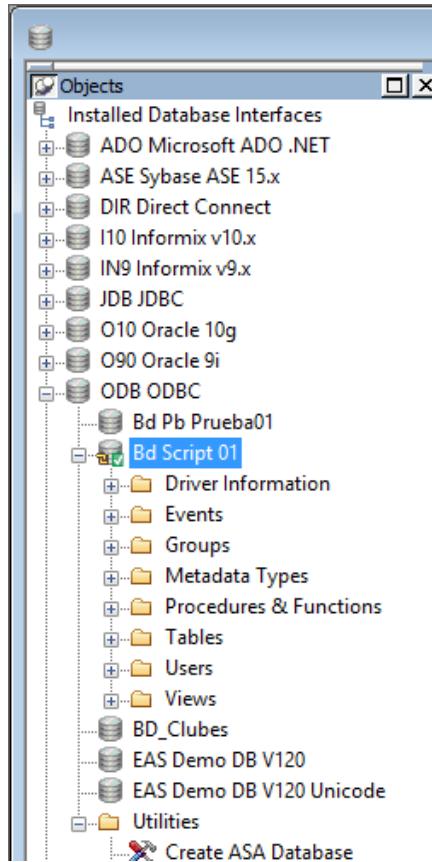
Codigo	Nombre	Email	Code C
1	Juan	juan@uns	
3	Jose	jose@uns	3
4	Ivan	ivan@uns	4
5	Adan	adan@uns	

En el panel de resultados, podemos observar que aquellos socios de la U, se han quedado en la tabla SOCIO, pero sin club, habiéndole el gestor de BD asignándole nulos a la columna clave foránea correspondiente.

Debemos tener cuidado cuando utilicemos las reglas CASCADA y SET NULL, porque puede significar perder data que nos es útil, probablemente hay mecanismo de recuperación, pero estos toman tiempo valioso de trabajo.

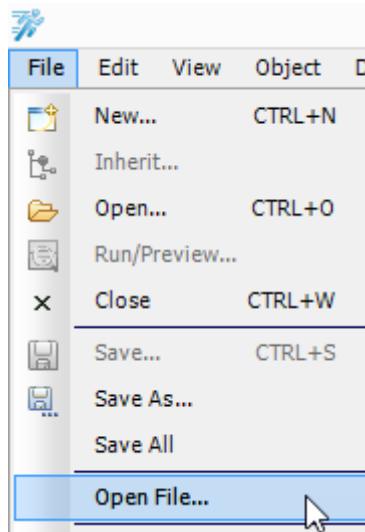
7.5. Creando y poblando una base de datos con un Script.

Siguiendo los pasos anteriores, desde el IDE PowerBuilder, vía ODBC, crear una base de datos de nombre: BD_Script_01

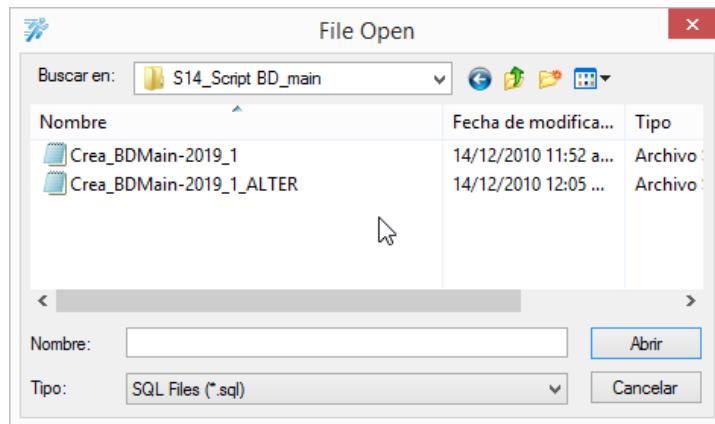


Cargaremos un Script que contiene las sentencias SQL de creación de las Tablas, PK, FK y poblamiento de las tablas con datos mínimos para su utilización.

Desde el menú principal, seleccionamos File, Open File:



Seleccionamos el Script: Crea_BDMain_2019-1 y abrimos



Cargándose, las sentencias contenidas en el Script, en el panel inferior en la pestaña ISQL Session (sesión de SQL interactivo)

```
ISQL Session 1 (Bd Script 01) - Crea_BDMain-2019_1.sql

CREATE TABLE REGIONS(
    REGION_ID INTEGER PRIMARY KEY,
    REGION_NAME CHAR(25),
)

INSERT INTO REGIONS VALUES(1,'Europe');
INSERT INTO REGIONS VALUES(2,'Americas');
INSERT INTO REGIONS VALUES(3,'Asia');
INSERT INTO REGIONS VALUES(4,'Middle East and Africa');

SELECT * FROM REGIONS;

--ALTER TABLE COUNTRIES DROP PRIMARY KEY CASCADE;
```

The screenshot shows the content of the 'ISQL Session 1 (Bd Script 01) - Crea_BDMain-2019_1.sql' window. It contains a CREATE TABLE statement for 'REGIONS' with columns 'REGION_ID' (primary key) and 'REGION_NAME'. It then inserts four rows into the table. Finally, it selects all rows from the 'REGIONS' table. A comment at the end indicates an intention to alter the 'COUNTRIES' table. Below the code, there is a navigation bar with tabs: 'Columns', 'ISQL Session', 'Results', and 'Activity Log'. The 'ISQL Session' tab is active.

Ejecutamos el Script



Creándose las tablas y poblando la Base de datos con los datos contenidos en el Script

Las tablas creadas en el panel de carpetas y algunos datos

ODB ODBC

- Bd Pb Prueba01
- Bd Script 01**
- Driver Information
- Events
- Groups
- Metadata Types
- Procedures & Functions
- Tables
 - countries
 - departments
 - employees
 - job_grades
 - job_history
 - jobs
 - locations
 - regions
- Users
- Views

Select 2 Results (Bd Script 01) - ISQL Session 1

Country Id	Country Name	Region Id
CA	Canada	2
DE	Germany	1
UK	United Kingdom	1
US	United States of America	2

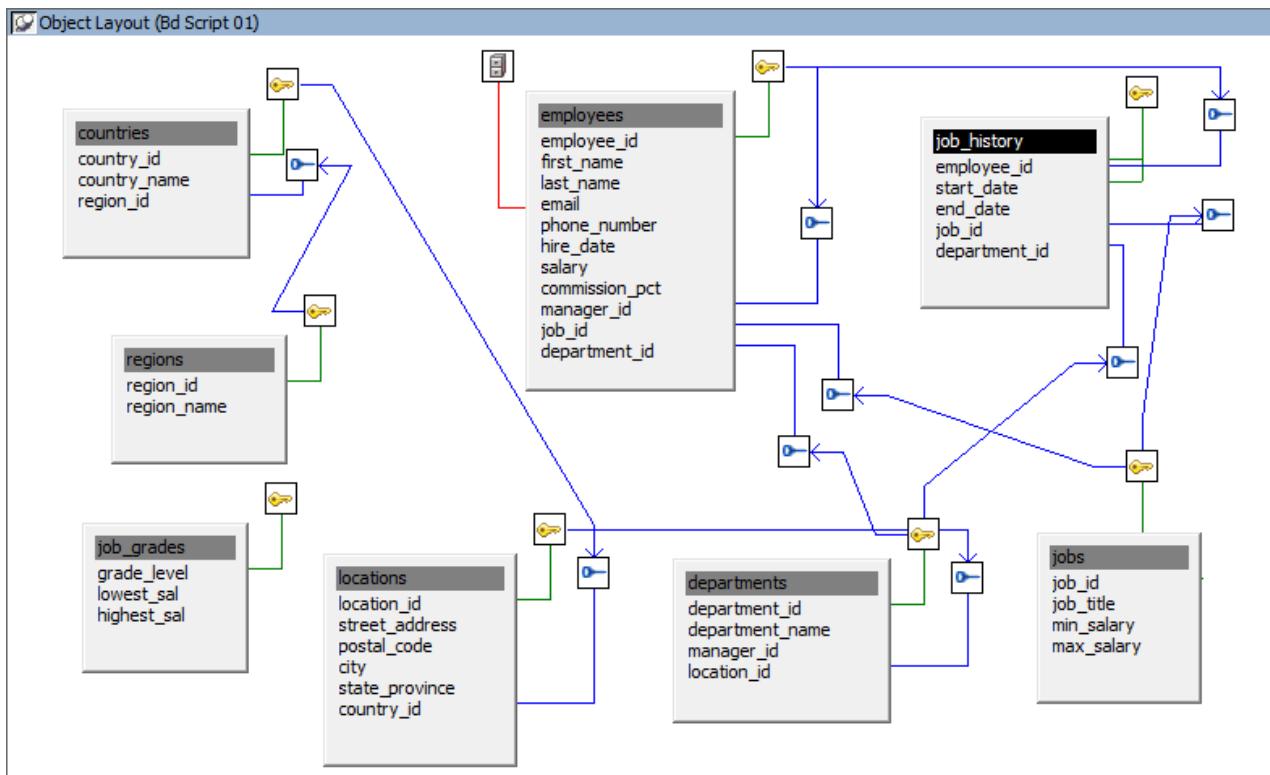
Select 3 Results (Bd Script 01) - ISQL Session 1

Grade Level	Lowest Sal	Highest Sal
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999

Select 4 Results (Bd Script 01) - ISQL Session 1

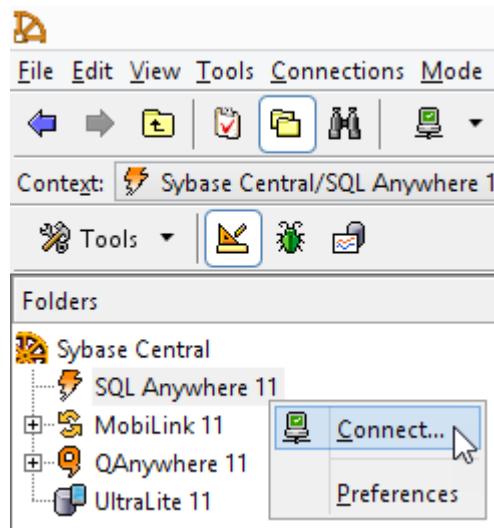
Job Id	Job Title	Min Salary	Max Salary
AD_PRES	President	20000	40000
AD_VP	Administration Vice President	15000	30000
AD_ASST	Administration Administration Assistant	3000	6000
AC_MGR	Administration Accounting Manager	8200	16000

Mirada gráfica de las tablas creadas en el panel objeto Layout.



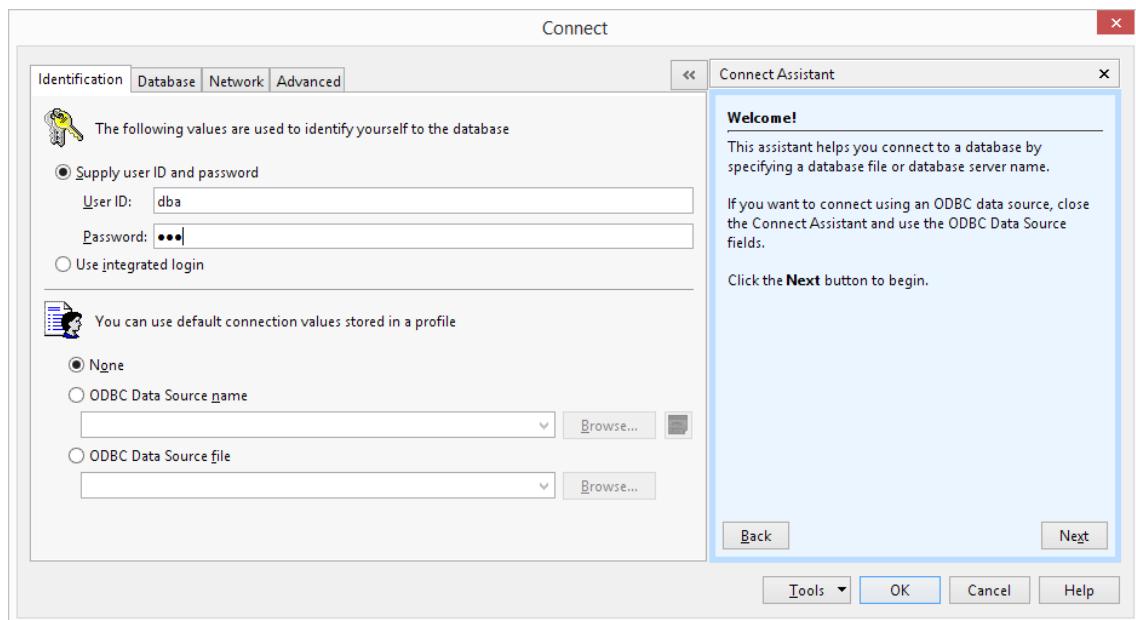
7.6. Abriendo la Base de datos Script con SYBASE Central

Ejecutar SYBASE Central,



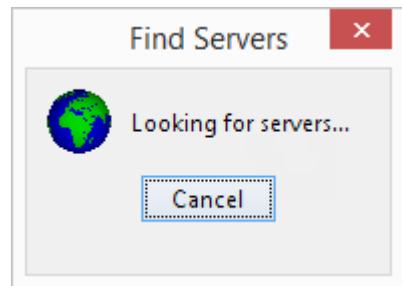
Desde el panel izquierdo Folders, SQL Anywhere 11, Connect

En la ventana emergente, en la primera pestaña Identification, proveer el User ID y el password.

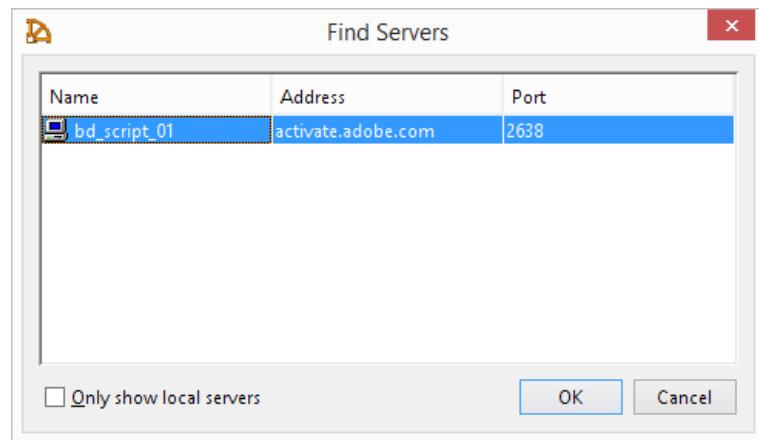


Luego en la siguiente pestaña Database, dado que ya hemos creado la base de datos desde el cliente, haremos una búsqueda del servidor activo, hacemos click en el botón Find.

Buscando servidores



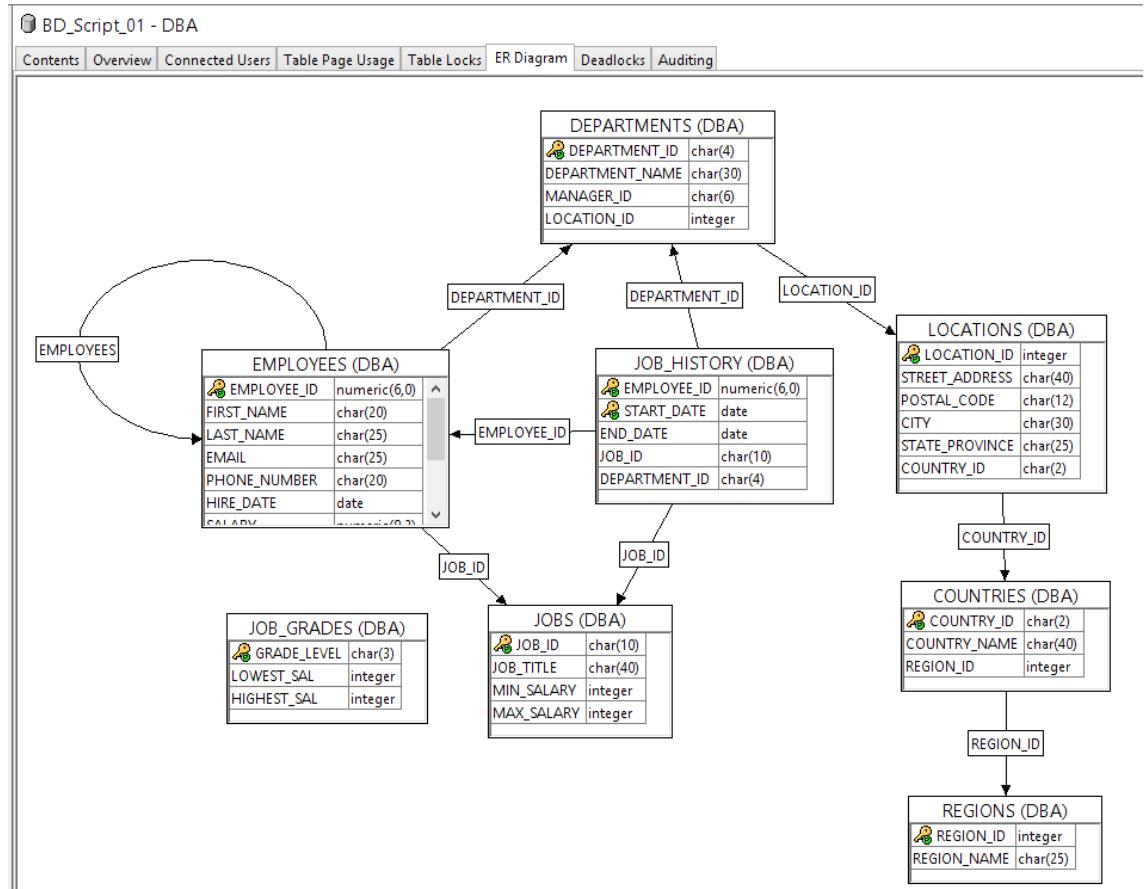
Encontrado el servidor activo: bd_script_01 lo seleccionamos y proseguimos con OK



Levantándose la base de datos desde SYBASE Central

The screenshot shows the Sybase Central application interface. On the left, there's a 'Folders' tree view with nodes for 'Sybase Central', 'SQL Anywhere 11', 'BD_Script_01', and 'BD_Script_01 - DBA'. Under 'BD_Script_01 - DBA', various database objects are listed: Tables, Views, Indexes, Text Indexes, Text Configuration Objects, Triggers, System Triggers, Procedures & Functions, Events, Domains, External Environments, Users & Groups, Login Policies, Login Mappings, SQL Remote Users, Mobilink Users, Publications, SQL Remote Subscriptions, Synchronization Subscriptions, Synchronization Profiles, Dbspaces, Remote Servers, Directory Access Servers, External Logins, Web Services, and Maintenance Plans. The right pane is titled 'BD_Script_01 - DBA' and contains tabs for 'Contents', 'Overview', 'Connected Users', 'Table Page Usage', 'Table Locks', and 'ER Diagram'. The 'Contents' tab is selected, showing the same list of objects as the tree view.

En el panel de la derecha, seleccionar la pestaña ER Diagram
 Y tendremos una vista gráfica de la base de datos.



Capítulo VIII

MEJORANDO LA CALIDAD DE LOS ESQUEMAS DE BASES DE DATOS

8.1 Teoría de la Normalización

Para diseñar una base de datos mediante el modelo relacional, tenemos distintas alternativas para obtener diferentes esquemas relacionales (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, tercera modificada 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 se debe aplicar los mismos cambios en varias tablas o relaciones), así como el problema de pérdida accidental de datos, o la dificultad de representación de determinados hechos.

Algunos problemas que se pueden presentar cuando la base de datos no está normalizada, 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.

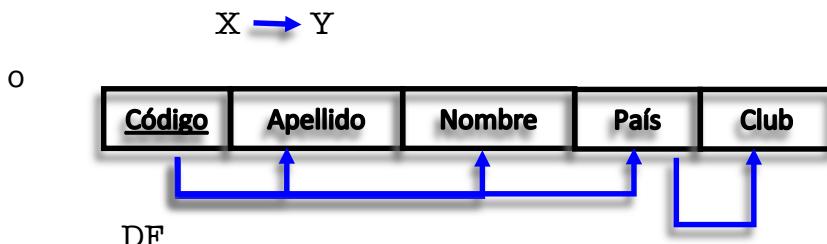
Algunas recomendaciones de diseño a priori de una base de datos relacional a continuación:

- 1° Semántica clara de los atributos. Implica diseñar un esquema relacional fácil de explicar su significado, para ello no se debe combinar atributos de varios tipos de entidad en una sola relación o tabla. Una entidad con sus atributos que le corresponden realmente es muy sencilla de explicar, en contraparte de uno que combina múltiples entidades e inclusive relaciones.
- 2° Tuplas redundantes y anomalías de actualización. Si las relaciones son naturales, es muy bajo el nivel de redundancia, caso contrario acarrearía las anomalías de actualización las cuales incluyen las de inserción, borrado y modificación. Es por ello necesario un análisis de las tablas resultantes para evitar este tipo de anomalías.
- 3° Acerca de los valores NULL. En algunas relaciones las tuplas al tener una gran variedad de atributos, no todos ellos aplican para todos los casos, pudiendo quedar gran cantidad de atributos sin valor alguno, que aparte de desperdiciar espacio de almacenamiento, nos puede inducir a error de interpretación sobre los valores de estos atributos:
 - El atributo no se aplica para la tupla
 - El valor del atributo es desconocido para la tupla, o
 - El valor es conocido, pero es faltante mientras no se grabe.Por ello, en las relaciones base se debe evitar atributos que contengan valores NULL y si no se pudiera evitar, que solo sea aplicado a un mínimo número de tuplas.
- 4° Evitar la generación de tuplas falsas. Puede ocurrir que se ha generado tablas, que incluye más de un atributo iguales en 2 entidades o tablas, y si le aplicamos una UNION NATURAL a estas tablas, se generaría, muchos valores de tuplas falsas, con valores que no tienen sentido ni concordancia. Es por ello que al diseñar un esquema relacional estos deben contener sus atributos para la condición de igualdad a través de PK y FK.

8.2 Dependencias funcionales

Dada una relación R, el atributo Y de R depende funcionalmente del atributo X de R. ($R.X \rightarrow R.Y$). $R.X$ 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).

La dependencia funcional la podemos representar de la siguiente manera:



8.3 Formas Normales

Estamos en construcción



Disculpen la molestia

Referencias Bibliográficas

- [1] Real Academia Española, «Diccionario de la lengua española,» 2018. [En línea]. Available: <https://dle.rae.es/?id=DgIqVCc>. [Último acceso: 2019].
- [2] Definiciona, «Definiciona. Definición y etimología,» 2019. [En línea]. Available: <https://definiciona.com/dato/>. [Último acceso: 2019].
- [3] R. Elmasri y S. B. Navathe, *Fundamentals of Database Systems*, USA: Pearson, 2016.
- [4] G. V. Post, *Database Management Systems. Designing & Building Business Applications*, USA: JerryPost.com, 2011.
- [5] C. J. Date, *An Introduction to Database Systems*, USA: Pearson Education Inc., 2004.
- [6] D. Kroenke, *Procesamiento de Base de Datos. Fundamentos diseño e implementación.*, México: Pearson Educación, 2003.
- [7] H. F. Korth, A. Silberschatz y S. Sudarshan, *Fundamentos de base de Datos*, España: McGraw Hill, 2002.
- [8] S. G. o. D. B. M. System, *Interim Report ANSI/X3/SPARC*, Washington: ANSI, 1975.
- [9] C. Batini, S. Ceri y S. Navathe, *Diseño Conceptual de Base de Datos*, USA: Addison-Wesley/Díaz de Santos, 2000.
- [10] P. P.-S. Chen, «The Entity-Relationship Model - Toward a Unified View of Data,» *ACM Transactions on Database Systems*, vol. 1, nº 1, pp. 9-36, 1976.
- [11] A. de Miguel Castaño y M. G. Piattini Velthuis, *Concepción y Diseño de Base de Datos. Del Modelo E/R al Modelo Relacional*, Wilmington, Delaware: Addison-Wesley Iberomaericana, S.A., 1993.
- [12] A. de Miguel, P. Martínez, E. Castro, J. M. Cavero, D. Cuadra, A. M. Iglesias y C. Nieto, *Diseño de base de datos. Problemas resueltos*, Madrid: Alfaomega - Rama, 2000.
- [13] A. de Miguel Castaño, M. Piattini Velthuis y E. Marcos Martínez, *Diseño de Bases de Datos Relacionales*, México: Alfaomega Grupo editor S.A., 2000.
- [14] SAP, «PowerDesigner Data Modeling,» 22 02 2016. [En línea]. Available: https://www.powerdesigner.biz/documentations/powerdesigner-16.6-documentation-en/data_modeling.pdf. [Último acceso: 15 02 2019].
- [15] E. F. Codd, «A Relational Model of Data for Large Shared Data Banks,» *Communications of the ACM*, vol. 13, nº 6, pp. 377-387, Junio 1970.
- [16] S. iAnywhere, *SQL Anywhere 16 Introduction*, SAP AG, 2014.