UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

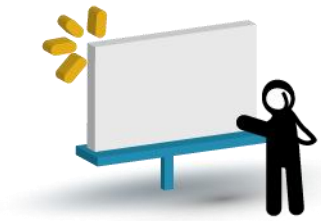
Centro de e-Learning

Diplomatura en Bases de Datos

Centro de e-Learning SCEU UTN - BA.
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Módulo 1: Introducción y Fundamentos de Bases de Datos

Unidad 2: Bases de datos relacionales



Presentación:

En esta Unidad exploramos el concepto de una base de datos relacional. Vemos cómo surgen las formas normales y cómo diseñar repositorios de datos teniendo en cuenta esos criterios. Comprendemos, también, la diferencia entre modelado lógico y modelado físico, para ser capaces de resolver ejercicios de diseño de repositorios de datos.

Objetivos:

Que los participantes:

- Entiendan el modelo relacional
- Tomen conocimiento de las formas normales y como implementarlas en un diseño
- Vean las diferencias entre los distintos tipos de modelados.
- Practiquen el diseño de modelos de datos.

Bloques temáticos:

1. ¿Qué es una base de datos relacionales?
2. ¿Qué son las formas normales?
3. Modelo lógico de una base de datos relacional
4. Modelo físico de una base de datos relacional
5. Ejemplos de modelado a cargo del alumno.

Consignas para el aprendizaje colaborativo

En esta Unidad los participantes se encontrarán con diferentes tipos de actividades que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

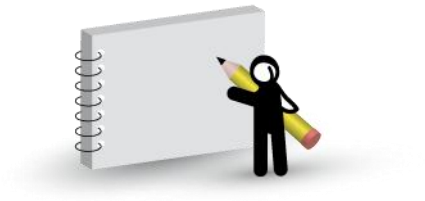
- Los foros proactivos asociados a cada una de las unidades.
- La Web 2.0.
- Los contextos de desempeño de los participantes.

Es importante que todos los participantes realicen algunas de las actividades sugeridas y compartan en los foros los resultados obtenidos.

Además, también se propondrán reflexiones, notas especiales y vinculaciones a bibliografía y sitios web.

El carácter constructivista y colaborativo del MEC nos exige que todas las actividades realizadas por los participantes sean compartidas en los foros.

**** El MEC es el modelo de E-learning colaborativo de nuestro Centro.***



Tomen nota:

Las actividades son opcionales y pueden realizarse en forma individual, pero siempre es deseable que se las realice en equipo, con la finalidad de estimular y favorecer el trabajo colaborativo y el aprendizaje entre pares. Tenga en cuenta que, si bien las actividades son opcionales, su realización es de vital importancia para el logro de los objetivos de aprendizaje de esta instancia de formación. Si su tiempo no le permite realizar todas las actividades, por lo menos realice alguna, es fundamental que lo haga. Si cada uno de los participantes realiza alguna, el foro, que es una instancia clave en este tipo de cursos, tendrá una actividad muy enriquecedora.

Asimismo, también tengan en cuenta cuando trabajen en la Web, que en ella hay de todo, cosas excelentes, muy buenas, buenas, regulares, malas y muy malas. Por eso, es necesario aplicar filtros críticos para que las investigaciones y búsquedas se encaminen a la excelencia. Si tienen dudas con alguno de los datos recolectados, no dejen de consultar al profesor-tutor. También aprovechen en el foro proactivo las opiniones de sus compañeros de curso y colegas.

1. ¿Qué es una base de datos relacional?

Tenemos que empezar por distinguir entre dos entidades:

- Base de datos
- Sistema de base de datos

El sistema de base de datos es todo el conjunto de hardware y software que brinda el soporte necesario para manejar la base de datos permitiendo:

- Agregar nuevos archivos vacíos a la base de datos
- Insertar datos dentro de los archivos existentes
- Recuperar datos de los archivos existentes
- Modificar datos en los archivos existentes
- Eliminar datos de los archivos existentes
- Eliminar archivos existentes dentro de la base de datos

Todo esto nos deja el problema de definir la propia base de datos

Empecemos por mostrar con un ejemplo a que nos referimos con archivos, datos y las operaciones.

Asumamos que estamos mirando el problema de la administración de los cursos virtuales de la universidad.

Para empezar queremos un archivo donde tengamos todos los cursos. Ese archivo podría tomar la siguiente forma:

Identificador	Descripción	Estado	Próximo Inicio
1	Diplomatura en Base de datos	Vigente	01/03/2018
2	Diplomatura en análisis de negocios	Vigente	01/02/2018
3	Diplomatura en inteligencia de	Vigente	01/04/2018

	negocios		
4	Diplomatura en Big Data	En preparación	NULL
5	Especialista en Apache Hadoop	Cancelado	NULL
6	Datascientist con R	Vigente	01/03/2018

¿Qué aprendemos de este archivo?

Tiene una estructura regular.

Vemos columnas que contienen todos elementos del mismo tipo y con la misma clase de información, por ejemplo la primer columna contiene siempre un número que se usa para identificar cada uno de los cursos.

La segunda columna tiene los títulos de los cursos y, por supuesto, siempre son cadenas de caracteres (letras)

La tercera columna contiene el estado del curso. Al parecer un curso puede estar en varios estados. Todos parecen tener cadenas de caracteres.

Finalmente la última columna contiene la fecha. Todos los elementos de la columna deben ser fechas.

Miremos ahora la otra dimensión: las filas. Vemos que cada fila nos habla de un curso distinto. A cada fila nos referíamos cuando decíamos un dato de un archivo.

Fijando ideas, el archivo es el conjunto de filas. Cada fila es un dato. Y tenemos además la restricción de que todas las filas deben ser isomorfas, esto es, tener la misma forma. (un número, un texto para el título, un texto para el estado y una fecha)

Para mayor claridad vamos a usar colores:

Identificador	Descripción	Estado	Próximo Inicio
1	Diplomatura en Base de datos	Vigente	01/03/2018
2	Diplomatura en análisis de negocios	Vigente	01/02/2018
3	Diplomatura en inteligencia de negocios	Vigente	01/04/2018
4	Diplomatura en Big Data	En preparación	NULL
5	Especialista en Apache Hadoop	Cancelado	NULL
6	Datascientist con R	Vigente	01/03/2018

Con fondo ROJO marco una fila. Con fondo AMARILLO marco una columna. Justo en la intersección entre la fila y la columna pongo el fondo en rojo y el de las letras en amarillo para sugerir que esa intersección es, a la vez, parte de la fila y la columna.

En términos de base de datos solemos llamar REGISTROS a las filas y CAMPOS a las columnas.

Como las columnas tienen que responder a un único concepto deben ser siempre del mismo tipo de datos. Por ejemplo: Fecha.

Las filas responden siempre a una única realidad:

- identificador del curso de Diplomatura en Inteligencia de Negocios
- titulo del curso de Diplomatura en Inteligencia de Negocios
- estado del curso de Diplomatura en Inteligencia de Negocios

- fecha de inicio del próximo curso de Diplomatura en Inteligencia de Negocios

Nota Conceptual:

Tabla: es un conjunto de filas y columnas que trae información sobre una entidad en particular. Cada individuo de esa entidad se describe en una fila. Dentro de esa fila las diferentes columnas tienen los atributos de esa entidad.

Registro: es el nombre que le damos a una fila de una tabla. Los registros son homogéneos en el sentido en que se refieren a una única instancia de una entidad.

Campo: es el nombre que le damos a una columna de una tabla. Los campos son homogéneos en el sentido en tratan siempre de un mismo atributo. Por este motivo tienen que ser siempre del mismo tipo de datos.

Operaciones básicas:

Vamos a empezar a ver, aún en forma genérica las operaciones básicas que hemos usado en nuestra descripción de las bases de datos:

- Insertar nuevos registros en una tabla
- Consultar registros de una tabla
- Modificar registros de una tabla
- Eliminar registros de una tabla

La forma concreta de instrumentar estas instrucciones puede cambiar con la herramienta concreta que estemos utilizando y su versión. Vamos a repasar estas diferencias cuando vayamos repasando las distintas herramientas en los capítulos subsiguientes.

También quedará para ver más adelante las instrucciones con las que creamos tablas y las destruimos.

En este lugar se recomienda a los participantes realizar el Ejercicio Conceptual 2.1 el Ejercicio Práctico 2.1.

Agregado de datos:

Para incorporar registros a una tabla la sentencia típica es **INSERT**

La sintaxis más sencilla es:

INSERT INTO [nombre de la tabla]([lista de campos]) values([lista de valores])

entre corchetes angulares [] pongo explicaciones

los paréntesis son requeridos por la sintaxis.

Ejemplo:

Supongamos que para el ejercicio conceptual 2.1 creamos una tabla Jugadores con los campos Nombre, Edad, Posición y DNI entonces la sentencia se escribiría:

INSERT INTO Jugadores(Nombre,Edad,Posicion,DNI) values('Alberto Perez', 21, 'Delantero', 31222333)

Hay que hacer notar que las cadenas de caracteres que usamos para la información del nombre y de la posición las anotamos entre comillas simples '

En este lugar se recomienda a los participantes realizar los ejercicios conceptual 2.2 y práctico 2.2

Recuperación de datos:

La sentencia básica para la recuperación de datos es SELECT

La sintaxis básica de SELECT es:

SELECT [lista de campos] FROM [nombre de la tabla]

Ejemplo:

SELECT Nombre,Edad,Posicion,DNI FROM Jugadores

El orden y la cantidad de campos no necesita corresponder con el que usamos en la definición de la tabla.

Si introducimos un campo que no existe en la definición de la tabla obtendremos un mensaje de error.

Si omitimos un campo simplemente no recibimos esa información pero no se produce error alguno.

En este punto se recomienda a los participantes realizar los ejercicios conceptual 2.3 y práctico 2.3

Borrado de registros:

La setencia con la que le ordenamos al sistema de base de datos que borre un registro es DELETE

La sintaxis es:

DELETE FROM [nombre de la tabla] WHERE [condición que deben cumplir los registros que queremos borrar]

Ejemplo:

Quiero borrar los arqueros:

DELETE FROM Jugadores WHERE Posicion = 'Arquero'

En este punto se recomienda los participantes la realización de los ejercicios conceptual 2.4 y práctico 2.4

Modificación de registros:

Modificar un registro significa alterar alguno o todos sus campos.

La sentencia que se utiliza es UPDATE

El uso básico es:

UPDATE [nombre de la tabla] SET [nombre del campo] = [valor nuevo] WHERE [condición que deben cumplir todos los registros que se modificarán]

Ejemplo:

UPDATE Jugadores SET Posicion = 'Arquero Suplente' WHERE Nombre = 'José Sanchez'

En ese punto se recomienda a los alumnos la realización de los ejercicios conceptual 2.5 y práctico 2.5

2. ¿Qué son las formas normales?

Cuando nos apartamos del caso de una sola tabla empezamos a tener varias formas de registrar las mismas cosas.

Empecemos por plantear un ejemplo que nos permita discutir ventajas y desventajas de hacer las cosas de un modo o de otro.

Vamos a considerar como podemos registrar todas las facturas que una empresa realiza. En una factura vamos a tener una fecha, los datos del cliente, los datos de los distintos artículos que ha comprado, un total y algunas consideraciones sobre el forma de pago e impuestos.

¿Cómo podemos registrar esto?

Empecemos con la aproximación más ingenua posible.

Podríamos armar una gran tabla que se llame Facturas donde tendríamos todos los datos del cliente todos los datos del artículo 1, todos los datos del artículo 2, todos los datos del artículo 3 y así siguiendo tantos artículos como existan en la factura papel. Al final agregaríamos los datos sobre la forma de pago, los impuestos correspondientes y el total.

¿Es una buena opción?

Consideremos algunas desventajas:

Voy a tener un montón de espacio en blanco ya que la mayoría de los clientes compra uno o dos artículos.

Si el nombre del cliente los escriben de diferentes formas me va a costar totalizar todas sus facturas y por lo tanto cobrarle su deuda.

Si los artículos se escriben de distinta forma me va a costar totalizar las ventas y por lo tanto llevar el stock.

Si la ciudad, la calle, la provincia, la registro a mano, seguramente la creatividad de los operadores multiplicará la cantidad de provincias. (Me acuerdo de haberme encontrado con una Argentina de como 400 provincias en un sistema que tenía este problema)

Podrían dos clientes distintos llamarse José Sanchez. ¿Cómo hago para saber que me compró, y por lo tanto que me debe, cada uno de ellos?

Hemos detectado problemas que tienen que ver con:

- Espacio de almacenamiento
- Consistencia de la información

El espacio de almacenamiento era una consideración muy importante. Como hemos visto en la primer unidad el espacio disponible era muy escaso y muy caro. Esto se ha ido solucionando con el tiempo. Sin embargo la solución es relativa porque, al tiempo que crece el espacio disponible y que bajan los costos se multiplica la cantidad de información que tratamos de registrar. Las consideraciones de espacio, más o menos agudas, seguirán vigentes.

La otra consideración, la de la consistencia, ha aumentado en importancia. Cuantos más son los datos más difícil es realizar luego procesos manuales que reconstruyan la consistencia perdida. Debemos asegurarla desde el principio!

Para mejorar un poco podemos empezar por repasar como definíamos tabla: Filas y columnas que hablan de una entidad... ¿Cuál es la entidad? ¿Las facturas? ¿Los clientes? ¿Los artículos?

Nuestro problema arranca porque tenemos varias entidades mezcladas en una única tabla.

Vayamos separando y veamos qué problemas nos surgen:

Una primera idea puede ser armar una tabla de Clientes:

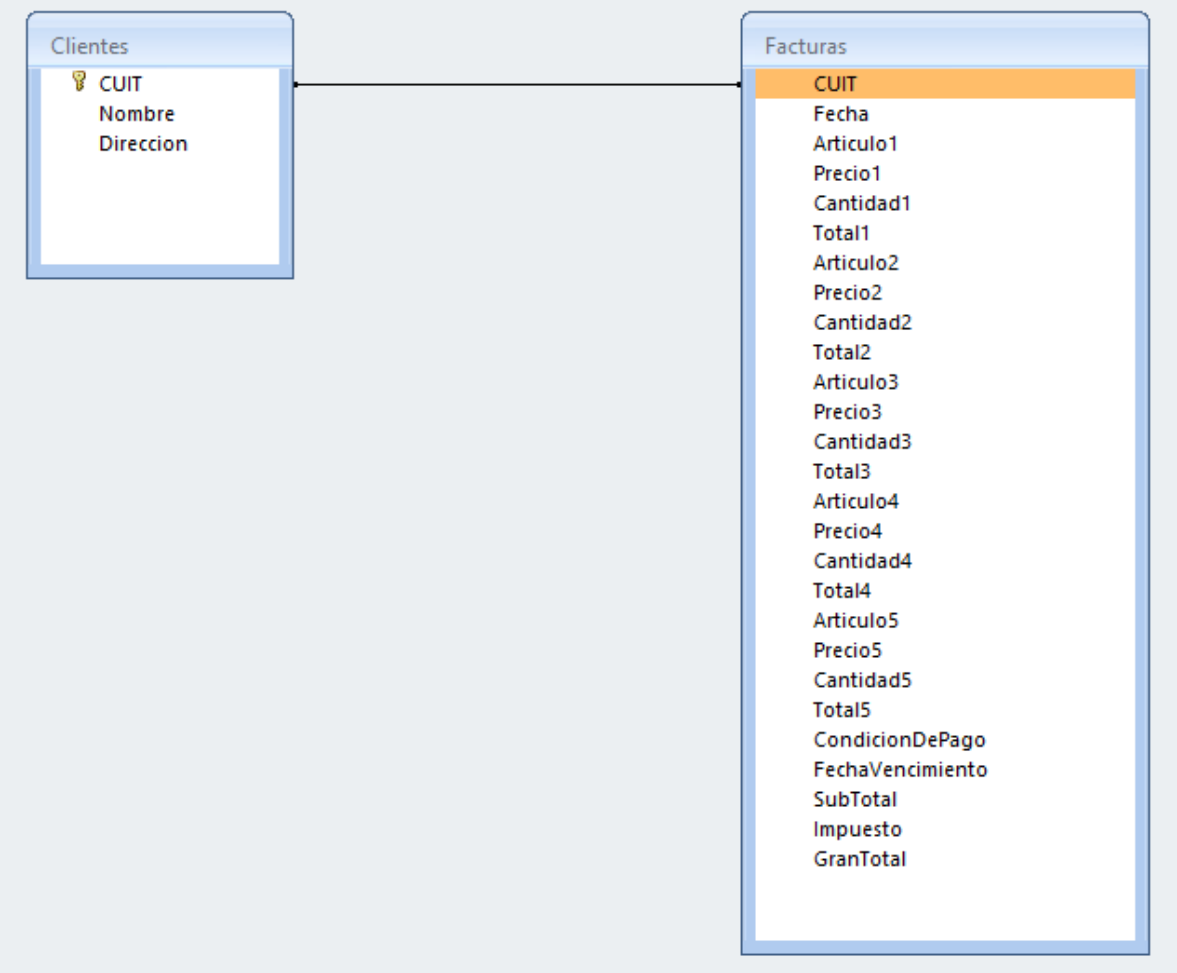
- Nombre
- Dirección
- CUIT

En vez de registrar luego, del lado de las facturas todos los datos del cliente vamos a registrar únicamente su CUIT. Cuando necesitemos encontrar los demás datos del cliente podemos recuperarlos con una operación SELECT de la tabla Clientes aprovechando que sabemos el CUIT.

Nota Conceptual:

Entidad: cualquier objeto del cual queremos registrar información.

Vamos a representar como nos van a quedar los datos:



Los que entiendan algo del negocio de las bases de datos se horrorizarán de lo que estoy dibujando. Paciencia...

Con esto estamos un poco mejor. El nombre del cliente y su dirección quedarán en un solo lugar. Con esto:

- Gano espacio
- Gano en consistencia

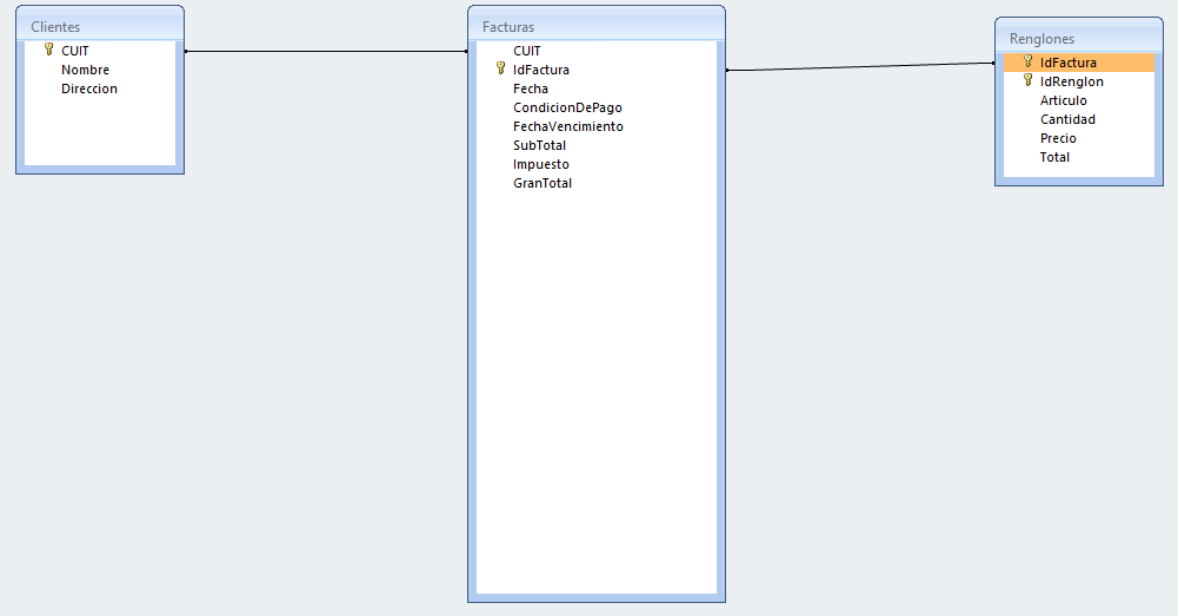
Pero ahora tengo dos tablas y una rayita en el medio que me indica que para completar los datos que me faltan en la tabla de Facturas viajo por esa ruta hacia la tabla Clientes y de allí puedo recuperar los datos que me faltan. Esa ruta es lo que llamamos una "Relación"

Nota Conceptual:
Relación: es un vínculo entre dos entidades que nos permite utilizar la información de una para saber más sobre la otra

¿Podremos utilizar alguna relación para mejorar nuestro almacén de datos, ya sea mejorando la consistencia o ahorrando espacio?

Algo que veníamos observando es que como no todas las facturas tendrán llenos todos los renglones con artículos allí habrá mucho espacio en blanco. ¿Y si reconocemos la existencia de una entidad "renglón de factura"?

El modelo nos quedaría:



Estamos un poco mejor en cuanto al espacio.

Para poder establecer la relación entre los renglones de una factura y la factura misma hemos tenido que identificar a las facturas.

Nota Conceptual:

Clave Primaria: es el identificador único del caso del que trata un registro. Todos los atributos quedan determinados cuando se conoce la clave primaria.
La clave primaria podrá ser simple cuando consta de un único campo o compuesta cuando hay varios campos involucrados.

En este punto se recomienda la realización de los ejercicios conceptual 2.6 y práctico 2.6

¿Cómo venimos con la consistencia? En cuanto a los datos del cliente parecemos estar protegidos. Para cada CUIT se registra un único nombre y una única dirección, no importa cuántas facturas le hagamos a ese cliente y cuantos renglones tengan.

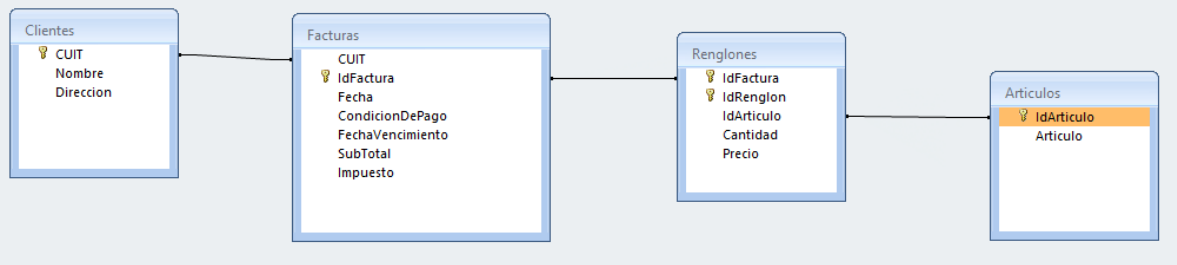
El gran total dentro de la tabla facturas se calcula como el sub total más los impuestos. ¿Qué pasaría si los números que tenemos registrados en una factura no cumplieran esa relación? Pues tendríamos una inconsistencia. Para asegurarnos la consistencia no tendríamos que registrar lo que pueda calcularse a partir de los demás datos. Para solucionar este problema bastaría con eliminar el campo GranTotal.

¿El cálculo del impuesto será también redundante? Al momento de considerar el repositorio de datos no sabemos cómo serán los impuestos. Quizá haya diferentes tasas para diferentes artículos, quizá cambien las tasas, en esta circunstancia el valor no queda predeterminado y deberemos registrarlo.

Veamos como venimos por el lado de los artículos. Para empezar estamos calculando el total que no es otra cosa que el precio por la cantidad.

En este punto se recomienda la realización de los ejercicios conceptuales 2.7 y 2.8 así como los ejercicios prácticos 2.7 y 2.8

Si las cosas nos salieron bien deberíamos obtener algo parecido a:



Alguien podría preguntarse por la consistencia de los precios. Digamos que como, presumiblemente, cambiarán con el tiempo, entonces necesitaríamos una nueva tabla de precios vigentes como función del tiempo. Sin embargo, como los precios podrían ser también función del cliente entonces podemos legítimamente dejarlos tal y como están.

A lo largo de todas estas transformaciones y de muchas más que están por venir empezamos a comprender que alcanzar un modelo de datos eficiente con el espacio y orientado a la consistencia de los datos es algo complejo y necesitamos formalizar las reglas que nuestro esquema debe seguir.

Si todo esto nos pasó arrancando de una única tabla podemos entonces pasarla bastante mal cuando el conjunto de información incluya cientos de ellas. (Para los que opinen que se trata de una exageración les comento que el modelo de SAP, el conocido ERP, tiene actualmente más de 10000 tablas)

El conjunto de reglas que debemos cumplir se conoce en el mundo de las bases de datos como "Formas Normales" y es lo que nos proponemos describir a continuación.

Las formas normales son 6. Representan niveles crecientes de cumplimiento con los objetivos de ahorro de espacio y consistencia.

En la práctica alcanza con llegar a satisfacer el criterio de la tercera forma normal, en particular con la formulación adicional de Boyce y Codd.

Por lo tanto, aunque vamos a tratar de explicarlas todas ya sabemos hasta donde vale la pena (desde una perspectiva práctica) esforzarse.

Primera forma normal:

Para poder decir que una tabla está en primera forma normal debemos asegurarnos de que cumple cinco condiciones:

1. No hay orden de arriba-a-abajo en las filas
Esto quiere decir que cualquiera sea el orden en el cual se presentan los registros no significa un cambio en la información que la tabla contiene.

2. No hay orden de izquierda-a-derecha en las columnas.
Esto quiere decir que cualquiera sea el orden en el cual se presentan los campos entonces no significa un cambio en la información que la tabla contiene.

3. No hay filas duplicadas.
Esto tiene consecuencias importantes: Significa que siempre será posible definir una clave primaria.

4. Cada intersección de fila-y-columna contiene exactamente un valor del dominio aplicable (y nada más).
Esto implica que un campo de un registro debe tener un único valor: por ejemplo no podrá ser la cantidad de muchos artículos (pues serían muchos valores)

5. Todas las columnas son regulares [es decir, las filas no tienen componentes como IDs de fila, IDs de objeto, o timestamps ocultos]: Cada campo de cada registro no debe tener varias informaciones de varias entidades distintas pegadas. Por ejemplo no deberíamos tener el nombre de la calle, el número de la puerta, el piso y el departamento de una dirección en un único campo. (Si las calles fueran una entidad que preocupa a nuestro sistema)

Segunda forma normal:

Para estar en segunda forma normal una tabla tiene que empezar por cumplir la primera forma normal.

Además debe cumplir que el valor de cada campo depende sólo de la clave primaria completa y no de un subconjunto de campos que formen la clave primaria.

Quizá sea menos críptico con un ejemplo:

Imaginemos que para registrar las entradas y salidas de mis empleados en un sistema de control de horarios tuviera en una tabla:

IdEmpleado: Identificador único del empleado

Fecha: año, mes y día

Hora: Hora, minutos y segundos

IdMovimientoEmpleado: es un número que empieza para cada empleado en 1 cada día y se va incrementando.

Empleado: Nombre del Empleado

La clave primaria la podría armar con IdEmpleado, Fecha y IdMovimientoEmpleado

En este caso no estaría en 2FN pues el nombre del empleado ya que el nombre del empleado depende solamente del IdEmpleado.

Para llevarla a 2FN basta con eliminar la columna Empleado pues para obtener el nombre alcanza con seguir la relación a la tabla de Empleados usando el campo IdEmpleado.

Tercera forma normal:

Cada atributo (esto es el valor de un campo para un dado registro) debe depender sólo de la clave primaria.

Un ejemplo de esto sería si en una tabla tuviéramos:

IdJugador

Nombre del Jugador

Fecha de Nacimiento

Edad

Claramente la edad depende de la fecha de nacimiento.

Podría argüirse que no si cumple con 2FN pero falla en la tercera ya que, por supuesto, la edad depende unívocamente de la fecha de nacimiento.

Versión de Boyce y Code de la tercera forma normal:

Esta es la forma elegida por la mayoría de los diseñadores para las bases de datos que soportan las aplicaciones transaccionales. En muchos casos, simplemente cumpliendo esto ya cumplen con las formas superiores (salvo casos patológicos que discutiremos más abajo)

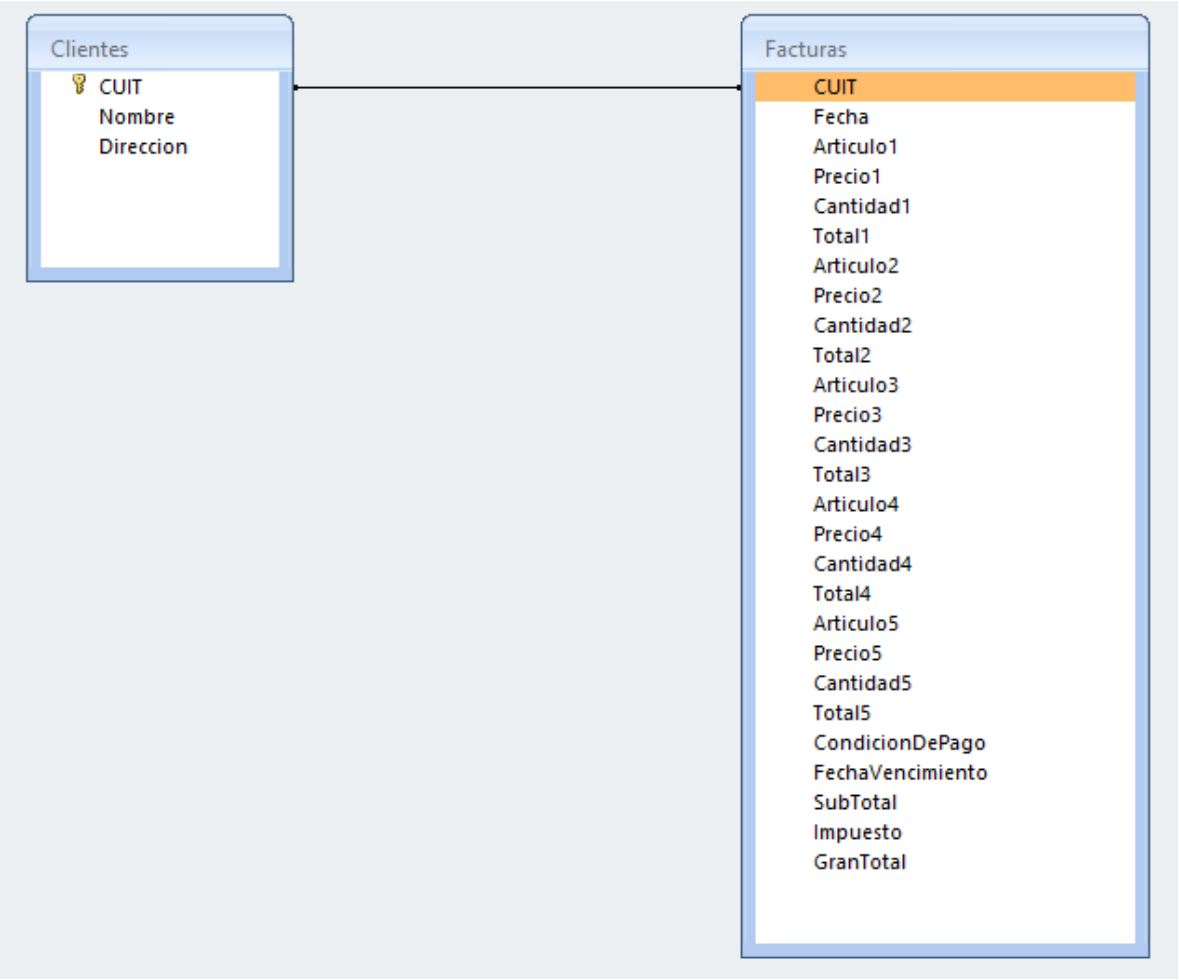
Para que una tabla cumpla con 3FNBC deberá valer que cada atributo depende exclusivamente de la clave, de la clave completa y de nada más que de la clave completa además de, por supuesto, cumplir con 1FN, 2FN y 3FN.

Cuarta forma normal

Para poder avanzar en la descripción de las formas normales superiores deberemos explorar un poco más el concepto de relación.

Una relación nos permitía ir a buscar un atributo que para poder cumplir con la 3FN queda guardado en otra tabla.

Tal fue lo que hicimos con los datos del cliente en nuestra tabla original de facturas:



Cuando estoy trabajando las facturas y necesito un dato del cliente, por ejemplo la dirección, para mandarle una nota de reclamo, entonces "camino" por la relación hacia el registro de ese cliente y encuentro la dirección.

Para este tipo de relación tenemos un único dato del lado del cliente que puede vincularse a las múltiples facturas que le hayamos emitido a ese cliente. Esto es lo que se llama una relación "1 a muchos"

Como se imaginan podríamos tener relaciones "1 a 1", "1 a muchos" o "muchos a muchos"

Las formas normales superiores se aplican a esas relaciones "muchos a muchos" o multi-variadas.

La primera consecuencia positiva de esto es que si una tabla cumple la 3FNBC y no tiene relaciones multivariadas entonces cumpla 4FN.

Veamos un ejemplo sencillo de una tabla con una relación multivariada:

Pizzería	Tipo de Pizza	Zona de Delivery
La Barra	Media masa	Necochea
Huggies	Piedra	Congreso
Huggies	Piedra	Microcentro
Huggies	Piedra	Barracas
Las Cuartetas	Piedra	Microcentro
Los Inmortales	Piedra	Microcentro
Banchero	Media masa	Caballito
Banchero	Piedra	Caballito
Banchero	Media masa	Flores
Banchero	Piedra	Flores
Guerrin	Masa completa	Microcentro

Asumimos que cada pizzería es consistente en el sentido de que entrega las mismas pizzas en todas sus áreas.

La clave primaria necesita ser la combinación de los tres campos ya que ninguno es función de los otros.

En este momento se recomienda la realización del ejercicio conceptual 2.9

¿Cuál es entonces el problema?

Si una pizzería agrega una variedad de pizza será necesario agregar varios registros, uno por barrio donde brinda delivery.

Si una pizzería agrega un nuevo barrio a su área de delivery entonces también habrá que agregar tantos registros como variedades maneje.

Ninguna de estas condiciones ayudará a conservar la integridad de los datos. Aquí es cuando la 4FN viene en nuestra ayuda.

Para que esta tabla cumpla con 4FN necesitamos reemplazarla por dos tablas unidas por una relación:

Pizzería	Tipo de Pizza
La Barra	Media masa
Huggies	Piedra
Las Cuartetas	Piedra
Los Inmortales	Piedra
Banchero	Piedra
Banchero	Media masa
Guerrin	Masa completa

Pizzería	Zona Delivery
La Barra	Necochea
Huggies	Congreso
Huggies	Microcentro
Huggies	Barracas
Las Cuartetas	Microcentro
Los Inmortales	Microcentro
Banchero	Caballito
Banchero	Flores
Guerrin	Microcentro

De esta manera tenemos desagregados dos datos que son independientes. Si una pizzería incorpora una nueva variedad se agrega un sólo registro. Si una pizzería se expande a un barrio se agrega un sólo registro.

Al cumplir la 4FN se ahorra espacio y se simplifican las actualizaciones.

En este punto se recomienda la realización del ejercicio conceptual 2.10

Quinta forma normal

En el caso anterior asumimos que las pizzerías eran gastronómicamente consistentes y producían los mismos tipos de pizza en todos los barrios.

Pongamos un ejemplo similar pero relacionado con la industria de la salud.

Para empezar tenemos médicos que realizan distintos tratamientos.

Luego tenemos coberturas de salud que financian distintos tratamientos.

Y, finalmente, no todos los médicos trabajan con todas las coberturas.

Si queremos reflejar eso en una única tabla podríamos poner:

Medico	Tratamiento	Asegurador
Perez	VMD	OSDE
Perez	VMD	SMG
Perez	VMD	OSDE
Perez	VMD	SMG
Rodriguez	VMD	SMG
Rodriguez	VMD	SMG
Sony	VMD	Galeno
Sony	VMD	Galeno

Nuevamente, cuando un médico empieza a desarrollar un nuevo tratamiento o un asegurador a reconocerlo hay que agregar varios registros tal y como nos pasaba con las tablas que no cumplían 4FN.

Para cumplir 5FN tenemos que desagregar en tres tablas cada una con sus relaciones:

Medico	Asegurador
Perez	OSDE
Perez	SMG
Sanchez	OSDE
Sanchez	Galeno
Rodriguez	SMG
Sony	Galeno

Tratamiento	Asegurador
VMD	OSDE
Clinica	OSDE
VMD	SMG
Clinica	Galeno

Medico	Tratamiento
Perez	VMD
Rodriguez	VMD
Sanchez	Clinico
Sony	Clinico
Sony	VMD

De esta manera cada tabla tiene información sobre un único tema y eso nos garantiza que el agregado o modificación de la información será más sencillo.

3. Modelado Lógico de una base de datos relacional

Podemos considerar tres visiones de una base de datos:

- Vista de los usuarios

- Vista de los diseñadores de los datos

- Vista de detalle

Los distintos usuarios de una base de datos podrán tener diferentes visiones por motivos de practicidad y seguridad.

Los usuarios podrán ver tablas que no existen dentro del modelo pero que se crean "al vuelo" para ellos, podrán no ver algunos campos o registros o tablas que si existan por razones de seguridad y tendrán posibilidad o no de escribir en distintos registros.

Los diseñadores de los datos verán un modelo lógico que les hablará de las relaciones y las entidades que ellos manejan.

La vista de detalle dará cuenta de cómo están físicamente guardados los datos y la estudiaremos en el modelado físico.

Entidades y relaciones

Antes que meternos con el modelo de entidad relación en sí mismo (DER) vamos a tratar de fijar los conceptos relacionados un poco más:

Entidad

Cada entidad representa una “cosa”, "objeto" o "concepto" del mundo real con existencia independiente, es decir, se diferencia claramente de otro objeto o cosa, incluso cuando son del mismo tipo.

Ejemplos:

Una persona. (Se diferencia de cualquier otra persona, incluso siendo gemelos). (Por ejemplo se diferencian por el DNI)

Un automóvil. (Aunque sean de la misma marca, el mismo modelo,..., tendrán atributos diferentes, por ejemplo, el número de chasis).

Una casa (Aunque sea exactamente igual a otra, aún se diferenciará en su dirección, número de catastro, etc.).

Una entidad no necesita ser un objeto con existencia física y tangible. También puede tratarse de un ente de razón como una empresa, una materia o un equipo.

Cada una de las ocurrencias de entidad queda descripta por sus atributos. Por ejemplo, la entidad Persona las características: Nombre, Apellido, Género, Estatura, Peso, Fecha de nacimiento describen la persona y son sus atributos.

Atributos

Son las distintas características que definen o identifican a una entidad. El diseñador solo usa las que considera relevantes para el problema que tiene entre manos.

En un conjunto de entidades del mismo tipo, cada entidad tiene valores específicos asignados para cada uno de sus atributos, de esta forma, es posible su identificación unívoca.

Ejemplos:

A la colección de entidades «alumnos del curso DBD», con el siguiente conjunto de atributos que tienen todos, (id, nombre, edad, versión de la cursada), pertenecen las entidades:

(1, Sophia, 35 años, 1)

(2, Josefa, 29 años, 5)

(3, Carlos, 40 años, 3)

...

Cada entidad perteneciente a este grupo se diferencia de las demás por sus atributos. Conviene subrayar que dos o más entidades diferentes pueden tener los mismos valores para algunos de sus atributos, pero nunca para todos. De lo contrario tendríamos registros idénticos y eso no sería 1NF

El problema particular que nos preocupa es la forma de identificar unívocamente una entidad de todas las entidades del mismo tipo. En el ejemplo que pusimos podría ser el Id del alumno (que suele ser un número que se asigna automáticamente en forma correlativa cuando se crea el registro) o también podría ser su DNI.

A estos atributos o combinaciones de atributos que podrían funcionar como identificadores únicos de la entidad (o sea como claves primarias) los llamamos clave alternativa.

hemos tenido que identificar a las facturas.

Nota Conceptual:

Clave alternativa: es un campo o conjunto de campos que identifica unívocamente a un registro y, por lo tanto, podría funcionar como una clave primaria.

Por supuesto tanto la clave primaria como las alternativas podrán ser un solo campo o un conjunto de ellos. Cuando hay varios campos involucrados lo llamamos clave compuesta por oposición a los que son de un solo campo que los llamamos clave simple.

Cada atributo podrá tomar ciertos y determinados valores. Por ejemplo una edad suele registrarse con enteros, un mes debe estar entre 1 y 12, un país debe ser una colección de caracteres, que, en realidad, no es cualquier combinación posible sino que responde a un conjunto mucho más restringido.

Este conjunto de valores posibles que puede tomar un campo o atributo es lo que se conoce como dominio. Al fijar el dominio queda claro qué tipo de datos se guardarán (números con punto decimal, números enteros, fechas, cadenas de caracteres, etc.)

Una cosa que resulta muy importante para el usuario de una base de datos es poder distinguir un valor faltante de un valor verdadero. En el caso de las bases de datos se usa el concepto de NULL para registrar un valor faltante o desconocido.

NULL nunca es igual a otro valor, ni siquiera a NULL. Si queremos preguntar si un campo es nulo no se puede usar la comparación como para un valor conocido. Ya iremos viendo en las diferentes implementaciones como se hace ese tipo de consulta.

Restricciones

Muchas veces las entidades y sus atributos deben cumplir reglas que tienen que ver con el negocio concreto que nuestra base de datos trata de reflejar.

Por ejemplo, un reglón de una factura debe pertenecer a una factura. Una factura debe tener por lo menos un renglón.

A veces esas restricciones se implementan a través de las relaciones. Por ejemplo, en el caso de las facturas el número de factura puede funcionar como clave primaria.

El número de factura se integrará a la clave primaria de la tabla de renglones de facturas. Por ejemplo la tabla de renglones de facturas podría tener como clave principal a una clave compuesta:

- Número de Factura
- Número de renglón

Cuando una tabla "toma prestada" para su clave principal la clave principal de otra tabla estamos ante una situación que se describe como "clave foránea"

Nota Conceptual:

Clave foránea: (en inglés: Foreign Key FK) es una limitación referencial entre dos tablas. La clave foránea identifica una columna o grupo de columnas en una tabla hija que se refiere a una columna o grupo de columnas en otra tabla (tabla maestra). Las columnas en la tabla referendo deben ser la clave primaria u otra clave candidata en la tabla referenciada.

En este punto se recomienda la realización del ejercicio conceptual 2.11

Cardinalidad de una relación:

Dada una relación en la que participan dos de entidades, la cardinalidad indica el número de entidades de una relación con las que puede estar relacionada una entidad dada.

Dado un conjunto de relaciones binarias y los conjuntos de entidades A y B, la correspondencia de cardinalidades puede ser:

Uno a Uno: (1:1) Un registro de una entidad A se relaciona con un único registro de la entidad B. (ejemplo dos entidades, profesor y departamento, con claves primarias, código_profesor y jefe_depto respectivamente, un profesor solo puede ser jefe de un departamento y un departamento solo puede tener un jefe).

Uno a Varios: (1:N) Un registro en una entidad en A se relaciona con cero, uno o muchos registros en una entidad B. Pero los registros de B solamente se relacionan con un único registro en A. (ejemplo: dos entidades, Facturas y Renglones, con claves primarias, número de factura y número de factura más número de renglón.).

Varios a Uno: (N:1) Funciona al revés que 1:N. Una entidad en A se relaciona exclusivamente con una entidad en B. Pero una entidad en B se puede relacionar con 0 o muchas entidades en A.

Varios a Varios: (N:M) Es un caso muy común. Una entidad en A se puede relacionar con 0 o con muchas entidades en B y viceversa (Una persona puede trabajar para varias empresas y para cada empresa trabajan muchas personas).

Ejemplos de relaciones que expresan cardinalidad:

Un policía (entidad) tiene (relación) un arma (entidad) siempre y cuando no realice funciones de oficina, pudiendo entonces tenerla o no asignada. Es una relación 0:1.

Cada esposo (entidad) está casado (relación) con una única esposa (entidad) y viceversa. Es una relación 1:1.

Una factura (entidad) se emite (relación) a una persona (entidad) y sólo una, pero una persona puede tener varias facturas emitidas a su nombre. Todas las facturas se emiten a nombre de alguien. Es una relación 1:N.

Un cliente (entidad) puede comprar (relación) varios servicios (entidad) y un servicio puede ser comprado por varios clientes distintos. Es una relación N:M.

Diagrama entidad-relación:

Ya hemos considerado tanto el concepto de entidad como el de relación. El diagrama de entidad-relación presenta ambos conceptos para las múltiples tablas que formarán nuestra base de datos en un único esquema.

Puede afirmarse desde una perspectiva formal que los diagramas ER son un lenguaje gráfico para describir conceptos. En la práctica son una forma de documentar y transferir información sobre la estructura de los datos que queremos registrar.

Normalmente las herramientas de software automatizan su creación. De hecho las ilustraciones con tablas y relaciones han sido elaboradas con la herramienta "Diagramas" de la versión 2007 de Microsoft Access.

Extensión del Diagrama de Entidad Relación (DER):

Para facilitar la transmisión de información se puede enriquecer la semántica del DER agregando algunos elementos a este lenguaje:

Entidades fuertes y débiles

Consideramos que una entidad es fuerte cuando puede ser unívocamente identificada por sus atributos. Por ejemplo, los alumnos del curso, que se identifican por su DNI hacen de los alumnos una entidad fuerte.

Una entidad es débil cuando, para construir su clave primaria, necesita atributos que vengan de otra. Los renglones de facturas, por ejemplo, necesitan del número de factura lo cual los hace una entidad débil.

Para reflejar esta diferencia a las entidades débiles se las suele reflejar mediante un rectángulo trazado con línea doble.

Se habla de dos tipos de entidades débiles

Se puede hablar de la existencia de 2 tipos de dependencias en las entidades débiles según sea necesario especificar la entidad fuerte relacionada:

Dependencia por existencia:

Las ocurrencias de la entidad débil pueden identificarse mediante un atributo identificador clave sin necesidad de identificar la entidad fuerte relacionada.

Dependencia por identidad

Cuando la entidad débil no puede ser identificada sin la entidad fuerte relacionada nos encontramos en el caso de dependencia por identidad. Por ejemplo si tenemos una entidad LIBRO y otra entidad relacionada EDICIÓN, para identificar una edición necesitamos conocer el identificador del libro.

4. Modelado Físico de una base de datos relacional

La implementación física del modelo lógico tiene como objetivo optimizar la performance de las respuestas de la base de datos.

Aquí tienen lugar consideraciones sobre el orden en el cual almacenar los datos, la creación de archivos auxiliares que aceleren las búsquedas (índices), etc.

Vamos a recorrer estas diferencias cuando veamos:

- Cómo y por qué crear índices
- Si la tabla debe guardarse en un orden u otro
- Si las relaciones conviene guardarlas con uno u otro esquema de relación
- Como los distintos tipos de datos ocupan la memoria y cómo conviene guardarlos para que sean buscados en forma eficiente
- Como afectan los índices a las operaciones de grabado (INSERT) modificación (UPDATE) y borrado (DELETE)

En este punto se recomienda realizar el ejercicio conceptual 2.12

5. Ejemplos de modelado a cargo del alumno

Se presentan varios casos para que los participantes construyan el modelo físico y el modelo lógico.

CASO 1

Se desea registrar la información relevante para la gestión de una empresa de transportes dedicada a repartir paquetes por todo el país. Los mismos conductores del vehículo son los encargados de llevar los paquetes.

Necesitamos guardar los datos del conductor de cada camión:

- dni
- nombre
- teléfono
- dirección
- salario
- ciudad de residencia

De los paquetes transportados interesa conocer:

- identificador único del paquete
- descripción
- destinatario
- dirección del destinatario
- remitente
- dirección del remitente

Un camionero distribuye muchos paquetes por día mientras que un paquete sólo puede ser distribuido por un único camionero.

De los camiones que conducen los camioneros, interesa conocer:

- la patente o matrícula
- modelo
- marca
- potencia del motor

Un camionero puede manejar diferentes camiones en fechas diferentes, y un camión puede ser conducido por varios camioneros a lo largo de su vida. Interesa poder saber que camionero estaba conduciendo que camión un día concreto.

CASO 2

Una empresa vende productos a varios clientes. Se quieren registrar los datos personales de los clientes (nombre, apellidos, dni, dirección y fecha de nacimiento). Los productos tienen nombre, código, y precio unitario.

Cada cliente puede comprar varios productos a la empresa, y por supuesto un mismo producto puede ser comprado por varios clientes.

Los productos vienen de diferentes proveedores. Una regla de negocio nos impone que cada producto sólo puede ser suministrado por un proveedor, y que un proveedor puede vendernos varios productos.

Vamos a registrar el CUIT de cada proveedor, su nombre y dirección.

CASO 3

Se quiere diseñar la base de datos de un Instituto de educación superior. Queremos registrar los datos de los profesores del Instituto:

- DNI
- nombre
- dirección
- teléfono

Los distintos profesores dictan módulos, y cada módulo tiene un código y un nombre.

Cada alumno está inscripto en uno o varios módulos.

Para cada alumno necesitamos guardar el número de su legajo, nombre, apellido y fecha de nacimiento.

Los profesores pueden impartir varios módulos, pero un módulo sólo puede ser impartido por un profesor durante una cursada.

Cada cursada tiene un grupo de alumnos.

CASO 4

Necesitamos diseñar una base de datos para registrar la información relevante para el sistema de gestión de un hospital.

De cada paciente se desea guardar:

- un código único
- nombre
- apellido
- dirección
- ciudad
- provincia
- código postal

- teléfono
- celular
- fecha de nacimiento
- financiador que cubre sus costos

De cada médico se desea guardar:

- un código único
- nombre
- apellido
- teléfono
- especialidades

Se desea llevar el control de cada uno de los ingresos y egresos que el paciente hace en el hospital.

Cada internación de un paciente queda registrada en la base de datos.

De cada internación se guardará

- un identificador único
- el identificador del paciente
- el número de habitación
- el número de cama
- la fecha de ingreso
- el identificador del médico que acepta la internación
- el diagnóstico preliminar

De cada alta de un paciente (egreso) queremos registrar:

- un identificador único del alta
- la fecha del alta
- el diagnóstico

- la identificador único del paciente
- el identificador único del médico que firma el alta

CASO 5

Nos piden diseñar una base de datos para registrar la información pertinente para un concesionario dedicado a la venta de automóviles.

La empresa tiene un conjunto de automóviles para su venta.

De cada auto necesitamos saber:

- matrícula
- marca
- modelo
- el color
- el precio de venta

Los datos que interesa conocer de cada cliente son:

- Identificador único que se autoincrementa
- DNI o CUIT
- nombre
- dirección
- ciudad
- número de teléfono

Un cliente puede comprar varios automóviles mientras que, un automóvil sólo puede venderse a un único cliente.

Una parte muy importante del negocio del concesionario son los servicios de Post Venta.

Cada servicio de mantenimiento tiene asociado un código único que se incrementa automáticamente por cada revisión que se haga.

De cada servicio se desea saber la fecha, que partes se cambiaron y cuál ha sido el kilometraje recorrido de manera de poder intentar predecir aproximadamente la fecha del siguiente servicio y hacerle una oferta directa que evite que vaya a hacerlo a otra concesionaria.

Bibliografía utilizada y sugerida

Date, C. J. An Introduction to Database Systems, Pearson Educación, Mexico, 2001

Elmasri/Navathe. Sistemas de Base de Datos. Conceptos Fundamentales Pearson-Addison - Wesley 2007

Korth, Henry F. Fundamentos de Bases de Datos. Mac Graw Hill 1992.

Teorey Toby J. - Database Modeling and Design. - Morgan Kaufmann 2005

Lo que vimos:

En esta Unidad vimos qué es una base de datos relacional, qué son las formas normales y cómo diseñar repositorios de datos.

Lo que viene:

En la próxima Unidad pasamos revista a los distintos objetos que viven dentro de una base de datos. También veremos cómo luce el código SQL para crear, destruir y usar esos objetos.