UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Diseño de base de datos para el Sistema Integral de Información de la Facultad de Ingeniería (SIIFI)

TESIS

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A

Omar Rubén Rangel Jiménez

DIRECTOR DE TESIS

Ing. Carlos Raúl Tlahuel Pérez

Ciudad Universitaria, Cd. Mx., 2017

Índice

Agradecimientos

A los dos pilares que siempre han estado sosteniendo mis triunfos y fracasos, de los que siempre me he apoyado para llegar cada vez más alto y que me alientan a ir más lejos cada vez. A esos pilares que son mis padres, que siempre han estado ahí para mí. Gracias por acompañarme en este camino que apenas comienza y en el que seguiremos juntos. Gracias Delia. Gracias Ray.

A mi hermana, Azucena, de la que sin necesidad de palabras he aprendido mucho.

A toda mi familia que siempre ha creído en mí y de la que sé puedo contar en cualquier momento y circunstancia.

A todos mis amigos que me acompañaron en la escuela preparatoria y en la universidad. Gracias por tantas risas y tanto apoyo. Espero con ansias que nuestros caminos se vuelvan a juntar.

Agradezco a todos esos profesores que me mostraron que camino era el que quería seguir y también a todos aquellos que me mostraron lo que no quería llegar a ser.

A mi director de tesis, Carlos, por toda la paciencia y dedicación para que este trabajo pudiese tomar forma.

A la UNAM y a la Facultad de Ingeniería por convertirse en mi eterno segundo hogar.

Gracias.

“Cuando la gente está de acuerdo conmigo siempre siento que debo estar equivocado.”

-Oscar Wilde

Preámbulo

Debido a la necesidad de contar con disponibilidad en la información en varios sectores de la sociedad, como lo puede ser una casa donde se tiene información personal guardada como lo son facturas de luz, agua, teléfono, o incluso categorizada por cada miembro de la familia, almacenada en un folder distinto, por ello es menester tener un método de organización de la misma. Otros ejemplos donde se almacenan datos en mayor cantidad son tiendas, universidades, bancos, etc. Como se puede notar, estas últimas contienen mayor número información y mucha más responsabilidad para resguardar la misma.

Se puede definir una base de datos (BD) como un *conjunto de datos persistente*[1], aclarando que no precisamente una BD sólo existe en un ambiente informático, demostrando esto con el ejemplo mencionado anteriormente de la información familiar resguardada en los hogares.

Aquel sistema computacional que se encarga de mantener información almacenada a través de un conjunto de programas que nos permiten tener acceso a ésta, recibe el nombre de Sistema Gestor de Bases de Datos (SGBD), el cual se apoya de los datos almacenados para poder ser mostrados en la forma que el usuario los necesita.

El SGBD ayuda a mantener grandes cantidades de información almacenada con estructuras previamente programadas por el administrador del sistema. Al contar con un orden estipulado, el realizar un proceso de búsqueda dentro de todos los datos almacenados será tarea de la computadora, facilitando así el trabajo humano, ya que el usuario únicamente tendrá que concretar los criterios de la consulta.

Imaginando que se cuenta con una base de datos de una universidad en la que están inscritos cinco mil alumnos, de estos mismos se desea saber la cantidad de estudiantes de la carrera de *Ingeniería Industrial* que cursan el cuarto semestre y que tienen un promedio mayor a ocho. Como se puede observar, existen tres consultas sobre la información. La primera es separar a los alumnos que cursan la carrera de *Ingeniería Industrial*, la segunda extraer a los que están en cuarto semestre y por último cuántos de ellos tienen un promedio mayor a ocho. ¿Se puede observar la necesidad de contar con una estructura en la base de datos para que la búsqueda de información sea sencilla y ordenada? De no ser así, sería una tarea tediosa y cansada.

Una vez establecida la importancia que tiene una base de datos estructurada dentro de cualquier organización que se vea obligada a trabajar con información de un número considerable de personas y otros aspectos de importancia en el sector, se procede a estudiar más particularmente el mismo caso aplicado a la institución universitaria y al propósito de este trabajo.

La Facultad de Ingeniería cuenta actualmente con un sistema diseñado hace ya varios años donde se tenían ciertas necesidades apegadas a su tiempo así como equipos que cubrían los requerimientos que entonces se presentaron. Hoy en día, las necesidades son distintas así como lo es también la tecnología con que se cuenta por lo que, en conjunto con la Unidad de Servicios de Cómputo Administrativos (USECAD), se planea crear un nuevo diseño que cumpla las necesidades actuales aprovechando la tecnología con que se cuenta. Para este nuevo diseño de sistema se necesita un nuevo diseño de base de

datos normalizada que permita trabajar con la información evitando a medida de lo posible la redundancia de la misma.

Se planea que los sistemas a cargo de la Secretaría de Servicios Administrativos (SSA) se conjunten en uno solo denominado Sistema Integral de Información de la Facultad de Ingeniería (SIIFI) cuya base de datos fungirá como el núcleo de la información general que los sistemas comparten.

El núcleo de información será una base de datos que permitirá a la comunidad universitaria, de la Facultad de Ingeniería, llevar a cabo ciertos procesos que conciernen a la institución así como también la consulta de dichos datos, ya sea en forma general o en forma de resumen, que permitan trazar una estadística deseada. Todo esto se consigue relacionando ciertos campos con los que la base de datos debe contar, como lo son la información de alumnos, carreras, divisiones, departamentos, académicos, salones, horarios, por mencionar algunos.

Este nuevo diseño de base de datos evitará la redundancia e incosistencias en los datos tal y como lo definen las formas normales

La importancia de una base de datos normalizada es crucial para el funcionamiento de un sistema, ya que dependiendo de la forma en que la serie de datos se encuentre estructurada puede influir de forma drástica en la eficiencia del mismo sistema.

La USECAD desarrollará este sistema partiendo de la base de datos diseñada a lo largo de este trabajo.

Definición del Sistema Integral de Información de la Facultad de Ingeniería (SIIFI)

El SIIFI es un sistema encargado de proporcionar información en forma general o en forma de resúmenes del registro escolar de la Facultad de Ingeniería, como lo pueden ser *alumnos, académicos, asignaturas, grupos, horarios, salones, carreras, divisiones* y *departamentos*.

El SIIFI es un sistema clave para la Facultad de Ingeniería, ya que su base de datos es explotada por otros sistemas, por lo que es ideal partir del rediseño de la base de datos para el SIIFI.

En la imagen A se puede ver un diagrama ejemplifica lo que es SIIFI, siendo su base de datos el centro de abastecimiento para diversos sistemas que toman la información que necesitan. Siendo así SIIFI un conjunto de sistemas.

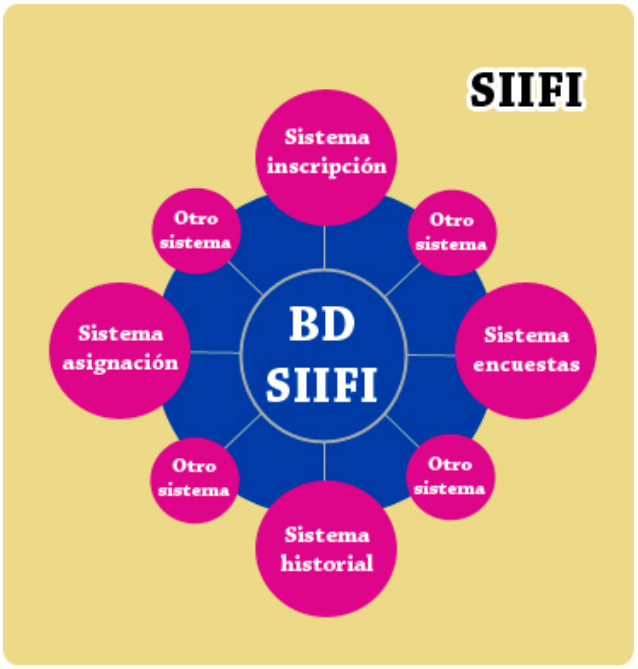


Imagen A SIIFI

Objetivo

Rediseñar una base de datos para el Sistema Integral de Información de la Facultad de Ingeniería cumpliendo con todos los requerimientos que presenta dicho sistema.

El diseño de la base de datos deberá ser normalizado para que evitar la redundancia según las formas normales.

Las tablas creadas tienen que tener todos los datos necesarios para ser identificadas de forma única de las demás, es decir tener por lo menos una llave candidata, así como también las relaciones entre ellas deberán ser las especificadas en los requerimientos.

El procedimiento a seguir será realizar el documento con las especificaciones necesarias a cumplir seguido de un diagrama entidad relación que nos servirá como un mapa para llevar a cabo la codificación de la misma.

Se estudiarán los caminos posibles para así poder elegir el más adecuado con respecto a las necesidades del sistema y poder realizar un diseño nuevo que se adapte a dichas necesidades.

Para resumir el objetivo de este trabajo en un enunciado, se dirá que el objetivo es "*aprovechar la tecnología con la que cuenta la Facultad de Ingeniería para realizar el diseño de la base de datos que utilizará el Sistema Integral de Información de la Facultad de Ingeniería*".

CAPÍTULO I
Introducción

1.1 Definición de una base de datos

Para poder empezar a tratar este tema de una forma ordenada, lo primero que se tiene que hacer es definir el tópico central, ilustrándolo con algunos ejemplos del ambiente en que se desarrolla.

Una base de datos se define como un conjunto de datos persistentes, a los cuales se puede acceder de diversas formas; se puede tener acceso a ellos mediante un Sistema Gestor de Base de Datos (SGBD), que es aquél *software* que se encarga de la interacción *cliente-servidor* de la información manejada. Se dice que los datos de la BD "persisten" debido en primer lugar que una vez aceptados por el SGBD para entrar en la base de datos, en lo sucesivo sólo pueden ser removidos por alguna solicitud explícita al SGBD, no como un mero efecto lateral de (por ejemplo) algún programa que termina su ejecución[1].

Hoy en día las bases de datos son usadas en varios entornos y se han vuelto una parte esencial en el manejo de la información, entre las aplicaciones más representativas de las bases de datos se tiene:

*El negocio bancario. Donde se almacena la información básica del cliente como también el manejo de sus cuentas.

*Líneas aéreas. En las cuales se pueden tener acciones de planificación como lo son reservaciones de vuelos por parte de los clientes, asignación de aviones a cada vuelo, establecer pistas de aterrizaje para cada avión, ordenar el horario de los vuelos, etc.

*Instituciones educativas. Llevar el control de alumnos, profesores y asignaturas, así como la relación que existe entre ellos.

*Telecomunicaciones. Almacenar el registro de llamadas realizadas en determinado mes y así poder facturar la cantidad total a pagar.

*Finanzas. Para guardar información de grandes empresas, venta y compra de documentos formales y financieros, como bolsa y bonos.

Así como estos ejemplos existen muchos sectores en donde las bases de datos son necesarias para tener un orden de almacenamiento de la información proporcionando una forma más sencilla y ordenada de consultarla.

Las bases de datos, según la variabilidad de los datos, pueden ser clasificadas en dos tipos: OLAP y OLTP.

Las bases de datos OLAP (On-Line Analytical Processing) son aquellas en donde los datos almacenados servirán en su mayoría para ser leídos ya que almacenan un histórico de la información la cual no cambiará a pesar del transcurso del tiempo. Un ejemplo de este

tipo de bases de datos es un registro de periódicos, ya que se guardará la información de que día fue publicado, que número de edición pertenece el ejemplar, las noticias que contiene, los autores de dichas noticias, así como otra información que pueda ser relevante para este fin. Como se puede observar esta información no cambiará con el tiempo, debido a que es información que ya pasó, no se debería cambiar un periódico que fue publicado tiempo atrás[2].

Por otra parte, las bases de datos OLTP (On-Line Transactional Processing), como su nombre lo indica, están orientadas al procesamiento de transacciones, que a causa de su propósito se encuentra en la necesidad de modificarse con el tiempo, sufriendo procesos de alta, baja y actualización. Como ejemplo podemos poner la información de los productos que ofrece un supermercado, que almacenará el nombre del producto, la marca, la cantidad que contiene y el precio. Al ser consumidores se está al tanto de la variación constante de los precios en los productos que se consumen, siendo una necesidad primordial para la información contar con la flexibilidad adecuada para que estos datos puedan ser modificados[2].

Las principales diferencias existentes con los sistemas OLTP y OLAP pueden ser observadas en la siguiente tabla:

Tabla 1.1 OLAP y OLTP

CARACTERÍSTICAS	OLTP	OLAP
Origen de datos	Interno	Interno y externo
Actualización	Actual	Histórico
Consultas	Predecible	Especializado
Actividad	Operacional	Analítica

En resumen, la diferencia entre una base de datos de tipo OLAP y una OLTP es la cantidad y el tipo de operaciones que realiza cada una por segundo. En el caso de las bases de datos OLTP se realizan muchas transacciones pequeñas mientras que en OLAP el número de transacciones es menor, pero las que se realizan son grandes transacciones.

1.2 La historia de las bases de datos

El término bases de datos fue escuchado por primera vez en un simposio celebrado en California en 1963.

Los orígenes de las bases de datos[3] se remontan a la antigüedad donde ya existían bibliotecas y toda clase de registros. Además también se utilizaban para recolectar información sobre las cosechas y censos de ciertas civilizaciones. Sin embargo, su búsqueda era lenta y poco eficaz y no se contaba con la ayuda de máquinas que pudiesen reemplazar el trabajo manual.

Posteriormente, el uso de las bases de datos se desarrolló a partir de las necesidades de almacenar grandes cantidades de información o datos. Sobre todo, desde la aparición de las primeras computadoras.

En 1884 *Herman Hollerith* creó la máquina automática de tarjetas perforadas, siendo nombrado así el primer ingeniero estadístico de la historia. En esta época, los censos se realizaban de forma manual. Ante esta situación, *Hollerith* comenzó a trabajar en el diseño de una maquina tabuladora, basada en tarjetas perforadas.

Posteriormente, en la década de los cincuenta se da origen a las cintas magnéticas, para automatizar la información y hacer respaldos. Esto sirvió para suplir las necesidades de información de las nuevas industrias. Y a través de este mecanismo se empezaron a automatizar información, con la desventaja de que solo se podía hacer de forma secuencial.

Posteriormente en la época de los sesenta, las computadoras bajaron los precios para que las compañías privadas las pudiesen adquirir; dando paso a que se popularizara el uso de los discos, cosa que fue un gran adelanto en la época, debido a que a partir de este soporte la información se guardaban continuamente, sin fragmentar.

En esta misma época se dio inicio a las primeras generaciones de bases de datos de red y las bases de datos jerárquicas, ya que era posible guardar estructuras de datos en listas y árboles.

Otro de los principales logros de los años sesenta fue la alianza de *IBM* y *American Airlines* para desarrollar SABRE (*Semi-automated Business Research Environment)*, un sistema operativo que manejaba las reservas de vuelos, transacciones e informaciones sobre los pasajeros de la compañía American Airlines.

Posteriormente, en esta misma década, se llevó a cabo el desarrollo del *IDS (Integrated Data Store)* desarrollado por *Charles Bachman* (que formaba parte de la *CODASYL*[4]) supuso la creación de un nuevo tipo de sistema de bases de datos conocido como modelo en red que permitió la creación de un estándar en los sistemas de bases de datos gracias a la creación de nuevos lenguajes de sistemas de información.

Los miembros de *CODASYL* pertenecían a industrias e instituciones gubernamentales relacionadas con el proceso de datos, cuya principal meta era promover un análisis, diseño e implementación de los sistemas de datos más efectivos; y aunque trabajaron en varios lenguajes de programación como *COBOL* (*Common Business-Oriented Language)*, nunca llegaron a establecer un estándar fijo, proceso que se llevó a cabo por *ANSI (American National Standards Institute)*.

Para la década de los 70s el científico informático *Edgar Frank Codd*, de la empresa *IBM*, introduce la idea de un modelo relacional de bases de datos en un documento titulado *"A Relational Model of data for Large Shared Banks"*[5], para el descontento de *Codd*, *IBM* no se apresura a explotar sus ideas para ser llevadas a la práctica sino hasta que sus empresas rivales como lo fue *Larry Ellison* que en base a las ideas de *Codd* diseña la base de datos de *Oracle*.

Este hecho dio paso al nacimiento de la segunda generación de los Sistemas Gestores de Bases de Datos.

Posteriormente en la época de los ochenta se desarrolló el lenguaje *SQL (Structured Query Language)* o lo que es lo mismo un lenguaje de consultas o lenguaje declarativo de acceso a bases de datos relacionales que permite efectuar consultas en base a operaciones de álgebra relacional con el fin de recuperar información de interés de una base de datos y hacer cambios sobre la base de datos de forma sencilla; además de analizar

grandes cantidades de información y permitir especificar diversos tipos de operaciones frente a la misma información, a diferencia de las bases de datos de los años ochenta que se diseñaron para aplicaciones de procesamiento de transacciones.

Por su parte, a principios de los años ochenta comenzó el auge de la comercialización de los sistemas relacionales, y *SQL* comienza a ser el estándar de la industria, ya que las bases de datos relacionales con su sistema de tablas pudieron competir con las bases jerárquicas y de red, como consecuencia de que su nivel de programación era sencillo y su nivel de programación era relativamente bajo.

En la década de los noventa la investigación en bases de datos giró en torno a las bases de datos orientadas a objetos. Las cuales han tenido bastante éxito a la hora de gestionar datos complejos en los campos donde las bases de datos relacionales no han podido desarrollarse de forma eficiente.

Así se creó la tercera generación de Sistemas Gestores de Bases de Datos.

Fue también en esta época cuando se empezó a modificar la primera publicación hecha por *ANSI* del lenguaje *SQL* y se comenzó a agregar nuevas expresiones regulares, *consultas recursivas, triggers* y algunas características orientadas a objetos, que posteriormente en el siglo XXI volverá a sufrir modificaciones introduciendo características de *XML*, *cambios en sus funciones, estandarización del objeto sequence* y de las *columnas autonuméricas*. Y además, se crea la posibilidad de que *SQL* se pueda utilizar conjuntamente con *XML*, y se define las maneras de cómo importar y guardar datos *XML* en una base de datos *SQL*. Dando así, la posibilidad de proporcionar facilidades que permiten a las aplicaciones integrar el uso de *XQuery* (lenguaje de consulta *XML*) para acceso concurrente a datos ordinarios *SQL* y documentos *XML*. Posteriormente, se da la posibilidad de usar la cláusula *order by*.

Aunque el boom de la década de los noventa es el nacimiento de la *World Wide Web* a finales de la década, ya que a través de este se facilitará la consulta a bases de datos.

En la actualidad, las tres grandes compañías que dominan el mercado de las bases de datos son IBM, Microsoft y Oracle. Por su parte, en el campo de internet, la compañía que genera gran cantidad de información es Google.

La evolución generacional de las bases de datos se puede representar como se ve en la imagen 1.1

Imagen 1.1 Evolución de las bases de datos

1.2.1 La historia de los dispositivos de almacenamiento

La historia del almacenamiento de los datos va de la mano con la evolución de los dispositivos de almacenamiento, estos dispositivos realizan las operaciones de lectura y/o escritura de los medios o soportes donde se almacenan o guardan, lógica y físicamente, los archivos de un sistema informático.

Los dispositivos de almacenamiento han ido evolucionando a lo largo de las últimas seis décadas, y una muestra de este avance tecnológico es que en lugar de almacenar la información en unidades flash, como se hace ahora. Antes esta información se almacenaba en pilas de tarjetas perforadas.

En la década de los 50s aparecieron las tarjetas perforadas las cuales tenían la capacidad de almacenar hasta *960 bytes*, siendo representadas por el sistema binario donde una perforación significaba un cero y un espacio no perforado un uno. Para ejemplificar la capacidad de estos dispositivos, se puede decir que para almacenar un archivo *MP3* de aproximadamente 2 minutos se tendrían que haber ocupado 40 000 tarjetas perforadas.

El dispositivo que sirvió para remplazar las tarjetas perforadas, fue la cinta magnética, donde una sola bobina de este tipo podía almacenar una capacidad de *5* a *10 Megabytes*, pero con el inconveniente de su tamaño en sus inicios.

Con la necesidad de tener un dispositivo que pudiera ser transportado con mayor facilidad de un ordenador a otro, surge el disquete de 5.25'', el cual podía almacenar hasta *1.2 Megabytes*.

Después surge el disquete de 3.5'', en los cuales se podía almacenar mayor información que su predecesor con capacidad de *1.44 Megabytes*.

Uno de los inconvenientes de los disquetes era su durabilidad, ya que eran demasiado sensibles. Aquí es cuando surge el Disco Compacto, o CD por sus siglas en inglés, estos dispositivos surgen en la época de los 90s teniendo una capacidad de almacenaje de hasta *650 Megabytes*, donde la durabilidad de este dispositivo fue mucho mayor. Como prueba, estos dispositivos aún son usados con frecuencia en nuestros días.

Uno de los dispositivos más usados hoy en día para transportar información de un lugar a otro son las unidades flash, como lo pueden ser las memorias USB o las unidades de estado sólido, entre algunas otras. Las cuales cuentan con una capacidad de hasta *256 Gigabytes*.

Pero de los dispositivos más utilizados para almacenar la información dentro de un ordenador son los discos duros que actualmente pueden almacenar hasta *4 Terabytes* de información. Es importante mencionar que debido a su tamaño, las unidades de estado sólido o *SSD* por sus siglas en inglés ya son usadas en ordenadores portátiles con la desventaja de que hoy en día su costo es elevado.

Por último una tendencia que ha surgido en los últimos años para el usuario, es el almacenaje en la nube, existiendo diversas empresas que rentan sus servidores para poder almacenar la información sin necesidad de tener un dispositivo de almacenamiento físico a la mano.

En la imagen 1.2 podemos observar los dispositivos de almacenamiento mencionados anteriormente.

Imagen 1.2 Dispositivos de almacenamiento

1.3 Modelos de bases de datos

A través de los años y en base a las necesidades que se fueron presentando, los modelos de bases de datos fueron evolucionando para cumplir con la demanda que la organización de datos requería, todo esto ayudándose de la tecnología que a su vez se fue desarrollando. En el siguiente apartado se verá más a detalle los distintos modelos de bases de datos.

1.3.1 Modelo jerárquico

Este modelo de base de datos tiene como objetivo establecer una jerarquía en la estructura de los campos, de manera que cada campo puede contener una lista con más campos.

Se define como una colección de segmentos (registros) que se conectan entre sí por medio de enlaces, cada segmento es una colección de campos (atributos) que contienen un solo valor cada uno de ellos. Un enlace es una asociación o unión entre dos segmentos exclusivamente[6].

Otra forma de ver este modelo es mediante nodos, existiendo nodos padres y nodos hijos, pudiendo ser el nodo hijo un nodo padre a su vez y viceversa. De esta forma, al nodo que se halla a la cabeza de un registro es llamado nodo padre, y a los nodos que dependen de él es llamado nodo hijo. A continuación, se muestra la imagen 1.3 que puede ejemplificar de mejor forma la estructura de un modelo jerárquico.

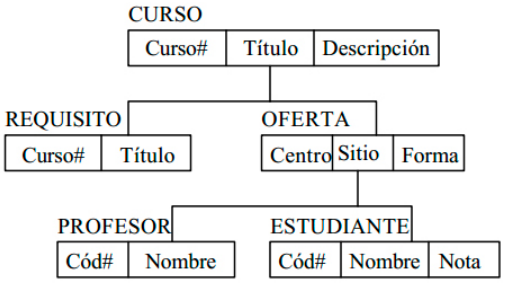


Imagen 1.3 Modelo jerárquico

Como se puede ver en la imagen 1.3, se tienen diversos nodos padres y nodos hijos. Por ejemplo, teniendo la mayor jerarquía del esquema el nodo *CURSO* siendo el nodo padre o nodo raíz de los nodos *REQUISITO* y *OFERTA*, estos últimos siendo nodos hijos, pero a su vez el nodo OFERTA también es nodo padre de los nodos *PROFESOR* y *ESTUDIANTE*.

Las características principales de implementar este modelo son:

-Globalización de la información: permite a los diferentes usuarios considerar la información como un recurso corporativo que carece de dueños específicos.

-Eliminación de información inconsistente: si existen dos o más archivos con la misma información, los cambios que se hagan a estos deberán ser hechos en todas las copias de dicho archivo.

-Permite compartir información: varios sistemas o usuarios pueden utilizar un mismo segmento.

-Permite mantener la integridad en la información: la integridad de la información es una de las cualidades altamente deseable y tiene por objetivo que sólo se almacena la información correcta.

-Independencia de datos: implica un divorcio entre programas y datos; es decir, se pueden hacer cambios a la información que contiene la base de datos o tener acceso a la base de datos de diferente manera, sin hace cambios en las aplicaciones o en los programas.

Algunas de las limitaciones que tiene el modelo jerárquico en las bases de datos son las siguientes:

-Al borrar un nodo padre, desaparecerán también sus nodos hijos.

-Sólo se podrá añadir un nodo hijo, si previamente ya existe un nodo padre.

-Tiene una estructura muy rígida la cual no permite relación entre nodos hijos de distintos nodos padres.

1.3.2 Modelo en red

Para fines prácticos, el modelo en red se puede ver como el intermedio entre el modelo jerárquico y el modelo relacional, ya que su estructura es parecida a la jerárquica con un nivel más complejo, con lo que consigue evitar algunas limitaciones mostradas en el modelo anterior.

Una base de datos de red está formada por una colección de registros, los cuales están conectados entre sí por medio de enlaces. El registro es un objeto único de datos implícitamente estructurado en una tabla, mientras que el enlace es la asociación entre dos registros exclusivamente. Una estructura de datos en red abarca más que la estructura de árbol, ya que un nodo hijo en el modelo en red puede tener más de un padre.

El modelo de red organiza dos fundamentales construcciones: registros y conjuntos. Los registros contienen campos y los conjuntos definen de una a varias relaciones entre los registros. Un campo se define como *un registro de base de datos reservado para almacenar un determinado tipo de información*[7].

Este modelo está basado en representar los registros de la base de datos por medio de ligas, existen relaciones en las que sólo participan dos entidades, a éstas se les llaman binarias; aquellas en las que participan en más de dos relaciones reciben el nombre de generales.

Una de las mejoras con las que cuenta el modelo de red respecto al jerárquico se puede mencionar que el de red puede tener una interrelación múltiple entre los nodos que componen al modelo.

El modelo de red y su interacción entre múltiples nodos se puede observar en la imagen 1.4, donde se ve que un cliente puede pedir una orden, pero a su vez el gerente atiende la misma, siendo el vendedor el que la surte a través de su relación con productos.

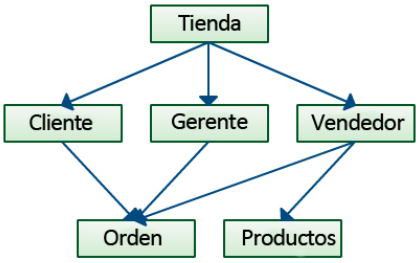


Imagen 1.4 Modelo de red

El modelo de red es una variación sobre el modelo jerárquico, al grado que es construido sobre el concepto de múltiples ramas (estructuras de nivel inferior) emanando en uno o varios nodos (estructuras de nivel superior). El modelo de red es capaz de representar la redundancia en datos de una manera más eficiente que en el modelo jerárquico[8].

1.3.3 Modelo relacional

Este modelo de bases de datos consiste en un conjunto de tablas, en donde cada una de ellas tiene un nombre exclusivo que la diferencia de las demás. Las tablas se conforman por filas y columnas, cada fila representa una relación entre un conjunto dado de valores[9]. Una relación en bases de datos se puede definir como un vínculo entre dos o más entidades y la interrelación que existe entre ellas[1].

En el modelo relacional se trabajará una entidad que está conformada por atributos, los cuales son representados como columnas que se definen como una propiedad de interés para dicha entidad. Al mencionar entidad, se refiere a la representación de un objeto o concepto del mundo real que se describe en una base de datos. Para cada atributo hay un conjunto de valores permitidos, llamado dominio del atributo. A continuación se mostrará la tabla 1.2 donde se ejemplifica mejor estos conceptos.

Tabla 1.2 Estudiante

Nombre	Número de Cuenta	Teléfono	Semestre	Promedio
Juan López	309175293	58231456	7	9.35
Pedro Martínez	304854569	21589636	9	8.25
Carmen Beltrán	308852456	14568989	5	9.99
Carlos Morales	302145825	55213232	8	6.23
José Pérez	307125469	54963147	6	7.25
María Márquez	305147853	52782963	6	8.96
Sandra Sánchez	305112849	15855485	7	8.33
Jorge Gutiérrez	301255845	22153636	8	9.25

En la tabla 1.2 se puede notar que los atributos de *ESTUDIANTE* son *nombre*, *número de cuenta*, *teléfono*, *semestre* y *promedio*. De la misma forma se nota que el dominio del atributo nombre son todos los nombres que aparecen en la tabla.

Suponiendo que *D1*da el conjunto de todos los nombres, *D2* el conjunto de todos los números de cuenta, *D3* el conjunto de los teléfonos, *D4* el conjunto de los semestres y *D5* el conjunto de los promedios, mientras tanto las filas se pueden definir como tuplas (secuencia ordenada de datos[1]) *v1*, *v2*, *v3*, *v4*, *v5* donde *v1* es un nombre que está en el dominio *D1*, *v2* un número de cuenta que está en el dominio *D2* y así sucesivamente. En general, una tabla de n atributos debe ser un subconjunto de

$$D1 \times D2 \times \ldots \times Dn\text{-}1 \times Dn$$

Se puede notar que en base de datos se utilizan los términos relación y tupla en lugar de fila y columna.

Otros conceptos importantes en el modelo relacional son los de superclave, clave candidata y clave primaria. Una superclave *es un conjunto de uno o más atributos que, tomados colectivamente permiten identificar de forma única una entidad*[10], pero el concepto de una superclave no es suficiente para identificar una tabla ya que puede contener atributos innecesarios, por lo que se propone tener una superclave mínima que permita la identificación de una entidad, a estas claves se les llama clave candidata. Así mismo se

utilizará el concepto de clave primaria para denotar una clave candidata que es elegida por el diseñador de la base de datos como el elemento principal para identificar las entidades dentro de un conjunto de las mismas. La clave primaria se debería elegir de manera que sus atributos nunca, o muy raramente, cambien.

Con todos los conceptos antes mencionados ya se puede dar una definición acertada de lo que es el modelo relacional en el ambiente de las bases de datos. Una base de datos relacional se define como *una base de datos en donde todos los datos visibles al usuario están organizados estrictamente como tablas de valores que se componen de atributos, tuplas y relaciones, y en donde todas las operaciones de la base de datos operan sobre estas tablas*[11].

Entre unas de las ventajas que presenta este modelo sobre sus predecesores es que provee herramientas que garantiza evitar la duplicidad de registros, también garantiza la integridad referencial, esto quiere decir que al eliminar un registro elimina de manera automática todos los registros relacionados dependientes; otra ventaja importante favorece a la normalización por ser comprensible y aplicable. La normalización se refiere a un proceso que consiste en aplicar una serie de reglas a las relaciones obtenidas tras el paso del diagrama entidad-relación al modelo relacional.

Entre las limitantes de este modelo está la deficiencia de trabajar con datos de tipo gráficos, multimedia, *CAD (Diseño Asistido por Computadora)* y sistemas de información geográfica. Tampoco se puede manipular de forma manejable los bloques de texto como tipo de dato.

1.3.4 Modelo orientado a objetos

Las bases de datos orientadas a objetos se propusieron, entre uno de sus objetivos, satisfacer las necesidades que no cumple el modelo relacional, y así complementar y no sustituir a las anteriormente mencionadas.

Estas bases de datos surgen más recientemente debido a la evolución de la programación orientada a objetos, en la cual se permite la creación de programas más grandes y complejos, tratando los problemas desde un punto de vista realista, y modelando cada uno de ellos como si se tratara de un conjunto de elementos u objetos que interrelacionan entre sí para solucionar el problema.

Algunos de los conceptos básicos que se tienen que saber para poder entender el modelo orientado a objetos son el de clase, estado, encapsulación, mensaje y herencia.

Una clase se define como un *conjunto de objetos agrupados. Cada objeto debe de pertenecer a una clase que definirá sus características generales*. El estado *son las características propias de cada objeto.* La encapsulación *es un mecanismo que consiste en organizar datos y métodos de una estructura, conciliando el modo en que el objeto se implementa, es decir, evitando el acceso a datos por cualquier otro medio distinto a los especificados*; la encapsulación garantiza la integridad de los datos que contiene un objeto. El mensaje *es aquel estímulo que se le envía a un objeto*. La herencia *establece toda una jerarquía de tipos o datos*[12].

Para poder utilizar este modelo, se sugiere que la programación ocupada sea del tipo orientada a objetos ya que la información representada en este modelo será mediante objetos, por lo que se necesita que la programación utilizada sea del mismo tipo. Al integrar las características de una base de datos con las de un lenguaje con programación orientado a objetos el resultado será un *Sistema Gestor de Bases de Datos Orientado a Objetos*, el cual hará que los objetos de la base de datos aparezcan como objetos de un lenguaje de programación en uno o más lenguajes de programación a los que dé soporte.

En la imagen 1.5 mostramos un ejemplo de modelo orientado a objetos donde se presentan los conceptos mostrados anteriormente.

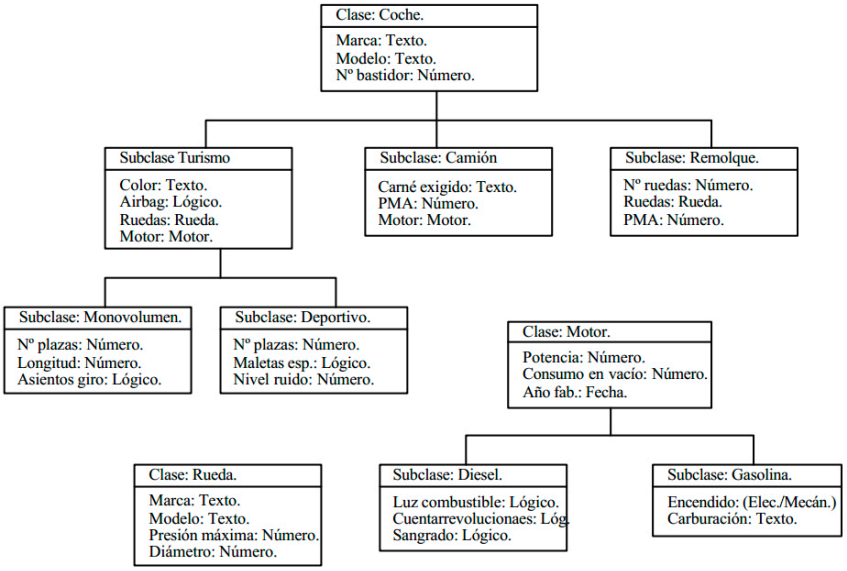


Imagen 1.5 Modelo orientado a objetos

En la orientación a objetos, un sistema se concibe como un *conjunto de objetos que se comunican entre sí mediante mensajes*[13]. Tradicionalmente los datos y procedimientos se almacenan por separado, los datos y sus relaciones en la base de datos, y los procedimientos en los programas de aplicación; la orientación a objetos, sin embargo, combina procedimientos de una entidad con sus datos, es decir, el comportamiento es parte de la entidad en sí, por lo que en cualquier lugar en que se utilice, se comporta de un modo predecible y conocido.

1.3.5 Diagrama entidad relación

En el diagrama entidad relación, se busca hacer una percepción del mundo real consistente en objetos básicos llamados entidades y de relaciones. Una entidad se define como *una cosa u objeto en el mundo real que se puede distinguir de todos los demás objetos*[14], una entidad tiene un conjunto de propiedades y los valores para un conjunto de propiedades pueden identificar una entidad de forma unívoca.

Este diagrama sirve como un mapa para construir una base de datos que tiene como modelo el relacional.

Cuando existe *un conjunto de entidades del mismo tipo que comparten las mismas propiedades o atributos, se le llama* conjunto de entidades[15]. Los conjuntos de entidades no son necesariamente disjuntos.

Una entidad es representada por un conjunto de atributos, los que se encargan de describir las propiedades que posee cada miembro de un conjunto de entidades. Para cada atributo hay un *conjunto de valores permitidos*[15] a los cuales se les conoce con el nombre de dominio. Formalmente, un atributo de un conjunto de entidades *es una función que asigna al conjunto de entidades un dominio*[15], como un conjunto de entidades puede contener atributos diferentes, cada entidad puede ser descrita como un conjunto de pares (atributo, valor).

Un atributo se puede categorizar por los siguientes tipos[22]:

-Simples y compuestos. Los atributos simples no están compuestos en sub-partes, en cambio, los compuestos pueden dividirse en otros atributos, como por ejemplo el nombre de una persona se podría dividir en nombre, apellido materno y apellido paterno.

Imagen 1.6 Atributos simples y compuestos

-Monovalorados y multivalorados. Los monovalorados tienen la característica de que sólo pueden contener un valor, mientras que los multivalorados como su nombre lo dice puede tener múltiples valores, como por ejemplo una persona puede tener más de un número telefónico. Cuando sea necesario se pueden establecer límites superiores e inferiores en el número de valores de un atributo multivalorado.

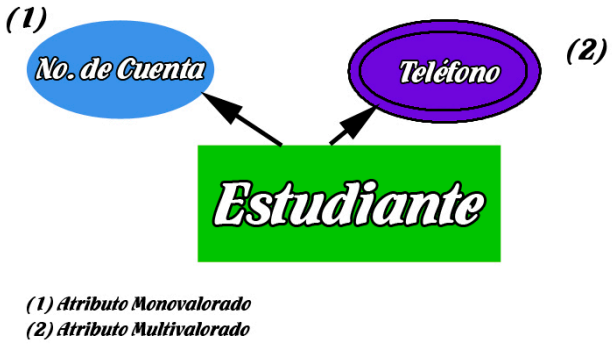


Imagen 1.7 Atributos monovalorados y multivalorados

-Atributos derivados. Estos atributos se refieren a aquellos atributos que pueden ser calculados a partir de otros atributos. Como ejemplo es que podemos calcular la edad en base a la fecha de nacimiento.

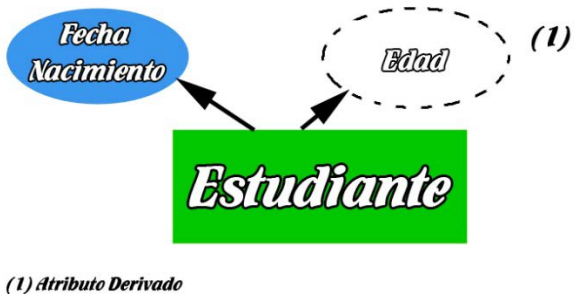


Imagen 1.8 Atributo derivado

Cabe mencionar que un atributo puede tomar un valor nulo cuando una entidad no tiene un valor para este atributo, a estos se les denomina nulos no aplicables.

En cuanto a las relaciones, se definen como *una asociación entre diferentes entidades*. La función que desempeña una entidad en una relación se llama papel de la entidad. Para poder relacionar una entidad con otra se tiene que usar una correspondencia de cardinalidades, que es la encargada de expresar el número de entidades a la que otra entidad puede estar asociada vía un conjunto de relaciones[17].

Para un conjunto de relaciones binarias R entre los conjuntos de entidades A y B, la correspondencia de cardinalidades de ser una de las siguientes[16]:

-Uno a uno: una entidad en *A* se asocia con a lo mucho una entidad en *B,* y una entidad en *B* se asocia a lo mucho con una entidad en *A*. Se expresa como *1:1*.

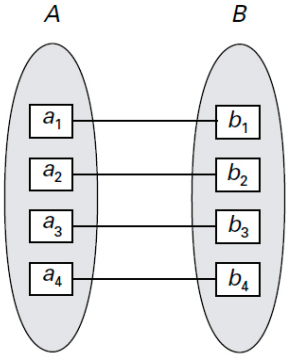


Imagen 1.9 Cardinalidad uno a uno.

-Uno a varios: una entidad en *A* se asocia con cualquier número de entidades en *B* (ninguna o varias), mientras una entidad en *B* se puede asociar con a lo mucho una entidad en *A*. Se expresa como *1:n*.

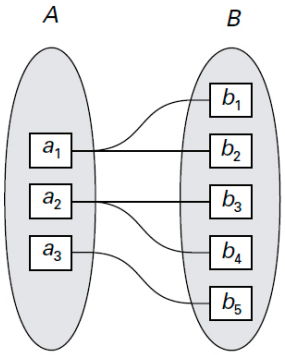


Imagen 1.10 Cardinalidad uno a varios.

-Varios a uno: una entidad en *A* se asocia con a lo mucho una entidad en *B*, mientras una entidad en *B* se puede asociar con cualquier número de entidades (ninguna o varias) en *A*. Se expresa como *n:1*.

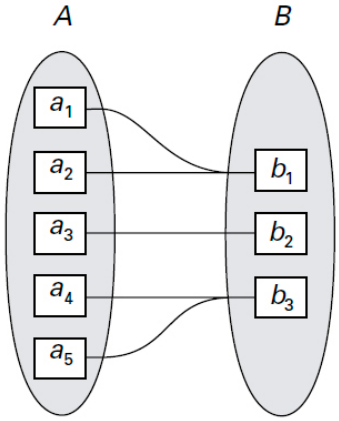


Imagen 1.11 Cardinalidad varios a uno.

-Varios a varios: una entidad en *A* se asocia con cualquier número de entidades (ninguna o varias) en *B*, y una entidad en *B* se asocia con cualquier número de entidades (ninguna o varias) en *A*. Se expresa como *n:n*.

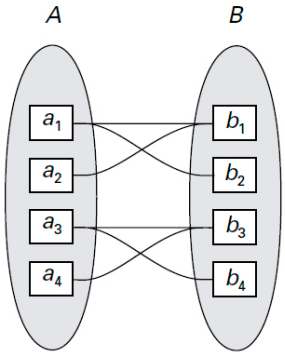


Imagen 1.12 Cardinalidad varios a varios.

La correspondencia de cardinalidades apropiada para un conjunto de relaciones particular depende obviamente de la situación del mundo real que el conjunto de relaciones modela.

Para representar una relación en un diagrama entidad relación de forma conceptual, se utiliza un rombo que a través de dos líneas relacionará las entidades. Para saber qué tipo de cardinalidad se maneja se ve representado con puntas de flecha como se muestra en la imagen 1.13

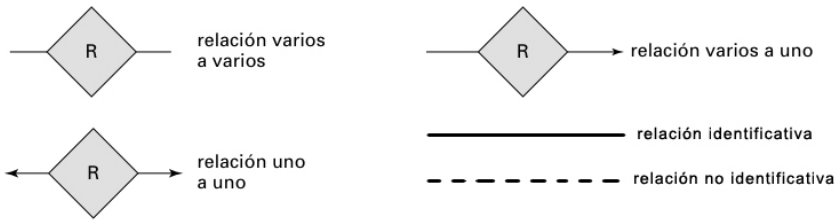


Imagen 1.13 relaciones y cardinalidad

Como también se puede observar en la imagen 1.13 existen las relaciones de tipo identificativas y no identificativas[16].

Las relaciones identificativas se encargan de difundir la llave primaria de la entidad padre a la llave primaria del hijo. Esta relación se representa con una línea continua y es obligatoria.

Las relaciones no identificativas heredan la llave primaria de la entidad padre a los atributos no-llave del hijo. Son representadas por una línea discontinua y pueden ser optativas.

En el diagrama entidad-relación es necesario definir distintos tipos de claves para poder identificar una entidad de otra. Los valores de los atributos de una entidad deben ser tales que permitan identificar *unívocamente* a la entidad, en otras palabras, no se permite que ningún par de entidades tengan exactamente los mismos valores de sus atributos. De esta forma podemos decir que la clave nos sirve para poder identificar un conjunto de atributos suficiente para distinguir las entidades entre sí.

Una superclave es un *conjunto de uno o más atributos que, tomados colectivamente, permiten identificar de forma única una entidad*[10], a menudo interesan las superclaves tales que los subconjuntos propios de ellas, estas son llamadas claves candidatas. Se usará el término de clave primaria para denotar *una clave candidata que es elegida por el diseñador de la base de datos como elemento principal para identificar las entidades dentro de un conjunto de entidades*[10], la clave primaria deberá ser elegida de manera que sus atributos nunca, o muy rara vez, cambien.

Un conjunto de entidades puede no tener suficientes atributos para formar una clave primaria, tal conjunto de entidades es denominado conjunto de entidades débiles, por el contrario aquel conjunto de entidades que tiene una clave primaria, es denominado conjunto de entidades fuertes. Cada entidad débil, debe estar asociada con una entidad identificadora, es decir que *un conjunto de entidades débiles depende existencialmente del conjunto de entidades identificadoras*.

Como complemento a los conceptos ya mencionados y con el fin de profundizar en las restricciones que se le pueden asignar a este diagrama, se hará referencia al diagrama entidad relación en su versión extendida.

El diagrama entidad relación extendido, describe con un alto nivel de abstracción la distribución de datos almacenados en un sistema. Esta versión del modelo incorpora deter-

minados conceptos o mecanismos de abstracción para facilitar la representación de ciertas estructuras del mundo real[17]:

La generalización, permite abstraer un tipo de entidad de nivel superior (supertipo) a partir de varios tipos de entidad (subtipos); en estos casos los atributos comunes y relaciones de los subtipos se asignan al supertipo.

La especialización es la operación inversa a la generalización, en ella un supertipo se descompone en uno o varios subtipos, los cuales heredan todos los atributos y relaciones del supertipo, además de tener los suyos propios.

Se denomina categoría al subtipo que aparece como resultado de la unión de varios tipos de entidad. En este caso, hay varios supertipos y un sólo subtipo.

La agregación es un concepto de abstracción para construir objetos compuestos a partir de sus objetos componentes. Permite combinar entidades entre las que existe una interrelación y formar una entidad de más alto nivel. Es útil cuando la entidad de más alto nivel se tiene que interrelacionar con otra entidad. Por ejemplo, se tienen los tipos de entidad empresa y solicitante de empleo relacionados mediante el tipo de relación entrevista; pero es necesario que cada entrevista se corresponda con una determinada oferta de empleo. Como no se permite la relación entre tipos de relación, se puede crear un tipo de entidad compuesto por empresa, entrevista y solicitante de empleo y relacionarla con el tipo de entidad oferta de empleo. El proceso inverso se denomina desagregación.

La asociación, consiste en relacionar dos tipos de entidades que normalmente son de dominios independientes, pero coyunturalmente se asocian.

La existencia de supertipos y subtipos, en uno o varios niveles, da lugar a una jerarquía, que permitirá representar una restricción del mundo real.

Una vez construido el diagrama entidad-relación, se tiene que analizar si se presentan redundancias. Para poder asegurar su existencia se deben estudiar con mucho detenimiento las cardinalidades mínimas de las entidades, así como la semántica de las relaciones.

Los atributos redundantes, los que se derivan de otros elementos mediante algún cálculo, deben ser eliminados del diagrama entidad-relación o marcarse como redundantes.

Igualmente, las relaciones redundantes deben eliminarse del diagrama, comprobando que al eliminarlas sigue siendo posible el paso, tanto en un sentido como en el inverso, entre las dos entidades que unían.

Ya con todos los conceptos anteriores que conforman el diagrama entidad relación, se puede decir que *la estructura lógica general de una base de datos se puede expresar gráficamente mediante un diagrama entidad-relación*[18].

Los diagramas son simples y claros, cualidades que pueden ser responsables del amplio uso del diagrama entidad relación, existen diferentes formas de realizar un diagrama de este tipo como lo es utilizar rectángulos para representar las entidades y líneas para las relaciones, así como distintas simbologías para expresar la cardinalidad. Los diagramas entidad-relación también proporcionan una forma de indicar restricciones más complejas sobre el número de veces en que cada entidad participa en las relaciones de un conjunto de relaciones.

Para concluir se puede decir que el diagrama entidad-relación sirve como base para realizar el modelo relacional, siendo un mapa perfecto para destacar las restricciones que deberá tener la base de datos.

1.4 Normalización

Al diseñar una base de datos mediante el modelo relacional se pueden obtener variadas alternativas, es decir, se pueden conseguir diferentes esquemas relacionales los cuales no todos son equivalentes entre sí, debido a que en algunos la realidad será mejor representada que en otros. Es sumamente necesario conocer qué propiedades debe tener un esquema relacional para representar una realidad determinada y cuáles son los problemas que se pueden derivar de un diseño inadecuado.

Para que las relaciones de la base de datos no tengan propiedades indeseables (como lo es la redundancia) se debe hacer que cumplan una serie de restricciones. A esto se le llama normalizar relaciones.

La teoría de la normalización es un método objetivo y riguroso que se aplica en el diseño de bases de datos relacionales. Esta teoría nos permite conocer las relaciones indeseables que existen en el diseño de la base de datos, de forma que pueden ser transformadas a otras de manera más conveniente[19].

Un esquema está en una determinada forma normal si satisface un conjunto determinado de restricciones sobre los atributos[20]. Cuantas más restricciones existan, menor será el número de relaciones que las satisfagan. Y cuanta más alta sea la forma normal en la que se encuentran los esquemas de relación, menores serán los problemas en el mantenimiento de la base de datos.

Las formas normales proporcionan los criterios para determinar el grado de vulnerabilidad de una tabla a inconsistencias y anomalías lógicas. Las formas normales son aplicables a tablas individuales; es decir, que una base de datos entera está en la forma normal *n*, en otras palabras, todas sus tablas están en la forma normal *n*[5].

La Primera Forma Normal (*1FN*) fue introducida por *Christopher Codd*, en su primer trabajo. Es una restricción inherente al modelo relacional por lo que su cumplimiento es obligatorio. Consiste en la prohibición de que en una relación existen grupos repetitivos, es decir, un atributo no puede tomar más de un valor del dominio subyacente. Una relación se encuentra en *1FN* cuando no hay grupos repetitivos en sus atributos, todos los dominios de los atributos contienen únicamente valores atómicos. Aquí se eliminan los atributos multivalorados[5].

La Segunda Forma Normal (*2FN*) de igual manera que la primera fue introducida por *Christopher Codd*. Una relación está en *2FN* si además de estar en *1FN* todos los atributos que no forman parte de ninguna clave candidata tienen dependencia funcional completa respecto a cada una de las claves. Toda clave cuya relación clave está formada por un solo atributo está en *2FN*. En esta forma normal se eliminan las dependencias funcionales no totales. Siempre es posible transformar un esquema de relación que no está en *2FN* en esquemas de relación en *2FN*, sin pérdida de información o de dependencias[5].

Una relación está en Tercera Forma Normal (*3FN*) si además de estar en *2FN*, los atributos que no forman parte de ninguna clave candidata facilitan información sólo acerca de las claves y no acerca de los atributos. Cada atributo no clave es dependiente no transitivamente de la clave primaria. Aquí se eliminan las dependencias funcionales transitivas. Siempre es posible transformar un esquema de relación que no esté en *3FN* en esquemas de relación *3FN*, sin pérdida de información o de dependencias[5].

Una relación está en Forma Normal de Boyce-Codd (*FNBC*) si lo está en *3FN* y además el conocimiento de las claves permite averiguar todas las relaciones existentes entre los datos de la relación. Las claves candidatas deben ser los últimos descriptores sobre los que se facilita información por cualquier otro atributo. Cada determinante (atributo con el cual otro atributo tiene dependencia funcional total) debe ser una clave candidata. Aquí se eliminan claves candidatas compuestas que se solapan. No siempre es posible transformar un esquema de relación en *FNBC* sin que se produzca pérdida de dependencias funcionales. Sí se puede hacer sin pérdida de información[21].

Ya mencionadas las formas normales, es importante dar el concepto de dependencia funcional así como también los tipos de dependencias funcionales que existen.

Una dependencia funcional[22] *es una restricción entre dos conjuntos de atributos de la base de datos*. Dada una relación *R* el atributo *y(R)* depende funcionalmente del atributo *x(R)* sí y sólo sí un solo valor de *y(R)* está asociado a cada valor de *X(R)* en cualquier momento dado. Los atributos *x* e *y* pueden ser compuestos, es decir, que pueden ser más de un atributo.

$$R.x \rightarrow R.y$$

X determina funcionalmente a *y* ó *y* depende funcionalmente de *x*.

Dependencia funcional total o completa[22]. Si *x* es compuesto, se dice que *y* tiene una dependencia funcional completa de *x* si depende funcionalmente de *x* pero no depende de ningún subconjunto del mismo[31].

Dependencia funcional trivial[22]. Una dependencia funcional $x \rightarrow y$ se dice que es trivial si *y* es un subconjunto de *x*.

Dependencia funcional transitiva[22]. Sea el esquema de la relación *R(x,y,z)*, en el que existe las siguientes dependencias funcionales.

$$x \rightarrow y,\ y \rightarrow z,\ x \rightarrow z$$

Se dice que *z* tiene una dependencia funcional transitiva respecto a *x* a través de *y*.

Conocer las dependencias funcionales en el momento del diseño de la base de datos permite crear mecanismos para evitar la redundancia.

1.5 Propiedades ACID (Atomicity – Consistency – Isolation – Durability)

Las propiedades ACID[23] es un conjunto de características necesarias en una base de datos para garantizar un comportamiento predecible reforzando la función de las transacciones como proposiciones de todo o nada diseñadas para reducir la carga de administración cuando hay muchas variables.

En sí, ACID es un acrónimo que responde a los términos Atomicity (atomicidad), Concistency (consistencia), Isolation (aislamiento) y Durability (durabilidad). A continuación se definirá cada uno de estos términos.

-Atomicidad: Esta propiedad se encarga de asegurar que la transacción se haya realizado por completo o en su defecto no se haya realizado en absoluto, garantizando que en dado caso de un fallo del sistema la operación no se vea realizada a medias. Se dice que una operación es atómica cuando es imposible para otra parte de un sistema encontrar pasos intermedios. Como ejemplo podemos decir que si una operación cuenta con cierto número de pasos a seguir para que la tarea sea completada, la atomicidad es la encargada de que si uno de los pasos no llega a ejecutarse en su totalidad la operación será cancelada por completo.

-Consistencia: Es la propiedad que asegura que sólo se empieza aquello que se puede acabar. La Integridad de la base de datos nos permite asegurar que los datos son exactos y consistentes, es decir que estén siempre intactos, sean siempre los esperados y que de ninguna manera cambien ni se deformen. De esta manera podemos garantizar que la información que se presenta al usuario será siempre la misma.

-Aislamiento: Esta propiedad se encarga de asegurar que una operación no afecte a otra. Esto nos permite garantizar que al realizar dos operaciones distintas sobre la misma información, cada operación será independiente y no generarán ningún error por dicha causa. Esta propiedad define cómo y cuándo los cambios producidos por una operación se hacen visibles para las demás operaciones concurrentes.

-Durabilidad: También conocida como *persistencia* es la encargada de asegurar que una vez que la operación haya sido realizada, ésta persistirá y no se podrá deshacer aunque exista alguna falla en el sistema y así los datos podrán sobrevivir.

Con estas propiedades puestas en práctica se garantizan aspectos que son de importancia para el funcionamiento de una base de datos.

1.6 Structured Query Language (SQL)

SQL[24], por sus siglas en inglés *Structured Query Languague,* es el lenguaje estándar que se utiliza para trabajar con bases de datos del tipo relacional y es soportado prácticamente por todos los productos en el mercado, originalmente fue desarrollado por *IBM* a principios de los sesentas y fue implementado por primera vez a gran escala en un prototipo de *IBM* de nombre *System R*. En un principio SQL fue diseñado para ser específicamente un sublenguaje de datos, sin embargo más tarde se convirtió en un lenguaje completo.

SQL se define como un lenguaje declarativo de acceso a las bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas. Una de sus principales características es el manejo del álgebra y el cálculo relacional que permite efectuar consultas con el fin de recuperar información de interés de bases de datos, así como también hacer cambios en ella.

En pocas palabras el lenguaje SQL nos permite ingresar datos en forma de código a la computadora que conformará la base de datos, así como también realizar las llamadas necesarias para consultar información de la base de datos y poder modificarla si se cuenta con los permisos necesarios.

SQL, a través de sus sentencias, se puede agrupar en una serie de lenguajes de acceso a la base de datos para de esta forma poder estructurar las operaciones que en ella se realizan. Para poder definir la base de datos se utiliza el *DDL (Data Definition Language)*, para realizar consultas y modificar datos dentro de la base se usa el *DML (Data Manipulation Language)*, y finalmente para controlar el acceso a los datos y administrar los permisos de usuario se hace uso del *DAL (Data Access Language)*.

El DDL es un lenguaje que definirá la estructura de la base de datos, el cual define como el sistema que organiza internamente los datos, se encarga de la creación, modificación y eliminación de los objetos que conforman a la base, es decir los metadatos, recordando que los metadatos son los datos de los datos. El *DDL* cuenta con una serie de instrucciones que ayudan al lenguaje a realizar su tarea, estas instrucciones serán vistas a continuación con un poco de detalle.

A continuación se mostrarán algunas sentencias básicas.

La sentencia *CREATE TABLE* sirve para crear la estructura de una tabla, nos permite definir las columnas que tiene y ciertas restricciones que deben cumplir esas columnas. El formato de la sentencia es el siguiente:

CREATE TABLE <nombre_tabla>(
[<columna 1>] [{tipo_de_datos_para_columna_1}],
[<columna 2>] [{tipo_de_datos_para_columna_2}],
...);

Una vez que se crea la tabla en la base de datos, hay ocasiones que es necesario cambiar la estructura de la tabla, entre los casos típicos en los que ocurre esta necesidad sobre las columnas está la eliminación, la agregación, el cambio de nombre, el cambiar el tipo de datos, cambiar la especificación de una clave primaria, agregar una restricción, entre otras. Para estas necesidades existe la sentencia *ALTER TABLE*, que nos ayudará a

realizar las modificaciones necesarias sobre una tabla en la base de datos cambiando así su estructura. La sintaxis para esta sentencia es:

ALTER TABLE <nombre_tabla>
[modificar especificación];

Ya creada y modificada la tabla, se puede decidir que necesitamos eliminar una tabla dentro de la base de datos por cualquier razón. De hecho sería problemático no poder contar con esta opción ya que crearía una serie de dificultades para el mantenimiento de la base. Para esto existe la sentencia *DROP TABLE*, que como ya mencionamos nos permite eliminar una tabla creada con anterioridad. La sintaxis de esta sentencia es:

DROP TABLE <nombre_tabla>;

En caso de desear eliminar los datos de una tabla sin borrar la estructura de la misma podemos servirnos de la sentencia *TRUNCATE TABLE*, que hará un borrado de todos los datos contenidos en la tabla pero sin borrar la tabla ni su estructura, permitiendo así almacenar nuevos datos. La sintaxis para esta sentencia es:

TRUNCATE TABLE <nombre_tabla>;

Estos comandos vistos anteriormente son parte del *DDL* que ayudan a la base de datos a formar y modificar su estructura mediante SQL.

El DML nos ayuda a consultar, insertar y modificar los datos dentro de la base de datos, para esto se auxilia de diversas sentencias que se explicarán a continuación.

La sentencia *SELECT* nos permite consultar los datos almacenados en una tabla de la base de datos. El formato de la sentencia es el siguiente:

SELECT [ALL | DISTINCT]
<nombre_campo> [{,<nombre_campo>}]
FROM <nombre_tabla>|<nombre_vista>
[{,<nombre_tabla>|<nombre_vista>}]
[WHERE <condicion> [{ AND|OR <condicion>}]]
[GROUP BY <nombre_campo> [{,<nombre_campo >}]]
[HAVING <condicion>[{ AND|OR <condicion>}]]
[ORDER BY <nombre_campo>|<indice_campo> [ASC | DESC]
[{,<nombre_campo>|<indice_campo> [ASC | DESC]}]]

La sentencia *INSERT* permite llevar a cabo el proceso de inserción de filas sobre las columnas de las tablas que conforman a la base de datos, estas filas pueden ser insertadas de forma individual o de forma múltiple. Para realizar la inserción individual de filas SQL posee la instrucción *INSERT INTO*. La inserción individual de filas es la que más comúnmente se utiliza. Su sintaxis es la siguiente:

INSERT INTO <nombre_tabla>
[(<campo1>[,<campo2>,...])]
values
(<valor1>,<valor2>,...);

La sentencia *INSERT* permite también insertar varios registros en una tabla. Para ello se utiliza una combinación de la sentencia *INSERT* junto a la sentencia *SELECT*. El resultado es

que se insertan todos los registros devueltos por la consulta. A continuación mostramos la sintaxis:

INSERT INTO <nombre_tabla>
[(<campo1>[,<campo2>,...])]
SELECT
[(<campo1>[,<campo2>,...])]
FROM
<nombre_tabla_origen>;

Para poder utilizar la inserción múltiple de filas se deben cumplir las siguientes normas:

*La lista de campos de las sentencias *INSERT* y *SELECT* deben coincidir en número y tipo de datos.

*Ninguna de las filas devueltas por la consulta debe infringir las reglas de integridad de la tabla en la que se realizará la inserción.

Ya insertada la información se puede tener la necesidad de modificarla, para esto se utiliza la sentencia *UPDATE*, que se encargará de actualizar la información conforme a la instrucción ejecutada. La sintaxis para esta sentencia es:

UPDATE <nombre_tabla>
SET <columna_1> = [nuevo valor]
WHERE <condición>;

Existen ocasiones en que es necesario borrar un registro de información, para eso sirve el comando *DELETE*, este comando borra los datos de una tabla que cumplan con una condición predeterminada. La sintaxis para esta instrucción es:

DELETE FROM <nombre_tabla>
[WHERE <condicion>];

Estos comandos forman parte del *DML* que ayuda a consultar, insertar y modificar datos dentro de nuestras tablas.

Otro aspecto importante a considerar en una base de datos es la seguridad, es decir los permisos que se darán a los usuarios y administradores para modificar la base de datos, para esto nos sirve el *DAL* que nos permite especificar directivas de control de acceso a los recursos, es decir a las tablas, campos, vistas y dominios.

Para conceder un privilegio sobre un recurso dado se utiliza la sentencia *GRANT* que de acuerdo a lo que especifique el enunciado, dará un privilegio para acceder a un recurso a un usuario que también será especificado. La sintaxis para esta sentencia es:

GRANT <Privilegio> ON <Recurso>
TO <Usuario>
[WITH GRANT OPTION];

Si por el contrario lo que se desea es quitar un permiso a un usuario se utilice el comando *REVOKE*, que funciona de forma inversa al comando *GRANT*. Su sintaxis es la siguiente:

REVOKE <Privilegio> ON <Recurso>
FROM <Usuario> [RESTRICT | CASCADE];

Estos dos comandos son los que nos permiten como administradores de la base de datos a dar y revocar permisos a los usuarios de acuerdo a sus necesidades.

Como se puede observar el tema de SQL es muy extenso ya que cuenta con un amplio número de comandos que se utilizan conforme a las especificaciones de la base de datos que se diseñará y las consultas que se ejecuten en ésta. Por esa razón se dio una breve descripción de la parte de SQL que se utilizará a lo largo de este trabajo con el tema que interesa que es el del diseño de una base de datos.

Un par de conceptos importantes a mencionar en este capítulo son los de vistas y transacciones. Una vista es una tabla virtual, resultado de una consulta SQL en la que se cargan los datos al momento de ser llamada[25]. Esta vista puede tener los datos de una tabla o de un conjunto de tablas, el proceso de una vista es agilizar el proceso de consulta en una base de datos.

Mientras que una transacción es un conjunto de órdenes que se ejecutan formando una unidad de trabajo, es decir, en formato invisible o atómico. Un Sistema Gestor de Base de Datos se puede considerar transaccional si es capaz de mantener la integridad de los datos, haciendo que estas transacciones se realicen por completo o de lo contrario no realizarse en absoluto, una transacción no puede quedar a la mitad de su proceso. Una transacción debe de cumplir con las propiedades *ACID* por sus siglas en inglés, atomicidad, consistencia, durabilidad y aislamiento. El leguaje SQL, provee mecanismos para especificar que un conjunto de acciones debe constituir una transacción.

-BEGIN TRAN: especifica que va a comenzar una transacción.

-COMMIT TRAN: indica al motor que puede considerar la transacción completada con éxito.

-ROLLBACK TRAN: indica que se ha alcanzado un fallo y debe restablecer la base al punto de integridad.

El ejemplo más claro de lo que es una transacción es el traspaso de dinero entre dos cuentas bancarias, esto se hace mediante dos operaciones: restar a la cuenta origen la cantidad a transferir y sumar esta cantidad a la cuenta destino. Para garantizar la atomicidad de esta transacción se tiene que verificar que las dos operaciones se hayan realizado con éxito.

1.7 Avance tecnológico

El avance tecnológico en el mundo es un fenómeno que se ha vivido a lo largo de las últimas décadas, haciendo la vida personal y laboral más sencilla. Lo que antes era un lujo en cuestiones tecnológicas hoy en día es una necesidad, como lo es un teléfono celular, que comenzando como un artículo de aplicación militar, se ha convertido en un dispositivo necesario para la comunicación del ser humano, acortando distancias y permitiendo a su portador tener comunicación a lo largo de todo el globo terráqueo.

El avance tecnológico se puede separar en dos vertientes, la del *hardware* y la del *software*. Estas dos vertientes van de la mano haciendo que el crecimiento de una ayude al crecimiento de la otra y viceversa. Por ejemplo en el caso de los celulares se necesitan tanto de un equipo físico como de un sistema que aproveche los recursos para crear una interfaz que pueda usar el usuario. *Software para usar el hardware.*

Para estudiar el fenómeno del avance en tecnología se hará uso de la Ley de Moore que se basa en el documento escrito por Gordon E. Moore titulado *Cramming more components onto integrated circuits*[26]. Esta ley es un postulado el cual fue enunciado en el año de 1965 y que posteriormente se ha ajustado en dos ocasiones, desde entonces se ha cumplido hasta nuestros días. La ley dice lo siguiente:

"La complejidad de los componentes se ha multiplicado aproximadamente por 2 cada año. A corto plazo, se puede esperar que esta tasa se mantenga o incluso que aumente. A largo plazo, la tasa de crecimiento es menos predecible, aunque no hay razón para creer que no permanecerá constante por lo menos durante otros 10 años. Es decir, en 1975 el número de componentes en cada circuito integrado de bajo coste será de 65.000. Creo que un circuito tan grande puede construirse en una única oblea de silicio".

Para una mejor comprensión de este enunciado es importante entender la suma importancia que tienen los procesadores en un equipo de cómputo, siendo el corazón de estos.

Aunque formulada de manera empírica, la ley de Moore se convirtió en un modelo perfectamente validado por la industria. El número de transistores por unidad de superficie se duplicaba e iba llegando una nueva tecnología (con mayor escala de integración) que dejaba obsoleta la anterior. El ritmo, al inicio, fue vertiginoso pero, en 1975, Gordon Moore tuvo que reajustar su propia ley para cambiar el factor del tiempo y ajustarlo a 2 años porque la industria no avanzaba tan rápidamente.

Desde el ajuste de 1975, la ley de Moore ha seguido siendo el patrón de referencia dentro del desarrollo de circuitos integrados pero, evidentemente, el aumento de la escala de integración (reducción del tamaño de transistores) implica muchísimos retos para los fabricantes. Los procesadores *Intel Haswell* están compuestos por 1.400 millones de transistores de apenas 22 nano metros de tamaño cada uno; la tecnología de 22 nanómetros es, prácticamente, la "*gran barrera del silicio*" y nos acercamos a un terreno en el que este material podría presentar inestabilidades si se sigue aumentando la escala de integración.

La ley de Moore no es una ley en el sentido científico, sino más bien una observación, y ha sentado las bases de grandes saltos de progreso.

Gordon Moore solía estimar que el número de transistores vendidos en un año era igual al número de hormigas en el mundo, pero para el 2003 la industria producía cerca de 10^{19} transistores y cada hormiga necesitaba cargar 100 transistores a cuestas para conservar la precisión de esta analogía.

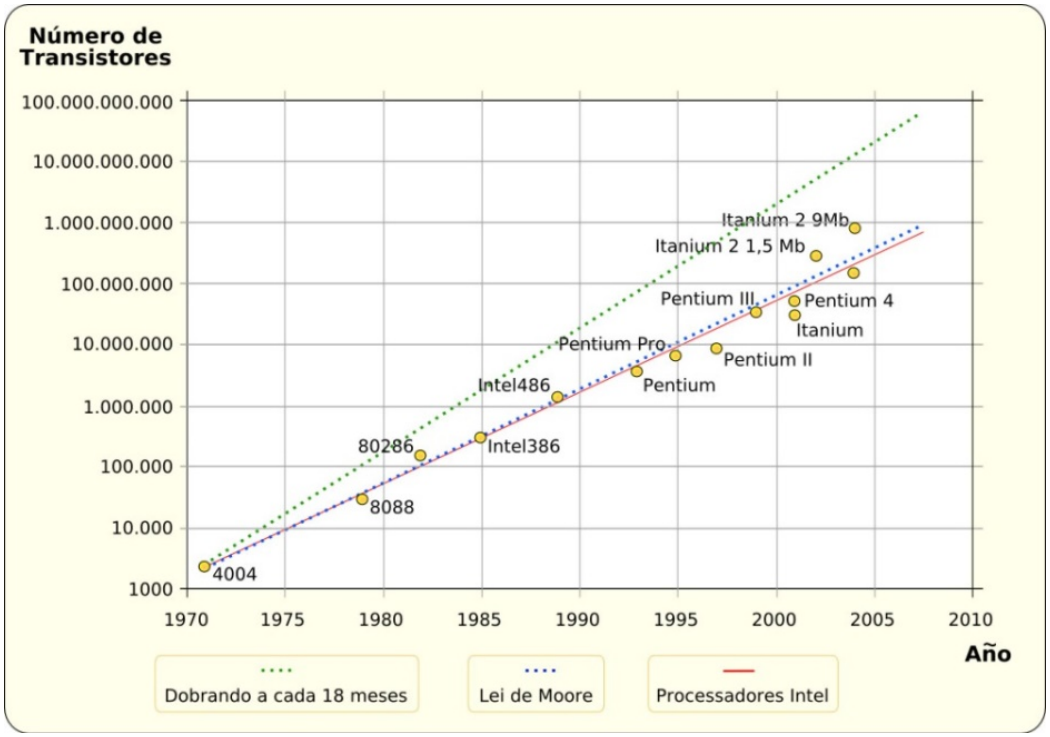


Imagen 1.14 Ley de Moore

En la imagen 1.14 se puede observar como esta ley se ha ido cumpliendo en materia de procesadores, dejando así en claro lo grande que el avance tecnológico que ha sufrido nuestro mundo.

Así, con el gran avance constante que se tiene en procesadores es posible que otro hardware crezca a la par para que se diseñe software más complejo que haga los procesos computacionales más sencillos y eficientes.

CAPÍTULO II
Análisis

2.1 Planteamiento del problema

Viviendo en la era de la información, donde los datos utilizados en una institución educativa tan importante, como lo es la Facultad de Ingeniería, son manejados a través de diversos sistemas de cómputo y alojados en bases de datos, es menester contar con una estructura confiable para el almacenamiento y consulta de dicha información. Esto se logra con un diseño de base de datos normalizado que permita cumplir con las necesidades que la institución presente.

La *Secretaría de Servicios Administrativos* (SSA) tiene a su cargo realizar varios procedimientos informáticos para el manejo de información, los cuales funcionan de manera independiente y cuyas bases de datos no se encuentran normalizadas.

Sumado a este problema, se tiene que dichos sistemas fueron diseñados de acuerdo a las necesidades y recursos del tiempo en que fueron planeados y debido a que con el paso del tiempo las necesidades de la institución han ido evolucionando es necesario que sus sistemas evolucionen del mismo modo.

Una de las limitantes que existió en el pasado para poder generar un diseño que fuese normalizado fue no contar con la memoria de procesamiento necesaria para alcanzar dicho fin; hoy en día esa limitante se ha superado, por lo que el diseñar una base de datos normalizada es posible.

En resumen, la problemática a resolver es generar un diseño de base de datos normalizado que sirva como centro de información a los sistemas que conforman a SIIFI y que cumpla con las necesidades existentes presentadas por la USECAD y la SSA.

2.2 Requerimientos

Para poder conocer con detalle la información necesaria para diseñar la base de datos, se debe partir de los requerimientos del sistema en sí, así que en primera instancia se presentará la lista de requerimientos para el Sistema Integral de Información de la Facultad de Ingeniería (SIIFI) y partiendo de esta información, se podrá conocer la serie de requerimientos que conformará a la base de datos que servirá al sistema.

Los requerimientos que se presentarán fueron especificados por la Unidad de Servicios de Cómputo Administrativos (USECAD) de acuerdo a las necesidades que deberá cumplir el SIIFI.

2.2.1 Requerimientos del sistema

Se debe crear un sistema capaz de almacenar e interactuar con la interrelación de información de los siguientes tópicos:

-Alumnos
-Académicos
-Divisiones
-Coordinaciones
-Departamentos
-Carreras
-Asignaturas
-Planes de estudio
-Módulos de salida
-Salones
-Asignación de grupos
-Historiales
-Horarios

Este sistema deberá considerar la información necesaria de cada una de los apartados anteriores para que el sistema sea capaz de diferenciarlas una de otras a través de sus llaves. La información necesaria que conformará cada tópico se presenta a continuación.

La entidad ALUMNO será uno de nuestros entes centrales y el cual se con la base de datos, ya que la mayoría de procesos a realizar el alumno tiene gran influencia. Por ejemplo, en uno de los procesos que se servirá de la base de datos de SIIFI, como lo es el proceso de reinscripción, el alumno se relaciona con las asignaturas que desea inscribir, los profesores que impartirán dicha asignatura, el plan de estudios al que pertenece, los horarios en que tomará sus materias, así como otros aspectos que se relacionan con el proceso.

SIIFI deberá ser capaz de mostrar las relaciones entre el alumno y las entidades con que éste interactúe. A continuación se muestran los datos básicos con los que cada alumno deberá de contar:

-Nombre completo del alumno
-Número de cuenta
-Edad
-Dirección de residencia
-Teléfono
-Carrera a la que pertenece
-Plan de estudios que cursa
-Módulo de salida (de existir en el plan)
-Generación
-Fotografía (de contar con la misma)
-Zona geográfica de residencia (Código postal)
-Correo electrónico

Con fines ilustrativos, si en el sistema se deseara ver la información básica de un alumno se vería como el ejemplo mostrado en la imagen 2.1

Imagen 2.1 Información alumno

Para ACADÉMICO, las relaciones con otras entidades serán variadas como lo es el proceso de reinscripción y la asignación semestral de horarios. La información básica con la que deberá de contar cada académico es la siguiente:

-Nombre completo
-Número de trabajador
-Registro Federal de Contribuyentes (RFC)
-Dirección (de contar con esta información)
-Teléfono (de contar con esta información)
-Correo electrónico

La imagen 2.2 muestra el ejemplo de los datos que deberá contener la consulta sobre la información básica de un académico en específico.

Imagen 2.2 Información académico

Sumado a esta información básica sobre el académico, como se menciona anteriormente, deberá proporcionar la información de los grupos que tiene asignados así como los horarios de los mismos (actuales y pasados) pero estas especificaciones se darán más adelante cuando se muestre el cómo deberá de funcionar el proceso de asignación.

En el caso de DIVISIÓN, COORDINACIÓN y DEPARTAMENTO, deberán obedecer un orden jerárquico en el cual la división ocupa el grado más alto y departamento el más bajo. Se puede ver como un conjunto en el que coordinación pertenece a división mientras qué departamento pertenece a coordinación. Esto se puede observar en la imagen 2.3

Imagen 2.3 Jerarquía

En palabras sencillas: una división tiene a su cargo coordinaciones mientras que dichas coordinaciones tienen a su cargo departamentos.

Para cada uno de estos tópicos se deberá de contar con las claves de cada uno (establecidas por la facultad) así como su nombre y en caso de tener una jerarquía posterior o anterior, tener esa referencia. En resumen los campos que deberá contener cada uno de estos tópicos son los siguientes:

Para DIVISIÓN:
- -Clave de la división
- -Nombre

Para COORDINACIÓN:
- -Clave coordinación
- -Nombre
- -División a la que pertenece

Para DEPARTAMENTO:
- -Clave departamento
- -Nombre
- -Coordinación a la que pertenece

Como ejemplo, si se quisiera conocer la información de una división con las coordinaciones a su cargo sería algo como lo mostrado en la imagen 2.4

Imagen 2.4 Información división

De igual manera, si lo que se desea es conocer los departamentos que están a cargo de alguna de las coordinaciones se observará como el ejemplo de la figura 2.5

Imagen 2.5 Información coordinación

En los dos ejemplos anteriores se demuestra la jerarquía entre división, coordinación y departamento, pero cabe aclarar que no necesariamente una coordinación tenga a su cargo algún departamento y de igual manera una división puede no estar a cargo de alguna coordinación, por lo que se deberá analizar esta situación para encontrar la forma más adecuada de estructurar esta información.

Continuando con el uso de jerarquías dentro de la Facultad de Ingeniería, es el turno de saber cómo se estructurará la información respecto a ASIGNATURA. Una asignatura forzosamente pertenece a un departamento y deberá contar con los siguientes datos:

-Clave de la asignatura
-Nombre

-Departamento al que pertenece (Planes anteriores algunas asignaturas sólo pertenecías a una Coordinación sin pertenecer a un Departamento)
-Si es teórica o laboratorio
-De ser laboratorio, la teoría a la que pertenece

Ésta es la información básica de las asignaturas en la facultad, aclarando que una teoría puede tener o no un laboratorio asignado, pero un laboratorio forzosamente debe pertenecer a una teoría. Existen otros aspectos que están relacionados con el tópico asignatura como lo son los créditos que tienen, la seriación previa y posterior, el semestre en que se imparte, entre otros aspectos, pero esta información varía respecto al plan de estudios que contiene a la asignatura por lo que estos requisitos se presentarán más adelante en este mismo apartado.

Un ejemplo de cómo se presentará la información base de una asignatura determinada se muestra en la imagen 2.6

Imagen 2.6 Información asignatura

Otro tópico que el Sistema Integral de Información de la Facultad de Ingeniería debe de contemplar es el de CARRERA. Cabe mencionar que cada carrera pertenece solo a una división por lo que se debe incluir a qué división pertenece cada carrera. Los datos que deberá tener carrera son:

-Clave de carrera
-Nombre
-División a la que pertenece

Para obtener la relación entre división y carreras puede ser consultada en los documentos oficiales de nuestra facultad de acuerdo al plan de estudios.

Como ya fue mencionado con anterioridad hay cierta información de las asignaturas que varía respecto al plan que pertenece por lo que ahora se presentan los datos que deberá contener PLAN DE ESTUDIOS.
Es importante que se haga mención que un plan de estudios pertenece a una sola carrera mientras que una carrera puede tener varios planes de estudios, ya que estos evolucionan con el tiempo con el objetivo de mejorar la calidad del mismo plan.

Para ejemplificar los requerimientos necesarios para este campo, tomaremos como base el plan de estudios 2010 de la carrera Ingeniería en Computación, mostrado en la figura 2.7

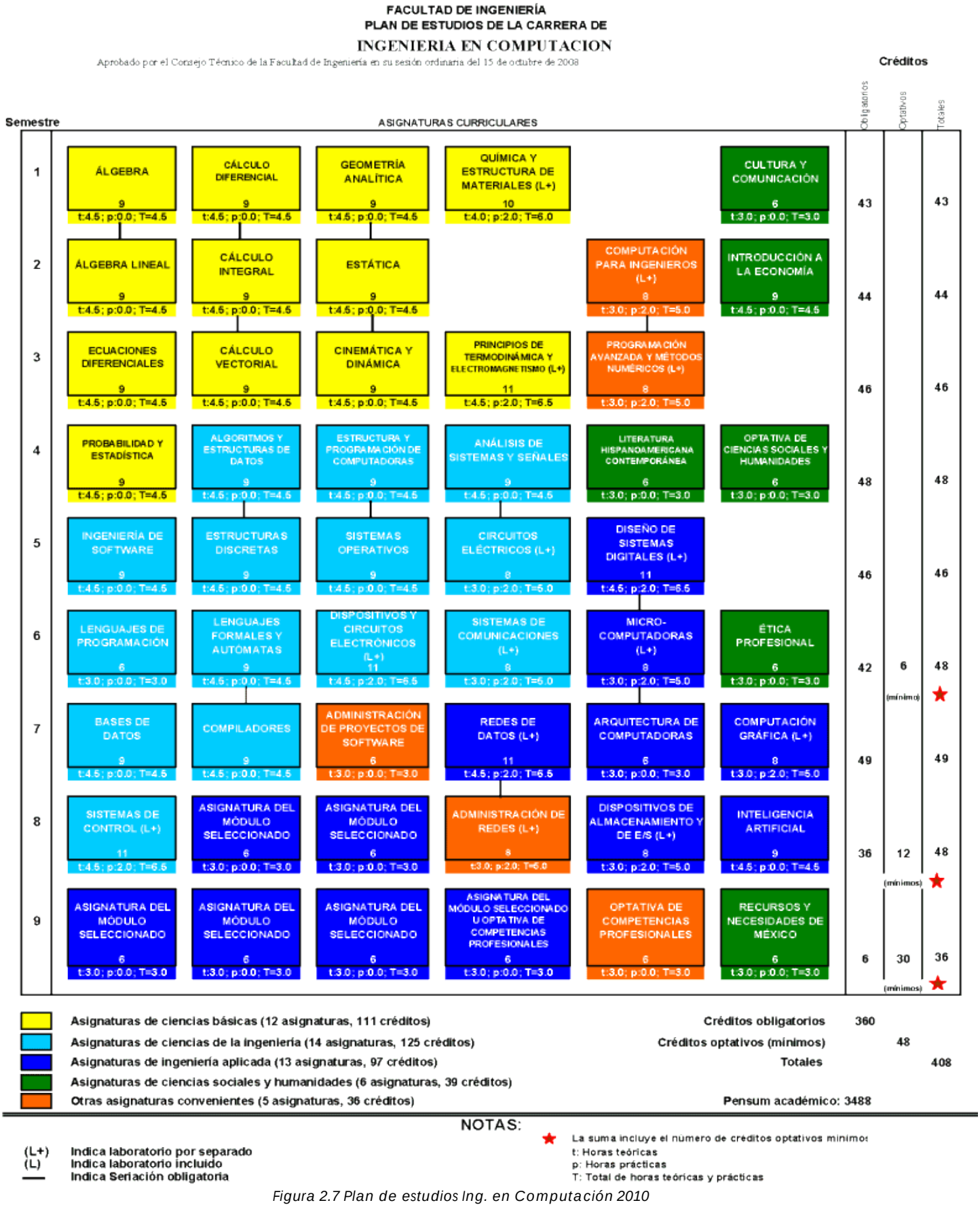


Figura 2.7 Plan de estudios Ing. en Computación 2010

Como se puede observar el plan de estudios está compuesto de varias asignaturas. Se muestra qué asignaturas tienen un laboratorio el cuál es obligatorio y cuya aprobación va ligada con la calificación de la teoría, esto representado con la letra 'L' en el plan de la figura 2.7, cuestión que se tomará en cuenta aclarando que una asignatura teórica no necesariamente tiene un laboratorio incluido, esto solo pasa en casos específicos. Existen asignaturas que son de tipo optativa, esto quiere decir que el alumno puede elegir tomarla o no, aunque en cada plan se deben de cumplir con un cierto número de créditos de este tipo de asignaturas por lo que es importante saber el tipo de cada asignatura, ya sea obligatoria u optativa. Del lado izquierdo de la imagen, se observa una serie de números que van del 1 al 9, que representan en semestre al que pertenece cada asignatura, este dato también tendrá que ser tomado en cuenta. En la parte inferior de cada asignatura se observan tres rubros representados por las letras 't', 'p' y 'T' seguidos de un número, cuyo significado es t=horas teóricas, p=horas prácticas y T=horas totales, en cuestión de estas horas es calculable el número de créditos de cada asignatura por lo cual es un dato a tomar en cuenta. Otro aspecto de relevancia es la seriación, la cual se encuentra representada a través de una línea vertical que une a dos asignaturas; la seriación en sí es un filtro jerárquico, el cual nos dice que no podemos cursar una asignatura sin antes haber aprobado la asignatura que le precede. Una asignatura puede tener cero o más asignaturas que cumplan con la restricción de seriación. Algunas carreras cuentan con un módulo terminal en el cual se imparten asignaturas, de ser el caso, el plan de estudios deberá especificar qué módulos están disponibles y las asignaturas que lo componen.

Aunado a la información presentada con relación a las asignaturas, existen datos que son propios de plan como lo son su clave, el año en que fue validado y la carrera a la que pertenece. En resumen, la información que deberá contemplar el tópico PLAN DE ESTUDIOS es la siguiente:

-Clave plan
-Carrera a la que pertenece
-Año de validación
-Asignaturas que lo componen
-Créditos de cada asignatura
-Horas teóricas
-Horas prácticas
-Horas totales
-Semestre de cada asignatura
-Créditos totales a cumplir
-Seriación entre asignaturas
-Módulos por carrera (de contar con ellos)
-Tipo de asignatura (optativa u obligatoria)

Toda esta información puede ser consultada en los medios digitales de la Facultad de Ingeniería o en las oficinas de la misma institución.

Algunas carreras cuentan con módulos de salida para la especialidad de cada alumno, para esto es importante contar con la información necesaria para cada módulo. La información con la que cada módulo de salida debe contar es la siguiente:

-Clave módulo
-Nombre módulo
-Plan al que pertenece
-Asignaturas que lo componen

La Facultad de Ingeniería cuenta con múltiples salones ubicados en los diferentes edificios de la institución, de los cuales nos interesa saber su ubicación, la cual se conforma por tres factores: edificio, piso y número de salón; con esta información concatenada se forma una clave única para cada salón, por ejemplo si un salón se encuentra en el edificio C, en el segundo piso y es el salón número 5, su clave es C205. Sumada a esta información es menester conocer la capacidad de dicho salón. Resumiendo lo dicho, la información con la que deberá contar SALÓN es la siguiente:

-Clave del salón
-Edificio
-Piso
-Número de salón
-Capacidad

La ASIGNACIÓN, como su nombre lo dice, se encargará de relacionar un grupo a una asignatura determinada, en la cual un académico será el encargado de impartir esta materia, la cual a su vez tendrá un horario que consta de una hora de inicio, una hora de finalización y una cantidad de días a la semana en la cual será dada esta clase. Para impartir una clase se necesita un lugar para hacerlo, por lo que a la asignación se incluye un salón que ocupará en el lapso estipulado. Cada asignación tendrá un cupo máximo de alumnos a inscribir, lo que se conoce como vacantes, cuyo número va descendiendo conforme los alumnos se inscriban. Nuestra institución cuenta con un programa de clases en inglés, el cual consta en que durante el periodo existen ciertos grupos en los que durante una semana se imparte la asignatura en lengua inglesa; el sistema deberá de contemplar esta situación e informar qué grupos cuentan con esta dinámica. Finalmente una asignación deberá contener una clave única la cual se conformará en parte del periodo en que se lleve a cabo por lo que también es importante conocer el periodo de dicha asignación.

La información que se menciona anteriormente aplica para el caso de una asignación ordinaria, lo que significa una inscripción ordinaria. Pero también el SIIFI deberá contemplar el caso de una asignación extraordinaria esto en el caso de los exámenes extraordinarios que se presentan sin necesidad de estar inscritos a un curso regular, si no que la inscripción es a un examen extraordinario a esto se asignan algunos datos parecidos a la asignación ordinaria como lo son la clave de la asignación, el número de grupo, el académico que aplicará el examen, el salón en que el evento se llevará a cabo, el periodo, también cuenta con una hora de inicio y una hora de finalización con la diferencia de que en cuanto al día en lugar de establecer que días de la semana se llevará a cabo la clase, tendrá una fecha fija compuesta de día, mes y año en que se llevará a cabo la práctica del examen además de la etapa a la que pertenece.

Para un mejor entendimiento de cómo deberá presentarse la información de los dos casos de asignación, se ejemplifica en la figura 2.8 el caso de la asignación ordinaria y en la figura 2.9 el caso de la asignación extraordinaria.

Imagen 2.8 Asignación ordinaria

Imagen 2.9 Asignación extraordinaria

Con los ejemplos anteriormente mostrados se puede resumir la información necesaria para cada caso.

Para una ASIGNACIÓN ORDINARIA:

-Clave asignación
-Asignatura
-Grupo
-Académico
-Hora inicio
-Hora finalización
-Días en que se imparte
-Salón
-Si la clase se impartirá en inglés
-Periodo

Para ASIGNACIÓN EXTRAORDINARIA:

-Clave asignación
-Asignatura
-Grupo
-Académico
-Hora inicio
-Hora finalización
-Día del examen
-Salón
-Etapa
-Periodo

La asignación es un elemento muy importante para evitar que un horario se traslape, por lo cual el sistema deberá ser responsable de evitar esta cuestión.

Unido todo lo anterior se debe considerar que es un sistema integrado que planea incorporar procesos como el de reinscripción y encuestas, por mencionar algunos, por lo que el diseño del sistema deberá quedar abierto para la integración de dichos procesos.

Así también debe tenerse en cuenta que es un sistema cuyos datos tendrán una actualización constante (semestre tras semestre) por lo que se tiene que contar con un proceso de administración que permita el alta, baja y actualización de información.

El Sistema Integral de Información de la Facultad de Ingeniería no está disponible para el público en general, sino que sólo un grupo específico de usuarios pueden tener acceso a él con distintos privilegios, como lo puede ser sólo de consulta o el permiso de realizar cambios dentro de la información almacenada en el sistema, por lo que se tiene que considerar una serie de usuarios y contraseñas para los usuarios del sistema.

Es importante destacar que con el tiempo las necesidades de un sistema suelen cambiar por lo que pueden surgir nuevos requerimientos a incorporar al sistema, por lo mismo el sistema deberá estar abierto a modificaciones en su núcleo, y para lograr esto es importante se tenga bien documentada la estructura del sistema para que en un futuro el equipo de trabajo dedicado a realizar alguna modificación tengan toda la documentación necesaria para realizarla de forma más sencilla y correcta.

2.2.2 Requerimientos de software y hardware

Por cuestiones de seguridad, ya que se maneja con información personal y este pretende ser un trabajo público, no se hará mención de la marca ni modelo de hardware así como tampoco los nombres del software a utilizar, sólo se mencionarán características de los mismos. Esto a petición de la USECAD.

Comenzando con el *hardware* es suficiente mencionar tres aspectos: potencia de procesador, espacio de almacenamiento y memoria de procesamiento del equipo. Así podemos decir que se cuenta con un servidor con un procesador que alcanza los 2.85 GHz de

potencia, cuenta con un almacenaje de 500 GB y una memoria de procesamiento de 32 GB.

En cuanto a *software* se utiliza un SGBD basado en el lenguaje SQL para bases de datos relacionales que cuenta con la posibilidad de llevar a cabo el diseño de este sistema. El sistema operativo es totalmente compatible con el servidor y el SGBD.

Ya con la lista de requerimientos tanto del sistema como del *hardware* y *software* que servirán a la base de datos se puede proceder con su análisis para el diseño.

2.2.3 Usuarios de SIIFI

Como ya ha sido mencionado, el Sistema Integral de Información de la Facultad de Ingeniería será un sistema no será abierto para el público general, sino que será exclusivo para miembros de la Facultad de Ingeniería que para acceder a él deberán contar con un usuario y contraseña que les proporcionará la institución.

Son bien sabidas las limitaciones que se deben tener cuando el manejo de información es el objetivo a tratar, asignando diversos permisos a distintos usuarios que podrán ver la información y en dado caso actualizar la misma. Dichos permisos serán determinados por el sistema, para el diseño de la base de datos basta saber que será administrada por un DBA (Data Base Administrator) y utilizada por múltiples usuarios como lo son alumnos, académicos, administrativos, profesores, etcétera, los cuales tendrán distintos permisos para consultar y/o modificar la información que reside en la base de datos.

2.3 Análisis de requerimientos

Una vez conocida la lista de requerimientos especificada por la USECAD para el sistema que se desea desarrollar, se puede proceder a planificar la base de datos que servirá a dicho sistema.

Como se vio en el apartado anterior es importante conocer tres vertientes: el funcionamiento del sistema, el entorno en que se desarrollará y los usuarios que accederán a él. De igual forma en que se presentaron dichos requerimientos se presenta su análisis.

2.3.1 Análisis de requerimientos del sistema

Analizando la lista de requerimientos presentada, se determina que el modelo a utilizar para el diseño de la base de datos es el modelo relacional ya que el software con que se cuenta es orientado a dicho modelo. Este modelo permitirá almacenar la información en forma de entidades las cuales pueden interactuar con otras entidades. Para poder traba-

jar con la relación entre entidades se hará uso del diagrama entidad-relación, a través de este diagrama se realizará el diseño de la base de datos en donde se especificarán las restricciones que regirán a la base de datos. Una vez que el diseño haya sido trazado la base de datos puede ser codificada y poblada.

Procediendo al análisis de los requerimientos, primero se expondrá la información requerida para analizar campo a campo formando grupos de información que más adelante se convertirán en las entidades del diagrama entidad relación.

Se comenzará con ALUMNO del cual se especificaron los siguientes campos:

-*NOMBRE COMPLETO*: Para este campo se utilizará el apellido paterno, el apellido materno y el o los nombres del alumno.

-*NÚMERO DE CUENTA*: Al ingresar a la UNAM le es asignado al alumno un número que lo identifica como estudiante de la institución, el cual se utilizará en este campo.

-*EDAD*: Como es bien sabido la edad de un individuo aumenta gradualmente al paso del tiempo, por lo cual sería muy tedioso modificar la edad de cada alumno año con año. Por esta razón se hará la edad calculable conociendo la *FECHA DE NACIMIENTO* de cada alumno.

-*DIRECCIÓN*: Este campo se compondrá por el domicilio en el que el alumno reside, a través de los parámetros: calle, colonia, delegación, número y código postal.

-TELÉFONO FIJO: Será un número telefónico fijo. Es un campo que podría estar vacío al no contar el alumno con ningún número telefónico.

-TELÉFONO CELULAR: Será un número telefónico móvil. Es un campo que podría estar vacío al no contar el alumno con ningún número telefónico móvil.

-TELÉFONO EMERGENCIAS: Será un número telefónico fijo para emergencias. Es un campo que podría estar vacío al no contar el alumno con ningún número telefónico para emergencias.

-TELÉFONO CELULAR DE EMERGENCIAS: Será un número telefónico móvil para emergencias. Es un campo que podría estar vacío al no contar el alumno con ningún número telefónico móvil para emergencias.

-*CARRERA*: Se indicará que carrera o carreras (en caso de carreras simultaneas) pertenecientes a la Facultad de Ingeniería cursa el alumno.

-*GENERACIÓN*: Respecto al año de ingreso se calculará a que generación pertenece el alumno.

-*PLAN DE ESTUDIOS*: De acuerdo con la carrera cursada y la generación perteneciente se indicará que plan de estudios corresponde a cada alumno.

-*MÓDULO DE SALIDA*: En caso de estar inscrito en un plan de estudios que cuente con la modalidad de módulos de salida, se indicará a cuál o cuáles pertenece el alumno una vez que el módulo haya sido elegido.

-*FOTOGRAFÍA*: Imagen pictográfica del rostro de cada alumno. Este campo puede ser opcional.

-CORREO ELECTRÓNICO: Dirección de correo de cada alumno. Este campo puede ser opcional.

Los campos pertenecientes a ACADÉMICO son los siguientes:

-*NOMBRE COMPLETO*: Para este campo se utilizará el apellido paterno, el apellido materno y el o los nombres del académico.

-*NÚMERO DE TRABAJADOR*: Al ser académico de la UNAM le es asignado un número de trabajador único con el cual es identificable, este número se utilizará para el campo.

-*REGISTRO FEDERAL DE CONTRIBUYENTE (RFC)*: Al ingresar a trabajar en la UNAM es requisito presentar el RFC para poder deducir impuestos del salario obtenido. Esta información se ocupará para el campo.

-*DIRECCIÓN*: Este campo se compondrá por el domicilio en el que el académico reside, a través de los parámetros: calle, colonia, delegación, número y código postal. Es un campo que no es obligatorio por lo que no es forzoso que dicho académico especifique esta información.

-*TELÉFONO*: Será un número. Es un campo que podría estar vacío al no contar el académico con ningún número telefónico.

-*FOTOGRAFÍA*: Imagen pictográfica del rostro de cada académico. Este campo puede ser opcional.

-CORREO ELECTRÓNICO: Dirección de correo de cada académico. Este campo puede ser opcional.

En lo referente a las divisiones, coordinaciones y departamentos existe una correlación jerárquica entre ellos en la cual la división tiene coordinaciones y las coordinaciones departamentos. Analizaremos uno a uno.

Para DIVISIÓN los campos que lo componen son:

-*CLAVE DE DIVISIÓN*: Cada división que pertenece a la Facultad de Ingeniería cuenta con una clave única la cual será el elemento ocupado por este campo.

-*NOMBRE DIVISIÓN*: Nombre otorgado a cada división. Por ejemplo: División de Ingeniería Eléctrica.

Los campos que conformarán COORDINACIÓN serán los siguientes:

-*CLAVE COORDINACIÓN*: Cada coordinación que pertenece a la Facultad de Ingeniería cuenta con una clave única la cual será el elemento ocupado por este campo.

-*NOMBRE COORDINACIÓN*: Nombre otorgado a cada coordinación. Por ejemplo: Coordinación de Matemáticas.

-*DIVISIÓN A LA QUE PERTENECE*: Por jerarquía una coordinación pertenece a una división, por lo que este campo indicará lo mismo.

En cuanto a DEPARTAMENTO los campos que lo definen son los siguientes:

-*CLAVE DEPARTAMENTO*: Cada departamento que pertenece a la Facultad de Ingeniería cuenta con una clave única la cual será el elemento ocupado por este campo.

-*NOMBRE DEPARTAMENTO*: Nombre otorgado a cada departamento. Por ejemplo: Departamento de Cálculo Diferencial.

-*COORDINACIÓN A LA QUE PERTENECE*: Por jerarquía un departamento pertenece a una coordinación, por lo que este campo indicará lo mismo.

En los dos casos anteriormente mencionados (coordinación y departamento) se puede dar la excepción de que la jerarquía no se cumpla pero por practicidad se pide que en dado caso se cree un nodo padre ficticio para respetar la categoría jerárquica por lo que es menester crear un nuevo campo opcional para indicar si el nodo es ficticio. A este campo le llamaremos *DESCRIPCIÓN* y se presentará en *COORDINACIÓN* y *DEPARTAMENTO*.

El siguiente tópico a analizar es el de ASIGNATURA cuyos campos son los siguientes:

-*CLAVE ASIGNATURA*: Cada asignatura cuenta con una clave única con la cual es identificable, información que se usará para este campo.

-*NOMBRE ASIGNATURA*: Nombre oficial de cada asignatura.

-*DEPARTAMENTO AL QUE PERTENECE*: Toda asignatura pertenece a un departamento, por lo cual este campo servirá para llenar esta información.

-*TEORÍA/LABORATORIO*: Este campo especifica si la asignatura es del tipo teoría o laboratorio. De ser laboratorio indicar la teoría a la que pertenece.

Para CARRERA se deberá contemplar los siguientes campos:

-*CLAVE CARRERA*: Cada carrera que pertenece a la Facultad de Ingeniería cuenta con una clave única la cual será el elemento ocupado por este campo.

-*NOMBRE CARRERA*: Nombre otorgado a cada carrera.

DIVISIÓN A LA QUE PERTENECE: Toda carrera pertenece a una división, por lo cual este campo servirá para llenar esta información.

El siguiente tópico de análisis va de la mano con *ASIGNATURA* ya que el PLAN DE ESTUDIOS está compuesto de muchas asignaturas y depende el plan y la carrera varían los campos a tratar. Los campos son los siguientes:

-*CLAVE DEL PLAN*: La clave de cada plan se conformará por una concatenación de caracteres entre el año en que entra en curso el plan y la carrera a la que pertenece. Cada plan tiene una clave única.

-*CARRERA A LA QUE PERTENECE*: Cada plan de estudios pertenece únicamente a una carrera, por lo que este campo definirá ese aspecto.

-*AÑO DE VALIDACIÓN*: Año en el que el plan entra en vigor.

-*ASIGNATURAS:* Todas las asignaturas que contiene el plan de estudios.

-*CRÉDITOS DE ASIGNATURA*: Cada asignatura tiene un número determinado de créditos dependiendo el plan al que pertenece, ese aspecto lo cubre este campo.

-*HORAS TEÓRICAS*: De cada asignatura el total de horas teóricas a la semana en que se imparte la materia.

-*HORAS PRÁCTICAS*: De cada asignatura el total de horas prácticas a la semana en que se imparte la materia.

-*TOTAL DE HORAS*: Éste es un campo calculable el cual suma el total de horas teóricas y las prácticas a lo largo de una semana.

-*SEMESTRE*: Semestre al cual una asignatura pertenece, esto servirá para la aplicación del bloque móvil.

-*CRÉDITOS TOTALES*: El total de créditos totales contemplados para el plan entre *OBLIGATORIOS* y *OPTATIVOS*.

-*SERIACIÓN*: En caso de que una asignatura tenga ligada a ella una asignatura previa o posterior indicar qué asignatura es ésta.

-*MÓDULOS DE SALIDA*: Un plan de estudios puede tener un número determinado de módulos, este campo indica cuáles son esos módulos.

-*TIPO DE ASIGNATURA*: De acuerdo al plan de estudios, este campo indicará si la asignatura pertenece al tipo obligatoria u optativa.

Es importante mencionar que en cada plan de asignaturas sólo están contempladas las asignaturas de tipo teóricas ya que los laboratorios van incluidos en estas últimas y se cuenta con una relación que lo indica.

Para MÓDULO DE SALIDA tiene los siguientes campos:

-*CLAVE MÓDULO*: Cada módulo cuenta con una clave que lo identifica de manera única.

-*NOMBRE MÓDULO*: Nombre otorgado a cada módulo.

-*PLAN AL QUE PERTENECE*: Cada plan de estudios pertenece a un plan de estudios.

-*ASIGNATURAS*: Asignaturas que conforman al módulo.

El siguiente tópico que es necesario analizar es SALÓN, el cual cuenta con los siguientes campos:

-*CLAVE DE SALÓN*: La clave es utilizada para identificar de forma única un determinado salón y es una concatenación de diversos elementos como lo son el edificio, el piso y el número de salón. Por ejemplo: A-201. En la que la letra 'A' se refiere al edificio, el número 2 se refiere al segundo piso y el número 01 al número de salón.

-*EDIFICIO*: Se compone de una sola letra y va de acuerdo al esquema organizacional trabajado por la Facultad de Ingeniería.

-*PISO*: Indica el nivel en que se encuentra ubicado cada salón.

-*NÚMERO DE SALÓN*: Un número consecutivo que indica la numeración de los salones.

-*CAPACIDAD*: Capacidad máxima de alumnos que el salón puede albergar.

Otro apartado por analizar es el de ASIGNACIÓN, que la relación mediante la cual una asignatura se subdivide en grupos a los cuales se les asigna un salón, un horario y al académico que impartirá la clase. Como se vio en la lista de requerimientos, existen dos tipos de asignación, así que se verá cada una por separado.

Para ASIGNACIÓN los campos que la componen son:

-*CLAVE DE ASIGNACIÓN*: Se trata de un número único para cada asignación el cual se compone del periodo en que fue realizada, la asignatura a la que pertenece y el grupo asignado.

-*ASIGNATURA*: Datos de la asignatura en la cual la asignación se encuentra involucrada.

-*GRUPO*: El número de grupo que tiene asignado esta orden.

-*ACADÉMICO*: Datos del académico que tendrá a su cargo dicha asignación.

-*HORA DE INICIO*: En formato de horas, la hora de comienzo de la clase.

-*HORA DE FINALIZACIÓN*: En formato de horas, la hora de término de la clase.

-*DÍAS QUE SE IMPARTE*: Días (de lunes a sábado) en que la clase tendrá actividad.

-*SALÓN*: Datos del salón en que la clase será impartida.

-*CLASE EN INGLÉS*: De acuerdo a la modalidad de la Facultad de Ingeniería de impartir cierta asignatura en inglés durante el periodo de una semana en algunos grupos, indica si la asignación en cuestión tiene dicha modalidad.

-*PERIODO*: Semestre en el que la asignación tiene validez. Por ejemplo: 2017-1.

Para ASIGNACIÓN EXTRAORDINARIA se deben contemplar los siguientes campos:

-*CLAVE DE ASIGNACIÓN*: Se trata de un número único para cada asignación el cual se compone del periodo en que fue realizada, la asignatura a la que pertenece, el grupo asignado y la etapa del examen en cuestión.

-*ASIGNATURA*: Datos de la asignatura en la cual la asignación se encuentra involucrada.

-*GRUPO*: El número de grupo que tiene asignado esta orden.

-*ACADÉMICO*: Datos del académico que tendrá a su cargo dicha asignación.

-*HORA DE INICIO*: En formato de horas, la hora de comienzo del examen.

-*HORA DE FINALIZACIÓN*: En formato de horas, la hora de término del examen.

-*DÍA DEL EXAMEN*: Fecha calendarizada en la que el examen tendrá lugar.

-*SALÓN*: Datos del salón en que el examen será aplicado.

-*ETAPA*: Cada examen extraordinario cuenta con un número de etapas en las que el examen es realizado. Este campo indica ese aspecto.

-*PERIODO*: Semestre en el que la asignación tiene validez. Por ejemplo: 2017-1.

La información que se ha visto y explicado es la que conforma la estructura básica de la base de datos *del Sistema Integral de Información de la Facultad de Ingeniería* y por la misma razón la que conformará a su base de datos. Ya analizados el grupo de datos se puede seguir a asignar propiedades particulares de cada uno de ellos para después formar las entidades debidas de nuestro Diagrama Entidad Relación generando a continuación las relaciones y restricciones con las que cada entidad cuenta.

2.3.2 Análisis de usuarios

La tarea de quién puede acceder a la base de datos es definida por el propio sistema, esto debido a que el usuario realiza la conexión directamente con el sistema y este a su vez conecta a la base de datos. Los permisos a nivel base de datos serán otorgados por un administrador de bases de datos cuyos usuarios conectarán directamente al sistema.

Se propone que se utilice un sistema integrado de políticas basado en RBAC (Roled-Based Access Control) para definir administradores y usuarios y los permisos de consulta y manipulación que tendrán sobre la información que se aloja en la base de datos.

Suponiendo que de la parte de administradores de base de datos se establecen dos niveles y que los usuarios a utilizar el sistema se dividen en 3: administrativos, alumnos y académicos, un ejemplo de algunos de los permisos que contarían los entes que interactúan con el sistema puede ser observado en la imagen 2.10

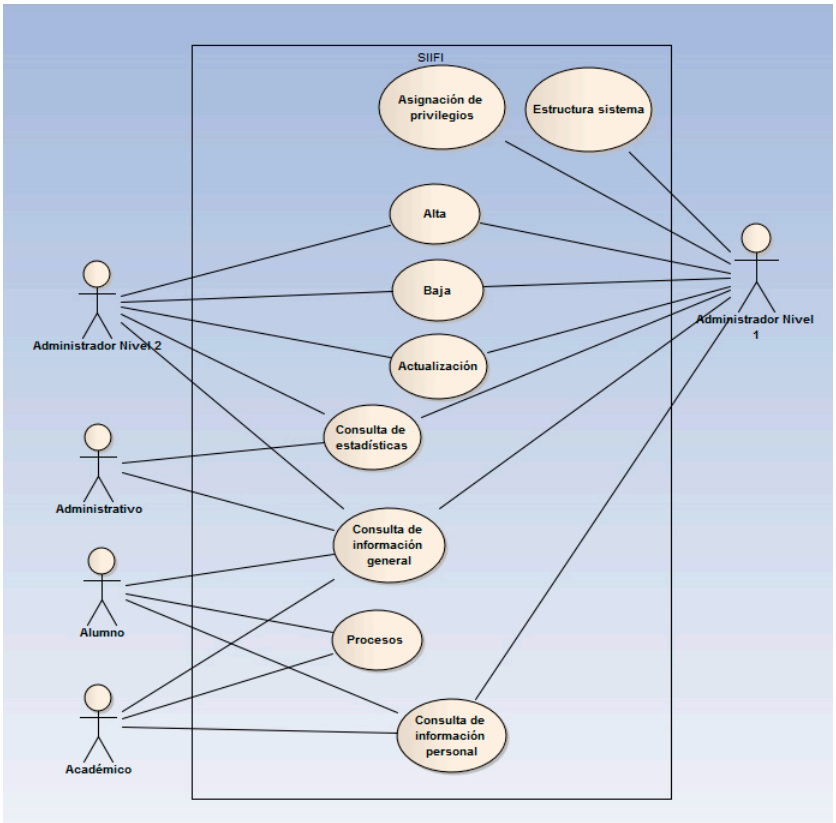


Imagen 2.10 Ejemplo RBAC

Cabe aclarar que la imagen anterior es un ejemplo básico tomando en cuenta solo dos administradores y tres tipos de usuario.

2.4 SIIFI como herramienta estadística

Sumado a las utilidades anteriormente mencionadas, SIIFI servirá como una herramienta estadística, ya que en base a los datos almacenados se podrán realizar diversas consultas que permitan arrojar un cálculo determinado. Todo esto en base a las características que nos permite el modelo relacional.

A continuación se mencionarán unos ejemplos estadísticos de la información que podría ser consultada en SIIFI. Con propósitos prácticos se mostrarán solamente las consultas sin una respuesta, esto para una vez diseñada la base de datos se puedan resolver estas consultas demostrando la funcionalidad de la base de datos. Algunas de las consultas son:

1. *Número alumnos inscribieron la asignatura de Cálculo Vectorial este semestre.*
2. *Listado de grupos de la asignatura Álgebra Lineal que cuentan con menos de 20 alumnos inscritos.*

3. *Listado de salones desocupados a las 11:30 h los días lunes.*

4. *Número de alumnos aprobados y reprobados de la materia Principios de Termodinámica y Electromagnetismo.*

5. *Número aproximado de alumnos que están en el segundo piso del Edificio A los días martes a las 7:00 h.*

6. *Académico con mayor número de alumnos reprobados en la asignatura Computación para Ingenieros.*

7. *Horario del alumno Juan Pérez para este semestre con asignaturas, grupos, profesores, horarios y salones.*

8. *Listado de todos los grupos que impartirá el profesor Luis Hernández con asignaturas, horarios, salones y número de alumnos inscritos a cada asignatura.*

9. *Estadística por carrera del número de alumnos que aprobaron la asignatura de Estática.*

10. *Lista de departamentos de la Facultad de Ingeniería ordenada por la división a la que pertenece cada uno.*

Como se puede observar, estas consultas pueden ayudarnos a detectar algún problema y así trazar una estrategia para resolverlo.

Las consultas anteriormente mencionadas serán procesadas por la base de datos y el resultado arrojado serán datos alfa numéricos que cumplan las condiciones especificadas. Con un propósito ilustrativo dichas consultas serán ejemplificadas a manera de reporte y una vez diseñada y codificada la base de datos se mostrará el resultado que se obtiene para dicha consulta. El propósito de este trabajo no va enfocado a un diseño gráfico de los resultados por lo que las imágenes que a continuación se muestran sólo son con fines ilustrativos.

Tomando como primer ejemplo la consulta número 2 que dice: *Listado de grupos de la asignatura Álgebra Lineal que cuentan con menos de 20 alumnos inscritos.* Podríamos obtener la consulta como se muestra a continuación.

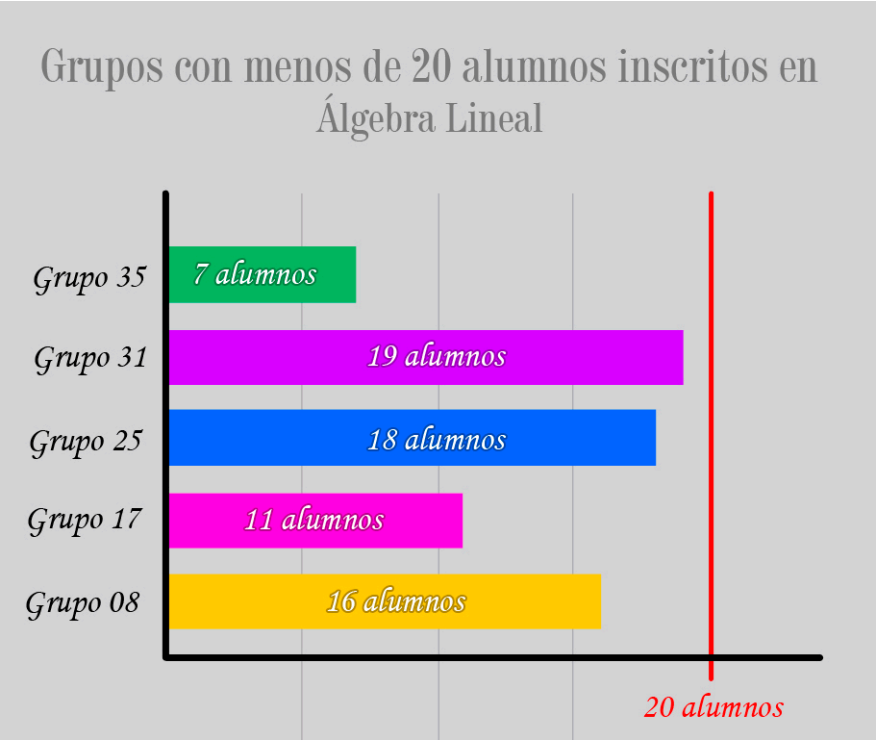


Imagen 2.11Reporte de consulta 2

Como se puede observar se resuelve la consulta indicando cinco grupos que cumplen con la restricción designada que en este caso es grupos de la materia de Álgebra Lineal con un número de alumnos inscritos menor a 20.

Como segundo ejemplo se tomará la consulta número 4 que dice: *Número de alumnos aprobados y reprobados de la materia Principios de Termodinámica y Electromagnetismo.* En la imagen 2.12 se puede observar el resultado de dicha consulta.

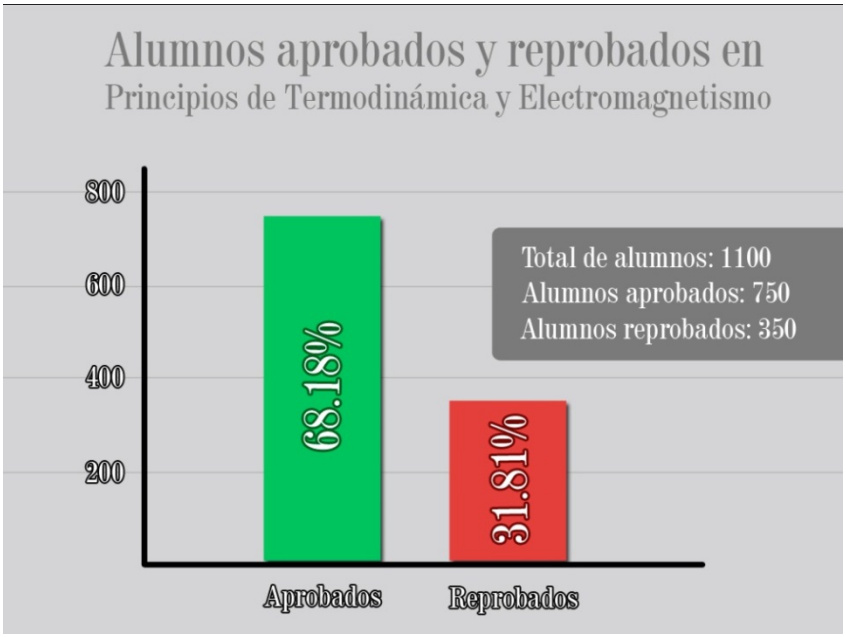


Imagen 2.12 Reporte de consulta 4

En la figura se puede observar el resultado de la consulta requerida dónde se muestra el índice de alumnos aprobados y reprobados en la asignatura de Principios de Termodinámica y Electromagnetismo.

Para el siguiente ejemplo se tomará como base la consulta número 8 que dice: *Listado de todos los grupos que impartirá el profesor Luis Hernández con asignaturas, horarios, salones y número de alumnos inscritos a cada asignatura.* A continuación se muestra el resultado correspondiente a esta consulta.

Horarios de grupos de académico

Ing. Luis Hernández

Grupo	Asignatura	L	M	M	J	V	S	Horario	Salón	Alumnos inscritos
10	Álgebra Lineal	*		*		*		7:00-8:30	A102	40
14	Cálculo Diferencial	*		*		*		8:30-10:00	A201	32
8	Programación Avanzada		*		*			8:30-10:00	B402	31
14	Álgebra Lineal		*		*			11:30-13:45	A304	15
19	Lab. Programación Avanzada						*	9:00-11:00	T001	8

Imagen 2.13 Reporte de consulta 8

En la imagen anterior se puede observar el resultado de la consulta, el cuál es el horario de grupos a impartir durante el semestre por un académico, donde se muestran los campos requeridos como lo son el número de grupo, el nombre de las asignaturas, los días que se imparten, el horario de clase, el salón asignado y el número de alumnos inscrito para dicho grupo.

El próximo ejemplo se basa en la consulta número 9, la cual dice: *Estadística por carrera del número de alumnos que aprobaron la asignatura de Estática.* En la siguiente imagen se muestra el resultado correspondiente a esta consulta.

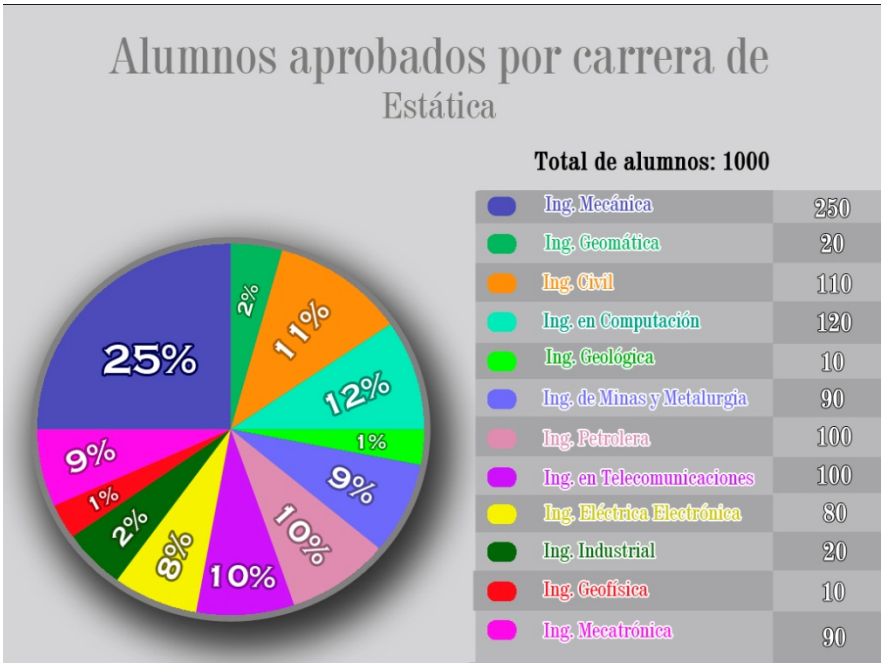


Imagen 2.14 Reporte de consulta 9

En la imagen se puede observar la consulta requerida donde se contemplan las 12 carreras que se imparten en la Facultad de Ingeniería, mostrando el porcentaje de alumnos que aprobaron una asignatura dada de cada carrera.

Por último se pondrá como ejemplo la consulta número 10 que dice: *Lista de departamentos de la Facultad de Ingeniería ordenada por la división a la que pertenece cada uno.* A continuación se mostrará el resultado correspondiente a dicha petición.

Imagen 2.15 Reporte de consulta 10.

Se puede observar que en esta ocasión se muestra la consulta el reporte requerida en forma de tablas, en la cual primeramente se catalogan por colores las divisiones que cuentan con departamentos a su cargo, seguido del listado de departamentos que tiene cada división.

De esta manera se ejemplifica algunas de las consultas que la base de datos puede procesar.

CAPÍTULO III
Desarrollo

3.1 Construcción de la base de datos

A lo largo de este capítulo se construirá el Diagrama Entidad-Relación del Sistema Integral de Información de la Facultad de Ingeniería.

Ya que se han precisado y analizado las especificaciones, procede construir la base de datos comenzando por generar el diseño

Podemos sintetizar a construcción de una base de datos principalmente en cuatro etapas (propuestas para este trabajo), las cuales son:

1. Entender el dominio del mundo real que se desea modelar.

2. Especificar un formalismo de diseño para la base de datos.

3. Traducir el diseño anterior a un lenguaje que el sistema manejador de base de datos sea capaz de comprender.

4. Poblar la base de datos.

Como se puede observar estas cuatro etapas definen la construcción básica de una base de datos.

La primera etapa de la construcción es un proceso de planeación mental, en el cual es de vital importancia se entienda el funcionamiento de la base de datos. Es una etapa de mucha importancia ya que en base a ésta trabajarán las demás, si no se logra entender el funcionamiento básico, se verá trasladado de manera errónea al lenguaje máquina.

Una vez que se ha entendido el proceso a diseñar, la segunda etapa es trasladar ese plan a papel. Esto a través de un mapa que permita pasar del mundo real a una traducción que pueda ser integrada al lenguaje máquina. El diagrama entidad-relación es ideal para esta tarea.

A través del diagrama entidad-relación se llevará a cabo el trazado de la base de datos donde encontraremos las entidades a manejar así como sus atributos y las relaciones existentes entre dichas entidades y la cardinalidad entre las mismas. Para el propósito de este trabajo se han definido algunas de las entidades que se ocuparán así como sus atributos y las características de cada uno de ellos.

Ya que haya sido trazado el diagrama que involucre todos los aspectos necesarios como lo son las restricciones y relaciones entre entidades, se puede proceder a la traducción del diseño a un lenguaje que el manejador de base de datos sea capaz de comprender, en palabras más sencillas, a codificar el diseño. El lenguaje utilizado para esta tarea es SQL, el cual como ya se ha mencionado es un lenguaje declarativo de acceso a bases

de datos relacionales que permite especificar diversos tipos de operaciones en ellas. Esta tarea se realizará mediante el sistema manejador de base de datos que nos permitirá realizar la construcción de la base de datos. Todo esto nos servirá para transformar el diagrama entidad-relación al modelo relacional.

Por último se procede a poblar la base de datos que es el llenado de la base de datos mediante los distintos datos que componen a las entidades.

Estas etapas mencionadas para la construcción de una base de datos se pueden resumir en la siguiente imagen.

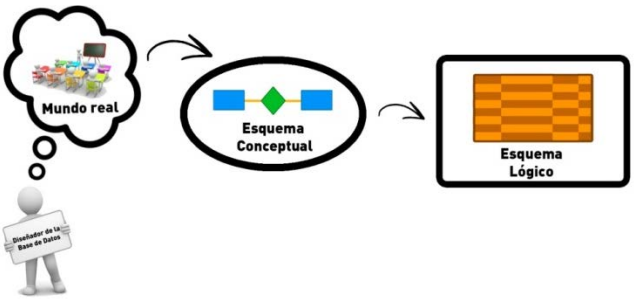


Imagen 3.1 Construcción de una base de datos

Como la imagen 3.1 ejemplifica, claramente se ven tres de las etapas de la construcción de la base de datos, donde en primera instancia el diseñador de la base tiene la percepción y comprensión de los procesos ocurridos en el mundo real. Una vez entendido es su totalidad esta etapa pasa a realizar el esquema conceptual auxiliándose del diagrama de entidad-relación. Ya que se cuenta con el diagrama que servirá como mapa para la construcción de la base de datos se procede a obtener el esquema lógico (lo que es el modelo relacional) mediante la codificación del diagrama previamente realizado.

3.2 Entidades, atributos y llaves primarias

Conforme a los datos mencionados en el apartado *2.3.1 Análisis de requerimientos del sistema,* se conformarán las entidades que servirán al diseño de la base de datos.

Utilizando el orden ocupado en el análisis se comenzará por crear la entidad ALUMNO.

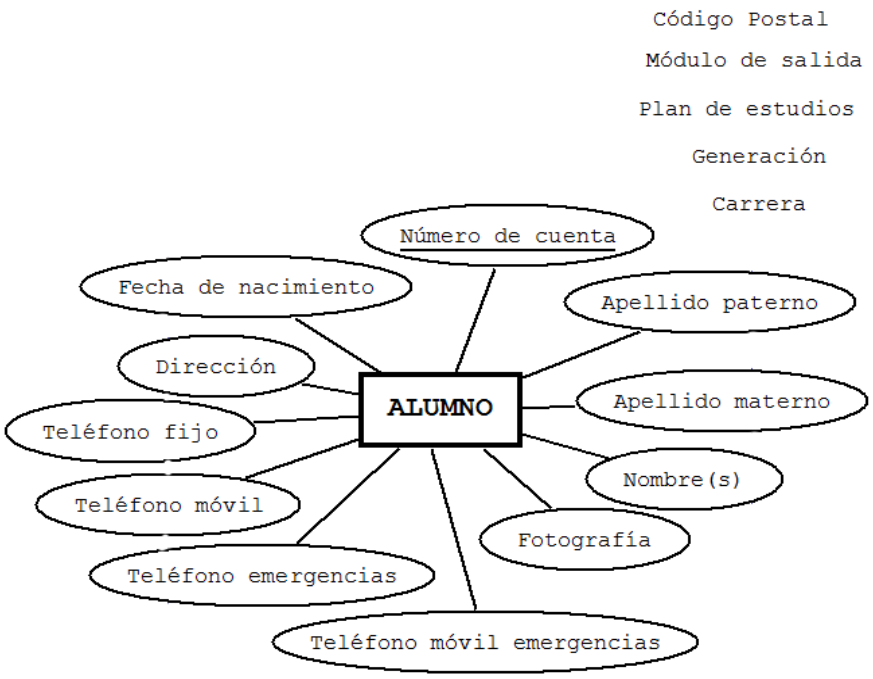


Imagen 3.2 Entidad Alumno

En la imagen 3.2 se puede observar la primera aproximación de lo que será la entidad ALUMNO. Revisando los atributos propios con los que cuenta la entidad, se observa que el atributo que servirá como *llave primaria (PK)* es el *Número de cuenta*, ya que es un número único para cada alumno y nos permite identificarlo sin que se vea repetido en otro alumno. En la imagen se hace la distinción que un atributo es llave primaria subrayándolo.

En el caso de los atributos *se* manejan separándolos en el Apellido paterno, Apellido materno y Nombre(s), esto con motivos de tener una mejor estructura en el diseño de la base y cumpliendo con las reglas de la normalización.

Los otros atributos propios de la entidad son: *Fecha de nacimiento*, *Dirección*, *Teléfono fijo, Teléfono celular, Teléfono de emergencias, Teléfono celular de emergencias* y *Fotografía*.

Por otro lado, se observa que en la parte superior derecha de la imagen 3.2 existe información que no es perteneciente a la entidad pero que es importante saber los la relación que existe con la misma.

Se debe de saber la *carrera* o *carreras* a las que está inscrito el alumno, la *generación* en la que comenzó cada carrera, el *plan de estudios* correspondiente a la carrera cursada, de haber *módulo de salida* existente en la carrera saber a cuál o cuáles pertenece cada alumno una vez que se haya inscrito a él y por último el *código postal* de la residencia del alumno. Estos casos son propios de las relaciones entre entidades, que se verá en el próximo apartado.

El caso de código postal se analizó detenidamente cuál era la forma más práctica de trabajarlo, ya que podría ir incluido en dirección pero complicaría la consulta de los datos. Por lo que manejaremos el código postal como una entidad que servirá de catálogo para identificar la zona geográfica de residencia del alumno a través de una consulta sobre la base de datos.

Al grupo de información anteriormente mencionada se le denominará como *información extra de la entidad*, y nos servirá para conformar las relaciones entre entidades y sus cardinalidades en el próximo apartado.

Nuestra siguiente entidad a crear es CÓDIGO POSTAL, recientemente explicado la razón de su creación.

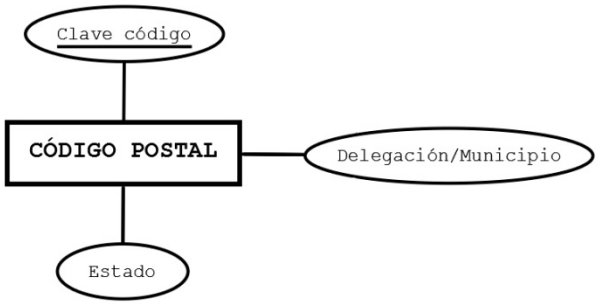


Imagen 3.3 Entidad Código Postal

Debido a que se trata de una entidad nueva de la cual no se ha hablado, se hará un análisis breve de lo que es y para lo que servirá el código postal.

El código postal es un esquema que asigna a distintas zonas o lugares de un país; un código que, adosado a la dirección, sirve para facilitar la ubicación de una zona geográfica. En México un código se compone por una cifra de cinco dígitos, donde los dos primeros indican la delegación o municipio de un estado[29].

Para el propósito del diagrama entidad relación no es necesario contar con un catálogo completo de los códigos postales existentes en todo el país, así que se limitará la lista de códigos a los alrededores de la sede de estudios.

Así, se puede observar la figura 3.3 donde se muestra la entidad a ocupar, conformada de tres atributos propios los cuales son: *Clave código*, *Delegación/Municipio* y *Estado*. La llave primaria será *Clave código*, que es la cifra de cinco dígitos que compone al código postal.

Un ejemplo de cómo se verían los datos de esta entidad es el siguiente:

Clave código: 09240
Delegación/Municipio: Iztapalapa
Estado: Ciudad de México

La siguiente entidad a crear es ACADÉMICO.

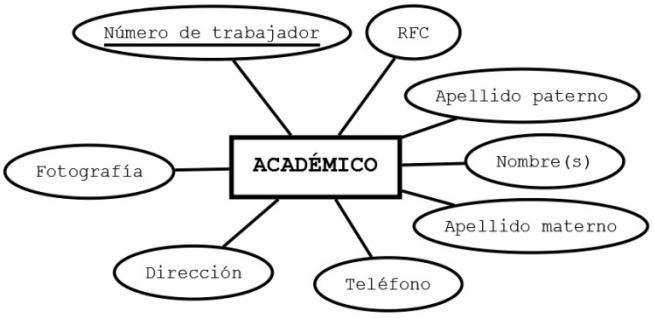


Imagen 3.4 Entidad académico

Esta entidad está encargada de almacenar la información personal de cada académico en base a los atributos que la componen los cuales son: *Apellido paterno*, *Apellido materno*, *Nombre(s)*, *RFC*, *Número de trabajador*, *Fotografía*, *Dirección* y *Teléfono*.
Se tomará como llave primaria a *Número de trabajador*, ya que no existen dos números de trabajadores iguales, permitiendo hacer a la entidad identificable a través de este atributo.

La entidad DIVISIÓN se formará de la siguiente manera:

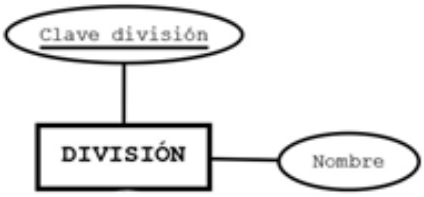


Imagen 3.5 Entidad división

Se conforma de dos atributos propios como lo son: *Clave división* y *Nombre*. Tomaremos el primero como llave primaria ya que es un atributo único para cada conjunto de información que forma a cada división.

La entidad COORDINACIÓN se construirá de la siguiente manera.

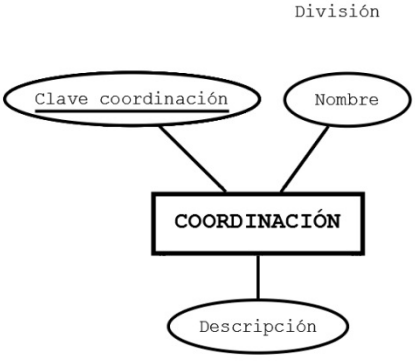


Imagen 3.6 Entidad coordinación

Esta entidad cuenta con tres atributos propios, de los que tomaremos *Clave coordinación* como llave primaria.

En cuanto a información extra de la entidad, se debe contemplar la *división* a la que cada una pertenece.

La siguiente entidad a crear será DEPARTAMENTO, la cual se formará de la siguiente manera.

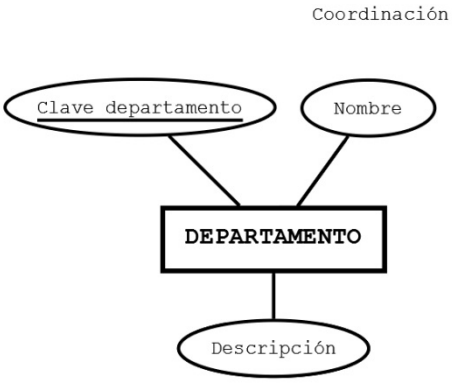


Imagen 3.7 Entidad departamento

Esta entidad es muy similar a la anterior, ya que cuenta con los mismos atributos pero aplicados a su tópico. Se tomará el atributo *Clave departamento* como llave primaria debido a ser único para cada departamento.

En la información extra se contempla que hay que conocer la *coordinación* a la que pertenece cada departamento.

La siguiente a entidad a crear es ASIGNATURA.

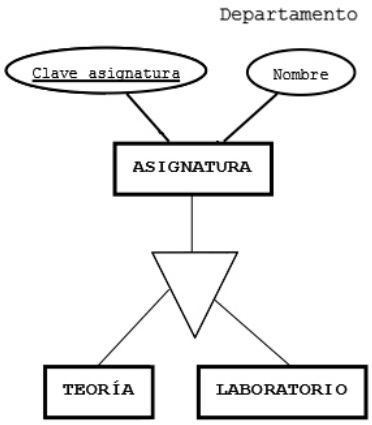


Imagen 3.8 Entidad asignatura

Como se puede observar, para esta entidad se hace uso del diagrama entidad relación extendido, ya que es necesario poder identificar si una asignatura es del tipo *teoría* o en su defecto *laboratorio*. Para ambas se utilizan los atributos *Clave asignatura* y *Nombre*, siendo el primero el elegido para funcionar como llave primaria.

Se debe considerar que independientemente de que la asignatura sea del tipo *teoría* o *laboratorio* se tiene que conocer a qué *departamento* pertenece.

Para la entidad CARRERA se creará lo siguiente.

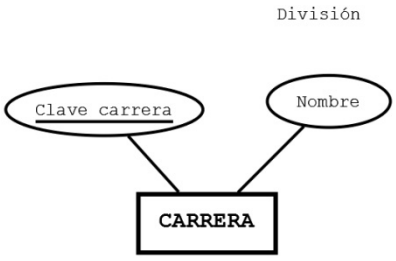


Figura 3.9 Entidad carrera

Esta entidad sólo cuenta con dos atributos propios, de los cuales utilizaremos *Clave carrera* como llave primaria ya que cada carrera tiene una clave única.

En el tema de la información extra, se deberá conocer a qué *división* pertenece cada carrera.

La entidad PLAN ESTUDIOS se conformará de la siguiente manera.

Figura 3.10 Entidad plan estudios

Como se puede observar, esta entidad cuenta con cinco atributos propios que son*: Clave plan*, *Año*, *Descripción*, *Créditos optativos* y *Créditos obligatorios*

El atributo que servirá de llave primaria para identificar a la entidad será *Clave plan*. La información recolectada de los atributos *Créditos obligatorios* y *Créditos optativos* servirá para conocer los créditos totales con los que cuenta cada plan de estudios a través de la suma de ambos atributos.

Para la cuestión de información extra, se tienen múltiples consideraciones a tener en cuenta como lo son: la *carrera* a la que pertenece el plan, las *asignaturas* con las que cuenta cada plan, los *créditos* de cada una de estas asignaturas, el *semestre* en que se debe tomar la asignatura, el *tipo* y *categoría* de cada asignatura, las *horas prácticas*, *teóricas* y *totales*, de existir *seriación*, conocer la misma y finalmente los *módulos de salida* de cada plan y asignaturas que contempla cada módulo.

Continuando con la construcción de la entidad SALÓN se tiene lo siguiente.

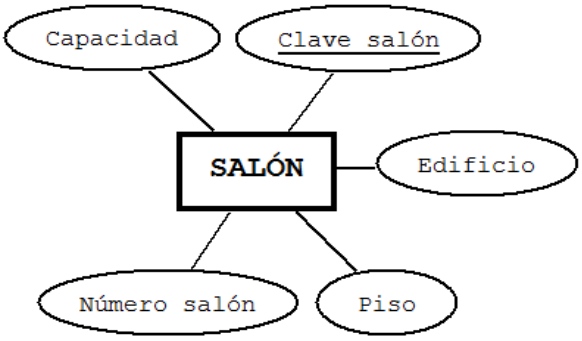


Imagen 3.11 Entidad salón

La entidad cuenta con cinco atributos, los cuales son: *Clave salón*, *Edificio*, *Piso*, *Número salón* y *Capacidad*. Elegiremos como llave primaria el atributo *Clave salón* debido a que no existen dos salones con la misma clave.

Se puede observar que existe un conjunto de atributos que se podrían agrupar en atributos del tipo derivados como lo serían localización. Se dejan los atributos por separado para ajustarse a las reglas de normalización y contar sólo con atributos atómicos.

Continuando con la construcción de entidades, toca el turno de ASIGNACIÓN.

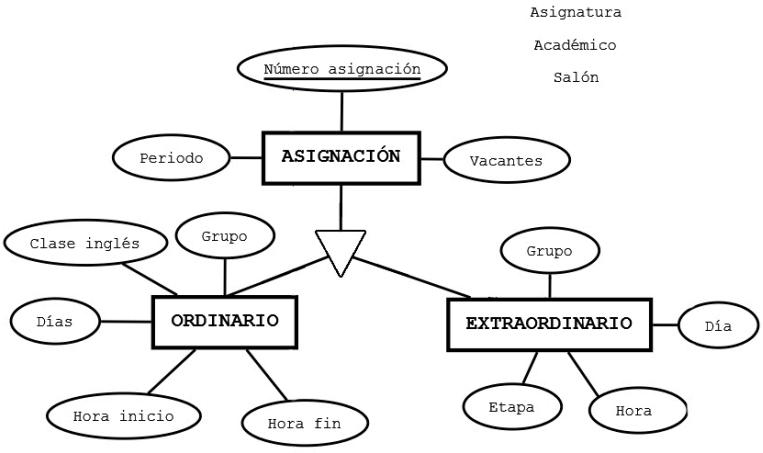


Imagen 3.12 Entidad asignación

Esta entidad también utiliza los conceptos del diagrama entidad relación extendido, que lo se utilizará para identificar si la asignación es *ordinaria* o *extraordinaria*, permitiendo así que compartan atributos en común y que cada una tenga sus atributos propios.

Los atributos que comparten ambos casos de asignación son: *Número de asignación*, *Periodo* y *Vacantes*, siendo *Número de asignación* el atributo que se utilizará como llave primaria.

Los atributos que pertenecen a ORDINARIO son: *Grupo*,
Días, Hora inicio, Hora fin y *Clase en inglés*.

Los atributos de la entidad EXTRAORDINARIO son: *Grupo*, *Día, Hora* y *Etapa*.

Por último, en cuestión de información extra que deberá contemplar la entidad asignación es respecto a *asignatura* a asignar, el *salón* en donde se impartirá la clase o el examen extraordinario y el *académico* encargado de la tarea.

Finalmente, se tiene la entidad MÓDULO SALIDA que se formará como sigue.

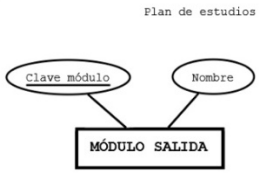


Figura 3.13 Entidad módulo salida

Esta entidad cuenta solo con dos atributos propios, de los que tomaremos *Clave módulo* como llave primaria.

En la información extra se debe considerar a qué *plan de estudios* pertenece cada módulo.

Cabe mencionar que las entidades tratadas a lo largo de esta sección es una primera aproximación para el diseño de la base de datos por lo que en el transcurso del diseño sufrirán ciertas modificaciones que ayuden a optimizar el diseño, así como la creación de nuevas entidades que auxilien a las ya mencionadas.

3.3 Relaciones entre entidades y cardinalidad

Una vez que se han creado las entidades del sistema con sus respectivos atributos y llaves primarias, procede relacionar dichas entidades para modelar una parte del mundo real con el diagrama entidad relación.

Recordando un poco lo visto en el capítulo 1 con respecto a la nomenclatura que se utiliza para representar las relaciones, tenemos que se utiliza un rombo que es conectado a través de líneas con las entidades. Para saber la cardinalidad que existe en la relación se utilizan flechas para indicar de qué tipo (uno a uno, uno a muchos, muchos a muchos) es como se muestra en la figura 3.14.

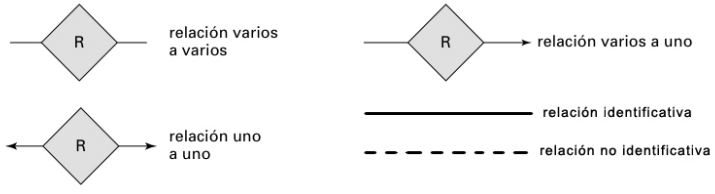


Imagen 3.14 Nomenclatura relaciones

Como también se observa en la imagen 3.14 se puede identificar si una relación es identificativa o no por el tipo de línea que conecta a la relación.

Una vez que se ha recordado la nomenclatura utilizada, procederemos a mostrar las relaciones entre las entidades estudiadas en el apartado anterior.

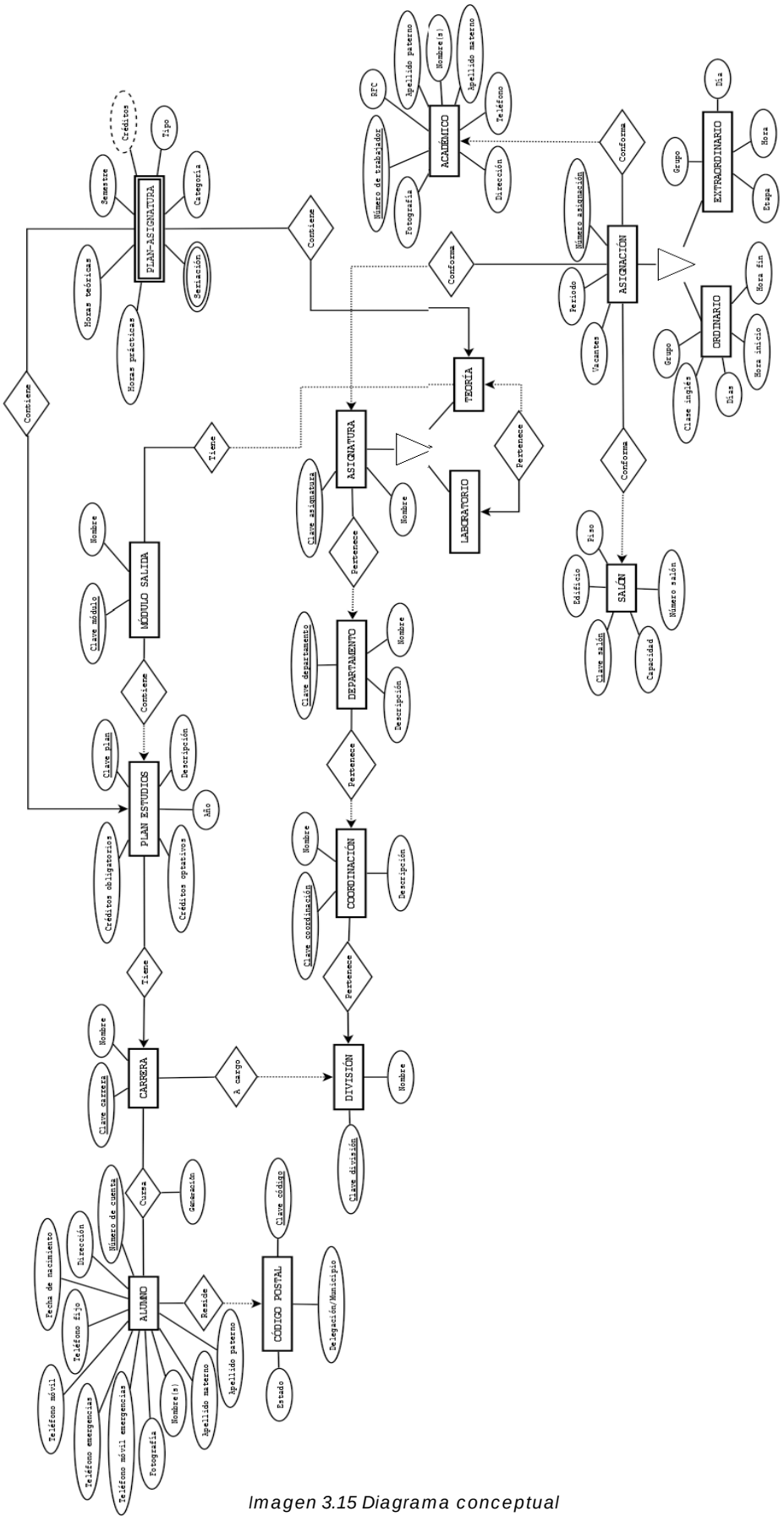


Imagen 3.15 Diagrama conceptual

Para trazar las relaciones mostradas en la figura 3.15 se tomó la llamada "*información extra"* de cada entidad.

Por ejemplo para alumno es necesario saber a qué carrera pertenece cada alumno así como también en que generación inició esta carrera. Por lo que se tomará esa información para relacionar la entidad ALUMNO con CARRERA y poniendo la generación como un atributo de la relación. Se observa que es una relación con cardinalidad *muchos a muchos,* ya que una carrera puede tener muchos alumnos y un alumno puede inscribir multiples carreras. Esto considerando el caso de que está permitido cursar carreras simultaneas.

En el mismo ejemplo de la entidad ALUMNO se desea saber qué código postal tiene su residencia, por lo que exige una relación entre estas entidades con una cardinalidad *uno a muchos* ya que un alumno sólo puede tener un código postal mientras que un código postal puede pertenecer a muchos alumnos o inclusive a ninguno.

El razonamiento utilizado en las relaciones pertenecientes a la entidad ALUMNO, es utilizado para el resto de las entidades y su *información extra,* por lo que a continuación se explicará las relaciones de casos particulares en los que sea necesario profundizar la explicación.

A través de las relaciones se ordena la jerarquía que existe entre las entidades: DIVISIÓN, COORDINACIÓN, DEPARTAMENTO y ASIGNATURA, que se puede resumir como: una división tiene a su cargo una o muchas coordinaciones mientras una coordinación debe pertenecer a una sola división; una coordinación puede tener a su cargo ninguno o muchos departamentos mientras que un departamento debe pertencer a una sola coordinación; un departamento puede tener a su cargo ninguna o muchas asignaturas mientras que una asignatura debe pertenecer a un solo departamento.

Una cuestión más a destacar es la creación de la *entidad débil* PLAN-ASIGNATURA cuya existencia depende de las entidades PLAN ESTUDIOS y TEORÍA. La entidad débil tiene sus propios atributos que son obtenidos de la relación entre las dos entidades fuertes. Como ejemplo podemos decir que depende de la asignatura y el plan al que ésta pertenece se determina a qué semestre pertenece, el número de horas prácticas y teóricas, la categoría y el tipo. A esta entidad se le agrega también un atributo multivaluado que indicarán la seriación de la asignatura de acuerdo al plan al que pertenezca.

En el caso anterior se utiliza la entidad TEORÍA en lugar de ASIGNATURA debido a que la relación existente entre TEORÍA y LABORATORIO nos permite tomar solo los datos de una de ellas para especificar los atributos de la nueva entidad. Como se puede observar se cuenta con un atributo de tipo calculable, el cual es el número de créditos totales que puede ser obtenido mediante la siguiente ecuación:

*CRÉDITOS=(HORAS_TEÓRICAS*2)+(HORAS_PRÁCTICAS)*

Por último se explicará la relación entre la entidad MÓDULO SALIDA y TEORÍA. Es una relación muchos a muchos debido a que un módulo de salida tiene que tener muchas asignaturas mientras que una asignatura puede pertenecer a ningún o a muchos módulos de salida. El caso de por qué se relaciona la entidad TEORÍA en lugar de ASIGNATURA es la misma situación que en el caso anterior y esto es debido a la existencia de la relación entre TEORÍA y LABORATORIO.
De esta forma se tiene una primera aproximación al diagrama entidad relación. El paso siguiente es definir los tipos de datos de cada atributo así como su extensión, para

después pasar las relaciones que ya se tienen a un módelo lógico que permitirá codificar el diagrama entidad relación a lenguaje SQL.

3.4 Tipos de datos

Una vez trazada la primera aproximación al diagrama entidad relación de la base de datos, sigue asignar un tipo de dato y extensión para cada atributo de las entidades.

La elección del tipo de dato es un proceso de máxima importancia en el manejo de la memoria, ya que dependiendo del tipo de dato elegido y su extensión, tendrá una repercusión directa en el rendimiento de la base de datos.

El tipo de dato sirve también como una restricción del valor que cada atributo represente. Por ejemplo, si el atributo *fecha_nac* se le asigna un tipo de dato *DATE*, es imposible que se ingrese algún valor que no corresponda con una fecha, asegurando así, que no existirá un campo ilegible para este atributo.

Para los atributos a manejar se seguirá tomando en cuenta que será una nomenclatura que se transportará a código SQL, por lo que se prescindirá de acentos y espacios. A continuación se muestra una relación del nombre del atributo en el mundo real del atributo y el nombre que el mismo tendrá para ser utilizado en el código.

Tabla 3.1 Atributos

ENTIDAD	ATRIBUTO	
	NOMBRE REAL	**NOMBRE CÓDIGO**
ALUMNO	Número de cuenta	no_cta
	Apellido paterno	ap_paterno
	Apellido materno	ap_materno
	Nombre	nombre
	Fecha de nacimiento	fecha_nac
	Teléfono fijo	telefono_fijo
	Teléfono móvil	telefono_movil
	Teléfono de emergencias	telefono_emerg
	Teléfono móvil de emergencias	telefono_movil_emerg
	Fotografía	fotografia
	Dirección	direccion
	Correo electrónico	correo_electrónico
	Clave código	clave_codigo
CÓDIGO POSTAL	Clave código	clave_codigo
	Municipio o delegación	mun_delegacion
	Estado	estado
CARRERA	Clave carrera	clave_carrera
	Nombre carrera	nombre
	Clave división	clave_division
ALUMNO-CARRERA	Generación de ingreso	generacion
	Número de cuenta	no_cta
	Clave carrera	clave_carrera
DIVISIÓN	Clave división	clave_division
	Nombre división	nombre
COORDINACIÓN	Clave coordinación	clave_coordinacion
	Nombre coordinación	nombre
	Descripción	descripcion
	Clave división	clave_division
DEPARTAMENTO	Clave departamento	clave_depto
	Nombre departamento	nombre

	Descripción	descripcion
	Clave coordinación	clave_coordinacion
ASIGNATURA	Clave asignatura	clave_asignatura
	Nombre asignatura	nombre
	Clave departamento	clave_depto
TEORÍA	Clave teoría	clave_teoria
LABORATORIO	Clave laboratorio	clave_laboratorio
	Clave teoría	clave_teoria
PLAN DE ESTUDIOS	Clave plan de estudios	clave_plan
	Año de validación	anio
	Descripción	descripcion
	Créditos optativos	creditos_opt
	Créditos obligatorios	creditos_obli
	Clave carrera	clave_carrea
PLAN-ASIGNATURA	Clave plan de estudios	clave_plan
	Clave teoría	clave_teoria
	Semestre	semestre
	Tipo	tipo
	Laboratorio separado	lab_separado
	Número de horas teóricas	horas_teoricas
	Número de horas prácticas	horas_practicas
	Créditos totales	creditos
MÓDULO DE SALIDA	Clave módulo de salida	clave_modulo
	Nombre módulo de salida	nombre
	Clave plan de estudios	clave_plan
MÓDULO-ASIGNATURA	Clave módulo de salida	clave_modulo
	Clave teoría	clave_teoria
SALÓN	Clave salón	clave_salon
	Edificio	edificio
	Número de piso	piso
	Número de salón	no_salon
	Capacidad salón	capacidad
ACADÉMICO	Número de trabajador	no_trbajador
	Registro Federal de Contribu-yentes	rfc
	Apellido paterno	ap_paterno
	Apellido materno	ap_materno
	Nombre	nombre
	Dirección	direccion
	Teléfono	telefono
	Fotografía	fotografia
	Correo electrónico	correo_electronico
ASIGNACIÓN	Clave asignación	clave_asignacion
	Periodo asignación	periodo
	Vacantes en el grupo	vacantes
	Clave asignatura	clave_asignatura
	Número de trabajador	no_trabajador
	Clave salón	clave_salon
ASIGNACIÓN ORIDNARIA	Clave asignación	clave_asignacion
	Número de grupo	grupo
	Días de clase	dias
	Horario de inicio	hora_inicio
	Horario de finalización	hora_fin
	Clases en inglés	clase_ingles
ASIGNACIÓN EXTRAORDINARIA	Clave asignación	clave_asignacion
	Número de grupo	grupo
	Día de extraordinario	dia
	Hora extraordinario	hora
	Etapa extraordinario	etapa
SERIACIÓN	Clave plan de estudios	clave_plan
	Clave teoría	clave_teo
	Clave de teoría de seriación	seriacion

A continuación se mostrará el tipo de dato de cada atributo de la base de datos, como también su extensión, un ejemplo, y de ser necesario una observación. Así también se agregará si el atributo cumple con algún constraint como lo puede ser llave primaria (PK), llave foránea (FK) y/o not null (NN).

Tabla 3.2 Atributos ALUMNO

ALUMNO							
Atributo	**Tipo de dato**	**Extensión**	**Ejemplo**	**PK**	**FK**	**NN**	**Observaciones**
no_cta	VARCHAR	10	305192563	✓	-	✓	---
ap_paterno	VARCHAR	15	López	-	-	✓	---
ap_matero	VARCHAR	15	Martínez	-	-	✓	---
nombre	VARCHAR	30	Arturo	-	-	✓	---
fecha_nac	DATE	N/A	25-DEC-1990	-	-	✓	Formato utilizado en SQL
telefono_fijo	VARCHAR	10	58246596	-	-	-	---
telefono_movil	VARCHAR	10	5517172931	-	-	-	---
telefono_emerg	VARCHAR	10	28325896	-	-	-	---
telefono_movil_emerg	VARCHAR	10	5585946321	-	-	-	---
fotografia	LONGBLOB	N/A	(Archivo de imagen)	-	-	-	---
direccion	VARCHAR	200	Av. Benito Juárez No.159 Col. Progreso	-	-	✓	---
correo_electrónico	VARCHAR	50	ejemplo@fi.unam	-	-	-	---
clave_codigo	CHAR	5	09240	-	✓	✓	---

Tabla 3.3 Atributos CODIGO_POSTAL

CODIGO_POSTAL							
Atributo	**Tipo de dato**	**Extensión**	**Ejemplo**	**PK**	**FK**	**NN**	**Observaciones**
clave_codigo	CHAR	5	09240	✓	-	✓	---
mun_delegacion	VARCHAR	40	Iztapalapa	-	-	✓	---
estado	VARCHAR	20	Distrito Federal	-	-	✓	D.F. y Edo. Mex. Contienen 16 caracteres

Tabla 3.4 Atributos CARRERA

CARRERA							
Atributo	**Tipo de dato**	**Extensión**	**Ejemplo**	**PK**	**FK**	**NN**	**Observaciones**
clave_carrera	CHAR	3	110	✓	-	✓	---
nombre	VARCHAR	50	Ingeniería en computaci-ón	-	-	✓	La carrera con mayor número de caracteres tiene 34 (Ing. Eléctrica y Electrónica)
clave_division	CHAR	4	2100	-	✓	✓	---

Tabla 3.5 Atributos ALUMNO_CARRERA

ALUMNO_CARRERA							
Atributo	**Tipo de dato**	**Extensión**	**Ejemplo**	**PK**	**FK**	**NN**	**Observaciones**
generacion	INT	N/A	2008	-	-	✓	---

no_cta	VARCHAR	10	305192563	-	✓	✓	---
clave_carrera	CHAR	3	110	-	✓	✓	---

Tabla 3.6 Atributos DIVISION

DIVISION							
Atributo	**Tipo de dato**	**Extensión**	**Ejemplo**	**PK**	**FK**	**NN**	**Observaciones**
clave_division	CHAR	4	2100	✓	-	✓	---
nombre	VARCHAR	60	División de Ciencias Básicas	-	-	✓	La división con mayor número de caracteres tiene 47 (DICT)

Tabla 3.7 Atributos COORDINACION

COORDINACION							
Atributo	**Tipo de dato**	**Extensión**	**Ejemplo**	**PK**	**FK**	**NN**	**Observaciones**
clave_coordinacion	CHAR	4	2110	✓	-	✓	---
nombre	VARCHAR	70	Coordinación de matemáticas	-	-	✓	La coordinación con mayor número de caracteres tiene 62 (Coord. De apoyo docente y difusión cultural)
descripcion	VARCHAR	100	---	-	-	-	---
clave_division	CHAR	4	2100	-	✓	✓	---

Tabla 3.8 Atributos DEPARTAMENTO

DEPARTAMENTO							
Atributo	**Tipo de dato**	**Extensión**	**Ejemplo**	**PK**	**FK**	**NN**	**Observaciones**
clave_depto	CHAR	4	2111	✓	-	✓	---
nombre	VARCHAR	100	Departamento de Cálculo Diferencial	-	-	✓	El departamento con mayor número de caracteres tiene 81 (Depto. de planeación, investigación de operaciones e ingeniería industrial)
descripcion	VARCHAR	100	---	-	-	-	---
clave_coordinacion	CHAR	4	2110	-	✓	✓	---

Tabla 3.9 Atributos ASIGNATURA

ASIGNATURA							
Atributo	**Tipo de dato**	**Extensión**	**Ejemplo**	**PK**	**FK**	**NN**	**Observaciones**
clave_asignatura	CHAR	4	1108	✓	-	✓	---
nombre	VARCHAR	110	Cálculo Diferencial	-	-	✓	La asignaturacon mayor número de caracteres tiene 90 (Temas selectos de filosofía de la ciencia y la tecnología: ciencia, tecnología y sociedad)
clave_depto	CHAR	4	2111	-	✓	✓	

Tabla 3.10 Atributos TEORIA

TEORIA							
Atributo	**Tipo de dato**	**Extensión**	**Ejemplo**	**PK**	**FK**	**NN**	**Observaciones**
clave_teoria	CHAR	4	1109	✓	✓	✓	---

Tabla 3.11 Atributos LABORATORIO

LABORATORIO							
Atributo	**Tipo de dato**	**Extensión**	**Ejemplo**	**PK**	**FK**	**NN**	**Observaciones**
clave_laboratorio	CHAR	4	4109	✓	✓	✓	---
clave_teoria	CHAR	4	1109	–	✓	✓	---

Tabla 3.12 Atributos PLAN_ESTUDIOS

PLAN_ESTUDIOS							
Atributo	**Tipo de dato**	**Extensión**	**Ejemplo**	**PK**	**FK**	**NN**	**Observaciones**
clave_plan	CHAR	5	10110	✓	-	✓	Primeros dos dígitos son el año. Siguientes tres dígitos la clave de la carrera(algoritmo propuesto)
anio	INT	N/A	2010	-	-	✓	---
descripcion	VARCHAR	100	Plan validado	-	-	-	---
creditos_opt	INT	N/A	48	-	-	✓	---
creditos_obli	INT	N/A	300	-	-	✓	---
clave_carrea	CHAR	3	110	-	✓	✓	---

Tabla 3.13 Atributos PLAN_ASIGNATURA

PLAN_ASIGNATURA							
Atributo	**Tipo de dato**	**Extensión**	**Ejemplo**	**PK**	**FK**	**NN**	**Observaciones**
clave_plan	CHAR	5	10110	✓	✓	✓	---
clave_teoria	CHAR	4	1108	✓	✓	✓	---
semestre	INT	N/A	1	-	-	✓	---
tipo	BIT	N/A	1	-	-	✓	0=Optativa; 1=Obligatoria
lab_separado	BIT	N/A	0	-	-	✓	0=No; 1=Sí
horas_teoricas	FLOAT	N/A	4.5	-	-	✓	---
horas_practicas	FLOAT	N/A	0	-	-	✓	---
creditos	FLOAT	N/A	9	-	-	✓	Se calculará con la siguiente ecuación CRÉDITOS=(HORAS_TEÓRICAS*2) +(HORAS_PRÁCTICAS)

Tabla 3.14 Atributos MODULO_SALIDA

MODULO_SALIDA							
Atributo	**Tipo de dato**	**Extensión**	**Ejemplo**	**PK**	**FK**	**NN**	**Observaciones**
clave_modulo	CHAR	4	1193	✓	-	✓	---
nombre	VARCHAR	70	Bases de datos	-	-	✓	El módulo mayor número de caracteres tiene 52 (Tecnología de radiofrecuencia óptica y microondas)
clave_plan	CHAR	5	10110	-	✓	✓	---

Tabla 3.15 Atributos MODULO_ASIGNATURA

MODULO_ASIGNATURA							
Atributo	**Tipo de dato**	**Extensión**	**Ejemplo**	**PK**	**FK**	**NN**	**Observaciones**
clave_modulo	CHAR	4	1193	✓	✓	✓	---
clave_teoria	CHAR	4	0608	✓	✓	✓	---

Tabla 3.16 Atributos SALON

SALON							
Atributo	**Tipo de dato**	**Extensión**	**Ejemplo**	**PK**	**FK**	**NN**	**Observaciones**
clave_salon	CHAR	4	A201	✓	-	✓	---
edificio	CHAR	1	A	-	-	✓	---
piso	INT	N/A	2	-	-	✓	---
no_salon	INT	N/A	01	-	-	✓	---
capacidad	INT	N/A	40	-	-	✓	---

Tabla 3.17 Atributos ACADEMICO

ACADEMICO							
Atributo	**Tipo de dato**	**Extensión**	**Ejemplo**	**PK**	**FK**	**NN**	**Observaciones**
no_trbajador	CHAR	6	435985	✓	-	✓	---
rfc	VARCHAR	12	JUGM701128	-	-	✓	Puede ser de 10 ó 12 digitos
ap_paterno	VARCHAR	15	Juárez	-	-	✓	---
ap_materno	VARCHAR	15	García	-	-	✓	---
nombre	VARCHAR	30	María Luisa	-	-	✓	---
direccion	VARCHAR	200	Av. López Mateos No.856 Col. Nacional	-	-	✓	---
telefono	VARCHAR	10	5521324525	-	-	–	---
fotografia	LONGBLOB	N/A	(Archivo de imagen)	-	-	–	---
correo_electronico	VARCHAR	50	profe@fi.unam.mx	-	-	–	---

Tabla 3.18 Atributos ASIGNACION

ASIGNACION							
Atributo	**Tipo de dato**	**Extensión**	**Ejemplo**	**PK**	**FK**	**NN**	**Observaciones**
clave_asignacion	CHAR	10	1521108030	✓	-	✓	Primeros 3 dígitos representan el periodo; siguientes 4 dígitos representan la asignatura; siguientes 2 dígitos representan número de grupo; último dígito representa la etapa de extraoridnario, en caso de ser ordinario se asigna un cero(algoritmo propuesto).
periodo	CHAR	3	152	-	-	✓	Se utiliza los últimos dos dígitos del año concatenados con la etapa del periodo
vacantes	INT	N/A	40	-	-	✓	---
clave_asignatura	CHAR	4	1108	-	✓	✓	---
no_trabajador	CHAR	6	435985	-	✓	✓	---
clave_salon	CHAR	4	A201	-	✓	✓	---

Tabla 3.19 Atributos ORDINARIO

ORDINARIO							
Atributo	**Tipo de dato**	**Extensión**	**Ejemplo**	**PK**	**FK**	**NN**	**Observaciones**
clave_asignacion	CHAR	10	1521108030	✓	✓	✓	---
grupo	INT	N/A	03	-	-	✓	---
dias	BOOLEAN	N/A	101010	-	-	✓	Cada bit representa un día de la semana(Lunes-Sábado). 0=No hay clase; 1=Sí hay clase
hora_inicio	TIME	N/A	7:00	-	-	✓	---
hora_fin	TIME	N/A	8:30	-	-	✓	---
clase_ingles	BIT	N/A	1	-	-	✓	0=No; 1=Sí

Tabla 3.20 Atributos EXTRAORDINARIO

EXTRAORDINARIO							
Atributo	**Tipo de dato**	**Extensión**	**Ejemplo**	**PK**	**FK**	**NN**	**Observaciones**
clave_asignacion	CHAR	10	1521108031	✓	✓	✓	---
grupo	INT	N/A	01	-		✓	---
dia	DATE	N/A	2015-01-15	-	-	✓	Formato usado en SQL
hora	TIME	N/A	14:00	-	-	✓	---
etapa	INT	N/A	2	-	-	✓	---

Tabla 3.21 Atributos SERIACION

SERIACION							
Atributo	**Tipo de dato**	**Extensión**	**Ejemplo**	**PK**	**FK**	**NN**	**Observaciones**
clave_plan	CHAR	5	10110	-	✓	✓	---
clave_teo	CHAR	4	1207	-	✓	✓	---
seriacion	CHAR	4	1108	-	–	–	Clave de asignatura que precede la seriación

Para la definición de cada tipo de dato, se realizó un análisis de cuál era la mejor elección en cuanto a tipo y extensión. Se tomó como base los tipos de datos que permite utilizar el lenguaje SQL:2012, su última versión estable.

Es importante mencionar que los tipos de datos se están asignando en un ambiente general del lenguaje SQL, pero podrían variar en cada manejador de base de datos. En caso de que el tipo de dato no exista en el manejador a utilizar, puede ser remplazado por uno similar que se adapte a las necesidades del dato.

3.5 Diagrama entidad relación

Hasta ahora se ha construido las relaciones entre entidades lo que ya puede ser considerado un diagrama entidad relación a nivel *conceptual*, pero recordando que el objetivo es crear un mapa para llegar al modelo relacional, se tiene que pasar de una forma *conceptual* a una *lógica*, la cual será el paso previo para la codificación de la base de datos siguiendo un modelo relacional. Para esto es importante seguir algunas reglas[30] para la transformación.

**Todo tipo de relación m:m se transformará en una nueva entidad y heredará las llaves primarias, de las entidades involucradas, como llaves foráneas; mientras que los atributos de la relación quedarán como atributos de la nueva entidad.*

**Para las relaciones* 1:m *se realizará lo que se denomina como propagación de llaves. Esto quiere decir que la llave primaria de la entidad con cardinalidad* 1 *pasará como llave foránea de la entidad con cardinalidad* m.

**Las relaciones* 1:1 *es un caso particular de una relación* 1:m*, por lo que se puede aplicar las dos opciones ya comentadas: crear una nueva entidad o realizar propagación de llaves.*

**No se permiten atributos multivaluados, ya que conforme a la* primera forma normal (1FN)*, los atributos deberán ser atómicos. Por consiguiente se creará una nueva entidad cuyos únicos atributos (y llave primaria) será la concatenación de la llave primaria de la entidad original y el atributo multivaluado.*

**Según la 3FN no se debe de contar con atributos no llave que dependan de otros atributos no llave, es el caso de los atributos de tipo calculable que no dependen de atributos llave. En un panorama real es complicado de cumplir estrictamente esta regla debido a que las necesidades que se presentan exigen incumplir de cierta forma la norma. Se tratará de reducir, en la medida de lo posible, los atributos de tipo calculable.*

Una vez aplicadas estas reglas de transformación obtendremos el diagrama entidad relación en su forma *lógica* cuya representación será distinta a la *conceptual*.

Otro asunto importante a tomar en cuenta es que a través del diagrama a trazar, se generará el código SQL para nuestra base de datos, por lo que hay que tener en cuenta que para programar no es posible utilizar caracteres como espacios y acentos por lo que siendo el diagrama el paso previo a la codificación se aplicará la omisión de caracteres no reconocidos por lenguaje máquina.

Revisada la terminología que el diagrama en forma lógica utilizará, se presenta en la figura 3.16 el diagrama entidad relación de la base de datos del *Sistema Integral de Información de la Facultad de Ingeniería*.

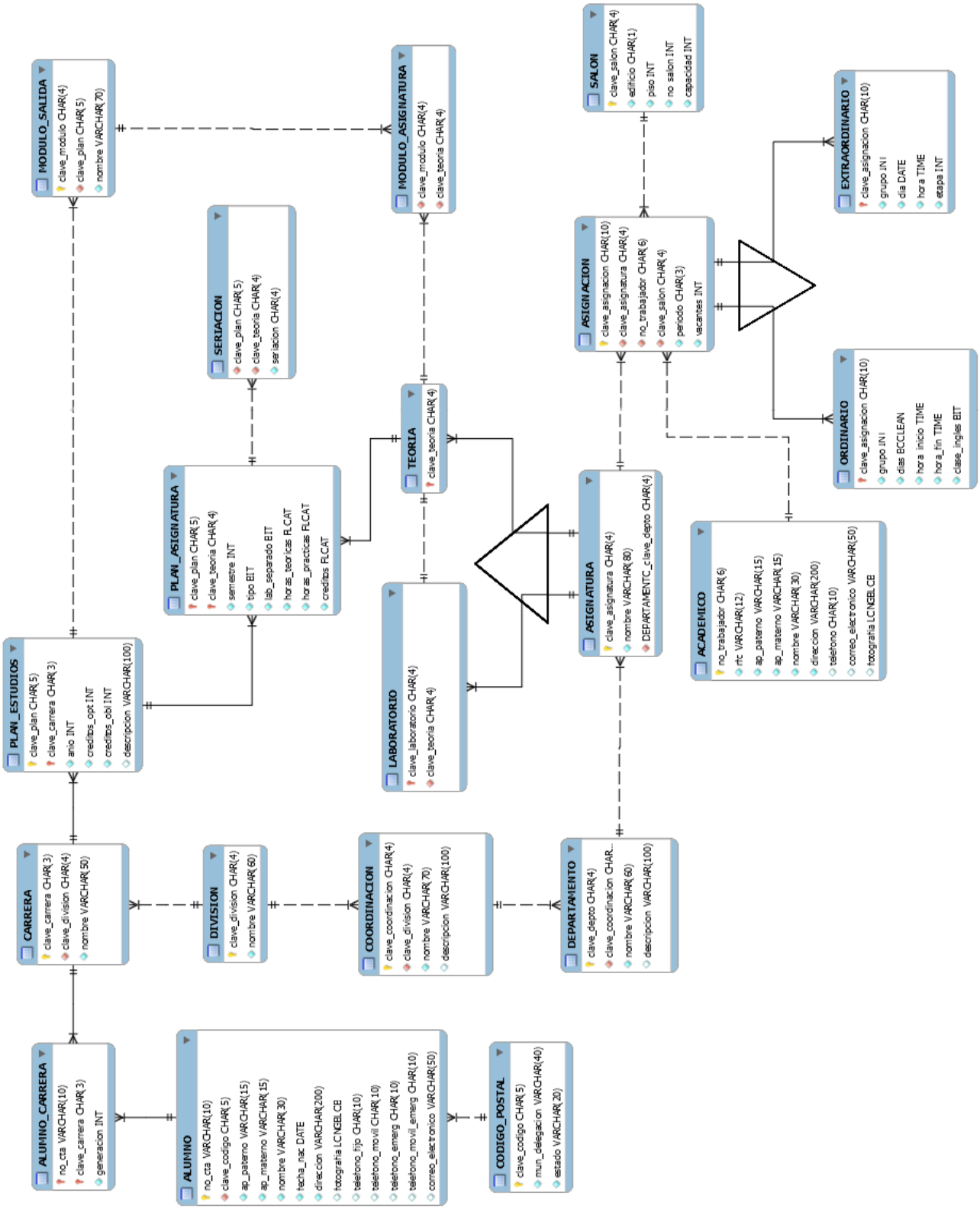
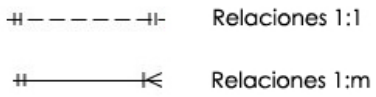


Imagen 3.16 DER

Como es notable, existen cambios en la forma de representar las relaciones y los tipos de atributos. Comenzando con las relaciones se representarán de la siguiente manera (según la notacón Martin para el diagrama entidad-relación[31]).



Como se observa se sigue respetando la forma en como se representa las relaciones identificativas y no identificativas.

En cuestión de los atributos se pueden tener llaves primarias (representadas con una llave amarilla), llaves foráneas (rombos rojos), atributos simples no nulos (rombos azules) y atributos que permiten valores nulos (rombos con contorno azul y fondo blanco). La representación es la siguiente:

De esta forma tenemos un diagrama entidad relación en su forma lógica que es el paso previo para que nuestro diseño pase a ser un modelo relacional, los siguientes pasos a realizar son asuntos involucrados con el rendimiento de la base de datos a nivel programación.

3.6 Índices

Al tener una cantidad considerable de registros dentro de una base de datos, la búsqueda de un registro en específico puede llegar a ser tardada y costosa ya que tendrá que recorrer todos los registros existentes hasta encontrar el que necesita.

Los índices se utilizan para mejorar la velocidad de las operaciones, por medio de la identificación de un elemento en una fila que pertenezca a una tabla. Los índices son construidos sobre árboles B, B+, B* o sobre una mezcla de ellos. Se suelen utilizar sobre aquellos campos en los cuales se hacen búsquedas frecuentes.

Una forma automática de generar índices, es a través de los *constraints* de llave primaria y llave única, por lo cual al ya haber definido las llaves primarias sobre campos estratégicos en nuestro diseño, se cuenta ya con ciertos índices que ayudarán a mejorar la velocidad de las operaciones efectuadas sobre la base de datos.

En el caso de los *constraints* de llave foránea, los índices no se crean automáticamente, razón por la cual de ser necesarios estos índices deberán ser creados de manera manual.

Sumado a los índices que se han generado de manera automática, se propondrá la creación de otros índices para un mejor funcionamiento del diseño. Esto se hará seleccionando los campos más trabajados de cada tabla.

Tabla 3.22 Índices

Atributo	Tabla	Tipo de constraint	Modo de generación
no_cta	ALUMNO	PK	Automática
ap_paterno	ALUMNO	---	Manual
clave_codigo	CODIGO_POSTAL	PK	Automática
clave_carrera	ALUMNO_CARRERA	FK	Manual
clave_carrera	CARRERA	PK	Automática
clave_division	DIVISION	PK	Automática
clave_coord	COORDINACION	PK	Automática
clave_depto	DEPARTAMENTO	PK	Automática
clave_plan	PLAN_ESTUDIOS	PK	Automática
clave_carrera	PLAN_ESTUDIOS	FK	Manual
clave_modulo	MODULO_SALIDA	PK	Automática
nombre	MODULO_SALIDA	---	Manual
clave_asignatura	MODULO_ASIGNATURA	FK	Manual
clave_asignatura	ASIGNATURA	PK	Automática
nombre	ASIGNATURA	---	Manual
clave_lab	LABORATORIO	FK/PK	Automática
clave_teo	TEORIA	FK/PK	Automática
clave_plan	PLAN_ASIGNATURA	FK/PK	Automática
clave_teo	PLAN_ASIGNATURA	FK/PK	Automática
clave_teo	SERIACION	FK	Manual
clave_asignacion	ASIGNACION	PK	Automática
periodo	ASIGNACION	---	Manual
no_trabajador	ACADEMICO	PK	Automática
ap_paterno	ACADEMICO	---	Manual
clave_asignacion	ORDINARIO	PK	Automática
grupo	ORDINARIO	---	Automática
clave_asignacion	EXTRAORDINARIO	PK	Automática
clave_salon	SALON	PK	Automática

La sentencia SQL para creación de índices es la siguiente:

```
CREATE INDEX [NOMBRE_ÍNDICE] ON [TABLA] (columna);
```

Por ejemplo, para crear el índice al campo grupo de la tabla ORDINARIO, la sentencia sería:

```
CREATE INDEX IDX_GRUPO ON ORDINARIO (grupo);
```

De esta manera, los índices expuestos son los que se proponen en un inicio para la creación de la base de datos, mas teniendo conocimiento de que una vez en funcionamiento podría ser necesario la creación de nuevos índices para garantizar un mejor funcionamiento de la misma, se deja esta tarea al administrador de base de datos encargado del mantenimiento.

En el Anexo Técnico 3 “Índices propuestos” se pueden encontrar las sentencias para los índices propuestos.

3.7 Diseño normalizado

Para este trabajo, se genera un diseño que se encuentra normalizado hasta la tercera forma normal (3FN), por lo que retomando el apartado *1.4 Normalización*, se hará una verificación de que dichas normas sean cumplidas sobre el diseño en cuestión.

Como se puede observar en el diseño, se omitieron los atributos compuestos desde la creación de las entidades, por lo cual cada atributo es atómico. Así se puede garantizar que la 1FN se cumple en el diseño.

Respecto a la segunda normal se revisará que los atributos cumplan con la dependencia funcional.

ALUMNO

Tabla 3.23 2FN ALUMNO

Atributo	Tipo de llave	Dependencia funcional
no_cta	PK	---
ap_paterno	---	✓
ap_materno	---	✓
nombre	---	✓
fecha_nac	---	✓
fotografia	---	✓
telefono_fijo	---	✓
telefono_movil	---	✓
telefono_emerg	---	✓
telefono_mov_emerg	---	✓
direccion	---	✓
correo_electronico	---	✓
clave_codigo	FK	---

Para la entidad *ALUMNO*, todos los atributos no llave dependen de la llave primaria *no_cta* ya que todos están relacionados con la misma.

CODIGO_POSTAL

Tabla 3.24 2FN CODIGO_POSTAL

Atributo	Tipo de llave	Dependencia funcional
clave_codigo	PK	---
mun_deleg	---	✓
estado	---	✓

Para *CODIGO_POSTAL*, los atributos dependen de la llave primaria *clave_codigo*.

CARRERA

Tabla 3.25 2FN CARRERA

Atributo	Tipo de llave	Dependencia funcional
clave_carrera	PK	---
nombre	---	✓
clave_division	FK	---

Para CARRERA, su único atributo no llave depende de la llave primaria clave_carrera.

ALUMNO_CARRERA

Tabla 3.26 2FN ALUMNO_CARRERA

Atributo	Tipo de llave	Dependencia funcional
no_cta	FK	---
clave_carrera	FK	---
generacion	---	---

Al no existir llave primaria en ALUMNO_CARRERA, no hay necesidad de 1FN.

DIVISION

Tabla 3.27 2FN DIVISION

Atributo	Tipo de llave	Dependencia funcional
clave_division	PK	---
nombre	---	✓

Para DIVISION, sus atributos dependen de su llave primaria clave_division.

COORDINACION

Tabla 3.28 2FN COORDINACION

Atributo	Tipo de llave	Dependencia funcional
clave_coordinacion	PK	---
nombre	---	✓
descripción	---	✓
clave_division	FK	---

Para COORDINACION, sus atributos no llave dependen de la llave primaria clave_coordinacion.

DEPARTAMENTO

Tabla 3.29 2FN DEPARTAMENTO

Atributo	Tipo de llave	Dependencia funcional
clave_depto	PK	---
nombre	---	✓
descripción	---	✓
clave_coordinacion	FK	---

Para la tabla DEPARTAMENTO, sus atributos no llave dependen de la llave primaria clave_depto.

ASIGNATURA

Tabla 3.30 2FN ASIGNATURA

Atributo	Tipo de llave	Dependencia funcional
clave_asignatura	PK	---
nombre	---	✓
clave_depto	FK	---

En ASIGNATURA, el atributo no llave depende de la llave primaria clave_asignatura.

PLAN_ESTUDIOS

Tabla 3.31 2FN PLAN_ESTUDIOS

Atributo	Tipo de llave	Dependencia funcional
clave_plan	PK	---
anio	---	✓
descripcion	---	✓
creditos_opt	---	✓
creditos_obl	---	✓
clave_carrera	FK	---

Los atributos no llave en PLAN_ESTUDIOS son dependientes de la llave primaria clave_plan.

PLAN_ASIGNATURA

Tabla 3.32 2FN PLAN_ASIGNATURA

Atributo	Tipo de llave	Dependencia funcional
clave_plan	FK/PK	---
clave_teo	FK/PK	---
semestre	---	✓
tipo	---	✓
lab-separado	---	✓
horas_teo	---	✓
horas_prac	---	✓
creditos	---	✓

Para PLAN_ASIGNATURA, los atributos no llaves dependen de la combinación de sus llaves primarias.

MODULO_SALIDA

Tabla 3.33 2FN PLAN_ASIGNATURA

Atributo	Tipo de llave	Dependencia funcional
clave_modulo	PK	---
nombre	---	✓
clave_plan	FK	---

Los atributos no llave de MODULO_SALIDA dependen de la llave primaria clave_modulo.

SALON

Tabla 3.34 2FN SALON

Atributo	Tipo de llave	Dependencia funcional
clave_salon	PK	---
edificio	---	✓
piso	---	✓
no_salon	---	✓
capacidad	---	✓

Se observa que los atributos no llave en SALON dependen de la llave primaria clave_salon.

ACADEMICO

Tabla 3.35 2FN ACADEMICO

Atributo	Tipo de llave	Dependencia funcional
no_trabajador	PK	---
rfc	---	✓
ap_paterno	---	✓
ap_materno	---	✓
nombre	---	✓
fotografia	---	✓
telefono	---	✓
direccion	---	✓
correo_electronico	---	✓

Para ACADEMICO, los atributos no llave dependen de la llave primaria no_trabajador.

ASIGNACION

Tabla 3.36 2FN ASIGNACION

Atributo	Tipo de llave	Dependencia funcional
clave_asignacion	PK	---
periodo	---	✓
vacantes	---	✓
clave_asignatura	FK	---
no_trabajador	FK	---
clave_salon	FK	---

Los atributos no llave que conforman a ASIGNACION dependen de la llave primaria clave_asignacion.

ORDINARIO

Tabla 3.37 2FN ORDINARIO

Atributo	Tipo de llave	Dependencia funcional
clave_asignacion	FK/PK	---
grupo	---	✓
dias	---	✓
hora_inicio	---	✓
hora_fin	---	✓
clase_ingles	---	✓

Para ORDINARIO, los atributos no llave dependen de la llave primaria clave_asignacion.

EXTRAORDINARIO

Tabla 3.38 2FN EXTRAORDINARIO

Atributo	Tipo de llave	Dependencia funcional
clave_asignacion	FK/PK	---
grupo	---	✓
dia	---	✓
hora	---	✓
etapa	---	✓

Los atributos no llave para EXTRAORDINARIO, dependen de la llave primaria clave_asignacion.

TEORIA

Tabla 3.39 2FN TEORIA

Atributo	Tipo de llave	Dependencia funcional
clave_teo	FK/PK	---

Al no tener atributos no llave, no es necesaria la aplicación de la 2FN.

LABORATORIO

Tabla 3.40 2FN LABORATORIO

Atributo	Tipo de llave	Dependencia funcional
clave_lab	FK/PK	---
clave_teo	FK	---

Al no tener atributos no llave, no es necesaria la aplicación de la 2FN.

SERIACION

Tabla 3.41 2FN SERIACION

Atributo	Tipo de llave	Dependencia funcional
clave_teo	FK	---
clave_plan	FK	---
seriacion	---	✓

El atributo seriación depende de la llave primaria.

Una vez revisadas todas las tablas y sus atributos se concluye que la dependencia funcional se cumple para el diseño de la base de datos, por lo que podemos decir que el diseño se encuentra en la 2FN.

Según la 3FN no se debe de contar con atributos no llave que dependan de otros atributos no llave, es el caso de los atributos de tipo calculable que no dependen de atributos llave. En un panorama real es complicado de cumplir estrictamente esta regla debido a que las necesidades que se presentan exigen incumplir de cierta forma la norma. Se trata-

rá de reducir, en la medida de lo posible, los atributos de tipo calculable. El único atributo calculable con el que este diseño cuenta es "créditos", perteneciente a la entidad PLAN-ASIGNATURA.

Con lo anterior y comprobado el cumplimiento de las tres primeras formas normales, podemos concluir que el diseño de la base de datos se encuentra normalizado a medida de lo posible.

3.8 Consideraciones semánticas

Al trabajar únicamente el diseño de la base de datos y siendo que la puesta en práctica del diseño será llevado a cabo por personal que no estuvo involucrado en esta etapa, se tienen que tener en cuenta ciertas consideraciones semánticas del Sistema Integral de Información de la Facultad de Ingeniería. Algunas de las consideraciones semánticas a tener en cuenta son las siguientes:

- La restricción de que un alumno no pueda inscribir una asignatura que tenga otra en seriación tendrá que ser impuesta por el sistema.
- La Facultad de Ingeniería cuenta con una restricción denominada "bloque móvil" que no permite inscribir materias de determinado semestre sin que se hayan acreditado las de ciertos semestres anteriores. Esta restricción tendrá que ser impuesta por el sistema.
- La creación de índices es sólo una propuesta ya que conforme se utilice la base de datos se podrá observar que índices son necesarios y sobre que campos
- Se realiza un diseño de base de datos contemplando una normalización que considera hasta la tercera forma normal ya que son las que se consideran necesarias para un buen diseño del diagrama entidad-relación.
- Es opcional utilizar la codificación presentada en este trabajo ya que dependiendo del Sistema Manejador de Base de Datos en el que se implementará el diseño trabajado, podrá ser creada la base con script o forma gráfica.
- Los tipos de datos propuestos son generales de SQL, aunque existe el caso de que el Sistema Manejador de Base de Datos en el que se implementará el diseño trabajado nombre de diferente forma al tipo de dato o que existan limitantes en cuanto al mismo por lo que se tendrá que hacer la adaptación para los tipos de datos que se requieran.

3.9 Codificación

Una vez que ya se cuenta con nuestro diseño en el que se definen las entidades, los atributos de cada uno, tipos de datos, restricciones e índices, se prosigue a la creación de las sentencias SQL para la creación de las tablas y la definición de los índices.

La codificación se hará en dos partes: la primera será la creación de las tablas con sus respectivos atributos, los tipos de datos, restricciones not null y llaves primarias y foráneas. La segunda parte de la codificación estará referida a la creación de índices manuales.

```
-- -----------------------------------------------------
-- Tabla CODIGO_POSTAL
-- -----------------------------------------------------
CREATE TABLE CODIGO_POSTAL(
  clave_codigo CHAR(5) NOT NULL,
  mun_delegacion VARCHAR(40) NOT NULL,
  estado VARCHAR(20) NOT NULL,
  PRIMARY KEY (clave_codigo)
  );

-- -----------------------------------------------------
-- Tabla ALUMNO
-- -----------------------------------------------------
CREATE TABLE ALUMNO(
  no_cta VARCHAR(10) NOT NULL,
  clave_codigo CHAR(5) NOT NULL,
  ap_paterno VARCHAR(15) NOT NULL,
  ap_materno VARCHAR(15) NOT NULL,
  nombre VARCHAR(30) NOT NULL,
  fecha_nac DATE NOT NULL,
  direccion VARCHAR(200) NOT NULL,
  fotografia LONGBLOB NULL,
  telefono_fijo CHAR(10) NULL,
  telefono_movil CHAR(10) NULL,
  telefono_emerg CHAR(10) NULL,
  telefono_movil_emerg CHAR(10) NULL,
  correo_electronico VARCHAR(50) NULL,
  PRIMARY KEY (no_cta),
  CONSTRAINT fk_ALUMNO_CODIGO_POSTAL1
    FOREIGN KEY (clave_codigo)
    REFERENCES CODIGO_POSTAL(clave_codigo)
  );

-- -----------------------------------------------------
-- Tabla DIVISION
-- -----------------------------------------------------
CREATE TABLE DIVISION(
  clave_division CHAR(4) NOT NULL,
  nombre VARCHAR(60) NOT NULL,
  PRIMARY KEY (clave_division)
  );

-- -----------------------------------------------------
-- Tabla CARRERA
-- -----------------------------------------------------
CREATE TABLE CARRERA(
  clave_carrera CHAR(3) NOT NULL,
  clave_division CHAR(4) NOT NULL,
  nombre VARCHAR(50) NOT NULL,
  PRIMARY KEY (clave_carrera),
  CONSTRAINT fk_CARRERA_DIVISION1
    FOREIGN KEY (clave_division)
    REFERENCES DIVISION(clave_division)
      );
```

```
-- -----------------------------------------------------
-- Tabla ALUMNO_CARRERA
-- -----------------------------------------------------
CREATE TABLE ALUMNO_CARRERA(
  no_cta VARCHAR(10) NOT NULL,
  clave_carrera CHAR(3) NOT NULL,
  generacion INT NOT NULL,
  PRIMARY KEY (no_cta, clave_carrera),
  CONSTRAINT fk_ALUMNO_CARRERA_ALUMNO1
    FOREIGN KEY (no_cta)
    REFERENCES ALUMNO(no_cta),
  CONSTRAINT fk_ALUMNO_CARRERA_CARRERA2
    FOREIGN KEY (clave_carrera)
    REFERENCES CARRERA (clave_carrera)
   );

-- -----------------------------------------------------
-- Tabla COORDINACION
-- -----------------------------------------------------
CREATE TABLE COORDINACION(
  clave_coordinacion CHAR(4) NOT NULL,
  clave_division CHAR(4) NOT NULL,
  nombre VARCHAR(70) NOT NULL,
  descripcion VARCHAR(100) NULL,
  PRIMARY KEY (clave_coordinacion),
  CONSTRAINT fk_COORDINACION_DIVISION1
    FOREIGN KEY (clave_division)
    REFERENCES DIVISION(clave_division)
   );

-- -----------------------------------------------------
-- Tabla DEPARTAMENTO
-- -----------------------------------------------------
CREATE TABLE DEPARTAMENTO(
  clave_depto CHAR(4) NOT NULL,
  clave_coordinacion CHAR(4) NOT NULL,
  nombre VARCHAR(60) NOT NULL,
  descripcion VARCHAR(100) NULL,
  PRIMARY KEY (clave_depto),
  CONSTRAINT fk_DEPARTAMENTO_COORDINACION1
    FOREIGN KEY (clave_coordinacion)
    REFERENCES COORDINACION(clave_coordinacion)
   );

-- -----------------------------------------------------
-- Tabla PLAN_ESTUDIOS
-- -----------------------------------------------------
CREATE TABLE PLAN_ESTUDIOS(
  clave_plan CHAR(5) NOT NULL,
  clave_carrera CHAR(3) NOT NULL,
  anio INT NOT NULL,
  creditos_opt INT NOT NULL,
  creditos_obl INT NOT NULL,
  descripcion VARCHAR(100) NULL,
  PRIMARY KEY (clave_plan, clave_carrera),
  CONSTRAINT fk_PLAN_ESTUDIOS_CARRERA2
    FOREIGN KEY (clave_carrera)
```

```
    REFERENCES CARRERA (clave_carrera)
   );

-- -----------------------------------------------------
-- Tabla MODULO_SALIDA
-- -----------------------------------------------------
CREATE TABLE MODULO_SALIDA(
  clave_modulo CHAR(4) NOT NULL,
  clave_plan CHAR(5) NOT NULL,
  nombre VARCHAR(70) NOT NULL,
  PRIMARY KEY (clave_modulo),
  CONSTRAINT fk_MODULO_SALIDA_PLAN_ESTUDIOS1
    FOREIGN KEY (clave_plan)
    REFERENCES PLAN_ESTUDIOS(clave_plan)
   );

-- -----------------------------------------------------
-- Tabla ASIGNATURA
-- -----------------------------------------------------
CREATE TABLE ASIGNATURA(
  clave_asignatura CHAR(4) NOT NULL,
  nombre VARCHAR(45) NOT NULL,
  DEPARTAMENTO_clave_depto CHAR(4) NOT NULL,
  PRIMARY KEY (clave_asignatura),
  CONSTRAINT fk_ASIGNATURA_DEPARTAMENTO1
    FOREIGN KEY (DEPARTAMENTO_clave_depto)
    REFERENCES DEPARTAMENTO(clave_depto)
   );

-- -----------------------------------------------------
-- Tabla TEORIA
-- -----------------------------------------------------
CREATE TABLE TEORIA(
  clave_teoria CHAR(4) NOT NULL,
  PRIMARY KEY (clave_teoria),
  CONSTRAINT fk_TEORIA_ASIGNATURA1
    FOREIGN KEY (clave_teoria)
    REFERENCES ASIGNATURA(clave_asignatura)
   );

-- -----------------------------------------------------
-- Tabla LABORATORIO
-- -----------------------------------------------------
CREATE TABLE LABORATORIO(
  clave_laboratorio CHAR(4) NOT NULL,
  clave_teoria CHAR(4) NOT NULL,
  PRIMARY KEY (clave_laboratorio),
  CONSTRAINT fk_LABORATORIO_ASIGNATURA1
    FOREIGN KEY (clave_laboratorio)
    REFERENCES ASIGNATURA(clave_asignatura),
  CONSTRAINT fk_LABORATORIO_TEORIA1
    FOREIGN KEY (clave_teoria)
    REFERENCES TEORIA (clave_teoria)
   );
```

```
-- -----------------------------------------------------
-- Tabla MODULO_ASIGNATURA
-- -----------------------------------------------------
CREATE TABLE MODULO_ASIGNATURA(
  clave_modulo CHAR(4) NOT NULL,
  clave_teoria CHAR(4) NOT NULL,
  CONSTRAINT fk_MODULO_ASIGNATURA_MODULO_SALIDA1
    FOREIGN KEY (clave_modulo)
    REFERENCES MODULO_SALIDA(clave_modulo),
  CONSTRAINT fk_MODULO_ASIGNATURA_TEORIA1
    FOREIGN KEY (clave_teoria)
    REFERENCES TEORIA` (clave_teoria)
   );

-- -----------------------------------------------------
-- Tabla PLAN_ASIGNATURA
-- -----------------------------------------------------
CREATE TABLE PLAN_ASIGNATURA(
  clave_plan CHAR(5) NOT NULL,
  clave_teoria CHAR(4) NOT NULL,
  semestre INT NOT NULL,
  tipo BIT NOT NULL,
  lab_separado BIT NOT NULL,
  horas_teoricas FLOAT NOT NULL,
  horas_practicas FLOAT NOT NULL,
  creditos FLOAT NOT NULL,
  PRIMARY KEY (clave_plan, clave_teoria),
  CONSTRAINT fk_PLAN_ASIGNATURA_PLAN_ESTUDIOS1
    FOREIGN KEY (clave_plan)
    REFERENCES PLAN_ESTUDIOS(clave_plan),
  CONSTRAINT fk_PLAN_ASIGNATURA_TEORIA1
    FOREIGN KEY (clave_teoria)
    REFERENCES TEORIA(clave_teoria)
   );

-- -----------------------------------------------------
-- Tabla SERIACION
-- -----------------------------------------------------
CREATE TABLE SERIACION(
  clave_plan CHAR(5) NOT NULL,
  clave_asignatura CHAR(4) NOT NULL,
  seriacion CHAR(4) NOT NULL,
  CONSTRAINT fk_SERIACION_PLAN_ASIGNATURA1
    FOREIGN KEY (clave_plan , clave_asignatura)
    REFERENCES PLAN_ASIGNATURA(clave_plan, clave_teoria)
   );

-- -----------------------------------------------------
-- Tabla ACADEMICO
-- -----------------------------------------------------
CREATE TABLE ACADEMICO(
  no_trabajador CHAR(6) NOT NULL,
  rfc VARCHAR(12) NOT NULL,
  ap_paterno VARCHAR(15) NOT NULL,
  ap_materno VARCHAR(15) NOT NULL,
  nombre VARCHAR(30) NOT NULL,
  direccion VARCHAR(200) NOT NULL,
```

```
  telefono CHAR(10) NULL,
  correo_electronico VARCHAR(30) NULL,
  fotografia LONGBLOB NULL,
  PRIMARY KEY (no_trabajador)
  );

-- -----------------------------------------------------
-- Tabla SALON
-- -----------------------------------------------------
CREATE TABLE SALON(
  clave_salon CHAR(4) NOT NULL,
  edificio CHAR(1) NOT NULL,
  piso INT NOT NULL,
  no_salon INT NOT NULL,
  capacidad INT NOT NULL,
  computadora BIT NOT NULL,
  pizarron_electr BIT NOT NULL,
  proyector BIT NOT NULL,
  PRIMARY KEY (clave_salon)
  );

-- -----------------------------------------------------
-- Tabla ASIGNACION
-- -----------------------------------------------------
CREATE TABLE ASIGNACION(
  clave_asignacion CHAR(10) NOT NULL,
  clave_asignatura CHAR(4) NOT NULL,
  no_trabajador CHAR(6) NOT NULL,
  clave_salon CHAR(4) NOT NULL,
  periodo CHAR(3) NOT NULL,
  vacantes INT NOT NULL,
  PRIMARY KEY (clave_asignacion),
  CONSTRAINT fk_ASIGNACION_ASIGNATURA1
    FOREIGN KEY (clave_asignatura)
    REFERENCES ASIGNATURA (clave_asignatura),
  CONSTRAINT fk_ASIGNACION_ACADEMICO1
    FOREIGN KEY (no_trabajador)
    REFERENCES ACADEMICO(no_trabajador),
  CONSTRAINT fk_ASIGNACION_SALON1
    FOREIGN KEY (clave_salon)
    REFERENCES SALON(clave_salon)
  );

-- -----------------------------------------------------
-- Tabla ORDINARIO
-- -----------------------------------------------------
CREATE TABLE ORDINARIO(
  clave_asignacion CHAR(10) NOT NULL,
  grupo INT NOT NULL,
  dias BOOLEAN NOT NULL,
  hora_inicio TIME NOT NULL,
  hora_fin TIME NOT NULL,
  clase_ingles BIT NOT NULL,
  PRIMARY KEY (clave_asignacion),
  CONSTRAINT fk_ORDINARIO_ASIGNACION1
    FOREIGN KEY (clave_asignacion)
    REFERENCES ASIGNACION (clave_asignacion)
```

```
 );

-- -----------------------------------------------------
-- Tabla EXTRAORDINARIO
-- -----------------------------------------------------
CREATE TABLE EXTRAORDINARIO(
  clave_asignacion CHAR(10) NOT NULL,
  grupo INT NOT NULL,
  dia DATE NOT NULL,
  hora TIME NOT NULL,
  etapa INT NOT NULL,
  PRIMARY KEY (clave_asignacion),
  CONSTRAINT fk_EXTRAORDINARIO_ASIGNACION1
    FOREIGN KEY (clave_asignacion)
    REFERENCES ASIGNACION (clave_asignacion)
 );

 -- -----------------------------------------------------
-- Creación de índices
 --  -----------------------------------------------------
CREATE INDEX idx_ap_paterno_alumno ON ALUMNO(ap_paterno);
CREATE INDEX idx_clave_carrera_alumno_carrera ON
ALUMNO_CARRERA(clave_carrera);
CREATE INDEX idx_clave_carrera_plan ON PLAN_ESTUDIOS(clave_carrera);
CREATE INDEX idx_nombre_modulo ON MODULO_SALIDA(nombre);
CREATE INDEX idx_clave_asignatura_modulo ON MODU-
LO_SALIDA(clave_asignatura);
CREATE INDEX idx_nombre_asignatura ON ASIGNATURA(nombre);
CREATE INDEX idx_clave_teoria_seriacion ON seriacion(clave_teoria);
CREATE INDEX idx_periodo_asignacion ON ASIGNACION(periodo);
CREATE INDEX idx_ap_paterno_academico ON ACADEMICO(ap_paterno);
CREATE INDEX idx_clave_carrera_alumno_carrera ON
ALUMNO_CARRERA(clave_carrera);
```

Con estas sentencias se pueden crear las tablas que la base de datos necesita conforme al diseño propuesto. La base de datos se crea con el SGBD correspondiente y ya sobre la base creada estas sentencias pueden ser ejecutadas.

CAPÍTULO IV
Pruebas

4.1 Plan de pruebas

Con el diseño de la base de datos culminado y con la construcción realizada, procede la etapa de pruebas. Para poder realizar pruebas se necesita primeramente poblar la base de datos con registros (que pueden ser ficticios).

Para iniciar, se estructurará qué datos conformarán a cada tabla.

Para la tabla CODIGO_POSTAL, se utilizarán sólo códigos postales pertenecientes al Distrito Federal, tomando unos cuantos códigos postales de cada delegación, esto se hará al azar.

En el caso de la tabla ALUMNO, se manejarán datos enteramente ficticios para no comprometer información personal de ningún miembro de la Facultad de Ingeniería, por lo que los nombres, números de cuenta, fechas de nacimiento, direcciones, teléfonos y correos electrónicos, son inventados. Para fines de pruebas se omitirá el atributo fotografía. Cabe mencionar que la UNAM maneja un algoritmo para generar los números de cuenta, pero como en este trabajo los datos son ficticios el número de cuenta también lo será por lo que no es necesario trabajar con el algoritmo manejado por la UNAM para la creación del mismo.

Para las tablas DIVISION, CARRERA, COORDINACION y DEPARTAMENTO, se utilizarán los datos reales con los que cuenta actualmente la Facultad de Ingeniería, agregando en algunos casos entes ficticios para conservar la jerarquía existente.

Para ALUMNO_CARRERA, se tomarán los valores ficticios generados para la tabla ALUMNO, mientras que se ocuparán sólo dos carreras que serán: Ingeniería en Computación e Ingeniería Eléctrica y Electrónica. En el caso del atributo generación, se utilizará una generación en común que es la 2010.

En PLAN_ESTUDIOS, se utilizará la información de los planes 2010 para las 12 carreras con las que cuenta actualmente la Facultad de Ingeniería. Se aprovechará la opción NULL con la que cuenta el atributo descripción para no agregar ninguna información a este campo.

La tabla MODULO_SALIDA se conjunta con los planes de estudio mencionados anteriormente, siendo esta información real con los módulos que pertenezcan a una carrera en los planes 2010.

Al contar con alumnos que están inscritos a sólo dos carreras distintas y recordando la finalidad del ambiente de pruebas, la tabla ASIGNATURA sólo contará con las asignaturas correspondientes las carreras de Ingeniería en Computación e Ingeniería Eléctrica y Electrónica.

Para las tablas TEORIA y LABORATORIO, se repartirán los datos de la tabla ASIGNATURA, conforme a los planes de estudio del año 2010.

La tabla MODULO_ASIGNATURA contará con información referente a las tablas MODULO_SALIDA y ASIGNATURA para las dos carreras que hemos mapeado.

Recordando que la tabla PLAN_ASIGNATURA es una tabla que nos sirve para romper la relación muchos a muchos entre PLAN_ESTUDIOS y ASIGNATURA, tomaremos la información de dichas tablas para componer a PLAN_ASIGNATURA, sumando la información propia de la tabla como lo es: semestre, tipo, lab_separado, horas_teoricas, horas_practicas y créditos.

La tabla SERIACIÓN se poblará respecto a las asignaturas correspondientes a las carreras que se han trabajado en esta sección (Ingeniería en Computación e Ingeniería Eléctrica y Electrónica).

Siguiendo el mismo principio de privacidad de la información de la tabla ALUMNOS, se poblará con datos ficticios la tabla ACADEMICO. Al igual que los números de cuenta, la UNAM maneja un algoritmo para generar los números de trabajador. Este algoritmo se omitirá para el ambiente de pruebas ya que dicho algoritmo no es relevante para el objetivo de la sección.

Para la tabla SALON, sólo se mapeará el edificio A con tres pisos.

En el caso de la tabla ASIGNACION y sus derivadas: ORDINARIO y EXTRAORDINARIO, no serán mapeadas en esta instancia ya que serán utilizadas para probar los procesos que se le pueden integrar al diseño de la base de datos en la sección *4.3 Integración de procesos*.

Otro aspecto a considerar para las pruebas, es la cuestión de poder integrar algún proceso como lo es el de inscripción.

Al ser el diseño independiente del Sistema Manejador de Base de Datos en el que se incorporará, se tiene la flexibilidad de una elección libre de manejador. Por esta razón las pruebas se llevarán a cabo en Oracle 11gR2, creando la base sobre este manejador en un sistema operativo Red Hat Enterprise Linux 5 y haciendo las pruebas correspondientes sobre el mismo ambiente.

4.2 Poblado de la base de datos

Con base en los registros mencionados en el apartado anterior se realizará el poblado de la base de datos a través de la sentencia INSERT.

Se limitará la impresión de inserción de datos, colocando solamente 5 registros por tabla.

Tabla: CODIGO_POSTAL

```
INSERT INTO CODIGO_POSTAL
VALUES ('01000','Álvaro Obregón','Distrito Federal');
INSERT INTO CODIGO_POSTAL
VALUES ('01010','Álvaro Obregón','Distrito Federal');
INSERT INTO CODIGO_POSTAL
VALUES ('01020','Álvaro Obregón','Distrito Federal');
INSERT INTO CODIGO_POSTAL
VALUES ('01028','Álvaro Obregón','Distrito Federal');
INSERT INTO CODIGO_POSTAL
VALUES ('01029','Álvaro Obregón','Distrito Federal');
```

...

Tabla: ALUMNO

```
INSERT INTO ALUMNO(no_cta, clave_codigo, ap_paterno, ap_materno, nombre,
fecha_nac, direccion, teléfono_movil, correo_electronico) VA-
LUES('306293604','01000','Álvarez ','Bernal','Antonio','9-JAN-
1990','Cerrada Juan Escutia No.
49','5512359874','alvarez_bernal@unam.mx');
INSERT INTO ALUMNO(no_cta, clave_codigo, ap_paterno, ap_materno, nombre,
fecha_nac, direccion, teléfono_movil, correo_electronico) VA-
LUES('306293605','01029','Almeyda','Ferrer ','Antonia','8-FEB-
1990','Calle Benito Juárez No.
19','5512359876','almeyda_ferrer@unam.mx');
INSERT INTO ALUMNO(no_cta, clave_codigo, ap_paterno, ap_materno, nombre,
fecha_nac, direccion, teléfono_movil, correo_electronico) VA-
LUES('306293606','02020','Barrios','Domínguez','Raúl','10-MAR-
1990','Residencial Los Pinos Edificio 56 Departamento
6','5512359878','barrios_dominguez@unam.mx');
INSERT INTO ALUMNO(no_cta, clave_codigo, ap_paterno, ap_materno, nombre,
fecha_nac, direccion, teléfono_movil, correo_electronico) VA-
LUES('306293607','02030','Bernal','Ibanez','Remedios','11-APR-
1990','Callejon Victoria No. 3','5512359880','bernal_ibanez@unam.mx');
INSERT INTO ALUMNO(no_cta, clave_codigo, ap_paterno, ap_materno, nombre,
fecha_nac, direccion, teléfono_movil, correo_electronico) VA-
LUES('306293608','03010','Cárdenas ','Fernández','Carlos','15-MAY-
```

```
1990','Calle Miguel Hidalgo No.
99','5512359882','cardenas_fernandez@unam.mx');
```

...

Tabla: DIVISION

```
INSERT INTO DIVISION
VALUES('2100', 'División de Ciencias Básicas');
INSERT INTO DIVISION
VALUES('2200', 'División de Ingeniería Civil y Geomática');
INSERT INTO DIVISION
VALUES('2300', 'División de Ingeniería en Ciencias de la Tierra');
INSERT INTO DIVISION
VALUES('2400', 'División de Ingeniería Eléctrica');
INSERT INTO DIVISION
VALUES('2500', 'División de Ingeniería Mecánica e Industrial');
```

...

Tabla: CARRERA

```
insert into carrera values('107', '2200','Ingeniería Civil');
insert into carrera values('108', '2300','Ingeniería de Minas y metalur-
gia');
insert into carrera values('109', '2400','Ingeniería Eléctrica y Electró-
nica');
insert into carrera values('110', '2400','Ingeniería en Computación');
insert into carrera values('111', '2400','Ingeniería en Telecomunicacio-
nes');
```

...

Tabla: ALUMNO_CARRERA

```
insert into alumno_carrera values ('306293604','110',2010);
insert into alumno_carrera values ('306293605','110',2010);
insert into alumno_carrera values ('306293606','110',2010);
insert into alumno_carrera values ('306293607','110',2010);
insert into alumno_carrera values ('306293608','110',2010);
```

...

Tabla: COORDINACION

```
insert into coordinacion(clave_coordinacion,clave_division,nombre) va-
lues('2110','2100','Coordinación de Matemáticas');
```

```
insert into coordinacion(clave_coordinacion,clave_division,nombre) va-
lues('2120','2100','Coordinación de Física General y Química');
insert into coordinacion(clave_coordinacion,clave_division,nombre) va-
lues('2130','2100','Coordinación de Ciencias Aplicadas');
insert into coordinacion(clave_coordinacion,clave_division,nombre) va-
lues('2140','2100','Coordinación de Área de Cómputo');
insert into coordinacion values('2209','2200','Coordinación de Ingeniería
Civil y Geomática','Coordinación virtual creada con fines jerárquicos');
```

...

Tabla: DEPARTAMENTO

```
insert into departamento(clave_depto, clave_coordinacion, nombre) va-
lues('2112', '2110','Departamento de Cálculo Integral');
insert into departamento(clave_depto, clave_coordinacion, nombre) va-
lues('2113', '2110','Departamento de Cálculo Vectorial');
insert into departamento(clave_depto, clave_coordinacion, nombre) va-
lues('2114', '2110','Departamento de Álgebra');
insert into departamento(clave_depto, clave_coordinacion, nombre) va-
lues('2115', '2110','Departamento de Álgebra Lineal');
insert into departamento(clave_depto, clave_coordinacion, nombre) va-
lues('2116', '2110','Departamento de Geometría Analítica');
```

...

Tabla: PLAN_ESTUDIOS

```
insert into plan_estudios(clave_plan,clave_carrera,anio, creditos_opt,
creditos_obl) values('10107','107',2010,36,362);
insert into plan_estudios(clave_plan,clave_carrera,anio, creditos_opt,
creditos_obl) values('10108','108',2010,24,405);
insert into plan_estudios(clave_plan,clave_carrera,anio, creditos_opt,
creditos_obl) values('10109','109',2010,54,346);
insert into plan_estudios(clave_plan,clave_carrera,anio, creditos_opt,
creditos_obl) values('10110','110',2010,48,360);
insert into plan_estudios(clave_plan,clave_carrera,anio, creditos_opt,
creditos_obl) values('10111','111',2010,24,386);
```

...

Tabla: MODULO_SALIDA

```
insert into modulo_salida values('1185', '10109','Electrónica');
insert into modulo_salida values('1186', '10109','Control y Robótica');
insert into modulo_salida values('1187', '10109','Ingeniería Biomédica');
insert into modulo_salida values('1188', '10109','Eléctrica de Poten-
cia');
```

```
insert into modulo_salida values('1189', '10109','Sistemas Energéticos');
```

...

Tabla: ASIGNATURA

```
insert into asignatura values('0012', 'Administración','2581');
insert into asignatura values('0021', 'Aire Acondicionado y Refrigera-
ción','2540');
insert into asignatura values('0061', 'Dibujo','2131');
insert into asignatura values('0062', 'Álgebra Lineal','2115');
insert into asignatura values('0063', 'Cálculo Vectorial','2113');
```

...

Tabla: TEORIA

```
insert into teoria values('0012');
insert into teoria values('0021');
insert into teoria values('0061');
insert into teoria values('0062');
insert into teoria values('0063');
```

...

Tabla: LABORATORIO

```
insert into laboratorio values('3021','0021');
insert into laboratorio values('3068','0068');
insert into laboratorio values('3071','0071');
insert into laboratorio values('3462','0462');
insert into laboratorio values('3507','0507');
```

...

Tabla: MODULO_ASIGNATURA

```
insert into modulo_asignatura values('1192','0530');
insert into modulo_asignatura values('1195','0575');
insert into modulo_asignatura values('1196','0930');
insert into modulo_asignatura values('1194','0602');
insert into modulo_asignatura values('1195','0602');
```

...

Tabla: PLAN_ASIGNATURA

```
insert into plan_asignatura values ('10110','1865',7,1,0,3,0,6);
insert into plan_asignatura values ('10110','1866',8,1,1,3,2,8);
insert into plan_asignatura values ('10110','1100',1,1,0,4.5,0,9);
insert into plan_asignatura values ('10110','0062',2,1,0,4.5,0,9);
insert into plan_asignatura values ('10110','1422',4,1,0,4.5,0,9);
…
```

Tabla: SERIACION

```
insert into seriacion values('10109','1100','0062');
insert into seriacion values('10109','1546','1656');
insert into seriacion values('10109','1546','1418');
insert into seriacion values('10109','1418','1546');

…
```

Tabla: ACADEMICO

```
insert into académico
(no_trabajador,rfc,ap_paterno,ap_materno,nombre,direccion) values
('123456','ejemplo','Álvarez','García','Lucio','Cerrada Juan Escutia No.
49');
insert into académico
(no_trabajador,rfc,ap_paterno,ap_materno,nombre,direccion) values
('123457','ejemplo','Alustiza','Gómez','Laura','Calle Benito Juárez No.
19');
insert into académico
(no_trabajador,rfc,ap_paterno,ap_materno,nombre,direccion) values
('123458','ejemplo','Arzamendi','Lara','Lorena','Residencial Los Pinos
Edificio 56 Departamento 6');
insert into académico
(no_trabajador,rfc,ap_paterno,ap_materno,nombre,direccion) values
('123459','ejemplo','Becerra','Torres','Marco','Callejon Victoria No.
3');
insert into académico
(no_trabajador,rfc,ap_paterno,ap_materno,nombre,direccion) values
('123460','ejemplo','Bolivar','Becerra','Marcos','Calle Miguel Hidalgo
No. 99');
…
```

Tabla: SALON

```
insert into salon values('A101', 'A', 1, 01, 60);
insert into salon values('A102', 'A', 1, 02, 60);
insert into salon values('A103', 'A', 1, 03, 60);
insert into salon values('A104', 'A', 1, 04, 60);
```

```
insert into salon values('A105', 'A', 1, 05, 60);
...
```

Con esto la base de datos se encuentra poblada para un ambiente de pruebas, a continuación se mostrarán la cantidad de registros creados en la base de datos para cada tabla.

```
SQL> select name from v$database;

NAME
---------
SIIFI

SQL> select count(*) from codigo_postal;

  COUNT(*)
----------
        80

SQL> select count(*) from alumno;

  COUNT(*)
----------
        50

SQL> select count(*) from division;

  COUNT(*)
----------
         7

SQL> select count(*) from carrera;

  COUNT(*)
----------
        12

SQL> select count(*) from coordinacion;

  COUNT(*)
----------
        31

SQL> select count(*) from departamento;

  COUNT(*)
----------
        45

SQL> select count(*) from plan_estudios;

  COUNT(*)
----------
        12

SQL> select count(*) from modulo_salida;

  COUNT(*)
----------
        26

SQL> select count(*) from asignatura;

  COUNT(*)
----------
       618

SQL> select count(*) from teoria;

  COUNT(*)
----------
       539

SQL> select count(*) from laboratorio
  2  ;

  COUNT(*)
----------
        79

SQL> select count(*) from modulo_asignatura;

  COUNT(*)
----------
       148

SQL> select count(*) from plan_asignatura;

  COUNT(*)
----------
       213

SQL> select count(*) from seriacion;

  COUNT(*)
----------
        55

SQL> select count(*) from academico;

  COUNT(*)
----------
        50

SQL> select count(*) from salon;

  COUNT(*)
----------
        27
```

Imagen 4.1 Número de registros

Como se puede observar, el hacer la carga de manera manual puede resultar una tarea tediosa cuando el número de registros necesarios para el poblado de la base de datos está en el orden de los miles e incluso en el de los millones, por lo que se deben buscar estrategias alternas que faciliten el desarrollo de dicha tarea. Una posibilidad para realizar la carga de los datos de manera semiautomática es a través de store procedures.

Un Store Procedure o Procedimiento almacenado, es un conjunto de instrucciones SQL almacenadas como un objeto dentro de la base de datos a las que se pueden recurrir llamando al procedimiento[24]. De esta forma se puede generar un conjunto de instrucciones que a su vez sean llamadas por un store procedure y ejecutar todas las instrucciones en grupo.

Otra estrategia que permitirá realizar carga de información de una manera más automatizada es a través de triggers. *Un trigger o disparador es un objeto de base de datos el*

cual se ejecuta cuando sucede un evento predeterminado sobre las tablas a las que se encuentra asociado. Los eventos que provocan que un trigger sea ejecutado son inserciones, borrados o actualizaciones de registros sobre una o varias tablas[32].

De esta forma se puede ejemplificar que la carga de información que se hizo de forma manual como prueba, se puede realizar de una forma más automatizada cuando el proyecto se encuentre en una fase productiva y el número de registros a insertar y/o modificar sea mucho mayor.

4.3 Integración de procesos

Como su nombre lo dice, el SIIFI es un sistema que pretende integrar algunos procesos a su funcionamiento, por lo que es necesario demostrar la factibilidad de esta actividad y si se cuenta con los datos necesarios para agregar nuevas entidades al diseño de la base de datos que ayuden al funcionamiento de un proceso dado. En esta etapa de pruebas, se busca integrar algunos de los procesos que conformarán al Sistema Integral de Información de la Facultad de Ingeniería. La primera prueba se realizará con un proceso que ya está integrado en el diseño de la base de datos, el cuál es el proceso de asignación.

Seguido a las pruebas realizadas a este proceso, se integrarán otros dos procesos al diseño de la base de datos en forma de entidades. Los procesos a anexar al ambiente de pruebas será el de inscripción e historial.

4.3.1 Proceso de asignación

Como se ha visto hasta el momento, el proceso de asignación consta en generar un grupo de relaciones, entre entidades, en las cuales se generan grupos con un profesor determinado, un horario, un salón y la asignatura con toda la información que conlleva cada grupo de datos. El diseño del proceso de asignación se puede ver en el diseño del diagrama entidad relación de la base de datos. La parte correspondiente a la asignación se muestra en la imagen 4.2

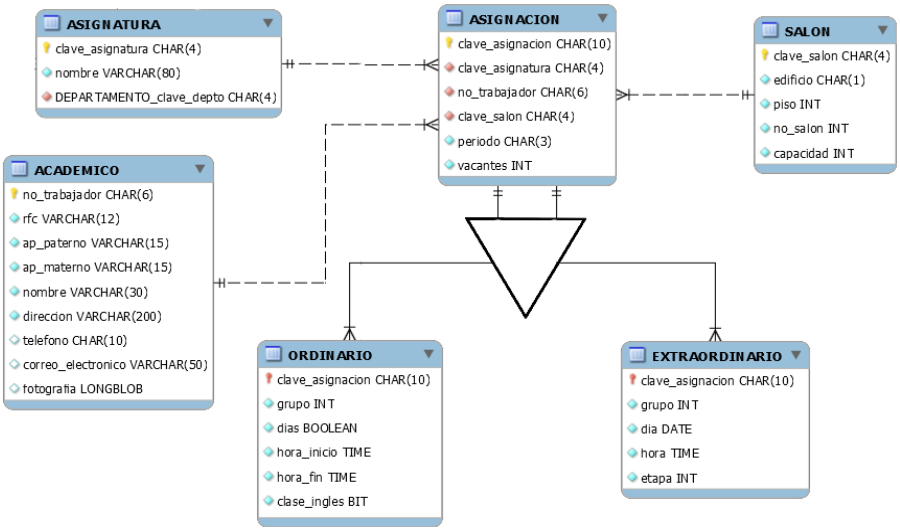


Imagen 4.2 Proceso de asignación

El proceso de asignación se puede hacer en dos vertientes, la primera de forma ordinaria que es el proceso que se realiza semestre a semestre para que un alumno pueda inscribir una materia en la que se presentará a las clases que se llevarán a lo largo del periodo escolar. La segunda forma es la asignación extraordinaria que es mediante el examen extraordinario que se realiza en una sola exhibición teniendo la oportunidad de presentar siguientes etapas en el caso de no aprobar las primeras.

Para ejemplificar dicho proceso a nivel base de datos se realizarán inserts en dos etapas utilizando los datos ficticios con los que se pobló la base de datos. Tomaremos como ejemplo que queremos crear un grupo ordinario de la materia de Álgebra, que será el grupo 01, cuya clase impartirá el profesor Lucio Álvarez García en el salón A101 los días lunes, miércoles y viernes en un horario de 7:00 a 8:30 horas. El periodo a impartir esta asignatura será el 2016-2 con un cupo máximo de 30 alumnos y con la particularidad que tendrá una semana de clase en el idioma inglés.

Con la información anteriormente mencionada, se comenzará a sacar los datos de importancia para nuestro primer insert en la tabla ASIGNACIÓN. Primeramente se necesita crear la clave que llevará la asignación, esto se hará con el algoritmo propuesto en la sección *3.4 "Tipos de datos"* de este trabajo, siendo los primeros 3 dígitos los que representan el periodo, los siguientes 4 dígitos representan la asignatura, los siguientes 2 dígitos representan número de grupo y el último dígito representa la etapa de extraoridnario, en caso de ser ordinario se asigna un cero. De esta forma tenemos que nuestra clave de asignación es la siguiente: *1620062010*

La siguiente información que se necesita para la tabla es la clave de la asignatura de Álgebra, que es *0062*,después sigue el número de trabajador del profesor *Lucio Álvarez García* que es el *123456*, continuando, se necesita la clave del salón que es *A101*, se conoce el periodo que será el *20162*, por último se necesita conocer el número máximo de cupo que tendrá el grupo que como lo sabemos es *30*. Con esta información podemos

ingresar el primer registro en la tabla ASIGNACIÓN. A continuación la tabla 4.1 mostrará el resumen de los datos a ingresar.

Tabla 4.1 Resgistros ASIGNACIÓN

Clave asignación	1620062010
Clave asignatura	0062
Número trabajador	123456
Clave salón	A101
Periodo	162
Límite de cupo	30

Se puede observar que se cuenta con la información necesaria para poblar un registro en la tabla ASIGNACIÓN por lo que se procede a realizar dicho insert:

```
insert into asignacion values (‘162006202010’,’0062’,’A101’,’162’,30);
```

De esta forma se puede proceder a trabajar en la tabla ORDINARIO, en la cual se necesita la misma clave de ASIGNACIÓN que es *1620062010*, el número de grupo que será el *01*, los días a impartir la clase que como se menciona serán los días lunes, miércoles y viernes que siguiendo la forma en como se ingresarán los datos de manera booleana, corresponde un valor de *101010*, la hora de inicio será a las *07:00* y la hora de finalización es a las *08:30* horas. Por otro lado se menciona que tendrá una semana de clase en inglés lo cual se representará con un *1*. La tabla 4.2 muestra los datos recolectados.

Tabla 4.2 Registros ORDINARIO

Clave asignación	1620062010
Número de grupo	1
Días de clase	101010
Hora inicio	07:00
Hora finalización	08:30
Clase inglés	1

Con los datos anteriormente mostrados, se puede proceder a realizar el insert para el primer registro de la tabla ORDINARIO:

```
insert into ordinario values ('1621306010',01,'101010',
TO_DATE( '07:00','HH24:MI'),TO_DATE( '08:30','HH24:MI'),'1');
```

Con se tiene completo el primer registro para el grupo *1* de la materia de *Álgebra* en el periodo *2016-1* en la base de datos. A través de las tablas ASIGNACIÓN y ORDINARIO puede ser consultada toda la información respecto a este grupo.

De la misma forma que se realizaron los insert para el primer registro de ambas tablas, se realizará un poblado de la base de datos para las tablas de ASIGNACIÓN y sus tablas dependientes (ORDINARIO y EXTRAORDINARIO) para poder ser utilizados en próximas pruebas.

Para ASIGNACIÓN:

```
insert into asignacion values
('1621306010','1306','123456','A101','162',40);
insert into asignacion values
('1621306020','1306','123457','A101','162',40);
insert into asignacion values
('1621306030','1306','123458','A101','162',40);
insert into asignacion values
('1621306011','1306','123456','A201','162',20);
insert into asignacion values
('1621306012','1306','123456','A201','162',20);
insert into asignacion values
('1621306013','1306','123456','A201','162',20);
...
```

Para ORDINARIO:

```
insert into ordinario values ('1621306010',01,'101010',TO_DATE(
'07:00','HH24:MI'),
TO_DATE( '08:30','HH24:MI'),'1');
insert into ordinario values ('1621306020',02,'010100',TO_DATE(
'07:00','HH24:MI'),
TO_DATE ('09:15','HH24:MI'),'0');
insert into ordinario values ('1621306030',03,'101010',TO_DATE
('08:30','HH24:MI'),
TO_DATE( '10:00','HH24:MI'),'0');
insert into ordinario values ('1621306040',04,'010100',TO_DATE(
'09:15','HH24:MI'),
TO_DATE ('11:30','HH24:MI'),'0');
insert into ordinario values ('1620063010',01,'101010',TO_DATE(
'08:30','HH24:MI'),
TO_DATE ('10:00','HH24:MI'),'1');
...
```

Para EXTRAORDINARIO

```
insert into extraordinario values
('1621306011',1,TO_DATE('01/07/2016','DD/MM/YYYY'),
TO_DATE('07:00','HH24:MI'),1);
insert into extraordinario values
('1621306012',1,TO_DATE('15/07/2016','DD/MM/YYYY'),
TO_DATE('07:00','HH24:MI'),2);
insert into extraordinario values
('1621306013',1,TO_DATE('30/07/2016','DD/MM/YYYY'),
TO_DATE('07:00','HH24:MI'),3);
insert into extraordinario values
('1620063011',1,TO_DATE('01/07/2016','DD/MM/YYYY')
,TO_DATE('10:00','HH24:MI'),1);
```

```
insert into extraordinario values
('1620063012',1,TO_DATE('15/07/2016','DD/MM/YYYY'),
TO_DATE('10:00','HH24:MI'),2);
```

4.3.2 Proceso de inscripción

Como se ha visto a lo largo de este trabajo, el proceso de inscripción tiene dos formas de llevarse a cabo: de forma ordinaria y de forma extraordinaria. Se comenzará ejemplificando el proceso de inscripción en forma ordinaria con se muestra en la imagen 4.3

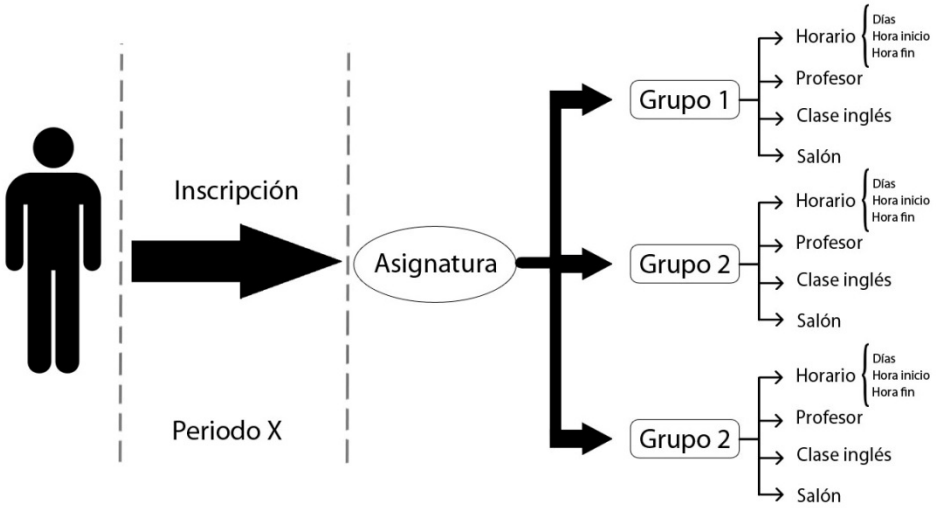


Imagen 4.3 Inscripción ordinaria

Como se observa en la imagen, un alumno se inscribe en un periodo, eligiendo una asignatura en la cual se seleccionará un grupo el cual ya cuenta con un horario asignado que consta de los días a impartir dicha asignatura, una hora de inicio y una hora de finalización, también tiene asignado un profesor, la información de si el curso tendrá semana de clases en el idioma inglés y el salón donde se impartirán las clases. La información mostrada es similar a la tabla que hemos manejado como asignación, por lo que en realidad el proceso de inscripción no será más que la relación entre el ente alumno y la entidad asignación.

El caso de la inscripción extraordinaria es muy similar a la inscripción ordinaria, cambia en algunos datos como lo son que se manejará una fecha determinada en el horario como una hora especifica. También se anexa la etapa del examen extraordinario y se elimina la información sobre la clase en inglés. Esto se puede observar en la imagen 4.4

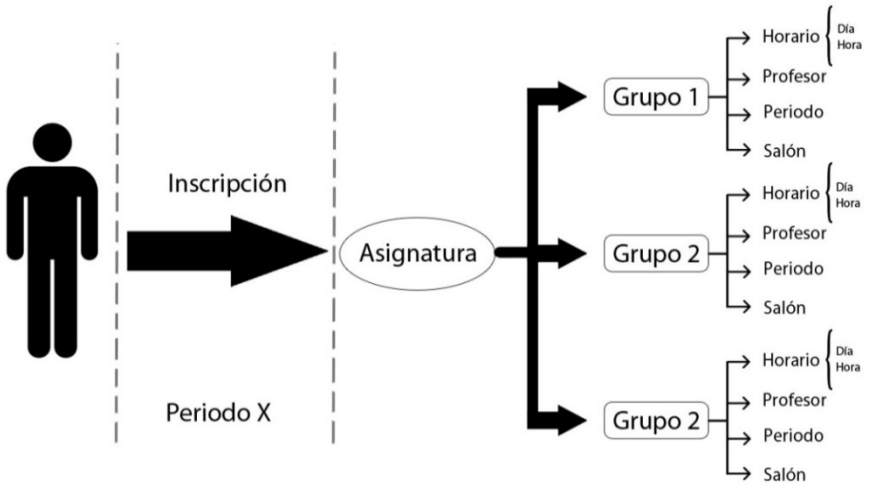


Imagen 4.4 Inscripción extraordinaria

Una vez entendido como se maneja una inscripción, ya sea ordinaria o extraordinaria, se puede modelar la relación de entidades entre ALUMNO y ASIGNACIÓN.

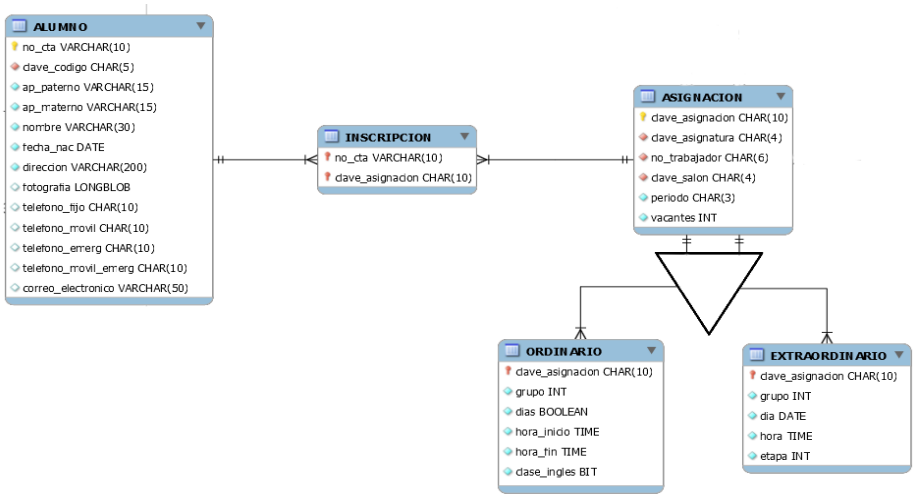


Imagen 4.5 Relación entidad ALUMNO-ASIGNACION

A través de la imagen 4.5 se puede ver el modelado de la relación entre las entidades ALUMNO y ASIGNACIÓN que dan como resultado una nueva entidad que servirá como el proceso de inscripción. Teniendo ya el modelo, se procede a la creación de la tabla vía código SQL y anexarla a la base de datos. El código quedará de la siguiente manera:

```
CREATE TABLE inscripcion(
clave_asignacion CHAR(10) NOT NULL,
no_cta VARCHAR(10) NOT NULL,
CONSTRAINT fk_INSCRIPCION_ASIGNACION
FOREIGN KEY (clave_asignacion)
REFERENCES ASIGNACION(clave_asignacion),
CONSTRAINT fk_INSCRIPCION_ALUMNO
FOREIGN KEY (no_cta)
```

```
REFERENCES ALUMNO(no_cta)
);
```

Ya con la tabla creada en la base de datos, se puede comenzar a poblarla utilizando los datos que ya existen en la tabla ASIGNACIÓN. Para esto se utilizará el ejemplo del grupo 1 de la asignatura de Álgebra impartida por el profesor *Lucio Álvarez García*. Esta información está resumida en la clave de asignación que es *1620062010*, la cual utilizaremos para el insert. Ahora, el alumno que quiere inscribir esta asignatura en el periodo *2016-2* es *Antonio Álvarez Bernal*, del cual necesitamos el número de cuenta que es *306293604*, con esto se tienen los dos datos necesarios para realizar la inscripción de la asignatura de Álgebra en el grupo *1*. El insert quedará de la siguiente manera:

insert into inscripción values('16200622010','306293604');

Se puede pensar que los datos para realizar la inscripción son pocos, pero gracias a la estructura de la base de datos, son suficientes para que con sólo esos datos pueda relacionarse varias tablas y obtener la información que se necesite.

Con los datos del poblado de la tabla ASIGNACIÓN, se realizará un poblado de la tabla INSCRIPCIÓN para los alumnos con que se cuentan.

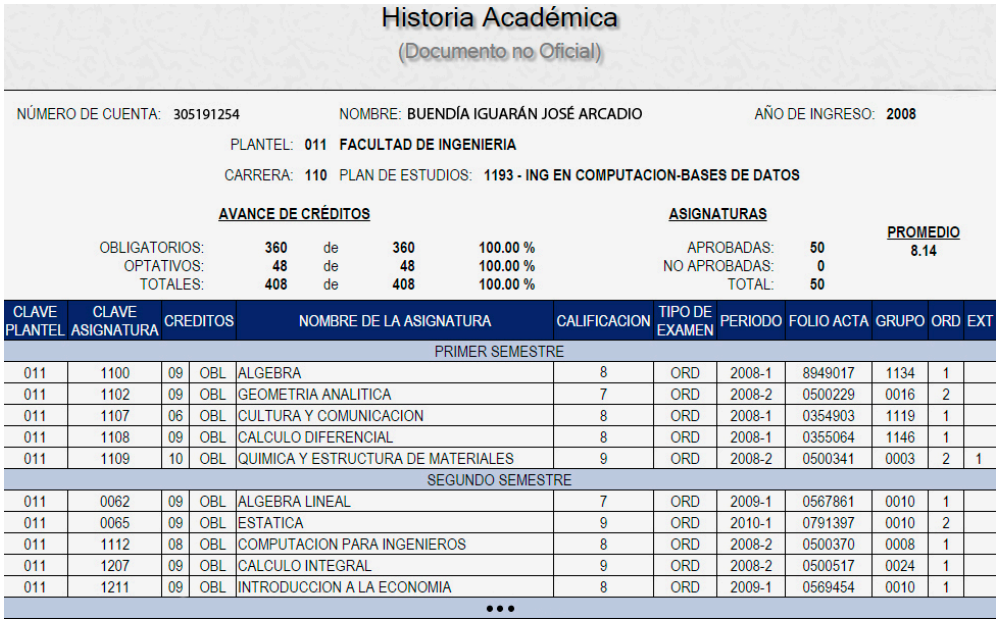
Para INSCRIPCIÓN:

```
insert into inscripcion values('306293604','1621306010');
insert into inscripcion values('306293605','1621306010');
insert into inscripcion values('306293606','1621306010');
insert into inscripcion values('306293607','1621306010');
insert into inscripcion values('306293608','1621306010');
...
```

4.3.3 Proceso historial

Para comenzar a describir el proceso historial, es menester conocer los datos necesarios para su formación, ya con esto se sabrá qué datos se tienen disponibles en otras entidades y la relación que se generará, así como también los datos a agregar al proceso.

Para darnos una idea de los datos necesarios que conformarán el proceso, tomaremos como referencia un ejemplo del historial académico que los alumnos consultan en el sitio oficial de la Dirección General de Administración Escolar de la UNAM (https://www.dgae-siae.unam.mx/). Esto con el propósito de tomar la información que nos sea de utilidad y descartar la no necesaria. La figura 4.6 muestra dicho ejemplo.



Historia Académica

(Documento no Oficial)

NÚMERO DE CUENTA: 305191254 NOMBRE: BUENDÍA IGUARÁN JOSÉ ARCADIO AÑO DE INGRESO: 2008

PLANTEL: 011 FACULTAD DE INGENIERIA

CARRERA: 110 PLAN DE ESTUDIOS: 1193 - ING EN COMPUTACION-BASES DE DATOS

AVANCE DE CRÉDITOS

OBLIGATORIOS:	360	de	360	100.00 %
OPTATIVOS:	48	de	48	100.00 %
TOTALES:	408	de	408	100.00 %

ASIGNATURAS

APROBADAS:	50
NO APROBADAS:	0
TOTAL:	50

PROMEDIO 8.14

CLAVE PLANTEL	CLAVE ASIGNATURA	CREDITOS		NOMBRE DE LA ASIGNATURA	CALIFICACION	TIPO DE EXAMEN	PERIODO	FOLIO ACTA	GRUPO	ORD	EXT
PRIMER SEMESTRE											
011	1100	09	OBL	ALGEBRA	8	ORD	2008-1	8949017	1134	1	
011	1102	09	OBL	GEOMETRIA ANALITICA	7	ORD	2008-2	0500229	0016	2	
011	1107	06	OBL	CULTURA Y COMUNICACION	8	ORD	2008-1	0354903	1119	1	
011	1108	09	OBL	CALCULO DIFERENCIAL	8	ORD	2008-1	0355064	1146	1	
011	1109	10	OBL	QUIMICA Y ESTRUCTURA DE MATERIALES	9	ORD	2008-2	0500341	0003	2	1
SEGUNDO SEMESTRE											
011	0062	09	OBL	ALGEBRA LINEAL	7	ORD	2009-1	0567861	0010	1	
011	0065	09	OBL	ESTATICA	9	ORD	2010-1	0791397	0010	2	
011	1112	08	OBL	COMPUTACION PARA INGENIEROS	8	ORD	2008-2	0500370	0008	1	
011	1207	09	OBL	CALCULO INTEGRAL	9	ORD	2008-2	0500517	0024	1	
011	1211	09	OBL	INTRODUCCION A LA ECONOMIA	8	ORD	2009-1	0569454	0010	1	
•••											

Figura 4.6 Historial académico DGAE

Tomando un orden de arriba abajo y de izquierda a derecha, el primer dato que se muestra es el número de cuenta y el nombre del alumno, información que en la base de datos se encuentra en la entidad ALUMNO. Después se tiene el año de ingreso, el cual se almacena en la entidad ALUMNO_CARRERA con el nombre de generación. Siguiendo el orden, el ejemplo muestra la clave y el nombre del plantel; para la base de datos se hará caso omiso de esta información ya que el sistema es propio de la Facultad de Ingeniería por lo que todos los alumnos que serán almacenados en la base, pertenecen a dicha institución. El siguiente dato es la clave de la carrera, que se almacena en la entidad CARRERA, seguido de la clave del plan de estudios almacenada en la entidad PLAN_ESTUDIOS. Continuando, los siguientes datos son el nombre de la carrera y de existir, el módulo de salida; el primer dato hace referencia a la entidad CARRERA, mientras que el segundo se guarda en la entidad MÓDULO_SALIDA. Para el avance de créditos se tiene créditos obligatorios y optativos (por plan de estudios) así como la suma de estos dos, dichos atributos pueden ser obtenidos de la entidad PLAN_ESTUDIOS llamados creditos_opt y creditos_obl. En cuanto al porcentaje de avance, se utilizará el atributo créditos de la entidad PLAN_ASIGNATURA, que especifica el número de créditos que tiene cada asignatura, y el total de créditos que se tienen que cumplir de acuerdo a cada plan de estudio y si la asignatura es del tipo obligatoria u optativa.

El siguiente campo es el referente al número de asignaturas aprobadas y reprobadas, esto se puede tomar fácilmente con un contador agregando el atributo calificación que tomará en cuenta la nota más reciente de la asignatura correspondiente, si es mayor o igual a 6 se toma como aprobada mientras que si es menor o igual a 5 será clasificada como reprobada. Para promedio se utilizará de igual forma un nuevo atributo que es el de promedio, haciendo una sumatoria de todas las calificaciones y dividiéndola en el número de materias totales. Después viene una tabla con datos de la asignatura y su calificación, todo esto organizado por semestre, campo que podemos obtener de la entidad

PLAN_ASIGNATURA. El primer dato de la lista es la clave del plantel que como ya se mencionó, será omitida debido a que todos los alumnos almacenados en el sistema son pertenecientes a la Facultad de Ingeniería, el siguiente dato es la clave de la asignatura que se obtiene de la entidad ASIGNATURA así como también el nombre de la misma; el número de créditos se obtiene del atributo créditos perteneciente a la entidad PLAN_ASIGNATURA. Para conocer si es de tipo obligatoria u optativa, se puede consultar el dato tipo de la entidad PLAN_ASIGNATURA. La calificación será el nuevo atributo a crear para este proceso. Para determinar el apartado tipo de examen, se utilizará la información de las tablas ORDINARIO o EXTRAORDINARIO, que servirá como un contador para representar las últimas dos columnas de la figura 4.6 que indica cuántas veces se cursó la materia y cuál fue el modo de aprobación. El apartado de periodo puede ser tomado directamente de la entidad ASIGNACIÓN.

Por último el grupo depende de las entidades ORDINARIO o EXTRAORDINARIO y su atributo del mismo nombre.

En resumen, la información que se necesita para el proceso HISTORIAL es la siguiente:

-Número de cuenta
-Nombre alumno
-Año de ingreso
-Carrera
-Plan de estudios
-Módulo de salida
-Créditos totales del plan
-Créditos de avance
-Número de asignaturas aprobadas y reprobadas
-Promedio
-Clave asignatura
-Nombre asignatura
-Créditos por asignatura
-Tipo de asignatura (obligatoria/optativa)
-Calificación
-Modo en que se aprobó (tipo de examen)
-Periodo
-Grupo
-Veces que se cursó de forma ordinaria
-Veces que se cursó de forma extraordinaria

Como se puede observar el único atributo que se creará es el de calificación ya que todos los demás se pueden tomar de otras entidades y ser relacionados a través de sus claves primarias. Por ejemplo, conociendo el número de cuenta de un alumno, se puede obtener información como lo es su nombre, la carrera en la que está inscrito, el año de ingreso, el plan de estudios que pertenece a su carrera y a su generación, y de ser el caso, si se encuentra inscrito a un módulo de salida.

Hay varias formas de modelar este proceso en la base de datos, una de ellas sería tomar la entidad de INSCRIPCIÓN que cuenta con todos los datos que se necesitan y agregar el atributo calificación con una condición de que puede ser NULL, así se inscribirá la materia y se dejará vació el campo calificación y una vez terminado el semestre sólo se actualizará dicho campo. Este modelado se presenta en la figura 4.7

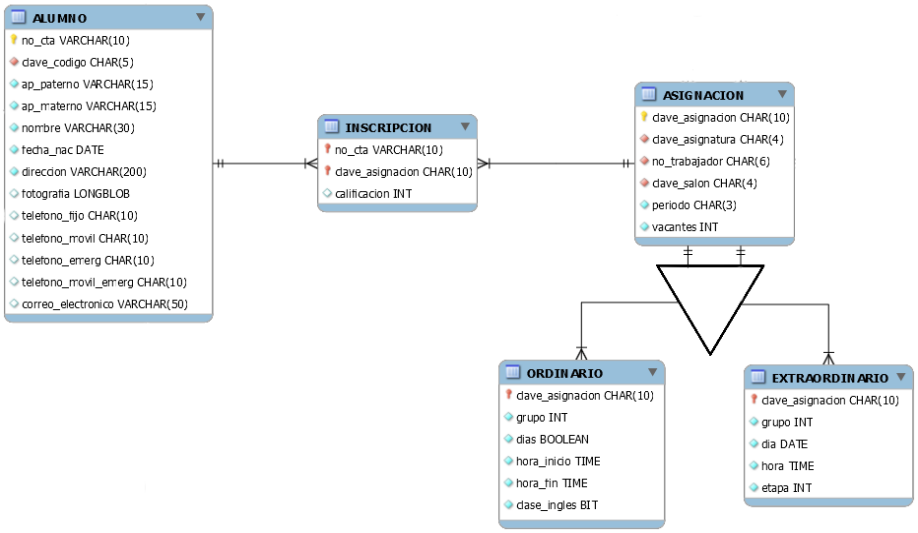


Figura 4.7 Proceso de historial con inscripción

Como se puede ver, el proceso de HISTORIAL será totalmente dependiente del de INSCRIPCIÓN y esto es correcto ya que de no haber una inscripción la asignatura no podrá ser calificada. De esta forma la creación de la tabla INSCRIPCIÓN se modificará agregándole el atributo calificación como a continuación se muestra:

```
ALTER TABLE inscripcion ADD calificacion INT NULL;
```

De esta manera, para formar un histórico de las materias que ya han sido inscritas y a su vez evaluadas, se procederá a hacer el insert de la siguiente manera:

```
insert into inscripción values (‘306293604’, ‘16200622010’,8);
```

Siendo el primer valor el correspondiente valor al número de cuenta del alumno, el segundo al número de asignación o clave de acta y el tercero a la calificación obtenida durante este curso.

Para anexar el valor de la calificación una vez que ya se ha insertado los valores correspondientes a clave de asignación y número de cuenta, se hará a través de un update como se muestra a continuación:

```
UPDATE inscripcion
SET calificación=8
WHERE clave_asignacion=’16200622010’
AND no_cta=’306293604’;
```

De esta forma se agregará la calificación obtenida en la asignatura/grupo que el alumno inscribió en ese periodo.

Para ejemplificar de mejor forma el proceso, se hará un llenado histórico de semestres anteriores con calificaciones obtenidas por una cantidad de alumnos inscritos a ciertas materias.

Para HISTORIAL:

```
update inscripcion set calificacion=10
where clave_asignacion='1621306010' and no_cta='306293604';
update inscripcion set calificacion=9
where clave_asignacion='1621306010' and no_cta='306293605';
update inscripcion set calificacion=8
where clave_asignacion='1621306010' and no_cta='306293606';
update inscripcion set calificacion=7
where clave_asignacion='1621306010' and no_cta='306293607';
update inscripcion set calificacion=6
where clave_asignacion='1621306010' and no_cta='306293608';
...
```

Para el ejemplo anterior, se calificó a los alumnos inscritos en el grupo 1 de la asignatura Ecuaciones Diferenciales.

Con esto se da por concluido la ejemplificación de cómo con la base de datos núcleo, pueden ser agregados diversos procesos que sean de utilidad para los sistemas ocupados en la Facultad de Ingeniería.

4.4 Consultas sobre la base de datos

Para estas pruebas se utilizará la estructura mostrada en los ejemplos expuestos en la sección *2.4 "SIIFI como herramienta estadística"*, las cuales fueron presentadas de manera teórica y es turno de mostrar cómo se resolverían con el diseño de la base de datos ya definido y los datos almacenados.

Consulta 1

Para comenzar, se resolverá el enunciado "*Número alumnos inscribieron la asignatura de Cálculo Vectorial este semestre*". Para obtener el número de alumnos inscritos a dicha asignatura, necesitamos saber cuál es la clave para Cálculo Vectorial y el periodo actual. Los datos que se necesitan son los siguientes:

Tabla 4.3 Consulta 1

Clave asignatura	0063
Periodo actual	162

El siguiente paso es saber de qué tablas se puede tomar la información. Revisando el diagrama entidad relación de la base de datos, obtenemos que tanto la clave de asignatura como el periodo pertenecen a la tabla ASIGNACIÓN, que está directamente relacionada con la tabla INSCRIPCIÓN que será de donde obtendremos el número de alumnos inscritos a través de un conteo. La forma de relacionar la similitud entre ASIGNACIÓN e INSCRIPCIÓN es por el atributo clave_asignacion, por lo que la consulta quedará de la siguiente manera:

```
SELECT COUNT(*) FROM INSCRIPCION I, ASIGNACION A
WHERE A.PERIODO='162'
AND A.CLAVE_ASIGNATURA='0063'
AND A.CLAVE_ASIGNACION=I.CLAVE_ASIGNACION;
```

Ejecutando esta consulta directamente sobre nuestra base de datos obtenemos el resultado mostrado en la imagen 4.8

```
SQL> select count(*) from inscripcion i, asignacion a
  2  where a.periodo='162'
  3  and a.clave_asignatura='0063'
  4  and a.clave_asignacion=i.clave_asignacion;

  COUNT(*)
----------
        30
```

Imagen 4.8 Número de alumnos inscritos en Cálculo Vectorial

En caso de querer saber los nombres de los alumnos que están inscritos en dicha asignatura y el grupo en que están inscritos, se tendría que involucrar a la tabla ALUMNO y relacionarla con INSCRIPCIÓN a través de su atributo común que es no_cta. La consulta quedaría de la siguiente forma.

```
SELECT AL.AP_PATERNO, AL.AP_MATERNO, AL.NOMBRE, O.GRUPO, A.PERIODO
FROM ALUMNO AL, ASIGNACION A, ORDINARIO O, INSCRIPCION I, ASIGNATURA AG
WHERE A.PERIODO='162'
AND A.CLAVE_ASIGNATURA='0063'
AND I.CLAVE_ASIGNACION=A.CLAVE_ASIGANCION
AND A.CLAVE_ASIGNACION=O.CLAVE_ASIGNACION
AND A.CLAVE_ASIGNATURA=AG.CLAVE_ASIGNATURA
AND I.NO_CTA=AL.NO_CTA;
```

El resultado de la consulta ejecutada sobre la base de datos puede ser observado en la imagen 4.9

```
SQL> select al.ap_paterno, al.ap_materno, al.nombre, o.grupo, a.periodo
from alumno al, asignacion a, ordinario o, inscripcion i,asignatura ag
where a.periodo='162'
and a.clave_asignatura='0063'
and i.clave_asignacion=a.clave_asignacion
and a.clave_asignacion=o.clave_asignacion
and a.clave_asignatura=ag.clave_asignatura
and i.no_cta=al.no_cta;

AP_PATERNO      AP_MATERNO      NOMBRE          GRUPO PER
--------------- --------------- --------------- ---------- ---
Álvarez         Bernal          Antonio              1 162
Almeyda         Ferrer          Antonia              1 162
Barrios         Domínguez       Raúl                 1 162
Bernal          Ibanez          Remedios             1 162
Cárdenas        Fernández       Carlos               1 162
Carrasco        Martínez        Karla                2 162
Durán           Tejada          Jorge                2 162
Domínguez       Almeyda         Paola                2 162
Escobar         González        Andrés               2 162
Estrada         Ordiales        Andrea               2 162
Fernández       Juárez          Gabriel              3 162

AP_PATERNO      AP_MATERNO      NOMBRE          GRUPO PER
--------------- --------------- --------------- ---------- ---
Ferrer          Nunez           Gabriela             3 162
García          Larios          Erick                3 162
González        Pereyra         Lizeth               3 162
Huerta          Bravo           Ricardo              3 162
Noriega         Torres          Alejandra            1 162
Ortega          García          Pedro                1 162
Ordiales        Zúniga          Ada                  1 162
Pérez           Álvarez         Raymundo             1 162
Pereyra         Urbina          Delia                1 162
Rodríguez       Estrada         Ramón                2 162
Reyes           Salcedo         Verónica             2 162

AP_PATERNO      AP_MATERNO      NOMBRE          GRUPO PER
--------------- --------------- --------------- ---------- ---
Suárez          López           Salvador             2 162
Salinas         Sánchez         Rosario              2 162
Tejada          Gómez           Israel               2 162
Torres          Durán           Socorro              3 162
Urbina          Sánchez         Víctor               3 162
Velasco         Noriega         Ana                  3 162
Valdez          López           Iván                 3 162
Zúniga          Juárez          Vanesa               3 162

30 rows selected.
```

Imagen 4.9 Consulta alumnos inscritos en Cálculo Vectorial

Consulta 2

Continuando con los ejemplos, se tomará de la misma sección *2.4* la estructura del segundo enunciado, por lo que la siguiente consulta a obtener será "*Listado de grupos del laboratorio de Principios de termodinámica y electromagnetismo que cuentan con menos de 15 alumnos inscritos en este semestre*".

Para comenzar se mostrará el resultado de los grupos de laboratorio y el número de alumnos inscritos a cada uno de ellos para posteriormente agregar la condición y que únicamente se muestre el resultado de aquellos que tengan menos de 15 alumnos inscritos. Esto se hará a través de la relación entre las tablas ASIGNATURA, ASIGNACIÓN, ORDINARIO e INSCRIPCIÓN, tomando las llaves foráneas que permiten relacionar unas tablas con otras. La consulta sería la siguiente:

```
select asi.nombre, o.grupo, count(i.clave_asignacion) as “ALUMNOS INSCRI-
TOS”
from asignatura asi, asignacion a, ordinario o, inscripcion i
where asi.clave_asignatura=4314
and asi.clave_asignatura=a.clave_asignatura
and a.clave_asignacion=o.clave_asignacion
and a.clave_asignacion=i.clave_asignacion
group by asi.nombre, o.grupo;
```

Y su ejecución sobre la base de datos da el resultado mostrado en la imagen 4.10

```
SQL> select asi.nombre, o.grupo, count(i.clave_asignacion) as "ALUMNOS INSCRITOS"
from asignatura asi, asignacion a, ordinario o, inscripcion i
where asi.clave_asignatura=4314
and asi.clave_asignatura=a.clave_asignatura
and a.clave_asignacion=o.clave_asignacion
and a.clave_asignacion=i.clave_asignacion
group by asi.nombre, o.grupo;  2    3    4    5    6    7

NOMBRE                                                                    GRUPO ALUMNOS INSCRITOS
------------------------------------------------------------------------ ---------- ----------
Laboratorio de Principios de Termodinámica y Electromagnetismo          1               10
Laboratorio de Principios de Termodinámica y Electromagnetismo          2               20
```

Imagen 4.10 Número de alumnos inscritos en los grupos

Como se puede observar en el resultado de la consulta, nos muestra todos los grupos que tiene el laboratorio de Principios de termodinámica y electromagnetismo así como el número de alumnos inscritos en cada uno de ellos. Ahora bien, se agregará la restricción de que sólo muestre los grupos que cuenten con menos de 15 alumnos inscritos, a través de un conteo, la consulta quedaría de la siguiente manera:

```
select asi.nombre, o.grupo, count(i.clave_asignacion) as "ALUMNOS INSCRI-
TOS"
from asignatura asi, asignacion a, ordinario o, inscripcion i
where asi.clave_asignatura=4314
and asi.clave_asignatura=a.clave_asignatura
and a.clave_asignacion=o.clave_asignacion
and a.clave_asignacion=i.clave_asignacion
having count(i.clave_asignacion)<15
group by asi.nombre, o.grupo;
```

Siendo el resultado de la consulta, ya con la restricción de menos de 15 alumnos, el mostrado en la imagen 4.11

```
SQL>
select asi.nombre, o.grupo, count(i.clave_asignacion) as "ALUMNOS INSCRITOS"
from asignatura asi, asignacion a, ordinario o, inscripcion i
where asi.clave_asignatura=4314
and asi.clave_asignatura=a.clave_asignatura
and a.clave_asignacion=o.clave_asignacion
and a.clave_asignacion=i.clave_asignacion
having count(i.clave_asignacion)<15
group by asi.nombre, o.grupo;
SQL>   2    3    4    5    6    7    8
NOMBRE                                                                    GRUPO ALUMNOS INSCRITOS
------------------------------------------------------------------------ ---------- ----------
Laboratorio de Principios de Termodinámica y Electromagnetismo          1               10
```

Imagen 4.11 Grupos de laboratorio con menos de 15 alumnos

Consulta 3

Para el tercer ejemplo de consulta, se tomará el siguiente enunciado: "*Listado de salones desocupados de las 11:30 h a las 13:00 h los días lunes en el edificio A*". Para esto primeramente mostraremos el listado de los salones pertenecientes al edificio A con ayuda de la tabla SALÓN, a través de la siguiente consulta:

```
select * from salón where edificio='A';
```

Teniendo como resultado el mostrado en la imagen 4.12

```
SQL> select * from salon
where edificio='A';

CLAV E       PISO   NO_SALON  CAPACIDAD
---- - ---------- ---------- ----------
A101 A          1          1         60
A102 A          1          2         60
A103 A          1          3         60
A104 A          1          4         60
A105 A          1          5         60
A106 A          1          6         60
A107 A          1          7         60
A108 A          1          8         60
A109 A          1          9         60
A201 A          2          1         40
A202 A          2          2         40

CLAV E       PISO   NO_SALON  CAPACIDAD
---- - ---------- ---------- ----------
A203 A          2          3         40
A204 A          2          4         40
A205 A          2          5         40
A206 A          2          6         40
A207 A          2          7         40
A208 A          2          8         40
A209 A          2          9         40
A301 A          3          1         50
A302 A          3          2         50
A303 A          3          3         50
A304 A          3          4         50

CLAV E       PISO   NO_SALON  CAPACIDAD
---- - ---------- ---------- ----------
A305 A          3          5         50
A306 A          3          6         50
A307 A          3          7         50
A308 A          3          8         50
A309 A          3          9         50

27 rows selected.
```

Imagen 4.12 Salones del edificio A

Ya con el resultado de salones que están en el edificio A, procederemos a relacionar las tablas necesarias para obtener aquellos que están desocupados los días lunes a las 11:30 horas. Las tablas que necesitaremos relacionar son ASIGNACIÓN, ORDINARIO y SALÓN.

La lógica a seguir será encontrar el listado de salones donde la hora de inicio de clases no esté entre las 11:30 y 13:00 horas al igual que la hora de finalización no coincida en este rango para los días lunes. Esto a través de la siguiente consulta:

```
select a.clave_salon, to_char(o.hora_inicio,'HH24:MI') as hora_inicio,
to_char(o.hora_fin,'HH24:MI') as hora_fin, o.dias
from asignacion a, ordinario o, salon s
where o.clave_asignacion=a.clave_asignacion
and a.clave_salon=s.clave_salon
and to_char(o.hora_inicio,'HH24:MI') not between
'11:30' and '13:00'
and to_char(o.hora_fin,'HH24:MI') not between
'11:30' and '13:00'
and s.edificio='A'
and o.dias like'1%';
```

El resultado de dicha consulta es el mostrado en la imagen 4.13

```
SQL> select a.clave_salon, to_char(o.hora_inicio,'HH24:MI') as hora_inicio,
to_char(o.hora_fin,'HH24:MI') as hora_fin, o.dias
from asignacion a, ordinario o, salon s
where o.clave_asignacion=a.clave_asignacion
and a.clave_salon=s.clave_salon
and to_char(o.hora_inicio,'HH24:MI') not between '11:30' and '13:00'
and to_char(o.hora_fin,'HH24:MI') not between '11:30' and '13:00'
and s.edificio='A'
and o.dias like'1%';  2    3    4    5    6    7    8    9

CLAV HORA_INICIO                    HORA_FIN                      DIAS
---- ------------------------------ ------------------------------ ------
A101 07:00                          08:30                         101010
A101 08:30                          10:00                         101010
A102 08:30                          10:00                         101010
A105 14:30                          16:00                         101000
A106 07:00                          08:30                         101010
A106 08:30                          10:00                         101010
A108 14:30                          16:00                         101010
A109 14:30                          16:00                         101000
A109 16:00                          17:30                         101000

9 rows selected.
```

Imagen 4.13 Salones desocupados de 11:30 a 13:00 h

Como se puede observar, se muestran los salones libres de 11:30 a 13:00 horas así como también los horarios en que dichos salones estarán ocupados y que todos son del día lunes.

Consulta 4

El siguiente enunciado a resolver es el siguiente: "*Número de alumnos aprobados y reprobados de la materia Ecuaciones diferenciales*". Para esto, se harán dos consultas: una que dé el número de alumnos aprobados y la segunda que indique el número de alumnos

reprobados. Esto a través de la relación entre las tablas INSCRIPCIÓN, ASIGNATURA, y ASIGNACIÓN, en donde el atributo de la tabla inscripción, calificación, sea mayor a 5, en el caso de los alumnos aprobados, o menor igual a 5 en el caso de los alumnos reprobados.

Para conocer los alumnos aprobados la consulta es la siguiente:

```
select asi.nombre, count(i.calificacion) as "alumnos aprobados"
from inscripcion i, asignatura asi, asignacion a
where asi.clave_asignatura='1306'
and asi.clave_asignatura=a.clave_asignatura
and a.clave_asignacion=i.clave_asignacion
and i.calificacion>5
group by asi.nombre;
```

El resultado de esta consulta es el mostrado en la imagen 4.14

```
SQL> select asi.nombre,count(i.calificacion) as "alumnos aprobados"
from inscripcion i, asignatura asi, asignacion a
where asi.clave_asignatura='1306'
and asi.clave_asignatura=a.clave_asignatura
and a.clave_asignacion=i.clave_asignacion
and i.calificacion>5
group by asi.nombre;   2    3    4    5    6    7

NOMBRE                                             alumnos aprobados
-------------------------------------------------- -----------------
Ecuaciones Diferenciales                                           8
```

Imagen 4.14 Alumnos aprobados materia Ecuaciones diferenciales

Como se observa en la imagen se obtiene que los alumnos que cumplen con la condición de calificación mayor a 5 son 8.

Para obtener la cantidad de alumnos reprobados, sólo se necesita cambiar la condición de calificación <= 5. La consulta queda de la siguiente manera:

```
select asi.nombre, count(i.calificacion) as "alumnos reprobados"
from inscripcion i, asignatura asi, asignacion a
where asi.clave_asignatura='1306'
and asi.clave_asignatura=a.clave_asignatura
and a.clave_asignacion=i.clave_asignacion
and i.calificacion<=5
group by asi.nombre;
```

El resultado de esta consulta es mostrado en la imagen 4.15

```
SQL> select asi.nombre,count(i.calificacion) as "alumnos reprobados"
from inscripcion i, asignatura asi, asignacion a
where asi.clave_asignatura='1306'
and asi.clave_asignatura=a.clave_asignatura
and a.clave_asignacion=i.clave_asignacion
and i.calificacion<=5
group by asi.nombre;   2    3    4    5    6    7

NOMBRE                                             alumnos reprobados
-------------------------------------------------- ------------------
Ecuaciones Diferenciales                                            2
```

Imagen 4.15 Alumnos reprobados materia Ecuaciones diferenciales

La imagen 4.15 muestra aquellos alumnos que no tienen una calificación mayor a 5, los cuales son 2.

Consulta 5

La siguiente consulta a resolver se basa en el siguiente enunciado: "*Número aproximado de alumnos que están en el segundo piso del Edificio A los días martes a las 7:00 h*". Para esto, se calculará el número de alumnos inscritos en los diversos grupos que tienen asignados su clase los días martes en el horario de las 7:00 h. Esto será un número aproximado ya que considera el total de los alumnos inscritos y no toma en cuenta si asistieron o no a la clase de ese día o si personas ajenas a dichos grupos se encuentran en los pasillos de dicho edificio. La consulta que nos dará este número aproximado es la siguiente:

```
select count(i.no_cta) as ALUMNOS
from inscripcion i, asignacion a, salon s, ordinario o
where s.edificio='A'
and s.clave_salon=a.clave_salon
and a.clave_asignacion=o.clave_asignacion
and a.clave_asignacion=i.clave_asignacion
and o.dias like '_1%'
and to_char(o.hora_inicio,'HH24:MI')='07:00';
```

El resultado de la consulta se puede ver en la imagen 4.16

```
SQL> select count(i.no_cta) as ALUMNOS from inscripcion i, asignacion a, salon s, ordinario o
where s.edificio='A'
and s.clave_salon=a.clave_salon
and a.clave_asignacion=o.clave_asignacion
and a.clave_asignacion=i.clave_asignacion
and o.dias like '_1%'
and to_char(o.hora_inicio,'HH24:MI')='07:00';  2    3    4    5    6    7

   ALUMNOS
----------
        10
```

Imagen 4.16 Número de alumnos en el edificio A

Consulta 6

El siguiente enunciado es "*Académico con mayor número de alumnos reprobados en la asignatura Ecuaciones diferenciales*". Para resolver dicho enunciado se tiene que involucrar a las tablas INSCRIPCIÓN, ASIGNACIÓN y ACADÉMICO así como la relación existente entre ellas. La lógica a seguir es encontrar las calificaciones en los grupos correspondientes a la asignatura Ecuaciones diferenciales, verificar la calificación que cada alumno obtuvo en el curso, de esas calificaciones detectar cuales son iguales o menores a 5 para etiquetarlas de reprobadas, realizar el conteo, por grupos, de número de alumnos reprobados, saber el nombre de los profesores que impartieron los grupos de la asignatura y por último ordenar a dichos profesores en orden descendente de número de alumnos reprobados.

La consulta sería la siguiente:

```
select ac.nombre,ac.ap_paterno,ac.ap_materno,count(i.calificacion) as
alumnos_reprobados
from inscripcion i, asignacion a, academico ac
where a.clave_asignatura='1306'
and a.no_trabajador=ac.no_trabajador
and a.clave_asignacion=i.clave_asignacion
and i.calificacion<=5
group by ac.nombre,ac.ap_paterno,ac.ap_materno
order by 4 desc;
```

Esta consulta dará el nombre de todos los académicos que tienen alumnos reprobados en la asignatura de Ecuaciones diferenciales, ordenados del que tiene mayor número de reprobados al menor. El resultado de la consulta se puede observar en la imagen 4.17

```
SQL> select ac.nombre,ac.ap_paterno,ac.ap_materno,count(i.calificacion) as alumnos_reprobados
from inscripcion i, asignacion a, academico ac
where a.clave_asignatura='1306'
and a.no_trabajador=ac.no_trabajador
and a.clave_asignacion=i.clave_asignacion
and i.calificacion<=5
group by ac.nombre,ac.ap_paterno,ac.ap_materno
order by 4 desc;  2    3    4    5    6    7    8

NOMBRE                         AP_PATERNO      AP_MATERNO      ALUMNOS_REPROBADOS
------------------------------ --------------- --------------- ------------------
Lucio                          Álvarez         García                           2
```

Imagen 4.17 Profesor con mayor número de alumnos reprobados

Consulta 7

Para la siguiente consulta se requiere tomar en cuenta el enunciado: "*Horario de la alumna Alejandra Noriega Torres con número de cuenta 306293629 para este semestre con asignaturas, grupos, profesores, horarios y salones*".

Esta consulta es muy práctica ya que su resultado será el horario que el alumno tendrá a lo largo de su semestre. Las tablas necesarias para construir la consulta serán asignatura,

ordinario, académico, salón, inscripción, asignación y alumno. Las condiciones a seguir son que el número de cuenta del alumno coincida con el especificado y el semestre sea del que se necesita el horario.

Para realizar la consulta se ejecutará lo siguiente:

```
select asi.nombre,a.periodo, o.grupo, ac.nombre||' '||ac.ap_paterno||'
'||ac.ap_materno as Profesor,
o.dias, to_char(o.hora_inicio,'HH24:MI') hora_inicio,
to_char(o.hora_fin,'HH24:MI') hora_fin, s.clave_salon
from asignatura asi, ordinario o, academico ac, salon s, inscripcion i,
asignacion a, alumno al
where al.no_cta=i.no_cta
and i.clave_asignacion=a.clave_asignacion
and a.clave_asignacion=o.clave_asignacion
and a.no_trabajador=ac.no_trabajador
and a.clave_salon=s.clave_salon
and a.clave_asignatura=asi.clave_asignatura
and al.no_cta='306293629'
and a.periodo='162';
```

Siendo el resultado de dicha consulta el mostrado en la imagen 4.18

```
SQL> select asi.nombre,a.periodo, o.grupo, ac.nombre||' '||ac.ap_paterno||' '||ac.ap_materno as Profesor, o.dias,
to_char(o.hora_inicio,'HH24:MI') hora_inicio,
to_char(o.hora_fin,'HH24:MI') hora_fin, s.clave_salon
from asignatura asi, ordinario o, academico ac, salon s, inscripcion i, asignacion a, alumno al where al.no_cta=i.no_cta
and i.clave_asignacion=a.clave_asignacion
and a.clave_asignacion=o.clave_asignacion
and a.no_trabajador=ac.no_trabajador
and a.clave_salon=s.clave_salon
and a.clave_asignatura=asi.clave_asignatura
and al.no_cta='306293629'
and a.periodo='162';  2    3    4    5    6    7    8    9   10   11

NOMBRE                                                    PER      GRUPO PROFESOR                      DIAS   HORA_ HORA_ CLAV
--------------------------------------------------------------------------------------------------------------------------------
Cálculo Vectorial                                         162          1 Marcos Bolivar Becerra        101010 08:30 10:00 A102
Cinemática y Dinámica                                     162          1 Yadhira Carrera Jiménez       101010 10:00 11:30 A103
Ecuaciones Diferenciales                                  162          1 Lucio Alvarez García          101010 07:00 08:30 A101
Programación Avanzada y Métodos Numéricos                 162          1 Julián Foca Pinto             101000 13:00 14:30 A105
Principios de Termodinámica y Electromagnetismo           162          1 Sandra Esquivel Nunez         101010 11:30 13:00 A104
Laboratorio de Principios de Termodinámica y Electromagnetismo 162     2 Sandra Esquivel Nunez         000001 07:00 09:00 H002
```

Imagen 4.18 Horario de la alumna Alejandra Noriega Torres

Como se observa en la imagen 4.18 se obtiene el listado de asignaturas que la alumna inscribió para el semestre en curso así como el grupo elegido, el profesor que impartirá la clase, los días y horarios programados y el salón donde tendrá que tomar el curso. Se podrían agregar otros campos que se consideren relevantes como lo puede ser la clave de la asignatura, esto modificando ligeramente la consulta.

Consulta 8

Para la siguiente consulta a probar, se tiene el enunciado: "*Listado de todos los grupos que impartirá la profesora Sandra Esquivel Núñez con periodo, asignatura, grupo, horario, salón y número de alumnos inscritos a cada asignatura".*

La estructura de esta consulta es muy parecida a la anterior, pero en este caso el horario que necesitamos saber es el de un profesor y no el de un alumno. Las tablas que necesitamos relacionar son ASIGNACIÓN, ASIGNATURA, ORDINARIO, ACADÉMICO, SALÓN e INSCRIPCIÓN.

Para resolver esto, ejecutaremos la siguiente consulta:

```
select a.periodo, asi.nombre, o.grupo, to_char(o.hora_inicio, 'HH24:MI')
hora_inicio, to_char(o.hora_fin, 'HH24:MI') hora_fin, o.dias,
s.clave_salon, count(i.no_cta) alumnos_inscritos
from asignacion a, asignatura asi, ordinario o, academico ac, salon s,
inscripcion i
where ac.no_trabajador='123468'
and a.periodo='162'
and ac.no_trabajador=a.no_trabajador
and a.clave_asignatura=asi.clave_asignatura
and a.clave_salon=s.clave_salon
and a.clave_asignacion=i.clave_asignacion
group by a.periodo, asi.nombre, o.grupo, to_char(o.hora_inicio,
'HH24:MI'), to_char(o.hora_fin, 'HH24:MI'),o.dias, s.clave_salon;
```

El resultado de dicha consulta puede ser observado en la imagen 4.19

```
SQL> select a.periodo, asi.nombre,o.grupo, to_char(o.hora_inicio,'HH24:MI') hora_inicio, to_char(o.hora_fin,'HH24:MI')
hora_fin, o.dias, s.clave_salon, count(i.no_cta) alumnos_inscritos
from asignacion a, asignatura asi, ordinario o, academico ac, salon s, inscripcion i
where ac.no_trabajador='123468'
and a.periodo='162'
and ac.no_trabajador=a.no_trabajador
and a.clave_asignacion=o.clave_asignacion
and a.clave_asignatura=asi.clave_asignatura
and a.clave_salon=s.clave_salon
and a.clave_asignacion=i.clave_asignacion
group by a.periodo, asi.nombre,a.clave_asignacion,o.grupo, to_char(o.hora_inicio,'HH24:MI'),
to_char(o.hora_fin,'HH24:MI'),o.dias, s.clave_salon;  2    3    4    5    6    7    8    9   10   11   12

PER NOMBRE                                                          GRUPO HORA_ HORA_ DIAS   CLAV ALUMNOS_INSCRITOS
--- ---------------------------------------------------------------- ---------- ----- ----- ------ ---- ----------------
162 Principios de Termodinámica y Electromagnetismo                      1 11:30 13:00 101010 A104                10
162 Laboratorio de Principios de Termodinámica y Electromagnetismo       2 07:00 09:00 000001 H002                20
```

Imagen 4.19 Horario de la profesora Sandra Esquivel Núñez

De esta forma y como se puede observar en la imagen 4.19 se muestran los nombres de las asignaturas y el número de grupo que tendrá que impartir el profesor durante el semestre en turno así como el horario y días en que cada materia se llevará a cabo. Adicionalmente se hace un conteo de los alumnos que se inscribieron en ese grupo.

Consulta 9

El enunciado a tomar en cuenta para la próxima consulta es el siguiente: *"Porcentaje de alumnos reprobados para la asignatura Ecuaciones diferenciales durante este periodo"*.

Esta consulta es parecida a la obtenida en la número 4, pero aquí en lugar de agruparla por profesor se hará por asignatura y no se mostrará el número de alumnos reprobados, sino que ese cálculo se realizará de forma interna y en base a ese resultado se obtendrá un porcentaje que será el de alumnos reprobados.

Estructurando la consulta por pasos, lo primero que se tiene que hacer es obtener el número de alumnos totales que inscribieron la asignatura de Ecuaciones diferenciales durante el periodo actual, esto a través de la siguiente consulta:

```
select count(i.calificacion)
from inscripcion i, asignatura asi, asignacion a
where asi.clave_asignatura='1306'
and asi.clave_asignatura=a.clave_asignatura
and a.periodo='162'
and a.clave_asignacion=i.clave_asignacion;
```

De esta consulta obtendremos un número que será el total de alumnos que inscribieron la asignatura para este periodo, esto se muestra en la imagen 4.20

```
SQL> select count(i.calificacion)
from inscripcion i, asignatura asi, asignacion a
where asi.clave_asignatura='1306'
and asi.clave_asignatura=a.clave_asignatura
and a.periodo='162'
and a.clave_asignacion=i.clave_asignacion;  2

COUNT(I.CALIFICACION)
---------------------
                   10
```

Imagen 4.20 Alumnos inscritos

El siguiente paso es obtener el número de alumnos que tienen una calificación menor o igual a 5 durante el periodo actual en la asignatura Ecuaciones diferenciales. Para eso ejecutaremos la siguiente consulta:

```
select count(i.calificacion)
from inscripcion i, asignatura asi, asignacion a
where asi.clave_asignatura='1306'
and asi.clave_asignatura=a.clave_asignatura
and a.clave_asignacion=i.clave_asignacion
and a.periodo='162'
and i.calificacion<=5;
```

De esta forma obtendremos el resultado de los alumnos que no aprobaron la asignatura. Estos e puede observar en la imagen 4.21

```
SQL> select count(i.calificacion)
from inscripcion i, asignatura asi, asignacion a
where asi.clave_asignatura='1306'
and asi.clave_asignatura=a.clave_asignatura
and a.periodo='162'
and a.clave_asignacion=i.clave_asignacion
and i.calificacion<=5;  2    3    4    5    6

COUNT(I.CALIFICACION)
---------------------
                    2
```

Imagen 4.21 Alumnos reprobados

Por último, queda utilizar los números obtenidos en las dos consultas anteriores para obtener el porcentaje de alumnos reprobados durante este semestre, esto a través de la siguiente regla de tres:

Tabla 4.4 Regla de tres

Alumnos inscritos	100%
Alumnos reprobados	X

Por lo que se tiene que multiplicar el número de alumnos reprobados por 100 y este resultado dividirlo entre el número de alumnos inscritos, quedando la ecuación de la siguiente manera:

X=((reprobados*100)/inscritos)

Esta ecuación puede ser planteada dentro de un select general que indique el nombre de la asignatura y la operación realizada entre las dos primeras consultas trabajadas.

La consulta que da el resultado total sería la siguiente:

```
select nombre,(((select count(i.calificacion)
            from inscripcion i, asignatura asi, asignacion a
            where asi.clave_asignatura='1306'
      and asi.clave_asignatura=a.clave_asignatura
            and a.clave_asignacion=i.clave_asignacion
      and a.periodo='162'
            and i.calificacion<=5)*100)/
      (select count(i.calificacion)
            from inscripcion i, asignatura asi, asignacion a
            where asi.clave_asignatura='1306'
            and asi.clave_asignatura=a.clave_asignatura
            and a.periodo='162'
            and a.clave_asignacion=i.clave_asignacion)) as
            "PORCENTAJE ALUMNOS REPROBADOS"
from asignatura
where clave_asignatura='1306';
```

Como se puede observar, las dos primeras consultas trabajadas van dentro de un select general en el cual se realiza la operación. El resultado de la consulta se puede ver en la imagen 4.22

```
SQL> select nombre,(((select count(i.calificacion)
from inscripcion i, asignatura asi, asignacion a
where asi.clave_asignatura='1306'
and asi.clave_asignatura=a.clave_asignatura
and a.clave_asignacion=i.clave_asignacion
and a.periodo='162'
and i.calificacion<=5)*100)/(select count(i.calificacion)
from inscripcion i, asignatura asi, asignacion a
where asi.clave_asignatura='1306'
and asi.clave_asignatura=a.clave_asignatura
and a.periodo='162'
and a.clave_asignacion=i.clave_asignacion)) as "PORCENTAJE ALUMNOS REPROBADOS"
from asignatura
where clave_asignatura='1306';   2    3    4    5    6    7    8    9   10   11

NOMBRE                    PORCENTAJE ALUMNOS REPROBADOS
------------------------- ----------------------------
Ecuaciones Diferenciales                            20
```

Imagen 4.22 Porcentaje de alumnos reprobados en Ecuaciones diferenciales

Consulta 10

Para culminar con las pruebas de consultas a la base de datos, se resolverá la consulta basada en el siguiente enunciado: "*Lista de departamentos de la Facultad de Ingeniería ordenada por la división a la que pertenece cada uno*".

Para resolver esta consulta se utilizarán las tablas DEPARTAMENTO y COORDINACIÓN, relacionándolas a través del atributo clave_coordinacion que en la tabla coordinación sirve como llave primaria mientras que para departamento funge como llave foránea. La consulta para obtener dicho resultado es la siguiente:

```
select c.nombre as coordinacion, d.nombre as departamento
from coordinación c, departamento d
where clave_coordinacion=d.clave_coordinacion
order by 1;
```

El resultado de la consulta es el mostrado en la imagen 4.23

```
SQL> select c.nombre, d.nombre from coordinacion c, departamento d
where c.clave_coordinacion=d.clave_coordinacion
order by 1;  2    3

NOMBRE                                                                           NOMBRE
-------------------------------------------------------------------------------- --------------------------------------------------------------------------------
Coodinación de Ingenieria de Sistemas                                            Departamento de Planeación. Investigación de Operaciones e Ingenieria Industrial
Coodinación de Ingenieria de Sistemas                                            Departamento de Transporte y Optimización Financiera
Coordinación de Ciencias Aplicadas                                               Departamento de Ecuaciones Diferenciales
Coordinación de Ciencias Aplicadas                                               Departamento de Estática
Coordinación de Ciencias Aplicadas                                               Departamento de Cinemática y Dinámica
Coordinación de Ciencias Aplicadas                                               Departamento de Probabilidad y Estadtística
Coordinación de Ciencias Aplicadas                                               Departamento de Matemáticas Avanzadas, Análisis Numéricos y Dibujo
Coordinación de Ciencias Sociales y Humanidades                                  Departamento de Asignaturas de la DCSyH
Coordinación de Física General y Química                                         Departamento de Termodinámica

NOMBRE                                                                           NOMBRE
-------------------------------------------------------------------------------- --------------------------------------------------------------------------------
Coordinación de Física General y Química                                         Departamento de Química
Coordinación de Física General y Química                                         Departamento de Física Experimental y Óptica
Coordinación de Física General y Química                                         Departamento de Electricidad y Magnetismo
Coordinación de Ingeniería Civil                                                 Departamento de Hidráulica
Coordinación de Ingeniería Civil                                                 Departamento de Sistemas, Planeación y Transporte
Coordinación de Ingeniería Civil                                                 Departamento de Ingeniería Sanitaria y Ambiental
Coordinación de Ingeniería Civil                                                 Deparamento de Geotecnia
Coordinación de Ingeniería Civil                                                 Departamento de Construcción
Coordinación de Ingeniería Civil                                                 Departamento de Estructuras
Coordinación de Ingeniería Civil y Geomática                                     Departamento de Proyectos
Coordinación de Ingeniería Civil y Geomática                                     Departamento de Ingeniería Civil y Geomática

NOMBRE                                                                           NOMBRE
-------------------------------------------------------------------------------- --------------------------------------------------------------------------------
Coordinación de Ingeniería Eléctrica y Electrónica                               Departamento de Sistemas Energéticos
Coordinación de Ingeniería Eléctrica y Electrónica                               Departamento de Ingeniería Eléctrica de Potencia
Coordinación de Ingeniería Eléctrica y Electrónica                               Departamento de Ingeniería Electrónica
Coordinación de Ingeniería Eléctrica y Electrónica                               Departamento de Ingeniería de Control y Robótica
Coordinación de Ingeniería Eléctrica y Electrónica                               Departamento de Procesamiento de Señales
Coordinación de Ingeniería Geofísica                                             Departamento de la Carrera de Ingeniería Geofísica
Coordinación de Ingeniería Geológica                                             Departamento de la Carrera de Ingeniería Geológica
Coordinación de Ingeniería Geológica                                             Departamento de Explotación del Petróleo
Coordinación de Ingeniería Geomática                                             Departamento de Fotogrametría y Geodesia
Coordinación de Ingeniería Geomática                                             Departamento de Sistemas de Información Geográfica y Topográfica
Coordinación de Ingeniería Industrial                                            Departamento de Ingeniería Industrial

NOMBRE                                                                           NOMBRE
-------------------------------------------------------------------------------- --------------------------------------------------------------------------------
Coordinación de Ingeniería Mecatrónica                                           Departamento de Ingeniería Mecatrónica
Coordinación de Ingeniería Mecatrónica                                           Departamento de Ingeniería Mecánica
Coordinación de Ingeniería Mecánica e Industrial                                 Departamento de Termofluidos
Coordinación de Ingeniería Mecánica e Industrial                                 Departamento de Cómputo
Coordinación de Ingeniería de Minas y Metalurgista                               Departamento de la Carrera de Ingeniería Minera y Metalurgista
Coordinación de Ingeniería en Computación                                        Departamento de Ingeniería en Computación
Coordinación de Ingeniería en Telecomunicaciones                                 Departamento de Ingeniería en Telecomunicaciones
Coordinación de Matemáticas                                                      Departamento de Álgebra Lineal
Coordinación de Matemáticas                                                      Departamento de Geometría Analítica
Coordinación de Matemáticas                                                      Departamento de Álgebra
Coordinación de Matemáticas                                                      Departamento de Cálculo Vectorial

NOMBRE                                                                           NOMBRE
-------------------------------------------------------------------------------- --------------------------------------------------------------------------------
Coordinación de Matemáticas                                                      Departamento de Cálculo Diferencial
Coordinación de Matemáticas                                                      Departamento de Cálculo Integral
Coordinación de Sistemas de Apoyo Docente y Difusión Cultural                    Departamento de Actividades Socioculturales

45 rows selected.
```

Imagen 4.23 Lista de departamentos

Se puede observar que se muestra los departamentos y la coordinación a la que pertenecen, ordenados por el nombre de la coordinación en cuestión.

Cabe mencionar que todas las consultas plantadas en este apartado que tienen que ver con asignaturas, fue tomando en cuenta solamente las asignaciones ordinarias, por lo que si se deseara agregar la parte de extraodinarios simplemente se tendría que hacer la modificación conveniente en cada consulta para tomar en cuenta este tipo de asignación, siendo muy similar a lo mostrado para el caso de las asignaciones ordinarias.

Con esto podemos concluir que todas las consultas planteadas al inicio de este trabajo, son factibles de resolver por la base de datos a través de las diversas relaciones con las que se cuentan y la estructura de las tablas.

CAPÍTULO V
Conclusiones

Como resultado de las etapas trabajadas para el diseño del Sistema Integral de Información de la Facultad de Ingeniería, se pueden obtener las siguientes conclusiones:

-Una de las etapas más importantes para el desarrollo de este proyecto es el proceso de análisis que incluye el levantamiento de requerimientos y necesidades que la institución tiene, esto ayudó a plantear los objetivos que dicho proyecto tendrá. Comprender esta etapa es esencial ya que de malentender los requerimientos y/o necesidades que se tienen se proyectará en etapas posteriores teniendo como resultado un mal diseño que será incapaz de cumplir con todos los objetivos.

-Al estudiar el planteamiento del problema se encontró que el avance tecnológico es un tema fundamental para el diseño de un sistema ya que la limitación de recursos puede repercutir en el cumplimiento de objetivos. Para poder atender las necesidades y requerimientos con que se cuentan es menester contar con la tecnología adecuada para cumplir dicho fin.

-Para poder plantear una estrategia de trabajo, se analizaron varios caminos a seguir, definiendo cuales eran los que más se adaptaban a los recursos y necesidades con los que se cuentan. De esta forma se eligió el modelo relacional como modelo de base de datos a seguir auxiliándonos de un diagrama entidad relación para la construcción del diseño de la base de datos.

-Una de las cuestiones que se encontraron al trabajar con un modelo relacional fue la cuestión de la normalización que nos ayuda a evitar la existencia de inconsistencias y anomalías lógicas en las tablas que conforman a la base de datos. Como se observa en el apartado *1.4 "Normalización"*, existen diversas formas normales que nos ayudan a alcanzar un diseño de bases de datos confiable. Pero al poner en práctica la teoría de normalización en el diseño de la base de datos se encontró que entre más alta sea la forma normal que se aplica al diseño, más complicado es cumplir con los requerimientos en la estructura de los atributos y las relaciones entre tablas. Uno de los problemas encontrados a la hora de normalizar es el caso encontrado en la sección *3.7 "Diseño normalizado"* en el que uno de los atributos de la tabla PLAN-ASIGNATURA tiene que incumplir la tercera forma normal para de esta forma satisfacer las necesidades del sistema. De lo cual podemos decir que nos vimos forzados a tener una ligera flexibilidad en el cumplimiento de dicha forma normal para realizar un diseño acorde a las necesidades presentadas.

-Dentro del mismo tema de la normalización, se concluyó (ayudándonos de lo que dice la teoría) que para que para que nuestro diseño de base de datos sea confiable, tiene que alcanzar la tercera forma normal, por lo que el diseño presentado se encuentra en dicha forma normal con la flexibilidad anteriormente mencionada.

-La definición de atributos llave fue una tarea que permitió generar un modelo relacional, ya que fueron las llaves las que nos permitieron llevar a cabo las relaciones entre entidades y conformar así el diagrama entidad relación de la base de datos.

-Una de las necesidades a cumplir que se definió, fue que el diseño fuese capaz de interrelacionar información para auxiliar a procesos incluidos en el control escolar por lo que una vez que se realizó el diseño del diagrama entidad relación, se llevaron a cabo pruebas de integración como lo fue el proceso de inscripción, de historial y el de asignación con lo cual se comprobó la factibilidad con la que el diseño cuenta para cumplir con la necesidad de integrar procesos que hagan uso del núcleo de la base de datos.

-Las pruebas sirvieron para afinar detalles en la etapa de desarrollo y mejorar cuestiones que fuesen más prácticas para la obtención de información de la base de datos diseñada.

-Con la etapa de pruebas se logró detectar algunas consideraciones semánticas que se deben tomar en cuenta para poder agregar al sistema y que deben congeniar con el diseño de la base de datos. Dichas consideraciones semánticas fueron expuestas en el apartado *3.8 "Consideraciones semánticas"*.

-El avance en la tecnología puede provocar que las necesidades sean cambiantes con el paso del tiempo por lo que se pude concluir que el diseño de base de datos presentado en este proyecto cumple con los requerimientos y necesidades que actualmente tiene nuestra institución, pero esto no lo exime de que dicho sistema se vuelva obsoleto una vez que la tecnología avance y las necesidades cambien.

-Como lo dice el título de este trabajo, se trata de un diseño de base de datos que servirá para un sistema, por lo que se busca que dicho sistema llegue a servir a la Facultad de Ingeniería auxiliando con el control escolar de la institución. Para que este fin se pueda cumplir, la base de datos tendrá que pasar por un proceso ETL (Extract, Transform and Load). Este proceso consiste en extraer información de la base de datos que actualmente es utilizada para el control escolar, dar el formato requerido a estos datos para posteriormente ser cargados en la nueva base de datos diseñada.

- Retomando los objetivos presentados al inicio de este trabajo podemos concluir que se ha diseñado la base de datos para el Sistema Integral de Información de la Facultad de Ingeniería cumpliendo con los requerimientos y necesidades presentadas. Este diseño es estructurado para permitir la escalabilidad del mismo, es decir, si fuese menester añadir información extra al diseño construido se podrán hacer modificaciones sin alterar el núcleo de información. El diseño está normalizando hasta la tercera forma normal lo que ayuda a evitar la existencia de inconsistencias y anomalías en las tablas. De esta forma se aprovecha la tecnología con la que cuenta la Facultad de Ingeniería cumpliendo con las nuevas necesidades que han surgido desde el último diseño que sirvió para la base de datos creada hace años, que sufría por limitantes de memoria por lo que no fue posible construir un diseño normalizado.

- Con todo lo anterior se puede concluir que los objetos propuestos en este trabajo se han cumplido y se puede tomar el diseño propuesto para dar inicio a la creación de SIIFI, tomando en cuenta las consideraciones que se deben tomar en cuenta para dicho fin.

REFERENCIAS

1: Introducción a los sistemas de bases de datos, C. J. Date. 7ma. ed.

2: Artículo "Características y tipos de bases de datos" (www.ibm.com)

3: Artículo "Historia de las bases de datos" Museo Informática (Universidad Politécnica de Valencia) (http://museo.inf.upv.es/)

4: CODASYL (Conference on Data Systems Languages) era un consorcio de industrias informáticas que tenían como objetivo la regularización de un lenguaje de programación estándar que pudiera ser utilizado en multitud de ordenadores.

5: Anexo Técnico 1 "A relational model of data for large shared data banks" by Edgar F. Codd"

6: Artículo "Modelo de datos jerárquico y de red" Universidad de Carabobo (2005)

7: Diccionario de la lengua española

8: Artículo "Informática y computación" Instituto Universitario del Estado de México (IUEM)

9: Artículo "Modelos de bases de datos" Universidad Politécnica de Valencia

10: Artículo "Fundamentos de bases de datos" Tecnológico de Pachuca

11: "Notas de bases de datos 2014" Facultad de Ciencias de la Computación, Benemérita Universidad Autónoma de Puebla

12: Notas de curos "Conceptos básicos de Programación Orientada a Objetos" Universidad Central de Venezuela

13: Artículo "Modelo Orientado a Objetos" Informáticos de la Generalitat Valenciana

14: Artículo "El modelo entidad-relación hacia una visión unificada de los datos" Peter Chen (1976)

15: Introducción a las bases de datos (4ta edición) – Universidad Politécnica de la Región Ribereña

16: Artículo "Diagrama Entidad Relación" Universidad Nacional Autónoma de Nicaragua

17: Artículo "Modelo entidad-relación" Atlantic International University (AIU)

18: Artículo "Conceptos fundamentales y terminología utilizada en el diseño y la implementación de una base de datos" Universidad Autónoma del Estado de México

19: Diseño de Bases de Datos (2da edición) – Universidad Interamericana para el Desarrollo

20: Artículo "Normalización de tablas relacionales" Universidad Autónoma del Estado de Hidalgo

21: Artículo "Normalización" Escuela Especializada en Ingeniería ITCA-FEPADE

22: Fundamentos de bases de datos (4ta edición) – Abraham Silberschats, Henry F. Korth, S. Sundarshan

23: Curso "Introducción a los Sistemas Distribuidos" Departamento de Sistemas e Informática, Universidad Nacional del Rosario

24: Fundamentos de SQL (3ra edición) – Andy Oppel, Robert Sheldon

25: Artículo "Vistas" Microsoft TechNet

26: Anexo técnico 2 "Cramming more components onto integrated circuits" by Gordon E. Moore

27: En planes anteriores algunas asignaturas sólo tenían coordinación y no departamento

28: Etapas propuestas para el propósito de este trabajo.

29: Correos de México (www.correosdemexico.gob.mx)

30: Blog especializado en base de datos

31: Artículo “Notaciones de entidad-relación de bases de datos” Universidad Nacional de Trujillo

32: Administración de bases de datos, diseño y desarrollo de aplicaciones, Michael V. Mannino (3ra edición).

ANEXO TÉCNICO 1

A relational model of data for large shared data banks

by Edgar F. Codd

A Relational Model of Data for Large Shared Data Banks

E. F. CODD
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data. In Section 1, inadequacies of these models are discussed. A model based on *n*-ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed and applied to the problems of redundancy and consistency in the user's model.

KEY WORDS AND PHRASES: data bank, data base, data structure, data organization, hierarchies of data, networks of data, relations, derivability, redundancy, consistency, composition, join, retrieval language, predicate calculus, security, data integrity
CR CATEGORIES: 3.70, 3.73, 3.75, 4.20, 4.22, 4.29

1. Relational Model and Normal Form

1.1. Introduction

This paper is concerned with the application of elementary relation theory to systems which provide shared access to large banks of formatted data. Except for a paper by Childs [1], the principal application of relations to data systems has been to deductive question-answering systems. Levein and Maron [2] provide numerous references to work in this area.

In contrast, the problems treated here are those of *data independence*—the independence of application programs and terminal activities from growth in data types and changes in data representation—and certain kinds of *data inconsistency* which are expected to become troublesome even in nondeductive systems.

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for noninferential systems. It provides a means of describing data with its natural structure only—that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations—these are discussed in Section 2. The network model, on the other hand, has spawned a number of confusions, not the least of which is mistaking the derivation of connections for the derivation of relations (see remarks in Section 2 on the "connection trap").

Finally, the relational view permits a clearer evaluation of the scope and logical limitations of present formatted data systems, and also the relative merits (from a logical standpoint) of competing representations of data within a single system. Examples of this clearer perspective are cited in various parts of this paper. Implementations of systems to support the relational model are not discussed.

1.2. Data Dependencies in Present Systems

The provision of data description tables in recently developed information systems represents a major advance toward the goal of data independence [5, 6, 7]. Such tables facilitate changing certain characteristics of the data representation stored in a data bank. However, the variety of data representation characteristics which can be changed *without logically impairing some application programs* is still quite limited. Further, the model of data with which users interact is still cluttered with representational properties, particularly in regard to the representation of collections of data (as opposed to individual items). Three of the principal kinds of data dependencies which still need to be removed are: ordering dependence, indexing dependence, and access path dependence. In some systems these dependencies are not clearly separable from one another.

1.2.1. *Ordering Dependence.* Elements of data in a data bank may be stored in a variety of ways, some involving no concern for ordering, some permitting each element to participate in one ordering only, others permitting each element to participate in several orderings. Let us consider those existing systems which either require or permit data elements to be stored in at least one total ordering which is closely associated with the hardware-determined ordering of addresses. For example, the records of a file concerning parts might be stored in ascending order by part serial number. Such systems normally permit application programs to assume that the order of presentation of records from such a file is identical to (or is a subordering of) the

stored ordering. Those application programs which take advantage of the stored ordering of a file are likely to fail to operate correctly if for some reason it becomes necessary to replace that ordering by a different one. Similar remarks hold for a stored ordering implemented by means of pointers.

It is unnecessary to single out any system as an example, because all the well-known information systems that are marketed today fail to make a clear distinction between order of presentation on the one hand and stored ordering on the other. Significant implementation problems must be solved to provide this kind of independence.

1.2.2. *Indexing Dependence.* In the context of formatted data, an index is usually thought of as a purely performance-oriented component of the data representation. It tends to improve response to queries and updates and, at the same time, slow down response to insertions and deletions. From an informational standpoint, an index is a redundant component of the data representation. If a system uses indices at all and if it is to perform well in an environment with changing patterns of activity on the data bank, an ability to create and destroy indices from time to time will probably be necessary. The question then arises: Can application programs and terminal activities remain invariant as indices come and go?

Present formatted data systems take widely different approaches to indexing. TDMS [7] unconditionally provides indexing on all attributes. The presently released version of IMS [5] provides the user with a choice for each file: a choice between no indexing at all (the hierarchic sequential organization) or indexing on the primary key only (the hierarchic indexed sequential organization). In neither case is the user's application logic dependent on the existence of the unconditionally provided indices. ID8 [8], however, permits the file designers to select attributes to be indexed and to incorporate indices into the file structure by means of additional chains. Application programs taking advantage of the performance benefit of these indexing chains must refer to those chains by name. Such programs do not operate correctly if these chains are later removed.

1.2.3. *Access Path Dependence.* Many of the existing formatted data systems provide users with tree-structured files or slightly more general network models of the data. Application programs developed to work with these systems tend to be logically impaired if the trees or networks are changed in structure. A simple example follows.

Suppose the data bank contains information about parts and projects. For each part, the part number, part name, part description, quantity-on-hand, and quantity-on-order are recorded. For each project, the project number, project name, project description are recorded. Whenever a project makes use of a certain part, the quantity of that part committed to the given project is also recorded. Suppose that the system requires the user or file designer to declare or define the data in terms of tree structures. Then, any one of the hierarchical structures may be adopted for the information mentioned above (see Structures 1–5).

Structure 1. Projects Subordinate to Parts

File	*Segment*	*Fields*
F	PART	part #
		part name
		part description
		quantity-on-hand
		quantity-on-order
	PROJECT	project #
		project name
		project description
		quantity committed

Structure 2. Parts Subordinate to Projects

File	*Segment*	*Fields*
F	PROJECT	project #
		project name
		project description
	PART	part #
		part name
		part description
		quantity-on-hand
		quantity-on-order
		quantity committed

Structure 3. Parts and Projects as Peers Commitment Relationship Subordinate to Projects

File	*Segment*	*Fields*
F	PART	part #
		part name
		part description
		quantity-on-hand
		quantity-on-order
G	PROJECT	project #
		project name
		project description
	PART	part #
		quantity committed

Structure 4. Parts and Projects as Peers Commitment Relationship Subordinate to Parts

File	*Segment*	*Fields*
F	PART	part #
		part description
		quantity-on-hand
		quantity-on-order
	PROJECT	project #
		quantity committed
G	PROJECT	project #
		project name
		project description

Structure 5. Parts, Projects, and Commitment Relationship as Peers

File	*Segment*	*Fields*
F	PART	part #
		part name
		part description
		quantity-on-hand
		quantity-on-order
G	PROJECT	project #
		project name
		project description
H	COMMIT	part #
		project #
		quantity committed

Now, consider the problem of printing out the part number, part name, and quantity committed for every part used in the project whose project name is "alpha." The following observations may be made regardless of which available tree-oriented information system is selected to tackle this problem. If a program P is developed for this problem assuming one of the five structures above—that is, P makes no test to determine which structure is in effect—then P will fail on at least three of the remaining structures. More specifically, if P succeeds with structure 5, it will fail with all the others; if P succeeds with structure 3 or 4, it will fail with at least 1, 2, and 5; if P succeeds with 1 or 2, it will fail with at least 3, 4, and 5. The reason is simple in each case. In the absence of a test to determine which structure is in effect, P fails because an attempt is made to execute a reference to a nonexistent file (available systems treat this as an error) or no attempt is made to execute a reference to a file containing needed information. The reader who is not convinced should develop sample programs for this simple problem.

Since, in general, it is not practical to develop application programs which test for all tree structurings permitted by the system, these programs fail when a change in structure becomes necessary.

Systems which provide users with a network model of the data run into similar difficulties. In both the tree and network cases, the user (or his program) is required to exploit a collection of user access paths to the data. It does not matter whether these paths are in close correspondence with pointer-defined paths in the stored representation—in IDS the correspondence is extremely simple, in TDMS it is just the opposite. The consequence, regardless of the stored representation, is that terminal activities and programs become dependent on the continued existence of the user access paths.

One solution to this is to adopt the policy that once a user access path is defined it will not be made obsolete until all application programs using that path have become obsolete. Such a policy is not practical, because the number of access paths in the total model for the community of users of a data bank would eventually become excessively large.

1.3. A Relational View of Data

The term *relation* is used here in its accepted mathematical sense. Given sets $S_1, S_2, \cdots, S_n$ (not necessarily distinct), R is a relation on these n sets if it is a set of n-tuples each of which has its first element from S_1, its second element from S_2, and so on.[1] We shall refer to S_j as the jth *domain* of R. As defined above, R is said to have *degree* n. Relations of degree 1 are often called *unary*, degree 2 *binary*, degree 3 *ternary*, and degree n *n-ary*.

For expository reasons, we shall frequently make use of an array representation of relations, but it must be remembered that this particular representation is not an essential part of the relational view being expounded. An array which represents an n-ary relation R has the following properties:

(1) Each row represents an n-tuple of R.
(2) The ordering of rows is immaterial.
(3) All rows are distinct.
(4) The ordering of columns is significant—it corresponds to the ordering $S_1, S_2, \cdots, S_n$ of the domains on which R is defined (see, however, remarks below on domain-ordered and domain-unordered relations).
(5) The significance of each column is partially conveyed by labeling it with the name of the corresponding domain.

The example in Figure 1 illustrates a relation of degree 4, called *supply*, which reflects the shipments-in-progress of parts from specified suppliers to specified projects in specified quantities.

supply	(*supplier*	*part*	*project*	*quantity*)
	1	2	5	17
	1	3	5	23
	2	2	7	9
	2	7	5	4
	4	1	1	12

Fig. 1. A relation of degree 4

One might ask: If the columns are labeled by the name of corresponding domains, why should the ordering of columns matter? As the example in Figure 2 shows, two columns may have identical headings (indicating identical domains) but possess distinct meanings with respect to the relation. The relation depicted is called *component*. It is a ternary relation, whose first two domains are called *part* and third domain is called *quantity*. The meaning of *component* (x, y, z) is that part x is an immediate component (or subassembly) of part y, and z units of part x are needed to assemble one unit of part y. It is a relation which plays a critical role in the parts explosion problem.

component	(*part*	*part*	*quantity*)
	1	5	9
	2	5	7
	3	5	2
	2	6	12
	3	6	3
	1	7	1
	6	7	1

Fig. 2. A relation with two identical domains

It is a remarkable fact that several existing information systems (chiefly those based on tree-structured files) fail to provide data representations for relations which have two or more identical domains. The present version of IMS/360 [5] is an example of such a system.

The totality of data in a data bank may be viewed as a collection of time-varying relations. These relations are of assorted degrees. As time progresses, each n-ary relation may be subject to insertion of additional n-tuples, deletion of existing ones, and alteration of components of any of its existing n-tuples.

[1] More concisely, R is a subset of the Cartesian product $S_1 \times S_2 \times \cdots \times S_n$.

In many commercial, governmental, and scientific data banks, however, some of the relations are of quite high degree (a degree of 30 is not at all uncommon). Users should not normally be burdened with remembering the domain ordering of any relation (for example, the ordering *supplier*, then *part*, then *project*, then *quantity* in the relation *supply*). Accordingly, we propose that users deal, not with relations which are domain-ordered, but with *relationships* which are their domain-unordered counterparts.[2] To accomplish this, domains must be uniquely identifiable at least within any given relation, without using position. Thus, where there are two or more identical domains, we require in each case that the domain name be qualified by a distinctive *role name*, which serves to identify the role played by that domain in the given relation. For example, in the relation *component* of Figure 2, the first domain *part* might be qualified by the role name *sub*, and the second by *super*, so that users could deal with the relationship *component* and its domains—*sub.part super.part, quantity*—without regard to any ordering between these domains.

To sum up, it is proposed that most users should interact with a relational model of the data consisting of a collection of time-varying relationships (rather than relations). Each user need not know more about any relationship than its name together with the names of its domains (role qualified whenever necessary).[3] Even this information might be offered in menu style by the system (subject to security and privacy constraints) upon request by the user.

There are usually many alternative ways in which a relational model may be established for a data bank. In order to discuss a preferred way (or normal form), we must first introduce a few additional concepts (active domain, primary key, foreign key, nonsimple domain) and establish some links with terminology currently in use in information systems programming. In the remainder of this paper, we shall not bother to distinguish between relations and relationships except where it appears advantageous to be explicit.

Consider an example of a data bank which includes relations concerning parts, projects, and suppliers. One relation called *part* is defined on the following domains:

(1) part number
(2) part name
(3) part color
(4) part weight
(5) quantity on hand
(6) quantity on order

and possibly other domains as well. Each of these domains is, in effect, a pool of values, some or all of which may be represented in the data bank at any instant. While it is conceivable that, at some instant, all part colors are present, it is unlikely that all possible part weights, part names, and part numbers are. We shall call the set of values represented at some instant the *active domain* at that instant.

Normally, one domain (or combination of domains) of a given relation has values which uniquely identify each element (n-tuple) of that relation. Such a domain (or combination) is called a *primary key*. In the example above, part number would be a primary key, while part color would not be. A primary key is *nonredundant* if it is either a simple domain (not a combination) or a combination such that none of the participating simple domains is superfluous in uniquely identifying each element. A relation may possess more than one nonredundant primary key. This would be the case in the example if different parts were always given distinct names. Whenever a relation has two or more nonredundant primary keys, one of them is arbitrarily selected and called *the* primary key of that relation.

A common requirement is for elements of a relation to cross-reference other elements of the same relation or elements of a different relation. Keys provide a user-oriented means (but not the only means) of expressing such cross-references. We shall call a domain (or domain combination) of relation R a *foreign key* if it is not the primary key of R but its elements are values of the primary key of some relation S (the possibility that S and R are identical is not excluded). In the relation *supply* of Figure 1, the combination of *supplier, part, project* is the primary key, while each of these three domains taken separately is a foreign key.

In previous work there has been a strong tendency to treat the data in a data bank as consisting of two parts, one part consisting of entity descriptions (for example, descriptions of suppliers) and the other part consisting of relations between the various entities or types of entities (for example, the *supply* relation). This distinction is difficult to maintain when one may have foreign keys in any relation whatsoever. In the user's relational model there appears to be no advantage to making such a distinction (there may be some advantage, however, when one applies relational concepts to machine representations of the user's set of relationships).

So far, we have discussed examples of relations which are defined on simple domains—domains whose elements are atomic (nondecomposable) values. Nonatomic values can be discussed within the relational framework. Thus, some domains may have relations as elements. These relations may, in turn, be defined on nonsimple domains, and so on. For example, one of the domains on which the relation *employee* is defined might be *salary history*. An element of the salary history domain is a binary relation defined on the domain *date* and the domain *salary*. The *salary history* domain is the set of all such binary relations. At any instant of time there are as many instances of the *salary history* relation in the data bank as there are employees. In contrast, there is only one instance of the *employee* relation.

The terms attribute and repeating group in present data base terminology are roughly analogous to simple domain

[2] In mathematical terms, a relationship is an equivalence class of those relations that are equivalent under permutation of domains (see Section 2.1.1).

[3] Naturally, as with any data put into and retrieved from a computer system, the user will normally make far more effective use of the data if he is aware of its meaning.

and nonsimple domain, respectively. Much of the confusion in present terminology is due to failure to distinguish between type and instance (as in "record") and between components of a user model of the data on the one hand and their machine representation counterparts on the other hand (again, we cite "record" as an example).

1.4. Normal Form

A relation whose domains are all simple can be represented in storage by a two-dimensional column-homogeneous array of the kind discussed above. Some more complicated data structure is necessary for a relation with one or more nonsimple domains. For this reason (and others to be cited below) the possibility of eliminating nonsimple domains appears worth investigating.[4] There is, in fact, a very simple elimination procedure, which we shall call normalization.

Consider, for example, the collection of relations exhibited in Figure 3(a). *Job history* and *children* are nonsimple domains of the relation *employee*. *Salary history* is a nonsimple domain of the relation *job history*. The tree in Figure 3(a) shows just these interrelationships of the nonsimple domains.

employee (*man#*, name, birthdate, jobhistory, children)
jobhistory (*jobdate*, title, salaryhistory)
salaryhistory (*salarydate*, salary)
children (*childname*, birthyear)

Fig. 3(a). Unnormalized set

employee' (*man#*, name, birthdate)
jobhistory' (*man#*, *jobdate*, title)
salaryhistory' (*man#*, *jobdate*, *salarydate*, salary)
children' (*man#*, *childname*, birthyear)

Fig. 3(b). Normalized set

Normalization proceeds as follows. Starting with the relation at the top of the tree, take its primary key and expand each of the immediately subordinate relations by inserting this primary key domain or domain combination. The primary key of each expanded relation consists of the primary key before expansion augmented by the primary key copied down from the parent relation. Now, strike out from the parent relation all nonsimple domains, remove the top node of the tree, and repeat the same sequence of operations on each remaining subtree.

The result of normalizing the collection of relations in Figure 3(a) is the collection in Figure 3(b). The primary key of each relation is italicized to show how such keys are expanded by the normalization.

[4] M. E. Senko of IBM, San Jose, independently recognized the desirability of eliminating nonsimple domains.

If normalization as described above is to be applicable, the unnormalized collection of relations must satisfy the following conditions:

(1) The graph of interrelationships of the nonsimple domains is a collection of trees.

(2) No primary key has a component domain which is nonsimple.

The writer knows of no application which would require any relaxation of these conditions. Further operations of a normalizing kind are possible. These are not discussed in this paper.

The simplicity of the array representation which becomes feasible when all relations are cast in normal form is not only an advantage for storage purposes but also for communication of bulk data between systems which use widely different representations of the data. The communication form would be a suitably compressed version of the array representation and would have the following advantages:

(1) It would be devoid of pointers (address-valued or displacement-valued).

(2) It would avoid all dependence on hash addressing schemes.

(3) It would contain no indices or ordering lists.

If the user's relational model is set up in normal form, names of items of data in the data bank can take a simpler form than would otherwise be the case. A general name would take a form such as

$$R(g).r.d$$

where R is a relational name; g is a generation identifier (optional); r is a role name (optional); d is a domain name. Since g is needed only when several generations of a given relation exist, or are anticipated to exist, and r is needed only when the relation R has two or more domains named d, the simple form $R.d$ will often be adequate.

1.5. Some Linguistic Aspects

The adoption of a relational model of data, as described above, permits the development of a universal data sublanguage based on an applied predicate calculus. A first-order predicate calculus suffices if the collection of relations is in normal form. Such a language would provide a yardstick of linguistic power for all other proposed data languages, and would itself be a strong candidate for embedding (with appropriate syntactic modification) in a variety of host languages (programming, command- or problem-oriented). While it is not the purpose of this paper to describe such a language in detail, its salient features would be as follows.

Let us denote the data sublanguage by R and the host language by H. R permits the declaration of relations and their domains. Each declaration of a relation identifies the primary key for that relation. Declared relations are added to the system catalog for use by any members of the user community who have appropriate authorization. H permits supporting declarations which indicate, perhaps less permanently, how these relations are represented in stor-

age. R permits the specification for retrieval of any subset of data from the data bank. Action on such a retrieval request is subject to security constraints.

The universality of the data sublanguage lies in its descriptive ability (not its computing ability). In a large data bank each subset of the data has a very large number of possible (and sensible) descriptions, even when we assume (as we do) that there is only a finite set of function subroutines to which the system has access for use in qualifying data for retrieval. Thus, the class of qualification expressions which can be used in a set specification must have the descriptive power of the class of well-formed formulas of an applied predicate calculus. It is well known that to preserve this descriptive power it is unnecessary to express (in whatever syntax is chosen) every formula of the selected predicate calculus. For example, just those in prenex normal form are adequate [9].

Arithmetic functions may be needed in the qualification or other parts of retrieval statements. Such functions can be defined in H and invoked in R.

A set so specified may be fetched for query purposes only, or it may be held for possible changes. Insertions take the form of adding new elements to declared relations without regard to any ordering that may be present in their machine representation. Deletions which are effective for the community (as opposed to the individual user or subcommunities) take the form of removing elements from declared relations. Some deletions and updates may be triggered by others, if deletion and update dependencies between specified relations are declared in R.

One important effect that the view adopted toward data has on the language used to retrieve it is in the naming of data elements and sets. Some aspects of this have been discussed in the previous section. With the usual network view, users will often be burdened with coining and using more relation names than are absolutely necessary, since names are associated with paths (or path types) rather than with relations.

Once a user is aware that a certain relation is stored, he will expect to be able to exploit[5] it using any combination of its arguments as "knowns" and the remaining arguments as "unknowns," because the information (like Everest) is there. This is a system feature (missing from many current information systems) which we shall call (logically) *symmetric exploitation* of relations. Naturally, symmetry in performance is not to be expected.

To support symmetric exploitation of a single binary relation, two directed paths are needed. For a relation of degree n, the number of paths to be named and controlled is n factorial.

Again, if a relational view is adopted in which every n-ary relation ($n > 2$) has to be expressed by the user as a nested expression involving only binary relations (see Feldman's LEAP System [10], for example) then $2n - 1$ names have to be coined instead of only $n + 1$ with direct n-ary notation as described in Section 1.2. For example, the 4-ary relation *supply* of Figure 1, which entails 5 names in n-ary notation, would be represented in the form

$$P\ (supplier,\ Q\ (part,\ R\ (project,\ quantity)))$$

in nested binary notation and, thus, employ 7 names.

A further disadvantage of this kind of expression is its asymmetry. Although this asymmetry does not prohibit symmetric exploitation, it certainly makes some bases of interrogation very awkward for the user to express (consider, for example, a query for those parts and quantities related to certain given projects via Q and R).

1.6. Expressible, Named, and Stored Relations

Associated with a data bank are two collections of relations: the *named set* and the *expressible set*. The named set is the collection of all those relations that the community of users can identify by means of a simple name (or identifier). A relation R acquires membership in the named set when a suitably authorized user declares R; it loses membership when a suitably authorized user cancels the declaration of R.

The expressible set is the total collection of relations that can be designated by expressions in the data language. Such expressions are constructed from simple names of relations in the named set; names of generations, roles and domains; logical connectives; the quantifiers of the predicate calculus;[6] and certain constant relation symbols such as $=$, $>$. The named set is a subset of the expressible set—usually a very small subset.

Since some relations in the named set may be time-independent combinations of others in that set, it is useful to consider associating with the named set a collection of statements that define these time-independent constraints. We shall postpone further discussion of this until we have introduced several operations on relations (see Section 2).

One of the major problems confronting the designer of a data system which is to support a relational model for its users is that of determining the class of stored representations to be supported. Ideally, the variety of permitted data representations should be just adequate to cover the spectrum of performance requirements of the total collection of installations. Too great a variety leads to unnecessary overhead in storage and continual reinterpretation of descriptions for the structures currently in effect.

For any selected class of stored representations the data system must provide a means of translating user requests expressed in the data language of the relational model into corresponding—and efficient—actions on the current stored representation. For a high level data language this presents a challenging design problem. Nevertheless, it is a problem which must be solved—as more users obtain concurrent access to a large data bank, responsibility for providing efficient response and throughput shifts from the individual user to the data system.

[5] Exploiting a relation includes query, update, and delete.

[6] Because each relation in a practical data bank is a finite set at every instant of time, the existential and universal quantifiers can be expressed in terms of a function that counts the number of elements in any finite set.

2. Redundancy and Consistency

2.1. Operations on Relations

Since relations are sets, all of the usual set operations are applicable to them. Nevertheless, the result may not be a relation, for example, the union of a binary relation and a ternary relation is not a relation.

The operations discussed below are specifically for relations. These operations are introduced because of their key role in deriving relations from other relations. Their principal application is in noninferential information systems—systems which do not provide logical inference services—although their applicability is not necessarily destroyed when such services are added.

Most users would not be directly concerned with these operations. Information systems designers and people concerned with data bank control should, however, be thoroughly familiar with them.

2.1.1. *Permutation.* A binary relation has an array representation with two columns. Interchanging these columns yields the converse relation. More generally, if a permutation is applied to the columns of an n-ary relation, the resulting relation is said to be a *permutation* of the given relation. There are, for example, $4! - 24$ permutations of the relation *supply* in Figure 1, if we include the identity permutation which leaves the ordering of columns unchanged.

Since the user's relational model consists of a collection of relationships (domain-unordered relations), permutation is not relevant to such a model considered in isolation. It is, however, relevant to the consideration of stored representations of the model. In a system which provides symmetric exploitation of relations, the set of queries answerable by a stored relation is identical to the set answerable by any permutation of that relation. Although it is logically unnecessary to store both a relation and some permutation of it, performance considerations could make it advisable.

2.1.2. *Projection.* Suppose now we select certain columns of a relation (striking out the others) and then remove from the resulting array any duplication in the rows. The final array represents a relation which is said to be a *projection* of the given relation.

A selection operator π is used to obtain any desired permutation, projection, or combination of the two operations. Thus, if L is a list of k indices[7] $L = i_1, i_2, \cdots, i_k$ and R is an n-ary relation ($n \geq k$), then $\pi_L(R)$ is the k-ary relation whose jth column is column i_j of R ($j = 1, 2, \cdots, k$) except that duplication in resulting rows is removed. Consider the relation *supply* of Figure 1. A permuted projection of this relation is exhibited in Figure 4. Note that, in this particular case, the projection has fewer n-tuples than the relation from which it is derived.

2.1.3. *Join.* Suppose we are given two binary relations, which have some domain in common. Under what circumstances can we combine these relations to form a ternary relation which preserves all of the information in the given relations?

The example in Figure 5 shows two relations R, S, which are joinable without loss of information, while Figure 6 shows a join of R with S. A binary relation R is *joinable* with a binary relation S if there exists a ternary relation U such that $\pi_{12}(U) = R$ and $\pi_{23}(U) = S$. Any such ternary relation is called a *join* of R with S. If R, S are binary relations such that $\pi_2(R) = \pi_1(S)$, then R is joinable with S. One join that always exists in such a case is the *natural join* of R with S defined by

$$R*S = \{(a, b, c) : R(a, b) \wedge S(b, c)\}$$

where $R(a, b)$ has the value *true* if (a, b) is a member of R and similarly for $S(b, c)$. It is immediate that

$$\pi_{12}(R*S) = R$$

and

$$\pi_{23}(R*S) = S.$$

Note that the join shown in Figure 6 is the natural join of R with S from Figure 5. Another join is shown in Figure 7.

Π_{31}(*supply*)	(*project*	*supplier*)
	5	1
	5	2
	1	4
	7	2

Fig. 4. A permuted projection of the relation in Figure 1

R	(*supplier*	*part*)	S	(*part*	*project*)
	1	1		1	1
	2	1		1	2
	2	2		2	1

Fig. 5. Two joinable relations

R*S	(*supplier*	*part*	*project*)
	1	1	1
	1	1	2
	2	1	1
	2	1	2
	2	2	1

Fig. 6. The natural join of R with S (from Figure 5)

U	(*supplier*	*part*	*project*)
	1	1	2
	2	1	1
	2	2	1

Fig. 7. Another join of R with S (from Figure 5)

Inspection of these relations reveals an element (element 1) of the domain *part* (the domain on which the join is to be made) with the property that it possesses more than one relative under R and also under S. It is this ele-

[7] When dealing with relationships, we use domain names (role-qualified whenever necessary) instead of domain positions.

ment which gives rise to the plurality of joins. Such an element in the joining domain is called a *point of ambiguity* with respect to the joining of R with S.

If either $\pi_{21}(R)$ or S is a function,[5] no point of ambiguity can occur in joining R with S. In such a case, the natural join of R with S is the only join of R with S. Note that the reiterated qualification "of R with S" is necessary, because S might be joinable with R (as well as R with S), and this join would be an entirely separate consideration. In Figure 5, none of the relations R, $\pi_{21}(R)$, S, $\pi_{21}(S)$ is a function.

Ambiguity in the joining of R with S can sometimes be resolved by means of other relations. Suppose we are given, or can derive from sources independent of R and S, a relation T on the domains *project* and *supplier* with the following properties:

$$(1)\ \pi_1(T) = \pi_2(S),$$

$$(2)\ \pi_2(T) = \pi_1(R),$$

$$(3)\ T(j, s) \rightarrow \exists p\,(R(S, p) \wedge S(p, j)),$$

$$(4)\ R(s, p) \rightarrow \exists j\,(S(p, j) \wedge T(j, s)),$$

$$(5)\ S(p, j) \rightarrow \exists s\,(T(j, s) \wedge R(s, p)),$$

then we may form a three-way join of R, S, T; that is, a ternary relation such that

$$\pi_{12}(U) = R, \qquad \pi_{23}(U) = S, \qquad \pi_{31}(U) = T.$$

Such a join will be called a *cyclic 3-join* to distinguish it from a *linear 3-join* which would be a quaternary relation V such that

$$\pi_{12}(V) = R, \qquad \pi_{23}(V) = S, \qquad \pi_{34}(V) = T.$$

While it is possible for more than one cyclic 3-join to exist (see Figures 8, 9, for an example), the circumstances under which this can occur entail much more severe constraints

R	(s	p)
	1	a
	2	a
	2	b

S	(p	j)
	a	d
	a	e
	b	d
	b	e

T	(j	s)
	d	1
	d	2
	e	2
	e	2

FIG. 8. Binary relations with a plurality of cyclic 3-joins

U	(s	p	j)
	1	a	d
	2	a	e
	2	b	d
	2	b	e

U′	(s	p	j)
	1	a	d
	2	a	d
	2	a	e
	2	b	d
	2	b	e

FIG. 9. Two cyclic 3-joins of the relations in Figure 8

than those for a plurality of 2-joins. To be specific, the relations R, S, T must possess points of ambiguity with respect to joining R with S (say point x), S with T (say y), and T with R (say z), and, furthermore, y must be a relative of x under S, z a relative of y under T, and x a relative of z under R. Note that in Figure 8 the points $x = a$; $y = d$; $z = 2$ have this property.

The natural linear 3-join of three binary relations R, S, T is given by

$$R*S*T = \{(a, b, c, d): R(a, b) \wedge S(b, c) \wedge T(c, d)\}$$

where parentheses are not needed on the left-hand side because the natural 2-join $(*)$ is associative. To obtain the cyclic counterpart, we introduce the operator γ which produces a relation of degree $n - 1$ from a relation of degree n by tying its ends together. Thus, if R is an n-ary relation $(n \geq 2)$, the *tie* of R is defined by the equation

$$\gamma(R) = \{(a_1, a_2, \cdots, a_{n-1}): R(a_1, a_2, \cdots, a_{n-1}, a_n) \wedge a_1 = a_n\}.$$

We may now represent the natural cyclic 3-join of R, S, T by the expression

$$\gamma(R*S*T).$$

Extension of the notions of linear and cyclic 3-join and their natural counterparts to the joining of n binary relations (where $n \geq 3$) is obvious. A few words may be appropriate, however, regarding the joining of relations which are not necessarily binary. Consider the case of two relations R (degree r), S (degree s) which are to be joined on p of their domains ($p < r$, $p < s$). For simplicity, suppose these p domains are the last p of the r domains of R, and the first p of the s domains of S. If this were not so, we could always apply appropriate permutations to make it so. Now, take the Cartesian product of the first r-p domains of R, and call this new domain A. Take the Cartesian product of the last p domains of R, and call this B. Take the Cartesian product of the last s-p domains of S and call this C.

We can treat R as if it were a binary relation on the domains A, B. Similarly, we can treat S as if it were a binary relation on the domains B, C. The notions of linear and cyclic 3-join are now directly applicable. A similar approach can be taken with the linear and cyclic n-joins of n relations of assorted degrees.

2.1.4. *Composition.* The reader is probably familiar with the notion of composition applied to functions. We shall discuss a generalization of that concept and apply it first to binary relations. Our definitions of composition and composability are based very directly on the definitions of join and joinability given above.

Suppose we are given two relations R, S. T is a *composition* of R with S if there exists a join U of R with S such that $T = \pi_{13}(U)$. Thus, two relations are composable if and only if they are joinable. However, the existence of more than one join of R with S does not imply the existence of more than one composition of R with S.

Corresponding to the natural join of R with S is the

[5] A function is a binary relation, which is one-one or many-one, but not one-many.

natural composition[5] of R with S defined by

$$R \cdot S = \pi_{13}(R*S).$$

Taking the relations R, S from Figure 5, their natural composition is exhibited in Figure 10 and another composition is exhibited in Figure 11 (derived from the join exhibited in Figure 7).

R·S	(*project*	*supplier*)
	1	1
	1	2
	2	1
	2	2

FIG. 10. The natural composition of R with S (from Figure 5)

T	(*project*	*supplier*)
	1	2
	2	1

FIG. 11. Another composition of R with S (from Figure 5)

When two or more joins exist, the number of distinct compositions may be as few as one or as many as the number of distinct joins. Figure 12 shows an example of two relations which have several joins but only one composition. Note that the ambiguity of point c is lost in composing R with S, because of unambiguous associations made via the points a, b, d, e.

R	(*supplier*	*part*)	S	(*part*	*project*)
	1	a		a	g
	1	b		b	f
	1	c		c	f
	2	c		c	g
	2	d		d	g
	2	e		e	f

FIG. 12. Many joins, only one composition

Extension of composition to pairs of relations which are not necessarily binary (and which may be of different degree) follows the same pattern as extension of pairwise joining to such relations.

A lack of understanding of relational composition has led several systems designers into what may be called the *connection trap*. This trap may be described in terms of the following example. Suppose each supplier description is linked by pointers to the descriptions of each part supplied by that supplier, and each part description is similarly linked to the descriptions of each project which uses that part. A conclusion is now drawn which is, in general, erroneous: namely that, if all possible paths are followed from a given supplier via the parts he supplies to the projects using those parts, one will obtain a valid set of all projects supplied by that supplier. Such a conclusion is correct only in the very special case that the target relation between projects and suppliers is, in fact, the natural composition of the other two relations—and we must normally add the phrase "for all time," because this is usually implied in claims concerning path-following techniques.

[5] Other writers tend to ignore compositions other than the natural one, and accordingly refer to this particular composition as *the* composition—see, for example, Kelley's "General Topology."

2.1.5. *Restriction.* A subset of a relation is a relation. One way in which a relation S may act on a relation R to generate a subset of R is through the operation *restriction* of R by S. This operation is a generalization of the restriction of a function to a subset of its domain, and is defined as follows.

Let L, M be equal-length lists of indices such that $L = i_1, i_2, \cdots, i_k$, $M = j_1, j_2, \cdots, j_k$ where $k \leq$ degree of R and $k \leq$ degree of S. Then the L, M restriction of R by S denoted $R_{L|M}S$ is the maximal subset R' of R such that

$$\pi_L(R') = \pi_M(S).$$

The operation is defined only if equality is applicable between elements of $\pi_{i_h}(R)$ on the one hand and $\pi_{j_h}(S)$ on the other for all $h = 1, 2, \cdots, k$.

The three relations R, S, R' of Figure 13 satisfy the equation $R' = R_{(2,3)|(1,2)}S$.

R	(*s*	*p*	*j*)	S	(*p*	*j*)	R′	(*s*	*p*	*j*)
	1	a	A		a	A		1	a	A
	2	a	A		c	B		2	a	A
	2	a	B		b	B		2	b	B
	2	b	A							
	2	b	B							

FIG. 13. Example of restriction

We are now in a position to consider various applications of these operations on relations.

2.2. Redundancy

Redundancy in the named set of relations must be distinguished from redundancy in the stored set of representations. We are primarily concerned here with the former. To begin with, we need a precise notion of derivability for relations.

Suppose θ is a collection of operations on relations and each operation has the property that from its operands it yields a unique relation (thus natural join is eligible, but join is not). A relation R is *θ-derivable* from a set S of relations if there exists a sequence of operations from the collection θ which, for all time, yields R from members of S. The phrase "for all time" is present, because we are dealing with time-varying relations, and our interest is in derivability which holds over a significant period of time. For the named set of relationships in noninferential systems, it appears that an adequate collection θ_1 contains the following operations: projection, natural join, tie, and restriction. Permutation is irrelevant and natural composition need not be included, because it is obtainable by taking a natural join and then a projection. For the stored set of representations, an adequate collection θ_2 of operations would include permutation and additional operations concerned with subsetting and merging relations, and ordering and connecting their elements.

2.2.1. *Strong Redundancy.* A set of relations is *strongly redundant* if it contains at least one relation that possesses a projection which is derivable from other projections of relations in the set. The following two examples are intended to explain why strong redundancy is defined this way, and to demonstrate its practical use. In the first ex-

ample the collection of relations consists of just the following relation:

employee (*serial#*, *name*, *manager#*, *managername*)

with *serial#* as the primary key and *manager#* as a foreign key. Let us denote the active domain by Δ_t, and suppose that

$$\Delta_1(manager\#) \subset \Delta_2(serial\#)$$

and

$$\Delta_t(managername) \subset \Delta_t(name)$$

for all time t. In this case the redundancy is obvious: the domain *managername* is unnecessary. To see that it is a strong redundancy as defined above, we observe that

$$\pi_{24}(employee) = \pi_{12}(employee)_1|_2\pi_3(employee).$$

In the second example the collection of relations includes a relation S describing suppliers with primary key $s\#$, a relation D describing departments with primary key $d\#$, a relation J describing projects with primary key $j\#$, and the following relations:

$$P(s\#, d\#, \cdots), \qquad Q(s\#, j\#, \cdots), \qquad R(d\#, j\#, \cdots),$$

where in each case $\cdots$ denotes domains other than $s\#$, $d\#$, $j\#$. Let us suppose the following condition C is known to hold independent of time: supplier s supplies department d (relation P) if and only if supplier s supplies some project j (relation Q) to which d is assigned (relation R). Then, we can write the equation

$$\pi_{12}(P) = \pi_{12}(Q) \cdot \pi_{21}(R)$$

and thereby exhibit a strong redundancy.

An important reason for the existence of strong redundancies in the named set of relationships is user convenience. A particular case of this is the retention of semi-obsolete relationships in the named set so that old programs that refer to them by name can continue to run correctly. Knowledge of the existence of strong redundancies in the named set enables a system or data base administrator greater freedom in the selection of stored representations to cope more efficiently with current traffic. If the strong redundancies in the named set are directly reflected in strong redundancies in the stored set (or if other strong redundancies are introduced into the stored set), then, generally speaking, extra storage space and update time are consumed with a potential drop in query time for some queries and in load on the central processing units.

2.2.2. *Weak Redundancy.* A second type of redundancy may exist. In contrast to strong redundancy it is not characterized by an equation. A collection of relations is *weakly redundant* if it contains a relation that has a projection which is not derivable from other members but is at all times a projection of *some* join of other projections of relations in the collection.

We can exhibit a weak redundancy by taking the second example (cited above) for a strong redundancy, and assuming now that condition C does not hold at all times. The relations $\pi_{12}(P)$, $\pi_{12}(Q)$, $\pi_{12}(R)$ are complex[12] relations with the possibility of points of ambiguity occurring from time to time in the potential joining of any two. Under these circumstances, none of them is derivable from the other two. However, constraints do exist between them, since each is a projection of some cyclic join of the three of them. One of the weak redundancies can be characterized by the statement: for all time, $\pi_{12}(P)$ is *some* composition of $\pi_{12}(Q)$ with $\pi_{21}(R)$. The composition in question might be the natural one at some instant and a nonnatural one at another instant.

Generally speaking, weak redundancies are inherent in the logical needs of the community of users. They are not removable by the system or data base administrator. If they appear at all, they appear in both the named set and the stored set of representations.

2.3. Consistency

Whenever the named set of relations is redundant in either sense, we shall associate with that set a collection of statements which define all of the redundancies which hold independent of time between the member relations. If the information system lacks—and it most probably will—detailed semantic information about each named relation, it cannot deduce the redundancies applicable to the named set. It might, over a period of time, make attempts to induce the redundancies, but such attempts would be fallible.

Given a collection C of time-varying relations, an associated set Z of constraint statements and an instantaneous value V for C, we shall call the state (C, Z, V) *consistent* or *inconsistent* according as V does or does not satisfy Z. For example, given stored relations R, S, T together with the constraint statement "$\pi_{12}(T)$ is a composition of $\pi_{12}(R)$ with $\pi_{12}(S)$", we may check from time to time that the values stored for R, S, T satisfy this constraint. An algorithm for making this check would examine the first two columns of each of R, S, T (in whatever way they are represented in the system) and determine whether

(1) $\pi_1(T) = \pi_1(R)$,

(2) $\pi_2(T) = \pi_2(S)$,

(3) for every element pair (a, c) in the relation $\pi_{12}(T)$ there is an element b such that (a, b) is in $\pi_{12}(R)$ and (b, c) is in $\pi_{12}(S)$.

There are practical problems (which we shall not discuss here) in taking an instantaneous snapshot of a collection of relations, some of which may be very large and highly variable.

It is important to note that consistency as defined above is a property of the instantaneous state of a data bank, and is independent of how that state came about. Thus, in particular, there is no distinction made on the basis of whether a user generated an inconsistency due to an act of omission or an act of commission. Examination of a simple

[12] A binary relation is complex if neither it nor its converse is a function.

example will show the reasonableness of this (possibly unconventional) approach to consistency.

Suppose the named set C includes the relations S, J, D, P, Q, R of the example in Section 2.2 and that P, Q, R possess either the strong or weak redundancies described therein (in the particular case now under consideration, it does not matter which kind of redundancy occurs). Further, suppose that at some time t the data bank state is consistent and contains no project j such that supplier 2 supplies project j and j is assigned to department 5. Accordingly, there is no element $(2, 5)$ in $\pi_{12}(P)$. Now, a user introduces the element $(2, 5)$ into $\pi_{12}(P)$ by inserting some appropriate element into P. The data bank state is now inconsistent. The inconsistency could have arisen from an act of omission, if the input $(2, 5)$ is correct, and there does exist a project j such that supplier 2 supplies j and j is assigned to department 5. In this case, it is very likely that the user intends in the near future to insert elements into Q and R which will have the effect of introducing $(2, j)$ into $\pi_{12}(Q)$ and $(5, j)$ in $\pi_{12}(R)$. On the other hand, the input $(2, 5)$ might have been faulty. It could be the case that the user intended to insert some other element into P—an element whose insertion would transform a consistent state into a consistent state. The point is that the system will normally have no way of resolving this question without interrogating its environment (perhaps the user who created the inconsistency).

There are, of course, several possible ways in which a system can detect inconsistencies and respond to them. In one approach the system checks for possible inconsistency whenever an insertion, deletion, or key update occurs. Naturally, such checking will slow these operations down. If an inconsistency has been generated, details are logged internally, and if it is not remedied within some reasonable time interval, either the user or someone responsible for the security and integrity of the data is notified. Another approach is to conduct consistency checking as a batch operation once a day or less frequently. Inputs causing the inconsistencies which remain in the data bank state at checking time can be tracked down if the system maintains a journal of all state-changing transactions. This latter approach would certainly be superior if few nontransitory inconsistencies occurred.

2.4. Summary

In Section 1 a relational model of data is proposed as a basis for protecting users of formatted data systems from the potentially disruptive changes in data representation caused by growth in the data bank and changes in traffic. A normal form for the time-varying collection of relationships is introduced.

In Section 2 operations on relations and two types of redundancy are defined and applied to the problem of maintaining the data in a consistent state. This is bound to become a serious practical problem as more and more different types of data are integrated together into common data banks.

Many questions are raised and left unanswered. For example, only a few of the more important properties of the data sublanguage in Section 1.4 are mentioned. Neither the purely linguistic details of such a language nor the implementation problems are discussed. Nevertheless, the material presented should be adequate for experienced systems programmers to visualize several approaches. It is also hoped that this paper can contribute to greater precision in work on formatted data systems.

Acknowledgment. It was C. T. Davies of IBM Poughkeepsie who convinced the author of the need for data independence in future information systems. The author wishes to thank him and also F. P. Palermo, C. P. Wang, E. B. Altman, and M. E. Senko of the IBM San Jose Research Laboratory for helpful discussions.

Received September, 1969; revised February, 1970

References

1. Childs, D. L. Feasibility of a set-theoretical data structure —a general structure based on a reconstituted definition of relation. Proc. IFIP Cong., 1968, North Holland Pub. Co., Amsterdam, p. 162 179.
2. Levein, R. E., and Maron, M. E. A computer system for inference execution and data retrieval. *Comm. ACM 10*, 11 (Nov. 1967), 715–721.
3. Bachman, C. W. Software for random access processing. *Datamation* (Apr. 1965), 36–41.
4. McGee, W. C. Generalized file processing. In *Annual Review in Automatic Programming 5*, 13, Pergamon Press, New York, 1969, pp. 77–149.
5. Information Management System/360, Application Description Manual H20-0524-1. IBM Corp., White Plains, N. Y., July 1968.
6. GIS (Generalized Information System), Application Description Manual H20-0574. IBM Corp., White Plains, N. Y., 1965.
7. Bleier, R. E. Treating hierarchical data structures in the SDC time-shared data management system (TDMS). Proc. ACM 22nd Nat. Conf., 1967, MDI Publications, Wayne, Pa., pp. 41–49.
8. IDS Reference Manual GE 625/635, GE Inform. Sys. Div., Pheonix, Ariz., CPB 1093B, Feb. 1968.
9. Church, A. *An Introduction to Mathematical Logic I*. Princeton U. Press, Princeton, N.J., 1956.
10. Feldman, J. A., and Rovner, P. D. An Algol-based associative language. Stanford Artificial Intelligence Rep. AI-66, Aug. 1, 1968.

ANEXO TÉCNICO 2

Cramming more components onto integrated circuits

by Gordon E. Moore

Cramming More Components onto Integrated Circuits

GORDON E. MOORE, LIFE FELLOW, IEEE

With unit cost falling as the number of components per circuit rises, by 1975 economics may dictate squeezing as many as 65 000 components on a single silicon chip.

The future of integrated electronics is the future of electronics itself. The advantages of integration will bring about a proliferation of electronics, pushing this science into many new areas.

Integrated circuits will lead to such wonders as home computers—or at least terminals connected to a central computer—automatic controls for automobiles, and personal portable communications equipment. The electronic wristwatch needs only a display to be feasible today.

But the biggest potential lies in the production of large systems. In telephone communications, integrated circuits in digital filters will separate channels on multiplex equipment. Integrated circuits will also switch telephone circuits and perform data processing.

Computers will be more powerful, and will be organized in completely different ways. For example, memories built of integrated electronics may be distributed throughout the machine instead of being concentrated in a central unit. In addition, the improved reliability made possible by integrated circuits will allow the construction of larger processing units. Machines similar to those in existence today will be built at lower costs and with faster turnaround.

I. Present and Future

By integrated electronics, I mean all the various technologies which are referred to as microelectronics today as well as any additional ones that result in electronics functions supplied to the user as irreducible units. These technologies were first investigated in the late 1950's. The object was to miniaturize electronics equipment to include increasingly complex electronic functions in limited space with minimum weight. Several approaches evolved, including microassembly techniques for individual components, thin-film structures, and semiconductor integrated circuits.

Reprinted from Gordon E. Moore, "Cramming More Components onto Integrated Circuits," *Electronics*, pp. 114–117, April 19, 1965.
Publisher Item Identifier S 0018-9219(98)00753-1.

Each approach evolved rapidly and converged so that each borrowed techniques from another. Many researchers believe the way of the future to be a combination of the various approaches.

The advocates of semiconductor integrated circuitry are already using the improved characteristics of thin-film resistors by applying such films directly to an active semiconductor substrate. Those advocating a technology based upon films are developing sophisticated techniques for the attachment of active semiconductor devices to the passive film arrays.

Both approaches have worked well and are being used in equipment today.

II. The Establishment

Integrated electronics is established today. Its techniques are almost mandatory for new military systems, since the reliability, size, and weight required by some of them is achievable only with integration. Such programs as Apollo, for manned moon flight, have demonstrated the reliability of integrated electronics by showing that complete circuit functions are as free from failure as the best individual transistors.

Most companies in the commercial computer field have machines in design or in early production employing integrated electronics. These machines cost less and perform better than those which use "conventional" electronics.

Instruments of various sorts, especially the rapidly increasing numbers employing digital techniques, are starting to use integration because it cuts costs of both manufacture and design.

The use of linear integrated circuitry is still restricted primarily to the military. Such integrated functions are expensive and not available in the variety required to satisfy a major fraction of linear electronics. But the first applications are beginning to appear in commercial electronics, particularly in equipment which needs low-frequency amplifiers of small size.

III. Reliability Counts

In almost every case, integrated electronics has demonstrated high reliability. Even at the present level of pro-

duction—low compared to that of discrete components—it offers reduced systems cost, and in many systems improved performance has been realized.

Integrated electronics will make electronic techniques more generally available throughout all of society, performing many functions that presently are done inadequately by other techniques or not done at all. The principal advantages will be lower costs and greatly simplified design—payoffs from a ready supply of low-cost functional packages.

For most applications, semiconductor integrated circuits will predominate. Semiconductor devices are the only reasonable candidates presently in existence for the active elements of integrated circuits. Passive semiconductor elements look attractive too, because of their potential for low cost and high reliability, but they can be used only if precision is not a prime requisite.

Silicon is likely to remain the basic material, although others will be of use in specific applications. For example, gallium arsenide will be important in integrated microwave functions. But silicon will predominate at lower frequencies because of the technology which has already evolved around it and its oxide, and because it is an abundant and relatively inexpensive starting material.

IV. Costs and Curves

Reduced cost is one of the big attractions of integrated electronics, and the cost advantage continues to increase as the technology evolves toward the production of larger and larger circuit functions on a single semiconductor substrate. For simple circuits, the cost per component is nearly inversely proportional to the number of components, the result of the equivalent piece of semiconductor in the equivalent package containing more components. But as components are added, decreased yields more than compensate for the increased complexity, tending to raise the cost per component. Thus there is a minimum cost at any given time in the evolution of the technology. At present, it is reached when 50 components are used per circuit. But the minimum is rising rapidly while the entire cost curve is falling (see graph). If we look ahead five years, a plot of costs suggests that the minimum cost per component might be expected in circuits with about 1000 components per circuit (providing such circuit functions can be produced in moderate quantities). In 1970, the manufacturing cost per component can be expected to be only a tenth of the present cost.

The complexity for minimum component costs has increased at a rate of roughly a factor of two per year (see graph). Certainly over the short term this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will not remain nearly constant for at least ten years. That means by 1975, the number of components per integrated circuit for minimum cost will be 65 000.

I believe that such a large circuit can be built on a single wafer.

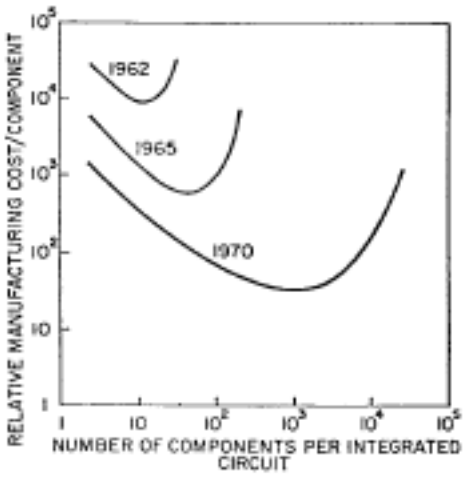


Fig. 1.

V. Two-mil Squares

With the dimensional tolerances already being employed in integrated circuits, isolated high-performance transistors can be built on centers two-thousandths of an inch apart. Such a two-mil square can also contain several kilohms of resistance or a few diodes. This allows at least 500 components per linear inch or a quarter million per square inch. Thus, 65 000 components need occupy only about one-fourth a square inch.

On the silicon wafer currently used, usually an inch or more in diameter, there is ample room for such a structure if the components can be closely packed with no space wasted for interconnection patterns. This is realistic, since efforts to achieve a level of complexity above the presently available integrated circuits are already under way using multilayer metallization patterns separated by dielectric films. Such a density of components can be achieved by present optical techniques and does not require the more exotic techniques, such as electron beam operations, which are being studied to make even smaller structures.

VI. Increasing the Yield

There is no fundamental obstacle to achieving device yields of 100%. At present, packaging costs so far exceed the cost of the semiconductor structure itself that there is no incentive to improve yields, but they can be raised as high as is economically justified. No barrier exists comparable to the thermodynamic equilibrium considerations that often limit yields in chemical reactions; it is not even necessary to do any fundamental research or to replace present processes. Only the engineering effort is needed.

In the early days of integrated circuitry, when yields were extremely low, there was such incentive. Today ordinary integrated circuits are made with yields comparable with those obtained for individual semiconductor devices. The same pattern will make larger arrays economical, if other considerations make such arrays desirable.

Fig. 2.

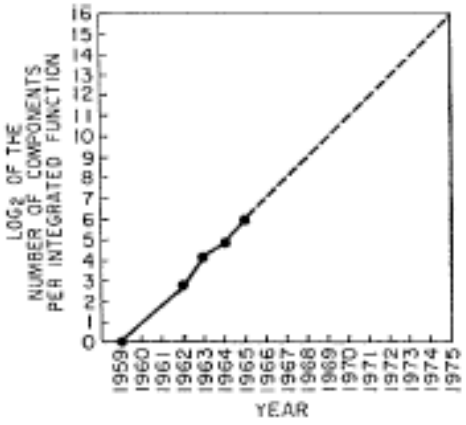


Fig. 3.

VII. Heat Problem

Will it be possible to remove the heat generated by tens of thousands of components in a single silicon chip?

If we could shrink the volume of a standard high-speed digital computer to that required for the components themselves, we would expect it to glow brightly with present power dissipation. But it won't happen with integrated circuits. Since integrated electronic structures are two dimensional, they have a surface available for cooling close to each center of heat generation. In addition, power is needed primarily to drive the various lines and capacitances associated with the system. As long as a function is confined to a small area on a wafer, the amount of capacitance which must be driven is distinctly limited. In fact, shrinking dimensions on an integrated structure makes it possible to operate the structure at higher speed for the same power per unit area.

VIII. Day of Reckoning

Clearly, we will be able to build such component-crammed equipment. Next, we ask under what circumstances we should do it. The total cost of making a particular system function must be minimized. To do so, we could amortize the engineering over several identical items, or evolve flexible techniques for the engineering of large functions so that no disproportionate expense need be borne by a particular array. Perhaps newly devised design automation procedures could translate from logic diagram to technological realization without any special engineering.

It may prove to be more economical to build large systems out of smaller functions, which are separately packaged and interconnected. The availability of large functions, combined with functional design and construction, should allow the manufacturer of large systems to design and construct a considerable variety of equipment both rapidly and economically.

IX. Linear Circuitry

Integration will not change linear systems as radically as digital systems. Still, a considerable degree of integration will be achieved with linear circuits. The lack of large-value capacitors and inductors is the greatest fundamental limitation to integrated electronics in the linear area.

By their very nature, such elements require the storage of energy in a volume. For high Q it is necessary that the volume be large. The incompatibility of large volume and integrated electronics is obvious from the terms themselves. Certain resonance phenomena, such as those in piezoelectric crystals, can be expected to have some applications for tuning functions, but inductors and capacitors will be with us for some time.

The integrated RF amplifier of the future might well consist of integrated stages of gain, giving high performance at minimum cost, interspersed with relatively large tuning elements.

Other linear functions will be changed considerably. The matching and tracking of similar components in integrated structures will allow the design of differential amplifiers of greatly improved performance. The use of thermal feedback effects to stabilize integrated structures to a small fraction of a degree will allow the construction of oscillators with crystal stability.

Even in the microwave area, structures included in the definition of integrated electronics will become increasingly important. The ability to make and assemble components small compared with the wavelengths involved will allow the use of lumped parameter design, at least at the lower frequencies. It is difficult to predict at the present time just how extensive the invasion of the microwave area by integrated electronics will be. The successful realization of such items as phased-array antennas, for example, using a multiplicity of integrated microwave power sources, could completely revolutionize radar.

ANEXO TÉCNICO 3

Índices propuestos

```
CREATE INDEX idx_ap_paterno_alumno ON ALUMNO(ap_paterno);

CREATE INDEX idx_carrera_alumno_carr ON ALUMNO_CARRERA(clave_carrera);

CREATE INDEX idx_clave_carrera_plan ON PLAN_ESTUDIOS(clave_carrera);

CREATE INDEX idx_nombre_modulo ON MODULO_SALIDA(nombre);

CREATE INDEX idx_clave_asignatura_modulo ON
MODULO_SALIDA(clave_plan);

CREATE INDEX idx_nombre_asignatura ON ASIGNATURA(nombre);

CREATE INDEX idx_clave_teoria_seriacion ON seriacion(clave_asignatura);

CREATE INDEX idx_periodo_asignacion ON ASIGNACION(periodo);

CREATE INDEX idx_ap_paterno_academico ON ACADEMICO(ap_paterno);
```

GLOSARIO

SIIFI. Sistema Integral de Información de la Facultad de Ingeniería

BD. Base de datos

SGBD. Sistema Gestor de Base de Datos

FI. Facultad de Ingeniería

USECAD. Unidad de Servicios de Cómputo Administrativos

SSA. Secretaría de Servicios Administrativos

OLAP. On-Line Analytical Processing

OLTP. On-Line Transactional Processing

SABRE. Semi-automated Business Research Environment

IDS. Integrated Data Store

CODASYL. Conference on Data Systems Languages

COBOL. Common Business-Oriented Language

ANSI. American National Standards Institute

SQL. Structured Query Language

XML. eXtensible Markup Language

XQuery. Lenguaje de consulta XML)

WWW. World Wide Web

CD. Compact Disc

SSD. Solid-State Drive

CAD. Computer-Aided Design

1FN. Primera Forma Normal

2FN. Segunda Forma Normal

3FN. Tercer Forma Normal

FNBC. Forma Normal de Boyce-Codd

ACID. Atomicity – Consistency – Isolation – Durability

DDL. Data Definition Language

DML. Data Manipulation Language

DAL. Data Access Language

DBA. Data Base Administrator

RFC. Registro Federal de Contribuyente

RBAC. Roled-Based Access Control

DER. Diagrama Entidad Relación

CP. Código Postal

PK. Primary Key

FK. Foreign Key

NN. Not Null