UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

"DISEÑO DE SISTEMAS DE INFORMACIÓN ORIENTADO A NEGOCIOS CON SQL SERVER Y SQL ORACLE".

I N F O R M E

QUE PARA OBTENER EL TÍTULO DE :

INGENIERO EN COMPUTACIÓN

P R E S E N T A :

CARLOS ALBERTO PARRALES CASTAÑEDA

ASESOR: ING. ANTONIA NAVARRO GONZÁLEZ.

MÉXICO 2010

DEDICATORIA.

Agradezco a mi familia, el haberme apoyado, brindarme confianza y su amor incondicional; especialmente a una extraordinaria, gentil y amorosa mujer que en cada momento de mi vida ha estado conmigo y de quien he recibido sabios consejos, Gracias Madre.

También le doy las gracias a mi hermosa y talentosa hermana, quién me ha apoyado siempre.

Mi agradecimiento a la Universidad Nacional Autónoma de México, por haberme recibido en sus aulas y prepararme, para convertirme en un profesionista y lograr mis metas.

Doy las gracias a cada uno de mis profesores por los conocimientos que me transmitieron en el transcurso de la carrera.

ÍNDICE.

INTRODUCCIÓN.

El presente trabajo de titulación consiste en un informe detallado de los temas, ejemplos y proyectos desarrollados a lo largo de los cinco módulos que conforman el Diplomado: "DISEÑO DE SISTEMAS DE INFORMACIÓN ORIENTADO A NEGOCIOS CON SQL SERVER Y SQL ORACLE", mismos que corresponden a los cinco capítulos de dicho escrito, y un apartado denominado Anexos, donde se describen en forma general los proyectos realizados de cada módulo.

El objetivo primordial de este informe, es hacer énfasis en la importancia de la integración de los datos, siendo los elementos determinantes de la confiabilidad y la veracidad de la información, proporcionando los conocimientos para el diseño y desarrollo de Bases de Datos, como una Solución Integral de los Sistemas de Información.

Estudia el Servidor de Bases de Datos: Microsoft SQL Server 2005, Oracle 9i y Oracle 10g, desde el punto de vista de un Diseñador y Programador de Bases de Datos, prestando atención en la creación, modificación y eliminación de objetos como: tablas, vistas, procedimientos almacenados, etcétera.

A continuación se describen brevemente los cinco capítulos y la parte de anexos, pertenecientes a este Informe de Actividades Desarrolladas durante el Diplomado.

- **Capítulo I. Análisis y Diseño de Sistemas de Información según las Reglas de Negocios.**

  El capítulo inicia con un breve bosquejo de los fundamentos y modelos de Bases de Datos, utiliza los conocimientos teóricos y prácticos en el diseño, ejecución y administración de Sistemas de Bases de Datos Relacionales, continuando con la descripción de los objetivos, funciones, elementos y tipos de Sistemas de Información.

  Se describen las herramientas y estrategias para el Desarrollo de Sistemas de Información según las Reglas de Negocios, establecidas en cada organización, enfatizando en el uso y manejo de la información como uno de los principales recursos que poseen las empresas; razón por la cual es importante destacar que los procesos de producción, distribución, seguridad, almacenamiento y recuperación de dicha información, sean empleados de forma correcta y eficiente.

  Se menciona el impacto estratégico y competitivo del uso de Tecnologías de Información (TI), dentro de la organización al automatizar sus procesos y actividades operativas, dando como resultado información adecuada, pertinente y oportuna, siendo la clave esencial del éxito o fracaso de ésta.

- **Capítulo II. Arquitectura y Lenguaje de Programación en SQL Server.**

  En este capítulo se conocerán las principales características y componentes del Servidor SQL Server 2005. Se identificarán los elementos básicos del Transact SQL como: los identificadores, tipos de datos, funciones, expresiones, operadores, palabras reservadas y los comentarios.

Se presenta al **Lenguaje de Manipulación de Datos** *(DML, Data Manipulation Language)* y el uso de sus sentencias: SELECT, INSERT, DELETE y UPDATE, al igual que las cláusulas FROM, WHERE, GROUP BY, HAVING, ORDER BY, COMPUTE y OPTION pertenecientes a la Sentencia SELECT, al igual se hace mención al **Lenguaje de Definición de Datos** *(DDL, Data Definition Language)*, con el uso de las sentencias: CREATE TABLE, CREATE INDEX, CREATE VIEW, CREATE SYNONYM, CREATE SEQUENCE y CREATE TABLESPACE, para crear objetos dentro de la Base de Datos, así como las sentencias DROP y ALTER, para eliminar y actualizar respectivamente dichos objetos, sin olvidar las sentencias TRUNCATE, STORAGE.

Por último se mencionan las Consultas Multitabla *(JOINS)*, que permiten recuperar datos de dos o más tablas según las relaciones lógicas entre ellas.

- **Capítulo III. Arquitectura y Lenguaje de Programación en SQL Oracle**.

En el capítulo se presenta al Lenguaje de Programación Procedural de Oracle 9i, es decir, el PL/SQL (Programming Language/SQL), que incluye las sentencias SQL: SELECT, INSERT, DELETE y UPDATE, examinadas en el módulo II, además de incorporar sus propias sentencias de: control condicional (IF…THEN…ELSE…ENDIF) y ciclos (FOR …LOOP y WHILE…LOOP).

Se estudia y ejemplifica la sintaxis de PL/SQL, para la declaración de variables, el uso de comentarios, la creación de cursores, funciones, procedimientos, paquetes y disparadores (triggers), dentro de la Base de Datos.

- **Capítulo IV: Administración Gráfica y por Consola en SQL Server.**

El capítulo describe la importancia de las Bases de Datos del Sistema: MASTER, MODEL, MSDB y TEMPDB, para el funcionamiento de una Instancia de Microsoft SQL Server 2005, enfatizando en la Administración de Bases de Datos, es decir, creación, modificación y eliminación mediante el Administrador Corporativo de Microsoft SQL Server 2005y el Transact SQL.

Se abarca el tema de Administración de la Seguridad, desde la seguridad de una instancia hasta la autenticación y administración de usuarios, creando así los Funciones o Roles de Bases de Datos.

Resalta la utilidad de las Copias de Seguridad y Restauración de Bases de Datos en la Administración de Microsoft SQL Server 2005 en caso de un desastre natural o técnico.

Ejemplifica la Importación y Exportación de Datos, a través de los **Servicios de Transformación de Datos** (*DTS o Data Transformation Services)*, que facilitan las tareas administrativas del Servidor de Microosft SQL Server 2005.

- **Capítulo V. Administración Gráfica y por Consola en SQL Oracle.**

En el capítulo se presenta al Asistente de Configuración y el Asistente de Administración de Oracle 10g, para realizar las operaciones de creación, modificación o eliminación de una Base de Datos.

Se emplean los conceptos de usuario, rol y privilegio, así como su respectiva administración y seguridad, así como las funciones y responsabilidades del Administrador de la Base de Datos.

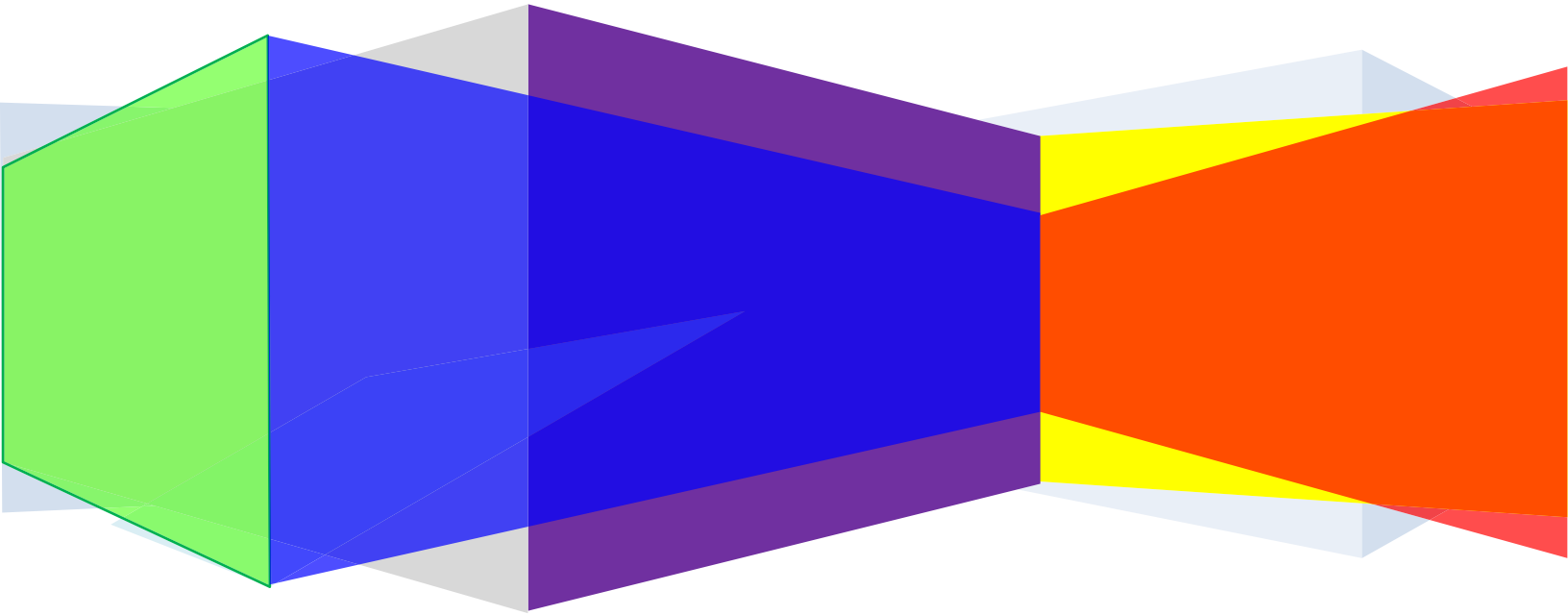
Para finalizar el capítulo se emplearon los distintos mecanismos de recuperación y copia de seguridad de la Base de Datos, para situaciones de caída del sistema.

- **Anexos.**

Es la parte de este informe que describe en forma general los proyectos finales, que evaluaron el conocimiento adquirido uno de los módulos del diplomado, presentando el análisis y diseño del sistema, en algunos casos se visualizan los códigos de los mismos y el software adecuado para su implantación.

CAPÍTULO I

Análisis y Diseño de Sistemas de Información según las Reglas de Negocios.



1.1. FUNDAMENTOS DE BASES DE DATOS.

En este módulo nos enseñaron los fundamentos del Diseño de Bases de Datos para constituirla como una solución integral de los Sistemas de Información, presentándolos continuación:

Los **datos** son hechos en bruto, que necesitan ser procesados para revelar su significado. Los datos contenidos en una Base de Datos son de dos tipos: **datos para el usuario final** que son hechos en bruto de vital importancia para este y los **metadatos** que son los datos acerca datos, es decir, comprenden las características y las relaciones de los propios datos.

Una **Base de Datos** está conformada por un conjunto de tablas lógicamente relacionadas entre sí. Una **tabla** es una estructura lógica bidimensional compuesta de **filas** denominadas **tuplas** o **registros** y **columnas** denominadas **campos** o **atributos**; un **registro** es un conjunto lógicamente conectado de uno o más atributos que describen a una persona, lugar o cosa, y un **atributo** es un carácter o grupo de caracteres (alfabéticos o numéricos), que definen y almacenan datos.

Por lo tanto una **Base de Datos** es una estructura lógica que almacena un conjunto de datos ordenados, coherentes y relacionados, que son procesados para generar información útil.

El **Diseño de Bases de Datos** es un proceso que describe la estructura que debe tener una **Base de Datos**, a través de modelos. Los **Modelos** son abstracciones de eventos y condiciones del mundo real, que permiten explorar las características de las entidades y sus relaciones. Una **Entidad**, es una persona, lugar o cosa sobre la que se desea reunir y guardar datos. Una **Relación** es una asociación entre entidades.

En concreto, un **Modelo de Bases de Datos** es un conjunto de construcciones lógicas (entidades y relaciones), utilizadas para describir y representar la estructura de los datos dentro de la **Base de Datos**, estos modelos se pueden agrupar en dos categorías:

a) El **Modelo Conceptual** enfocado en lo que está representado en la **Base de Datos**, y no en cómo está representado. Este tipo de modelos incluyen el Modelo de Entidad - Relación (E-R) y el Modelo Orientado a Objetos.

   Los Modelos Conceptuales utilizan tres tipos de relaciones para describir las asociaciones entre los datos: uno a muchos **(1:M)**, muchos a muchos **(M:N)** y uno a uno **(1:1)**. Los siguientes ejemplos ilustran las diferencias que existen entre las tres relaciones:

   1. **Relación Uno a Muchos:** un pintor pinta varios cuadros diferentes, pero cada uno de ellos es pintado únicamente por él; por lo tanto, el pintor (el "uno"), está relacionado con los cuadros (los "muchos").

      Relación **PINTOR** ***pinta*** **CUADROS** como ***"1:M"***

   2. **Relación Muchos a Muchos:** un empleado podría aprender varias especialidades de trabajo y cada una podría aprenderla muchos empleados.

      Relación **EMPLEADO** ***aprende*** **ESPECIALIDAD** como ***"M:N"***

   3. **Relación Uno a Uno:** una empresa necesita identificar a cada empleado con una matrícula y cada matrícula solo debe tener asignado uno y solo un empleado.

      Relación **EMPLEADO** ***tiene*** **MATRÍCULA** como ***"1:1"***

b) El **Modelo de Ejecución** enfocado en cómo los datos están representados en la **Base de Datos** o en cómo se ejecutan las estructuras de datos para representar lo que se está modelando. Este tipo de modelos incluyen el Modelo de Bases de Datos Jerárquico, de Red, el Relacional y el Orientado a Objetos.

Una **Base de Datos** bien diseñada facilita la **Administración de los Datos,** convirtiéndose en un valioso generador de información útil, mientras que una mal diseñada probablemente se encuentre repleta de **Datos Redundantes**, es decir, datos innecesariamente duplicados.

La **Normalización** es un proceso que reduce las redundancias de datos y ayuda a eliminar las anomalías en los datos, a través de la asignación de atributos a las entidades, dando como resultado la redundancia de datos controlada que permite vincular las tablas de la **Base de Datos**.

Un **Sistema de Administración de Base de Datos** *(DBMS, Database Management System)*, es un conjunto de programas que garantizan la integridad y la consistencia de los datos almacenados dentro de la **Base de Datos**. Sus funciones son las siguientes:

a) **Administración de un Diccionario de Datos.** El DBMS necesita almacenar definiciones de los datos, es decir, los metadatos.

b) **Administración del Almacenamiento de Datos.** El DBMS crea las estructuras complejas necesarias para el almacenamiento de datos.

c) **Transformación y Presentación de Datos.** El DBMS transforma los datos que se introducen de acuerdo con la estructura adecuada para guardarlos.

d) **Administración de la Seguridad.** El DBMS resguarda la seguridad del usuario y la privacidad de los datos dentro de la Base de Datos.

e) **Control de Acceso de Usuarios Múltiples.** El DBMS utiliza algoritmos complejos que permiten el acceso de usuarios múltiples a la Base de Datos, sin comprometer o dañar la integridad de ésta.

f) **Administración de Tareas de Respaldo y Recuperación.** La confiabilidad de recuperar la Base de Datos después de alguna falla.

g) **Administración de la Integridad de los Datos.** Reduce al mínimo la redundancia de datos.

h) **Lenguajes de Acceso a Base de Datos e Interfaces de Programación de Aplicaciones.** El DBMS permite el acceso a los datos mediante un **Lenguaje de Consulta** compuesto por: un **Lenguaje de Definición de Datos** *(DDL, Data Definition Language)*, que define las estructuras donde se alojan los datos y un **Lenguaje de Manipulación de Datos** *(DML, Data Modification Language)*, que permite que los usuarios extraigan los datos de la Base de Datos, asimismo también permite que los programadores tengan acceso a los datos mediante lenguajes de programación como COBOL, C, C++, C#, PASCAL, VISUAL BASIC, etc.

i) **Interfaces de Comunicación de Bases de Datos.** El DBMS proporciona rutinas de comunicación especiales que permiten que la Base de Datos acepte solicitudes del usuario final en un ambiente de Red de Computadoras.

1.2. DISEÑO DE UNA BASE DE DATOS.

Como la Base de Datos es el componente más básico y de mayor importancia para un Sistema de Información, el proceso de su diseño y ejecución es de particular interés para este módulo y a continuación se describe:

1. El análisis de los datos y la recopilación de los requerimientos ayudan a determinar las necesidades de los usuarios.

2. Se produce la definición de entidades, atributos y relaciones adecuadas que conducen a crear el Modelo Entidad-Relación, del que se desprenden un conjunto de tablas normalizadas.

3. El Modelo Conceptual se verifica mediante la identificación de sus procesos principales y mediante la definición de las reglas INSERT, UPDATE y DELETE. Asimismo se selecciona el software más adecuado para la Base de Datos.

4. En el Proceso de Verificación se revisan las visualizaciones del usuario y las restricciones en los datos.

5. Se analiza un diseño de Base de Datos Distribuida para evaluar la ubicación de las tablas y los requerimientos de acceso.

6. En el Diseño Lógico se transforma el esquema conceptual en la definición específica del DBMS de tablas, vistas, etcétera y por consiguiente es independiente del hardware.

7. En el Diseño Físico se definen las estructuras de almacenamiento y rutas de acceso específicas a los datos y por consiguiente es independiente del hardware.

8. La Ejecución es el paso final, en la cual se cargan los datos, se crean las tablas y visualizaciones. La codificación y pruebas de las aplicaciones se completan durante este paso.

1.3. INTRODUCCIÓN A LOS SISTEMAS DE INFORMACIÓN.

Se describe el concepto de Sistema de Información y sus características más esenciales. Un **Sistema de Información** es un conjunto de elementos que interactúan entre sí con el fin de satisfacer las demandas de información de una organización, para alcanzar sus objetivos o metas.

Todas las organizaciones son sistemas que actúan recíprocamente con su medio ambiente recibiendo entradas, procesándolas y produciendo salidas. Los sistemas pueden estar formados por otros sistemas más pequeños denominados **Subsistemas**, que son empleados para alcanzar fines específicos. Sin embargo, los objetivos, propósitos o metas se alcanzan sólo cuando se mantiene un modelo de control básico consistente en:

- Un ***estándar*** para lograr un desempeño aceptable.
- Un método para ***medir*** el desempeño actual.
- Un medio para ***comparar*** el desempeño actual contra el estándar.
- Un método de ***retroalimentación***.

- **Objetivos de los Sistemas de Información:**

a) Garantizar información exacta y confiable, así como su almacenamiento de tal forma que éste disponible cuando se necesite.

b) Proporcionar datos oportunos y exactos que permitan decisiones acertadas y mejorar la relación entre los recursos de la empresa.

- **Funciones de un Sistema de Información:**

a) **Entrada de Información**: Proceso que toma los datos que se requieren procesar para generar información, las entradas pueden ser manuales (proporcionadas directamente por el usuario), o automáticas (provenientes de otros sistemas o módulos, denominadas interfaces automáticas).

b) **Almacenamiento de Información**: El sistema recuerda la información guardada en la sección o proceso anterior, como todos los movimientos del mes (pagos, depuraciones, actualizaciones, etc.).

c) **Procesamiento de Información**: Capacidad para efectuar cálculos de acuerdo con una secuencia de operaciones preestablecida. Estos cálculos pueden efectuarse con datos introducidos recientemente en el sistema o bien con datos previamente almacenados, permitiendo la transformación de datos fuente en información.

d) **Salida de Información**: Es la información procesada o bien datos de entrada a otro sistema, es decir, una interface automática de salida. Por ejemplo, el Sistema de Control de Clientes tiene una interface automática de salida con el Sistema de Contabilidad, ya que genera las pólizas contables de los movimientos procesados de los clientes (reporte de pagos, estados de cuenta, pólizas contables, etc.).

- **Elementos de un Sistema de Información:**

a) El **Hardware**, está formado por las computadoras y el equipo periférico capaz de conectarse a ellas.

b) El **Recurso Humano o Usuarios,** está formado por todas las personas que utilizan el sistema, alimentándolo con datos o utilizando los resultados que genere. Se identifican de forma general cuatro tipos de usuarios:

- **Directivos.** Incorporan al Sistema; planes y estrategias de la organización, evaluando los riesgos a los que se expone la organización, originados por fallas.
- **Administradores.** Supervisan y controlar el desarrollo o uso del sistema.
- **Usuario Final Directo.** Opera directamente el sistema.
- **Usuario Final Indirecto.** Emplea la información generada por el sistema, pero no lo opera.

c) Los **Datos o Información Fuente**, son todas las entradas que necesita el sistema para generar como resultado la información que se desea (Bases de Datos, Informes, Reportes o Estadísticas).

d) Los **Programas,** son el software del sistema, que hará que los datos de entrada introducidos sean procesados correctamente y generen los resultados que se esperan.

e) Los **Procedimientos Administrativos**, son el conjunto de reglas y políticas de la organización, que rigen el comportamiento de los usuarios frente al sistema.

- **Tipos de Sistemas de Información:**

a) **Sistema para el Procesamiento de Transacciones (TPS).** Son los que llevan a cabo las actividades cotidianas de la organización. Los procedimientos estándares de operación que facilitan el manejo de las transacciones, los programas de cómputo que controlan la entrada de datos, almacenamiento y presentación tanto de datos como de información, etc.

b) **Sistema de Información Administrativa.** Están orientados hacia la toma de decisiones y utilizan datos relacionados con las transacciones así como cualquier otra información que sea generada dentro de la organización.

c) **Sistemas para el Soporte de Decisiones.** Tienen la finalidad de ayudar a los directivos que enfrentan problemas de decisión únicos (no recurrentes). Con frecuencia un aspecto importante de estas decisiones es determinar qué información es la que se debe considerar, dada la dificultad de predecir las necesidades de información, es posible diseñar de antemano los reportes.

1.4. EL PAPEL DEL ANALISTA DE SISTEMAS.

En este tema se consideran dos conceptos de vital importancia:

A. El **Análisis de Sistemas**, proceso de clasificación e interpretación de hechos, diagnóstico de problemas y empleo de la información existente, para hacer las modificaciones o mejoras al sistema.

B. El **Diseño de Sistemas**, proceso de planificar, reemplazar o complementar un Sistema existente.

En resumen el Análisis y Diseño de Sistemas se refiere al proceso de examinar la situación actual de la empresa; con el propósito de mejorarla con métodos y procedimientos más adecuados para la solución de problemas, así como los diferentes tipos de usuarios que participan en este proceso, el encargado de llevar a cabo estas tareas es el **Analista de Sistemas**.

Antes de que el **Analista de Sistemas** pueda diseñar un sistema para capturar datos, actualizar archivos y emitir reportes, necesita averiguar más acerca de la empresa como: la documentación con la cuenta, informes o reportes, listas de reabastecimiento, pedidos pendientes, facturas, registros manuales de almacén, cómo y dónde se origina esta información, es decir, el **Flujo de Información del Sistema**.

A la recopilación de toda esta información se le denomina **Estudio del Sistema**, y es el momento en el que el **Analista de Sistemas** se encuentra en la posición de determinar cómo y dónde un Sistema de Información será benéfico para todos los usuarios de la empresa.

En el caso en que el Estudio del Sistema este orientado hacia el futuro, y no existe algún sistema previo; el análisis comprende de manera meticulosa las necesidades futuras de la empresa y los cambios que deben considerarse para satisfacer esas necesidades, siendo lo más usual tener varias soluciones posibles.

Al trabajar con los gerentes y empleados de la organización los **Analistas de Sistemas** recomiendan qué solución(es) adoptar de acuerdo a las características de la empresa y al ambiente en particular de los usuarios, quienes trabajaran con el Sistema de Información. Es importante considerar que el tiempo necesario para desarrollar una solución, comparado con el de otras, es el aspecto más crítico, al igual que los costos y los beneficios también son factores determinantes.

1.5. ESTRATEGIAS PARA EL DESARROLLO DE SISTEMAS DE INFORMACIÓN.

Para iniciar el desarrollo de un Sistema de Información, se implementaron los siguientes métodos que a continuación se describen:

I. **Método del Ciclo de Vida de Desarrollo de Sistemas.** Es el conjunto de actividades que emprenden los analistas y diseñadores para desarrollar e implantar un Sistema de Información, incluye actividades de investigación preliminar, recolección de datos junto con la determinación de los requerimientos, el diseño de sistema, selección de software, pruebas, etc. Algunas de estas actividades pueden realizarse en forma paralela, permitiendo que diferentes partes del sistema se encuentren a igual tiempo, pero en distintos grados de avance.

II. **Método del Análisis Estructurado.** Este método incorpora elementos de Análisis y Diseño de Sistemas, teniendo como objetivo principal eliminar la dificultad de comprender sistemas robustos y complejos, a través de la **división del sistema en componentes** y la **construcción de un modelo del sistema**. Los elementos del Análisis Estructurado son:

   a) ***Descripción Gráfica.*** Se crea un bosquejo del sistema, que señale sus características y la interacción con otros elementos.

   b) ***Diagramas de Flujo de Datos (DFD).*** Herramienta que permite visualizar un sistema como una red de procesos funcionales, conectados entre sí por "líneas de flujo" y "bloques de almacenamiento de datos".

   c) ***Diccionario de Datos.*** Aquí se encuentran detalladas todas las definiciones de los elementos en el sistema: flujos de datos, procesos y almacenes de datos.

   d) ***Diseño Estructurado.*** Emplea la descripción gráfica enfocada en el desarrollo de especificaciones de software. Su objetivo es crear programas formados por módulos independientes unos de otros desde el punto de vista funcional, facilitando el mantenimiento de los mismos cuando surja la necesidad de hacerlo. Su herramienta es el **Diagrama Estructurado**.

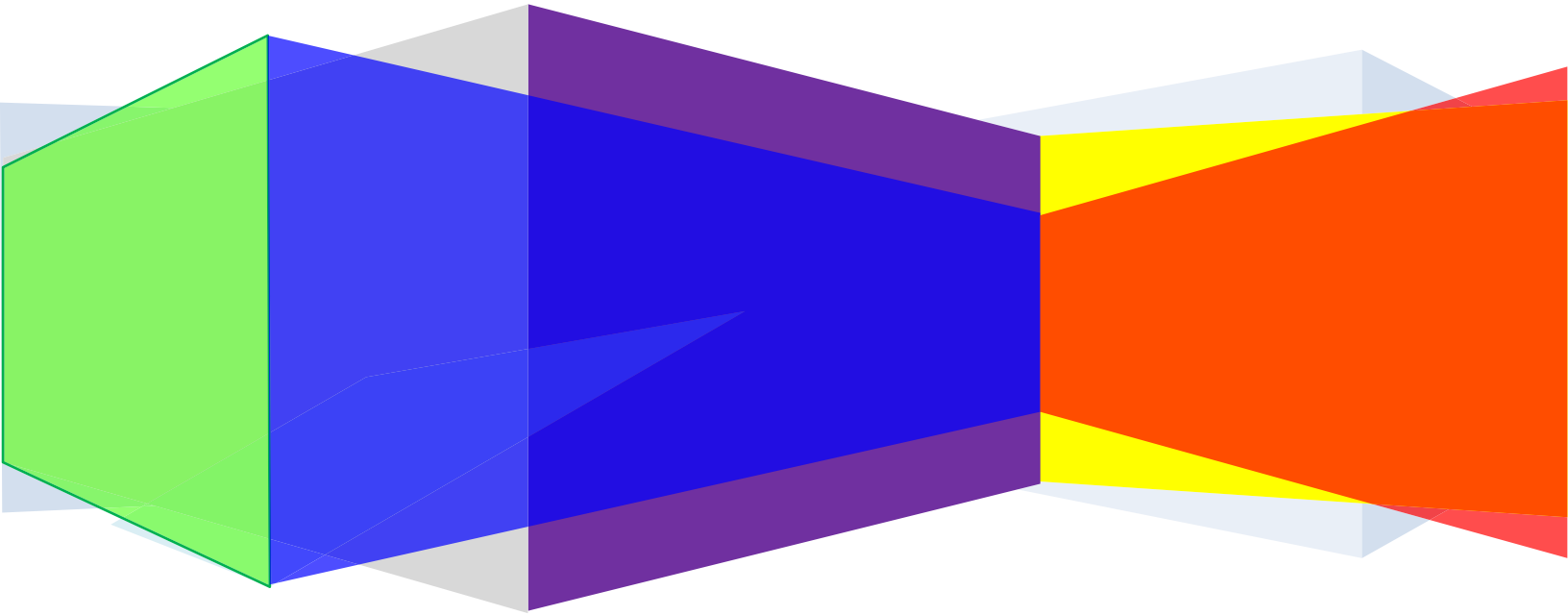
III. ***Método del Prototipo de Sistemas.*** La construcción de **Prototipos** es una estrategia de desarrollo apropiada cuando no es posible determinar todos los requerimientos del usuario, es decir, se crea una versión del Sistema de Información que se emplea de inmediato, teniendo los detalles necesarios en la interface con el usuario y un desempeño poco eficiente. El **Analista de Sistemas** junto con el usuario evalúan los resultados con la finalidad de identificar deficiencias, características faltantes y los ajustes necesarios. Cada vez que se repite este proceso se hacen mejoras y en determinado momento es posible que el prototipo se convierta en el sistema deseado o sea el inicio de un nuevo sistema.

1.6. PROYECTO 1. CREACIÓN DE UN SISTEMA DE INFORMACIÓN EN UN CENTRO DE CÓMPUTO.

La definición y desarrollo de este proyecto se describe en el Anexo I. Centro de Cómputo, de este documento.

CAPÍTULO II

Arquitectura y Lenguaje de Programación en SQL Server.



2.1. INTRODUCCIÓN A SQL SERVER 2005.

Esta parte del módulo nos capacita en el empleo de Microsoft SQL Server 2005, definiéndolo como un **Sistema de Gestión de Bases de Datos Relacionales** (***SGBDR***), desarrollado por Microsoft, conformada de dos partes esenciales:

a) **SQL (*Structured Query Language, Lenguaje de Consulta Estructurado*).** Es un Lenguaje para la Definición y Manipulación de Datos.

b) **Server**. Posee una Arquitectura Cliente/Servidor, que se encarga de atender los diferentes procesos de los clientes.

Microsoft SQL Server 2005 utiliza una extensión al SQL estándar, que se denomina **Transact SQL**, en otras palabras; soporta el SQL de ANSI, además de funciones adicionales, no contempladas en el estándar, y que son específicas para este producto.

Microsoft SQL Server 2005 cuenta con cuatro ediciones, según las necesidades de trabajo de los clientes**:**

- **Edición Express**. Fácil de usar, pensada para construir y poner en marcha aplicaciones sencillas de gestión de datos.
- **Edición Workgroup**. Utilizada para gestionar los datos en grupos de trabajo pequeños dentro de grandes corporaciones.
- **Edición Estándar**. Potencia, fiabilidad y robustez, para organizaciones con aplicaciones robustas en acceso a los datos. Soporta arquitecturas de 64 bits, funciones Data Warehouse y OLAP.
- **Edición Enterprise**. Máxima potencia y escalabilidad, para plataformas de 64 bits con todas las funcionalidades del producto.

2.2. COMPONENTES DE SQL SERVER 2005.

Se conoce la función principal de los componentes que integran a Microsoft SQL Server 2005, que se describen a continuación:

a) **Database Engine (*Motor de Base de Datos*)**. Servicio principal para almacenar, procesar y proteger los datos de la Base de Datos.

b) **Replication Services (*Servicios de Réplica*)**. Conjunto de tecnologías destinadas a la copia y distribución de datos y objetos de una Base de Datos a otra.

c) **Notification Services (*Servicios de Notificación*)**. Entorno orientado a desarrollar e implementar aplicaciones que generan y envían notificaciones o mensajes de información personalizados a una gran diversidad de dispositivos locales y móviles.

d) **Integration Services (*Servicios de Integración / SQL Server 2005 Integration Services [SSIS]*)**. Capacidades de extracción, transformación y carga de datos para almacenarlos e integrarlos. Incluye herramientas gráficas y asistentes para generar y depurar paquetes.

e) **Analysis Services (*Servicios de Análisis*)**. Funciones de Procesamiento Analítico en Línea (OLAP), permitiendo al usuario crear y administrar estructuras multidimensionales de datos agregados de Bases de Datos Relacionales y Minería de Datos (*Data Mining*), para aplicaciones de Inteligencia de Negocios (Business Intelligence).

f) **Reporting Services (*Servicios de Informes*)**. Capacidades para crear y administrar informes de manera física (impresos) o habilitados para la Web.

g) **Service Broker**. Ayuda a los programadores a crear aplicaciones de Base de Datos escalables, robustas y seguras.

h) **Búsqueda de Texto**. Se utiliza para realizar consultas de texto en datos sin formato.

2.3. HERRAMIENTAS DE SQL SERVER 2005.

Se conoce y emplea las herramientas del Administrador Corporativo Microsoft SQL Server Management Studio, que a continuación se describen:

a) **SQL Server Management Studio**. Herramienta para Administrar Bases de Datos Relacionales y de Business Intelligences, para escribir código Transact SQL, MDX y XML.

b) **Business Intelligence Development Studio**. Herramienta para desarrollar cubos de Business Intelligence.

c) **Configuración de Superficie de SQL Server 2005**. Herramienta para configurar opciones básicas de conectividad e inicio automático.

d) **Administrador de Configuración de SQL Server 2005**. Herramienta para configurar opciones avanzadas de inicio automático.

e) **Analizador de SQL Server 2005**. Herramienta para capturar y supervisar actividades en ejecución.

f) **Asistente para la Optimización del Motor de Base de Datos**. Herramienta para mejorar el rendimiento de la Base de Datos.

g) **Utilidades del Símbolo del Sistema**.

- Administrar y Configurar SQL Server 2005.
- Determinar la Información de Catálogo de una copia de SQL Server 2005.
- Diseñar y Probar Consultas para la Obtención de Datos.
- Copiar, Importar, Exportar y Transformar Datos.
- Proporcionar Información de Diagnóstico.
- Iniciar y Detener SQL Server 2005.

2.4. TRANSACT SQL.

Se conoce el **Transact SQL** definiéndolo como el Lenguaje de Base de Datos que utiliza Microsoft SQL Server 2005; para la creación, actualización y eliminación de objetos, y las categorías de sentencias que emplea para la manipulación de los datos contenidos en las Bases de Datos del sistema, por ejemplo:

a) **DQL (Data Query Language / Lenguaje de Consulta de Datos).** Utilizadas para obtener datos de Bases de Datos. El ejemplo esencial es SELECT.

b) **DDL (Data Definition Language / Lenguaje de Definición de Datos).** Utilizadas para crear, alterar o borrar objetos de Base de Datos, tales como: esquemas, tablas, columnas, vistas y secuencias. Ejemplos característicos son los comandos CREATE, ALTER y DROP.

c) **DML (Data Modification Language / Lenguaje de Modificación de Datos).** Utilizadas en la interrogación y manipulación de datos en esquemas ya existentes. Ejemplos característicos son los comandos INSERT, UPDATE y DELETE.

d) **TCL (Transaction Control Language / Lenguaje de Control de Transacciones).** Utilizadas para confirmar o restaurar transacciones de Base de Datos, que son unidades de trabajo que realizan una o más sentencias SQL relacionadas entre sí. Ejemplos característicos son los comandos COMMIT y ROLLBACK.

e) **DCL (Data Control Language / Lenguaje de Control de Datos).** Utilizadas en el control de acceso a datos en la Base de Datos. Ejemplos característicos son los comandos GRANT y REMOKE.

f) **CCL (Cursor Control Language / Lenguaje de Control de Cursores).** Utilizadas para operar sobre filas individuales de una tabla resultado que consta de varios registros. Ejemplos característicos son los comandos DECLARE CURSOR, FETCHINTO y UPDATE WHERE CURRENT.

Dentro del Transact SQL se tienen varios elementos de sintaxis que son utilizados en la mayor parte de las instrucciones. Destacando los siguientes:

a) **Identificadores.** Son los nombres de objetos como tablas, vistas, columnas, bases de datos y servidores.

b) **Tipos de Datos.** Definen el tipo de datos que contienen los objetos de la Base de Datos.

c) **Funciones.** Son elementos de sintaxis que toman cero, uno o más valores de entrada y devuelven un valor escalar o un conjunto de valores en forma de tabla.

d) **Expresiones.** Son unidades de sintaxis que Microsoft SQL Server 2005 puede resolver en valores únicos. Ejemplos de expresiones son las constantes, las funciones que devuelven un valor único, una referencia a una columna o una variable.

e) **Operadores.** Funcionan con una o más expresiones individuales para formar una expresión más compleja.

f) **Palabras Reservadas.** Son palabras que utiliza SQL Server 2005 y no deben emplearse como nombres de objetos de una Base de Datos.

g) **Comentarios.** Son fragmentos de texto insertado en instrucciones o secuencias de comandos de Transact SQL para explicar el objetivo de la instrucción.

2.5. SENTENCIAS O COMANDOS UTILIZADOS EN EL DIPLOMADO.

Se emplearon a través de ejemplos y ejercicios prácticos el uso de las sentencias CREATE, ALTER y DROP del Lenguaje de Definición de Datos (*DDL, Data Definition Language*), para crear, alterar o borrar respectivamente objetos de la Base de Datos, tales como:

I. **Tablas (TABLE).** Se conoce como la estructura de datos básica para las Bases de Datos Relacionales, constituida por un conjunto de registros o filas, todas ellas con los mismos atributos, campos o columnas.

Dentro de las tablas se contempla la "Definición de Columnas", parte que describe qué clase de datos o tipos de datos aceptará la columna y las "Restricciones de Integridad", entre las que destacan:

a) Restricciones **PRIMARY KEY (PK).** Cada tabla tiene una columna o combinación de columnas, cuyos valores identifican de forma única cada fila de la tabla.

b) Restricciones **FOREIGN KEY (FK).** Es una columna o combinación de columnas que se utiliza para establecer y exigir un vínculo entre datos de dos tablas.

c) Restricciones **UNIQUE.** Se utiliza para asegurar que no se escriban valores duplicados en columnas específicas que no formen parte de una clave principal o **PRIMARY KEY**.

d) Restricciones **CHECK.** Exigen la integridad del dominio mediante la limitación de los valores que puede aceptar una columna.

e) Restricciones **DEFAULT.** Cada columna debe contener un valor, aunque se trte de un valor NULL.

f) La aceptación de **NULL.** Determina si las filas de una tabla pueden contener un valor NULL o no, dentro de la columna.

II. **Índices (INDEX).** Son estructuras que proporcionan un acceso rápido a las filas de una tabla en base a los valores de una o más columnas.

III. **Vistas (VIEW).** Una vista es una tabla virtual cuyo contenido está definido por una consulta. Las filas y las columnas de datos, proceden de tablas a las que hace referencia la consulta que define la vista. Existen diferentes tipos de vistas:

a) **Vistas Horizontales.** Se restringe el acceso de un usuario únicamente en las filas seleccionadas de una tabla.

b) **Vistas Verticales.** Se restringe el acceso de un usuario únicamente a determinadas columnas seleccionadas de una tabla.

c) **Vistas Agrupadas.** Las que contienen una cláusula **GROUP BY** que permite agrupar filas relacionadas de datos y producir una fila de resultados para cada grupo que resume los datos de ese grupo.

d) **Vistas Compuestas.** Se crean especificando en la definición de la vista una consulta que involucre dos o más tablas.

IV. **Sinónimos (SYNONYM).** Es una redefinición de un objeto dentro de la Base de Datos.

V. **Secuencias (SEQUENCE).** Es un objeto de base de datos que genera números secuenciales. Se suele utilizar para asignar valores a campos autonuméricos.

a) La cláusula **START WITH** define el valor desde el que empezará la generación de números. Si no se incluye, se empezará a partir de MINVALUE.

b) La cláusula **INCREMENT BY** indica la diferencia que habrá entre un número y el siguiente. Puede ser cualquier número entero (positivo o negativo) distinto de 0.

c) La cláusula **MAXVALUE** indica el valor máximo que podrá alcanzar la secuencia. Se podrá incluir la cláusula **NOMAXVALUE** para no definir máximo de 1027.

d) La cláusula **MINVALUE** indica el valor mínimo de la secuencia. Se podrá incluir la cláusula **NOMINVALUE** para definir un mínimo de –1026.

e) La cláusula **CYCLE** permite que se empiece a contar en **MINVALUE** cuando se llegue a **MAXVALUE**. Por defecto las secuencias se crean **NOCYCLE**.

También se conoció de manera práctica el Lenguaje de Modificación de Datos (*DML, Data Modification Language*), cuyas sentencias se usan en la interrogación y manipulación de datos en esquemas de Bases de Datos ya existentes, por ejemplo:

I. **Sentencia SELECT. S**irve para recuperar registros de una o varias tablas o Bases de Datos, si se especifica el símbolo *, se obtendrán todos los campos de la tabla. La Sintaxis de la Sentencia SELECT se compone de tres partes: atributos, tablas y condición (opcional).

a) **FROM** <tablas>. Permite especificar la tabla de la cual se desean obtener los datos. Si se específica más de una tabla, éstas irán separadas por comas.

b) **WHERE** <condición>. Permite establecer una condición de recuperación de las filas de la(s) tabla(s). Sólo se obtendrán aquellas tuplas que verifiquen dicha condición, que será opcional. En el caso de que se omita esta parte, se recuperarán todas las filas.

c) **GROUP BY** <atributos>. Permite establecer una selección de campos cuando se utilizan funciones escalares o de conteo.

d) **HAVING** <condición>. Establece una condición para los atributos obtenidos como resultado de la aplicación de funciones escalares.

e) **ORDER BY** <atributos>. Permite obtener el resultado de la consulta ordenado por los atributos especificados.

f) **COMPUTE** <atributos>. Genera totales que aparecen como columnas de resumen adicionales al final del conjunto de resultados.

g) **OPTION**. Especifica que en toda la consulta se debe utilizar la sugerencia de consulta especificada.

En el caso de que se especifiquen varias tablas, en la cláusula **FROM**, será conveniente denotar los campos de la cláusula **SELECT** precedidos por el nombre de la tabla donde se encuentra y un punto, para que, en el caso de que dicho campo exista en más de una tabla, se sepa en cada momento a cuál de ellos nos estamos refiriendo, evitando en este caso el problema de ambigüedad.

Las funciones escalares que soporta la sentencia **SELECT** en el Transact SQL son las siguientes:

a) **SUM**. Realiza una suma acumulativa de un atributo para todas las filas accedidas mediante una consulta SQL.

b) **COUNT.** Cuenta todas las filas de las tablas accedidas mediante una consulta SQL.

c) **AVG.** Realiza una media aritmética de los atributos para todas las filas accedidas mediante la consulta SQL.

d) **MAX.** Obtiene el máximo valor del atributo especificado, de entre todas las filas seleccionadas mediante la sentencia SQL.

e) **MIN.** Obtiene el mínimo valor del atributo especificado, de entre todas las filas seleccionadas mediante la sentencia SQL.

II. **Sentencia INSERT.** Permite añadir datos al esquema, es decir, con esta sentencia podemos añadir información a la base de datos. Recordemos que estamos en el modelo relacional, por lo que la información se añadirá a una tabla en forma de filas. Si sólo queremos insertar un valor para un atributo, el resto de los de la tabla deberá contener el valor nulo (NULL). Sin embargo, habrá ciertas ocasiones en que esto no será posible, cuando el atributo esté definido como NO NULO, en cuyo caso deberemos especificar un valor para éste.

III. **Sentencia UPDATE.** Permite actualizar los valores de una o varias filas de una tabla, sin necesidad de borrarla e insertarla de nuevo.

IV. **Sentencia DELETE.** Permite borrar filas de una tabla, cumpliendo con las condiciones de seguridad determinadas por el administrador y las reglas de integridad referencial.

V. **Sentencia TRUNCATE.** La sentencia TRUNCATE pertenece al conjunto de las sentencias DDL, y permite vaciar todos los registros de una tabla.

VI. **Cláusula STORAGE.** El objetivo de esta cláusula es definir ciertas propiedades de almacenamiento para el objeto creado, por ejemplo:

a) La cláusula **INITIAL** define el tamaño que tendrá la extensión inicial y NEXT el tamaño de las siguientes extensiones.

b) La cláusula **MINEXTENTS** indica el número mínimo de extensiones para el objeto y **MAXEXTENTS** indica el máximo número de extensiones.

2.6. CONSULTAS MULTITABLA O *JOINS.*

Las **Consultas Multitabla** o **JOINS**, también denominadas combinaciones o composiciones, permiten recuperar datos de dos o más tablas según las relaciones lógicas entre ellas. Las combinaciones indican cómo debería utilizar SQL Server 2005 los datos de una tabla para seleccionar las filas de otra tabla. Una condición de combinación o composición define la forma en la que dos tablas se relacionan en una consulta al:

a) Especificar la columna de cada tabla que debe usarse para la combinación. Una condición de combinación típica especifica una clave externe de una tabla y su clave asociada en otra tabla.

b) Especificar un operador lógico, para usarlo en valores de comparación de las columnas.

Los distintos tipos de Consultas Multitabla o JOINS que se emplearon en este módulo, se describen a continuación:

a) **JOIN de Dos Tablas.** Es una composición entre las mismas tablas, basada en la coincidencia exacta de dos columnas, una de cada tabla. El JOIN forma parejas de filas haciendo coincidir los contenidos de las columnas relacionales.

b) **JOINS con Criterios de Selección de Filas.** El JOIN puede combinarse con otras condiciones de búsqueda que restrinjan aún más las filas devueltas en el resultado.

c) **JOINS con Múltiples Columnas de Emparejamiento.** Cuando dos tablas están relacionadas por varias relaciones clave foráneas/clave primaria, pueden especificarse ambos pares de columnas de emparejamiento en la condición de búsqueda del JOIN.

d) **JOINS de Tres o más Tablas.** Forma filas haciendo coincidir los contenidos de las columnas relacionadas, una de cada tabla.

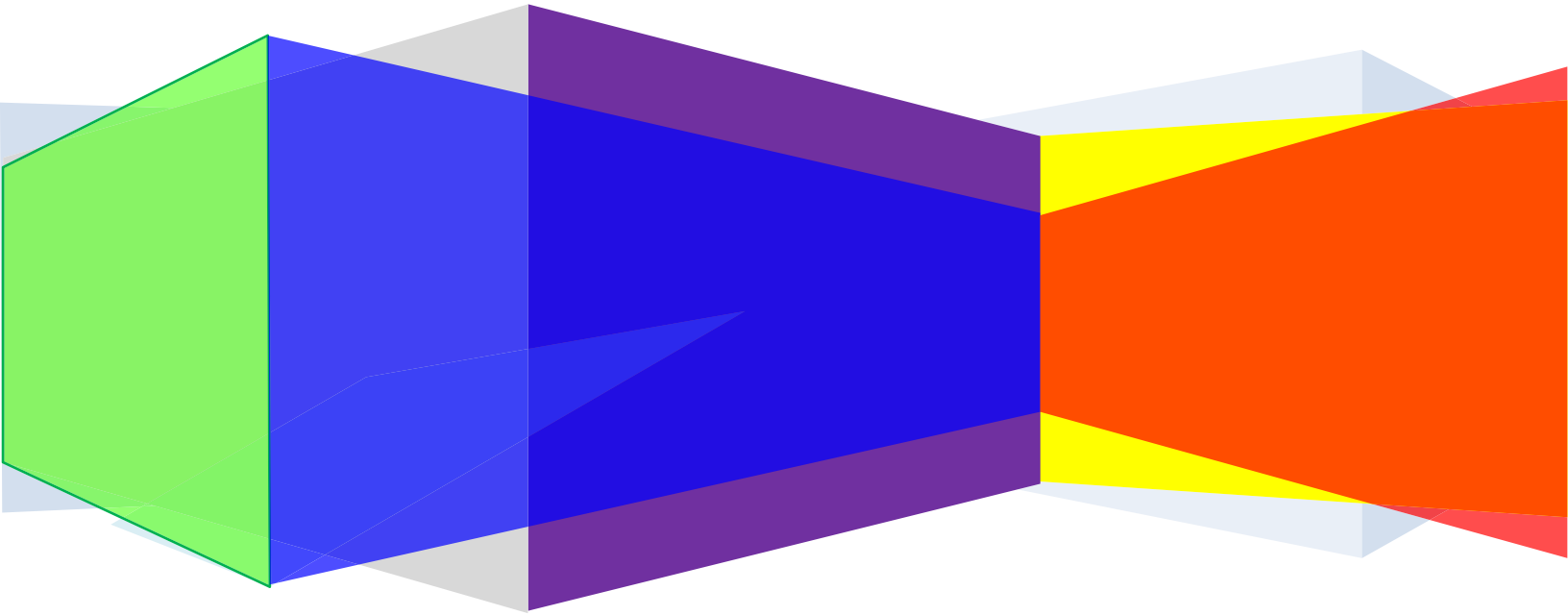
e) **INNER JOIN.** Combina registros de dos tablas siempre que existan valores coincidentes en un campo común.

2.7. PROYECTO 2: DISEÑO DE LA BASE DE DATOS DEL CATASTRO MUNICIPAL DE LA REPÚBLICA MEXICANA.

La definición y desarrollo de este proyecto se describe en el Anexo II. Catastro Municipal, de este documento.

CAPÍTULO III

Arquitectura y Lenguaje de Programación en SQL Oracle.



3.1. PROGRAMACIÓN EN PL/SQL.

Se emplea el Lenguaje de Programación de Oracle 9i, llamado **PL/SQL**, el cual es un lenguaje portable, procedural y de transacción muy potente, de fácil manejo, con las siguientes características:

a) Incluye todos los comandos de SQL estudiados en el capítulo II:

- SELECT.
- INSERT.
- UPDATE.
- DELETE.

b) Es una extensión de SQL, ya que este es un lenguaje no completo dado que no incluye las herramientas clásicas de programación. Por eso, PL/SQL amplia sus posibilidades al incorporar las siguientes sentencias:

- Control Condicional.

  IF ... THEN ... ELSE ... ENDIF

- Ciclos.

  FOR ... LOOP
  WHILE ... LOOP

c) Incorpora opciones avanzadas en:

- Control y tratamiento de errores llamado excepciones.
- Manejo de cursores.
- Variedad de procedimientos y funciones empaquetadas incorporadas en el módulo
- *SQL*Forms* para la programación de disparadores (***Trigger***) y procedimientos del usuario (***Procedure***).

3.2. FUNCIONES DE ORACLE 9i.

Se conocen las principales funciones de Oracle 9i, presentándolas a continuación:

- **Estructura del Bloque de Código.** Está compuesto por cuatro secciones *DECLARE, BEGIN, EXCEPTION y END* como se detalla a continuación:

  [<< nombre del bloque >>]
  Etiqueta que identifica al Bloque.

  [DECLARE]
  Declaración de
  Variable o Constante
  ExcepciónVariables para control de errores.

BEGIN
Código.

[EXCEPTION]

END [nombre del bloque];
Fin del Bloque.

- **Asignación de Valores.** Las dos formas que existen para asignar valores a variables de memoria, vistas en el ejemplo anterior, son:

  - Con el operador de asignación :=, como cuando calculamos el promedio de las ventas asignándole valor a la variable xprom con la siguiente sentencia:

    xprom:=xtotal/xcant;

  - Con la sentencia **SELECT** que contiene la orden **INTO**, como se muestra, es la asignación de valores a las variables xtotal y xcant con el siguiente código:

```
SELECT SUM(valor),
       COUNT(valor)
       INTO xtotal,xcant
       FROM ventas
       WHERE fecha=sysdate;
```

- ***SELECT* con Control de Excepciones.** La sentencia **SELECT** en **PL/SQL** no muestra en pantalla las filas resultantes de la consulta, como ocurre en Microsoft SQL Server 2005, sino que, según sea la acción a realizar, así será la cantidad de filas devueltas por la consulta, existiendo en este caso una de las tres posibles situaciones presentadas en la Tabla No. 1:

Cantidad de filas Acción	Acción
Una Se realiza la siguiente sentencia	Se realiza la siguiente sentencia.
Más de una	Ocurre la excepción TOO_MANY_ROWS
Ninguna	Ocurre la excepción NO_DATA_FOUND

Tabla No. 1

- **Manejo de Cursores.** El conjunto de filas resultantes de una consulta con la sentencia **SELECT**, puede estar compuesto por ninguna, una o varias filas, dependiendo de la condición que define la consulta. Para poder procesar individualmente cada fila de la consulta debemos definir un cursor, que contiene los datos de las filas de la tabla consultada por la sentencia SELECT. Los pasos para el manejo de cursores, son:

  - Definir el cursor, especificando la lista de parámetros con sus correspondientes tipos de
  - datos y estableciendo la consulta a realizar con la sentencia **SELECT**.
  - Abrir el cursor para inicializarlo, siendo éste el momento en que se realiza la consulta.
  - Leer una fila del cursor, pasando sus datos a las variables locales definidas a tal efecto.
  - Repetir el proceso fila a fila hasta llegar a la última.
  - Cerrar el cursor una vez que se terminó de procesar su última fila.

- **Disparadores.** El módulo ***SQL*Forms*** tiene incorporado una colección de procedimientos y funciones llamados "empaquetados" que se pueden incluir en el código de procedimientos o disparadores (*TRIGGER*) definidos por el usuario. El disparador es un bloque de código que se activa cuando se pulsa una determinada tecla u ocurre cierto evento, como puede ser:

    - Mover el cursor hacia o desde un campo, registro, bloque o forma.
    - Realizar una consulta.
    - Validar un dato.
    - Hacer una transacción al insertar, modificar o eliminar registros de la base de datos.

3.3. EL ENTORNO DE SQL PLUS.

Se emplea SQL*Plus como una herramienta que permite establecer conexión con el servidor de Base de Datos Oracle 9i y comenzar el trabajo con la información de la Base de Datos. Los comandos SQL para el trabajo con la Base de Datos se introducen en la línea de comandos finalizándose con un punto y coma. Al pulsar ENTER se ejecuta el comando y su resultado aparece inmediatamente debajo.

Las consultas SQL pueden ocupar varias líneas en la ventana de aplicación. Para pasar de una línea a otra basta pulsar ENTER. Finalizada la escritura de la consulta se ejecutará. Se observa que cada línea de una consulta SQL viene precedida de su correspondiente número.

Si una vez escrita y ejecutada una consulta, es necesario corregirla, lo más sencillo es utilizar el editor eligiendo Editar → Editor → Invocar Editor. Se obtiene la pantalla de edición de la consulta actual de la figura, en la cual se pueden hacer correcciones en modo editor de textos (por defecto, Oracle 9i utiliza el Bloc de Notas → Guardar → Salir y la nueva ventana sintaxis aparece en la ventana de aplicación.

Se utiliza SQL*Plus para manejar el Lenguaje SQL (que permite almacenar y recuperar datos en Oracle) y su extensión PL/SQL (que permite enlazar varios comandos SQL a través de procedimientos lógicos). SQL*Plus permite adicionales entre las que destacan las siguientes:

- Introducir, editar, almacenar, recuperar y ejecutar comandos SQL y bloques PL/SQL.
- Dar formato, ejecutar cálculos, almacenar e imprimir resultados de consulta en formato de informe.
- Listar definiciones de columnas para cualquier tabla.
- Acceder y copiar datos entre Bases de Datos SQL.
- Enviar mensajes a usuarios y recibir sus respuestas.
- Ejecutar tareas de administración de Bases de Datos.

En SQL*Plus es posible guardar la sintaxis generada al realizar consultas o construir comandos para utilizarla o editarla posteriormente.

- Para guardar en fichero la sintaxis del comando actual se utiliza SAVE nombre_fichero.
- Para editar de nuevo el fichero almacenado previamente se usa GET nombre_fichero.
- Para ejecutar la sintaxis de un comando almacenado se usa START nombre_fichero o también se usa @ nombre_fichero.
- Para ejecutar la sintaxis de la última consulta se usa el símbolo "/" en la línea de comandos.
- Para borrar el búfer de memoria se utiliza CLEAR BUFFER.

3.4. iSQL*PLUS.

Entre los componentes SQL*Plus se empleo iSQL*Plus, que es un navegador basado en la interfaz de SQL*Plus que posibilita utilizar un navegador Web para conectar a Oracle 9i y ejecutar las mismas tareas que se realizarían en la línea de comandos clásica de SQL*Plus. Es posible utilizar iSQL*Plus para escribir comandos SQL*Plus, SQL y PL/SQL con vistas a:

- Introducir, editar, ejecutar y guardar comandos SQL y bloques PL/SQL..
- Calcular e imprimir resultados de consultas.
- Listar definiciones de columnas para cualquier tabla.
- Acceder y copiar datos entre Bases de Datos.
- Ejecutar administración de Bases de Datos.

La Arquitectura de iSQL*PLUS se ajusta a un modelo de tres niveles comprendiendo:

a) Nivel Cliente (la interfaz de usuario de iSQL*Plus, generalmente un navegador Web).
b) Nivel Intermedio (iSQL*Plus Server, Oracle Net y Oracle HTTP Server).
c) Nivel Base de Datos (Oracle 9i).

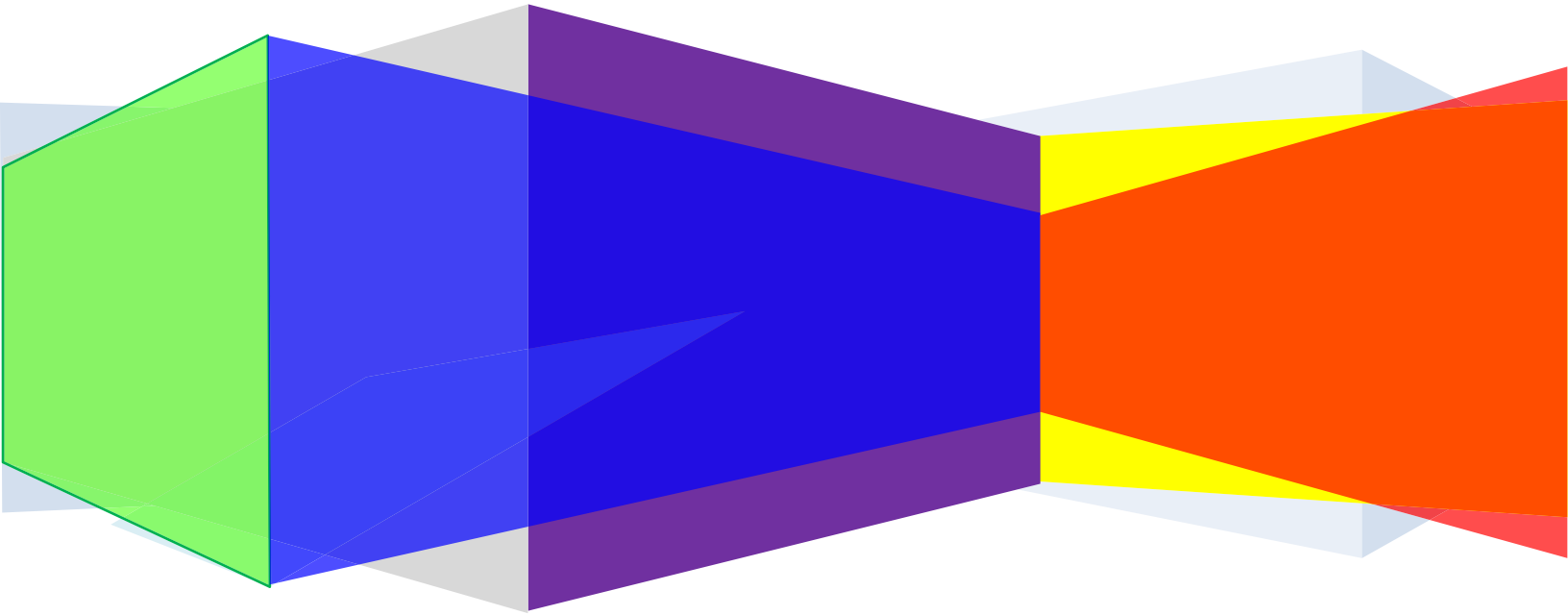
La interfaz de usuario iSQL*PLUS se ejecuta en un anvegador Web conectado a Internet o la Intarnet local. Sólo es necesario conocer la URL de Oracle HTTP para acceder a Oracle 9i. El nivel intermedio contiene los servidores Oracle HTTP e interfaz de usuario de iSQL*PLUS y Oracle 9i. Cada sesión iSQL*PLUS es unívocamnete identificada, ya que se haber múltiples sesiones concurrentes abiertas en Oracle 9i.

3.5. PROYECTO 3. CREACIÓN DE UNA BASE DE DATOS PARA UNA UNIVERSIDAD.

Los códigos para crear la Base de Datos de una Universidad, fueron elaborados durante todo este tercer módulo, reforzando los conocimientos adquiridos del Módulo II. Arquitectura y Lenguaje de Programación en SQL Server, los cuales se encuentran en el Anexo III. Universidad, de este documento.

CAPÍTULO IV

Administración Gráfica y por Consola en SQL Server.



4.1. INICIO DEL SERVIDOR DE SQL SERVER 2005.

Este módulo se enfoca al uso del Administrador Corporativo de Microsoft SQL Server 2005; aplicando los temas abordados en el Capítulo II. Arquitectura y Lenguaje de Programación en SQL Server, de este documento.

Para empezar a trabajar con Microsoft SQL Server 2005, es necesario arrancar el Servidor de la Base de Datos, es decir, iniciar una Instancia de Base de Datos, asociar esta Instancia con la Base de Datos, para posteriormente poder acceder a la Base de Datos, como se ejemplifica a continuación:

Paso 1. Ir al Botón Inicio → Todos los Programas → Microsoft SQL Server 2005 → SQL Server Management Studio Express.

Paso 2. Aparece la pantalla de Configuración de Entorno.

Paso 3. Aparece la pantallas Conexión al Servidor y Modo de Autenticación de Windows.

Paso 4. Dar clic en el botón ***Connect*** y automáticamente se realiza la conexión al Servidor, obteniendo el Administrador Corporativo Microsoft SQL Server Management Studio Express. Se observa que después de la conexión, el icono del servidor para la instancia que aparece en el panel de la izquierda, se identifica como un círculo verde en su interior (instancia conectada), que por el momento es la instancia que se instala por defecto en Microsoft SQL Server 2005.

4.2. INSTANCIA EN SQL SERVER 2005.

Se tiene presente que al iniciar el Administrador Corporativo Microsoft SQL Server Management Studio Express, presenta un esquema de la(s) instancia(s) que posee; considerando que una **Instancia** de SQL Server 2005, es la agrupación de Bases de Datos, Sistema de Usuario, Sistema de Administración y Asignación de Memoria (a Nivel de Instancia y no A Nivel de Bases de Datos), Seguridad y Servicios.

Las Bases de Datos de una Instancia se clasifican en:

a) **Bases de Datos del Sistema.** Contienen la información necesaria para que la instancia funcione.

- **MASTER.** Base de Datos del Sistema vital para el funcionamiento de la instancia, que contiene el catálogo de la instancia para vistas, tablas, etc.
- **MODEL.** Contiene los scripts de creación de objetos y gestiona las plantillas para la creación de nuevas Bases de Datos.
- **MSDB.** Almacena todas las tareas que se crean, incluyendo las tareas programadas del servidor.
- **TEMPDB.** Gestiona la creación automática de objetos temporales que adquieren importancia al realizar operaciones de ordenación, agrupación, etc.

b) **Bases de Datos de Usuario de Datos.** Son las creadas por el usuario y se agrupan en las instancias debajo de las Bases de Datos del Sistema.

Las instancias de SQL Server 2005 tienen asociados distintos servicios, que se especifican a continuación:

a) **Servicio SQL Server (MSSQLSERVER).** Arranca y detiene la instancia, proporciona almacenamiento, procesamiento y acceso controlado de datos.

b) **Servicio SQL Server Analisys Services (MSSQLSERVER).** Proporciona Procesamiento Analítico en Línea (OLAP) y funcionalidades de Minería de Datos.

c) **Servicio SQL Server Integration Services.** Proporciona soporte de administración para el almacenamiento y ejecución de paquetes DTS.

d) **Servicio SQL Server Reporting Services (MSSQLSERVER).** Administra, ejecuta, soporta, programa y entrega informes.

e) **Servicio Agente SQL Server (MSSQLSERVER).** Ejecuta trabajos, supervisa SQL Server 2005 y activa alertas.

f) **Servicio Búsqueda de Texto de SQL Server (MSSQLSERVER).** Facilita las tareas de búsqueda de texto, ordenación y generación de índices de texto de contenido y propiedades de datos estructurados y semiestructurados, para habilitar sus búsquedas lingüísticas.

4.3. ADMINISTRACIÓN DE BASES DE DATOS EN SQL SERVER 2005.

Usando el Administrador Corporativo Microsoft SQL Server Management Studio Express se gestionan las tareas esenciales de la Administración de Base de Datos, como la creación, modificación y eliminación.

a) Para crear una Base de Datos mediante el Administrador Corporativo Microsoft SQL Server Management Studio Express, se despliega su árbol de navegación, situándose sobre la carpeta ***Databases*** (Bases de Datos), haciendo clic sobre ella con el botón derecho del ratón y eligiendo la opción ***New Database*** (Nueva Base de Datos). A continuación se rellenan adecuadamente los campos de las pantallas ***General*** (General), ***Option*** (Opciones) y ***Filegroups*** (Grupos de Archivos) de la pantalla ***New Database*** (Nueva Base de Datos).

b) Para modificar una Base de Datos mediante el Administrador Corporativo Microsoft SQL Server Management Studio Express, se despliega su árbol de navegación, situándose sobre la Base de Datos a modificar, haciendo clic sobre ella con el botón derecho del ratón y eligiendo la opción ***Properties*** (Propiedades). A continuación se rellenan adecuadamente los campos de las pantallas ***General*** (General), ***Files*** (Archivos), ***Filegroups*** (Grupos de Archivos), ***Options*** (Opciones), ***Permissions*** (Permisos) y ***Extended Properties*** (Propiedades Extendidas) de la pantalla ***Database Properties*** (Propiedades de la Base de Datos).

c) Para eliminar una Base de Datos mediante el Administrador Corporativo Microsoft SQL Server Management Studio Express, se despliega su árbol de navegación situándose sobre la Base de Datos a eliminar, haciendo clic sobre ella con el botón derecho del ratón y eligiendo la opción ***Delete*** (Eliminar).

4.4. SEGURIDAD EN SQL SERVER 2005.

Con el Administrador Corporativo Microsoft SQL Server Management Studio Express, se aprendió a establecer la seguridad del Servidor de SQL Server 2005, a través de la Autenticación y Administración de Usuarios, la Administración de Privilegios, Funciones y Contraseñas de Usuario, tanto para la(s) Instancia(s) como para las Bases de Datos y sus objetos.

a) En la **Seguridad de Instancia**, se establecen condiciones para acceder a esta, para crear nuevas Bases de Datos, etc.

b) En la **Seguridad de Bases de Datos**, se establecen permisos para acceder a estas, para realizar operaciones sobre sus objetos, etc.

Todo esto es posible desplegando el árbol del Administrador Corporativo de Microsoft SQL Server Management Studio Express, para ver los objetos de una Instancia u objetos de la(s) Base(s) de Dato(s) dentro de la Instancia; se encuentra la opción ***Security*** (Seguridad), cuyos apartados permitirán definir la seguridad de la propia Instancia o la(s) Base(s) de Dato(s).

4.5. AUTENTICACIÓN Y ADMINISTRACIÓN DE USUARIOS EN SQL SERVER 2005.

Se establece que cualquier usuario que desee conectarse a una Base de Datos, solo podrá hacerlo con un *Nombre de Usuario* determinado; para que Microsoft SQL Server 2005 autentique si dicho individuo está autorizado a usar la cuenta. Cada usuario puede obtener acceso a una Instancia de Microsoft SQL Server 2005 a través de una *Cuenta de Inicio de Sesión,* que define su autenticación.

A esta *Cuenta de Inicio de Sesión* se le asigna una *Cuenta de Usuario* de SQL Server 2005, que se utiliza para controlar todas las acciones realizadas en la Base de Datos (validación de permisos). Posteriormente, se concede permiso a este *Inicio de Sesión* para conectarse a una Instancia de Microsoft SQL Server 2005.

Es importante destacar que si en una Base de Datos no existe una *Cuenta de Usuario*; el usuario no podrá acceder a ella, aunque pueda conectarse a una Instancia de Microsoft SQL Server 2005, ya que el *Inicio de Sesión* se crea en Microsoft Windows y no en Microsoft SQL Server 2005.

Se contempla que los nombres de inicio de sesión, usuarios, funciones y contraseñas de Microsoft SQL Server 2005, pueden contener de 1 a 128 caracteres (letras, símbolos y números), exceptuando el carácter barra diagonal inversa (\), salvo que se refiera a un usuario o grupo existente de Windows, ya que separa el nombre del equipo o dominio de Windows del nombre del usuario.

Las actividades realizadas para este tema, a través del Administrador Corporativo de Microsoft SQL Server Management Studio Express, son las siguientes:

a) Crear una Contraseña de Usuario para Acceso a SQL Server 2005.

b) Crear una Cuenta de Inicio de Sesión en SQL Server 2005.

c) Conceder a cuentas de SQL Server el Acceso a una Base de Datos.

4.6. COPIAS DE SEGURIDAD Y DUPLICACIÓN DE BASES DE DATOS EN SQL SERVER 2005.

En este tema se realizó y comprobó la actividad de realizar Copias de Seguridad de las Bases de Datos en un equipo de manera rápida, sencilla y eficaz; para posteriormente restaurarlas en otros equipos sin perder la información de estas, a través de cada uno de los tres Modelos de Recuperación de Bases de Datos de Microsoft SQL Server 2005, que a continuación se presentan:

a) **Modelo de Recuperación Simple.** Recupera la Base de Datos completa hasta la copia de seguridad (bachup) más reciente.

b) **Modelo de Recuperación Completa.** Recupera la Base de Datos hasta el momento del error (último backup y las operaciones de log hasta el momento de la caída).

c) **Modelo de Recuperación de Registro Masivo.** Recupera hasta el último registro log.

En cuanto a la actividad de Duplicación de Bases de Datos, se ejemplifico al copiar y enviar los datos a uno más equipos de cómputo (servidores), permitiendo que los usuarios realicen sus propias modificaciones, para posteriormente sincronizar todas las Bases de Datos y lograr su coherencia. En el proceso de Duplicación se manejaron los siguientes conceptos:

a) **Publicador.** Servidor que coloca los datos a disposición de otros servidores para duplicarlos.

b) **Distribuidor.** Servidor que aloja la Base de Datos de distribución y almacena los datos históricos, transacciones y metadatos. Existen dos tipos de Distribuidor:

- **Distribuidor Remoto** es un servidor ubicado en un lugar diferente al Publicador, configurado como distribuidor de duplicación.

- **Distribuidor Local** es un servidor configurado como publicador y distribuidor de duplicación.

c) **Suscriptores.** Servidores que reciben los datos duplicados.

d) **Publicación.** Conjunto de uno o más artículos; es decir, datos de una Base de Datos, que facilita especificar un conjunto de datos relacionados lógicamente u objetos de Base de Datos, que se requieran duplicar conjuntamente.

e) **Artículo.** Puede ser una tabla entera, algunas columnas o filas, un procedimiento almacenado, una vista o función, para realizar su duplicación.

f) **Suscripción.** Petición de copia de datos o de objetos de base de datos para duplicar.

Existen tres tipos de Duplicación en Microsoft SQL Server 2005:

a) **Duplicación de Instantáneas.** Proceso de copia y distribución de datos y objetos de Bases de Datos exactamente como aparecen en un tiempo determinado.

b) **Duplicación Transaccional.** Se emite una instantánea inicial de datos a los suscriptores, posteriormente se efectúan modificaciones en el publicador y las transacciones individuales se capturan, para enviarlos de nuevo a los suscriptores.

c) **Duplicación de Mezcla.** Permite que varios sitios funcionen en línea o desconectados de manera autónoma, mezclando las modificaciones de datos efectuadas en un resultado único y uniforme.

4.7. IMPORTACIÓN Y EXPORTACIÓN DE DATOS EN SQL SERVER 2005.

Se conoce y empleo el proceso de Importación de Datos, que consiste en recuperar los datos desde orígenes externos a Microsoft SQL Server 2005, por ejemplo: datos de texto ASCII, para posteriormente insertarlos en sus tablas. Asimismo se trabajo con la Exportación de Datos, que consiste en extraer datos desde una instancia de Microsoft SQL Server 2005 a formatos especificados por el usuario, por ejemplo: una Hoja de cálculo en Excel o una tabla en Microsoft Access).

Se utilizaron los Servicios de Transformación de Datos (DTS, Data Transformation Services), definidos como un conjunto de herramientas gráficas y objetos programables que permiten la extracción, transformación y consolidación de datos de distintos orígenes en uno o varios destinos, a través de:

a) Importación y Exportación de Datos. Los DTS puede importar datos de un archivo de texto o de un origen de datos OLE DB en datos para Microsoft SQL Server 2005 y viceversa.

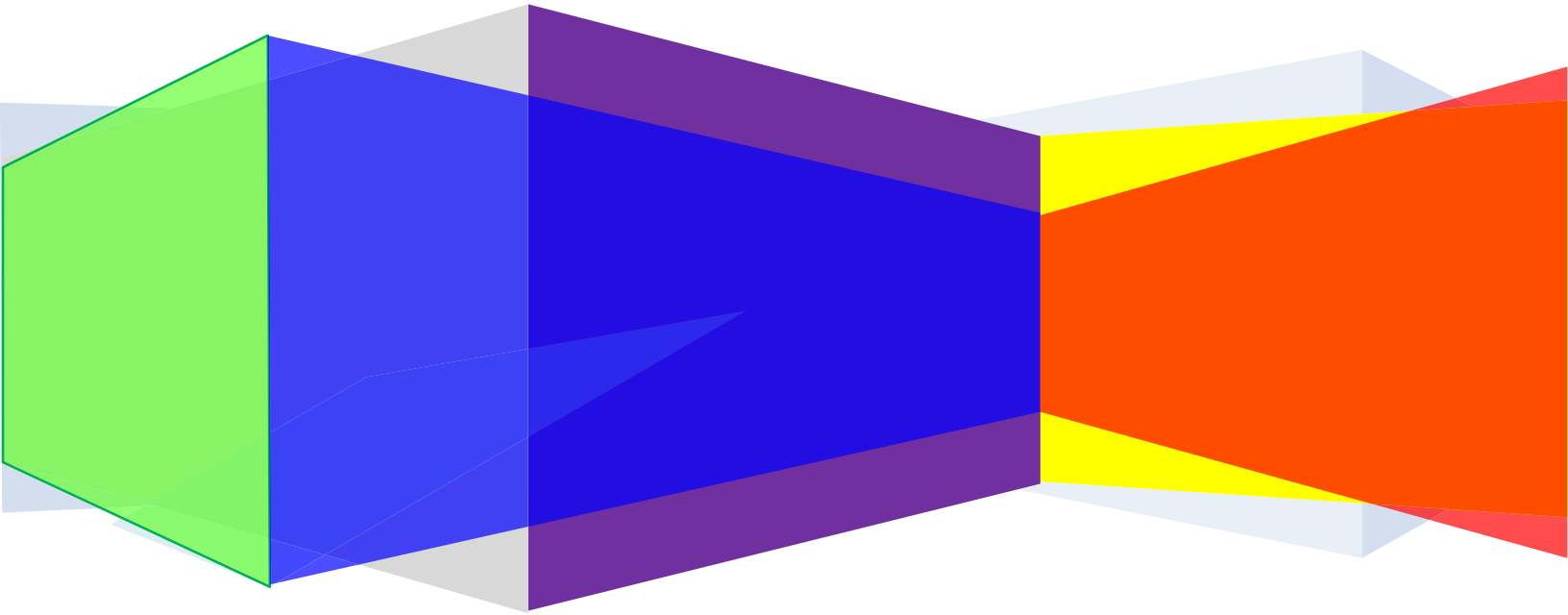
b) Copia de Objetos de Base de Datos. Los DTS pueden transferir; datos, índices, vistas, inicios de sesión, procedimientos almacenados, desencadenadores, reglas, valores predeterminados, restricciones y tipos de datos definidos por el usuario.

4.8. PROYECTO 4. CREACIÓN DE UNA BASE DE DATOS PARA EL CONTROL DE INVENTARIOS EN UNA TIENDA DE AUTOSERVICIO.

La definición, desarrollo y los códigos empleados para la elaboración de proyecto se encuentran en el Anexo IV. Tienda de Autoservicio, de este documento.

CAPÍTULO V

Administración Gráfica y por Consola en SQL Oracle.



5.1. ASISTENTE DE CONFIGURACIÓN DE ORACLE 10g.

En este módulo se emplea el Asistente de Configuración de Bases de Datos de Oracle 10g, para realizar las operaciones de creación, modificación o eliminación de una Base de Datos, asimismo se aprende a crear una Base de Datos de una lista de plantillas predefinidas o emplear una Base de Datos consolidada, para crear una nueva Base de Datos o una nueva plantilla.

Todo lo anterior es posible gracias a la Gestión de Plantillas, que ahorran tiempo al guardar la definición de la Base de Datos, en formato XML dentro de la unidad de disco duro local (C:). Existen tres formas para crear una plantilla y a continuación se describen:

a) De una **Plantilla existente**, crea una nueva plantilla, empleando los valores de una plantilla predefinida.

b) De una **Plantilla existente (sólo estructura)**, crea una nueva plantilla idéntica a la Base de Datos existente.

c) De una **Base de Datos existente (estructura y datos)**, Crea una nueva plantilla con la estructura y los datos de una Base de Datos existente.

5.2. ASISTENTE DE ADMINISTRACIÓN DE ORACLE 10g.

Se emplea el Asistente de Administración de Oracle para Microsoft Windows XP, como una herramienta que permite:

a) Creación y Gestión de Administradores de Bases de Datos del Sistema Operativo para Instancias.

b) Creación y Gestión de Operadores de Bases de Datos del Sistema Operativo para Instancias.

c) Creación y Gestión de Usuarios Externos del Sistema Operativo.

d) Creación y Gestión de Roles Locales.

e) Creación y Gestión de Roles Externos del Sistema Operativo.

a) Creación de un Administrador de Bases de Datos del Sistema Operativo para todas las Instancias de Bases de Datos.

b) Creación de un Operador de Base de Datos del Sistema Operativo para todas las Instancias de Base de Datos.

5.3. AUTENTICACIÓN Y ADMINISTRACIÓN DE USUARIOS EN ORACLE 10g.

Al igual que Microsoft SQL Server 2005, se establece que cualquier usuario que desee conectarse a una Base de Datos, solo podrá hacerlo con un *Nombre de Usuario* determinado; para que Oracle 10g autentique si dicho individuo está autorizado a usar la cuenta. Cada usuario puede obtener acceso a una Instancia de Oracle 10g a través de una *Cuenta de Inicio de Sesión,* que define su autenticación.

Oracle 10g utiliza la autenticación de contraseña, autenticación de sistema operativo y autenticación global de usuario, las cuales se explican a continuación:

a) **Autenticación de Contraseña**, consiste en un nombre de usuario y una contraseña asociada, típica en sistemas distribuidos (Cliente/Servidor).

b) **Autenticación de Sistema Operativo**, consiste en que Oracle 10g autentica un nombre de usuario usando el sistema operativo del computador que ejecuta el servidor de Base de Datos, típica en sistemas de terminales con conexión directa al servidor.

c) **Autenticación Global de Usuario**, consiste en un nombre de usuario usando un servicio de red externo, suele utilizarse en redes no seguras y en accesos de usuarios a varias Bases de Datos de Oracle 10g.

5.4. ADMINISTRACIÓN DE PRIVILEGIOS EN ORACLE 10g.

Después de aprender a crear usuarios en Oracle 10g, es importante asignarles privilegios para que puedan realizar operaciones específicas en la(s) Base(s) de Datos(s). Estos privilegios suelen clasificarse en:

a) **Privilegios del Sistema**, que permiten al usuario realizar operaciones que afectan a todo el sistema. Los privilegios de sistema más comunes son: **CREATE SESSION** (conexión al servidor de Base de Datos y establecimiento de sesión), **CREATE TABLE** (crear una tabla en el esquema propio), **CREATE ANY NTABLE** (crear una tabla en cualquier esquema de la Base de Datos), **EXECUTE ANY PROCEDURE** (ejecutar cualquier procedimiento almacenado, función o componente de paquete de Base de Datos) y **ALTER DATABASE** (modificar la estructura física y la capacidad del Sistema de Base de Datos).

b) **Privilegios de Objeto**, que permiten al usuario realizar operaciones específicas sobre objetos específicos de la Base de Datos. Los privilegios de objeto dependen de que tipo sea este, es decir, para una tabla los privilegios típicos son. **SELECT**, **INSERT**, **UPDATE**, **DELETE**, **ALTER**, **INDEX** y **REFERENCES**. Para una vista los privilegios más comunes son: **SELECT**, **INSERT**, **UPDATE** y **DELETE**. Para una secuencia los privilegios más comunes son: **SELECT** y **ALTER**, y para un procedimiento el privilegio más común es **EXECUTE**.

5.5. ROLES.

Los roles utilizados para este módulo son:

a) **Rol CONNECT**, que permite al usuario conectarse a la Base de datos y crear tablas, vistas, secuencias, etc.

b) **Rol RESOURCE**, que permite al usuario utilizar los recursos típicos para la programación de aplicaciones (clústeres, disparadores, paquetes, funciones, etc.).

c) **Rol DBA**, que permite al usuario realizar cualquier función de Base de Datos y disponer de cualquier privilegio.

d) **Rol EXECUTE_CATALOG_ROLE**, que permite al usuario ejecutar los paquetes de utilidad DBMS incorporados.

e) **Rol SELECT_CATALOG_ROLE**, que permite al usuario realizar peticiones a las vistas del diccionario de datos del administrador.

5.6. MECANISMOS DE PROTECCIÓN DE DATOS DE ORACLE 10g.

Se conocieron y aplicaron los distintos mecanismos de protección para situaciones de caída del sistema, fallos del disco duro, etc. Los mecanismos de recuperación y copia de seguridad deben recuperar la Base de Datos de cualquier tipo de pérdida de archivos, a través de:

a) **Registro de Transacciones.** Se conecta a la Base de Datos, pero con privilegios de administrador, para obtener acceso a los privilegios: **SYSOPER** que puede iniciar y cerrar un servidor de Base de Datos Oracle, montar, abrir, crear, copiar y recuperar una Base de Datos y administrar su estructura de registro de transacciones, y **SYSDBA** que puede llevar a cabo cualquier operación de Base de Datos y conceder cualquier privilegio de sistema a otros usuarios.

b) **Copia de Seguridad de la Base de Datos (BACKUP) con RECOVERY MANAGER (RMAN).** Utilidad que permite hacer rápidamente un backup de toda la Base de Datos o de una parte de esta.

c) **Recuperación de Bases de Datos con RMAN.** Una vez realizados copias de seguridad con el comando **BACKUP**, pueden recuperarse posteriormente con el comando RECOVER o de modo similar con **RECOVER DATABASE**. Por otro lado Oracle 10g también puede recuperar objetos de la Base de Datos dañados ante una caída de sistema, una pérdida de datos, etcétera, utilizando el comando **RESTORE**.

5.7. PROYECTO 5: DISEÑO DE UN SISTEMA DE VENTAS PARA UNA TIENDA DE ABARROTES.

La definición y desarrollo de este proyecto se describe en el Anexo V. Sistema de Ventas, de este documento.

CONCLUSIONES.

Los datos constituyen la materia prima con la que se crea la información, esta transformación de los datos en información útil e indispensable para las organizaciones, se produce cuando el software (código de programación), opera sobre los datos, dando origen a la producción de aplicaciones avanzadas como los "Sistemas de Información".

Los Sistemas de Información pertenecen al rubro de las Tecnologías de la Información, en su análisis, diseño y desarrollo intervienen diversos factores, destacando al factor humano como el principal.

Para la implementación de un Sistema de Información es importante enfatizar que tal situación de cambio con esa magnitud dentro de cualquier organización, probablemente ocasioné que el personal se muestre renuente a adoptar los nuevos procedimientos o que los desarrolle plenamente de acuerdo a los lineamientos que se establecieron para la implantación del sistema.

Desde el Análisis y Diseño de una Base de Datos, considerando que es una labor difícil, larga, exhaustiva y costosa; las implicaciones y repercusiones de su creación hasta la integración con el Sistema de Información aplica a toda la organización, convirtiéndola en una Política Empresarial, comúnmente conocida como "Reglas de Negocios".

Por este motivo se realiza una planeación estratégica tomando en cuenta todos los requerimientos o las necesidades presentes y futuras de la organización, es decir, se realiza una investigación preliminar y un estudio de factibilidad del proyecto que deseamos implantar.

El objetivo principal de este diplomado es conocer el Panorama de los Sistemas de Información en el Marco de una Economía Global, desde la Investigación Preliminar, el Papel que desempeñan los Usuarios y el proceso de examinar la Factibilidad Operacional, Técnica y Económica de un Proyecto, a través del uso y manejo de las diversas herramientas que brinda Microsoft SQL Server 2005, Oracle 9i y Oracle 10g.

El conocimiento adquirido en el diplomado, se centra en el Proceso de Desarrollo de Sistemas de Información, enfatizando la integración de las Bases de Datos como una solución integral de estos proyectos de software, asumiendo el papel de Analista de Sistemas, considerando la necesidad de automatizar las actividades y procesos de una determinada organización, en función de los resultados que genera.

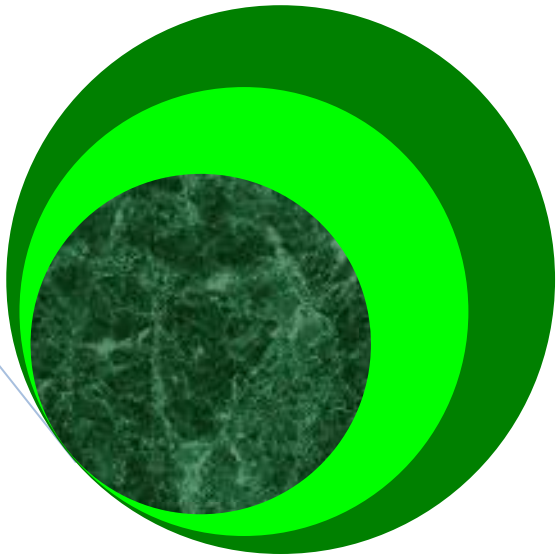
Utilizando eficientemente las Tecnologías de la Información se pueden lograr ventajas competitivas inimaginables, pero es indispensable encontrar procedimientos acertados para mantener tales ventajas como una constante, en un mundo que evoluciona a pasos colosales y que se encuentra inmerso en las nuevas tecnologías que salen al mercado.

El Sistema de Información tiene que modificarse y actualizarse con suma regularidad si se desea tener ventajas competitivas continuas. El uso creativo e inteligente de las Tecnologías de la Información puede proporcionar a los administradores y ejecutivos de cualquier organización, una poderosa herramienta para diferenciar sus recursos humanos, productos y/o servicios respecto de sus competidores, creando nuevas estrategias de mercado, que permiten producir una gran variedad productos a un precio más bajo y en menor tiempo que la competencia.

Dentro de los conocimientos adquiridos en el diplomado cabe destacar que un buen Sistema de Información podría transformar cualquier organización en una más competitiva. La organización puede contar con el personal capacitado y motivado, pero si éste carece de información no podrá desempeñarse al nivel de su potencial.

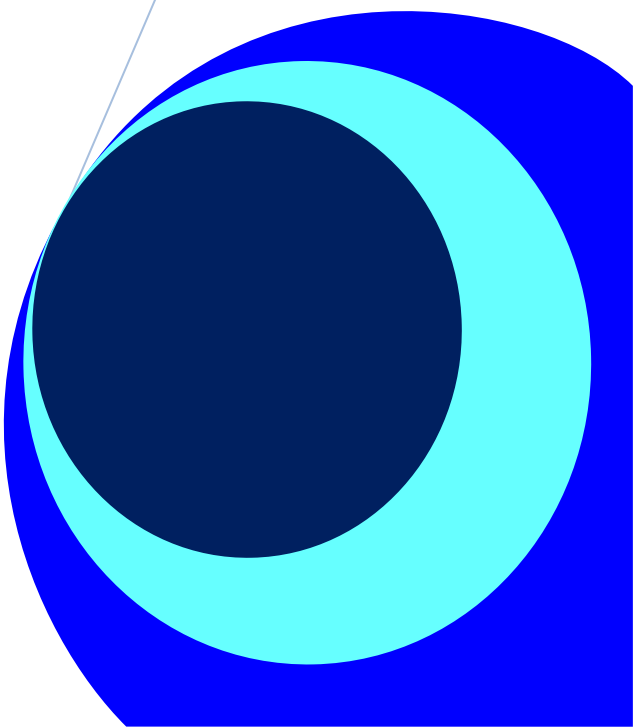
La elaboración de un sólido Sistema de Información Orientado a Negocios es una de las tareas más importantes que enfrentan las organizaciones de cualquier tamaño y ámbito, en especial aquellas en crecimiento.

Por tal motivo un buen Sistema de Información puede revolucionar el trabajo del personal, siempre y cuando el sistema sea capaz de producir información precisa, oportuna y útil sobre sus operaciones y procesos.



ANEXO I

Centro de Cómputo.



CENTRO DE CÓMPUTO.

- **Presentación.**

Un Centro de Cómputo es el departamento encargado de satisfacer las necesidades de información de una organización, de manera precisa, veraz y oportuna. Es responsable de centralizar, procesar y custodiar gran parte de los datos con los que opera la compañía, es decir, la mayoría de las actividades y toma de decisiones que realizan otros departamentos se basan en la información proporcionada por este mismo.

- **Objetivos de un Centro de Cómputo:**

1. Concentrar el procesamiento de datos e información de manera sistematizada y automática.
2. Generar información útil para la organización.
3. Prestar servicios a las diferentes áreas o departamentos dentro de la organización y fuera de ella.

- **Elementos de un Centro de Cómputo:**

a) **Hardware.** Es el conjunto de elementos físicamente tangibles en un sistema de cómputo.

b) **Software.** Es el conjunto de programas, instrucciones y reglas informáticas para realizar ciertas tareas en un sistema de cómputo.

- **Funciones de un Centro de Cómputo:**

a) Operar y mantener el Sistema de Cómputo Central funcionando apropiadamente y estar vigilante para detectar y corregir fallas en el mismo.

b) Ejecutar los procesos asignados conforme a los programas de producción y calendarios preestablecidos.

c) Registrar y Revisar los resultados de los procesos e incorporar acciones correctivas o preventivas.

d) Marcar y/o señalizar los productos de los procesos ejecutados.

e) Llevar registros de fallas, problemas, soluciones, acciones desarrolladas, respaldos, recuperaciones y trabajos realizados.

f) Realizar las copias de respaldo (back-up) de la información y procesos de cómputo conforme a los parámetros preestablecidos.

g) Cumplir y aplicar estrictamente las normas, reglamentos y procedimientos establecidos por la Dirección para el desarrollo de las funciones asignadas.

h) Realizar labores de mantenimiento y limpieza a los equipos de cómputo y telecomunicaciones.

i) Mantener informado al jefe inmediato sobre el funcionamiento del Centro de Cómputo.

- **Principales Departamentos de un Centro de Cómputo:**

a) **Explotación de Sistemas o Aplicaciones.** Consiste en la utilización y aprovechamiento de los sistemas con los que cuenta la organización, a través de previsión de fechas de realización de trabajos, operación general del sistema, control y manejo de soportes, seguridad del sistema, supervisión de trabajos, etc.

b) **Soporte Técnico. S**e ocupa de seleccionar, instalar y mantener el sistema operativo adecuado en los equipos de cómputo, el control y mantenimiento de la estructura de la base de datos, la gestión de los equipos de teleproceso, el estudio y evaluación de las necesidades y rendimientos del sistema y la ayuda directa a usuarios.

c) **Gestión y Administración del propio Centro de Cómputo.** Engloban operaciones de supervisión, planificación y control de proyectos, seguridad general de las instalaciones, equipos y sistemas, etc.

- **Sistemas de Distribución de Cableado en un Centro de Cómputo:**

Los sistemas de distribución de cableado especifican la forma como es canalizado y distribuido el cableado en un centro de cómputo. Pueden ser: suelo falso, techo falso y escalerillas.

a) **Características Eléctricas.** Existen dos sistemas de distribución de energía; el AC y el DC, pero para energizar un centro de cómputo se debe usar un sistema AC. El centro de cómputo deberá constar con un sistema de respaldo de energía. En caso de fallo de energía eléctrica un generador entrará en funcionamiento.

b) **Condiciones Ambientales.** Un Centro de Cómputo deberá mantenerse a una climatización adecuada para equipos de telecomunicaciones. Equipos acondicionadores de aire deberán usarse para regular temperatura y humedad dentro de la sala.

c) **Cableado Estructurado.** Los cables usados son: UTP, STP, F.O., como los más principales.

- **Estándares para el montaje de Sistemas de Distribución de Cableado en un Centro de Cómputo:**

Las principales organizaciones como el ANSI, TIA, EIA, BICSI, IEEE y el NFPA entre otras, emiten periódicamente nuevos estándares para la edificación de Centros de Cómputos, por ejemplo:

a) Estándar **ANSI/TIA/EIA-568A** de alambrado de telecomunicaciones para edificios comerciales.

b) Estándar **ANSI/TIA/EIA-569** de rutas y espacios de telecomunicaciones para edificios comerciales.

c) Estándar **ANSI/TIA/EIA-606** de administración para la infraestructura de telecomunicaciones de edificios comerciales.

d) Estándar **ANSI/TIA/EIA-607** de requerimientos de puesta a tierra y punteado de telecomunicaciones de edificios comerciales.

e) **Building Industry Consulting Service International**, manual de métodos de distribución de telecomunicaciones.

f) **ISO/IEC 11801** cableado genérico.

g) **Código Eléctrico Nacional 1992 (CODEC),** establece normas para la manipulación de conductores y equipos eléctricos.

PROYECTO 1. CREACIÓN DE UN SISTEMA DE INFORMACIÓN EN UN CENTRO DE CÓMPUTO.

- **Definición del Proyecto.**

El proyecto comprende la metodología básica para la elaboración de un Sistema de Información, considerando:

a) Número de Horas requeridas.

b) Recursos Humanos requeridos.

c) Diagramas de Tiempo.

- **Objetivo:**

Establecer una guía para la elaboración de un Sistema de Información en un Centro de Cómputo.

- **Cronograma de Actividades.**

Tarea	Descripción	Esfuerzo (Meses)	Tipo Brooks	Recursos Humanos	Tareas Antecesoras
A	Análisis de Requerimientos	3	1	2 Analistas	
B	Diseño de la Base de Datos	1	2	1 Analista	A
C	Diseño de Procesos	4	1	2 Analistas	A
D	Construcción de Prototipo	1	2	1 Programador	C, E
E	Desarrollo de Esquema	0.5	1	1 Analista	B
F	Codificación	8	1	4 Programadores	C, E
G	Revisión de Prototipo	0.5	2	1 Analista	D
H	Revisión Código con Mejoras Solicitadas	2	1	2 Programadores	F, G
I	Pruebas	2	1	2 Programadores	H
J	Instalación de Sistema	1	1	2 Programadores	I
K	Mantenimiento Inicial	2	2	1 Programador	J

Tabla No. 2

Se observa en primera instancia, 11 tareas o actividades contempladas, sus tareas antecesoras, el tiempo necesario para llevarlas a cabo y el número de recursos necesarios para la realización del Sistema de Información.

A continuación en la Tabla No., se presenta su realización en un Diagrama de Gantt, que muestra gráficamente la utilización de tiempos por cada recurso y los tiempos totales.

Tarea	Descripción	Esfuerzo (Meses)	R.H.	Total	T.A.	Meses																					
						1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
A	Análisis de Requerimientos	3.0	2	6.0																							
B	Diseño de la Base de Datos	1.0	1	1.0	A																						
C	Diseño de Procesos	4.0	2	8.0	A																						
D	Construcción de Prototipo	1.0	1	1.0	C, E																						
E	Desarrollo de Esquema	0.5	1	0.5	B																						
F	Codificación	8.0	4	32.0	C, E																						
G	Revisión de Prototipo	0.5	1	0.5	D																						
H	Revisión de Código con Mejoras Solicitadas	2.0	2	4.0	F, G																						
I	Pruebas	2.0	2	4.0	H																						
J	Instalación de Sistema	1.0	2	2.0	I																						
K	Mantenimiento Inicial	2.0	1	2.0	J																						
	Total	**25.0**	**19**	**61.0**																							

Tabla No. 3

R.H. (Recursos Humanos)
T.A. (Tareas Antecesoras)

Podemos observar que el proyecto se termina en 22 Meses, utilizando 61 Meses/Recurso (Meses/Hombre), lo cual ya nos permite tener un estimado preliminar de los tiempos del proyecto. Lo anterior, nos lleva a obtener el Diagrama de Ruta Critica para nuestro proyecto, que se muestra en la Figura No. 1:

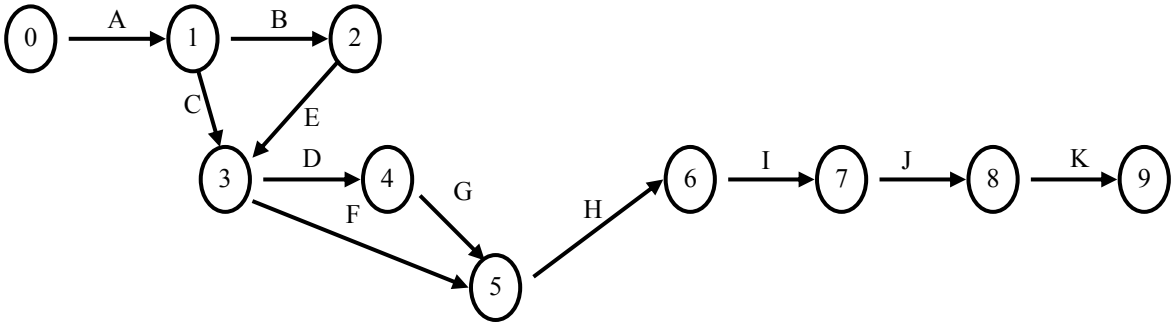


Figura No. 1

Donde se coloca entre paréntesis los tiempos acumulados de realización de cada actividad, permitiendo hacer un seguimiento hasta la conclusión del proyecto, que se muestra en la Figura No. 2:

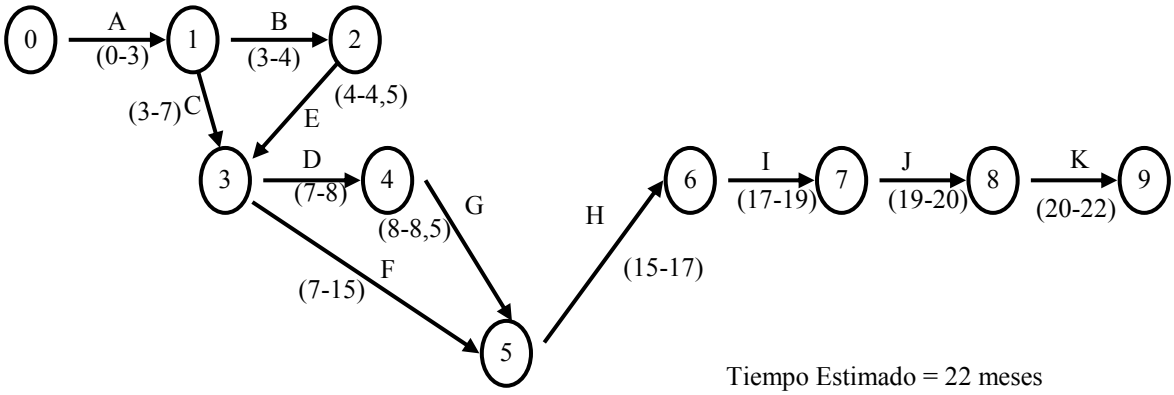


Figura No. 2

Asimismo se definen los tiempos de holgura y la ruta crítica esperada, que se muestra en la Figura No. 3:

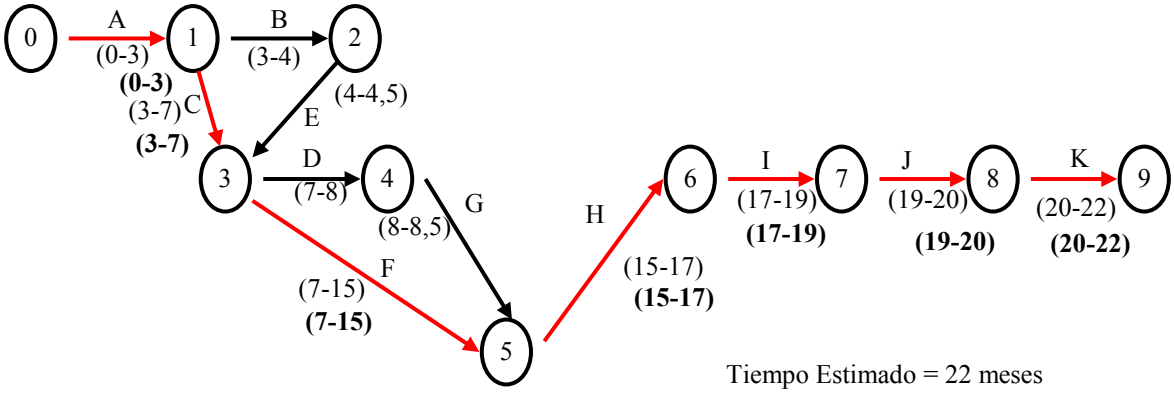


Figura No. 3

ANEXO II

Catastro Municipal.

CATASTRO MUNICIPAL DE LA REPÚBLICA MEXICANA.

- **Presentación.**

El Catastro es una función realizada por la Administración Pública de cada Municipio, que consiste en un Censo Analítico con el propósito de ubicar, describir y registrar las características físicas, económicas y jurídicas de cada bien inmueble como son: la referencia catastral, el valor catastral, el titular catastral, la superficie, el uso o destino, la clase de cultivo o aprovechamiento, la calidad de construcción, la representación gráfica, etcétera; que lo definen tanto material como espacialmente.

Actualmente el Catastro Municipal se ejecuta en colaboración estrecha con el Gobierno de los Estados de la República Mexicana.

- **Objetivos del Catastro Municipal:**

a) Determinar el Valor Catastral de los bienes inmuebles asentados en su territorio.
b) Conocer la situación Jurídica y Económica de los bienes inmuebles con respecto a sus propietarios.
c) Captación de recursos a través del cobro de diferentes impuestos a la propiedad inmobiliaria de acuerdo a su Valor Catastral.

- **Tipos de Catastro Municipal:**

1. **Catastro Urbano.** Es más complejo debido a que el uso de los predios y construcciones es de carácter productivo, lo que origina que la propiedad inmobiliaria se destine a fines industriales, comerciales y sociales, pues en territorios de esta índole las modificaciones patrimoniales suelen ser más frecuentes y valor mucho más elevadas.

   El Catastro Urbano se orienta a la tasación y valuación de predios y construcciones, así como el apoyo en la formulación de planes de desarrollo urbano, de zonas conurbadas o bien de zonas que son propensas a la urbanización.

2. **Catastro Rural.** Destaca la utilidad productiva del suelo en materia agropecuaria, razón por la cual las construcciones dentro de éste mismo, aunque tomadas en cuenta, no tienen el mismo interés que los predios.

   El Catastro Rural se orienta a detectar los usos productivos del suelo rural y ubicar a los propietarios de los predios rurales.

- **Funciones del Catastro Municipal:**

1. **Administración del Impuesto Predial.** Tiene como objetivo primordial la valuación catastral a través del cobro de las contribuciones o impuestos prediales, así como el registro de sus propietarios.

2. **Actualización de Registros Catastrales.** Los bienes inmuebles tienden a sufrir cambios constantes como transferencia de derechos de propiedad o modificaciones físicas a la construcción, por lo que es necesario registrar todos estos cambios, para actualizar los planos de manzana o zona, con .lo cual se produce un nuevo avalúo y se cobra un nuevo impuesto.

3. **Recepción y Envío de Documentos.** Los Catastros Municipales reciben, emiten y envían mucha documentación como: notificaciones de avalúo, pago de impuesto, modificaciones al padrón de causantes o contribuyentes, etc. Asimismo los Catastros deben tener una sección de archivo para preservar toda esta información generada y recibida.

4. **Apoyos a la Comunidad y al Gobierno del Estado**. El catastro Municipal es el área encargada de prestar apoyos diversos para la planeación del desarrollo urbano municipal, para la planeación y prestación de los servicios públicos, así como para el correcto cumplimiento de los compromisos contraídos con el Gobierno del Estado de conformidad con los convenios que se hayan celebrado con éste.

- **Bases Jurídicas del Catastro Municipal:**

1. **Nivel Federal.** Se ocupa de la propiedad de tierras comprendidas dentro de los límites del territorio municipal, así como las modalidades que pueden imponerse a dicha propiedad, por interés público.

2. **Nivel Estatal.** Establece los elementos generales que regulan el comportamiento de la propiedad inmobiliaria, su registro y las competencias que les corresponden a los municipios.

3. **Nivel Municipal.** Establece los elementos que sirvan de orientación en el manejo de los registros catastrales para la administración del impuesto sobre el bien inmueble.

PROYECTO 2. DISEÑO DE LA BASE DE DATOS DEL CATASTRO MUNICIPAL DE LA REPÚBLICA MEXICANA.

- **Descripción del Proyecto.**

La Base de Datos del Catastro Municipal es un inventario metódicamente ordenado de datos concernientes a los bienes inmuebles y sus propietarios de cualquier Municipio de la República Mexicana, basado en la mensura de sus límites, mostrando su naturaleza, tamaño, valor y los derechos o restricciones legales asociados.

- **Objetivo:**

Constituir un registro territorial, con la finalidad de capturar información, incorporarle valor añadido, distribuirla y publicitarla a Nivel Nacional.

- **Alcance:**

La Base de Datos se puede tomar como plataforma primordial de un Sistema de Información del Territorio Mexicano, al servicio de todas las Administraciones Nacionales y del Ciudadano.

- **Límites:**

La Base de Datos no contempla el uso y resguardo de Cartografía y Topografía Digitalizada.

- **Identificación de Necesidades:**

a) **Administrativos.** Organizar y actualizar toda la documentación oficial acerca de los bienes inmuebles y sus propietarios.

b) **Tecnológicos.** Manejo electrónico de la documentación oficial de bienes inmuebles y sus propietarios, así como la interpretación de datos catastrales.

- **Estudio de Factibilidad:**

a) **Técnica.** En el mercado existe toda la tecnología necesaria para soportar el sistema, tanto en hardware (Servidores, Equipos de Interconexión, Unidades de Almacenamiento Externo, etc.) y Software (Plataformas Windows al igual que paquetería y herramientas especializado para Bases de datos, etc.).

b) **Económica.** Se cuenta con el capital necesario para desarrollar el proyecto, tanto la compra de hardware como las licencias del software necesario para la implementación).

c) **Operativa.** Para realizar el proyecto se necesitaría capacitar al personal existente, lo cual se llevaría a cabo mediante cursos, este aspecto sería factible ya que el sistema se desarrollará en un entorno visual, lo cual es amigable para el usuario. No será necesaria la contratación de personal extra para operar el sistema.

- **Ventajas:**

1. Registro público, que permite la consulta y certificación de sus datos.
2. Coordinación con otros sistemas registrales.
3. Inventario total de los bienes inmuebles.
4. Mantener actualizado los datos y registros catastrales.
5. Determinar y gestionar el cobro del impuesto predial.

- **Descripción General de las Tablas de la Base de Datos.**

Sobre este punto se despliegan cinco rubros de variables o campos:

1. Información General acerca de la República Mexicana.

a) **Tabla ESTADO**. Se considera 31 Estados y un Distrito Federal, es decir 32 entidades federativas.

Atributos:

- Código del Estado.
- Nombre del Estado.
- Superficie del Estado.
- Coordenadas del Estado.
- Total de Municipios del Estado.
- Total de Habitantes del Estado.

b) **Tabla MUNICIPIO.** Se contemplan 2,456 municipios en toda la República Mexicana.

Atributos:

- Código del Municipio.
- Nombre del Municipio.
- Superficie del Municipio.
- Coordenadas del Municipio.
- Total de Colonias del Municipio.
- Total de Habitantes del Municipio.

c) **Tabla COLONIA.** Se contemplan 23,273 colonias en toda la República Mexicana.

Atributos:

- Código de la Colonia.
- Nombre de la Colonia.
- Código Postal de la Colonia.
- Superficie de la Colonia.
- Coordenadas de la Colonia.
- Total de Manzanas de la Colonia.
- Total de Lotes de la Colonia.
- Total de Calles de la Colonia.
- Total de Avenidas de la Colonia.
- Total de Habitantes de la Colonia.

2. Características, Identificación y Ubicación de los Predios.

a) **Tabla CATASTRO.** Las variables o campos que tienen como propósito describir cómo está constituido el predio en cuestión.

Atributos:

- Clave Catastral.
- Entidad Federativa.
- Municipio.
- Localidad.
- Colonia.
- Domicilio Oficial.
- Nombre de Inmueble.

3. Propietarios.

a) **Tabla PROPIETARIO.** Se requiere tener información acerca del propietario de cada bien inmueble.

Atributos:

- Nombre.
- Fecha de Nacimiento.
- Lugar de Nacimiento.
- Sexo.
- CURP.
- Ocupación.
- Estado Civil.
- Domicilio Oficial.

4. Características de la Construcción.

a) **Tabla CONSTRUCCION.** Las variables o campos que tienen como propósito describir cómo está construido el predio en cuestión.

Atributos:

- Ámbito.
- Régimen de Propiedad.
- Superficie.
- Clasificación del Predio urbano.
- Clase de Tierra del Predio Rural.
- Uso del Suelo Urbano.
- Uso del Suelo Rural.
- Servicios Básicos.
- Forma.
- Topografía o Relieve.
- Tamaño (Frente y Fondo).
- Ubicación en la Manzana.

5. Aspectos Técnicos de la Información.

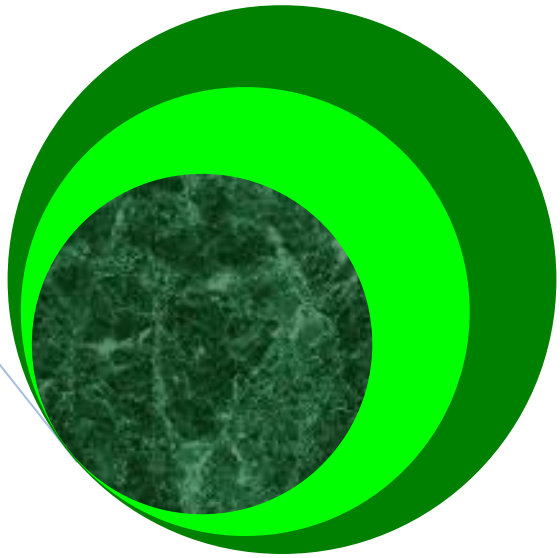
a) **Tabla ASPECTO_TECNICO.** El objetivo de analizar esta información se enfoca a conocer las variables que utilizan los catastros para generar sus datos.

Atributos:

- Elipsoide.
- Datum Horizontal.
- Datum Vertical.
- Proyección.
- Topografía.
- Numeración de Vértices.
- Coordenadas.
- Azimut.
- Distancias.
- Convergencias.
- Factor de Escala.
- Colidancias.

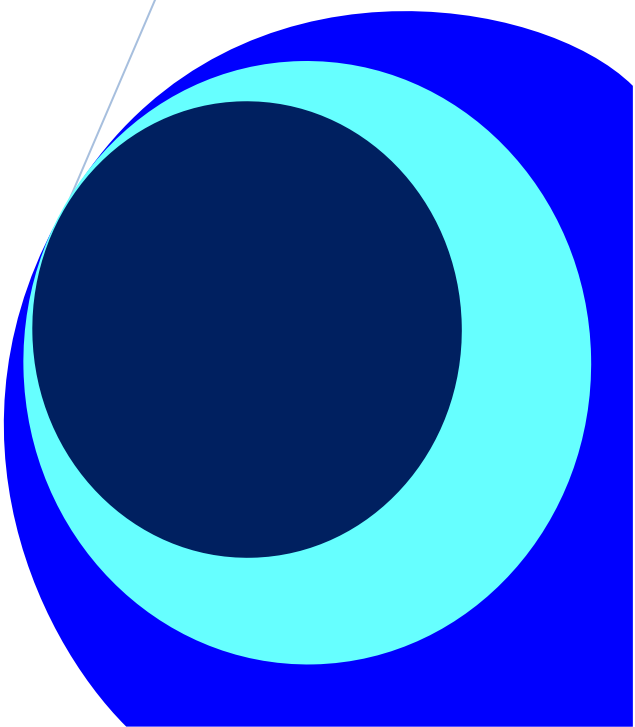
♦ **Software Empleado.**

a) Microsoft Windows XP SP2 Professional.
b) Microsoft Windows XP Service Pack 3.
c) Windows Installer 3.1.
d) Microsoft .NET Framework 3.5 Service Pack 1.
e) Microsoft SQL Server 2005 Express Edition with Advanced Services SP2.
f) Libros en Pantalla de Microsoft SQL Server 2005.



ANEXO III

Universidad.



PROYECTO 3. CREACIÓN DE UNA BASE DE DATOS PARA UNA UNIVERSIDAD.

- **Código 1. Conectarse con Microsoft SQL Server 2005.**

```
REM CONNECT USUARIO/CONTRASEÑA @BASE DE DATOS
CONNECT SYSTEM/aryekiel @DIPLO1 AS SYSDBA
```

- **Código 2. Crear la Base de Datos.**

```
BEGIN
  DBMS_SESSION.SET_NLS('NLS_NUMERIC_CHARACTERS', '".,"');
END;
/
CREATE TABLESPACE "TBSDIPLOMADO"
   LOGGING
   DATAFILE 'C:\ORACLE\ORADATA\DIPLO1\DIPLOMADO.DBF' SIZE 5M
   REUSE EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;

DROP ROLE SQLADMIN;
CREATE ROLE SQLADMIN NOT IDENTIFIED;

REM GARANTIZA PRIVILEGIOS A SQLADMIN.
GRANT SELECT ANY TABLE TO SQLADMIN WITH ADMIN OPTION;
GRANT CREATE TABLE TO SQLADMIN WITH ADMIN OPTION;
GRANT ANALYZE ANY TO SQLADMIN WITH ADMIN OPTION;

REM CREA USUARIO.
DROP USER DIPLOMADO;
CREATE USER DIPLOMADO  PROFILE DEFAULT
     IDENTIFIED BY "aryekiel"
DEFAULT TABLESPACE TBSDIPLOMADO
     ACCOUNT UNLOCK;

REM GARANTIZA PRIVILEGIOS A DIPLOMADO.
GRANT AQ_ADMINISTRATOR_ROLE TO DIPLOMADO;
GRANT CONNECT TO DIPLOMADO WITH ADMIN OPTION;
GRANT SQLADMIN TO DIPLOMADO WITH ADMIN OPTION;
GRANT RESOURCE TO DIPLOMADO WITH ADMIN OPTION;
```

- **Código 3. Crear las Tablas: ALUMNOS, PROFESOR y DEPARTAMENTO.**

```
CREATE TABLE DIPLOMADO.DEPARTAMENTO
(
  Id_depto char(3) PRIMARY KEY,
  Nombre VARCHAR2(30),
  Nro_profesores INTEGER
)TABLESPACE TBSDIPLOMADO;

CREATE TABLE DIPLOMADO.PROFESOR
(
  Nro_reg INTEGER PRIMARY KEY,
  Nombre VARCHAR2(10),
  Apellidos VARCHAR2(20),
  Departamento CHAR(3) REFERENCES DIPLOMADO.DEPARTAMENTO
)TABLESPACE TBSDIPLOMADO;

CREATE TABLE DIPLOMADO.ALUMNOS
(
  Nro_exp INTEGER PRIMARY KEY,
  Nombre VARCHAR2(10),
  Apellidos VARCHAR2(20),
  Curso INTEGER,
  Especialidad VARCHAR2(10),
  Tutor INTEGER REFERENCES DIPLOMADO.PROFESOR
)TABLESPACE TBSDIPLOMADO;

INSERT INTO DIPLOMADO.DEPARTAMENTO VALUES(1, 'tsi', 13);

INSERT INTO DIPLOMADO.PROFESOR VALUES(11, 'Maria','Martínez', 1);
INSERT INTO DIPLOMADO.PROFESOR VALUES(12, 'Pepe','Martínez', 1);

INSERT INTO DIPLOMADO.ALUMNOS VALUES (1, 'Luis', 'Fernández', 1, 'LSI', 11);
INSERT INTO DIPLOMADO.ALUMNOS VALUES (2, 'Maria', 'Ruiz', 1, 'LSI', 11);
INSERT INTO DIPLOMADO.ALUMNOS VALUES (3, 'Carmen', 'López', 1, 'LSI', 11);
INSERT INTO DIPLOMADO.ALUMNOS VALUES (4, 'Elena', 'Martin', 1, 'LSI', 12);
COMMIT;

SELECT * FROM DIPLOMADO.ALUMNOS;
```

- **Código 4. Crear Tablas: SALONES y MATERIAS.**

```
CREATE TABLE DIPLOMADO.MATERIAS
(
 ID_Materia Integer PRIMARY KEY,
 Nombre_Materia Varchar2(20),
 Antecedente_Materia Integer,
 Creditos_Materia Integer,
 Profesor_Materia Integer REFERENCES DIPLOMADO.PROFESOR
)TABLESPACE TBSDIPLOMADO;

CREATE TABLE DIPLOMADO.SALONES
(
 ID_Salon Integer PRIMARY KEY,
 Edificio_salon varchar2(20),
 Capacidad_salon Integer,
 Materia_salon Integer REFERENCES DIPLOMADO.MATERIAS
 )TABLESPACE TBSDIPLOMADO;

 INSERT INTO DIPLOMADO.MATERIAS VALUES (1,'Matemáticas I',0,12,11);
 INSERT INTO DIPLOMADO.MATERIAS VALUES (2,'Matemáticas II',1,12,11);
 INSERT INTO DIPLOMADO.MATERIAS VALUES (3,'Algebra II',2,12,12);
 INSERT INTO DIPLOMADO.ALUMNOS VALUES (5, 'José', 'Domínguez', 2, 'LST', 12);
 INSERT INTO DIPLOMADO.ALUMNOS VALUES (6, 'Benito', 'Hernández', 2, 'LST', 11);
 INSERT INTO DIPLOMADO.ALUMNOS VALUES (7, 'Juan', 'Pérez', 3, 'LST', 12);
 COMMIT;

CREATE OR REPLACE VIEW DIPLOMADO.vista1 as
SELECT DIPLOMADO.ALUMNOS.nombre, DIPLOMADO.ALUMNOS.apellidos,
DIPLOMADO.MATERIAS.Nombre_Materia
FROM DIPLOMADO.ALUMNOS, DIPLOMADO.PROFESOR, DIPLOMADO.MATERIAS
WHERE DIPLOMADO.ALUMNOS.tutor = DIPLOMADO.PROFESOR.nro_reg
  AND DIPLOMADO.ALUMNOS.curso = DIPLOMADO.MATERIAS.ID_Materia
  AND DIPLOMADO.PROFESOR.departamento = 1;

SELECT * FROM DIPLOMADO.vista1;

SELECT * FROM DIPLOMADO.vista1 WHERE TUTOR = 11;
```

- **Código 5. Crear Tabla CALIFICACIONES.**

```
CREATE TABLE DIPLOMADO.CALIFICACIONES
(
  Nro_exp INTEGER,
  ID_Materia INTEGER,
  Calificacion INTEGER,
  Promedio INTEGER,
  CONSTRAINT PK_Nro_Exp1 PRIMARY KEY (Nro_Exp, ID_Materia),
  CONSTRAINT FK_Nro_Exp1 FOREIGN KEY (ID_Materia) REFERENCES
DIPLOMADO.MATERIAS(ID_Materia)
);

INSERT INTO DIPLOMADO.CALIFICACIONES VALUES (1, 1, 9, 9);
INSERT INTO DIPLOMADO.CALIFICACIONES VALUES (1, 2, 8, 8);
INSERT INTO DIPLOMADO.CALIFICACIONES VALUES (1, 3, 7, 7);
INSERT INTO DIPLOMADO.CALIFICACIONES VALUES (2, 1, 7, 7);
INSERT INTO DIPLOMADO.CALIFICACIONES VALUES (2, 2, 8, 8);
INSERT INTO DIPLOMADO.CALIFICACIONES VALUES (2, 3, 9, 8);
INSERT INTO DIPLOMADO.CALIFICACIONES VALUES (3, 1, 6, 6);
INSERT INTO DIPLOMADO.CALIFICACIONES VALUES (3, 2, 6, 6);
INSERT INTO DIPLOMADO.CALIFICACIONES VALUES (3, 3, 7, 7);
INSERT INTO DIPLOMADO.CALIFICACIONES VALUES (4, 1, 9, 9);
INSERT INTO DIPLOMADO.CALIFICACIONES VALUES (4, 2, 8, 8);
INSERT INTO DIPLOMADO.CALIFICACIONES VALUES (4, 3, 7, 7);
INSERT INTO DIPLOMADO.CALIFICACIONES VALUES (5, 1, 8, 8);
INSERT INTO DIPLOMADO.CALIFICACIONES VALUES (5, 2, 9, 9);
INSERT INTO DIPLOMADO.CALIFICACIONES VALUES (5, 3, 9, 9);
COMMIT;

SELECT * FROM DIPLOMADO.CALIFICACIONES;
```

- **Código 6. Crear Tablas: GRUPOS, MATERIA GRUPO e INSCRIPCIONES_DETALLE.**

```
CREATE TABLE DIPLOMADO.SALONES
(
 ID_Salon Integer PRIMARY KEY,
 Edificio_salon varchar2(20),
 Capacidad_salon Integer,
 Materia_salon Integer REFERENCES DIPLOMADO.MATERIAS
)TABLESPACE TBSDIPLOMADO;

CREATE TABLE DIPLOMADO.GRUPOS
(
  ID_Grupo INTEGER,
  Descripcion VARCHAR2(20)
)TABLESPACE TBSDIPLOMADO;

CREATE TABLE DIPLOMADO.MATERIA_GRUPO
(
  ID_Materia_Grupo INTEGER,
  ID_Materia INTEGER,
  ID_Grupo INTEGER,
  ID_Salon INTEGER
)TABLESPACE TBSDIPLOMADO;

CREATE TABLE DIPLOMADO.INSCRIPCIONES_DETALLE
(
  ID_Nro_Inscripcion FLOAT,
  Nro_Exp INTEGER,
  Nro_Reg INTEGER,
  ID_Materia INTEGER,
  ID_Grupo INTEGER,
  ID_Salon INTEGER,
  Comentarios VARCHAR2(20)
)TABLESPACE TBSDIPLOMADO;

ALTER TABLE DIPLOMADO.GRUPOS
  ADD
  (
     CONSTRAINT PK_ID_Grupo PRIMARY KEY (ID_Grupo)
  );

ALTER TABLE DIPLOMADO.MATERIA_GRUPO
  ADD
  (
     CONSTRAINT PK_ID_Materia_Grupo PRIMARY KEY (ID_Materia_Grupo),
     CONSTRAINT FK_ID_Materia1 FOREIGN KEY (ID_Materia) REFERENCES
DIPLOMADO.MATERIAS(ID_Materia),
     CONSTRAINT FK_ID_Grupo1 FOREIGN KEY (ID_Grupo) REFERENCES
DIPLOMADO.GRUPOS(ID_Grupo),
     CONSTRAINT FK_ID_Salon1 FOREIGN KEY (ID_Salon) REFERENCES
DIPLOMADO.SALONES(ID_Salon)
  );
```

```
ALTER TABLE DIPLOMADO.INSCRIPCIONES_DETALLE
  ADD
  (
     CONSTRAINT PK_ID_Nro_Inscripcion PRIMARY KEY (ID_Nro_Inscripcion),
     CONSTRAINT FK_Nro_Exp2 FOREIGN KEY (Nro_Exp) REFERENCES
DIPLOMADO.ALUMNOS(Nro_Exp),
     CONSTRAINT FK_Nro_Reg2 FOREIGN KEY (Nro_Reg) REFERENCES
DIPLOMADO.PROFESOR(Nro_Reg),
     CONSTRAINT FK_ID_Materia2 FOREIGN KEY (ID_Materia) REFERENCES
DIPLOMADO.MATERIAS(ID_Materia),
     CONSTRAINT FK_ID_Grupo2 FOREIGN KEY (ID_Grupo) REFERENCES
DIPLOMADO.GRUPOS(ID_Grupo),
     CONSTRAINT FK_ID_Salon2 FOREIGN KEY (ID_Salon) REFERENCES
DIPLOMADO.SALONES(ID_Salon)
  );

INSERT INTO DIPLOMADO.SALONES VALUES (10001, 'EDIF1', 50, 1);
INSERT INTO DIPLOMADO.SALONES VALUES (9001, 'EDIF1', 50, 1);
INSERT INTO DIPLOMADO.SALONES VALUES (3002, 'EDIF1', 50, 2);
INSERT INTO DIPLOMADO.SALONES VALUES (3003, 'EDIF1', 50, 2);
INSERT INTO DIPLOMADO.SALONES VALUES (9002, 'EDIF1', 50, 3);
INSERT INTO DIPLOMADO.SALONES VALUES (9003, 'EDIF1', 50, 3);
COMMIT;

INSERT INTO DIPLOMADO.GRUPOS VALUES (1051, 'GRUPO 1051');
INSERT INTO DIPLOMADO.GRUPOS VALUES (1052, 'GRUPO 1052');
INSERT INTO DIPLOMADO.GRUPOS VALUES (1053, 'GRUPO 1053');
INSERT INTO DIPLOMADO.GRUPOS VALUES (2051, 'GRUPO 2051');
INSERT INTO DIPLOMADO.GRUPOS VALUES (2052, 'GRUPO 2052');
INSERT INTO DIPLOMADO.GRUPOS VALUES (2053, 'GRUPO 2053');
INSERT INTO DIPLOMADO.GRUPOS VALUES (3051, 'GRUPO 3051');
INSERT INTO DIPLOMADO.GRUPOS VALUES (3052, 'GRUPO 3052');
COMMIT;

INSERT INTO DIPLOMADO.MATERIA_GRUPO VALUES (1, 1, 1051, 10001);
INSERT INTO DIPLOMADO.MATERIA_GRUPO VALUES (2, 1, 1052, 9001);
INSERT INTO DIPLOMADO.MATERIA_GRUPO VALUES (3, 2, 2051, 3002);
INSERT INTO DIPLOMADO.MATERIA_GRUPO VALUES (4, 2, 2052, 3003);
INSERT INTO DIPLOMADO.MATERIA_GRUPO VALUES (5, 2, 2053, 3003);
INSERT INTO DIPLOMADO.MATERIA_GRUPO VALUES (6, 3, 3051, 9002);
INSERT INTO DIPLOMADO.MATERIA_GRUPO VALUES (7, 3, 3052, 9003);
COMMIT;

INSERT INTO DIPLOMADO.INSCRIPCIONES_DETALLE VALUES (20061104001, 1, 11, 1, 1051,
10001, 'SIN COMENTARIOS');
INSERT INTO DIPLOMADO.INSCRIPCIONES_DETALLE VALUES (20061104002, 1, 11, 2, 2051, 3002,
'SIN COMENTARIOS');
INSERT INTO DIPLOMADO.INSCRIPCIONES_DETALLE VALUES (20061104003, 1, 12, 3, 3051, 9002,
'SIN COMENTARIOS');
INSERT INTO DIPLOMADO.INSCRIPCIONES_DETALLE VALUES (20061104004, 2, 11, 1, 1051,
10001, 'SIN COMENTARIOS');
INSERT INTO DIPLOMADO.INSCRIPCIONES_DETALLE VALUES (20061104005, 2, 11, 2, 2052, 3003,
'SIN COMENTARIOS');
INSERT INTO DIPLOMADO.INSCRIPCIONES_DETALLE VALUES (20061104006, 2, 12, 3, 3051, 9002,
'SIN COMENTARIOS');
```

```
INSERT INTO DIPLOMADO.INSCRIPCIONES_DETALLE VALUES (20061104007, 3, 11, 1, 1051,
10001, 'SIN COMENTARIOS');
INSERT INTO DIPLOMADO.INSCRIPCIONES_DETALLE VALUES (20061104010, 3, 11, 2, 2052, 3003,
'SIN COMENTARIOS');
INSERT INTO DIPLOMADO.INSCRIPCIONES_DETALLE VALUES (20061104011, 3, 12, 3, 3051, 9002,
'SIN COMENTARIOS');
INSERT INTO DIPLOMADO.INSCRIPCIONES_DETALLE VALUES (20061104012, 4, 11, 1, 1051,
10001, 'SIN COMENTARIOS');
INSERT INTO DIPLOMADO.INSCRIPCIONES_DETALLE VALUES (20061104013, 4, 11, 2, 2052, 3003,
'SIN COMENTARIOS');
INSERT INTO DIPLOMADO.INSCRIPCIONES_DETALLE VALUES (20061104014, 4, 12, 3, 3051, 9002,
'SIN COMENTARIOS');
INSERT INTO DIPLOMADO.INSCRIPCIONES_DETALLE VALUES (20061104015, 5, 11, 1, 1051,
10001, 'SIN COMENTARIOS');
INSERT INTO DIPLOMADO.INSCRIPCIONES_DETALLE VALUES (20061104016, 5, 11, 2, 2052, 3003,
'SIN COMENTARIOS');
INSERT INTO DIPLOMADO.INSCRIPCIONES_DETALLE VALUES (20061104017, 5, 12, 3, 3051, 9002,
'SIN COMENTARIOS');
INSERT INTO DIPLOMADO.INSCRIPCIONES_DETALLE VALUES (20061104018, 6, 11, 1, 1051,
10001, 'SIN COMENTARIOS');
INSERT INTO DIPLOMADO.INSCRIPCIONES_DETALLE VALUES (20061104019, 6, 11, 2, 2053, 3003,
'SIN COMENTARIOS');
INSERT INTO DIPLOMADO.INSCRIPCIONES_DETALLE VALUES (20061104020, 6, 12, 3, 3051, 9002,
'SIN COMENTARIOS');
INSERT INTO DIPLOMADO.INSCRIPCIONES_DETALLE VALUES (20061104021, 7, 11, 1, 1051,
10001, 'SIN COMENTARIOS');
INSERT INTO DIPLOMADO.INSCRIPCIONES_DETALLE VALUES (20061104022, 7, 11, 2, 2053, 3003,
'SIN COMENTARIOS');
INSERT INTO DIPLOMADO.INSCRIPCIONES_DETALLE VALUES (20061104023, 7, 12, 3, 3052, 9003,
'SIN COMENTARIOS');
COMMIT;
```

- **Código 7. Crear tabla BITACORA_INSCRIPCIONES.**

```
CREATE TABLE DIPLOMADO.BITACORA_INSCRIPCIONES
(
  ID_Nro_Inscripcion NUMBER(15),
  Nro_Exp INTEGER,
  Nro_Reg INTEGER,
  ID_Materia INTEGER,
  ID_Grupo INTEGER,
  ID_Salon INTEGER,
  Fecha DATE,
  Comentarios VARCHAR2(20)
)TABLESPACE TBSDIPLOMADO;

CREATE OR REPLACE TRIGGER DIPLOMADO.Bitacora_Modificaciones
  BEFORE
  DELETE OR INSERT OR UPDATE
  ON DIPLOMADO.INSCRIPCIONES_DETALLE
  FOR EACH ROW
  BEGIN
    IF INSERTING THEN  --INSERCION
      INSERT INTO DIPLOMADO.BITACORA_INSCRIPCIONES
      (
        DIPLOMADO.BITACORA_INSCRIPCIONES.ID_Nro_Inscripcion,
        DIPLOMADO.BITACORA_INSCRIPCIONES.Nro_Exp,
        DIPLOMADO.BITACORA_INSCRIPCIONES.Nro_Reg,
        DIPLOMADO.BITACORA_INSCRIPCIONES.ID_Materia,
        DIPLOMADO.BITACORA_INSCRIPCIONES.ID_Grupo,
        DIPLOMADO.BITACORA_INSCRIPCIONES.ID_Salon,
        DIPLOMADO.BITACORA_INSCRIPCIONES.Fecha,
        DIPLOMADO.BITACORA_INSCRIPCIONES.Comentarios
      )
      VALUES
      (
        :NEW.ID_Nro_Inscripcion,
        :NEW.Nro_Exp,
        :NEW.Nro_Reg,
        :NEW.ID_Materia,
        :NEW.ID_Grupo,
        :NEW.ID_Salon,
        SYSDATE,
        'INSERCION'
      );
    ELSIF DELETING THEN--ELIMINACION
      INSERT INTO DIPLOMADO.BITACORA_INSCRIPCIONES
      (
        DIPLOMADO.BITACORA_INSCRIPCIONES.ID_Nro_Inscripcion,
        DIPLOMADO.BITACORA_INSCRIPCIONES.Nro_Exp,
        DIPLOMADO.BITACORA_INSCRIPCIONES.Nro_Reg,
        DIPLOMADO.BITACORA_INSCRIPCIONES.ID_Materia,
        DIPLOMADO.BITACORA_INSCRIPCIONES.ID_Grupo,
        DIPLOMADO.BITACORA_INSCRIPCIONES.ID_Salon,
        DIPLOMADO.BITACORA_INSCRIPCIONES.Fecha,
        DIPLOMADO.BITACORA_INSCRIPCIONES.Comentarios
      )
```

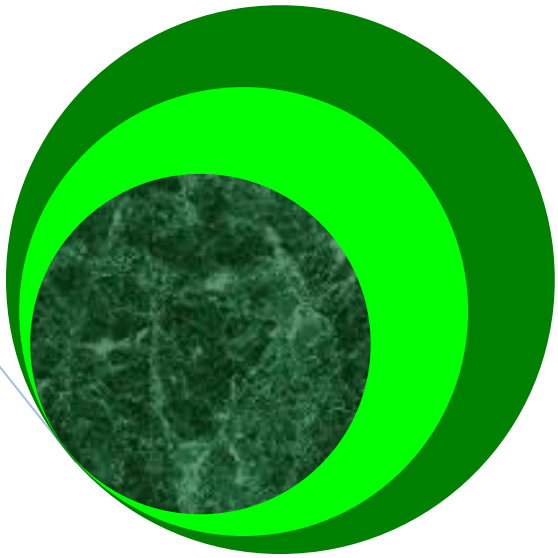
```
      VALUES
      (
         :OLD.ID_Nro_Inscripcion,
         :OLD.Nro_Exp,
         :OLD.Nro_Reg,
         :OLD.ID_Materia,
         :OLD.ID_Grupo,
         :OLD.ID_Salon,
         SYSDATE,
         USER
      );
   ELSE            --ACTUALIZACION
      INSERT INTO DIPLOMADO.BITACORA_INSCRIPCIONES
      (
         DIPLOMADO.BITACORA_INSCRIPCIONES.ID_Nro_Inscripcion,
         DIPLOMADO.BITACORA_INSCRIPCIONES.Nro_Exp,
         DIPLOMADO.BITACORA_INSCRIPCIONES.Nro_Reg,
         DIPLOMADO.BITACORA_INSCRIPCIONES.ID_Materia,
         DIPLOMADO.BITACORA_INSCRIPCIONES.ID_Grupo,
         DIPLOMADO.BITACORA_INSCRIPCIONES.ID_Salon,
         DIPLOMADO.BITACORA_INSCRIPCIONES.Fecha,
         DIPLOMADO.BITACORA_INSCRIPCIONES.Comentarios
      )
      VALUES
      (
         :OLD.ID_Nro_Inscripcion,
         :OLD.Nro_Exp,
         :OLD.Nro_Reg,
         :OLD.ID_Materia,
         :OLD.ID_Grupo,
         :OLD.ID_Salon,
         SYSDATE,
         USER
      );
   END IF;
END;
```

- **Código 8. Creación de una Función (Obtener de Promedio).**

```
CREATE OR REPLACE FUNCTION DIPLOMADO.Suma_Alumnos_Promedio
(
   Entrada_Nro_Exp IN INTEGER
)
RETURN NUMBER
IS
   Nuevo_Promedio INTEGER;
BEGIN
   UPDATE DIPLOMADO.ALUMNOS
   SET DIPLOMADO.ALUMNOS.Promedio = DIPLOMADO.ALUMNOS.Promedio + 0.6
   WHERE DIPLOMADO.ALUMNOS.Nro_exp = Entrada_Nro_exp;

   SELECT DIPLOMADO.ALUMNOS.Promedio
   INTO Nuevo_Promedio
   FROM DIPLOMADO.ALUMNOS
   WHERE DIPLOMADO.ALUMNOS.Nro_exp = Entrada_Nro_exp;
   RETURN(Nuevo_Promedio);
END;
/
```

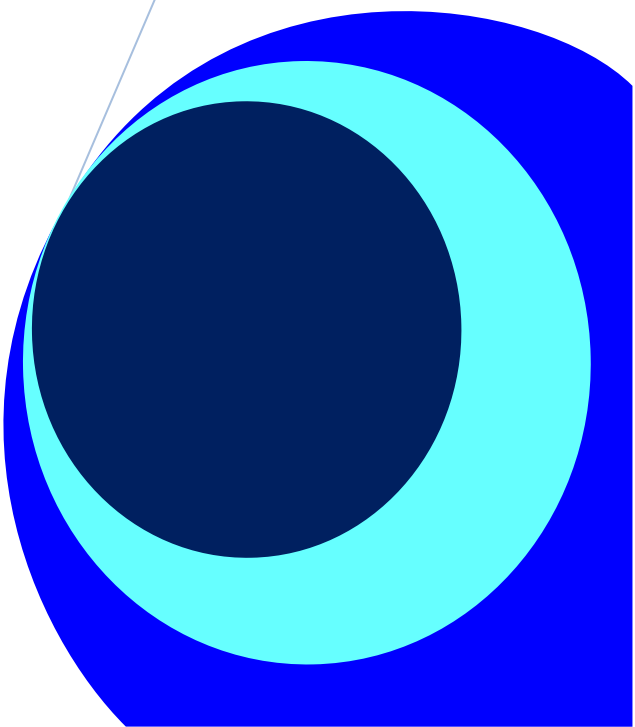
- **Código 9. Creación de un Procedimiento.**

```
CREATE OR REPLACE PROCEDURE DIPLOMADO.Alumnos_Promedio_Todos
(
   Entrada_Nro_Reg_Profesor IN INTEGER,
   Cuantos OUT INTEGER
)
AS
BEGIN
   SELECT COUNT(*)
   INTO Cuantos
   FROM DIPLOMADO.ALUMNOS
   WHERE DIPLOMADO.ALUMNOS.Tutor = Entrada_Nro_Reg_Profesor;

   DBMS_OUTPUT.PUT_LINE('EL NUMERO DE REGISTROS ES: '|| Cuantos);

   EXCEPTION
      WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('ERROR EN PROCEDIMIENTO
DIPLOMADO.Alumnos_Promedio_Todos'||SQLCODE||SQLERRM);
END;
/
```



ANEXO IV

Tienda de Autoservicio.



TIENDA DE AUTOSERVICIO.

- **Presentación.**

Una Tienda de Autoservicio es un canal de distribución minorista directa al consumidor, constituida por departamentos individuales con amplio surtido; donde el vendedor como persona física ha desaparecido totalmente de la transacción comercial y el producto se encuentra frente al cliente, quién lo coloca directamente en el carrito o cesta de comprar. Por su parte la propia clientela debe aprender a orientarse dentro de la tienda de autoservicio, leer la oferta, seleccionar el producto, decidir su compra y encaminarse al lugar de pago.

- **Características de una Tienda de Autoservicio:**

1. Organización individual en Departamentos.
2. Amplio surtido de Mercancías.
3. Cobro de la mercancía en cajas centrales.
4. Cambio y devolución de mercancías.
5. Personal mínimo (aproximadamente 30 Individuos).

- **Áreas de una Tienda de Autoservicio:**

1. **Recibo.** Lugar donde los proveedores entregan la mercancía que la tienda le ha pedido, de la misma manera salen las mercancías que no se vendieron, que se descontinuaron o que se mandaron a devolución.

2. **Bodegas.** Lugar donde se depositan o almacenan las mercancías que llegan de Recibo.

a) **Bodega de Abarrotes.** Resguarda productos alimenticios no perecederos, artículos para mascotas, artículos de limpieza (papel higiénico, papel de cocina, jabones, detergentes, etc.), artículos de desechables (vasos, platos, servilletas, bolsas para basura), también carbón, velas, veladoras de cera o cebo, refrescos, jugos, vinos, licores, etc.

b) **Bodega de Líneas Generales.** Resguarda artículos que son clasificados como no básicos y se dividen en los siguientes departamentos:

- Artículos de Electrónica (Televisores, minicomponentes, celulares, computadoras, etc.).
- Artículos de Electrodomésticos o Línea Blanca (Refrigeradores, licuadoras, lavadoras, etc.).
- Muebles (Sillas, camas, burós, muebles para computadoras, closets, etc.).
- Blancos (Artículos para baño: jaboneras, toallas, artículos de plomería, etc.).
- Hogar (Artículos para uso doméstico: cristalería, vajillas, ollas, botes de basura, etc.).
- Discos y Películas.
- Revistas y Libros.
- Ferretería (Artículos para autos y arreglo del hogar.).
- Perfumería (Artículos de aseo personal: perfumes, cosméticos, desodorantes, etc.).
- Farmacia.
- Papelería (Artículos de oficina y escuela: cuadernos, plumas, calculadoras, agendas, etc.).
- Jardinería.
- Juguetería.
- Regalos.
- Ropa de Dama, Caballero y Niños.
- Artículos para bebés.
- Zapatería.

c) **Perecederos.** Es toda aquella mercancía que tiene una caducidad, que debe tener la debida rotación antes de que se pudra o se convierta en merma. Sus departamentos son:

- Frutas y verduras.
- Pescados y mariscos.
- Carnes.
- Embutidos.
- Panadería.
- Tortillería.
- Lácteos.

3. **Piso de Venta.** Lugar donde se negocia, se trata o se produce la venta de las mercancías entre la tienda y su clientela.

4. **Cocina.** Lugar donde se preparan los alimentos que los clientes consumirán ya sea dentro de la tienda o fuera de ella.

5. **Cafetín (Fuente de Sodas).** Lugar donde los clientes pueden detenerse a tomar un refrigerio.

6. **Área de Cajas.** Lugar donde los clientes que finalmente han escogido su mercancía, puedan pagarla.

7. **El Checado.** Lugar donde la clientela deposita el producto que ya no quiere comprar.

8. **Atención al Cliente.** Lugar donde se entregan las garantías de los artículos adquiridos, atención a cualquier tipo de pregunta, reclamo o sugerencia por parte del cliente, la realización del cambio físico de una mercancía por otra o la devolución de su dinero, etc.

9. **Paquetería.** Área responsable de recoger los paquetes o pertenencias de los clientes, que son ajenas a la mercancía existente del establecimiento.

10. **Gerencia.** Es el departamento líder de la tienda, conformado por un gerente y dos o tres subgerentes.

11. **Recursos Humanos.** Área encargada de contratar el personal de la tienda.

12. **Mesa de Control de Inventarios.** Se usa cuando la tienda necesita hacer un inventario o una auditoría, convocando a varios empleados como los: auxiliares de piso de ventas, cajeros, personal de seguridad, de perecederos y de la misma mesa de control, citados por los gerentes para realizar los conteos correspondientes de la mercancía módulo por módulo y anaquel por anaquel, para obtener un reporte de los excedentes o faltantes que existen en la tienda.

13. **Cacheo o Revisión.** Lugar donde el personal de seguridad se encarga de evitar que los empleados, los promotores y las demostradoras roben mercancía de la tienda, por medio de una revisión corporal.

14. **Seguridad.** Área responsable de controlar las cámaras de vigilancia.

15. **Display.** Área responsable de rotular, crear los cartelones y anuncios de los precios de las mercancías y las promociones actuales de la tienda.

16. **Consumos Internos.** Área responsable de proporcionar a los distintos departamentos las herramientas de trabajo como son: escobas, franelas, plumones o marcadores, cinta canela, navajas, etc. Su importancia radica en llevar un control de las mercancías que los empleados utilizan, o de lo contrario, solo estarían generando bajas o pérdidas que afectan la economía de la tienda.

PROYECTO 4. CREACIÓN DE UNA BASE DE DATOS PARA EL CONTROL DE INVENTARIOS EN UNA TIENDA DE AUTOSERVICIO.

- **Descripción del Proyecto.**

Crear una Base de Datos donde se identifiquen y clasifiquen todos los artículos de una Tienda de Autoservicio de acuerdo al departamento o línea de estos, al igual facilitar la continuidad del proceso productivo y la satisfacción de los pedidos de los consumidores y clientes.

- **Objetivos:**

1. Asegurar que se mantengan los registros adecuados de los artículos o productos existentes y de sus cantidades.

2. Facilitar la continuidad del proceso productivo y la satisfacción de los pedidos de los consumidores y clientes.

- **Funciones:**

1. Mantener un registro actualizado de las existencias. La periodicidad depende de unas empresas a otras y del tipo de producto. Existe el sistema de inventario continuo (diario e informatizado) y el periódico que coloquialmente se llama "hacer inventario".

2. Informar del nivel de existencias, para saber cuándo se debe hacer un pedido y cuánto se debe pedir de cada uno de los productos.

3. Notificar situaciones anormales, que pueden constituir síntomas de errores o de un mal funcionamiento del sistema.

4. Elaborar informes para la dirección y para los responsables de los inventarios.

- **Identificación de Necesidades:**

A. Se mantendrán **Grandes Niveles** de Inventarios cuando:

a) Los costes de realización de pedidos son elevados.
b) Los costes de almacenamiento son bajos.
c) Realizando grandes pedidos es posible obtener grandes descuentos.
d) Crecimiento considerable de la demanda.
e) Precios elevados de productos.

B. Se mantendrán **Bajos Niveles** de Inventarios cuando:

a) Los costes de almacenamiento son elevados y los de pedido bajos.
b) La demanda de la empresa es estable.
c) Los proveedores son de confianza y no hay problemas de reaprovisionamiento.
d) No es posible aplazar el pago a los proveedores y existen problemas de financiación.
e) Se esperan importantes disminuciones de los precios.

- **Generación de Informes en los siguientes rubros:**

1. **Informes de Compras:**

   a) Por Proveedor.
   b) Fecha de Emisión.
   c) Libro de Compras
   d) Facturas de Compra Canceladas y por Pagar.

2. **Informe de Ventas:**

   a) Por Cliente.
   b) Por Fecha de Emisión.
   c) Libro de Ventas
   d) Facturas de Venta Canceladas y por Pagar.

3. **Informes de Productos.**

   a) Por Período.
   b) Por Familia
   c) Por Producto
   d) Según Compras y Ventas.
   e) Catálogo de Productos.
   f) Lista de Precios.

- **Software Empleado.**

a) Microsoft Windows XP SP2 Professional.
b) Microsoft Windows XP Service Pack 3.
c) Windows Installer 3.1.
d) Microsoft .NET Framework 3.5 Service Pack 1.
e) Oracle9i Release 2 (9.2.0.1.0) for Windows.

- **Códigos para Crear las Tablas de la Base de Datos.**

  - **Código 1. Tabla COUNTRIES (PAÍSES).**

```
DECLARE
    l_lineas DBMS_SQL.varchar2s;
    nCuantos NUMBER;

    COUNTRY_ID countries.country_id%TYPE;
    COUNTRY_NAME countries.country_name%TYPE;
    REGION_ID countries.region_id%TYPE;

    CURSOR cdModulo5_14
    IS
       SELECT country_id,
            country_name,
            region_id
       FROM HR.COUNTRIES;
BEGIN
    l_lineas(1) := 'TAREA HR.COUNTRIES';


    SELECT COUNT(*) INTO nCuantos FROM HR.COUNTRIES;

    OPEN cdModulo5_14;
    FOR i IN 1..nCuantos LOOP
        FETCH cdModulo5_14 INTO
            COUNTRY_ID,
            COUNTRY_NAME,
            REGION_ID;
        l_lineas(i+1) := COUNTRY_ID   || ', ' ||
                    COUNTRY_NAME || ', ' ||
                    REGION_ID;
        EXIT WHEN cdModulo5_14%NOTFOUND;
    END LOOP;
    HR.respalda_tabla('TEMP','COUNTRIES.txt',l_lineas);
    DBMS_OUTPUT.PUT_LINE
    (
         'SE RESPALDO EXITOSAMENTE LA TABLA HR.COUNTRIES'
    );
    CLOSE cdModulo5_14;
    COMMIT;
END;
```

- **Código 2. Tabla DEPARMENTS (DEPARTAMENTOS).**

```
DECLARE
    l_lineas DBMS_SQL.varchar2s;
    nCuantos NUMBER;

    DEPARTMENT_ID departments.department_id%TYPE;
    DEPARTMENT_NAME departments.department_name%TYPE;
    MANADER_ID departments.manager_id%TYPE;
    LOCATION_ID departments.location_id%TYPE;

    CURSOR cdModulo5_14
    IS
       SELECT department_id,
            department_name,
            manager_id,
            location_id
       FROM HR.DEPARTMENTS;
BEGIN
    l_lineas(1) := 'TAREA HR.DEPARTMENTS';


    SELECT COUNT(*) INTO nCuantos FROM HR.DEPARTMENTS;

    OPEN cdModulo5_14;
    FOR i IN 1..nCuantos LOOP
         FETCH cdModulo5_14 INTO
            DEPARTMENT_ID,
            DEPARTMENT_NAME,
            MANADER_ID,
            LOCATION_ID;
         l_lineas(i+1) := DEPARTMENT_ID   || ', ' ||
                     DEPARTMENT_NAME || ', ' ||
                     MANADER_ID      || ', ' ||
                     LOCATION_ID;
         EXIT WHEN cdModulo5_14%NOTFOUND;
    END LOOP;
    HR.respalda_tabla('TEMP','DEPARTMENTS.txt',l_lineas);
    DBMS_OUTPUT.PUT_LINE
    (
          'SE RESPALDO EXITOSAMENTE LA TABLA HR.DEPARTMENTS'
    );
    CLOSE cdModulo5_14;
    COMMIT;
END;
```

- **Código 3. DESC_X_VIGENCIA (DESCUENTO POR VIGENCIA).**

```
DECLARE
    l_lineas DBMS_SQL.varchar2s;
    nCuantos NUMBER;

    COD_PRODUCTO desc_x_vigencia.cod_producto%TYPE;
    VIGENCIA_DESDE desc_x_vigencia.vigencia_desde%TYPE;
    VIGENCIA_HASTA desc_x_vigencia.vigencia_hasta%TYPE;

    CURSOR cdModulo5_14
    IS
       SELECT cod_producto,
            vigencia_desde,
            vigencia_hasta
       FROM HR.DESC_X_VIGENCIA;
BEGIN
    l_lineas(1) := 'TAREA HR.DESC_X_VIGENCIA';


    SELECT COUNT(*) INTO nCuantos FROM HR.DESC_X_VIGENCIA;

    OPEN cdModulo5_14;
    FOR i IN 1..nCuantos LOOP
        FETCH cdModulo5_14 INTO
           COD_PRODUCTO,
           VIGENCIA_DESDE,
           VIGENCIA_HASTA;
        l_lineas(i+1) := COD_PRODUCTO    || ', ' ||
                    VIGENCIA_DESDE  || ', ' ||
                    VIGENCIA_HASTA;
        EXIT WHEN cdModulo5_14%NOTFOUND;
    END LOOP;
    HR.respalda_tabla('TEMP','DESC_X_VIGENCIA.txt',l_lineas);
    DBMS_OUTPUT.PUT_LINE
    (
         'SE RESPALDO EXITOSAMENTE LA TABLA HR.DESC_X_VIGENCIA'
    );
    CLOSE cdModulo5_14;
    COMMIT;
END;
```

- **Código 4. Tabla EMPLOYEES (EMPLEADOS).**

```
DECLARE
    l_lineas DBMS_SQL.varchar2s;
    nCuantos NUMBER;

    EMPL_ID employees.employee_id%TYPE;
    FIRST_N employees.first_name%TYPE;
    LAST__N employees.last_name%TYPE;
    EMAIL   employees.email%TYPE;
    PHONE_NUMBER employees.phone_number%TYPE;
    HIRE_DATE employees.hire_date%TYPE;
    JOB_ID  employees.job_ID%TYPE;
    SALARY  employees.salary%TYPE;
    COMMISSION_PCT employees.commission_pct%TYPE;
    MANAGER_ID employees.manager_id%TYPE;
    DEPT_ID employees.department_id%TYPE;

    CURSOR cdModulo5_14
    IS
       SELECT employee_id,
         first_name,
         last_name,
         email,
         phone_number,
         hire_date,
         job_ID,
         salary,
         commission_pct,
         manager_id,
         department_id
       FROM HR.EMPLOYEES;
BEGIN
    l_lineas(1) := 'TAREA HR.EMPLOYEES';


    SELECT COUNT(*) INTO nCuantos FROM HR.EMPLOYEES;

    OPEN cdModulo5_14;
    FOR i IN 1..nCuantos LOOP
          FETCH cdModulo5_14 INTO
             EMPL_ID,
             FIRST_N,
             LAST__n,
             EMAIL,
             PHONE_NUMBER,
             HIRE_DATE,
             JOB_ID,
             SALARY,
             COMMISSION_PCT,
             MANAGER_ID,
             DEPT_ID;
          l_lineas(i+1) := EMPL_ID || ', ' ||
                           FIRST_N || ', ' ||
                           LAST__N || ', ' ||
```

```
                    EMAIL   || ', ' ||
                    PHONE_NUMBER   || ', ' ||
                    HIRE_DATE    || ', ' ||
                    JOB_ID  || ', ' ||
                    SALARY  || ', ' ||
                    COMMISSION_PCT || ', ' ||
                    MANAGER_ID || ', ' ||
                    DEPT_ID;
            EXIT WHEN cdModulo5_14%NOTFOUND;
    END LOOP;
    HR.respalda_tabla('TEMP','EMPLOYEES.txt',l_lineas);
    DBMS_OUTPUT.PUT_LINE
    (
          'SE RESPALDO EXITOSAMENTE LA TABLA HR.EMPLOYEES'
    );
    CLOSE cdModulo5_14;
    COMMIT;
END;
```

- **Código 5. Tabla FAMILY (FAMILIAS).**

```
DECLARE
    l_lineas DBMS_SQL.varchar2s;
    nCuantos NUMBER;

    COD_FAMILIA familias.cod_familia%TYPE;
    DESCRIPCION familias.descripcion%TYPE;

    CURSOR cdModulo5_14
    IS
       SELECT cod_familia,
            descripcion
       FROM HR.FAMILIAS;
BEGIN
    l_lineas(1) := 'TAREA HR.FAMILIAS';


    SELECT COUNT(*) INTO nCuantos FROM HR.FAMILIAS;

    OPEN cdModulo5_14;
    FOR i IN 1..nCuantos LOOP
        FETCH cdModulo5_14 INTO
            COD_FAMILIA,
            DESCRIPCION;
        l_lineas(i+1) := COD_FAMILIA   || ', ' ||
                    DESCRIPCION;
        EXIT WHEN cdModulo5_14%NOTFOUND;
    END LOOP;
    HR.respalda_tabla('TEMP','FAMILIAS.txt',l_lineas);
    DBMS_OUTPUT.PUT_LINE
    (
         'SE RESPALDO EXITOSAMENTE LA TABLA HR.FAMILIAS'
    );
    CLOSE cdModulo5_14;
    COMMIT;
END;
```

- **Código 6. Tabla JOB_HISTORY (HISTORIAL LABORAL).**

```
DECLARE
    l_lineas DBMS_SQL.varchar2s;
    nCuantos NUMBER;

    EMPLOYEE_ID job_history.employee_id%TYPE;
    START_DATE job_history.start_date%TYPE;
    END_DATE job_history.end_date%TYPE;
    JOB_ID   job_history.job_id%TYPE;
    DEPARTMENT_ID job_history.department_id%TYPE;

    CURSOR cdModulo5_14
    IS
       SELECT employee_id,
            start_date,
            end_date,
            job_id,
            department_id
       FROM HR.JOB_HISTORY;
BEGIN
    l_lineas(1) := 'TAREA HR.JOB_HISTORY';


    SELECT COUNT(*) INTO nCuantos FROM HR.JOB_HISTORY;

    OPEN cdModulo5_14;
    FOR i IN 1..nCuantos LOOP
        FETCH cdModulo5_14 INTO
            EMPLOYEE_ID,
            START_DATE,
            END_DATE,
            JOB_ID,
            DEPARTMENT_ID;
        l_lineas(i+1) := EMPLOYEE_ID   || ', ' ||
                    START_DATE    || ', ' ||
                    END_DATE      || ', ' ||
                    JOB_ID        || ', ' ||
                    DEPARTMENT_ID;
        EXIT WHEN cdModulo5_14%NOTFOUND;
    END LOOP;
    HR.respalda_tabla('TEMP','JOB_HISTORY.txt',l_lineas);
    DBMS_OUTPUT.PUT_LINE
    (
         'SE RESPALDO EXITOSAMENTE LA TABLA HR.JOB_HISTORY'
    );
    CLOSE cdModulo5_14;
    COMMIT;
END;
```

- **Código 7. Tabla JOBS (EMPLEOS).**

```
DECLARE
    l_lineas DBMS_SQL.varchar2s;
    nCuantos NUMBER;

    JOB_ID jobs.job_id%TYPE;
    JOB_TITLE jobs.job_title%TYPE;
    MIN_SALARY jobs.min_salary%TYPE;
    MAX_SALARY jobs.max_salary%TYPE;

    CURSOR cdModulo5_14
    IS
       SELECT job_id,
            job_title,
            min_salary,
            max_salary
       FROM HR.JOBS;
BEGIN
    l_lineas(1) := 'TAREA HR.JOBS';


    SELECT COUNT(*) INTO nCuantos FROM HR.JOBS;

    OPEN cdModulo5_14;
    FOR i IN 1..nCuantos LOOP
         FETCH cdModulo5_14 INTO
            JOB_ID,
            JOB_TITLE,
            MIN_SALARY,
            MAX_SALARY;
         l_lineas(i+1) := JOB_ID     || ', ' ||
                     JOB_TITLE  || ', ' ||
                     MIN_SALARY || ', ' ||
                     MAX_SALARY;
         EXIT WHEN cdModulo5_14%NOTFOUND;
    END LOOP;
    HR.respalda_tabla('TEMP','JOBS.txt',l_lineas);
    DBMS_OUTPUT.PUT_LINE
    (
          'SE RESPALDO EXITOSAMENTE LA TABLA HR.JOBS'
    );
    CLOSE cdModulo5_14;
    COMMIT;
END;
```

- **Código 8. Tabla LOCATIONS (LOCALIDADES).**

```
DECLARE
    l_lineas DBMS_SQL.varchar2s;
    nCuantos NUMBER;

    LOCATION_ID locations.location_id%TYPE;
    STREET_ADDRESS locations.street_address%TYPE;
    POSTAL_CODE locations.postal_code%TYPE;
    CITY locations.city%TYPE;
    STATE_PROVINCE locations.state_province%TYPE;

    CURSOR cdModulo5_14
    IS
       SELECT location_id,
            street_address,
            postal_code,
            city,
            state_province
       FROM HR.LOCATIONS;
BEGIN
    l_lineas(1) := 'TAREA HR.LOCATIONS';


    SELECT COUNT(*) INTO nCuantos FROM HR.LOCATIONS;

    OPEN cdModulo5_14;
    FOR i IN 1..nCuantos LOOP
        FETCH cdModulo5_14 INTO
            LOCATION_ID,
            STREET_ADDRESS,
            POSTAL_CODE,
            CITY,
            STATE_PROVINCE;
        l_lineas(i+1) := LOCATION_ID    || ', ' ||
                    STREET_ADDRESS  || ', ' ||
                    POSTAL_CODE     || ', ' ||
                    CITY            || ', ' ||
                    STATE_PROVINCE;
        EXIT WHEN cdModulo5_14%NOTFOUND;
    END LOOP;
    HR.respalda_tabla('TEMP','LOCATIONS.txt',l_lineas);
    DBMS_OUTPUT.PUT_LINE
    (
         'SE RESPALDO EXITOSAMENTE LA TABLA HR.LOCATIONS'
    );
    CLOSE cdModulo5_14;
    COMMIT;
END;
```

- **Código 9. Tabla PRECIOS.**

```
DECLARE
    l_lineas DBMS_SQL.varchar2s;
    nCuantos NUMBER;

    PRECIO precios.precio%TYPE;
    COD_PRODUCTO precios.cod_producto%TYPE;
    VIGENCIA_DESDE precios.vigencia_desde%TYPE;
    VIGENCIA_HASTA precios.vigencia_hasta%TYPE;

    CURSOR cdModulo5_14
    IS
       SELECT precio,
            cod_producto,
            vigencia_desde,
            vigencia_hasta
       FROM HR.PRECIOS;
BEGIN
    l_lineas(1) := 'TAREA HR.PRECIOS';


    SELECT COUNT(*) INTO nCuantos FROM HR.PRECIOS;

    OPEN cdModulo5_14;
    FOR i IN 1..nCuantos LOOP
        FETCH cdModulo5_14 INTO
           PRECIO,
           COD_PRODUCTO,
           VIGENCIA_DESDE,
           VIGENCIA_HASTA;
        l_lineas(i+1) := PRECIO       || ', ' ||
                    COD_PRODUCTO  || ', ' ||
                    VIGENCIA_DESDE || ', ' ||
                    VIGENCIA_HASTA;
        EXIT WHEN cdModulo5_14%NOTFOUND;
    END LOOP;
    HR.respalda_tabla('TEMP','PRECIOS.txt',l_lineas);
    DBMS_OUTPUT.PUT_LINE
    (
         'SE RESPALDO EXITOSAMENTE LA TABLA HR.PRECIOS'
    );
    CLOSE cdModulo5_14;
    COMMIT;
END;
```

- **Código 10. Tabla PRODUCTS (PRODUCTOS).**

```
DECLARE
    l_lineas DBMS_SQL.varchar2s;
    nCuantos NUMBER;

    COD_PRODUCTO productos.cod_producto%TYPE;
    COD_FAMILIA productos.cod_familia%TYPE;
    DESCRIPCION productos.descripcion%TYPE;

    CURSOR cdModulo5_14
    IS
       SELECT cod_producto,
            cod_familia,
            descripcion
       FROM HR.PRODUCTOS;
BEGIN
    l_lineas(1) := 'TAREA HR.PRODUCTOS';


    SELECT COUNT(*) INTO nCuantos FROM HR.PRODUCTOS;

    OPEN cdModulo5_14;
    FOR i IN 1..nCuantos LOOP
          FETCH cdModulo5_14 INTO
             COD_PRODUCTO,
             COD_FAMILIA,
             DESCRIPCION;
          l_lineas(i+1) := COD_PRODUCTO   || ', ' ||
                       COD_FAMILIA    || ', ' ||
                       DESCRIPCION;
          EXIT WHEN cdModulo5_14%NOTFOUND;
    END LOOP;
    HR.respalda_tabla('TEMP','PRODUCTOS.txt',l_lineas);
    DBMS_OUTPUT.PUT_LINE
    (
          'SE RESPALDO EXITOSAMENTE LA TABLA HR.PRODUCTOS'
    );
    CLOSE cdModulo5_14;
    COMMIT;
END;
```

- **Código 11. REGIONS (REGIONES)**

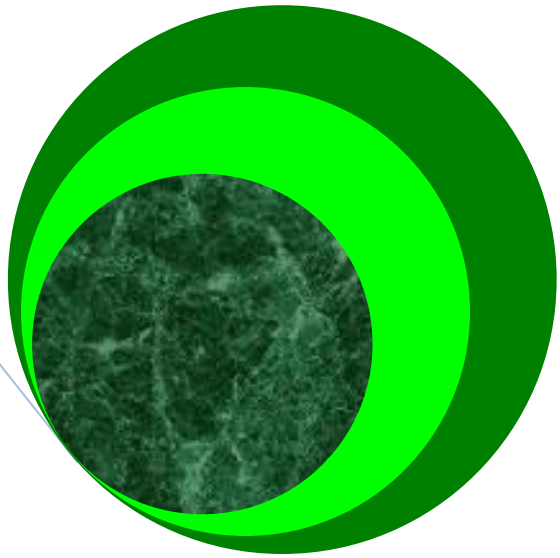
```
DECLARE
    l_lineas DBMS_SQL.varchar2s;
    nCuantos NUMBER;

    REGION_ID regions.region_id%TYPE;
    REGION_NAME regions.region_name%TYPE;

    CURSOR cdModulo5_14
    IS
       SELECT region_id,
            region_name
       FROM HR.REGIONS;
BEGIN
    l_lineas(1) := 'TAREA HR.REGIONS';


    SELECT COUNT(*) INTO nCuantos FROM HR.REGIONS;

    OPEN cdModulo5_14;
    FOR i IN 1..nCuantos LOOP
        FETCH cdModulo5_14 INTO
            REGION_ID,
            REGION_NAME;
        l_lineas(i+1) := REGION_ID   || ', ' ||
                    REGION_NAME;
        EXIT WHEN cdModulo5_14%NOTFOUND;
    END LOOP;
    HR.respalda_tabla('TEMP','REGIONS.txt',l_lineas);
    DBMS_OUTPUT.PUT_LINE
    (
         'SE RESPALDO EXITOSAMENTE LA TABLA HR.REGIONS'
    );
    CLOSE cdModulo5_14;
    COMMIT;
END;
```

- **Código 12. Tabla VENTAS.**

```
DECLARE
    l_lineas DBMS_SQL.varchar2s;
    nCuantos NUMBER;

    NUM_VENTA ventas.num_venta%TYPE;
    FECHA_VENTA ventas.fecha_venta%TYPE;

    CURSOR cdModulo5_14
    IS
       SELECT num_venta,
            fecha_venta
       FROM HR.VENTAS;
BEGIN
    l_lineas(1) := 'TAREA HR.VENTAS';


    SELECT COUNT(*) INTO nCuantos FROM HR.VENTAS;

    OPEN cdModulo5_14;
    FOR i IN 1..nCuantos LOOP
          FETCH cdModulo5_14 INTO
              NUM_VENTA,
              FECHA_VENTA;
          l_lineas(i+1) := NUM_VENTA   || ', ' ||
                       FECHA_VENTA;
          EXIT WHEN cdModulo5_14%NOTFOUND;
    END LOOP;
    HR.respalda_tabla('TEMP','VENTAS.txt',l_lineas);
    DBMS_OUTPUT.PUT_LINE
    (
          'SE RESPALDO EXITOSAMENTE LA TABLA HR.VENTAS'
    );
    CLOSE cdModulo5_14;
    COMMIT;
END;
```

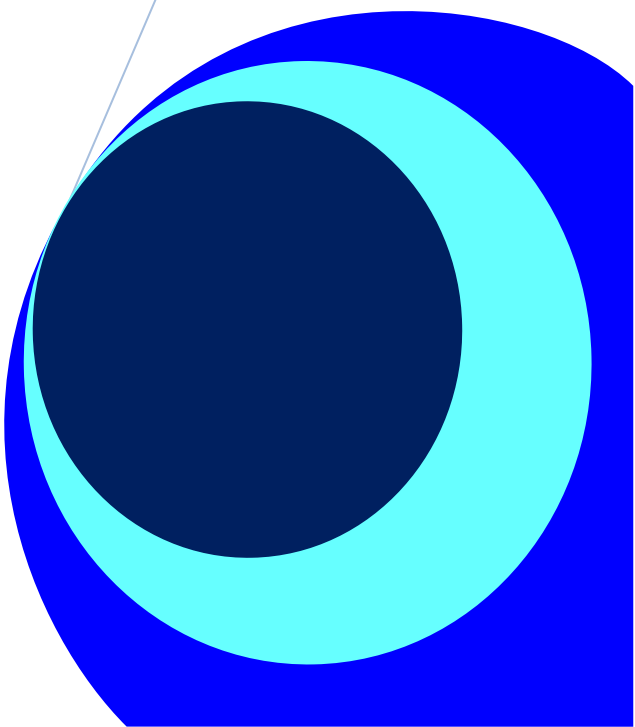
- **Código 13. Tabla VENTAS_DETALLE.**

```
DECLARE
    l_lineas DBMS_SQL.varchar2s;
    nCuantos NUMBER;

    NUM_VENTA ventas_detalle.num_venta%TYPE;
    COD_PRODUCTO ventas_detalle.cod_producto%TYPE;
    CANTIDAD ventas_detalle.cantidad%TYPE;
    MONTO ventas_detalle.monto%TYPE;

    CURSOR cdModulo5_14
    IS
       SELECT num_venta,
            cod_producto,
            cantidad,
            monto
       FROM HR.VENTAS_DETALLE;
BEGIN
    l_lineas(1) := 'TAREA HR.VENTAS_DETALLE';


    SELECT COUNT(*) INTO nCuantos FROM HR.VENTAS_DETALLE;

    OPEN cdModulo5_14;
    FOR i IN 1..nCuantos LOOP
         FETCH cdModulo5_14 INTO
            NUM_VENTA,
            COD_PRODUCTO,
            CANTIDAD,
            MONTO;
         l_lineas(i+1) := NUM_VENTA     || ', ' ||
                     COD_PRODUCTO  || ', ' ||
                     CANTIDAD || ', ' ||
                     MONTO;
         EXIT WHEN cdModulo5_14%NOTFOUND;
    END LOOP;
    HR.respalda_tabla('TEMP','VENTAS_DETALLE.txt',l_lineas);
    DBMS_OUTPUT.PUT_LINE
    (
          'SE RESPALDO EXITOSAMENTE LA TABLA HR.VENTAS_DETALLE'
    );
    CLOSE cdModulo5_14;
    COMMIT;
END;
```



ANEXO V

Sistema de Ventas.



PROYECTO 5. DISEÑO DE UN SISTEMA DE VENTAS PARA UNA TIENDA DE ABARROTES.

- **Presentación.**

La venta es un conjunto de actividades diseñadas para promover la compra de un producto o servicio; motivo por el cual la venta requiere de un proceso que ordene la implementación y ejecución de sus diversas actividades.

El Sistema de Ventas cuenta con diversos módulos que permiten realizar de manera eficaz el control y seguimiento del ciclo comercial; desde el ingreso del pedido, pasando por la aprobación, facturación, despacho, cobranzas, administración de cartera, manejo de concesiones y consignaciones, brindando consultas y reportes para su gestión, y control de auditoría para cada operación efectuada en el sistema.

- **Objetivos:**

a) Automatizar el control y seguimiento de las Oportunidades de Venta (OV).

b) Almacenar información concerniente a los clientes.

c) Almacenar información concerniente a estudios de mercado.

- **Identificación de Necesidades:**

1. **Investigación de las particularidades de cada cliente en perspectiva**.

Se busca información más específica del cliente en perspectiva, por ejemplo: Nombre Completo, Edad, Sexo, Hobbies, Estado Civil, Nivel de Educación, Teléfono, Correo Electrónico, etc.

Adicionalmente, es necesario buscar información relacionada con la parte comercial, por ejemplo: Productos similares que usa actualmente, Motivos para usar los productos, Que piensa de ellos, Estilo de compra, etc.

2. **Preparación de la presentación de ventas enfocada en el posible cliente**.

Con la información obtenida se prepara una presentación de ventas adaptada a las necesidades cada cliente en perspectiva.

Para la presentación, se sugiere elaborar una lista de todas las características que tiene el producto, transformarlas en beneficios para el cliente y finalmente se establece las ventajas con relación a la competencia.

- **Identificación de Procesos:**

a) **Aparición de una Oportunidad de Venta**. Llega a la organización una información por la que un sujeto, sea cliente o prospecto (posible cliente), tiene la posibilidad de comprar productos o servicios ofertados por la misma.

b) **Alta de Oportunidad de Venta**. El ingreso de los datos de la OV (el alta) se realiza: automáticamente por el sistema (por ejemplo cuando el origen es la Web de la Empresa) o manualmente por uno de los empleados autorizados para ello.

c) ¿**Cliente o Prospecto?.** Mientras se introduce la OV, el sistema o el empleado; comprueba si el sujeto es ya cliente, si no lo es, lo da de alta como "prospecto" o posible cliente.

d) **Asignar Responsable**. Una vez introducidos los datos de la OV, ésta llega al Coordinador de Oportunidades, quién lo asigna a un empleado que cuente con las características más adecuadas para dar seguimiento a la OV. Este empleado se convierte en el Responsable de la OV y como tal, recibe en su workflow una tarea del proceso diseñado para el seguimiento de la OV. Si la OV va a ser tratado por un "partner" (concesionario), es decir, si se trata de una venta indirecta o por canal, dicho "partner" será el responsable de la OV.

e) **Acciones de Seguimiento**. El Responsable de la OV va introduciendo en dicha tarea (que se mantiene activa todo el tiempo que dura el seguimiento de la OV hasta su conversión en pedido o su desestimación), cada una de las acciones que componen el seguimiento, para que se realicen eficazmente y a tiempo. Esta es la parte fundamental del procedimiento, ya que los éxitos individuales de las acciones son los que determinan el éxito final de la OV. Entre estas acciones cabe destacar las relativas a la preparación y envío de la oferta al sujeto y su conversión en pedido como consecuencia de la firma de conformidad por su parte.

f) **Conversión en Pedido o Desestimación**. Cuando el sujeto toma su decisión final, ésta puede ser de desestimación o aceptación de la oferta. En este último caso, la oferta se transforma en pedido y el procedimiento sigue con el proceso de suministro (aprovisionamiento, producción y entrega) de los bienes y servicios solicitados.

g) **Observatorios de los Responsables**. Cada responsable puede ver, en un observatorio, el conjunto de acciones de seguimiento de todas las OV que tiene a su cargo a lo largo del tiempo con indicación de los costos asociados e incluso exportar la información a su "planning" global privado, junto con las demás actividades previstas en su agenda.

h) **Observatorio del Coordinador**. El Coordinador, desde su propio observatorio, tiene acceso a todas las OV, con la información de todas las acciones propuestas, en curso y realizadas, por cada uno de los responsables de su equipo.

- **Reportes Generados:**

a) **Reporte General de Ventas.** Muestra todas las ventas registradas en la tienda.

b) **Reporte de Ventas por Caja.** Muestra las ventas registradas de cada caja de la tienda.

c) **Reporte de Ventas por Cajero.** Muestra las ventas que realizo cada cajero.

d) **Reporte de Ventas por Fecha.** Muestra las ventas por la fecha en que fueron realizadas.

e) **Reporte de Ventas por Día de la Semana.** Muestra las ventas realizadas de cualquier día y semana.

f) **Reporte de Ventas por Hora del Día.** Muestra las ventas realizadas en cualquier hora del día.

g) **Reporte de Ventas por Sección.** Muestra las ventas realizadas por cada una de las secciones o departamentos de la tienda.

- **Fases o Etapas de la Base de Datos:**

Para el diseño y desarrollo del Sistema de Ventas, se utiliza el Lenguaje Unificado de Modelado (UML) del Paradigma Orientado a Objetos:

a) **Workflow de Negocio.** El modelado del negocio muestra la función de la compañía en el mundo. Sus propósitos principales son:

- Entender la estructura y dinámica de la organización.
- Comprender problemas existentes en la organización e identificar mejoras potenciales.
- Asegurar que clientes, usuarios y desarrolladores tengan una visión común y completa del sistema.

b) **Workflow de Requerimientos.** Se desarrolla un modelo del sistema que se va a construir, identificando los casos de uso más importantes y los actores involucrados, de manera de llegar a un acuerdo con el cliente sobre el sistema a desarrollar, incluyendo los siguientes pasos:

- Comprender el contexto del sistema.
- Capturar requisitos funcionales y no funcionales; identificando los requisitos mediante casos de uso, representando las acciones que utilizará para el sistema y las propiedades, restricciones y características del mismo.
- Un conjunto de esbozos de interfaces de usuario y de prototipos para cada actor.

c) **Workflow de Análisis.** Es un modelo de objetos conceptual que analiza los requisitos mediante su refinamiento y estructuración; incluyendo su arquitectura. A través de:

- Diagrama de colaboración de use case.
- Agrupación de use case.
- Refinamiento del diagrama de clases.

d) **Workflow de Diseño.** Su entrada principal es el modelo de análisis, pero adapta el entorno de implementación elegido, adquiriendo una mayor comprensión de los requisitos no funcionales y restricciones relacionadas con el lenguaje de programación; que incluye la definición de clasificadores, sus relaciones entre estos y colaboraciones que llevan a cabo los casos de uso. Los modelos utilizados en esta etapa son:

- **Modelo de Diseño** que comprende el diagrama de clases de diseño, derivación a tablas para su implementación en base de datos relacional, definición de cada atributo y diagrama de estado.

- **Modelo de Despliegue** que comprende el diagrama de despliegue y la descripción del ambiente de implementación.

e) **Workflow de Implementación.** En esta etapa se toma el resultado del diseño y se implementa el sistema en términos de componentes que comprenden: ficheros de código fuente, script, ficheros de código binario, ejecutables, etc. Los propósitos de esta etapa son:

- Planificación de las integraciones del sistema en cada iteración.

- Distribuir el sistema asignando componentes ejecutables a cada nodo.

- Implementar clases y subsistemas encontrados durante el diseño.

- Probar los componentes individualmente para luego enlazarlos en uno o más ejecutables.

Los modelos desarrollados en esta etapa son:

- **Modelo de Implementación** que muestra como los componentes de diseño se transforman en componentes de implementación.

- **Modelo de Despliegue** que contiene la vista de la arquitectura del modelo de despliegue y la asignación de componentes.

f) **Workflow de Prueba.** Se verifica que el sistema integre su funcionalidad correctamente, a través de un modelo de prueba compuesto por casos y procedimientos de prueba. Sus objetivos son:

- Planificar las pruebas de cada construcción y del sistema.

- Diseñar e implementar pruebas.

- Realizar las pruebas y manejar los resultados de cada prueba sistemáticamente.

♦ **Software Empleado.**

a) Microsoft Windows XP SP2 Professional.
b) Microsoft Windows XP Service Pack 3.
c) Windows Installer 3.1.
d) Microsoft .NET Framework 3.5 Service Pack 1.
e) Oracle Database 10g Express Edition en Español.
f) Java (TM) SE Development Kit 6 Update 17.
g) Net Beans IDE 6.7.1.
h) VISOR SVG de Adobe Versión 3.0.

BIBLIOGRAFÍA.

1) **LUCAS**, Henry C., Sistemas de Información. Análisis, Diseño y Puesta a Punto.,
Madrid, Ed. Paraninfo, 1987, pp. 593.

2) **MURDICK**, Robert G y John C. Munson., Sistemas de Información Administrativa.,
México, Ed. Prentice Hall, 1988, pp. 722.

3) **PÉREZ LÓPEZ**, **César**., Microsoft SQL Server 2005. Administración y Análisis de Bases de Datos.,
México, Ed. Alfaomega Ra-ma, 2007, pp. 778.

4) **PÉREZ LÓPEZ**, César., Oracle 10g: Administración y Análisis de Bases de Datos, 2ª ed.,
México, Ed. Alfaomega Ra-ma, 2008, pp. 693.

5) **PÉREZ LÓPEZ**, César., Oracle PL/SQL.,
México, Ed. Alfaomega, 2008, PP. 400.

6) **STANEK**, William R., SQL Server 2005. Manual del Administrador.,
México, Ed. McGraw Hill, 2005, pp. 549.

7) **ROB**, **Peter y Carlos Coronel**., Sistemas de Bases de Datos. Diseño, Implementación y Administración.,
5ª ed.,
México, Ed. Thomson, 2006, pp. 838.