CAP 2. DISEÑO E IMPLEMENTACION DE BD RELACIONALES

CONTENIDO

2.1. INTRODUCCION

La información clara, oportuna y útil requiere datos precisos. Tales datos deben generarse apropiadamente y deben guardarse adecuadamente en un formato que sea fácil de acceder y de procesar. La administración de datos es una disciplina enfocada a la generación, almacenamiento y recuperación apropiada de datos.

En general la administración eficiente de datos requiere el uso de una base de datos de computadora, una estructura integrada, compartida que aloja un conjunto de datos para el usuario final, es decir hechos interesantes para el usuario, también contiene metadatos, que son los que describen las características de los datos y las relaciones que vinculan a aquellos que están incluidos en la base de datos. El software que ayuda a manejar las bases de datos es el Sistema de Gestión de Base de datos (SGBD), que es un conjunto de programas que maneja la estructura de las bases de datos desde su creación y controla el acceso a los datos guardados en ellas. El SGBD comparte las bases de datos entre múltiples aplicaciones y usuarios.

Una buena base de datos no es algo que solamente suceda, la estructura de su contenido debe diseñarse con cuidado. Una base de datos bien diseñada facilita la administración de datos y se convierte en un valioso y confiable generador de información. Mientras que una mal diseñada probablemente se convierta en tierra de cultivo de datos redundantes, es decir datos innecesariamente duplicados. En muchas ocasiones los datos son los causantes de errores de información difíciles de rastrear.

Como las bases de datos son las fuentes en las que se genera la información, su diseño es tema de estudio detallado en el ambiente de datos moderno. El diseño de las bases de datos es muy importante como para dejarlo a la suerte, es por eso que los estudiantes universitarios estudian diseño de bases de datos y las organizaciones de todos los tipos y tamaños envían a su personal a cursos y seminarios sobre bases de datos, los consultores de diseño de bases de datos con frecuencia son muy bien remunerados. Debido a que la información es realmente importante en la llamada era de la información, la capacidad para producir bases de datos funcionales tiene un gran valor práctico.

2.2. DISEÑO DE BASES DE DATOS RELACIONALES

Actualmente la mayoría de la Empresas e Instituciones trabajan con aplicaciones de escritorio o basados en la web que incluyen bases de datos relacionales. La manipulación de datos es algo común en el mundo del manejo de información, las grandes y pequeñas empresas confían su información en bases de datos, proceso que suele tomar un tiempo valioso no solo en el almacenamiento, sino también en la recuperación.

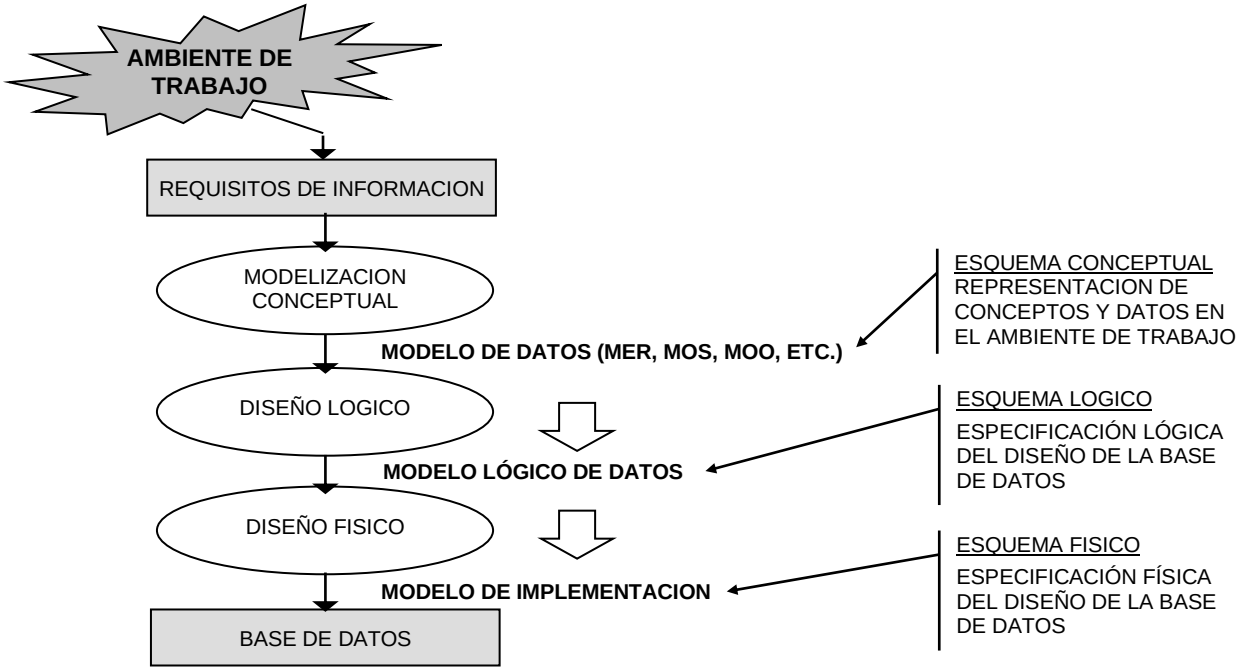
Las metodologías de diseño constituyen un **conjunto de técnicas** que se aplican de acuerdo con una secuencia de pasos o etapas. Cada paso tiene una meta específica y con frecuencia usan diferentes técnicas. Las técnicas de diseño, por lo general son de dos clases:

(1) técnicas de análisis de datos (modelado), que permiten capturar el significado de los datos del usuario.

(2) técnicas de diseño lógico y técnico, que permiten convertir los resultados del análisis en una implementación física de la base de datos, a través de una especificación lógica de la misma.

Por tanto son tres los pasos o etapas que se necesitan para el diseño de una base de datos:

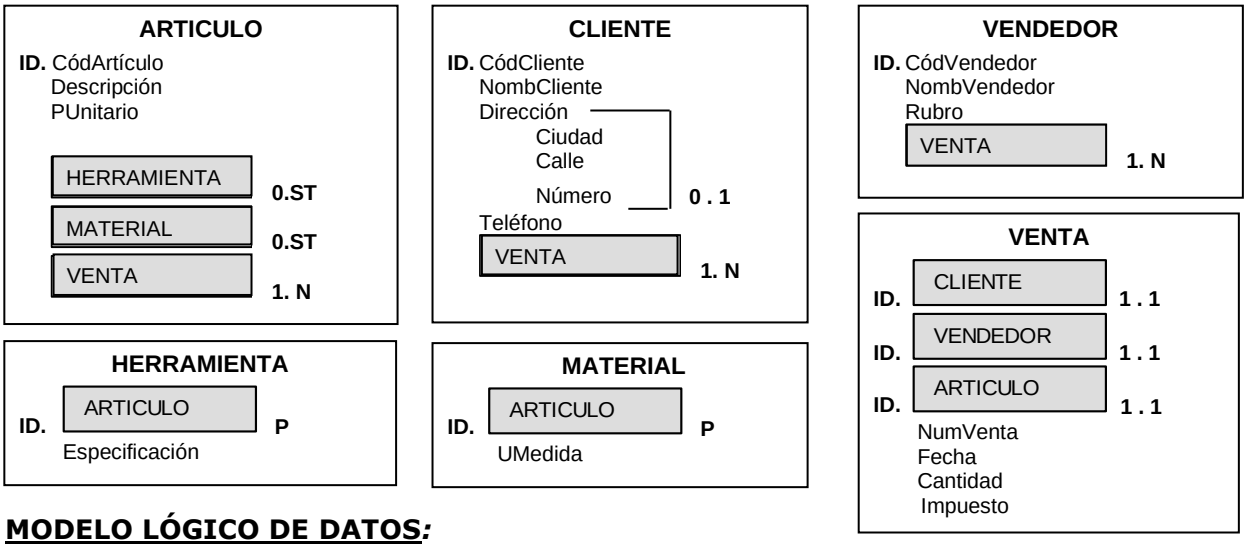
- **Modelización Conceptual:** proceso que implica crear una representación de la visión que los usuarios tienen de los datos en un determinado entorno de trabajo. El modelo obtenido es un conjunto de conceptos utilizados para describir la estructura de la base de datos:
- **Diseño Lógico:** proceso que permite obtener el modelo lógico de datos, que es la especificación lógica de la estructura de la base diseñada. Es decir las tablas, los atributos, llaves primarias, llaves foráneas y las correspondientes referencias relacionales, tomando en cuenta la multiplicidad de las mismas.
- **Diseño Físico:** permite obtener el modelo de implementación, que se constituye en la especificación física del diseño. Incluyendo los detalles físicos de la estructura de las tablas: campos (atributos), tipos de datos, longitudes, formatos, etc.



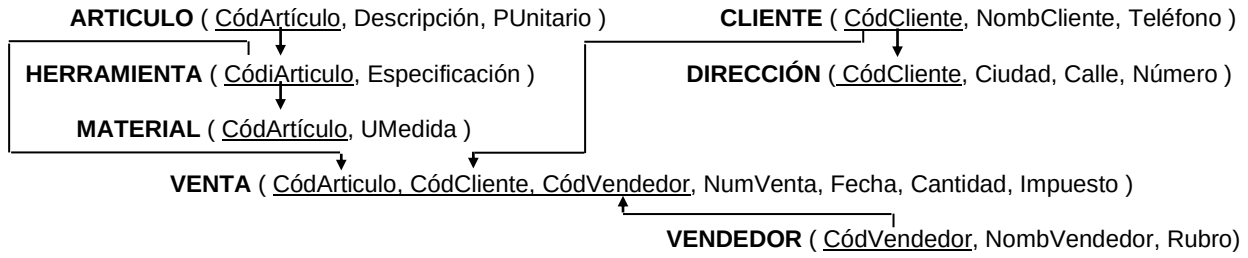
EJEMPLO:

El siguiente ejemplo muestra los tres pasos o etapas de la metodología de diseño de bases de datos relacionales.

MODELO DE DATOS:



MODELO LÓGICO DE DATOS:



MODELO FISICO (MODELO DE IMPLEMENTACIÓN):

No.	Atributo	Tipo de dato	Longitud	Decimales	Formato
TABLA: ARTICULO					
1	CódArtículo	Cadena	6	-	A-xxxx
2	Descripción	Cadena	35	-	-
3	PUnitario	Real	8	2	xxxxxxx.xx
HERRAMIENTA					
1	CódArtículo	Cadena	6	-	A-xxxx
2	Especificación	Cadena	40	-	-
MATERIAL					
1	CódArtículo	Cadena	6	-	A-xxxx
2	UMedida	Cadena	4	-	xxxx
CLIENTE					
1	CódCliente	Cadena	6	-	C-xxxx
2	NombCliente	Cadena	35	-	-
3	Teléfono	Entero	8	-	xxxxxxx
DIRECCIÓN					
1	CódCliente	Cadena	6	-	C-xxxx
2	Ciudad	Cadena	20	-	-
3	Calle	Cadena	30	-	-
4	Número	Entero	8	-	xxxxxxx
VENDEDOR					
1	CódVendedor	Cadena	6	-	V-xxxx
2	NombVendedor	Cadena	35	-	-
3	Rubro	Cadena	40	-	-
VENTA					
1	CódArticulo	Cadena	6	-	A-xxxx
2	CódCliente	Cadena	6	-	C-xxxx
3	CódVendedor	Cadena	6	-	V-xxxx
4	NumPedido	Entero	8	-	xxxxxxx
5	Fecha	Fecha	10	-	xx / xx / xxxx
6	Cantidad	Real	8	2	xxxxxxx.xx
7	Impuesto	Real	8	2	xxxxxxx.xx

EJERCICIO:

Se desea controlar la actividad que realizan los profesionales de una Universidad. De cada profesional se conoce su carnet de identidad, nombre, sexo, título universitario y año de graduación. Un profesional puede estar trabajando como investigador o como docente. Para cada investigador se conoce su categoría y el centro de investigación donde trabaja. Por otra parte para cada docente se conoce también su categoría docente, la facultad, el departamento en que trabaja. En la Universidad se contemplan distintas disciplinas (física, química, matemáticas, informática, etc.) De cada disciplina se sabe su código, nombre y la cantidad de asignaturas que la componen. En la Universidad también se desarrollan varias especialidades (industrial, mecánica, electrónica, sistemas, etc.). De cada especialidad se conoce el código, nombre y facultad encargada de desarrollarla. Una disciplina se imparte en varias especialidades y en una especialidad se imparten diferentes disciplinas. Una disciplina y una especialidad, forman un perfil. Para cada perfil se conoce la cantidad de horas de clase. Un docente imparte clases en diferentes perfiles y en un perfil imparten clases muchos docentes. Un investigador realiza su labor investigativa asociada a una disciplina, pero existen muchos investigadores que realizan sus investigaciones asociados a una misma disciplina. **Realizar el diseño de la base de datos**.

- **ANALISIS:**

Consiste en identificar los objetos (entidades), los atributos y las relaciones binarias en el ambiente de trabajo especificado en el enunciado del problema.

1.- OBJETOS **2.- ATRIBUTOS**

UNIVERSIDAD	NOMBRE, DIRECION
ESPECIALIDAD	CODIGO, NOMBRE, FACULTAD_ENCARGADA
DISCIPLINA	CODIGO, NOMBRE, CANTIDAD_ASIGNATURAS
PROFESIONAL	CARNET_ID, NOMBRE, TITULO_UNIVERSIDATIO, AÑO_GRADUACION
INVESTIGADOR	CATEGORIA, CENTRO_INVESTIGACION
DOCENTE	CATEGORIA, FACULTAD, DEPARTAMENTO
PERFIL	CANTIDAD_HORAS

El objeto Universidad se puede o no incluir en el modelo, pero con la finalidad de obtener un sistema que pueda ser configurado para cualquier universidad, dicho objeto se tomara en cuenta en el análisis.

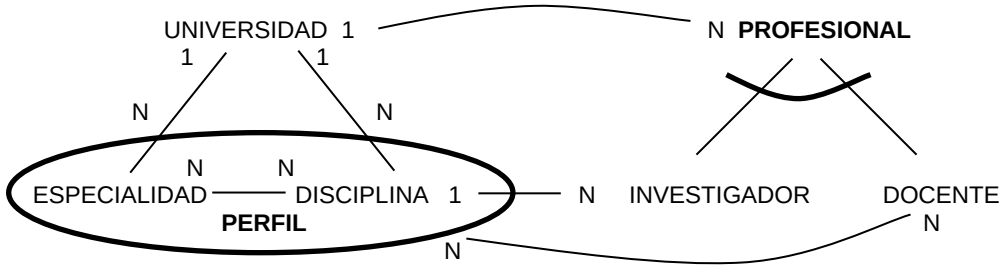
3.- RELACIONES BINARIAS

UNIVERSIDAD 1 -----**Trabaja**----- N PROFESIONAL
UNIVERSIDAD 1 ----**Desarrolla**--- N ESPECIALIDAD
UNIVERSIDAD 1 ---**Contempla**--- N DISCIPLINA
INVESTIGADOR 1 ----**Asociado**---- N DISCIPLINA
DISPLINA N ---**Imparte**--- N ESPECIALIDAD
DOCENTE N ---**Imparte**--- N PERFIL

DISCIPLINA – ESPECIALIDAD → **PERFIL** (AGREGACION)

INVESTIGADOR, DOCENTE → **PROFESIONAL** (GENERALIZACION)

4.- GRAFO DE RELACIONES



SIMBOLOS EN GRAFOS RELACIONALES

RELACION BINARIA ——————

NOMBRE RELACION < nombre >

RELACION CON ENTIDAD DEBIL - - - - - - - - - - -

CARDINALIDADES 1 N

RELACION TERNARIA

ATRIBUTO DE RELACION

AGREGACION

GENERALIZACION

REGLAS PARA LAS RELACIONES EN EL MLD

R1: LAS ENTIDADES BUSCAN LAS LLAVES PRIMARIAS EN LAS TABLAS DE RELACION

TABLA A (CODA, AT1, AT2, ...)

TABLA B (CODB, BT1, BT2, ...)

RELACION (CODA, CODB, RT1, RT2, ...)

R2: LA LLAVE FORANEA BUSCA A LA LLAVE PRIMARIA DE LA OTRA TABLA

TABLA A (CODA, AT1, AT2, ...)

TABLA B (CODB, BT1, BT2, ... CODA)

R3: LA ENTIDAD FUERTE DA LA LLAVE PRIMARIA A LA DEBIL

TABLA A (CODA, AT1, AT2, ...)

T_DEBIL (CODA, GT1, GT2, ...)

• MODELO DE OBJETOS SEMANTICOS (MOS)

UNIVERSIDAD

ID. CODUNI
NOMBUNI
DIRECCION

PROFESIONAL	1 .N
ESPECIALIDAD	1 . N
DISCIPLINA	1 . N

DISCIPLINA

ID. CODDIS
NOMBDIS
CANTASIG

UNIVERSIDAD	1 . 1
PERFIL	1 . N
INVESTIGADOR	1 . N

ESPECIALIDAD

ID. CODESP
NOMBESP
FACULTADENC

UNIVERSIDAD	1 . 1
PERFIL	1 . N

PERFIL

ID.	ESPECIALIDAD	1 . 1
ID.	DISCIPLINA	1 . 1

CANTHORAS

DOCENTE	1 . N

PROFESIONAL

ID. CI
NOMBPRO
SEXO
TITULO
AG
CAT

INVESTIGADOR	0 . ST
DOCENTE	0 . ST
UNIVERSIDAD	1 . 1

INVESTIGADOR

ID. PROFESIONAL **P**
CENTRO
DISCIPLINA 1 .1

DOCENTE

ID. PROFESIONAL **P**
FACULTAD
DEPTO
PERFIL 1 . N

• MODELO LOGICO DE DATOS

UNIVERSIDAD (CODUNI, NOMBUNI, DIRECION)

ESPECIALIDAD (CODESP, NOMBESP, FACULTADENC, CODUNI)

DISCIPLINA (CODDIS, NOMBDIS, CANTASIG, CODUNI)

PROFESIONAL (CI, NOMBPRO, SEXO, TITULO, AG, CAT, CODUNI)

INVESTIGADOR (CI, CATEGORIA, CENTRO)

DOCENTE (CI, CATEGORIA, FACULTAD, DEPTO)

PERFIL (CODESP, CODDIS, CANTHORAS)

IMPARTIR (CODESP, CODDIS, CI)

(A) (A)

- **MODELO DE IMPLEMENTACIÓN:**

No.	Atributo	Tipo de dato	Longitud	Decimales	Formato
TABLA: UNIVERSIDAD					
1	CodUni	Cadena	6	-	U-xxxx
2	NombUni	Cadena	35	-	-
3	Dirección	Cadena	30	-	-
	ESPECIALIDAD				
1	CodEsp	Cadena	6		E-xxxx
2	NombEsp	Cadena	35	-	-
3	FacultadEnc	Cadena	30	-	-
4	CodUni	Cadena	6		U-xxxx
	DISCIPLINA				
1	CodDis	Cadena	6		D-xxxx
2	NombDis	Cadena	35	-	-
3	CantAsig	Entero	8	-	Xxxxxxxx
4	CodUni	Cadena	6		U-xxxx
	PROFESIONAL				
-	-	-	-	-	-
-	-	-	-	-	-

2.3. IMPLEMENTACION DE BD RELACIONALES

Los problemas inherentes al manejo de datos originan el uso de los sistemas de bases de datos. El término "Sistema de bases de datos ", se refiere a la organización de componentes que definen y regulan la recolección, almacenamiento, administración y uso de los datos dentro de un ambiente de base de datos.

Por tanto para la implementación de bases de datos, se debe utilizar los denominados Sistemas de gestión de bases de datos (SGBD) que son aplicaciones (conjunto de programas) que permiten definir, crear, mantener las bases de datos, así mismo permite almacenar, administrar y recuperar los datos contenidos en ella proporcionando un acceso controlado a las bases de datos.

El Sistema de gestión de base de datos (SGBD), junto con las bases de datos y los usuarios, constituyen el "Sistema de bases de datos "

CLASIFICACION DE LOS SISTEMAS DE GESTIÓN DE BD (SGBD)

Los SGBDs, se pueden clasificar tomando en cuenta tres aspectos: número de usuarios, la ubicación de la base de datos y el tipo o grado de usos esperados.

El número de usuarios determina si el SGBD se clasifica como de **usuario único** (**monousuario)** o de **usuarios múltiples** (**multiusuario)**. Un SGBD para usuario único soporta solo un usuario a la vez. Si una base de datos de usuario único corre en una computadora personal, también se llama **BD de escritorio**. En contraste, un SGBD para usuarios múltiples soporta a varios usuarios al mismo tiempo. Si la base de datos multiusuario soporta un número relativamente pequeño de usuarios en un departamento específico dentro de una organización, se llama **BD de grupo de trabajo**. Si la base de datos es utilizada pos usuarios de toda la organización en muchos departamentos se conoce como **BD empresarial**.

El SGBD también puede clasificarse de acuerdo con la ubicación de la base de datos, así un SGBD que soporta una base de datos localizada en un solo sitio se llama **Centralizado** y el que soporta una base de datos localizada en forma total o particionada en varios sitios, se denomina **Distribuido**.

El tipo y grado de uso clasifica a los SGBD como **Transaccional** o de **Producción**, principalmente diseñados para soportar transacciones que requieren de una respuesta inmediata. Una transacción es un grupo de instrucciones que modifican la base de datos. En contraste una base de datos de soporte de decisiones requiere de un almacén de datos que guarde un gran volumen de información histórica extraída de diversas fuentes, estas últimas se denominan **BD para soporte de decisiones** (DSS), **Datawarehouse** (DW) o **DataMarts** (DM).

Otros usos actuales de las bases de datos están relacionados con aplicaciones más específicas, como son las bases de datos Cliente-servidor, bases de datos multimedia, bases de datos de información geográfica (SIG), bases de datos de comercio electrónico, etc.

ESTRUCTURA GENERAL DE UN SGBD

Los SGBDs son paquetes de software muy complejos y sofisticados, no se puede generalizar sobre los elementos que lo componen ya que varían mucho unos de otros, sin embargo entre los módulos más comunes y que realizan una función específica en el SGBD, se tienen: el procesador de consultas. Administrador de base de datos, administrador de ficheros, preprocesador y compilador de instrucciones y el administrador del diccionario de datos.

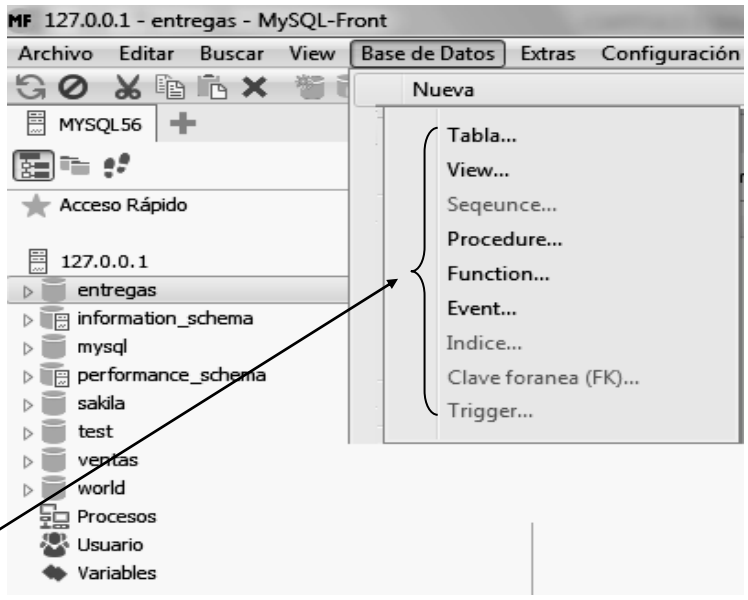
Entre otros componentes también importantes se tienen: el administrador de seguridad de acceso y control de autorización, administrador de concurrencia, administrador de la integridad de datos, administrador de presentación de datos y optimizador de consultas, administrador de transacciones, administrador de almacenamiento de datos, y el administrador de respaldo y recuperación.

COMPONENTES DE LA BASE DE DATOS

Una base de datos consta de una colección de tablas con datos y otros objetos como Vistas, índices, procedimientos almacenados, funciones, reglas, desencadenadores y otros que se definen para poder llevar a cabo distintas operaciones con los datos.

El SGBD puede trabajar con un gran número de bases de datos y cada una de ellas puede almacenar datos interrelacionados o datos no relacionados con los de las otras bases de datos.

Para crear una base de datos que responda a las necesidades se debe conocer cómo diseñar, crear y mantener cada uno de esos componentes para asegurar un funcionamiento óptimo de la base de datos.



USUARIOS DE BASES DE DATOS

Se puede definir como usuarios a toda persona que tenga todo tipo de contacto con el sistema de base de datos desde que se diseña, elabora, termina y se usa:

- **Programadores de aplicaciones:** Profesionales en computación e informática que interactúan con el sistema por medio de lenguajes de definición y manipulación de datos ya sean como el SQL o como Cobol, C++, Java, etc.
- **Usuarios sofisticados:** Son usuarios que interactúan con el sistema recuperando datos escribiendo instrucciones principalmente de preguntas utilizando algún lenguaje de consulta de base de datos como SQL, QBE, etc.
- **Usuarios especializados:** Usuarios sofisticados que escriben aplicaciones de bases de datos especializadas como son de salud, estadística, gubernamentales, comerciales, etc.
- **Usuarios comunes y principiantes:** Son usuarios no sofisticados que interactúan con el sistema a través de algún programa de interface gráfico o algún programa de aplicación previamente desarrollado. Es un usuario que no conoce el diseño interno de la base de datos, solo le interesa los datos contenidos en la misma.

LENGUAJE DE CONSULTA ESTRUCTURADO SQL

El lenguaje SQL está compuesto por comandos, cláusulas, operadores y funciones de agregado. Estos elementos se combinan en las instrucciones para crear, actualizar y manipular y consultar las bases de datos.

Existen dos tipos de comandos SQL, que corresponden al lenguaje de definición de datos (LDD) y al lenguaje de manipulación de datos (LMD).

SQL
- LDD (LENGUAJE DE DEFINICION DE DATOS)
- LMD (LENGUAJE DE MANIPULACION DE DATOS)

- El **LDD** que permite crear y definir nuevas bases de datos, tablas, atributos e índices.

LDD
- CREATE (CREAR)
- ALTER (MODIFICAR)
- DROP (BORRAR)

COMANDOS LDD

COMANDO	DESCRIPCIÓN
CREATE	UTILIZADO PARA CREAR NUEVAS BD, TABLAS, ATRIBUTOS E ÍNDICES
DROP	EMPLEADO PARA BORRAR O ELIMINAR BD, TABLAS E ÍNDICES
ALTER	UTILIZADO PARA MODIFICAR LAS BD, TABLAS AGREGANDO ATRIBUTOS O CAMBIANDO LA DEFINICIÓN DE LOS ATRIBUTOS.

- El **LMD** que permite manipular, actualizar, generar consultas para ordenar, filtrar y extraer datos de la base de datos.

LMD
- INSERT (INSERTAR)
- UPDATE (ACTUALIZAR)
- DELETE (ELIMINAR)
- SELECT (SELECCIONAR)

COMANDOS LMD

COMANDO	DESCRIPCIÓN
INSERT	UTILIZADO PARA INSERTAR TUPLAS DE DATOS EN LA BASE DE DATOS EN UNA ÚNICA OPERACIÓN.
UPDATE	UTILIZADO PARA MODIFICAR LOS VALORES DE LOS ATRIBUTOS Y TUPLAS ESPECIFICADAS
DELETE	UTILIZADO PARA ELIMINAR TUPLAS DE UNA TABLA DE UNA BASE DE DATOS
SELECT	UTILIZADO PARA CONSULTAR DATOS DE LA BASE DE DATOS QUE SATISFAGAN UN CRITERIO DETERMINADO.

LENGUAJE DE DEFINICION DE DATOS (LDD).

SQL tiene 3 sentencias básicas para la definición y eliminación de estructuras: La sentencia CREATE (Crear), ALTER (Modificar) y DROP (Eliminar).

CREATE
ALTER
DROP
} **OBJETO** (DATABASE, TABLE, INDEX, VIEW, PROCEDURE, ETC.)

La sintaxis de estos comandos varía de acuerdo al SGBD que se utilice. Para el caso de **MySQL** se tienen la siguiente sintaxis:

1.- COMANDOS PARA LA BASE DE DATOS:

CREATE
ALTER
DROP
} **DATABASE** NOMBREBASEDATOS;

- **CREACION DE LA BASE DE DATOS:**

Crea una nueva base de datos y especifica el conjunto de caracteres a utilizar y la colación correspondiente al mismo. Se puede usar **IF NOT EXIST** antes del nombre.

SINTAXIS:

```
CREATE DATABASE NOMBREBASEDATOS
    CHARACTER SET utf8 COLLATE utf8_general_ci;
```

utf8 : Es una codificación de caracteres de amplitud variable, que puede representar el conjunto de caracteres UNICODE compatible con ASCII.

- **ELIMINAR LA BASE DE DATOS:**

  Al quitar una base de datos se eliminan la base de datos y los archivos de disco que utiliza. Se puede usar **IF EXIST** antes del nombre.

  SINTAXIS: **DROP DATABASE** NOMBREBASEDATOS*;*

- **MODIFICAR LA DEFINICION DE LA BASE DE DATOS:**

  Permite modificar la especificación del conjunto de caracteres a utilizar y la colación correspondiente al mismo. Se puede usar **IF EXIST** antes del nombre.

  SINTAXIS:

  ALTER DATABASE NOMBREBASEDATOS
  CHARACTER SET xxxx **COLLATE** xxxx_general_ci;

  xxxx : Puede ser: utf8, utf8mb4, ascii, binary,latin1, latin2, dec8, armscii8, greek, hp8, koi8u,cp1250, gb2312, tis620, utf16, utf32, usc2, etc.

  La colación como: xxxx_general.ci, xxxx_german2.ci, xxxx_hungarian.ci, xxxx_latvian.ci, xxxx_persian.ci, xxxx_roman.ci, xxxx_slovak.ci, xxxx_spanish.ci, xxxx_unicode.ci, etc.

2.- COMANDOS PARA LAS TABLAS:

Las Bases de datos y sus correspondientes tablas en MySQL, se pueden crear utilizando el formato **InnoDB** o el formato de versiones anteriores **MyISAM**. A diferencia de las tablas MyISAM, las tablas InnoDB usan un table space para almacenar todas las tablas con sus índices. En versiones anteriores era necesario especificar el tamaño y ruta del table space, ahora eso no es necesario ya que se hace de manera automática. Cuando MysQL soporta tablas InnoDB, se crea el archivo **ibdata1** con un tamaño de 10MB y cuando este archivo requiere de más tamaño se expande 8MB. Adicionalmente se crean dos archivos lógicos de 5MB. **ib_logfile0** y **ib_logfile1**. Estos tres archivos están en formato binario.

InnoDB y MyISAM, son subsistemas de almacenamiento del MySQL.

Para realizar la creación de tablas dentro de una determinada Base de datos, previamente se debe abrir la misma, utilizando el comando USE.

SINTAXIS: **USE** NOMBREBASEDATOS;

A continuación los comandos relacionados con el manejo de tablas:

CREATE
ALTER } **TABLE** NOMBRETABLA;
DROP

- **CREACION DE UNA TABLA:**

  Define la estructura de una tabla y sus relaciones dentro de una determinada base de datos previamente especificada. Se puede usar **IF NOT EXIST** antes del nombre.

  SINTAXIS:

```
CREATE TABLE NOMBRETABLA
    ( ATRIB1    TIPODATO1,
      ATRIB2    TIPODATO2,
      -
      - ) ENGINE = InnoDB;
```

Incluye la lista de atributos de la tabla, el nombre del atributo, el tipo de dato, longitud y las restricciones respectivas, así como el formato de almacenamiento InnoDB o MyISAM (se puede utilizar **ENGINE** o **TYPE**).

Ejemplo de creación de la tabla contrato en la Base de datos Ventas:

```
USE VENTAS;
CREATE TABLE IF NOT EXISTS CONTRATO
    (
       NCONTRATO INT(4) AUTO_INCREMENT NOT NULL UNIQUE,
       FECHA DATETIME DEFAULT NOW(),
       MONTOTOTAL FLOAT(6,2) NOT NULL,
       DURACIÓN INT(8) DEFAULT 0
       NIT CHAR(12),
       CHECK (MONTOTOTAL > 0),
       PRIMARY KEY (NCONTRATO),
       FOREING KEY (NIT) REFERENCES EMPRESA(NIT)
    ) ENGINE = InnoDB  AUTO_INCREMENT =1001;
```

Las restricciones que generalmente se utilizan en MySQL son:

NOT NULL	VALORES NO NULOS.
UNIQUE	VALORES ÚNICOS.
PRIMARY KEY	LLAVE PRIMARIA.
FOREIGN KEY	LLAVE FORANEA.
CHECK	LIMITA LOS VALORES QUE PUEDEN INTRODUCIRSE EN EL ATRIBUTO.
AUTO_INCREMENT	NUMERA AUTOMÁTICAMENTE CADA VALOR DEL ATRIBUTO
DEFAULT	ESTABLECE VALORES POR DEFECTO.
NOW()	FUNCIÓN QUE DEVUELVE LA FECHA Y HORA ACTUAL (DEL SISTEMA).

El conjunto de tipos de datos en MySQL es el siguiente:

NUMÉRICOS ENTEROS:

BIT	ENTEROS CON VALOR 1 Ó 0.
INT	ENTEROS (NÚMEROS ENTEROS)
SMALLINT, TINYINT	ENTEROS PEQUEÑOS.
MEDIUMINT, BIGINT	ENTEROS MEDIANO Y GRANDE.

NUMÉRICOS REALES:

DECIMAL, NUMERIC	REALES DE PRECISIÓN Y ESCALA NUMÉRICA FIJA.
FLOAT, REAL	NÚMEROS CON PRECISIÓN DE UNTO FLOTANTE.
DOUBLE	REALES LARGOS.

FECHA Y HORA:

DATETIME, TIMESTAMP	FECHA Y HORA.
DATE	SOLO FECHA (DIA, MES Y AÑO).
TIME	SOLO HORA (HORA. MINUTO Y SEGUNDO).
YEAR	SOLO AÑO DEL DATO.

CADENAS DE CARACTERES:

CHAR(N)	CADENA (CARACTERES) DE LONGITUD FIJA.
VARCHAR(N)	CADENA (CHARACTER) DE LONGITUD VARIABLE.
TEXT, TINYTEXT	CADENA TEXTO DE TAMAÑO VARIABLE.
MEDIUMTEXT, LONGTEXT	CADENA TEXTO DE TAMAÑO VARIABLE MEDIO Y LARGO.
BLOB, TINYBLOB	TIPO BLOB (BINARY LARGE OBJECT)
MEDUMBLOB, LONGBLOB	TIPO BLOB (BINARY LARGE OBJECT) MEDIO Y LARGO.
ENUM	PERMITE INTRODUCIR UNA LISTA DE POSIBLES VALORES.
SET	CONJUNTO LISTA DE POSIBLES VALORES (COMBINAR).

- **ELIMINAR UNA TABLA:**

  Quita la definición de tabla y todos los datos contenidos en la misma. Referencia por parte de otros objetos a la tabla que se elimina, se deben eliminar previamente. Se puede usar **IF EXIST** antes del nombre.

  SINTAXIS: **DROP TABLE** NOMBRETABLA;

  EJ. **USE** VENTAS;
  DROP TABLE IF EXISTS CONTRATO;

- **MODIFICAR LA TABLA:**

  Cambia la definición de la tabla al alterar, agregar o quitar atributos, cambiar tipos de datos, longitudes y deshabilitar o habilitar restricciones. Se puede usar **IF EXIST** antes del nombre.

  SINTAXIS: **ALTER TABLE** NOMBRETABLA;

  EJEMPLO QUE CAMBIA EL FORMATO DE ALMACENAMIENTO:

  USE VENTAS;
  ALTER TABLE IF EXISTS CONTRATO **ENGINE** = MyISAM;

  OTROS EJEMPLOS:

  - Eliminar un atributo de la tabla:

    ALTER TABLE NOMBRETABLA **DROP COLUMN** ATRIBUTO;

  - Eliminar varios atributos de la tabla**:**

    ALTER TABLE NOMBRETABLA **DROP COLUMN** ATRIBUTO1,
    DROP COLUMN ATRIBUTO2;

  - Eliminar una clave primaria y clave externa:

    ALTER TABLE NOMBRETABLA **DROP PRIMARY KEY** ;

    ALTER TABLE NOMBRETABLA **DROP FOREIGN KEY** ATRIBUTO;

  - Añadir un nuevo atributo al final de la tabla:

    ALTER TABLE NOMBRETABLA **ADD** ATRIBUTOX TIPODATO;

  - Añadir un nuevo atributo después de otra**:**

    ALTER TABLE NOMBRETABLA **ADD** ATRIBUTOX TIPODATO **AFTER** ATRIBUTOX-1;

  - Añadir un nuevo atributo en la primera posición de la tabla:

    ALTER TABLE NOMBRETABLA **ADD** ATRIBUTOX TIPODATO **FIRST**;

  - Cambiar el tipo de dato de un atributo**:**

    ALTER TABLE NOMBRETABLA **CHANGE** ATRIBUTOX-1 ATRIBUTOX;

  - Cambiar el nombre o renombrar un atributo**:**

    ALTER TABLE NOMBRETABLA **CHANGE** ATRIBUTOX ATRIBUTOY ;

  - Asignar como clave primaria a un determinado atributo:

    ALTER TABLE NOMBRETABLA **ADD PRIMARY KEY**(ATRIBUTOX);

  - Modificar el valor de un atributo con propiedad auto_increment**:**

    ALTER TABLE NOMBRETABLA **AUTO_INCREMENT**=15000;

3.- COMANDOS PARA LOS INDICES:

- **CREACION DE INDICES:**

  La definición de un índice sobre un determinado atributo o sobre varios atributos de una tabla, puede ser de dos tipos: Ordinario o de Texto completo.

  SINTAXIS:

  Índice Ordinario:

  CREATE INDEX NOMBREINDICE **ON** NOMBRETABLA(ATRIBUTO);

  Índice Texto completo:

  CREATE FULLTEXT INDEX NOMBREINDICE **ON** NOMBRETABLA(ATRIBUTO);

- **ELIMINACION DE INDICES:**

  Quita la definición del índice asociado a uno o más atributos de la tabla.

  SINTAXIS:

  DROP INDEX NOMBRETABLA.NOMBREINDICE;

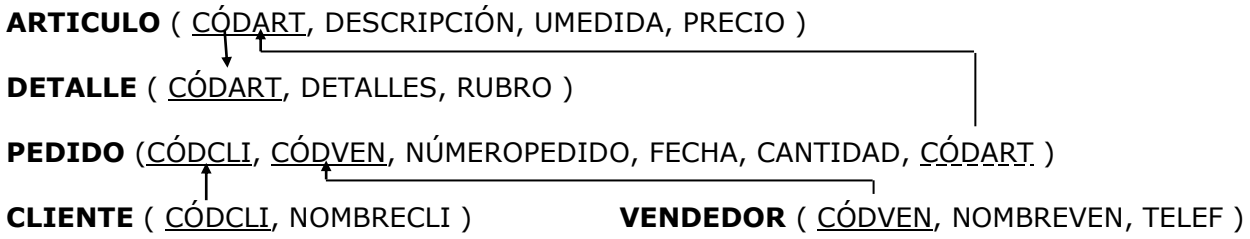
  Añadir un índice a un atributo y eliminar un índice, modificando la tabla**:**

  ALTER TABLE NOMBRETABLA **ADD INDEX** (ATRIBUTO);

  ALTER TABLE NOMBRETABLA **DROP INDEX** NOMBREINDICE;

EJEMPLO:

Crear la base de datos **Ventas** y las tablas y relaciones del MLD. Así mismo las validaciones para valores positivos de la cantidad y precio, código de articulo auto numerado a partir de 100001 con incremento del 1 y la fecha por defecto la actual.



- Creación de la base de datos:

```
CREATE DATABASE VENTAS
    CHARACTER SET utf8 COLLATE utf8_general_ci;
```

- Definición de las tablas y relaciones comenzando por las tablas independientes:

```
USE VENTAS;
CREATE TABLE ARTICULO
(   CODART INT(6) AUTO_INCREMENT PRIMARY KEY,
    DESCRIPCION VARCHAR(35) DEFAULT ' ',
    UMEDIDA ENUM ('KGR.', 'LTS.', 'PZA.', 'MTS.', 'BLS.', 'CJA.', 'SBR.'),
    PRECIO FLOAT(6,2) DEFAULT 0.0,  CHECK (PRECIO > 0.0)
)   ENGINE = InnoDB AUTO_INCREMENT = 100001 DEFAULT CHARSET = utf8;
```

```
CREATE TABLE CLIENTE
(   CODCLI CHAR(6) NOT NULL PRIMARY KEY,
    NOMBRECLI VARCHAR(35)  DEFAULT ' '
)   ENGINE = InnoDB DEFAULT CHARSET = utf8;

CREATE TABLE VENDEDOR
(   CODVEN CHAR(6) NOT NULL UNIQUE,
    NOMBREVEN VARCHAR(35) DEFAULT ' ',
    TELEF INT(7) DEFAULT 0,
    PRIMARY KEY (CODVEN)
 )  ENGINE = InnoDB  DEFAULT CHARSET = utf8;
```

R E G L A S

- **RELACION MUCHOS A MUCHOS:**

LLAVE PRIMARIA → LLAVE PRIMARIA

(TABLA ENTIDAD) (TABLA RELACION)

- **RELACION UNO A MUCHOS:**

LLAVE PRIMARIA ← LLAVE FORANEA

(TABLA INDEPED.) (TABLA DEPEND.)

- **REL. ENTIDAD A ENTIDAD DEBIL:**

LLAVE PRIMARIA → LLAVE PRIMARIA

(TABLA ENTIDAD) (ENTIDAD DEBIL)

- A continuación las tablas dependientes:

```
CREATE TABLE DETALLE
(   CODART  INT(6) NOT NULL PRIMARY KEY,
    DETALLES VARCHAR(30) DEFAULT ' ',
    RUBRO VARCHAR(15) DEFAULT ' ',
    FOREIGN KEY (CODART) REFERENCES ARTICULO(CODART)
)   ENGINE = InnoDB  DEFAULT CHARSET = utf8;

CREATE TABLE PEDIDO
(   CODCLI CHAR(6) NOT NULL,
    CODVEN CHAR(6) NOT NULL,
    NUMEROPEDIDO INT(8) DEFAULT 0,
    FECHA DATETIME DEFAULT NOW(),
    CANTIDAD INT(8), CHECK (CANTIDAD > 0),
    CODART  INT(6),
    PRIMARY KEY (CODCLI, CODVEN),
    FOREIGN KEY (CODCLI) REFERENCES CLIENTE(CODCLI),
    FOREIGN KEY (CODVEN) REFERENCES VENDEDOR(CODVEN),
    FOREIGN KEY (CODART) REFERENCES ARTICULO(CODART)
)   ENGINE = InnoDB DEFAULT CHARSET = utf8;
```

LENGUAJE DE MANIPULACION DE DATOS (LMD)

SQL tiene tres sentencias básicas que actúan sobre las tuplas del objeto tabla y hacen posible la manipulación de datos: INSERT (Insertar), UPDATE (Actualizar), DELETE (Eliminar) y SELECT (Seleccionar).

INSERT
UPDATE
DELETE
SELECT } **OBJETO** (TABLE)

La sintaxis de estos comandos varía de acuerdo al SGBD que se utilice. Para el caso de **MySQL**, se tienen la siguiente sintaxis:

- **INSERTAR TUPLAS:**

  Adicionar Tuplas a una tabla definida previamente que puede ser recientemente creada (sin Tuplas) o una que ya contenga datos. Omitir la lista de atributos, equivale a especificar una lista de todos los atributos de la tabla.

SINTAXIS:

INSERT INTO NOMBRETABLA (LISTA ATRIBUTOS)
VALUES (LISTA VALORES);

EJEMPLOS:

- Adición de una Tupla a la tabla artículo, especificando los atributos:

  INSERT INTO ARTICULO (CODART, DESCRIPCIÓN, UMEDIDA, PRECIO)

  VALUES ('A-0016', 'CEMENTO PORTLAND', 'BLS', 55.5);

- Adición de una Tupla a la tabla artículo, si la lista de valores es completa, se puede omitir la lista de atributos:

  INSERT INTO ARTICULO **VALUES** ('A-0016', 'CEMENTO PORTLAND', 'BLS', 55.5) ;

- Adición de más de Tupla en la tabla artículo :

  INSERT INTO ARTICULO (CODART, DESCRIPCIÓN, UMEDIDA, PRECIO)

  VALUES ('A-0017', 'ARENA FINA', 'BLS',15.0), ('A-0018', 'YESO', 'BLS', 10.5);

- Adición de una Tupla a la tabla artículo, para CODART INT(n) auto_increment:

  INSERT INTO ARTICULO **VALUES** (, 'CEMENTO PORTLAND', 'BLS', 55.5);

- Adición de una Tupla a la tabla artículo, sin valor de la unidad de medida, el sistema le asigna **Null** (si no especifico DEFAULT ' ' en la creación de la tabla):

  INSERT INTO ARTICULO (CODART, DESCRIPCIÓN, UMEDIDA, PRECIO)

  VALUES ('A-0016', 'CEMENTO PORTLAND', ' ', 55.5);

- Insertar a una TABLA1 varias Tuplas procedentes de otra TABLA2 Tuplas que cumplen una determinada condición, especificada en una selección:

  INSERT INTO NOMBRETABLA1 (LISTA ATRIBUTOS) **SELECT** LISTA ATRIBUTOS

  FROM NOMBRETABLA2 **WHERE** CONDICIÓN;

- La sentencia **REPLACE INTO** tiene una funcionalidad similar al INSERT pero al insertar evita la duplicidad de claves únicas. Por ejemplo, insertar una Tupla en la tabla artículo siempre y cuando el CODART no se repita.

  REPLACE INTO ARTICULO (CODART, DESCRIPCIÓN, UMEDIDA, PRECIO)

  VALUES ('A-0016', 'CEMENTO COLA', 'BLS', 25.0);

- **ACTUALIZAR TUPLAS:**

Permite hacer modificaciones a los atributos de Tuplas especificadas (una o varias) de una determinada tabla.

SINTAXIS:

UPDATE NOMBRETABLA **SET** ATRIBUTO1 = VALOR1, ATRIBUTO2 = VALOR2, ……

WHERE CONDICIÓN;

Dónde:

- ✓ La cláusula **SET** permite especificar los atributos que reciben nuevos valores.
- ✓ La cláusula **WHERE** permite indicar la condición, que deben cumplir las Tuplas cuyos valores se modifican (subconjunto de Tuplas que cumplan la condición).

EJEMPLOS:

- En la tabla vendedor modificar el teléfono del Sr. Julio Ortiz a 57321:

  UPDATE VENDEDOR **SET** TELEF = 57321 **WHERE** NOMBRE = 'JULIO ORTIZ';

- En la tabla vendedor modificar para el código de vendedor igual a 1020453, el nombre y teléfono:

  UPDATE VENDEDOR **SET** NOMBRE = 'JULIO JAVIER ORTIZ', TELEF = 57213
  WHERE CODVEN = '1020453';

- En la tabla vendedor cambiar el primer digito de los teléfonos de 4 a 5:

  UPDATE VENDEDOR **SET** TELEF = TELEF + 10000
  WHERE (TELEF > = 40000) **AND (**TELEF < 50000);

- En la tabla vendedor cambiar el descuento y el salario del vendedor, solo a los que tienen un salario superior a 5000 Bs.:

  UPDATE VENDEDOR **SET** DESCUENTO = SALARIO*0.30, SALARIO = SALARIO*0.70
  WHERE (SALARIO > 5000);

- **ELIMINAR TUPLAS:**

Se eliminan todas las Tuplas que cumplan la condición especificada en la cláusula WHERE, correspondientes a una determinada tabla. En ausencia de la cláusula WHERE, se eliminan todas las Tuplas

SINTAXIS: **DELETE FROM** NOMBRETABLA **WHERE** CONDICIÓN;

EJEMPLOS:

- Elimina al cliente cuyo CI es igual a 1020453:

  DELETE FROM CLIENTE **WHERE** CI = '1020453'; {ELIMINA UNA SOLA TUPLA}

- Elimina los clientes cuyo sexo es Masculino (M):

  DELETE FROM CLIENTE **WHERE** SEXO = 'M'; {ELIMINA VARIAS TUPLAS}

- Elimina todas las Tuplas de la tabla cliente:

  DELETE FROM CLIENTE;

- Elimina todos los clientes cuya edad esta entre 15 y 55 años:

  DELETE FROM CLIENTE **WHERE** EDAD >= 15 AND EDAD <= 55;

- Elimina todas las Tuplas de la tabla cliente, en un máximo de 15:

  DELETE FROM CLIENTE **LIMIT** = 15;

- Elimina todos los clientes en los que la dirección no está especificada.

  DELETE FROM CLIENTE **WHERE** DIRECCIÓN **IS NULL** OR DIRECCIÓN = ' ';

- Elimina todas las Tuplas de la tabla cliente, cuyo nombre no sea nulo.

  DELETE FROM CLEINTE **WHERE** NOMBRECLI **IS NOT NULL** OR NOMBRECLI <> ' ';

- **SELECCIONAR TUPLAS (CONSULTAS):**

Todas las consultas o recuperación de datos en SQL, son especificadas mediante el comando SELECT (seleccionar). Mostrando los datos en una CUADRICULA.

SINTAXIS Y ORDEN DE EJECUCION:

SELECT LISTA ATRIBUTOS ..... **[7]**
FROM LISTA DE TABLAS ..... **[1]**
WHERE CONDICIONES TUPLAS ..... **[2]**
ORDER BY ATRIBUTO(S) **ASC** / **DESC** ..... **[4]**
GROUP BY ATRIBUTO(S) **HAVING** CONDICIONES GRUPOS
..... **[3]** ..... **[5]**
LIMIT n; ..... **[8]**

Dónde:

- ✓ **DISTINCT** o **DISTINCTROW,** son clausulas opcionales que especifican que las Tuplas duplicadas no se tomen en cuenta.
- ✓ Cláusula **FROM**, indica tablas o relaciones que se utilizan en la evaluación de la consulta.
- ✓ Clausula **WHERE**, opcional que especifica la condición o las condiciones que deben cumplir las Tuplas seleccionadas. Si no se incluye esta cláusula, significa que se seleccionan todas las Tuplas.
- ✓ La "Lista de atributos", especifica los atributos de una o más tablas que tienen que ser proyectadas en la relación resultante. El símbolo asterisco (*) o ALL, denota que todos los atributos de la relación resultante tienen que ser proyectados.
- ✓ La "Condición", puede ser simple o compuesta, conformada por más de dos condiciones simples, que contienen las conectivas lógicas y de relación:

  =, <>, >, <, >=, <=, NOT, AND y OR

- ✓ La Cláusula **LIMIT**, indica el número de Tuplas devueltas. Por ejemplo si n = 5 devuelve 5 primeras Tuplas, si n =5,10 devuelve de 6 a la 15 Tuplas.

Las consultas de selección se utilizan para indicar al SGBD que devuelva información de las bases de datos, esta información es devuelta en forma de conjunto de Tuplas.

1. SELECCION Y PROYECCION:

Se eligen los atributos (proyección) y las Tuplas (selección) que se mostraran en la cuadricula**.**

- Dada la tabla persona, realizar una consulta que muestre: a) Todos los atributos de la tabla sin duplicar, b) Los atributos CI y Nombrep de las primeras 10 personas c) Todas las personas cuyo sexo es Masculino y tengan una edad mayor a 21 años.

PERSONA (CI, NOMBREP, EDAD, SEXO, DIRECCIÓN, TELF)

a) **SELECT** DISTINT ALL **FROM** PERSONA;

ALIAS DE LA TABLA (NOMBRE CORTO)

b) **SELECT** P.CI, P. NOMBREP **FROM** PERSONA *AS* P **LIMIT** 10;

c) **SELECT** * **FROM** PERSONA P **WHERE** (P.SEXO = 'M') **AND (**P.EDAD > 21);

- Sea la tabla inscripción, se desea realizar una consulta para obtener los números, nombres de alumnos con notas de calificación mayor que el alumno con número 31482 inscrito en el curso FIS200.

INSCRIPCION (NUMERO, NOMBRE, CURSO, EDAD, DIRECCIÓN, TELEF, NOTA)

SELECT A.NÚMERO, A.NOMBRE, A.NOTA **FROM** INSCRIPCIÓN A, INSCRIPCIÓN B
WHERE (A.NOTA > B.NOTA) **AND** (B.NUMERO = 31482) **AND** (B.CURSO = 'FIS200');

2. ORDENACION Y AGRUPACION:

Se realiza en la sentencia **SELECT**, por medio de las clausulas: **ORDER BY** (Ordenar por), **GROUP BY** (Agrupar por) y **HAVING** (condición de grupo).

- ✓ **ORDER BY:** Permite la clasificación u ordenamiento de la relación resultante al aplicar SELECT según un determinado atributo en forma ascendente (**ASC**) o descendente (**DESC**). Se puede escoger más de un atributo para la clasificación. Si se utiliza **ORDER BY RAND**(), ordena al azar por la llave primaria.

  SINTAXIS: **SELECT** LISTA ATRIBUTOS **FROM** LISTA TABLAS **WHERE** CONDICIÓN
  ORDER BY ATR1 **ASC / DESC**, ATR2 **ASC / DESC**, ……. ;

- ✓ **GROUP BY:** Permite agrupar Tuplas en forma lógica (no física), que tengan el mismo valor de un atributo especificado

  SINTAXIS: **SELECT** LISTA ATRIBUTOS **FROM** LISTA TABLAS **WHERE** CONDICIÓN
  GROUP BY ATR1, ATR2, …… ;

- ✓ **HAVING:** Permite identificar el subconjunto de grupos que se mostraran en la cuadricula. Es decir HAVING es a los grupos lo que WHERE es a las Tuplas. Por tanto, especifica los grupos que cumplen una determinada condición.

  SINTAXIS:

  SELECT LISTA ATRIBUTOS **FROM** LISTA TABLAS **WHERE** CONDICIÓN TUPLAS
  GROUP BY ATRIBUTOS DE AGRUPACIÓN **HAVING** CONDICIÓN GRUPOS;

EJEMPLOS:

- Consulta sobre la tabla persona, que ordena alfabéticamente a todos las personas de sexo masculino.

  SELECT DISTINCTROW * **FROM** PERSONA **WHERE** SEXO = 'M'
  ORDER BY NOMBRE **ASC** ;

- Consulta sobre la tabla persona, que agrupa en hombres y mujeres a las personas registradas con una edad que no está comprendida entre 30 y 40 años inclusive, ordenando su ci en forma decreciente.

  SELECT CI, NOMBREP, SEXO, EDAD **FROM** PERSONA
  WHERE (EDAD <= 30) **OR** (EDAD >= 40)
  GROUP BY SEXO
  ORDER BY CI **DESC** ;

- Consulta sobre la tabla inscripción, agrupando los alumnos por cursos en orden alfabético, tomando en cuenta los que tienen una edad comprendida entre 15 y 45 años. Considerando en la consulta los cursos que tengan más de 20 alumnos.

```
SELECT * FROM INSCRIPCION
WHERE EDAD BETWEEN 15 AND 45
GROUP BY CURSO HAVING COUNT(*) > 20
ORDER BY NOMBRE ASC ;
```

3. FUNCIONES DE AGREGADO:

Las funciones de agregado se usan dentro de una cláusula SELECT en grupos de tuplas para devolver un único valor que se aplica a dicho grupo. SQL ofrece una serie de funciones especiales para ampliar su capacidad básica de recuperación:

SINTAXIS:
```
SELECT ATRIBUTO1, FUNCIÓN (ATRIBUTO2)
FROM LISTA TABLAS
WHERE CONDICIÓN
GROUP BY ATRIBUTO1;
```

FUNCIÓN	DESCRIPCIÓN
AVG	PROMEDIO
COUNT	NÚMERO DE TUPLAS
SUM	SUMA
MAX	VALOR MÁS ALTO
MIN	VALOR MÁS BAJO
CONCAT	CONCATENAR.

Para las funciones SUM, AVG, MAX Y MIN, el atributo debe ser de tipo numérico y en general es necesario usar la cláusula DISTINCT para no considerar los valores repetidos.

Para la función COUNT el atributo puede ser de cualquier tipo.

La función CONCAT, solo se aplica a atributos de tipo carácter o cadena.

EJEMPLOS:

- Consulta sobre la tabla persona, que cuenta el número de registro o Tuplas.

```
SELECT COUNT(*) FROM PERSONA;
```

- Consulta sobre la tabla persona, que cuenta el número de personas de sexo Femenino (F), sin repetición

```
SELECT COUNT(DISTINCT NOMBRE) FROM PERSONA WHERE SEXO = 'F';
```

- Consulta en la tabla inscripción, que permite ajustar las calificaciones incrementando 20% a los alumnos del curso FIS200 que tienen una nota menor a 51, visualizando el nombre y la nueva nota, utilizando etiquetas.

```
SELECT NOMBRE AS ALUMNO, NOTA * 1.2 AS CALIFICACION
FROM INSCRIPCION
WHERE (NOTA < 51) AND (CURSO = 'FIS200');
```

- Consulta en la tabla Personal, que obtenga la suma de los salarios de todos los mayores a la edad mínima de los mismos.

```
SELECT SUM(SALARIO) AS TOTALSALARIOS FROM PERSONAL
WHERE EDAD > ( SELECT MIN(EDAD) FROM PERSONAL);
```

- Consulta sobre la tabla inscripción, que muestre el número de alumnos con una nota mayor al promedio del curso FIS200.

```
                  SELECT CURSO, COUNT(NUMERO) FROM INSCRIPCION
DEBE              WHERE NOTA > ( SELECT AVG(NOTA) FROM INSCRIPCION
COINCIDIR                          WHERE CURSO = 'FIS200'  )
                  GROUP BY CURSO;
```

- Consulta en la tabla Personal, el nombre completo y la edad valida comprendida entre la edad promedio y la edad máxima inclusive.

```
SELECT CONCAT (NOMBRE, '  ', APELLIDO) AS NOMBRECOMPLETO,
               EDAD AS EDADVALIDA
FROM PERSONAL
WHERE EDAD >= ( SELECT AVG(EDAD) FROM PERSONAL) AND
      EDAD <= ( SELECT MAX(EDAD) FROM PERSONAL);
```

4. UTILIZACION DE BETWEEN, IN Y LIKE:

- **OPERADOR BETWEEN.**

Para indicar que deseamos recuperar Tuplas según el intervalo de valores de un atributo se utiliza el operador BETWEEN.

SINTAXIS: ATRIBUTO O EXPRESION **BETWEEN** VALOR1 **AND** VALOR2

En este caso la consulta devolvería Tuplas que contengan en atributo un valor incluido en el intervalo valor1, valor2 (ambos inclusive). Si se antepone NOT se devuelven aquellos valores no incluidos en el intervalo.

ATRIBUTO O EXPRESION **NOT BETWEEN** VALOR1 **AND** VALOR2

EJEMPLOS:

SELECT * **FROM** PERSONAL **WHERE** EDAD BETWEEN 25 **AND** 50;

También: **SELECT** * **FROM** PERSONAL **WHERE** EDAD >= 25 AND EDAD =< 50;

SELECT * **FROM** PEDIDO **WHERE** CANTIDAD **BETWEEN** 28000 **AND** 28999;

SELECT * **FROM** PERSONAL **WHERE** SUELDO **BETWEEN** 500 **AND** 1000;

- **OPERADOR IN.**

Este operador devuelve aquellas Tuplas cuyo atributo indicado coincide con alguno de los en una lista.

Su sintaxis es:

SINTAXIS: ATRIBUTO O EXPRESION **IN** (VALOR1, VALOR2, …..)

También: ATRIBUTO O EXPRESION **NOT IN** (VALOR1, VALOR2, …..)

EJEMPLOS:

SELECT * **FROM** PERSONAL **WHERE** SEXO **IN** ('F', 'M');

También: **SELECT** * **FROM** PERSONAL **WHERE** SEXO = 'F' **OR** SEXO = 'M' ;

SELECT * **FROM** PEDIDO **WHERE** CIUDAD **IN** ('SUCRE', 'POTOSI', 'TARIJA');

SELECT * **FROM** PERSONAL **WHERE** NOMBRE **IN** ('NICOLAS','GUSTAVO');

SELECT * **FROM** PERSONAL **WHERE** NOMBRE **NOT IN** ('PABLO','GUSTAVO');

- **OPERADOR LIKE.**

LIKE permite la búsqueda de patrones de coincidencia en cadenas de caracteres. Se aplica a atributos de tipo cadena y permite comodines en las consultas.

SINTAXIS: ATRIBUTO O EXPRESION **LIKE** PATRON

También: ATRIBUTO O EXPRESION **NOT LIKE** PATRON

COMODINES	
_	SUSTITUYE A CUALQUIER CARACTER.
%	SUSTITUYE UNA CADENA DE CARACTERES.
[ ]	SUSTITUYE A UN RANGO ESPECIFICADO DE CARACTERES

La utilización de los comodines, se ejemplifica a continuación:

ATRIBUTO **LIKE** 'B**%**' CADENAS QUE COMIENZAN CON B.

ATRIBUTO **LIKE** '**%**E**%**' CADENAS QUE CONTIENEN UNA E.

ATRIBUTO **LIKE** '**%**B' CADENAS QUE TERMINAN EN B.

ATRIBUTO **LIKE** '**_ _** S**%**' CADENAS QUE TIENEN UNA **S** COMO TERCERA LETRA.

EJEMPLOS:

SELECT * **FROM** PERSONA **WHERE** SEXO = 'F' **AND** NOMBRE **LIKE** 'R**%**' ;

SELECT COUNT(*) **FROM** PERSONA **WHERE** CI **LIKE** '1**%**';

SELECT * **FROM** PERSONA **WHERE** NOMBRE **LIKE** '**_**A**%**' **OR**
NOMBRE **LIKE** '**_** E**%**' **OR** NOMBRE **LIKE** '**_** I**%**' **OR** NOMBRE **LIKE** '**_** O**%**' **OR**
NOMBRE **LIKE** '**_**U**%**' ;

Otra forma: **SELECT** * **FROM** PERSONA **WHERE** NOMBRE **LIKE** '**_** [A,E,I,O,U]% ' ;

Mayusculas y minusculas:

SELECT * **FROM** PERSONA **WHERE** NOMBRE **LIKE** '**_** [A,E,I,O,U,a,e,i,o,u]% '

Otra forma: **SELECT** * **FROM** PERSONA **WHERE** NOMBRE **LIKE** '**_** [A,E,I,O,U]% '
OR NOMBRE **LIKE** '**_** [a,e,i,o,u]% ';

SELECT * **FROM** PERSONA **WHERE** CI **LIKE** ' [1,2,3,4,5]-% '

Otra forma: **SELECT** * **FROM** PERSONA **WHERE** CI **LIKE** ' [1-5]-% ';

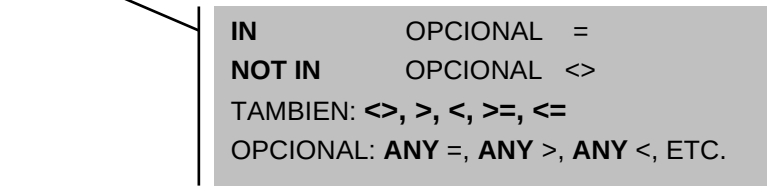
5. CONSULTAS EN VARIAS TABLAS:

Se trata de consultas realizadas sobre dos o más tablas relacionadas o no.

- **SUBCONSULTAS:**

  Una subconsulta, es una **expresión SELECT** anidada dentro de otra expresión del mismo tipo. Son posibles muchos niveles. Las subconsultas se utilizan para representar el conjunto de valores en el cual se realiza una búsqueda mediante una condición **IN** (en) o el término equivalente **ANY**, que se puede utilizar en combinación con operadores lógicos y relacionales.

  SINTAXIS: **SELECT** LISTA ATRIBUTOS **FROM** LISTA TABLAS 1
  WHERE ATRIBUTO1 **=** (**SELECT** ATRIBUTO2 **FROM** LISTA TABLAS 2
  WHERE CONDICIÓN);

  La forma de leer y procesar los SELECT anidados, es de adentro hacia afuera o de abajo hacia arriba.

  EJEMPLOS:

  - Consulta en la tabla inscripción, que obtiene todas las Tuplas con la calificación mayor a la calificación más alta.

    SELECT * **FROM** INSCRIPCIÓN
    WHERE NOTA > (**SELECT MAX**(NOTA) **FROM** INSCRIPCIÓN);

  - En la tabla inscripción, consulta que obtiene:

    Las Tuplas de los alumnos que pertenecen al curso FIS200

    SELECT DISTINCT * **FROM** INSCRIPCIÓN
    WHERE NÚMERO **=** (**SELECT** NÚMERO **FROM** INSCRIPCIÓN
    WHERE CURSO = 'FIS200');

    Las Tuplas de los alumnos que no pertenecen a dicho curso.

    SELECT * **FROM** INSCRIPCIÓN
    WHERE NÚMERO <> (**SELECT** NÚMERO **FROM** INSCRIPCIÓN
    WHERE CURSO = 'FIS200');

  - Las Tuplas de los alumnos que tengan las notas entre la promedio y la nota más baja del curso FIS200.

    SELECT NOMBRE, CURSO, NOTA **FROM** INSCRIPCIÓN
    WHERE NOTA **>** (**SELECT** MIN(NOTA) **FROM** INSCRIPCIÓN **WHERE** CURSO = 'FIS200')
    AND NOTA **<** (**SELECT** AVG(NOTA) **FROM** INSCRIPCIÓN **WHERE** CURSO = 'FIS200');

- **UNIONES:**

Llamada también Unión externa, permite realizar la unión de los atributos de dos o más tablas que no están relacionadas. Esta operación por lo general se realiza junto a operaciones de proyección.

SINTAXIS:

SELECT ATRIBUTOS1 **FROM** TABLA1 **UNION SELECT** ATRIBUTOS2 **FROM** TABLA2;

También: **SELECT** ATRIBUTOS1, ATRIBUTOS2 **FROM** TABLA1, TABLA2;

EJEMPLO:

Consulta que realiza la unión de atributos de las tablas personas y estudiantes:

SELECT * **FROM** PERSONA **UNION SELECT** * **FROM** ESTUDIANTE;

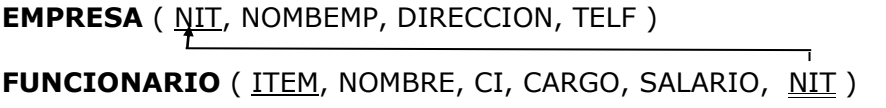
También: **SELECT** * **FROM** PERSONA, ESTUDIANTE;

- **UNION RELACIONAL:**

Llamada también unión interna, se realiza la combinación de dos o más tablas relacionadas a través de atributos comunes (llaves foráneas y llaves primarias). Se realiza la unión de cada Tupla de una de las tablas con todas las demás de la otra cuyos atributos coincidan, es decir, sus llaves foráneas y primarias estén relacionadas. Se realiza por medio de la búsqueda en la cláusula WHERE de atributos comunes (iguales) o se puede utilizar el operador **INNER JOIN** en la cláusula FROM.

EJEMPLOS:

- Dadas las siguientes tablas. Se tiene los siguientes ejemplos:



a) Consulta que realiza la unión relacional de las tablas empresa y funcionario.

SELECT * **FROM** EMPRESA *AS* E, FUNCIONARIO *AS* F **WHERE** E.NIT = F.NIT;

Utilizando el operador **INNER JOIN**:

SELECT * **FROM** EMPRESA *AS* E **INNER JOIN** FUNCIONARIO *AS* F **ON** E.NIT = F.NIT;

b) Consulta que muestra los funcionarios de la empresa AMERICA.

SELECT * **FROM** FUNCIONARIO F **INNER JOIN** EMPRESA E **ON** E.NIT = F.NIT;
WHERE E.NOMBEMP = 'AMERICA';

c) Consulta que muestra el nombre de la empresa en un grupo de nombres de funcionarios y sus correspondientes salarios.

SELECT E.NOMBEMP, F.NOMBRE, F.SALARIO **FROM** EMPRESA E, FUNCIONARIO F
WHERE E.NIT = F.NIT
GROUP BY E.NOMBEMP, F.NOMBRE, F.SALARIO;

- Dadas las siguientes tablas y sus relaciones, se tienen los siguientes ejemplos:

EMPRESA (NIT, NOMBRE, DIRECCIÓN, TELEF, PROPIETARIO)

COMPRA (NIT, CÓDIGO, CANTIDAD)

MATERIAL (CÓDIGO, DESCRIPCIÓN, UNIDAD, PRECIO)

a) Consulta que realiza la unión relacional de las tres tablas.

```
SELECT * FROM COMPRA AS C INNER JOIN EMPRESA AS E  ON C.NIT = E.NIT
                    INNER JOIN MATERIAL AS M  ON C.CODIGO = M.CODIGO;
```

b) Muestra el nombre de la empresa, su propietario, el material comprado, descripción, cantidad total y costo total, agrupando por empresa. Listando las compras con más de 5 Items, el nombre de propietarios de la **A** hasta **G**.

```
SELECT E.NOMBRE, E.PROPIETARIO,
        SUM(C.CANTIDAD) AS CANTIDADTOTAL,
        SUM(C.CANTIDAD * M.PRECIO) AS COSTOTOTAL
FROM COMPRA C INNER JOIN EMPRESA E ON C.NIT = E.NIT
        INNER JOIN MATERIAL M ON C.CODIGO = M.CODIGO
WHERE (E.PROPIETARIO LIKE '[A-G] %')
GROUP BY E.NOMBRE, E.PROPIETARIO HAVING COUNT(M.CÓDIGO) > 5;
```