INSTITUTO SUPERIOR TECNOLÓGICO TENA
Tecnología, Innovación y Desarrollo

DESARROLLO DE SOFTWARE

Instrumento para facilitar el proceso de enseñanza-aprendizaje de la asignatura

GUÍA GENERAL DE ESTUDIO DE LA ASIGNATURA 20230020

BASE DE DATOS AVANZADA

Período académico Cuarto

Agosto - 2023

Ing. Juan M. Espín Montesdeoca., Mg

GUIA GENERAL DE ESTUDIO DE LA ASIGNATURA – BASE DE DATOS AVANZADA

INSTITUTO SUPERIOR TECNOLÓGICO TENA

Carrera de Tecnología Superior en Desarrollo de Software

ISTT DS Primera Edición – Tena, agosto 2023

SIN ISBN

Instituto Superior Tecnológico Tena
Km. 1 1/2 Vía Tena - Archidona
Tena, Ecuador

AUTORES - REDACCIÓN Y FORMULACIÓN DE CONTENIDOS
Ing. Juan M. Espín Montesdeoca, Mg.
Ing. Patricio Guanipatín Ramírez
Profesores del Instituto Superior Tecnológico Tena

REVISIÓN DE PARES
Lcdo. Segundo Calisto Rochina Chileno
Mg. Alvaro Santiago Toalombo Díaz
Mg. Henry Fabian Chango Chango
Ing. Agustin Gonzalo Guanipatin Ramirez

Comisión de revisión técnica de guías de estudio del Instituto Superior Tecnológico Tena

APROBACIÓN

Mg. Danilo Alexander Zamora Núñez
Coordinador de Investigación, Desarrollo Tecnológico e Innovación

Impreso y hecho en Ecuador.

TABLA DE CONTENIDOS

ÍNDICE

secretaria.general@itstena.edu.ec km 1 1/2 vía Tena- Archidona

GUIA GENERAL DE ESTUDIO DE LA ASIGNATURA

DATOS GENERALES DE LA ASIGNATURA							
Carrera	Tecnólogo Superior en Desarrollo de Software			**Nombre asignatura**		Base de Datos Avanzada	
Modalidad	Presencial			**Campo de Formación**		Adaptación e Innovación Tecnológica	
Jornada	Nocturna			**Unidad de Organización Curricular**		Formación Profesional	
Período académico	Tercero			**Código de la asignatura**		DS316	
Ciclo académico	2023 IIS			**N° Total de horas de la asignatura**		**168**	
Distribución de horas en las actividades de aprendizaje							
N° de horas Docencia	**72**	**N° de horas Aprendizaje Práctico Experimental**				**N° de horas Autónomo**	**60**
		En contacto con docente	**8**	**Autónomo**	**28**		

PRERREQUISITOS Y CORREQUISITOS			
Prerrequisitos de la asignatura		**Correquisitos de la asignatura**	
Asignatura	**Código**	**Asignatura**	**Código**
Base de Datos	DS208		

DESCRIPCIÓN DE LA ASIGNATURA

La asignatura de Base de Datos Avanzada complementa el principio fundamental del funcionamiento de los sistemas informáticos, esta es parte de la Unidad de Organización Curricular Profesional donde el Tecnólogo Superior en Desarrollo de Software podrá crear script de bases de datos utilizando sentencias DDL y DML, identificadas con sus índices, triggers, procedimientos de vistas, aplicación y creación de funciones, trabajos con índices cursores y transacciones y finalizando con una revisión de sobre la seguridad requerida para la utilización de las bases de datos.

OBJETIVO GENERAL

Utilizar herramientas CASE que permitan a los estudiantes Tecnólogo Superior en Desarrollo de Software realizar modelos prácticos de entidad-relación.

CONTRIBUCIÓN DE LOS RESULTADOS DE APRENDIZAJE DE LA ASIGNATURA AL PERFIL DE EGRESO DE LA CARRERA

Resultados de aprendizaje de la asignatura	Resultados de aprendizaje del perfil de egreso de la carrera	Contribución (alta – media – baja)
Analiza los problemas planteados y genera modelos relacionales para manejo del proceso.	• Aplica metodologías y técnicas de investigación en la búsqueda, fundamentación y elaboración de soluciones informáticas.	Alta
Desarrolla sentencias SQL para manejo y administración de la información. Relaciona la base de datos con el desarrollo de software.	• Elabora el modelamiento, ilustración y evaluación del proceso de base de datos.	Alta
Desarrolla scripts de creación de bases de datos. Describe el proceso mediante el esquema de administración de la base de datos.	• Elabora documentación técnica.	Media
	• Aplica técnicas en seguridades de base de datos	Media

CONTENIDOS DE LA ASIGNATURA (descripción mínima de contenidos de la asignatura)

UNIDAD 1: SENTENCIAS DDL Y DML
1.1 Sentencias DDL
1.2 Sentencias DML
1.3 Integridad referencial y restricciones

secretaria.general@itstena.edu.ec km 1 1/2 vía Tena- Archidona ISTT

1.4 Introducción a transacciones (relación con DML)
UNIDAD 2: IMPLEMENTACIÓN DE ÍNDICES 2.1 Implementación de Índices. 2.2 Actividades de las Unidades 1 y 2
UNIDAD 3: IMPLEMENTACIÓN DE TRIGGER, VISTAS, PROCEDIMIENTOS Y FUNCIONES 3.1. Implementación de Trigger 3.2. Creación de Vistas 3.3. Procedimientos, índices y cursores 3.4. Funciones
UNIDAD 4: TRANSACCIONES Y SEGURIDAD 4.1 Transacciones 4.2 Seguridad

ESTRATEGIAS METODOLÓGICAS Y RECURSOS DIDÁCTICOS

ESTRATEGIAS METODOLÓGICAS	HABILIDADES BLANDAS	FINALIDAD
Activas para la enseñanza y aprendizaje	**Valores vinculados a la autonomía del sujeto**: confianza, crítica y autocrítica, honestidad, integridad	• Generar confianza/ Promover el pensamiento crítico. • Permite a los estudiantes cumplir un rol activo dentro de su formación. • Construye una sociedad participante.
Aprendizaje y trabajo cooperativo	**Valores elementales de convivencia y civilidad:** crítica y autocrítica, tolerancia, empatía, respeto, justicia, lealtad, paciencia	• Promover un ambiente de colaboración/ trabajo en equipo/ Saber escuchar/Promover el pensamiento crítico/ fomentar el liderazgo/ adaptabilidad. • Mantener una comunicación abierta con el equipo/ tolerancia a los errores, aceptar y aprender de las críticas. • Fomentar el sentido de pertenencia
Aprendizaje individual	**Valores vinculados a la autonomía del sujeto**: responsabilidad, honestidad, integridad, efectividad, autonomía	• Facilitar la asimilación del contenido por parte del estudiante/ Plantear preguntas para promover la comunicación efectiva /Promover el pensamiento crítico • Lectura comprensiva para fijar contenidos/ Promover el pensamiento crítico

RECURSOS DIDÁCTICOS	
MATERIALES CONVENCIONALES	*Material impreso: libros, folletos, fotocopias, periódicos, etc.*
	Tableros didácticos: pizarra
MATERIALES AUDIOVISUALES	*Imágenes fijas proyectables (fotos): diapositivas y fotografías.*
	Materiales audiovisuales (vídeo): películas y vídeos
NUEVAS TECNOLOGÍAS	*Programas informáticos: Moodle, MyAQL, PowerPoint, Hojas de Cálculo, Procesador de texto.*
	Servicios telemáticos: páginas web, plataforma EVA, correo electrónico, chats

secretaria.general@itstena.edu.ec km 1 1/2 vía Tena- Archidona ISTT

BIBLIOGRAFÍA		
Bibliografía Básica de la Asignatura:	**Físico**	**Digital**
Abraham, S., Henry, F. y Sudarshan, S. (2006). *Fundamentos de diseño de bases de datos.* (quinta edición). España. ISBN: 978-84-481-5671-8. Número de inventario en biblioteca: ISTT-DS-0097.	X	
Francisco, M. y Amalia, G. (2017). *Programación de base de datos relacionales.* (primera edición). Colombia. **ISBN:** 978-958-762-684-1, **Número de inventario en biblioteca:** ISTT-DS-0027.	X	
• Miguel, C., Mario, P. y Esperanza M. (2017). *Diseño de base de datos relacionales.* (primera edición). México, **ISBN:** 970-15-0526-3, **Número de inventario en biblioteca:** ISTT-DS-0031.	X	
Bibliografía de consulta de la Asignatura:	**Físico**	**Digital**
Antolín, M. (2017) *Administración básica de base de datos con ORACLE 12c.* España. **ISBN:** 9786076229897, 6076229896. https://www.google.com.ec/books/edition/Administraci%C3%B3n_b%C3%A1sica_de_bases_de_dato/DgdzEAAAQBAJ?hl=es-419&gbpv=1&dq=SQL+f%C3%A1cil&pg=PA52&printsec=frontcover		X
José, C. Gestión de Base de Datos. (segunda edición). España. Editorial RA-MA. **ISBN:** 978-84-9964-367-0. https://www.google.com.ec/books/edition/Gesti%C3%B3n_de_bases_de_datos_2%C2%AA_Edici%C3%B3n/dI-fDwAAQBAJ?hl=es-419&gbpv=1&dq=Gesti%C3%B3n+de+Base+de+datos+con+SQL&printsec=frontcover		X

DESCRIPTIVA DE LAS BASES DE DATOS AVANZADA

Este documento presenta una descripción detallada para desarrollar un conocimiento integral que combine diseño, administración y protección de bases de datos avanzadas, aplicando estos conceptos a problemas reales de la industria, tomado en cuenta el modelado avanzado de datos, optimización y rendimiento, consultas avanzadas en SQL, transacciones y concurrencia y seguridad en base de datos.

UNIDAD 1: SENTENCIAS DDL Y DML

- Diseñar y modificar esquemas de bases de datos mediante la creación, alteración y eliminación de tablas, columnas, y restricciones, asegurando la integridad referencial.
- Configurar adecuadamente restricciones como claves primarias, foráneas, y reglas de unicidad para garantizar la consistencia y el diseño óptimo de la base de datos.
- Construir y ejecutar consultas eficientes para insertar, actualizar, eliminar y recuperar información, optimizando el uso de recursos del sistema.
- Manejar transacciones utilizando las operaciones básicas de DML, asegurando el cumplimiento de las propiedades ACID en las modificaciones de datos.

UNIDAD 2: IMPLEMENTACIÓN DE ÍNDICES

- Diseñar y programar triggers que ejecuten acciones automáticas en respuesta a eventos específicos, como inserciones, actualizaciones o eliminaciones.
- Garantizar la correcta interacción de los triggers con la lógica de la base de datos, evitando bucles infinitos y conflictos con las restricciones.

UNIDAD 3: IMPLEMENTACIÓN DE TRIGGER

- Identificar los riesgos asociados con el mantenimiento de equipos y sistemas es esencial para implementar medidas preventivas adecuadas.
- Conocer los tipos de dispositivos de seguridad aeroportuaria y su sistema de control.

UNIDAD 4: CREACIÓN DE VISTAS

- Crear vistas que simplifiquen consultas complejas, proporcionando a los usuarios una interfaz de datos más accesible y personalizada.
- Utilizar vistas como herramientas de seguridad para restringir el acceso directo a ciertas columnas o datos sensibles, controlando qué información está disponible para cada usuario.

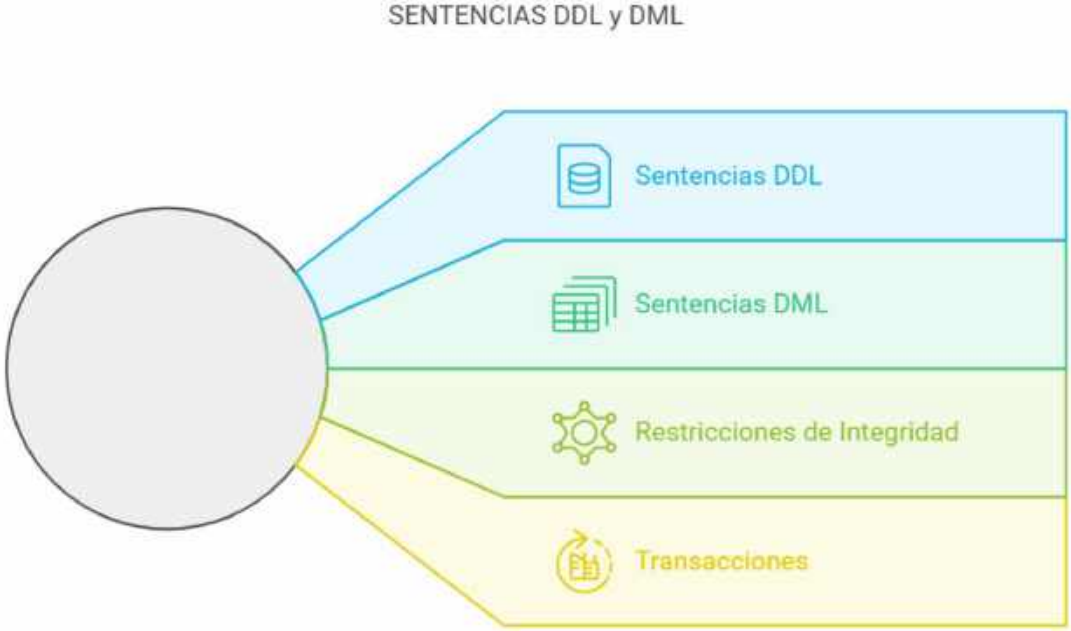
UNIDAD 1: SENTENCIAS DDL Y DML

1.1 Sentencias DDL
1.2 Sentencias DML
1.3 Integridad referencial y restricciones
1.4 Introducción a transacciones (relación con DML)

Resultado de Aprendizaje

El estudiante será capaz de crear y manipular estructuras de bases de datos relacionales, así como gestionar los datos contenidos en ellas, garantizando la integridad y consistencia del sistema.

DIAGRAMA DE APRENDIZAJE



SÍNTESIS

En esta unidad aprenderemos a crear, modificar y eliminar estructuras de bases de datos sentencias DDL) y a manipular los datos almacenados en ellas (sentencias DML). Exploraremos cómo definir tablas, columnas y restricciones para garantizar la integridad de los datos, así como las operaciones para insertar, actualizar, consultar y eliminar registros de manera eficiente.

Además, veremos cómo utilizar transacciones para gestionar cambios en los datos, asegurando que se mantengan consistentes y seguros. Este conocimiento nos permitirá trabajar de forma estructurada y optimizada con bases de datos relacionales, aplicando buenas prácticas en su diseño y manipulación.

UNIDAD 1: SENTENCIAS DDL Y DML

1.1. Sentencias DDL complejas

Arévalo (2013) afirma: Los DLL (Data Definition Language) que permiten crear y definir nuevas bases de datos, campos e índices. (pág. 1).

Tabla 1. Sentencias DDL

Comando	Descripción
CREATE	Utilizado para crear nuevas tablas, campos e índices
DROP	Empleado para eliminar tablas e índices
ALTER	Utilizado para modificar las tablas agregando campos o cambiando la definición de los campos.1

"El lenguaje de definición de datos (en inglés Data Definition Language, o DDL), es el que se encarga de la modificación de la estructura de los objetos de la base de datos.

Incluye órdenes para modificar, borrar o definir las tablas en las que se almacenan los datos de la base de datos. Existen cuatro operaciones básicas: CREATE, ALTER, DROP y TRUNCATE.

1.1.1. CREATE

Este comando crea un objeto dentro del gestor de base de datos. Puede ser una base de datos, tabla, índice, procedimiento almacenado o vista.". (Arévalo, 2013, pág. 2)

Ejemplo (crear una tabla):

```
CREATE TABLE Empleado
(
id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
Nombre VARCHAR(50),
Apellido VARCHAR(50),
Direccion VARCHAR(255),
Ciudad VARCHAR(60),
Telefono VARCHAR(15),
```

1 https://geotalleres.readthedocs.io/es/latest/conceptos-sql/conceptos_sql.html

secretaria.general@itstena.edu.ec km 1 1/2 vía Tena- Archidona ISTT

Edad INT
)

1.1.2. ALTER

Arévalo (2013) afirma: Este comando permite modificar la estructura de un objeto. Se pueden agregar/quitar campos a una tabla, modificar el tipo de un campo, agregar/quitar índices a una tabla, modificar un trigger, etc. (pág. 2).

Ejemplos:

Añadir un nuevo campo:
ALTER TABLE 'NOMBRE_TABLA' **ADD** NUEVO_CAMPO INT;

Eliminar una columna:
ALTER TABLE 'NOMBRE_TABLA' **DROP COLUMN** NOMBRE_COLUMNA;

Cambiar el nombre a una columna:
ALTER TABLE clientes **CHANGE** nombrecompania nombrempresa VARCHAR(20);

Solamente cambiar el tipo de dato de una columna:
ALTER TABLE clientes **MODIFY** nombrecompania **DATE NOT NULL**;

Asignar como clave primaria a una columna:
ALTER TABLE ingreso_alumno **ADD PRIMARY KEY**(cedula);

Asignar como clave secundaria a una columna:
ALTER TABLE usuarios **ADD FOREIGN KEY** (id_tipo) **REFERENCES** tipo_usuario(id);

Eliminar clave primaria:
ALTER TABLE usuarios **DROP PRIMARY KEY**;

Renombrar y/o cambiar el nombre la tabla:
ALTER TABLE usuario **RENAME** usuarios

Añadir una nueva columna después de un campo elegida:
ALTER TABLE usuarios **ADD** telefono **VARCHAR**(10) **AFTER** nombre_usuario;

Añadir una nueva columna en la primera posición de la tabla:
ALTER TABLE usuarios **ADD** correo **VARCHAR**(5) **FIRST**;

1.1.3. DROP

Arévalo (2013) afirma: Este comando elimina un objeto de la base de datos. Puede ser una tabla, vista, índice, trigger, función, procedimiento o cualquier otro objeto que el motor de la base de datos soporte. Se puede combinar con la sentencia ALTER (pág. 4).

Ejemplo:

Eliminar una tabla de la base de datos: DROP TABLE 'NOMBRE_TABLA';

Eliminar toda la base de datos: DROP DATABASE 'BASEDATOS';

1.1.4. TRUNCATE

"Este comando trunca todo el contenido de una tabla. La ventaja sobre el comando DROP, es que, si se quiere borrar todo el contenido de la tabla, es mucho más rápido, especialmente si la tabla es muy grande. La desventaja es que TRUNCATE sólo sirve cuando se quiere eliminar absolutamente todos los registros, ya que no se permite la cláusula WHERE. Si bien, en un principio, esta sentencia parecería ser DML (Lenguaje de Manipulación de Datos), es en realidad una DDL, ya que internamente, el comando TRUNCATE borra la tabla y la vuelve a crear y no ejecuta ninguna transacción". (Arévalo, 2013, pág. 4)

Ejemplo:

TRUNCATE TABLE 'NOMBRE_TABLA';

1.2. Sentencias DML complejas

Definición

"Un lenguaje de manipulación de datos (Data Manipulation Language, o DML en inglés) es un lenguaje proporcionado por el sistema de gestión de base de datos que permite a los usuarios llevar a cabo las tareas de consulta o manipulación de los datos, organizados por el modelo de datos adecuado. El lenguaje de manipulación de datos más popular hoy día es SQL, usado para ordenar, filtrar y extraer datos en una base de datos relacional". (Arévalo, 2013, pág. 4).

Tabla 2. Sentencias DML

Comando	Descripción
SELECT	Utilizado para consultar registros de la base de datos que satisfagan un criterio determinado
INSERT	Utilizado para cargar lotes de datos en la base de datos en una única operación.
UPDATE	Utilizado para modificar los valores de los campos y registros especificados Utilizado para modificar las tablas agregando campos o cambiando la definición de los campos.
DELETE	Utilizado para eliminar registros de una tabla2

2 https://geotalleres.readthedocs.io/es/latest/conceptos-sql/conceptos_sql.html

1.2.1. Consultas de selección (SELECT)

Arévalo (2013) afirma: Las consultas de selección se utilizan para indicar al motor de datos que devuelva información de las bases de datos, esta información es devuelta en forma de conjunto de registros. Este conjunto de registros es modificable. (pág. 8).

BÁSICAS

La sintaxis básica de una consulta de selección es:

SELECT Campos **FROM** Tabla;

SELECT Nombre, Telefono **FROM** Clientes;

Consultas con predicado

Arévalo (2013) afirma: **ALL** Si no se incluye ninguno de los predicados se asume ALL. El Motor de base de datos selecciona todos los registros que cumplen las condiciones de la instrucción SQL: (pág. 8).

SELECT ALL FROM Empleados;

SELECT * FROM Empleados;

1.2.3. INSERT

"Una sentencia INSERT de SQL agrega uno o más registros a una (y sólo una) tabla en una base de datos relacional.

Forma básica:

INSERT INTO "tabla" ("columna1", ["columna2,... "]) **VALUES** ("valor1", ["valor2,..."])

Las cantidades de columnas y valores deben ser iguales. Si una columna no se especifica, le será asignado el valor por omisión. Los valores especificados (o implícitos) por la sentencia INSERT deberán satisfacer todas las restricciones aplicables. Si ocurre un error de sintaxis o si alguna de las restricciones es violada, no se agrega la fila y se devuelve un error" (Arévalo, 2013, pág. 5).

Ejemplo:

INSERT INTO producto (codigo_producto,nombre_producto,codigo_proveedor, cantidad_unidad,precio_unidad) **VALUES** ("p008", "parlante","102", "5","8");

1.2.4. *UPDATE*

Arévalo (2013) afirma: Una sentencia UPDATE de SQL es utilizada para modificar los valores de un conjunto de registros existentes en una tabla (pág. 5).

Ejemplo:

UPDATE mi_tabla **SET** campo1 = 'nuevo valor campo1' **WHERE** campo2 = 'N';

1.2.5. *DELETE*

Arévalo (2013) afirma: Una sentencia DELETE de SQL borra uno o más registros existentes en una tabla (pág. 5).

Forma básica:

DELETE FROM 'tabla' **WHERE** 'columna1' = 'valor1'

Ejemplo:

DELETE FROM My_table **WHERE** field2 = 'N';

1.2.6. *CLAUSULAS*

Arévalo (2013) afirma: Las cláusulas son condiciones de modificación utilizadas para definir los datos que desea seleccionar o manipular (pág. 6).

Tabla 3. Cláusulas de las sentencias DML

Comando	Descripción
FROM	Utilizada para especificar la tabla de la cual se van a seleccionar los registros
GROUP BY	Utilizada para separar los registros seleccionados en grupos específicos
HAVING	Utilizada para expresar condición que debe satisfacer cada grupo
ORDER BY	Utilizada para ordenar los registros seleccionados de acuerdo con un orden específico
WHERE	Utilizada para determinar los registros seleccionados en la cláusula FROM3

3 https://geotalleres.readthedocs.io/es/latest/conceptos-sql/conceptos_sql.html

Clausula FROM:

Utilizada para especificar la tabla de la cual se van a seleccionar los registros

Ejemplo:

SELECT * FROM producto;

Clausula WHERE:

Arévalo (2013) afirma: La cláusula WHERE puede usarse para determinar qué registros de las tablas enumeradas en la cláusula FROM aparecerán en los resultados de la instrucción SELECT. WHERE es opcional, pero cuando aparece debe ir a continuación de FROM: (pág. 10).

Ejemplo:

SELECT Apellidos, Salario FROM Empleados **WHERE** Salario > 21000;

SELECT Id_Producto, Existencias FROM Productos **WHERE** Existencias <= Nuevo_Pedido;

1.2.7. ORDENAR LOS REGISTROS

Arévalo (2013) afirma: Se puede especificar el orden en que se desean recuperar los registros de las tablas mediante la cláusula **ORDER BY**: (pág. 8).

Ejemplo:

SELECT CodigoPostal, Nombre, Telefono FROM Clientes ORDER BY Nombre;

Se pueden ordenar los registros por más de un campo:

SELECT CodigoPostal, Nombre, Telefono FROM Clientes ORDER BY CodigoPostal, Nombre;

Y se puede especificar el orden de los registros: ascendente mediante la claúsula (ASC -se toma este valor por defecto) o descendente (DESC):

SELECT CodigoPostal, Nombre, Telefono FROM Clientes ORDER BY CodigoPostal DESC, Nombre ASC;

1.3. OPERADORES:

OPERADORES LÓGICOS

Tabla 4. Operadores lógicos

Operador	Uso
AND	Es el "y" lógico. Evalúa dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.
OR	Es el "o" lógico. Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta.
NOT	Es el "no" lógico. Evalúa la negación de una condición.4

Ejemplo:

Utilización del Operador AND

SELECT * **FROM** Empleados **WHERE** Edad > 25 **AND** Edad < 50;

Utilización del Operador OR

SELECT * **FROM** Empleados **WHERE** (Edad > 25 **AND** Edad < 50) **OR** Sueldo = 100;

Utilización del Operador NOT

SELECT * **FROM** Empleados **WHERE NOT** Estado = 'Soltero';

Consulta con condiciones lógicas anidadas (combinadas)

SELECT * **FROM** Empleados **WHERE** (Sueldo > 100 **AND** Sueldo < 500) **OR** (Provincia = 'Madrid' **AND** Estado = 'Casado');

4 https://geotalleres.readthedocs.io/es/latest/conceptos-sql/conceptos_sql.html

secretaria.general@itstena.edu.ec km 1 1/2 vía Tena- Archidona ISTT

OPERADORES DE COMPARACIÓN

Tabla 5. Operadores de comparación

Operador	Uso
<	Menor que
>	Mayor que
<>	Distinto de
<=	Menor o igual que
>=	Mayor o igual que
BETWEEN	Intervalo
LIKE	Comparación
In	Especificar5

Ejemplos:

Operador Menor que: SELECT * FROM producto WHERE precio_unidad<10;

Operador Mayor que: SELECT * FROM producto WHERE precio_unidad>10;

Operador Diferente: SELECT * FROM producto WHERE precio_unidad <> 10;

Operador DISTINCT

Arévalo (2013) afirma: Omite los registros que contienen datos duplicados en los campos seleccionados. Para que los valores de cada campo listado en la instrucción SELECT se incluyan en la consulta deben ser únicos: (pág. 8).

Ejemplo: SELECT DISTINCT Apellido FROM Empleados;

Operador DISTINCTROW

Arévalo (2013) afirma: Devuelve los registros diferentes de una tabla; a diferencia del predicado anterior que sólo se fijaba en el contenido de los campos seleccionados, éste lo hace en el contenido del registro completo independientemente de los campos indicados en la cláusula SELECT: (pág. 9).

Ejemplo: SELECT DISTINCTROW Apellido FROM Empleados;

Operador Mayor igual: SELECT * FROM producto WHERE precio_unidad>=10;

5 https://geotalleres.readthedocs.io/es/latest/conceptos-sql/conceptos_sql.html

Operador Menor igual: SELECT * FROM producto WHERE precio_unidad<=10;

Operador BETWEEN

Arévalo (2013) afirma: Para indicar que deseamos recuperar los registros según el intervalo de valores de un campo emplearemos el operador Between: (pág. 9).

Ejemplos:

SELECT * FROM Pedidos WHERE CodPostal Between 28000 And 28999;

SELECT If (CodPostal Between 28000 And 28999, 'Provincial', 'Nacional') FROM Editores;

(Devuelve el valor 'Provincial' si el código postal se encuentra en el intervalo, 'Nacional' en caso contrario)

Operador LIKE

Arévalo (2013) afirma: Se utiliza para comparar una expresión de cadena con un modelo en una expresión SQL. Su sintaxis es: (pág. 9).

Tabla 6. Operadores de comparación LIKE

Expresión LIKE modelo	Descripción
WHERE Nom_Campo LIKE 'b%'	Encuentra cualquier valor que comience con "b"
WHERE Nom_Campo LIKE '%a'	Encuentra cualquier valor que termine con "a"
WHERE Nom_Campo LIKE '%or%'	Encuentra cualquier valor que tenga "o" en cualquier posición.6

Ejemplos:

SELECT * FROM Store_Information WHERE Store_Name LIKE '%AN%';

SELECT * FROM alumno WHERE apellidos_nombres Like 'A%';

Operador IN:

Arévalo (2013) afirma: Este operador devuelve aquellos registros cuyo campo indicado coincide con alguno de los indicados en una lista.

Su sintaxis es: **Expresión [Not] In(valor1, valor2, . . .)** (pág. 10).

6 https://geotalleres.readthedocs.io/es/latest/conceptos-sql/conceptos_sql.html

Ejemplo:

SELECT * FROM Pedidos **WHERE** Provincia **In** ('Madrid', 'Barcelona', 'Sevilla');

(Devuelve los pedidos realizados en la provincia de 'Madrid', 'Barcelona', 'Sevilla')

Operador NOT IN

Arévalo (2013) afirma: Este operador devuelve aquellos registros cuyo campo indicado no coincide con alguno de los indicados en una lista.

Su sintaxis es: **Expresión [Not] In(valor1, valor2, . . .)** (pág. 10)**.**

Ejemplo:

SELECT * FROM Pedidos **WHERE** Provincia Not In ('Madrid', 'Barcelona', 'Sevilla');

(Devuelve los pedidos que no estén realizados en la provincia de 'Madrid', 'Barcelona', 'Sevilla')

AVG

Arévalo (2013) afirma: Calcula la media aritmética de un conjunto de valores contenidos en un campo especificado de una consulta:

Su sintaxis es: **Avg(expr)**

La función Avg no incluye ningún campo Null en el cálculo. Un ejemplo del funcionamiento de AVG: (pág. 10).

Ejemplo:

SELECT Avg(Gastos) AS Promedio FROM Pedidos WHERE Gastos > 100;

MAX, MIN:

Arévalo (2013) afirma: Devuelven el mínimo o el máximo de un conjunto de valores contenidos en un campo específico de una consulta.

Su sintaxis es: Min(expr), Max(expr) (pág. 10).

Ejemplo:

SELECT Min(Gastos) AS ElMin FROM Pedidos WHERE Pais = 'Costa Rica';

SELECT Max(Gastos) AS ElMax FROM Pedidos WHERE Pais = 'Costa Rica';

SUM:

Arévalo (2013) afirma: Devuelve la suma del conjunto de valores contenido en un campo específico de una consulta.

Su sintaxis es: **Sum(expr)** (pág. 11).

Ejemplo:

SELECT Sum(PrecioUnidad * Cantidad) AS Total FROM DetallePedido;

GROUP BY:

Arévalo (2013) afirma: Combina los registros con valores idénticos, en la lista de campos especificados, en un único registro: (pág. 11).

Ejemplo:

SELECT campos FROM tabla WHERE criterio GROUP BY campos del grupo

Todos los campos de la lista de campos de SELECT deben o bien incluirse en la cláusula

GROUP BY o como argumentos de una función SQL agregada:

SELECT Id_Familia, Sum(Stock) FROM Productos GROUP BY Id_Familia;

HAVING

"Es similar a WHERE, determina qué registros se seleccionan. Una vez que los registros se han agrupado utilizando GROUP BY, HAVING determina cuáles de ellos se van a mostrar.

SELECT Id_Familia Sum(Stock) FROM Productos GROUP BY Id_Familia HAVING Sum(Stock) > 100 AND NombreProducto Like BOS*;" (Arévalo, 2013, pág. 11).

TODOS LOS TIPOS DE JOIN EN SQL

"Los JOINs en SQL sirven para combinar filas de dos o más tablas basándose en un campo común entre ellas, devolviendo por tanto datos de diferentes tablas. Un JOIN se produce cuando dos o más tablas se juntan en una sentencia SQL.

Existen más tipos de joins en SQL que los que aquí se explican, como CROSS JOIN, O SELF JOIN, pero no todos ellos están soportados por todos los sistemas de bases de datos.

Los más importantes son los siguientes:

1. **INNER JOIN:** Devuelve todas las filas cuando hay al menos una coincidencia en ambas tablas.
2. **LEFT JOIN:** Devuelve todas las filas de la tabla de la izquierda, y las filas coincidentes de la tabla de la derecha.
3. **RIGHT JOIN:** Devuelve todas las filas de la tabla de la derecha, y las filas coincidentes de la tabla de la izquierda.
4. **OUTER JOIN:** Devuelve todas las filas de las dos tablas, la izquierda y la derecha. También se llama FULL OUTER JOIN". (Lázaro, 2018, pág. 1)

1. INNER JOIN

Lázaro (2018) afirma: **INNER JOIN** selecciona todas las filas de las dos columnas siempre y cuando haya **una coincidencia entre las columnas en ambas tablas**. Es el tipo de JOIN más común.

Ejemplo:

```
SELECT nombreColumna(s)
FROM tabla1
INNER JOIN tabla2
ON tabla1.nombreColumna=table2.nombreColumna; (pág. 2).
```

Se ve más claro utilizando una **imagen**:

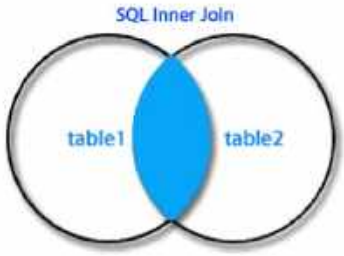


Gráfico 1. SQL Inner Join
Desarrollado por: Lázaro diego

Vamos a verlo también con un ejemplo, mediante las tablas **Clientes** y **Pedidos**:

Clientes:

Tabla 7. Clientes

ClienteID	NombreCliente	Contacto
1	Marco Lambert	456443552
2	Lydia Roderic	445332221
3	Ebbe Therese	488982635
4	Sofie Mariona	412436773

Desarrollado por: Lázaro diego

Pedidos:

Tabla 8. Pedidos

PedidoID	ClienteID	Factura
234	4	160
235	2	48
236	3	64
237	4	92

Desarrollado por: Lázaro diego

Lázaro (2018) afirma: La siguiente **sentencia SQL** devolverá **todos los clientes con pedidos**:

```
SELECT Clientes.NombreCliente, Pedidos.PedidoID FROM Clientes
INNER JOIN Pedidos ON Clientes.ClienteID=Pedidos.ClienteID
ORDER BY Clientes.NombreCliente;
```

Si hay filas en Clientes que no tienen coincidencias en Pedidos, los Clientes no se mostrarán. La sentencia anterior mostrará el siguiente resultado: (pág. 2).

Tabla 9. Resultado de la consulta Clientes-Pedidos

NombreCliente	PedidoID
Ebbe Therese	236
Lydia Roderic	235
Sofie Mariona	234
Sofie Mariona	237

Desarrollado por: Lázaro diego

Lázaro (2018) afirma: ***Sofie Mariona*** aparece dos veces ya que ha realizado dos pedidos.

No aparece ***Marco Lambert***, pues no ha realizado ningún pedido (pág. 3).

2. LEFT JOIN

"LEFT JOIN mantiene **todas las filas de la tabla izquierda** (la tabla1). Las filas de la tabla derecha se mostrarán si hay una coincidencia con las de la izquierda. Si existen valores en la tabla izquierda pero no en la tabla derecha, ésta mostrará null.
Ejemplo:

SELECT nombreColumna(s)
FROM tabla1
LEFT JOIN tabla2
ON tabla1.nombreColumna=tabla2.nombreColumna;". (Lázaro, 2018, pág. 3)

La **representación de LEFT JOIN en una imagen** es:

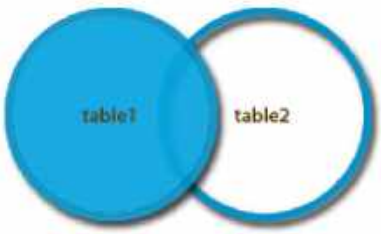


Gráfico 2. SQL Left Join
Desarrollado por: Lázaro diego

Lázaro (2018) afirma: Tomando de nuevo las tablas de **Productos** y **Pedidos**, ahora queremos mostrar **todos los clientes**, **y cualquier pedido** que pudieran haber encargado:

SELECT Clientes.NombreCliente, Pedidos.PedidoID
FROM Clientes LEFT JOIN Pedidos

ON Clientes.ClienteID=Pedidos.ClienteID
ORDER BY Clientes.NombreCliente; (pág. 3).

La sentencia anterior devolverá lo siguiente:

Tabla 10. Resultado de la consulta Left Join

NombreCliente	PedidoID
Ebbe Therese	236
Lydia Roderic	235
Marco Lambert	(null)
Sofie Mariona	234
Sofie Mariona	237

Desarrollado por: Lázaro diego

Lázaro (2018) afirma: Ahora vemos que se muestran todas las filas de la tabla **Clientes**, que es la tabla de la izquierda, tantas veces como haya coincidencias con el lado derecho. ***Marco Lambert*** no ha realizado ningún pedido, por lo que se muestra **null** (pág. 3).

3. RIGHT JOIN

"Es igual que **LEFT JOIN** pero al revés. Ahora **se mantienen todas las filas de la tabla derecha** (tabla2). Las filas de la tabla izquierda se mostrarán si hay una coincidencia con las de la derecha. Si existen valores en la tabla derecha pero no en la tabla izquierda, ésta se mostrará **null**.

Ejemplo:

SELECT nombreColumna(s)
FROM tabla1
RIGHT JOIN tabla2
ON tabla1.nombreColumna=tabla2.nombreColumna;". (Lázaro, 2018, pág. 4)

La **imagen que representa a RIGHT JOIN** es:

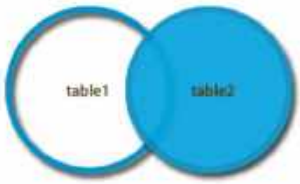


Gráfico 3. Right Join

Desarrollado por: Lázaro diego

"De nuevo tomamos el ejemplo de **Clientes** y **Pedidos**, y vamos a hacer el mismo ejemplo anterior, pero cambiado LEFT por RIGHT:

```
SELECT Pedidos.PedidoID, Clientes.NombreCliente
FROM Clientes RIGHT JOIN Pedidos
ON Clientes.ClienteID=Pedidos.ClienteID
ORDER BY Pedidos.PedidoID;
```

Ahora van a aparecer todos los pedidos, y los nombres de los clientes **que han realizado un pedido**. Nótese que también se ha cambiado el orden, y se han ordenado los datos por PedidoID". (Lázaro, 2018, pág. 4)

Tabla 11. Resultado de la consulta Right Join

PedidoID	NombreCliente
234	Sofie Mariona
235	Lydia Roderic
236	Ebbe Therese
237	Sofie Mariona

Desarrollado por: Lázaro diego

4. OUTER JOIN

"OUTER JOIN o **FULL OUTER JOIN** devuelve todas las filas de la tabla izquierda (tabla1) y de la tabla derecha (tabla2). Combina el resultado de los joins **LEFT** y **RIGHT**. Aparecerá null en cada una de las tablas alternativamente cuando no haya una coincidencia. **Ejemplo:**

```
SELECT nombreColumna(s)
FROM tabla1
```

OUTER JOIN tabla2

ON tabla1.nombreColumna=tabla2.nombreColumna;". (Lázaro, 2018, pág. 5)

La **imagen que representa el OUTER JOIN** es la siguiente:

Gráfico 4. SQL Outer Join

Desarrollado por: Lázaro diego

"Vamos a obtener **todas las filas** de las tablas **Clientes** y **Pedidos**:

```
SELECT Clientes.NombreCliente, Pedidos.PedidoID
FROM Clientes OUTER JOIN Pedidos
ON Clientes.ClienteID=Pedidos.ClienteID
ORDER BY Clientes.NombreCliente;
```

La sentencia devolverá **todos los Clientes** y **todos los Pedidos**, si un cliente no tiene pedidos mostrará **null** en **PedidoID**, y si un pedido no tuviera un cliente mostraría **null** en **NombreCliente** (en este ejemplo no sería lógico que un Pedido no tuviera un cliente).

La sintaxis de **OUTER JOIN o FULL OUTER JOIN no existen en MySQL**, pero se puede conseguir el mismo resultado de diferentes formas, esta es una:

```
SELECT Clientes.NombreCliente, Pedidos.PedidoID
FROM Clientes
LEFT JOIN Pedidos ON Clientes.ClienteID=Pedidos.ClienteID
UNION
SELECT Clientes.NombreCliente, Pedidos.PedidoID
FROM Clientes
```

RIGHT JOIN Pedidos ON Clientes.ClienteID=Pedidos.ClienteID;". (Lázaro, 2018, pág. 5)

UNIDAD 2: IMPLEMENTACIÓN DE ÍNDICES

2.1 Implementación de Índices.
2.2 Actividades de las Unidades 1 y 2

Resultado de Aprendizaje

El estudiante desarrollará habilidades para implementar y gestionar índices en bases de datos relacionales, maximizando la eficiencia en la ejecución de consultas y asegurando un diseño óptimo del sistema.

DIAGRAMA DE APRENDIZAJE

SÍNTESIS

En esta unidad, se explorará el uso de índices en bases de datos como herramienta para mejorar el rendimiento de las consultas.

2.1. Implementación de Índices

Uso de los índices o porque indexar

"El objetivo de los índices es permitir un acceso más rápido a la información durante las extracciones (SELECT) o actualizaciones (INSERT, UPDATE y DELETE) de datos, reduciendo el tiempo necesario para localizar un registro.

". (Gabillaud, 2013, pág. 99)

"Normalmente se utilizan sobre aquellos campos sobre los cuales se hacen las búsquedas más frecuentes.

Un índice funciona de forma similar al índice de un libro, guardando parejas de elementos: el elemento que se desea indexar y su posición en el fichero de la base de datos.

Podemos destacar dos tipos de índices generales:

Hash: se refiere a una función o método para generar claves que representen de manera casi univoca a un documento, registro, archivo, etc. Y que resuma o identifique un dato a través de la probabilidad, utilizando una función hash o el algoritmo hash.

Arboles-B: Estos son estructuras de datos de árbol que se encuentran comúnmente en las de base de datos y sistemas de archivos". (Lopez Sanz, Soltero Domingo, Sanchez Fuquene, Moreno Perez, Bollati, & Vara Mesa, 2016, pág. 168)

2.1.1. Crear un índice

"Un índice se puede crear en cualquier momento, haya o no datos en la tabla.
Sin embargo, si se tiene que importar los datos, es mejor importarlos primero y después definir los índices. En caso contrario (los índices se definen antes de una importación importante de datos), es necesario reconstruir los índices para garantizar un reparto equilibrado de los datos en el índice". (Gabillaud, 2013, pág. 105)
Las principales opciones y argumentos del comando de creación del índice son las siguientes:

CREATE UNIQUE INDEX nombresss ON usuario (nombre,apellido);

2.1.2. Eliminar un índice

Drop index nombre_index on alumno;

Ver como esta creado un índice.

SHOW INDEX FROM alumno;

2.1. ACTIVIDADES DE LA UNIDAD 1 y 2

INDIVIDUAL

ACTIVIDAD 1

1. **Crear la base de datos**

   TERCERO_DESARROLO_SOFTWARE

2. **Crear las siguientes tablas**

   Alumno (cedula, nombre, apellido, teléfono, dirección, correo)

   Profesor (cedula, nombre, apellido, teléfono, correo, titulo)

   Curso (código_curso, nombre_curso, fecha_inicio, fecha_fin, horas)

3. **Crear las llaves primarias a las tablas anteriores.**

ACTIVIDAD 2

Crear la siguiente base de datos, con sus debidas llaves principales.

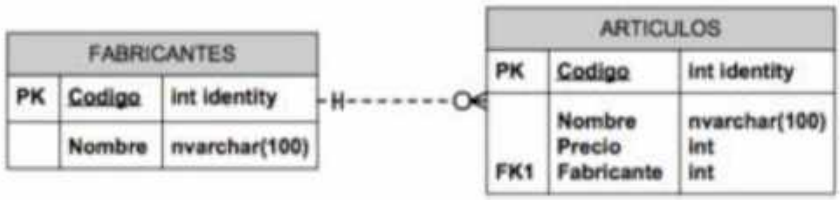


Gráfico 5. Relación entre Fabricantes y Artículos

ACTIVADED 3

Crear la siguiente base de datos que se ilustra a continuación.

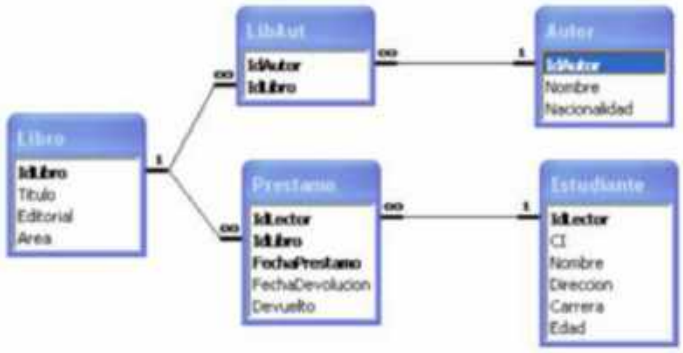


Gráfico 6. Entidad Relación de una Bibliotéca

ACTIVIDAD 4

Crear la siguiente base de datos e ingresar los registros que se ilustran cada tabla.

***Tabla 12**. Estudiantes*

Estudiante

Cedula	Nombre	Apellido	Dirección	teléfonos	Correo
1501112351	CARLOS RAMON	ALVARADO GREFA	Guayas	0967506678	micorreo@yahoo.es
1500899073	JEFERSON NIXON	ALVARADO GREFA	Archidona	0967506679	micorreo@yahoo.es
1550201261	WENDY SHAKIRA	ALVARADO GREFA	Muyuna	0967506680	micorreo@yahoo.es
1550040222	INTI LUIS	AVILES ANDI	recinto feria	0967506681	micorreo@yahoo.es

Desarrollado por: El Autor

***Tabla 13**. Profesor*

Profesor

Cedula	nombre	Apellido	Correo	telefono
1502021221	Patricio	Guanipatin	profesor@yahoo.es	0987878787
1502021222	Carlos	Tipan	profesor@yahoo.es	0987878787
1502021223	Libinton	Lara	profesor@yahoo.es	0987878787

Desarrollado por: El Autor

Tabla 14. Asignatura

Asignatura		
Código	**Nombre**	**horas**
BDA3	base de datos avanzada	168
DG3	diseño grafico	120
CDI3	calculo diferencial e integral	150

Desarrollado por: El Autor

Tabla 15. Titulo

Titulo				
Código	**Nombre**	**Institución**	**anio**	**cedula**
ISC1	Ingeniero en sistemas computacionales	universidad estatal de bolivar	2013	1502021221
IR1	Ingeniero en redes	Universidad central	2005	1502021222
IM1	Ingeniero en matemáticas	Universidad IKIAN	2010	1502021223

Desarrollado por: El Autor

Tabla 16. Estudiante_Asignatura

estudiante_asignatura	
Código	Cedula
CDI3	1501112351
BDA3	1550201261
DG3	1550040222
BDA3	1500899073
CDI3	1550201261

Desarrollado por: El Autor

Tabla 17. Asignatura_Profesor

asignatura-profesor	
Cedula	**Código**
1502021221	BDA3
1502021222	DG3
1502021223	CDI3
1502021221	DG3
1502021222	BDA3

Desarrollado por: El Autor

ACTIVIDAD 5

ALTERAR LAS TABLAS CON LOS SIGUIENTES VALORES

1. Cambiar el nombre de la tabla alumno por estudiantes
2. Cambiar el nombre de la tabla maestro por profesor
3. Cambiar el nombre del campo No_Casa por número de casa
4. Cambiar el nombre del campo nombre por nom_usuario en la tabla usuario.
5. Anadir los siguientes campos (edad, estado civil y fecha de nacimiento) en la tabla persona.
6. Cambiar el tipo de dato al campo teléfono en la tabla persona por tipo entero.
7. Eliminar la llave principal a la tabla dirección.
8. Anadir una llave principal a la tabla dirección.
9. Anadir una llave foránea a la tabla persona.

ACTIVIDAD 6

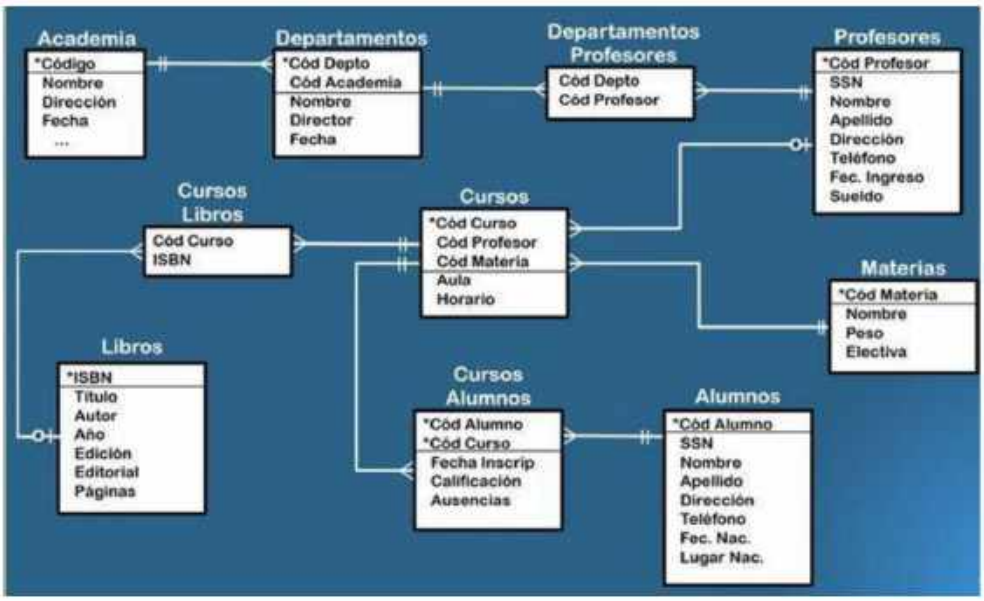


Gráfico 7. Entidad Relación de un sistema de Curso Virtual

1. **Crear la siguiente base de datos, que se ilustra en la imagen anterior.**
2. **Ingresar registros (Datos reales)**
   - **2.1.** Ingresar los datos de cada uno de sus profesores.
   - **2.2.** Ingresar los datos de todos los estudiantes.
   - **2.3.** Ingresar las asignaturas (Materia) que están recibiendo
   - **2.4.** Ingresar un libro por asignatura (Materia). Ingresar al internet y copiar todos los datos del libro (datos reales)
   - **2.5.** En curso ingresar (laboratorio 1 y laboratorio 2)
3. **Realizar las siguientes consultas**
   - **3.1.** Obtener el listado de todos los profesores y ordenado Ascendentemente.
   - **3.2.** Obtener el listado de todos los alumnos y ordenado Descendentemente.
   - **3.3.** Obtener el listado de los alumnos que reciben la asignatura de Programación Visual.
   - **3.4.** Obtener el listado de los alumnos que reciben la asignatura de Base de datos avanzado y en que aula lo reciben.
   - **3.5.** Obtener el listado de los alumnos que reciben la asignatura de Diseño Multimedia y el nombre del profesor que lo imparte.
   - **3.6.** Obtener el listado de los alumnos que empiecen sus apellidos con la letra "S"
   - **3.7.** Obtener los datos del profesor que imparte la asignatura de Programación Visual, Diseño Multimedia y Calculo Diferencial e Integral
   - **3.8.** Obtener el nombre del profesor que imparte la cátedra en el laboratorio 2 y el nombre de la asignatura.

3.9. Obtener el nombre del libro y a que asignatura pertenece.

3.10. Obtener el nombre del libro, el aula, del profesor que imparte la asignatura de base de datos.

3.11. Obtener el nombre del libro que fue publicado en el año 2010.

3.12. Obtener el nombre del profesor que su sueldo sea mayor a 1212.

3.13. Obtener el nombre del profesor que su sueldo este entre 1212 y 986.

3.14. Obtener los datos del estudiante que su calificación sea muy buena.

3.15. Quiero obtener el listado de los estudiantes que no recibe la materia de Base de Datos Avanzado.

3.16. Obtener el listado de los estudiantes que reciben la asignatura Fundamentos de Administración y Base de Datos Avanzado, también quiero saber el nombre del profesor y el nombre del libro.

EN GRUPO

ACTIVIDAD 1

Crear la siguiente base de datos en SQL

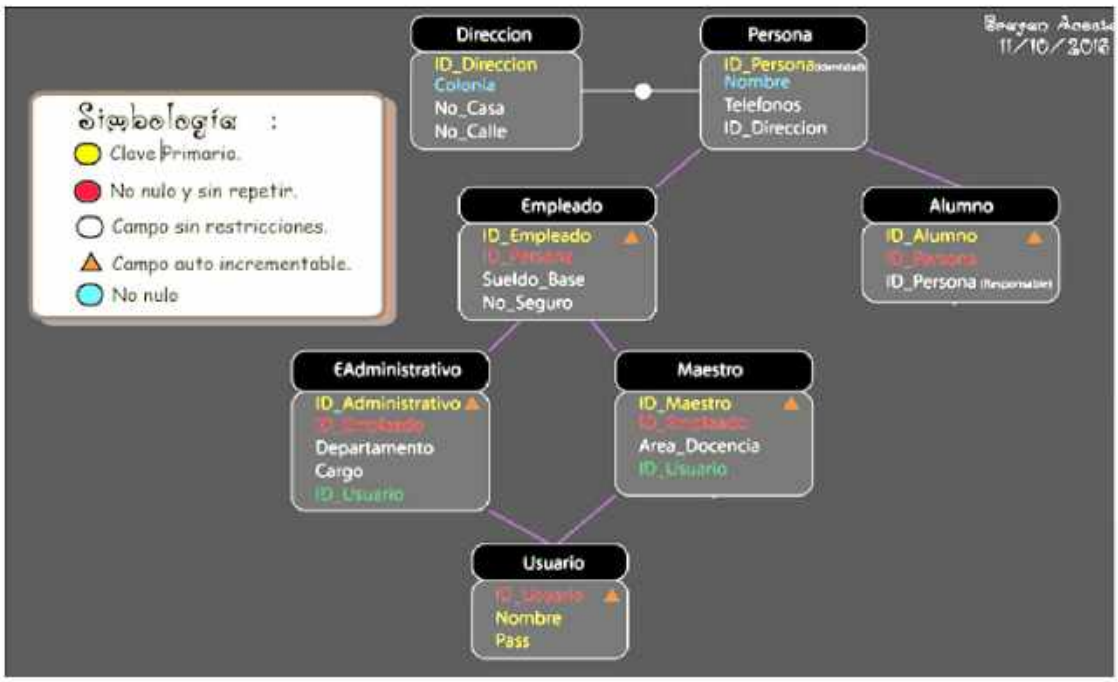


Gráfico 8. Entidad Relación de un sistema de Matricula

UNIDAD 3: IMPLEMENTACIÓN DE TRIGGER, VISTAS, PROCEDIMIENTOS Y FUNCIONES

3.1. Implementación de Trigger
3.2. Creación de Vistas
3.3. Procedimientos, índices y cursores

Resultado de Aprendizaje

Al finalizar esta unidad, el estudiante será capaz de: Implementación de Trigger, Creación de Vistas, Procedimientos, Índices y Cursores y Funciones

DIAGRAMA DE APRENDIZAJE

SÍNTESIS

En esta unidad se abordarán herramientas avanzadas de bases de datos para automatizar procesos, organizar el acceso a datos y optimizar su manipulación: Implementación de Triggers, Creación de Vistas, Procedimientos, Índices y Cursores y Funciones. Proporcionando al estudiante herramientas avanzadas para gestionar, optimizar y automatizar tareas en bases de datos, asegurando eficiencia, seguridad y reutilización de componentes.

3.1. Trigger

3.1.1. Implementación de Trigger, estructura básica:

Días (2016) afirma: Un triggers es un objeto que creas en una base de datos, y este objeto siempre va a estar asociado a una tabla

Este triggers desencadenará una acción cuando ocurre algo en esa tabla de base de datos (pág. 1).

Que puede ocurrir en una base de datos y que acción se puede ejecutar:

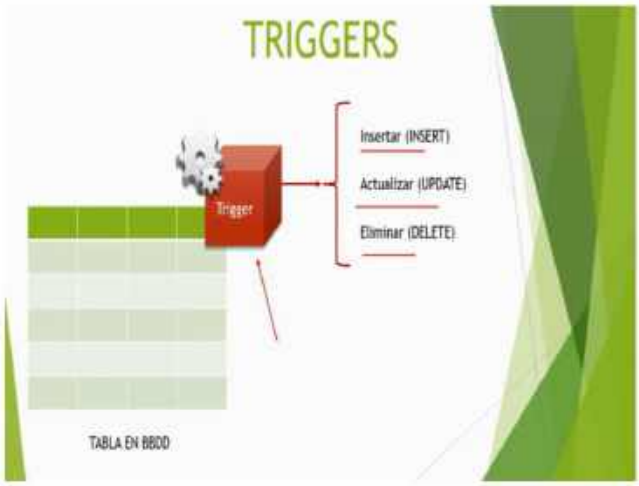


Gráfico 9. En las acciones que se ejecuta el Trigger

La acción que va a realizar el triggers: es copiar los registros que se ha realizado de forma automática.

Gráfico 10. Cuando se ejecuta el Trigger en cada acción

PASOS PARA CREAR UN TRIGGERS

1. Crear una nueva tabla con los mismo campos y tipos de datos de la tabla en donde va a ejecutar el triggers, más un campo adicional para guardar el nombre del usuario y la fecha en que realizó esa acción.

2. Crear el trigger asociando a la tabla creada anteriormente.

PROCESO PARA CREAR UN TRIGGERS

Gráfico 11. Sintáxis de creación de Triggers

Elaborado por: Autores

ELIMINAR UN TRIGGERS

DROP TRIGGER nombre del trigger;

3.2. Creación de vistas

"Una vista es una tabla virtual cuyo contenido está definido por una consulta. Al igual que una tabla, una vista consta de un conjunto de columnas y filas de datos con un nombre. Las filas y las columnas de datos proceden de tablas a las que se hace referencia en la consulta que define la vista y se producen de forma dinámica cuando se hace referencia a la vista.

Las vistas suelen usarse para centrar, simplificar y personalizar la percepción de la base de datos para cada usuario. Las vistas pueden emplearse como mecanismos de seguridad, que permiten a los usuarios obtener acceso a los datos por medio de la vista, pero no les conceden el permiso de obtener acceso directo a las tablas base subyacente de la vista". (Porro Chulli, 2018, pág. 3)

¿Por qué usar Vistas?

- Para restringir el acceso a la B.D.
- Para realizar consultas complejas de manera fácil.
- Para obtener una independencia de los datos
- Para presentar diferentes vistas de los mismos datos.

SINTAXIS:

Porro (2018) afirma: Una vista permite guardar una instrucción SQL que podrás ejecutar las veces que quieras sin tener que repetir la instrucción SQL.

Esta es la sintaxis para una vista:

CREATE VIEW nombre_vista AS instrucción SQL (pág. 5).

Ejemplo:

- Crear una vista para obtener los datos de la tabla producto.
  - **CREATE VIEW** consulta_producto **AS** SELECT CODIGO_PRODUCTO, NOMBRE_PRODUCTO FROM productos;

Para ejecutar la vista, basta con lanzar la SELECT:

SELECT * FROM nombre_vista.

Para eliminar la vista:

DROP VIEW nombre_vista;

ACTIVIDADES DE LA UNIDAD 2

INDIVIDUAL

Gráfico 12. Alumno

ACTIVIDAD 1

Crear TRIGGERS para que se ejecute, al momento de ingresar un registro en la tabla alumnos

ACTIVIDAD 2

Crear TRIGGERS para que se ejecute, al momento de actualizar un registro en la tabla alumnos

ACTIVIDAD 3

Crear TRIGGERS para que se ejecute, al momento de eliminar un registro en la tabla alumnos

ACTIVIDAD 4

CREAR VISTAS PARA LAS SIGUIENTES CONSULTAS:

1. Quiero obtener el nombre del producto y el proveedor (**nombre_compania).**
2. Quiero obtener el nombre del producto, el proveedor (**nombre_compania**) y que cantidad de pedido lo realizó.
3. Obtener la fecha en que lo realizó el pedido a un **X** proveedor **(nombre del proveedor).**
4. Quiero saber que empleado realizó un pedido de un **X** producto y a que proveedor (**nombre_compania**) lo realizó, cuando la fecha de pedido sea igual a **"2018/07/09"**.

EN GRUPO

ACTIVIDAD 1

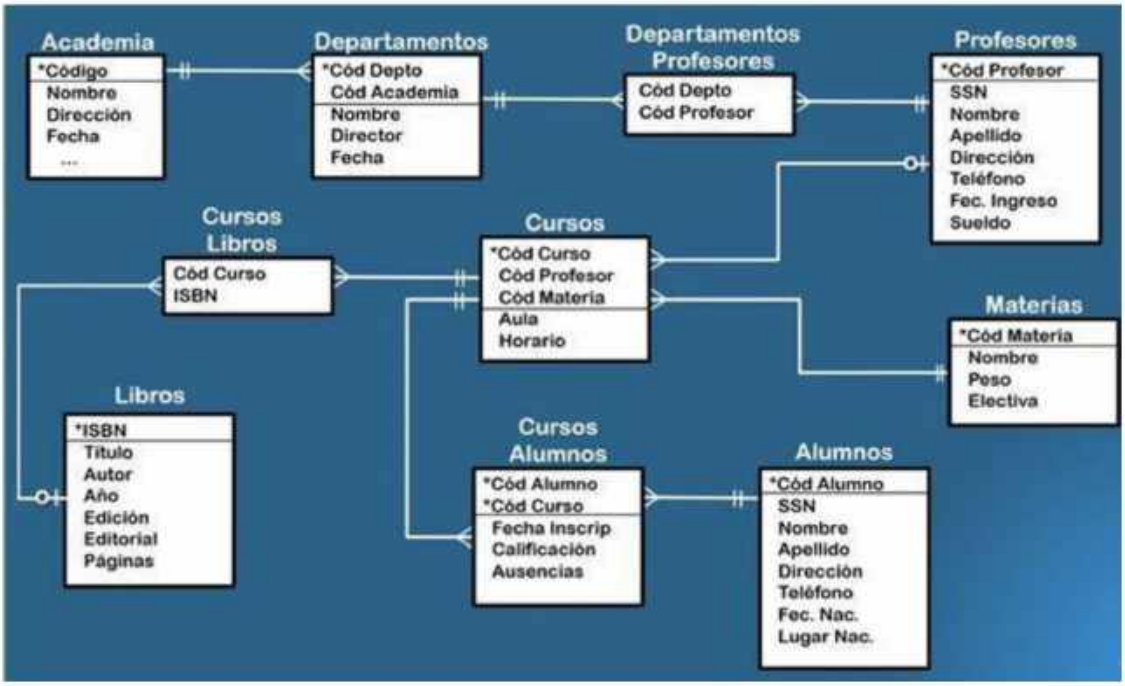


Gráfico 13. Entidad Relación de un sistema de Curso Virtual

1. **CREAR VISTAS PARA LAS SIGUIENTES CONSULTAS:**
   - **1.1.** Obtener el listado de todos los profesores y ordenado Ascendentemente.
   - **1.2.** Obtener el listado de todos los alumnos y ordenado Descendentemente.
   - **1.3.** Obtener el listado de los alumnos que reciben la asignatura de Programación Visual.
   - **1.4.** Obtener el listado de los alumnos que reciben la asignatura de Base de datos avanzado y en que aula lo reciben.
   - **1.5.** Obtener el listado de los alumnos que reciben la asignatura de Diseño Multimedia y el nombre del profesor que lo imparte.
   - **1.6.** Obtener el listado de los alumnos que empiecen sus apellidos con la letra "S"
   - **1.7.** Obtener los datos del profesor que imparte la asignatura de Programación Visual, Diseño Multimedia y Calculo Diferencial e Integral
   - **1.8.** Obtener el nombre del profesor que imparte la catedra en el laboratorio 2 y el nombre de la asignatura.
   - **1.9.** Obtener el nombre del libro y a que asignatura pertenece.
   - **1.10.** Obtener el nombre del libro, el aula, del profesor que imparte la asignatura de base de datos.
   - **1.11.** Obtener el nombre del libro que fue publicado en el año 2010.
   - **1.12.** Obtener el nombre del profesor que su sueldo sea mayor a 1212.
   - **1.13.** Obtener el nombre del profesor que su sueldo este entre 1212 y 986.
   - **1.14.** Obtener los datos del estudiante que su calificación sea muy buena.
   - **1.15.** Quiero obtener el listado de los estudiantes que no recibe la materia de Base de Datos Avanzado.
   - **1.16.** Obtener el listado de los estudiantes que reciben la asignatura Fundamentos de Administración y Base de Datos Avanzado, también quiero saber el nombre del profesor y el nombre del libro.

3.3.Procedimientos, índices y cursores

3.3.1. Procedimientos almacenados

3.3.1.1. Definición de procedimientos:

- Porro (2018) afirma: Es un programa almacenado físicamente en una base de datos. Está formado por un conjunto de comandos SQL.
- Su implementación varía de un gestor de base de datos a otro (pág. 3).

Ventajas:

- Únicamente se programa una vez y es reutilizable.
- Solo se realiza una conexión a la base de datos y este ejecuta todo el comando sin tener que establecer una nueva conexión.
- Esta directamente bajo el control del motor del gestor de base de datos, aumentando con ello la rapidez de procesamiento de las peticiones del usuario.

Desventajas:

- Como los procedimientos se encuentran guardados en la base de datos están propensos a perderse si la base de datos se corrompe.
- Si se maneja un proceso de backup periódicamente para la base de datos, podemos evitar este tipo de problemas.

Sintaxis :

- Para iniciar el trabajo con procedimientos almacenados debemos declarar nuevos delimitadores con la sentencia DELIMITER seguido del nuevo símbolo para delimitar.
- El procedimiento es creado con la sentencia CREATE PROCEDURE seguido del nombre.
- Después del nombre vienen los parámetros de entrada, estos son opcionales.
- Los parámetros tienen la siguiente estructura: Modo, Nombre, Tipo.

- Modo: es opcional y puede ser IN, OUT e INOUT.
- Nombre: es el nombre del parámetro.
- Tipo: es cualquier tipo de dato de los provistos por MySql.

- Luego de los parámetros sigue la definición, este es el cuerpo del procedimiento y está compuesto por el procedimiento en sí. Aquí se define que hace, como lo hace y bajo qué circunstancias lo hace. Esta definición debe llevar la sentencia BEGIN al inicio y END al final. EL END debe estar seguido del símbolo delimitador que definimos al inicio.
- Al finalizar el trabajo con los procedimientos almacenados debemos regresar el delimitador a su símbolo original con la sentencia DELIMITER y el símbolo ";".

Ejemplo de la estructura

```
DELIMITER $$
CREATE PROCEDURE procedimiento (IN v_id INT)
BEGIN
   SELECT * FROM table WHERE id = v_id;
END $$
DELIMITER;
```

Ejecución

Para ejecutar un procedimiento se utiliza la sentencia CALL seguido del nombre del procedimiento y sus respectivos parámetros.

- Ejemplo
  CALL procedimiento (2)

Sentencias a utilizar

- **CREATE PROCEDURE:**
  Nos permite crear un procedimiento

- **DROP PROCEDURE:**
  Nos permite eliminar un procedimiento

3.3.1.2. Índices y Cursores

Uso de los índices o porque indexar

"El objetivo de los índices es permitir un acceso más rápido a la información durante las extracciones (SELECT) o actualizaciones (INSERT, UPDATE y DELETE) de datos, reduciendo el tiempo necesario para localizar un registro". (Gabillaud, 2013, pág. 99)

"Normalmente se utilizan sobre aquellos campos sobre los cuales se hacen las búsquedas más frecuentes.

Un índice funciona de forma similar al índice de un libro, guardando parejas de elementos: el elemento que se desea indexar y su posición en el fichero de la base de datos.

Podemos destacar dos tipos de índices generales:

Hash: se refiere a una función o método para generar claves que representen de manera casi univoca a un documento, registro, archivo, etc. Y que resuma o identifique un dato a través de la probabilidad, utilizando una función hash o el algoritmo hash.

Arboles-B: Estos son estructuras de datos de árbol que se encuentran comúnmente en las de base de datos y sistemas de archivos". (Lopez Sanz, Soltero Domingo, Sanchez Fuquene, Moreno Perez, Bollati, & Vara Mesa, 2016, pág. 168)

Crear un índice

"Un índice se puede crear en cualquier momento, haya o no datos en la tabla.
Sin embargo, si se tiene que importar los datos, es mejor importarlos primero y después definir los índices. En caso contrario (los índices se definen antes de una importación importante de datos), es necesario reconstruir los índices para garantizar un reparto equilibrado de los datos en el índice". (Gabillaud, 2013, pág. 105).

Las principales opciones y argumentos del comando de creación del índice son las siguientes:

CREATE UNIQUE INDEX nombresss ON usuario (nombre,apellido);

Eliminar un índice

Drop index nombre_index on alumno;

Ver como esta creado un índice.

SHOW INDEX FROM alumno;

3.4. FUNCIONES

Definición de Funciones:

Al igual que los procedimientos almacenados, las funciones están formadas por un conjunto de comandos SQL y están almacenados físicamente en la base de datos.

Ventajas y Desventajas:

- Las funciones comparten las mismas ventajas y desventajas que

Uso:

El uso que se le da a las funciones es cuando necesitamos recibir un único valor simple que se obtiene de una operación recurrente.

Sintaxis:

- Para iniciar el trabajo con funciones debemos declarar nuevos delimitadores con la sentencia DELIMITER seguido del nuevo símbolo para delimitar.
- La función es creada con la sentencia CREATE FUNCTION seguido del nombre.
- Luego del nombre vienen los parámetros de entrada, estos son opcionales.

  Los parámetros tienen la siguiente estructura: Nombre Tipo

  - Nombre: Es el nombre del parámetro.
  - Tipo: Es cualquier tipo de dato de los provistos por MySql.

- Después de los parámetros de entrada se ingresa la sentencia RETURNS seguido del tipo de dato de la respuesta que se devolverá la función.
- Después de la sentencia RETURNS ingresamos la sentencia BEGIN para identificar donde inicia la definición.
- Posterior a la sentencia BEGIN sigue la definición, este es el cuerpo de la función y está compuesto por las operaciones a realizar.
- Al final de la definición se introduce la sentencia RETURN seguido de la variable que contenga el resulta de la operación.
- Termina la función con la sentencia END seguido del símbolo delimitador que definamos al inicio.
- Al finalizar el trabajo con funciones debemos regresar el delimitador a su símbolo original con la sentencia DELIMITER y el símbolo ";".

Ejemplo de la estructura:

```
DELIMITER $$
CREATE FUNCTION Hola (s CHAR (20))
RETURNS CHAR (50)
BEGIN
  RETURN CONCAT ('Hola', s);
END $$
DELIMITER;
```

Ejecución:

- La ejecución de una función se hace a través de la sentencia SELECT.

  SELECT Hola ('mundo')
- **La respuesta es:** Hola mundo

Sentencias a utilizar:

- CREATE FUNCTION: Nos permite crear una función.
- DROP FUNCTION: Nos permite eliminar una función.

ACTIVIDADES DE LA UNIDAD 3

INDIVIDUAL

ACTIVIDAD 1

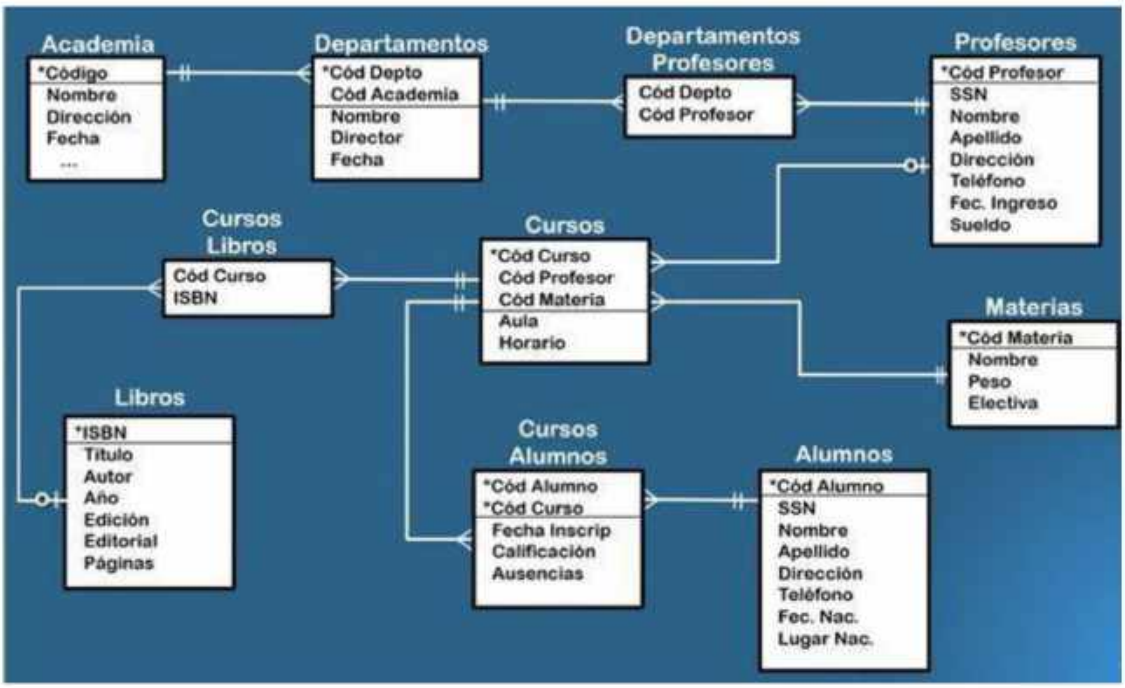


Gráfico 14. Entidad Relación de un sistema de Curso Virtual

1. CREAR PROCEDIMIENTOS ALMACENADOS A CADA UNA DE LAS SIGUIENTES CONSULTAS:

1.1. Obtener el listado de todos los profesores y ordenado Ascendentemente.
1.2. Obtener el listado de todos los alumnos y ordenado Descendentemente.
1.3. Obtener el listado de los alumnos que reciben la asignatura de Programación Visual.
1.4. Obtener el listado de los alumnos que reciben la asignatura de Base de datos avanzado y en que aula lo reciben.
1.5. Obtener el listado de los alumnos que reciben la asignatura de Diseño Multimedia y el nombre del profesor que lo imparte.
1.6. Obtener el listado de los alumnos que empiecen sus apellidos con la letra "S"
1.7. Obtener los datos del profesor que imparte la asignatura de Programación Visual, Diseño Multimedia y Calculo Diferencial e Integral
1.8. Obtener el nombre del profesor que imparte la catedra en el laboratorio 2 y el nombre de la asignatura.
1.9. Obtener el nombre del libro y a que asignatura pertenece.
1.10. Obtener el nombre del libro, el aula, del profesor que imparte la asignatura de base de datos.
1.11. Obtener el nombre del libro que fue publicado en el año 2010.
1.12. Obtener el nombre del profesor que su sueldo sea mayor a 1212.
1.13. Obtener el nombre del profesor que su sueldo este entre 1212 y 986.
1.14. Obtener los datos del estudiante que su calificación sea muy buena.

1.15. Quiero obtener el listado de los estudiantes que no recibe la materia de Base de Datos Avanzado.

1.16. Obtener el listado de los estudiantes que reciben la asignatura Fundamentos de Administración y Base de Datos Avanzado, también quiero saber el nombre del profesor y el nombre del libro.

ACTIVIDAD 2

Crear un índice para inserte un dato único

ACTIVIDAD 3

Crear un índice para que almacene dos letras en el índice y así realice más rápido una consulta

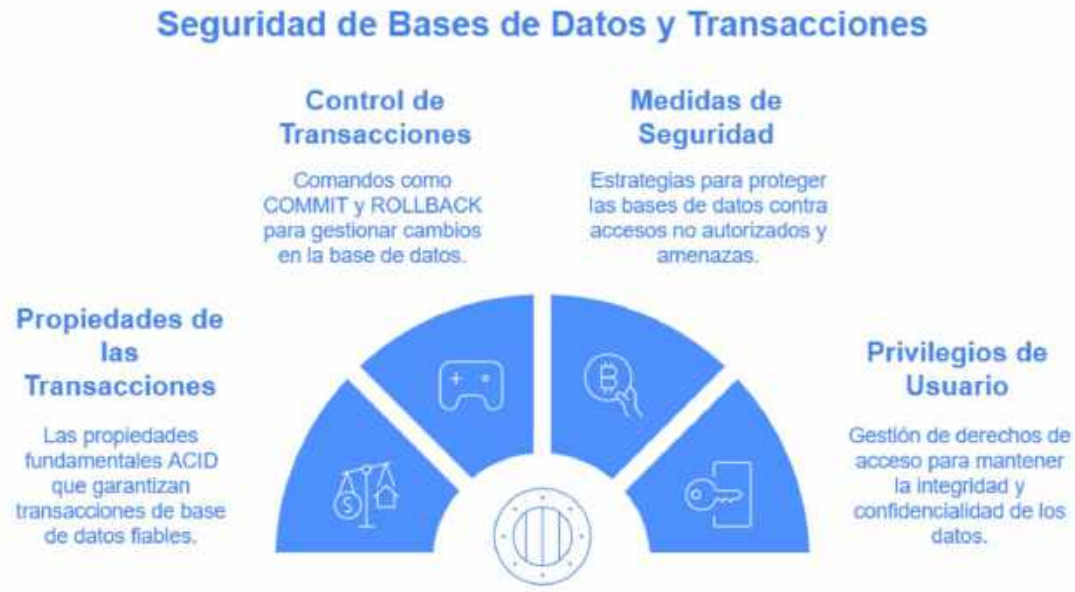
UNIDAD 4: TRANSACCIONES Y SEGURIDAD

4.1 Transacciones
4.2 Seguridad

Resultado de Aprendizaje

Los estudiantes deben ser capaces de implementar transacciones en bases de datos, como también desarrollar habilidades para implementar medidas de seguridad en bases de datos, incluyendo la gestión de usuarios y permisos, así como la protección contra accesos no autorizados.

DIAGRAMA DE APRENDIZAJE



SÍNTESIS

La unidad "Transacciones y Seguridad" se centra en dos conceptos clave en la gestión de bases de datos: las transacciones y la seguridad de la información. A continuación, se presenta un resumen de los temas principales abordados en esta unidad.

4.1.Transacciones

4.1.1. Transacciones (Commit, rollback), concurrencias.

1. ¿Qué son las transacciones?

"Las transacciones en SQL son unidades o secuencias de trabajo realizadas de forma ordenada y separada en una base de datos. Normalmente representan cualquier cambio en la base de datos, y tienen dos objetivos principales:

- Proporcionar secuencias de trabajo fiables que permitan poder recuperarse fácilmente ante errores y mantener una base de datos consistente incluso frente a fallos del sistema.
- Proporcionar aislamiento entre programas accediendo a la vez a la base de datos.

Una transacción es una propagación de uno o más cambios en la base de datos, ya sea cuando se crea, se modifica o se elimina un registro. En la práctica suele consistir en la agrupación de consultas SQL y su ejecución como parte de una transacción.

2. Propiedades de las transacciones

Las transacciones siguen cuatro propiedades básicas, bajo el acrónimo ACID (Atomicity, Consistency, Isolation, Durability):

- Atomicidad: aseguran que todas las operaciones dentro de la secuencia de trabajo se completen satisfactoriamente. Si no es así, la transacción se abandona en el punto del error y las operaciones previas retroceden a su estado inicial.
- Consistencia: aseguran que la base de datos cambie estados en una transacción exitosa.
- Aislamiento: permiten que las operaciones sean aisladas y transparentes unas de otras.
- Durabilidad: aseguran que el resultado o efecto de una transacción completada permanezca en caso de error del sistema.

3. Control de las transacciones

Existen tres comandos básicos de control en las transacciones SQL:

- **COMMIT.** Para guardar los cambios.
- **ROLLBACK.** Para abandonar la transacción y deshacer los cambios que se hubieran hecho en la transacción.
- **SAVEPOINT.** Crea checkpoints, puntos concretos en la transacción donde poder deshacer la transacción hasta esos puntos.

Los comandos de control de transacciones se usan sólo con INSERT, DELETE y UPDATE. No pueden utilizarse creando tablas o vaciándolas porque las operaciones se guardan automáticamente en la base de datos". (Lázaro, 2018)

Ejemplo:

BEGIN;

INSERT INTO alumno (cedula,nombre,apellido) VALUE ("1500141414", "pato", "patricio");

SELECT * FROM alumno;

COMMIT; // para aceptar la transacción ó

ROLLBACK; //si quieres cancelar la transacción

4. ¿Qué es Concurrencia?

Gutiérrez (2009) afirma: Es cuando muchas transacciones acceden a la misma Base de Datos al mismo tiempo. Especialmente, cuando acceden a los mismos datos de la misma Base de Datos al mismo tiempo (pág. 5).

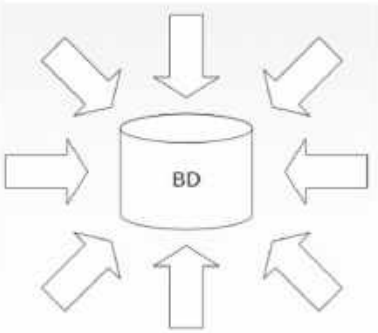


Gráfico 15. Base de Datos

ACTIVIDADES DE LA UNIDAD 4

ACTIVIDAD 1: Transacciones SQL

Realizar una transferencia de dinero entre cuentas bancarias (por ejemplo 100 euros)

Contenidos de la Unidad 5

4.1.SEGURIDAD

4.1.1. Seguridad en base de datos

"Definición de un esquema de seguridad:

Al concepto de seguridad también se le puede llamar privacidad.

El problema de la seguridad consiste en lograr que los recursos de un sistema sean, bajo toda circunstancia, utilizados para los fines previstos.

La fiabilidad del sistema:

El concepto de seguridad lo medimos en:

- La protección del sistema frente a ataques externos.
- La protección frente a caídas o fallos en el software o en el equipo.
- La protección frente a manipulación por parte de usuarios no autorizados.

Principios básicos para la seguridad:

Suponer que el diseño del sistema es público:

- El defecto debe ser: sin acceso.
- Chequear permanentemente.
- Los mecanismos de protección deben ser simples, uniformes y construidos en las capas más básicas del sistema.

Medidas de seguridad:

FÍSICAS: Controlar el acceso al equipo.

- Mediante tarjetas de acceso…

PERSONAL: Acceso solo de personal autorizado.

- Identificación directa de personal…

SGBD: Uso de herramientas que proporcione el SGBD

- Perfiles de usuario, vistas, restricciones de uso de vista.

4.1.1. Hay dos tipos de seguridad:

- **Direccional**

  Se usa para otorgar y revocar privilegios a los usuarios a nivel de archivos, registros, campos en un modo determinado (consulta o modificación).

- **Obligatoria**

  Sirve para imponer seguridad de varios niveles tanto para los usuarios como para los datos.

Para eso se utilizan mecanismos de protección.

4.1.2. Requisitos para la seguridad de las bd:

- La base de datos debe ser protegida contra el fuego, el robo y otras formas de destrucción.
- Los datos deben ser reconstruibles, ya que siempre pueden ocurrir accidentes.
- Los datos deben poder ser sometidos a procesos de auditoría.
- El sistema debe diseñarse a prueba de intromisiones, no deben poder pasar por alto los controles.
- Ningún sistema puede evitar las intromisiones malintencionadas, pero es posible hacer que resulte muy difícil eludir los controles.
- El sistema debe tener capacidad para verificar que sus acciones han sido autorizadas.
- Las acciones de los usuarios deben ser supervisadas, de modo tal que pueda descubrirse cualquier acción indebida o errónea.

Características principales:

El objetivo es proteger la Base de Datos contra accesos no autorizados.

Las 3 principales características de la seguridad en una base de datos son:

- La Confidencialidad de la información
- La Integridad de la información
- La Disponibilidad de la información

Tipos de usuarios:

Hay dos tipos de usuarios:

- Usuario con derecho a crear, borrar y modificar objetos y que además puede conceder privilegios a otros usuarios sobre los objetos que ha creado.

- Usuario con derecho a consultar, o actualizar, y sin derecho a crear o borrar objetos. Privilegios sobre los objetos, añadir nuevos campos, indexar, alterar la estructura de los objetos, etc.

Identificación y autentificación:

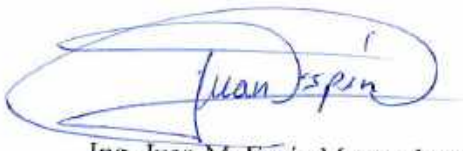
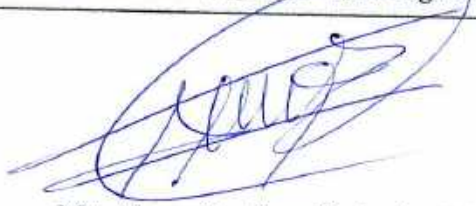
En un SGBD existen diversos elementos que ayudan a controlar el acceso a los datos.

En primer lugar, el sistema debe identificar y autentificar a los usuarios utilizando alguno de las siguientes formas:

- Código y contraseña
- Identificación por hardware
- Conocimiento, aptitudes y hábitos del usuario
- Información predefinida (Aficiones, cultura…)

Además, el administrador deberá especificar los privilegios que un usuario tiene sobre los objetos:

- Usar una B.D.
- Consultar ciertos datos
- Actualizar datos
- Crear o actualizar objetos
- Ejecutar procedimientos almacenados
- Referenciar objetos
- Indexar objetos
- Crear identificadores". (gplsi.dlsi.ua.es)

ELABORACIÓN, REVISIÓN Y APROBACIÓN DE PARES	
Profesor(a)	
 Ing. Juan M. Espín Montesdeoca, Mg.	
Fecha de elaboración: 01/8/2023	
Comisión de revisión de pares de guías de estudio del Instituto Superior Tecnológico Tena	
 Lcdo. Segundo Calisto Rochina Chileno	 Mg. Alvaro Santiago Toalombo Díaz
 Mg. Henry Fabian Chango Chango	 Ing. Agustin Gonzalo Guanipatin Ramirez
Fecha de revisión: 04/09/2023	
Coordinador de Investigación, Desarrollo Tecnológico e Innovación	
 Mg. Danilo Alexander Zamora Núñez	
Fecha de aprobación: 02/10/2023	