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

20250043

BASE DE DATOS AVANZADO

Período académico

Segundo

Ing. Patricio Guanipatin.

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

INSTITUTO SUPERIOR TECNOLÓGICO TENA

Carrera de Tecnología en Desarrollo de Software

ISTT DSW Primera Edición – Tena, noviembre 2025

SIN ISBN

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

AUTOR - REDACCIÓN Y FORMULACIÓN DE CONTENIDOS
Ing. Patricio Guanipatin

Profesores del Instituto Superior Tecnológico Tena

REVISIÓN DE PARES

Mg. Alvaro Santiago Toalombo Díaz
Mg. Henry Fabian Chango Chango
Mg. Martha Janina Duarte Mora
Mg. Danilo Alexander Zamora Núñez
Lcda. María Angélica Campoverde Encalada

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.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE GRÁFICOS

GUIA GENERAL DE ESTUDIO DE LA ASIGNATURA

1. DATOS GENERALES DE LA ASIGNATURA							
Carrera	Tecnología en Desarrollo de Software			**Nombre asignatura**	Base de datos avanzado		
Modalidad	Presencial			**Campo de Formación**	Adaptación e Innovación Tecnológica		
Jornada	Matutina/Nocturna			**Unidad de Organización Curricular**	Profesional		
Período académico	Segundo			**Código de la asignatura**	DSW-205		
Ciclo académico	2024 IIS			**N° Total de horas de la asignatura**	192		
Distribución de horas en las actividades de aprendizaje							
N° de horas Docencia	**64**	**N° de horas Aprendizaje Práctico Experimental**				**N° de horas Autónomo**	**48**
		En contacto con docente	**32**	**Autónomo**	**48**		

2. PRERREQUISITOS Y CORREQUISITOS			
Prerrequisitos de la asignatura		**Correquisitos de la asignatura**	
Asignatura	**Código**	**Asignatura**	**Código**

4. 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.

5. OBJETIVO GENERAL

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

6. EJE TRANSVERSAL

Eje Transversal	Temáticas	Descripción
Formación ciudadana integral	Valores y habilidades blandas	Desarrollo de valores humanos universales, el cumplimiento de las obligaciones ciudadanas, la toma de conciencia de los derechos, el desarrollo de la identidad y el respeto, el aprendizaje de la convivencia dentro de una sociedad intercultural y plurinacional, la tolerancia hacia las ideas y costumbres de los demás y el respeto a las decisiones de la mayoría.
	Conciencia ambiental	Incentivar el mejoramiento y protección del medio ambiente para promover el desarrollo sostenible nacional en armonía con los derechos de la naturaleza constitucionalmente reconocidos, gentes de cambio que contribuyan a la construcción de la sociedad más sostenible.

7. 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)
Diseña de bases de datos eficientes, seguras y escalables, utilizando herramientas y tecnologías modernas.	Brinda asistencia técnica en el diseño, modelamiento e ilustración del proceso de base de datos.	Alta
Administra bases de datos mediante procesos de control y seguimiento de las operaciones para el manejo adecuado de la información.	Brinda asistencia técnica en el diseño de bases de datos mediante procesos de control y	Media

	seguimiento de las operaciones para el manejo adecuado de la información.	

9. CONTENIDOS DE LA ASIGNATURA (DISTRIBUCIÓN DE CONTENIDOS POR SEMANA, FECHA Y HORAS)

Unidad 1
1. Sentencias e índices.
1.1. Sentencias DDL y DML complejas
1.2. Implementación de índices
1.3. Ejercicios resueltos
1.4. Ejercicios propuestos

Unidad 2
2. Generación de Trigger y Vistas.
2.1. Implementación de Trigger, estructura básica.
2.2. Ejercicios resueltos
2.3. Creación de vistas
2.4. Ejercicios resueltos
2.5. Ejercicios propuestos

Unidad 3
3. Generación de procedimientos, índices y cursores.
3.1. Procedimientos almacenados o funciones.
3.2. Ejercicios resueltos
3.3. Índices y Cursores
3.4. Ejercicios resueltos
3.5. Ejercicios propuestos

Unidad 4
4. Transacciones y Seguridad
4.1. Transacciones (Commit, rollback), concurrencias
4.2. Ejercicios resueltos
4.3. Seguridad en base de datos
4.4. Ejercicios resueltos
4.5. Ejercicios propuestos

10. 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): vídeos
NUEVAS TECNOLOGÍAS	*Programas informáticos: procesador de palabras, hojas de cálculo, presentaciones*
	Servicios telemáticos: páginas web, plataforma EVA, correo electrónico, chats

11. EVALUACIÓN DEL ESTUDIANTE POR RESULTADOS DE APRENDIZAJE

PARCIAL	COMPONENTES DE EVALUACIÓN				
	EVALUACIÓN FORMATIVA			EVALUACIÓN SUMATIVA	TOTAL
	Actividades en el aula de clase	Actividades de refuerzo académico	Actividades prácticas y experimentales	Evaluación	
PRIMER PARCIAL	2.0	3.0	3.0	2.0	10
SEGUNDO PARCIAL	2.0	3.0	3.0	2.0	10
				PROMEDIO FINAL	10
				NOTA DE RECUPERACIÓN	*
				CALIFICACIÓN FINAL	10

CONSIDERACIONES ESPECIALES

Asistencia

- Es responsabilidad de los docentes del ISTT constatar la asistencia de los estudiantes y registrarla en el SIGA.
- Los estudiantes deben cumplir con un porcentaje mínimo del 70% de asistencia, para la aprobación respectiva de la asignatura, curso o su equivalente, como complemento a las calificaciones obtenidas en el período académico.
- Los estudiantes que tengan un porcentaje mayor al 30% de inasistencias, reprobarán automáticamente la asignatura, curso o su equivalente.
- Es responsabilidad del estudiante asistir a todas las evaluaciones y actividades académicas de las cuales se derive una evaluación; de no hacerlo, se consignará la nota de cero (0,00).

Escala Valorativa

- La escala de valoración del ISTT comprende el rango de cero (0) a diez (10) puntos.
- Todas las actividades de evaluación de los aprendizajes serán calificadas sobre diez (10) puntos; y, los cómputos de las notas finales de cada parcial, previo al registro en la plataforma del Sistema Integrado de Gestión Académica (SIGA), se calcularán respetando la ponderación descrita, tanto para evaluaciones formativas y sumativas.
- El puntaje máximo de aprobación de cada asignatura, curso o su equivalente será sobre 10 puntos.
- El puntaje mínimo de aprobación de una asignatura, curso o su equivalente será de siete (7,00) puntos.
- Si un estudiante obtiene un puntaje final de la asignatura, curso o su equivalente inferior a tres (3,00) puntos, como resultado del promedio de los dos parciales, reprobará de forma automática la asignatura, curso o su equivalente, sin tener la posibilidad de estar habilitado a la actividad evaluativa de recuperación.

Recuperación Final

- El ISTT considera una actividad evaluativa de recuperación de fin de curso para los estudiantes que no alcancen el puntaje mínimo de aprobación de las asignaturas, curso o sus equivalentes.
- Los estudiantes que poseen en la asignatura, curso o su equivalente un promedio final en el rango de 3,01 a 6,99 sobre diez (10) puntos, como resultado del promedio de los dos parciales de una asignatura, curso o su equivalente, estarán habilitados para rendir una actividad evaluativa de recuperación.
- El ISTT considera actividades tutoriales de acompañamiento y refuerzo académico previo a la actividad evaluativa de recuperación.
- Las actividades evaluativas de recuperación se calificarán sobre un máximo de 10 puntos.
- Las actividades evaluativas de recuperación únicamente se considerarán al obtener una nota mínima de siete puntos sobre diez (7/10), caso contrario, no será considerado y reprobará automáticamente la asignatura, curso o su equivalente.
- La ponderación válida del puntaje para la recuperación será del 50% de la nota obtenida.

La nota final de una asignatura, curso o su equivalente en la cual el estudiante ha rendido una actividad evaluativa de recuperación se obtiene de la siguiente manera:

- El promedio obtenido como resultado de los dos parciales, se suma a la nota de la evaluación de recuperación, ponderada por el 50% de lo obtenido en la actividad evaluativa de recuperación.
- La nota final de una asignatura, curso o su equivalente obtenido, una vez rendida la actividad evaluativa de recuperación, no permite las aproximaciones al inmediato superior en caso de que ésta tenga decimales.
- Los estudiantes que reprueban una asignatura, curso o su equivalente por inasistencia, retiro o deserción, no estarán habilitados para rendir la actividad evaluativa de recuperación.

	• Los estudiantes que cursan una asignatura, curso o su equivalente con tercera matrícula no estarán habilitados para rendir la actividad evaluativa de recuperación, y la nota final se obtendrá del promedio de las calificaciones obtenidas del primero y segundo parcial.

12. BIBLIOGRAFÍA:

Bibliografía Básica de la Asignatura:	**Físico**	**Digital**
Silberschatz, A., Korth, H. F., & Sudarshan, S. (2006). *Programación de base de datos relacionales.* (Quinta Edición). McGraw-hill/Interamericana. España. **ISBN:** 978-84-481-5671-8, **Número de inventario en biblioteca:** ISTT-DS-0097	X	
Martínez López, F. J., & Gallegos Ruiz, A. (2017). *Programación de base de datos relacionales.* (Primera Edición). Ra-ma. Colombia. **ISBN:** 978-958-762-684-1, **Número de inventario en biblioteca:** ISTT-DS-0027	X	
Bibliografía de consulta de la Asignatura:	**Físico**	**Digital**
Castaño, M., Piattini, M., & Esperanza, M. (2000). *Diseño de base de datos relacionales.* (Primera Edición). Alfaomega Grupo Editorial S.A. México. **ISBN:** 970-15-0526-3, **Número de inventario en biblioteca:** ISTT-DS-0031	X	
Pérez Marqués, M. (2016). *Administración básica de base de datos con Oracle 12 c SQL Prácticas y Ejercicios.* (Primera Edición). Alfaomega. Colombia. **ISBN:** 978-958-778-202-8, **Número de inventario en biblioteca:** ISTT-DS-0057	X	

DESCRIPTIVA DE LAS COMPETENCIAS DE LA GUÍA DE ESTUDIO DE BASE DE DATOS

La guía de estudio está orientada al desarrollo de competencias específicas, generales y transversales que permitan al estudiante adquirir los conocimientos y habilidades necesarias para diseñar, implementar y optimizar bases de datos. Estas competencias se han estructurado con base en las unidades temáticas y responden a las demandas actuales del campo profesional.

COMPETENCIAS ESPECÍFICAS

Unidad 1: Sentencias e índices. (DDL & DML)

Aplicación de las sentencias en la creación y ejecución de los procesos en la base de datos.

Unidad 2: Generación de Trigger y Vistas.

Creación de trigger para implementación de seguridad en cada proceso.

Unidad 3: Generación de procedimientos, índices y cursores.

Administra bases de datos mediante procesos de control y seguimiento de las operaciones para el manejo adecuado de la información.

Unidad 4: Transacciones y Seguridad.

Diseña de bases de datos eficientes, seguras y escalables, utilizando herramientas y tecnologías modernas.

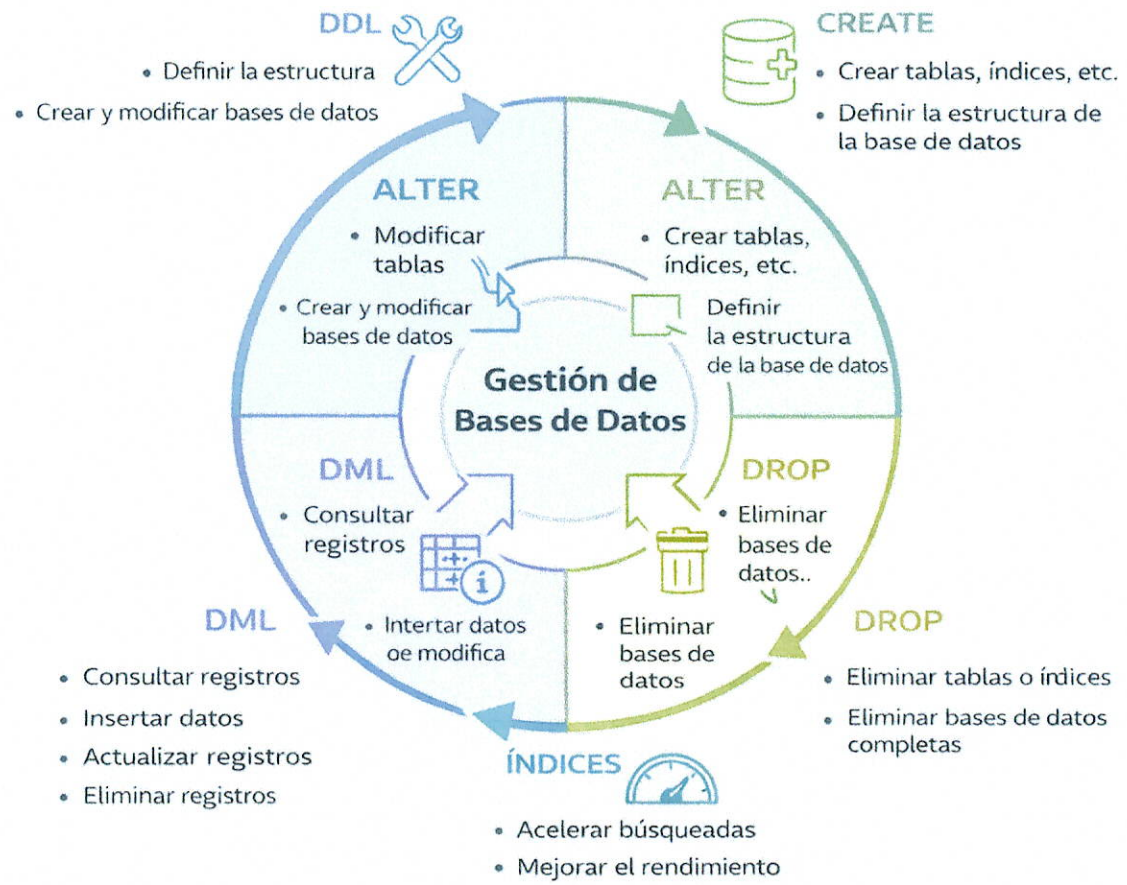
UNIDAD 1: Sentencias e índices (DDL & DML)

1. Sentencias DDL complejas
2. Implementación de Índices

Resultado de aprendizaje

Diseña de bases de datos eficientes, seguras y escalables, utilizando herramientas y tecnologías modernas.

DIAGRAMA DE APRENDIZAJE



SÍNTESIS

Esta unidad permite comprender la forma de crear la base de datos utilizando sentencias DDL y la manipulación de información utilizando las sentencias DML, y así a los estudiantes incorporar el hábito de programar.

1. Sentencias DDL complejas

Arévalo (2013) afirma: Los DDL (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] https://geotalleres.readthedocs.io/es/latest/conceptos-sql/conceptos_sql.html

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),
Edad INT
)
```

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**;

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';

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';

1.1. 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 tabla[2]

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

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;

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");

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';

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';

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 FROM[3]

Clausula FROM:

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

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

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;

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;

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

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

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');

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	Especificar[5]

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

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;

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]

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

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).

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**:

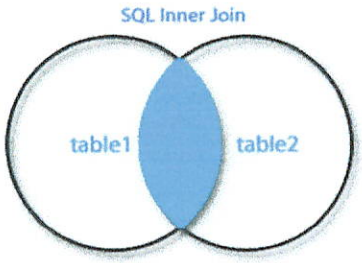


Gráfico 1. SQL Inner Join
Desarrollado por: Lázaro diego

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:

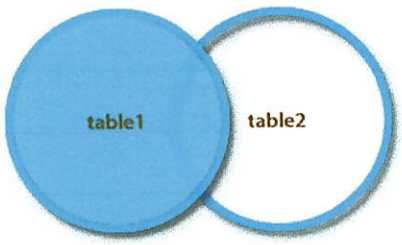


Gráfico 2. SQL Left Join

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:

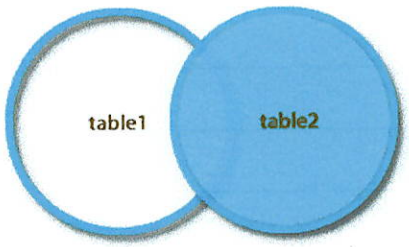


Gráfico 3. Right Join

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:

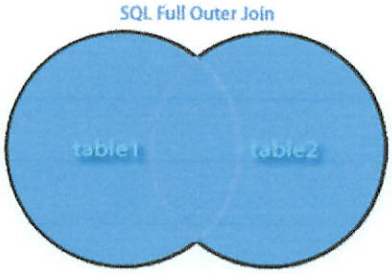


Gráfico 4. SQL Outer Join

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)

1.2. 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)

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;`

ACTIVIDADES DE LA UNIDAD 1

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.

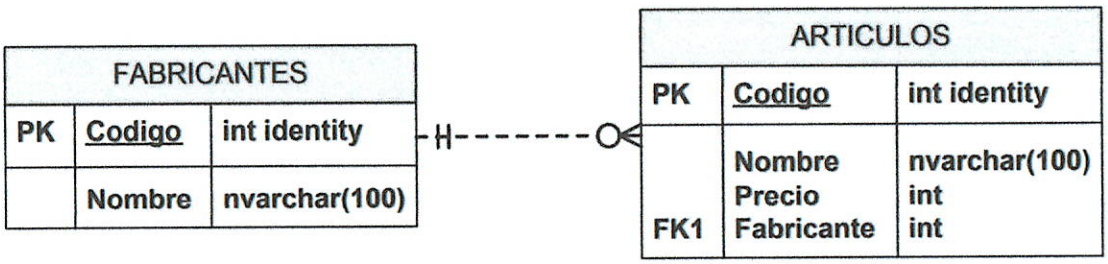


Gráfico 5. Relación entre Fabricantes y Artículos

ACTIVADED 3

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

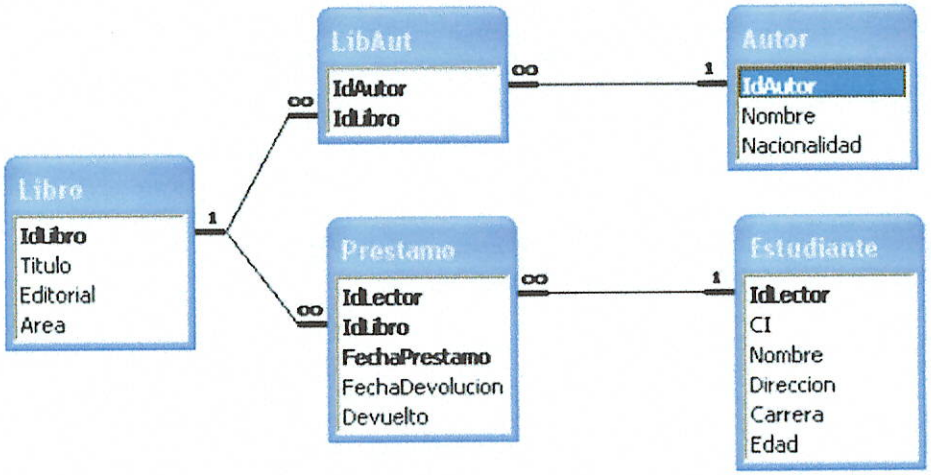


Gráfico 6. Entidad Relación de una Bibliotéca

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	JEFERSO N 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 bolívar	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

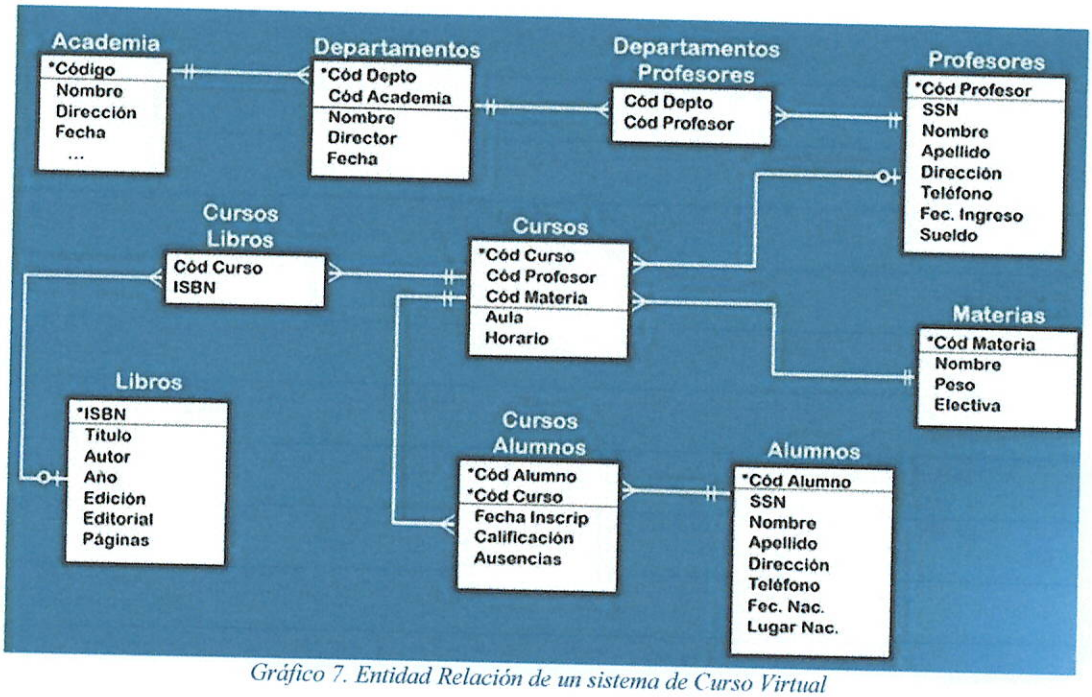


Gráfico 7. Entidad Relación de un sistema de Curso Virtual

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

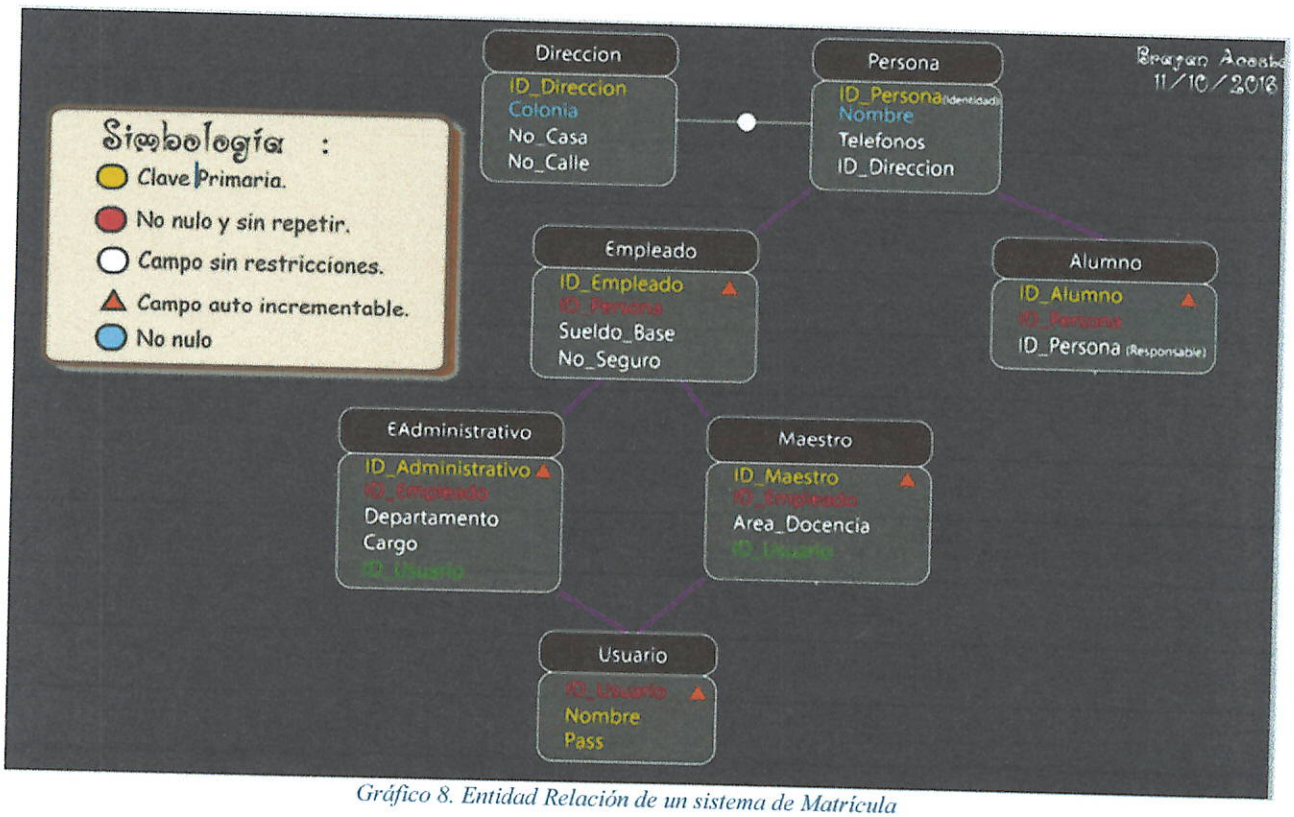


Gráfico 8. Entidad Relación de un sistema de Matrícula

UNIDAD 2: Generación de Trigger y Vistas

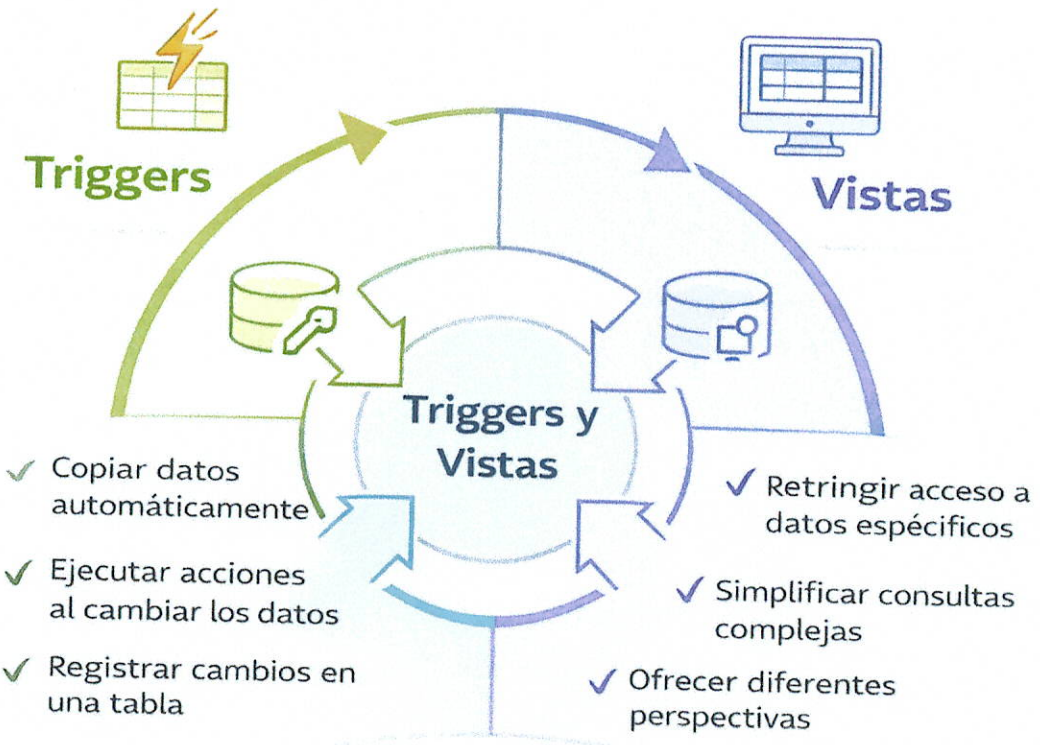
2.1. Implementación de Trigger, estructura básica

2.2. Creación de vistas

Resultado de aprendizaje

Administra bases de datos mediante procesos de control y seguimiento de las operaciones para el manejo adecuado de la información.

DIAGRAMA DE APRENDIZAJE



SÍNTESIS

Esta unidad permite comprender la forma de implementar el Triggers, para que se ejecute por cada acción que se realice a las tablas de la base de datos y también la creación de vistas, para

emplearse como mecanismos de seguridad, que permiten a los usuarios obtener acceso a los datos por medio de la vista.

2. Trigger, Vistas

2.1. 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:

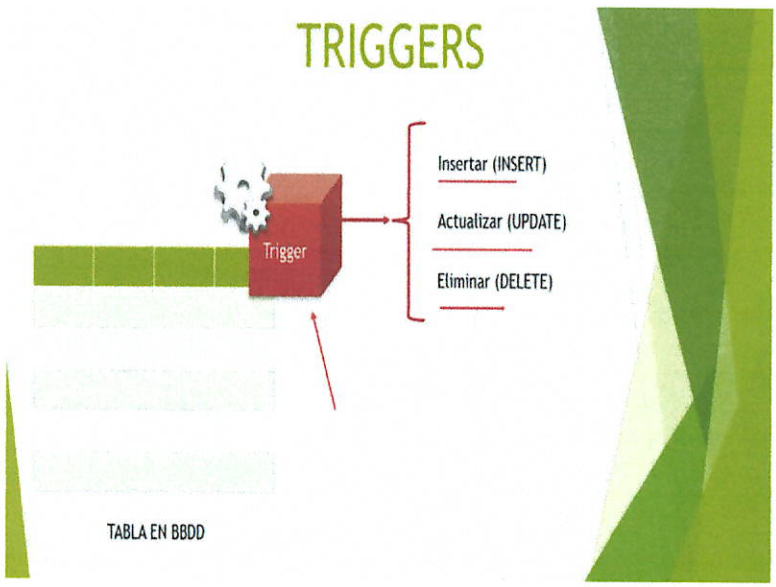


Gráfico 9. En las acciones que se ejecuta el Trigger

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

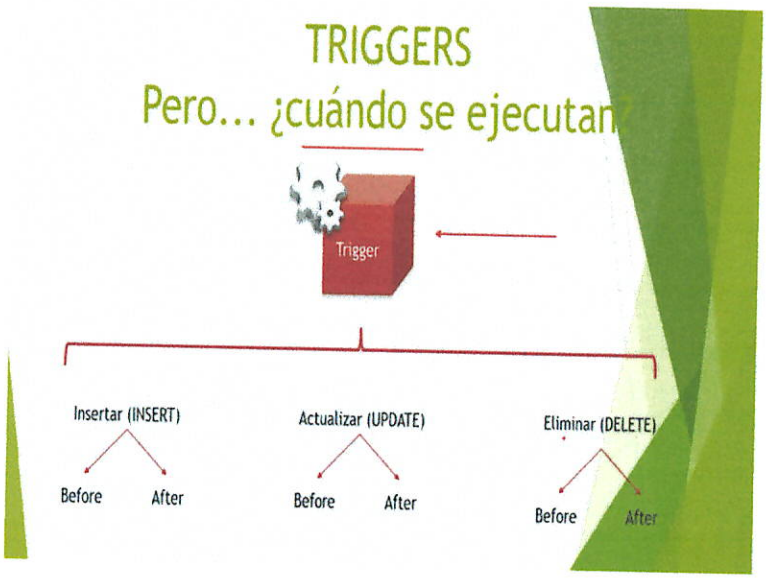


Gráfico 10. Cuando se ejecuta el Trigger en cada acción

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

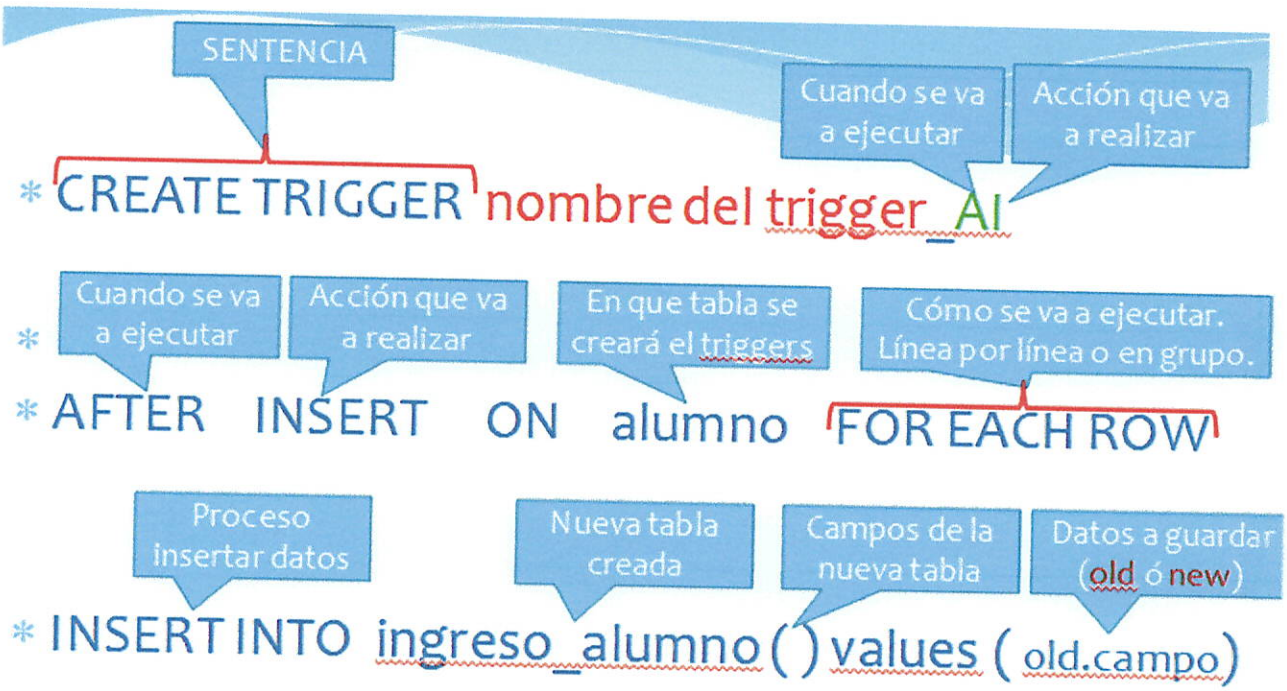


Gráfico 11. Sintaxis de creación de Triggers

Elaborado por: Autor

ELIMINAR UN TRIGGERS

DROP TRIGGER nombre del trigger;

2.2. 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

ACTVIDAD 1

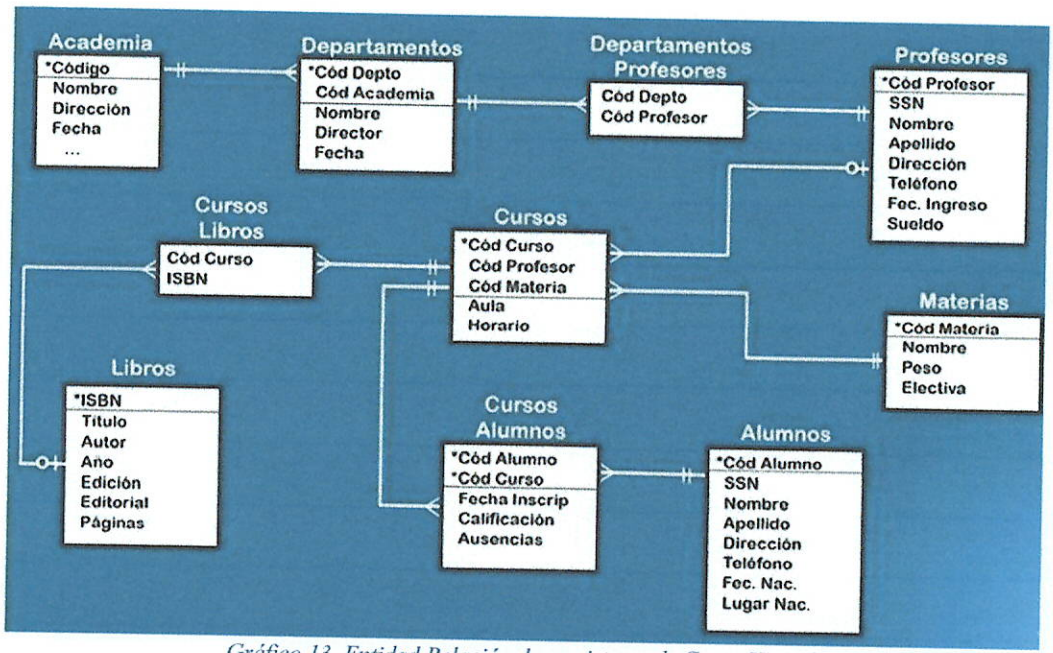


Gráfico 13. Entidad Relación de un sistema de Curso Virtual

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.

UNIDAD 3: Generación de procedimientos, índices y cursores

3.1. Procedimientos almacenados o funciones.

3.2. Índices y Cursores

Resultado de aprendizaje

Administra bases de datos mediante procesos de control y seguimiento de las operaciones para el manejo adecuado de la información.

DIAGRAMA DE APRENDIZAJE

SÍNTESIS

Esta unidad permite comprender la forma de crear los procedimientos almacenados para poder reutilizarlo los códigos en cada proceso que se quiera realizar y también la implementación de índices, para así 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.

3. Procedimientos, índices y cursores

3.1. Procedimientos almacenados o funciones.

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

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.

3.2. 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;
```

ACTIVIDADES DE LA UNIDAD 3

INDIVIDUAL

ACTIVIDAD 1

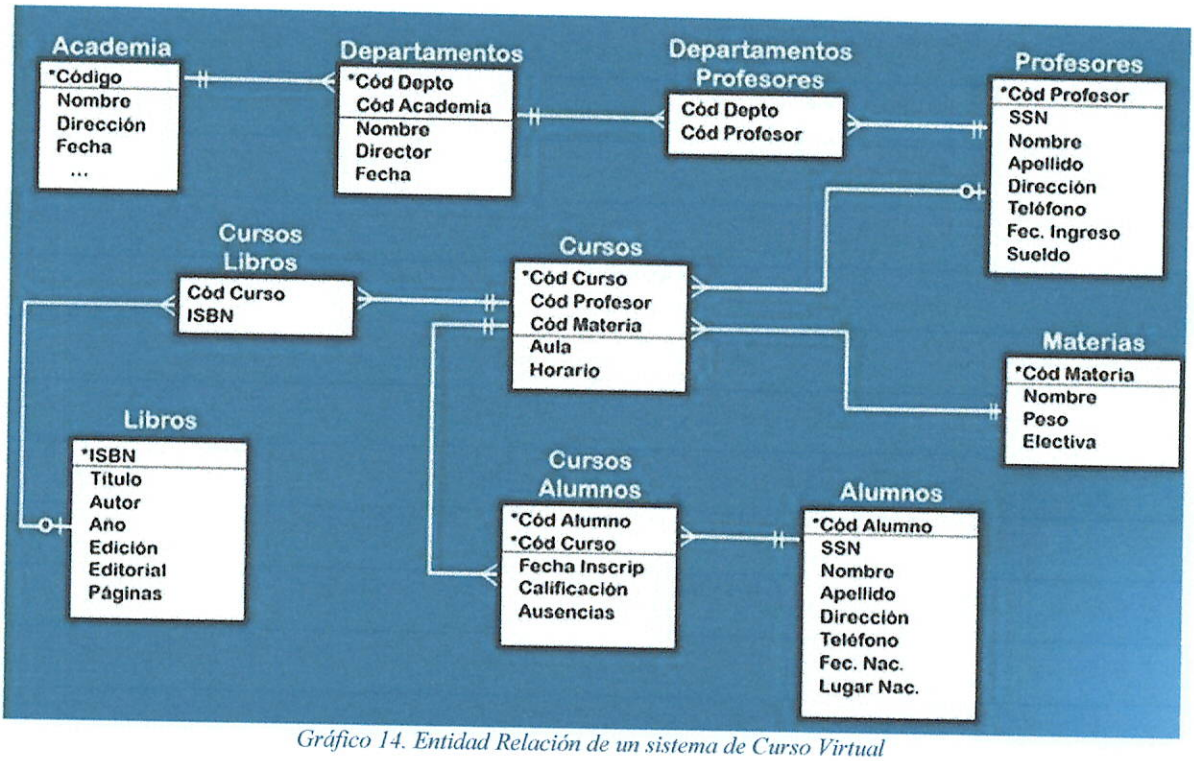


Gráfico 14. Entidad Relación de un sistema de Curso Virtual

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

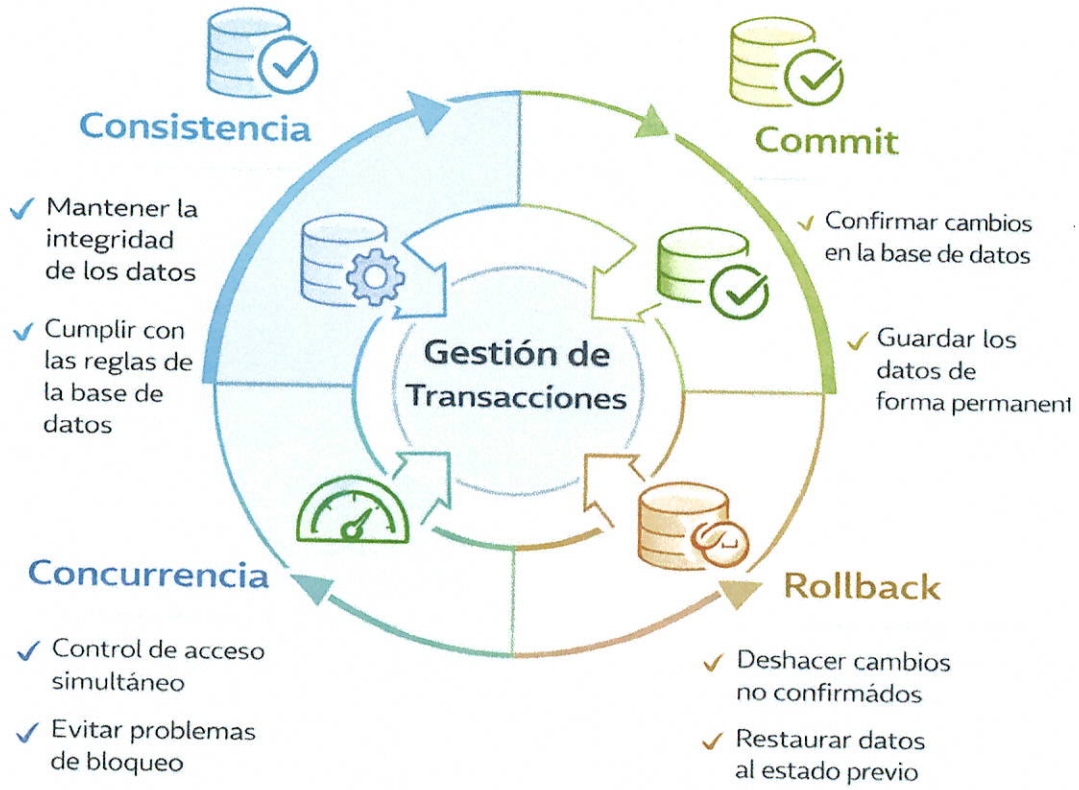
UNIDAD 4: Transacciones y Seguridad

4.1. Transacciones (Commit, rollback), concurrencias.

Resultado de aprendizaje

Administra bases de datos mediante procesos de control y seguimiento de las operaciones para el manejo adecuado de la información.

DIAGRAMA DE APRENDIZAJE



SÍNTESIS

Esta unidad permite comprender la forma de como procesa las transacciones en las bases de datos y así brindar confianza en cada proceso que realice en la base de datos.

4. Transacciones

4.1. Transacciones (Commit, rollback), concurrencias.

¿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

¿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).

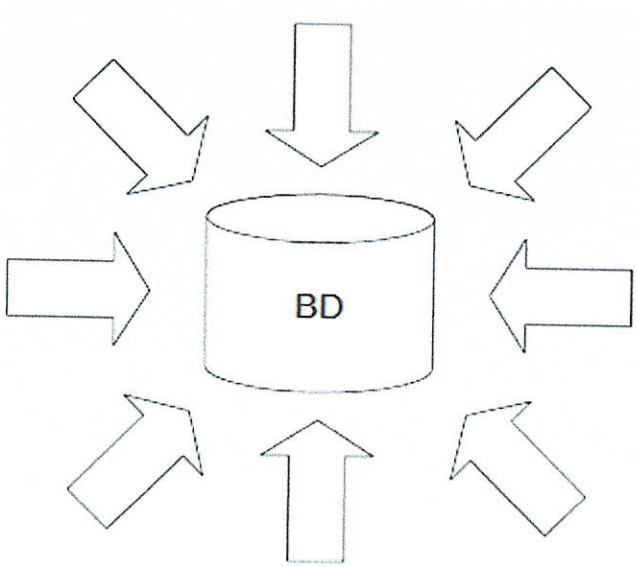


Gráfico 15. Base de Datos

4.2. Seguridad en base de datos utilizando tecnologías emergentes y en tendencias.

La seguridad de bases de datos abarca la protección de la confidencialidad, integridad y disponibilidad (CIA) de los datos almacenados en los sistemas de gestión de bases de datos (SGBD). Pero hoy, con tecnologías emergentes (nube, multi-modelo, edge, blockchain, IA, Zero Trust, etc.) y nuevas amenazas (persistentes, sofisticadas, ransomware centrado en bases de datos),

la temática evoluciona rápidamente. Por lo tanto, es clave conocer tanto los retos como las tendencias tecnológicas en base de datos y seguridad.

Retos y escenarios actuales

- Amenazas de inyección (por ejemplo SQLi), acceso privilegiado, insiders.
- Bases de datos en entornos de nube, edge, distribuidos: complican la supervisión, el control, la latencia, la replicación segura.
- Tecnologías emergentes que requieren nuevos enfoques: bases de datos multi-modelo, serverless, blockchain para registro inmutable, etc.

Tendencias / tecnologías emergentes relevantes

- **Arquitectura Zero Trust:** "nunca confiar, siempre verificar", acceso mínimo, verificación continua.
- **Monitoreo de actividad de base de datos (Database Activity Monitoring, DAM):** permite auditar y detectar anomalías en tiempo real.
- **Cifrado de datos en reposo y en tránsito,** tokenización, mascaramiento dinámico de datos.
- **Bases de datos inmutables / blockchain-inspiradas para auditoría:** especialmente para entornos donde la trazabilidad es crítica.
- **Edge / distribuido / multi-nube:** asegurar bases de datos que no están solo en un centro de datos centralizado.

- **Inteligencia artificial / aprendizaje automático para detección de anomalías** en acceso a bases de datos, patrones de uso atípicos.

Buenas prácticas generales

- Aplicar el principio de menor privilegio para usuarios, roles, servicios.
- Separar roles administrativos y operativos, registrar cada acceso.
- Segmentar las bases de datos sensibles, usar redes privadas, firewall perimetral + controles internos.
- Proteger los backups y replicaciones (ver sección siguiente) porque un backup comprometido es una vía de fuga.
- Realizar auditorías periódicas, pruebas de penetración (pentesting) específicas para bases de datos.
- Integrar seguridad en el ciclo de vida (desde diseño, desarrollo, operaciones).

4.3. Backup de base de datos

El backup de base de datos es la práctica de crear copias de los datos (y, en muchos casos, del estado del sistema) para poder restaurarlos ante pérdida, corrupción, ataque o desastre. Es un pilar esencial de la estrategia de recuperación de desastres (DR) y continuidad del negocio.

Métodos y tipos de backup

- **Backup completo (Full):** copia de todos los datos.
- **Backup incremental:** sólo los datos que han cambiado desde el último backup (completo o incremental).

- **Backup diferencial:** cambios desde el último backup completo.
- **Mirror backup, snapshots,** backup en línea (hot backup) vs offline (cold backup) según el motor de base de datos.

Consideraciones de seguridad en el backup

- Los backups también deben protegerse: cifrado, acceso restringido, almacenamiento en ubicación segura.
- Mantener múltiples copias, idealmente siguiendo la regla 3-2-1: 3 copias de los datos, 2 medios distintos, 1 fuera del sitio.
- Verificar que los backups funcionan (pruebas de restauración).
- Documentar RTO (Recovery Time Objective) y RPO (Recovery Point Objective).

Tendencias y buenas prácticas actuales

- Automatización de backups en entornos de nube y bases de datos distribuidas.
- Uso de snapshots y replicación continua para reducir ventana de backup.
- Integración del backup con la estrategia de seguridad: por ejemplo, backups inmutables para protección contra ransomware.
- Gestión de backups para bases de datos "como servicio" (DBaaS) en la nube, asegurando que el proveedor cumple con estándares de seguridad.

Ejemplo: Backup completo de una base de datos

```
mysqldump -u root -p nombre_base_datos > backup_nombre_base_datos.sql
```

- **-u root:** usuario de MySQL.
- **-p:** pedirá la contraseña.
- **nombre_base_datos:** nombre de tu base de datos.
- **>:** redirecciona la salida al archivo .sql.

Ejemplo práctico

mysqldump -u root -p empresa > backup_empresa_2025.sql

Restaurar un backup

Para restaurar el archivo .sql:

mysql -u root -p nombre_base_datos < backup_nombre_base_datos.sql

Ejemplo práctico:

mysql -u root -p empresa < backup_empresa_2025.sql

4.4. Fundamentos para la integración de base de datos en aplicaciones

La integración de base de datos en aplicaciones se refiere a cómo las aplicaciones consumen, manipulan y exponen los datos almacenados en uno o varios sistemas de bases de datos. Aquí se incluyen aspectos como el diseño de la base de datos, la conexión, la abstracción, el acceso, la sincronización, la interoperabilidad, la integración de datos provenientes de múltiples fuentes.

Conceptos fundamentales

- **Modelado de datos:** diseño lógico/físico conforme a los requisitos de la aplicación. Ver Fundamentals of Database Systems como referencia.

- **Arquitectura de acceso:** cómo la aplicación se conecta al SGBD (ORM, APIs, servicios).
- **Integración de datos de múltiples fuentes:** heterogeneidad, esquemas diferentes, sincronización.
- **Consistencia, transacciones, concurrencia:** garantizar que la base de datos funcione correctamente cuando es accedida por la aplicación.
- **Separación de responsabilidades, capas de abstracción:** acceso a datos, lógica de negocio, presentación.

Enfoque de integración de bases de datos en entornos actuales

- **Microservicios y bases de datos:** cada servicio puede tener su propio almacenamiento, lo que implica coordinación e integración.
- **Arquitecturas API-first, GraphQL, REST:** acceso a datos mediante servicios intermedios.
- **Bases de datos heterogéneas (relacional + NoSQL) usadas juntos:** necesidad de integración, replicación, consistencia.
- **ETL / ELT, data-warehousing, data-lakes:** cuando la aplicación necesita datos integrados de múltiples fuentes.

ACTIVIDADES DE LA UNIDAD 4

ACTIVIDAD 1: Transacciones SQL

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

BIBLIOGRAFÍA

Arévalo, J. (15 de 10 de 2013). *Geotalleres*. Recuperado el 06 de 06 de 2019, de https://geotalleres.readthedocs.io/es/latest/conceptos-sql/conceptos_sql.html

Díaz, J. (11 de 02 de 2016). *pildorasinformaticas*. Recuperado el 06 de 06 de 2019, de https://www.pildorasinformaticas.es/course/curso-sql/curriculum/

Gabillaud, J. (2013). *SQL Server 2012 - SQL, Transact SQL: Diseño y creación de una base de datos*. España: Angel Maria Sanchez Conejo.

gplsi.dlsi.ua.es. (s.f.). *gplsi.dlsi.ua.es*. Recuperado el 06 de 06 de 2019, de http://gplsi.dlsi.ua.es/bbdd/bd1/lib/exe/fetch.php?media=bd1:0910:trabajos:seguridadbd.pdf

Gutiérrez, D. (01 de 2009). *codecompiling.net*. Recuperado el 06 de 06 de 2019, de http://www.codecompiling.net/files/slides/BD_clase_11_transacciones.pdf

Lázaro, D. (2018). *diego.com.es*. Recuperado el 31 de 03 de 2020, de https://diego.com.es/principales-tipos-de-joins-en-sql

Lopez Sanz, M., Soltero Domingo, F. J., Sanchez Fuquene, D. M., Moreno Perez, A., Bollati, V. A., & Vara Mesa, J. M. (2016). *PROGRAMACIÓN WEB EN EL ENTORNO SERVIDOR*. Madrid: Ra-Ma 2016.

Porro Chulli, M. A. (22 de 04 de 2018). *SlideShare*. Recuperado el 06 de 06 de 2019, de https://es.slideshare.net/LisbethOcaaBueno/vistas-94647190?from_action=save

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2006). *Programación de base de datos relacionales*. (Quinta Edición). McGraw-hill/Interamericana. España. ISBN: 978-84-481-5671-8, Número de inventario en biblioteca: ISTT-DS-0097

Martínez López, F. J., & Gallegos Ruiz, A. (2017). *Programación de base de datos relacionales*. (Primera Edición). Ra-ma. Colombia. ISBN: 978-958-762-684-1, Número de inventario en biblioteca: ISTT-DS-0027

Castaño, M., Piattini, M., & Esperanza, M. (2000). *Diseño de base de datos relacionales*. (Primera Edición). Alfaomega Grupo Editorial S.A. México. ISBN: 970-15-0526-3, Número de inventario en biblioteca: ISTT-DS-0031

Pérez Marqués, M. (2016). *Administración básica de base de datos con Oracle 12 c SQL Prácticas y Ejercicios*. (Primera Edición). Alfaomega. Colombia. ISBN: 978-958-778-202-8, Número de inventario en biblioteca: ISTT-DS-0057

ELABORACIÓN, REVISIÓN Y APROBACIÓN DE PARES	
Profesores	
Ing. Patricio Guanipatin	
Fecha de elaboración: 07/11/2025	
Comisión de revisión de pares de guías de estudio del Instituto Superior Tecnológico Tena	
Mg. Alvaro Santiago Toalombo Díaz	Mg. Henry Fabian Chango Chango
Mg. Martha Janina Duarte Mora	Lcda. María Angélica Campoverde Encalada
Mg. Danilo Alexander Zamora Núñez	
Fecha de revisión: 21/01/2025	
Coordinador de Investigación, Desarrollo Tecnológico e Innovación	
Mg. Danilo Alexander Zamora Núñez INSTITUTO SUPERIOR TECNOLÓGICO TENA COORDINACIÓN DE INVESTIGACIÓN	
Fecha de aprobación: 22/01/2025	