Entregable 2

Clase: SQL

Comisión: 81805

Nombre: st_coder_rox

Alumna: Roxanne Colosia Servin

Entregable 2

Roxanne Colosia Servin - Coder_SQL

Presentación/Objetivo:

Este proyecto desarrolla una base de datos para la gestión de técnicos, clientes y servicios asociados a estos. La idea de la base permitirá registrar a los técnicos, los clientes, los materiales y equipos solicitados, así como los pedidos y vacaciones de los técnicos.

El objetivo de este seria implementar un modelo de datos que facilite la administración de información para llevar rentabilidad técnica y un control operativo del personal y disponibilidad

Problemática:

La gestión de técnicos, clientes y servicios se realiza de forma dispersa, utilizando documentos separados o registros manuales. Esto genera duplicidad de información, dificultad para rastrear pedidos y problemas para controlar las vacaciones o disponibilidad del personal. La implementación de una base de datos centralizada solucionará estas brechas y un control más organizado, así como se podría sacar costos operativos en base a clientes para un futuro análisis de rentabilidad y/o noción de gastos y control de equipos.

Modelo de Negocio:

La empresa ficticia brindaría servicio técnico a clientes que adquieren productos químicos. El modelo establece que, según el volumen de compra, los clientes pueden acceder a equipos en comodato o consignación para el uso de dichos productos. Asimismo, se gestionan reactivos para titulaciones y refacciones para mantenimiento, garantizando soporte técnico continuo y un control eficiente de recursos.

Entregable 2

Roxanne Colosia Servin - Coder_SQL

Listado de Tablas:

tecnicos				
Campo	**Tipo de Dato**	**PK**	**FK**	**NULL**
id_tecnico	INT	X		
nombre	VARCHAR(100)			
apellido	VARCHAR(100)			
fecha_de_ingreso	DATE			
email	VARCHAR(100)			X

clientes				
Campo	**Tipo de Dato**	**PK**	**FK**	**NULL**
id_cliente	INT	X		
unidad	VARCHAR(100)			
razon_social	VARCHAR(100)			

equipos_material				
Campo	Tipo de Dato	PK	FK	NULL
codigo	VARCHAR(100)	X		
descripcion	VARCHAR(100)			
tipo_de_producto	VARCHAR(100)			
costo	DECIMAL(10,2)			

base_de_servicio				
Campo	**Tipo de Dato**	**PK**	**FK**	**NULL**
numero_servicio	INT	X		
id_tecnico	INT		X	
id_cliente	INT		X	
fecha_de_servicio	DATE			
tipo_de_servicio	VARCHAR(100)			

Entregable 2

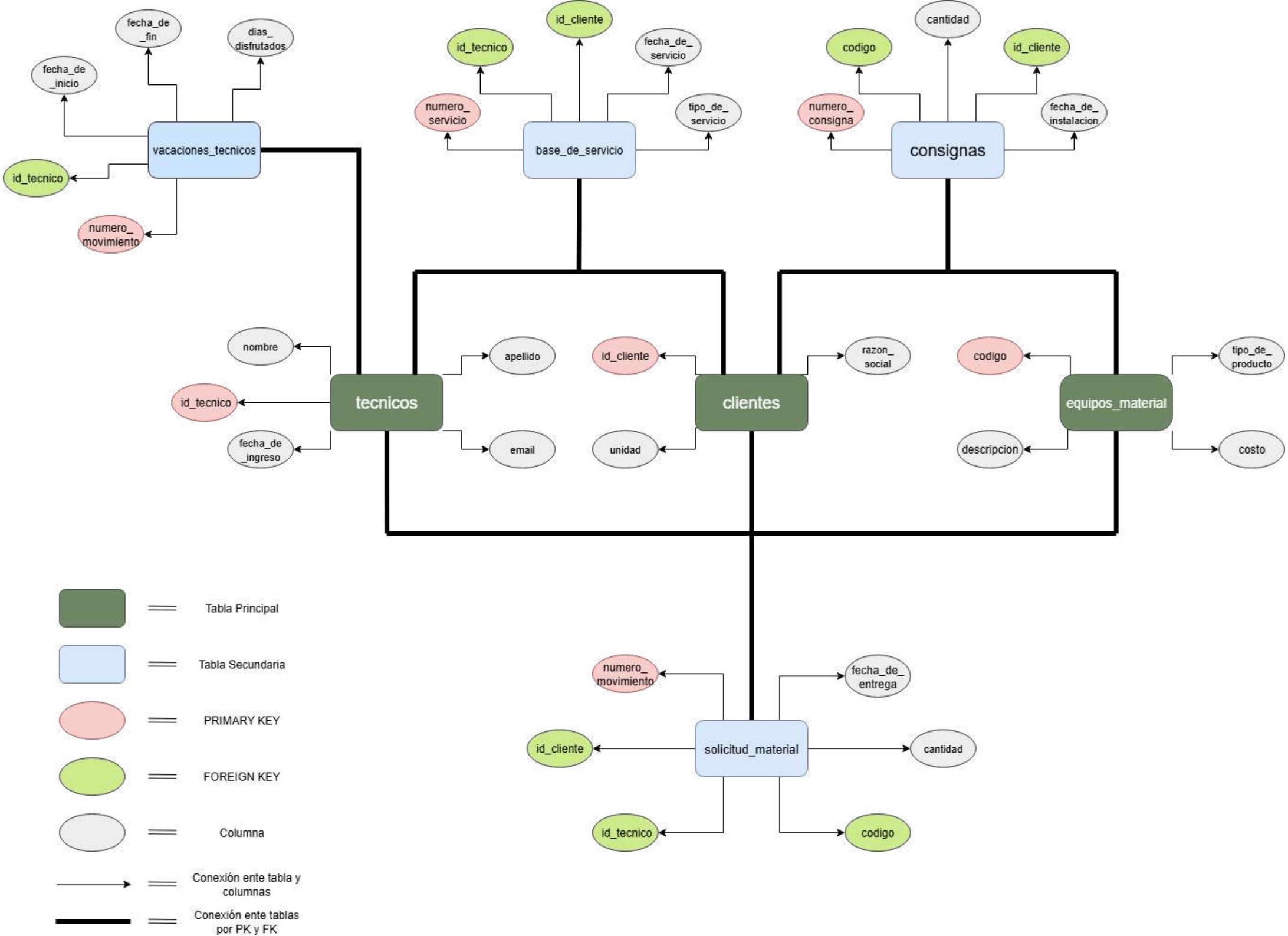
Roxanne Colosia Servin - Coder_SQL

solicitud_material				
Campo	**Tipo de Dato**	**PK**	**FK**	**NULL**
numero_movimiento	INT	X		
id_cliente	INT		X	
id_tecnico	INT		X	
codigo	VARCHAR(100)		X	
cantidad	INT			
fecha_de_entrega	DATE			

consigna				
Campo	**Tipo de Dato**	**PK**	**FK**	**NULL**
numero_consigna	INT	X		
codigo	VARCHAR(100)		X	
cantidad	INT			
id_cliente	INT		X	
fecha_instalacion	DATE			

vacaciones_tecnicos				
Campo	**Tipo de Dato**	**PK**	**FK**	**NULL**
numero_movimiento	INT	X		
id_tecnico	INT		X	
fecha_de_inicio	DATE			
fecha_de_fin	DATE			
dias_disfrutados	INT			

Diagrama Entidad Relación:


fecha_de_fin
dias_disfrutados
fecha_de_inicio
vacaciones_tecnicos
id_tecnico
numero_movimiento
id_cliente
id_tecnico
fecha_de_servicio
numero_servicio
tipo_de_servicio
base_de_servicio
cantidad
codigo
id_cliente
numero_consigna
fecha_de_instalacion
consignas
nombre
apellido
id_tecnico
tecnicos
fecha_de_ingreso
email
id_cliente
razon_social
clientes
unidad
codigo
tipo_de_producto
equipos_material
descripcion
costo
numero_movimiento
fecha_de_entrega
id_cliente
solicitud_material
cantidad
id_tecnico
codigo
Tabla Principal
Tabla Secundaria
PRIMARY KEY
FOREIGN KEY
Columna
Conexión ente tabla y columnas
Conexión ente tablas por PK y FK

Entregable 2

Roxanne Colosia Servin - Coder_SQL

Inserción de datos

Para alimentar la base de datos se realizaron inserciones de información mediante archivos **CSV**, principalmente en las tablas **clientes** y **técnicos**, consideradas como dos de las más importantes. Asimismo, se cargaron datos en la tabla **equipos_material**, utilizada para la obtención de información complementaria.
Adicionalmente, para el resto de las tablas se emplearon sentencias **INSERT INTO** con el fin de establecer una base inicial de datos.

Vistas

Con el propósito de facilitar la consulta y obtener información de manera más clara, se diseñaron **cinco vistas**, cada una enfocada en un aspecto particular:

1. **vista_servicio**
   Muestra la información de la tabla *base_de_servicio* de forma representativa mediante un **JOIN** entre las tablas *técnicos* y *clientes*. De esta manera, se obtiene una visualización más descriptiva en lugar de únicamente datos numéricos.
2. **consignas_pendientes**
   Permite identificar a los clientes a quienes aún no se les han consignado equipos en comodato. Para ello, se utilizó un **LEFT JOIN** combinado con la condición **WHERE IS NULL**.
3. **solicitud_materia_cliente**
   Presenta las solicitudes de materiales realizadas por cada cliente. Esta vista, construida mediante un **doble JOIN** entre las tablas *clientes* y *equipos_material*, permite en un futuro priorizar la entrega de materiales según la fecha o el cliente solicitante.
4. **servicios_pendientes**
   Muestra los servicios aún no realizados durante el mes, a partir de la relación de clientes. Se implementó con un **LEFT JOIN** y la condición **WHERE IS NULL**, lo que permite identificar de manera directa los casos pendientes.
5. **vacaciones_tomadas**
   Resume la cantidad de vacaciones utilizadas por cada técnico, permitiendo llevar un control claro sobre los días tomados.

Entregable 2

Roxanne Colosia Servin - Coder_SQL

Funciones

Con el propósito de optimizar la consulta de datos y obtener información de manera más clara, se diseñaron **tres funciones**, cada una con un objetivo específico:

1. **calcular_antiguedad**
   Esta función emplea la cláusula **DETERMINISTIC**, junto con **TIMESTAMPDIFF**, **YEAR** y **CURDATE()**, para determinar la antigüedad de un técnico según su año de ingreso en comparación con el año actual.
2. **costo_solicitud**
   Función definida como **DETERMINISTIC**, que utiliza un **INNER JOIN** para calcular el costo de una solicitud de material. Con ello se obtiene una visión más clara del costo operativo asociado a la obtención de los equipos.
3. **total_servicios_cliente**
   También declarada como **DETERMINISTIC**, esta función permite determinar la cantidad de servicios realizados en el mes para cada cliente, o bien para un cliente específico según la consulta realizada.

Stored Procedures

De manera complementaria, se desarrollaron **tres procedimientos almacenados (Stored Procedures)**, orientados a simplificar y automatizar tareas frecuentes dentro de la base de datos:

1. **registrar_solicitud**
   Facilita el registro de nuevas solicitudes de material, insertando la información directamente en la tabla **solicitud_material**.
2. **registrar_vacaciones**
   Este procedimiento presenta mayor complejidad, ya que además de calcular la diferencia entre dos fechas, busca descontar los domingos del período marcado. Para ello, se emplearon tres declaraciones (**DECLARE**) y tres asignaciones (**SET**), aplicando la fórmula:

$$\text{FLOOR}\left(\frac{\text{DATEDIFF}(fechaFin, fechaInicio) - 1 + (8 - 1)}{7}\right)$$

Entregable 2

Roxanne Colosia Servin - Coder_SQL

Con este cálculo, se obtiene un valor más aproximado de los días efectivos de vacaciones tomados.

3. **reporte_servicio**
   Permite consultar de manera específica los servicios realizados a un cliente determinado, brindando una visión detallada de las atenciones prestadas.

Triggers

Para garantizar la integridad de los datos y prevenir inconsistencias en la base de datos, se implementaron **dos triggers**, cada uno con una validación específica:

1. **validar_vacaciones**
   Este trigger se diseñó para evitar errores en el registro de vacaciones. Su función es comprobar que la fecha de fin no sea anterior a la fecha de inicio, de modo que no se registren períodos negativos por error de sintaxis. En caso de incumplirse la condición, se muestra el mensaje:
   "La fecha de fin debe ser posterior a la fecha de inicio".
2. **prevenir_borrado_cliente**
   Con este trigger se evita la eliminación de clientes de la tabla **clientes** en aquellos casos donde exista una consigna activa vinculada. Si se intenta ejecutar la operación, se genera el mensaje:
   "No se puede borrar cliente con consigna registrada".

README

Este repositorio contiene los archivos organizados por componentes para facilitar su análisis y ejecución de manera sencilla. Los archivos se encuentran separados en las siguientes categorías:

- **Triggers**
- **Stored Procedures (SP)**
- **Vistas**
- **Inserciones de datos** (en código SQL y en archivos CSV)
- **Funciones**

Entregable 2

Roxanne Colosia Servin - Coder_SQL

Orden de ejecución recomendado

Para garantizar la correcta carga y funcionamiento de la base de datos, se sugiere seguir el siguiente orden de ejecución:

1. **Schema**
   - Crear la estructura de las tablas de la base de datos.
2. **Archivos CSV**
   - Importar los datos iniciales de clientes, técnicos y equipos.
3. **Inserción de datos**
   - Ejecutar las sentencias INSERT INTO adicionales para poblar las tablas con información base.
4. **Vistas**
   - Crear las vistas definidas para facilitar la consulta de datos.
5. **Funciones**
   - Implementar las funciones diseñadas para cálculos y consultas específicas.
6. **Stored Procedures (SP)**
   - Registrar los procedimientos almacenados que automatizan operaciones frecuentes.
7. **Triggers**
   - Finalmente, activar los triggers que aseguran la integridad y validación de la información.

Notas adicionales

- Aunque después de la inserción de datos se puede ejecutar el resto del código en el orden que se desee, el flujo sugerido asegura una configuración más clara y libre de errores.
- Cada archivo está documentado y puede revisarse individualmente para su comprensión.
- Por favor revisar el PDF en el que se informa de manera más concreta todo el proyecto

LINK GITHUB

https://github.com/Roxanne-Hudan/st_coder_rox.git