Proyecto Base de Datos de Municipio

Curso SQL Coder House

Comisión: 47365

Federico Ibañez

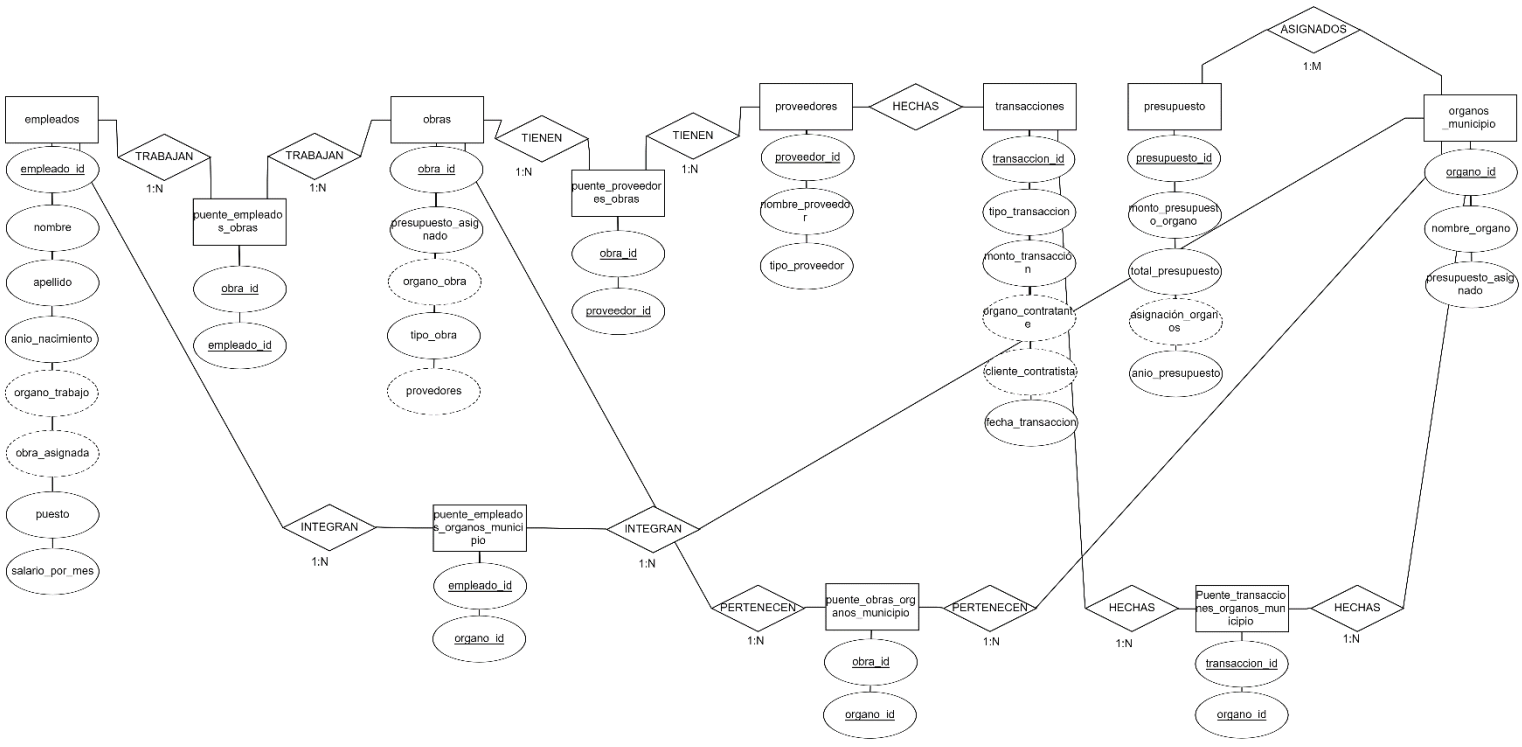
Índice

1. Descripción de la Base de Datos

Esta base de datos trata sobre la organización de transacciones y obras a nivel de un municipio que se relacionan con empleados que trabajan en las obras y pertenecen a un órgano determinado de ese municipio. Además, estas obras están asignadas a un órgano específico, los cuales tienen un presupuesto por año asignado a cada uno de ellos.

Las transacciones que se detallan en la Base de datos están asignadas a un órgano específico, y a un proveedor o cliente.

2. Diagrama Entidad Relación



(el DER también se encuentra en la carpeta de OneDrive)

3. Descripción de tablas

Tabla 1: "*Empleados*"

La tabla Empleados nos muestra todos los datos acerca de quienes trabajan dentro del municipio y están asignados a un órgano y a una obra en específico. Tiene como PK al campo Empleado_ID. Tiene como FK los campos "Organo_trabajo" y "Obra_asignada"

Clave	Campo2	Tipo de dato	Descripción
PK	empleado_id	INT	Clave de identificación de la tabla Empleados
	nombre	VARCHAR (30)	Nombre del empleado
	apellido	VARCHAR (30)	Apellido del empleado
	anio_nacimiento	SMALLINT	Año de nacimiento del empleado
FK	organo_trabajo	INT	Organo / Sector en donde trabaja el empleado
FK	obra_asignada	INT	Obra a la que fue asignado el empleado
	puesto	VARCHAR (50)	Nombre del puesto del empleado
	salario_por_mes	INT	Numero de salario por mes

Tabla 2: "*Obras*"

La tabla OBRAS nos muestra todos los datos sobre los proyectos o iniciativas que se llevan a cabo en un Municipio. Están relacionados con los empleados asignados a cada obra y con el órgano correspondiente de ésta. Además, se relaciona con los correspondientes proveedores.

Clave	Campo2	Tipo de dato	Descripción
PK	obra_id	INT	Clave de identificación de la tabla Obras
	presupuesto_asignado	INT	Presupuesto asignado a cada obra

FK	organo_obra	INT	Nombre del organo encargado de la obra. FK de Organos
	tipo_obra	VARCHAR (60)	Clasificación de la obra
FK	proveedores	INT	Nombre del proveedor encargado de la obra

Tabla 3: "*Proveedores*"

La tabla PROVEEDORES nos muestra los datos sobre aquellas empresas o individuos que brindaron servicios o bienes al municipio. Nos encontramos con información como el nombre y el tipo de proveedor.

Clave	Campo2	Tipo de dato	Descripción
PK	proveedor_id	INT	Clave de identificación de la tabla Proveedores
	nombre_proveedor	VARCHAR (50)	Nombre del proveedor encargado de la obra
	tipo_proveedor	VARCHAR (50)	Clasificación del proveedor

Tabla 4: "*Transacciones*"

En la tabla TRANSACCIONES nos encontramos con datos acerca de todas las transferencias de dinero a cambio de un servicio o bien con proveedores, o en viceversa con clientes. La tabla nos muestra información sobre el tipo de transacción, el monto, la fecha y los involucrados en la transacción

Clave	Campo2	Tipo de dato	Descripción
PK	transaccion_id	INT	Clave de identificación de la tabla Transacciones
	tipo_transaccion	VARCHAR (50)	Clasificación de la transaccion
	monto_transaccion	INT	Monto de la transaccion realizada
FK	organo_contratante	INT	Nombre del organo que contrata y realiza la transaccion. FK de Organos
FK	cliente_contratista	INT	Nombre del proveedor en la transaccion. FK de proveedores

	fecha_transaccion	DATE	Fecha de la transacción

Tabla 5: "*Presupuesto*"

La tabla PRESUPUESTO provee datos sobre la cantidad de dinero a disponibilidad del Municipio, discriminando por año y por presupuesto asignado a cada órgano municipal.

Clave	Campo2	Tipo de dato	Descripción
PK	id_presupuesto	INT	Clave de identificación de la tabla Presupuesto
	monto_presupuesto_organo	INT	Monto del presupuesto asignado a cada órgano
	total_presupuesto	INT	Total del presupuesto disponible por año
FK	asignacion_organos	INT	Organo al que fue asignado el presupuesto
	anio_presupuesto	SMALLINT	Año del presupuesto asignado a cada órgano

Tabla 6: "*Organos_municipio*"

En la tabla ORGANOS_MUNICIPIO encontramos los datos referidos a cada subdivisión del municipio, como secretarías y sectores específicos, los cuales cuentan con un presupuesto asignado, empleados, y obras.

Clave	Campo2	Tipo de dato	Descripción
PK	id_organo	INT	Clave de identificación de la tabla Organo_dependiente
	nombre_organo	VARCHAR (50)	Nombre del Órgano del Municipio.
FK	presupuesto_asignado	INT	Presupuesto asignado al órgano. FK de la tabla Presupuesto

Tablas Puente

Tabla 7 *"puente empleados órganos municipio"*

Esta tabla sirve como puente entre la tabla "Empleados" y la tabla "Órganos municipio"

Clave	Campo2	Tipo de dato	Descripción
FK	empleado_id	INT	Foreign Key de la tabla "empleados"
FK	id_organo	INT	Foreign Key de la tabla "Organos municipio"

Tabla 8 *"Puente Empleados Obras"*

Esta tabla sirve como puente entre la tabla "Empleados" y la tabla "Obras".

Clave	Campo2	Tipo de dato	Descripción
FK	obra_id	INT	Foreign Key de la tabla "Obras"
FK	empleado_id	INT	Foreign Key de la tabla "empleados"

Tabla 9 *"Puente Obras Órganos Municipio"*

Esta tabla sirve como puente entre la tabla "Obras" y la tabla "Órganos Municipio".

Clave	Campo2	Tipo de dato	Descripción
FK	obra_id	INT	Foreign Key de la tabla "Obras"
FK	id_organo	INT	Foreign Key de la tabla "Organos Municipio"

Tabla 10 ***"Puente Proveedores Obras"***

Esta tabla sirve como puente entre la tabla "Obras" y la tabla "Proveedores".

Clave	Campo2	Tipo de dato	Descripción
FK	obra_id	INT	Foreign Key de la tabla "Obras"
FK	proveedor_id	INT	Foreign Key de la tabla "Proveedores"

Tabla 11 ***"Puente Transacciones órganos Municipio"***

Esta tabla sirve como puente entre la tabla "Transacciones" y la tabla "Órganos Municipio"

Clave	Campo2	Tipo de dato	Descripción
FK	transaccion_id	INT	Foreign Key de la tabla "Transacciones"
FK	id_organo	INT	Foreign Key de la tabla "Organos Municipio"

4. Script DDL

```
DROP SCHEMAIF EXISTS `municipio_proyecto_final`;CREATE SCHEMA
IF NOT EXISTS `municipio_proyecto_final`;
USE `municipio_proyecto_final`;


CREATE TABLE `empleados`
  (
     `empleado_id`      INT NOT NULL,
     `nombre`           VARCHAR(30) DEFAULT NULL,
     `apellido`         VARCHAR(30) DEFAULT NULL,
     `anio_nacimiento` SMALLINT DEFAULT NULL,
     `organo_trabajo`   INT DEFAULT NULL,
     `obra_asignada`    INT DEFAULT NULL,
     `puesto`           VARCHAR(50) DEFAULT NULL,
     `salario_por_mes` INT DEFAULT NULL,
     UNIQUE INDEX `ak_empleados` (`empleado_id`, `anio_nacimient
o`),
     PRIMARY KEY (`empleado_id`)
  );



CREATE TABLE `organos_municipio`
  (
     `id_organo`             INT,
     `nombre_organo`         VARCHAR(50),
     `presupuesto_asignado` INT,
     PRIMARY KEY (`id_organo`)
  );



CREATE TABLE `puente_empleados_organos_municipio`
  (
     `empleado_id` INT,
     `id_organo`    INT,
     CONSTRAINT `pk_puente_empleados_organos_municipio` PRIMARY
KEY (
     `empleado_id`, `id_organo`),
```

```
    FOREIGN KEY (empleado_id) REFERENCES empleados (empleado_id
),
    FOREIGN KEY (id_organo) REFERENCES organos_municipio (id_or
gano)
  );



CREATE TABLE `obras`
  (
    `obra_id`               INT,
    `presupuesto_asignado` INT,
    `organo_obra`           INT,
    `tipo_obra`             VARCHAR(60),
    `proveedores`           INT,
    PRIMARY KEY (`obra_id`)
  );



CREATE TABLE `puente_empleados_obras`
  (
    `obra_id`     INT,
    `empleado_id` INT,
    CONSTRAINT `pk_puente_empleados_obras` PRIMARY KEY (`obra_i
d`,
    `empleado_id`),
    FOREIGN KEY (empleado_id) REFERENCES empleados (empleado_id
),
    FOREIGN KEY (obra_id) REFERENCES obras (obra_id)
  );



CREATE TABLE `puente_obras_organos_municipio`
  (
    `obra_id`   INT,
    `id_organo` INT,
    CONSTRAINT `pk_puente_obras_organos_municipio` PRIMARY KEY
(`obra_id`,
    `id_organo`),
    FOREIGN KEY (obra_id) REFERENCES obras (obra_id),
    FOREIGN KEY (id_organo) REFERENCES organos_municipio (id_or
gano)
  );
```

```
CREATE TABLE `proveedores`
  (
     `proveedor_id`     INT,
     `nombre_proveedor` VARCHAR(50),
     `tipo_proveedor`   VARCHAR(60),
     PRIMARY KEY (`proveedor_id`)
  );


CREATE TABLE `puente_proveedores_obras`
  (
     `obra_id`      INT,
     `proveedor_id` INT,
     CONSTRAINT `pk_puente_proveedores_obras` PRIMARY KEY (`obra
_id`,
     `proveedor_id`),
     FOREIGN KEY (obra_id) REFERENCES obras (obra_id),
     FOREIGN KEY (proveedor_id) REFERENCES proveedores (proveedo
r_id)
  );



CREATE TABLE `transacciones`
  (
     `transaccion_id`      INT,
     `tipo_transaccion`    VARCHAR(50),
     `monto_transaccion`   INT,
     `organo_contratante`  INT,
     `cliente_contratista` INT,
     `fecha_transaccion`   DATE,
     PRIMARY KEY (`transaccion_id`),
     UNIQUE INDEX `ak_transaccion` (`transaccion_id`, `monto_tra
nsaccion`),
     FOREIGN KEY (cliente_contratista) REFERENCES proveedores (p
roveedor_id)
  );



CREATE TABLE `puente_transacciones_organos_municipio`
  (
     `transaccion_id` INT,
     `id_organo`      INT,
     CONSTRAINT `pk_puente_transacciones_organos_municipio` PRIM
ARY KEY (
     `transaccion_id`, `id_organo`),
     FOREIGN KEY (transaccion_id) REFERENCES transacciones (tran
saccion_id),
     FOREIGN KEY (id_organo) REFERENCES organos_municipio (id_or
```

```
gano)
  );
```

```
CREATE TABLE `presupuesto`
  (
     `id_presupuesto`            INT,
     `anio_presupuesto`          SMALLINT,
     `total_presupuesto`         INT,
     `asignacion_organo`         INT,
     `monto_presupuesto_organo` INT,
     PRIMARY KEY (`id_presupuesto`),
     FOREIGN KEY (asignacion_organo) REFERENCES organos_municipi
o (id_organo)
  );
```

5. VISTAS

VIEW 1 *"Conteo empleados por Órgano"*

(vw_count_empleados_por_organo)

Esta vista tiene por objetivo mostrar un conteo de todos los empleados por cada órgano del municipio. Las tablas que integran esta vista son la tabla "Empleados" y la tabla "Órganos Municipio".

```
CREATE VIEW vw_count_empleados_por_organo
AS
  SELECT org.nombre_organo,
         Count(e.empleado_id) AS cantidad
  FROM   empleados AS e
         JOIN organos_municipio AS org
           ON e.organo_trabajo = org.id_organo
  GROUP  BY org.nombre_organo;
```

VIEW 2 *"Conteo de obras por Órgano + Presupuesto total"*

(vw_obras_por_organo)

Esta vista tiene por objetivo mostrar la cantidad de obras que están relacionadas con cada Órgano del municipio. También muestra el presupuesto sumado de todas las obras. Las tablas que participan son "Obras" y "Órganos Municipio".

```
CREATE VIEW vw_obras_por_organo
AS
  SELECT org.nombre_organo,
         Count(obr.obra_id)             AS cantidad_obras,
         Sum(obr.presupuesto_asignado) AS total_presupuest
o
  FROM   obras AS obr
         JOIN organos_municipio AS org
           ON org.id_organo = obr.organo_obra
  GROUP  BY org.nombre_organo;
```

VIEW 3 “Detalle de proveedores”

(vw_proveedores)

Esta vista tiene por objetivo mostrar el total del monto de transacciones por cada proveedor y el tipo de proveedor en el que se agrupa. En esta vista participan las tablas “Proveedores” y “Transacciones”

```
CREATE VIEW vw_proveedores
AS
  SELECT pro.nombre_proveedor,
         pro.tipo_proveedor,
         SUM(tra.monto_transaccion) monto_total
  FROM   proveedores AS pro
         join transacciones AS tra
           ON pro.proveedor_id = tra.cliente_contratista
  GROUP  BY pro.nombre_proveedor,
            pro.tipo_proveedor
  ORDER  BY monto_total DESC;
```

VIEW 4 *“Transacciones por Órgano”*

(vw_transacciones_por_organo)

Esta vista tiene por objetivo mostrar la cantidad de transacciones y gasto por parte de cada Órgano Municipal. En esta vista participan las tablas “Transacciones ” y “Órganos Municipales”.

```
      CREATE VIEW vw_transacciones_por_organo
AS
  SELECT org.nombre_organo,
         Count(tra.transaccion_id)  AS cantidad_transaccio
nes,
         SUM(tra.monto_transaccion) AS monto_total
  FROM   transacciones AS tra
         join organos_municipio AS org
           ON tra.organo_contratante = org.id_organo
  GROUP  BY org.nombre_organo
  ORDER  BY cantidad_transacciones DESC,
            monto_total DESC;
```

VIEW 5 *"Empleados por debajo del Salario Mínimo"*

(vw_empleados_debajo_salario_minimo)

Esta vista tiene por objetivo mostrar todos los datos de aquellos empleados que estén por debajo del salario mínimo. En esta vista participan las tablas "Empleados" y "Órganos Municipales"

```
CREATE VIEW vw_empleados_debajo_salario_minimo
AS
  SELECT emp.nombre,
         emp.apellido,
         org.nombre_organo,
         emp.puesto,
         emp.salario_por_mes
  FROM   empleados AS emp
         join organos_municipio AS org
           ON emp.organo_trabajo = org.id_organo
  WHERE  emp.salario_por_mes < 675000
  ORDER  BY salario_por_mes;
```

6. FUNCIONES

FUNCIÓN *"Salario Neto"*

(fn_salario_neto)

Esta función tiene como objetivo mostrar el salario neto por mes para un empleado, introduciendo un ID y la cantidad que se descuenta. Las tabla que participa es la tabla "Empleados"

Consulta:

```
      SELECT salario_por_mes - 30000 AS Neto
-- , salario_por_mes
-- , empleado_id
FROM   empleados
WHERE  empleado_id = 5;
```

Función:

```
      DROP FUNCTIONIF EXISTS fn_salario_neto;DELIMITER //CR
EATE FUNCTION fn_salario_neto ( p_descuento int,
p_empleado_id                                 int )
returns int deterministic
BEGINDECLARE v_salario_neto INT;SET v_salario_neto =
  (
         SELECT salario_por_mes - p_descuento AS neto
         FROM   empleados
         WHERE  empleado_id = p_empleado_id);RETURN v_sala
rio_neto;END //
delimiter ;
```

FUNCIÓN *"Clasificación de Transacción"*

(fn_clasif_transaccion)

Esta función tiene como objetivo mostrar una clasificación (alto, medio o bajo) del monto de una transacción introduciendo ese mismo monto. La tabla que participa es la tabla "Transacciones".

```
DROP FUNCTIONIF EXISTS fn_clasif_transaccion;DELIMITER //C
REATE FUNCTION fn_clasif_transaccion ( p_transaccion_id in
t)
returns varchar (10) deterministic
BEGIN
```

```
  CASE
  WHEN p_transaccion_id >= 9000000 THENRETURN "alta";WHEN
p_transaccion_id >= 2000000 THENRETURN "media";WHEN p_tran
saccion_id < 2000000 THENRETURN "baja";END
  CASE;END // delimiter ;SELECT Fn_clasif_transaccion (200
0005) AS clasificacion_transaccion;
```

7. STORE PROCEDURES

STORE PROCEDURE *"salarios"*

(sp_salarios)

Este procedimiento sirve para determinar si el salario de un empleado se encuentra por encima, por debajo, o igualado al salario mínimo de cada país. Los parámetros de entrada son "Salario empleado" y "Salario mínimo".

```
DROP PROCEDUREIF EXISTS sp_salarios;DELIMITER //CREATE PRO
CEDURE sp_salarios ( in salario_empleado int,
                                IN salario_minimo   int )
BEGINDECLARE v_salario_empleado INT;DECLARE v_salario_mini
mo   INT;IF salario_empleado < salario_minimo then
  SELECT salario_minimo,
         salario_empleado,
         "por debajo del salario minimo" AS estado;ELSEIF
salario_empleado > salario_minimo THENSELECT salario_emple
ado,
         salario_minimo,
         "por encima del salario minimo" AS estado;ELSEIF
salario_empleado = salario_minimo THENSELECT salario_emple
ado,
         salario_minimo,
         "ambos salarios son iguales" AS estado;ELSESELECT
 salario_empleado,
         salario_minimo;ENDIF;END //
delimiter ;
```

STORE PROCEDURE *"En edad jubilatoria"*

(clasif_edad)

Este procedimiento determina si un empleado está en la edad jubilatoria establecida en cada caso (país, contrato, etc..). Los parámetros de entrada son "Año de nacimiento" y "Edad jubilatoria".

```
      DROP PROCEDUREIF EXISTS clasif_edad;DELIMITER //CREAT
E PROCEDURE clasif_edad ( in anio_nacimiento  int,
                                IN edad_jubilatoria int )
BEGINDECLARE cant_anios_dif INT;DECLARE minimo_anios   INT
;SET minimo_anios = year(CURRENT_DATE()) - edad_jubilatori
a;IF anio_nacimiento > minimo_anios then
  SELECT minimo_anios,
         "NO en edad de jubilación" ;ELSEIF anio_nacimient
o < minimo_anios THENSELECT minimo_anios,
         "en edad de jubilaci?n"
  FROM   empleados;ELSESELECT "-";ENDIF;END //
delimiter ;
```