Instituto Tecnológico y de Estudios Superiores de Monterrey.

Campus Puebla

ICA 2.1

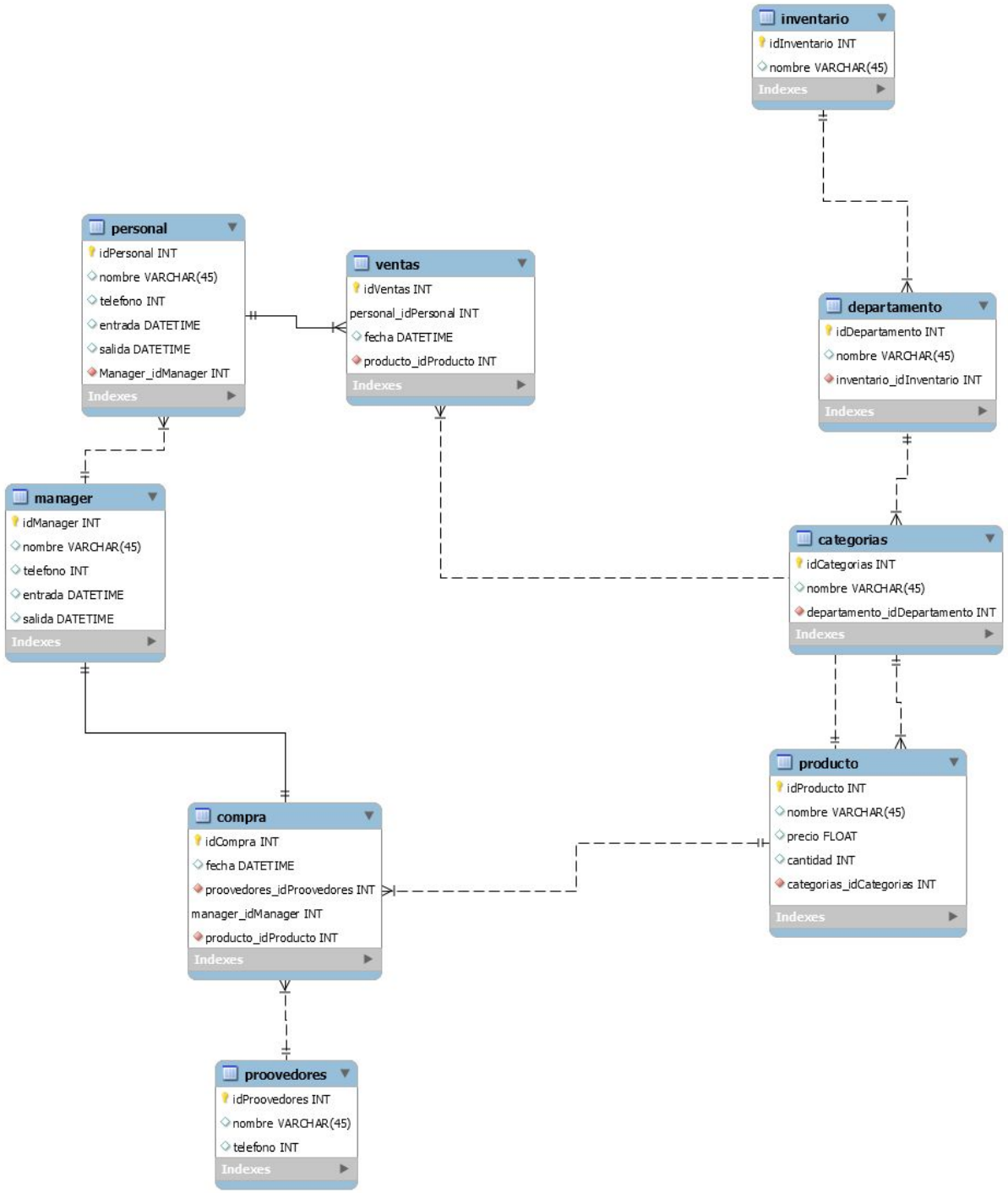
Nombre de los alumnos:

Fernanda Montaño Rios	***A01730440***
Abigail Gonzalez Hidalgo	***A00819967***
Oscar Cañongo Vergara	***A01730443***

Profesor:

Daniel Pérez Rojas

Situación previa a este ICA



Modificaciones necesarias al esquema para soportar lo solicitado en este ICA

- Se creó una tabla "orden_compra" para que cuando un artículo llegue a cero, inmediatamente se genere una entrada que pida 10 artículos del que se terminó.

Field	Type	Null	Key	Default	Extra
idCompra	int	NO	PRI	NULL	auto_increment
id_producto	int	YES	MUL	NULL	
cantidad	float	YES		NULL	
fecha	datetime	YES		NULL	

- Se creó una tabla “historial_precios” que registra cuando se realiza un cambio de precio, de precio previo a precio nuevo .

Field	Type	Null	Key	Default	Extra
idPrecios	int	NO	PRI	NULL	auto_increment
id_producto	int	YES	MUL	NULL	
precioPrevio	float	YES		NULL	
precioActual	float	YES		NULL	
action	varchar(50)	YES		NULL	
changedate	datetime	YES		NULL	

- Se creó una tabla “resumen_ventas” que genera un resumen de las ventas de las últimas 8 horas.

Field	Type	Null	Key	Default	Extra
idVentas	int	NO		0	
fecha	datetime	YES		NULL	
nombre	varchar(45)	YES		NULL	
cantidad	int	YES		NULL	
precioDeVenta	float	YES		NULL	

Detalles de cada uno de los pasos incluyendo capturas de pantalla de la ejecución de cada parte.

- **Agregar una parte que trabaje con historial de precios**
  1) Se creó la tabla historial_precios.

```
-- Trigger Historial_Precios
CREATE TABLE historial_precios(
idPrecios INT NOT NULL  auto_increment,
id_producto INT,
precioPrevio float,
precioActual float,
action varchar(50) default null,
changedate datetime default null,
CONSTRAINT pkPrecios PRIMARY KEY (idPrecios),
CONSTRAINT fkProducto FOREIGN KEY (id_Producto) REFERENCES producto(idProducto)
);
```

  2) Se creó el trigger after_update_precio

```
DELIMITER $$
  CREATE TRIGGER after_update_precio
  AFTER UPDATE ON producto
  FOR EACH ROW
  BEGIN
    INSERT INTO historial_precios
    SET action = 'Update',
    id_producto = NEW.idProducto,
    precioPrevio = OLD.precio,
    precioActual = NEW.precio,
    changedate = NOW();
  END$$
DELIMITER ;
```

  3) Tabla producto

idProducto	nombre	precio	cantidad
1	Bujia	170	10
2	Filtro de aceite	212	77
3	Limpiaparabrisas	48	10
4	Bateria	4659	10
5	Luz intenrior	449.9	10
6	Rondana	1501	10
7	Rampa	600	10
8	Red de carga	199.99	1

  4) Se procede a modificar el precio de filtro de aceite y del limpiaparabrisas

```
UPDATE producto SET precio=212 Where idProducto = 2;
UPDATE producto SET precio=48 Where idProducto = 3;
```

5) Se comprueban los cambios

idProducto	nombre	precio	cantidad
1	Bujia	170	10
2	Filtro de aceite	250	77
3	Limpiaparabrisas	50	10
4	Bateria	4659	10
5	Luz intenrior	449.9	10
6	Rondana	1501	10
7	Rampa	600	10
8	Red de carga	199.99	1

6) Se ve una nueva entrada en la tabla historial_precios

idPrecios	nombre	precioPrevio	precioActual	action	changedate
47	Filtro de aceite	212	250	Update	2020-04-07 13:20:19
48	Limpiaparabrisas	48	50	Update	2020-04-07 13:20:19

- **Escribir una función que calcule el tiempo que lleva trabajando un trabajador**

1) Se creó la función HorasTrabajadas.

```
DELIMITER $$
  CREATE FUNCTION HorasTrabajadas(horaDeEntrada time, horaDesalida time) RETURNS time
  DETERMINISTIC
  BEGIN
    DECLARE tiempo time;
    DECLARE hora time DEFAULT now();
    IF hora<=horaDeEntrada THEN
    SET tiempo = null;
    ELSEIF horaDeSalida<=hora THEN
    SET tiempo = horaDeSalida-horaDeEntrada;
    ELSE
    SET tiempo = hora-horaDeEntrada;
    END IF;
    RETURN (tiempo);
  END$$
DELIMITER ;
```

2) En la tabla personal tenemos la hora de entrada y salida de los trabajadores

idPersonal	nombre	telefono	entrada	salida
1	Kayro	2281273199	16:00:00	23:00:00
2	El Bicho	2282445610	08:00:00	16:00:00
3	Messi	2287552164	08:00:00	16:00:00

3) Se llama la función

```
select HorasTrabajadas(entrada, salida) from personal;
```

4) Siendo la una de la tarde con 28 minutos el resultado es el siguiente

```
+-----------------------------------+
| HorasTrabajadas(entrada, salida) |
+-----------------------------------+
| NULL                              |
| 05:28:05                          |
| 05:28:05                          |
+-----------------------------------+
```

- **Escribir un stored procedure que tenga la lógica para hacer una venta.**

1) Se creó el stored procedure procedimientoVentas.

```
DELIMITER //
CREATE PROCEDURE procedimientoVentas(IN producto_idProducto int, cantidad1 int, idVentas int )
  BEGIN
    DECLARE productoCantidad INT;
    DECLARE precioVenta DOUBLE;
     SET productoCantidad =
     (SELECT cantidad FROM producto
      WHERE idProducto = producto_idProducto);
     SET precioVenta =
     (SELECT precio FROM producto
      WHERE idProducto = producto_idProducto) * cantidad1;
    IF productoCantidad >= cantidad1 THEN
     UPDATE producto
     SET cantidad = productoCantidad-cantidad1
     WHERE idProducto = producto_idProducto;
     INSERT INTO ventas (idVentas, personal_idPersonal, fecha, producto_idProducto, cantidad, precioDeVenta)
     VALUES (idVentas, 1,NOW(),producto_idProducto, cantidad1, precioVenta);
    END IF;
  END //
DELIMITER ;
```

2) Tabla producto.

```
+------------+------------------+--------+----------+
| idProducto | nombre           | precio | cantidad |
+------------+------------------+--------+----------+
|          1 | Bujia            |    170 |       10 |
|          2 | Filtro de aceite |    250 |       77 |
|          3 | Limpiaparabrisas |     50 |       10 |
|          4 | Bateria          |   4659 |       10 |
|          5 | Luz intenrior    |  449.9 |       10 |
|          6 | Rondana          |   1501 |       10 |
|          7 | Rampa            |    600 |        9 |
|          8 | Red de carga     | 199.99 |        1 |
+------------+------------------+--------+----------+
```

3) Se llama al stored procedure.

```
CALL procedimientoVentas(7,1,1);
```

4) Se observan los cambios en la tabla ventas.

idVentas	nombre	fecha	nombre	cantidad	precioDeVenta
1	Kayro	2020-04-07 16:09:33	Rampa	1	600

5) Se observan los cambios en la tabla producto.

idProducto	nombre	precio	cantidad
1	Bujia	170	10
2	Filtro de aceite	250	77
3	Limpiaparabrisas	50	10
4	Bateria	4659	10
5	Luz intenrior	449.9	10
6	Rondana	1501	10
7	Rampa	600	8
8	Red de carga	199.99	1

- **Escribir un trigger que genere una orden de compra cuando los artículos disponibles de un cierto producto lleguen a cero.**

1) Se crea la tabla orden_compra

```
CREATE TABLE orden_compra(
idCompra INT NOT NULL auto_increment,
id_producto INT,
cantidad float,
fecha datetime,
PRIMARY KEY (idCompra),
FOREIGN KEY (id_Producto) REFERENCES producto(idProducto)
);
```

2) Se crea el trigger de orden de compra

```
DELIMITER $$
  CREATE TRIGGER after_ordenCompra
  BEFORE UPDATE ON producto
  FOR EACH ROW
  BEGIN
   IF NEW.cantidad = 0 THEN
    SET NEW.cantidad = 10;
    INSERT INTO orden_compra(id_producto, cantidad, fecha) VALUES (OLD.idProducto, 10, now());
   END IF;
  END$$
DELIMITER ;
```

3) Tabla producto

idProducto	nombre	precio	cantidad
1	Bujia	169	34
2	Filtro de aceite	174.9	100
3	Limpiaparabrisas	169.9	50
4	Bateria	4659	50
5	Luz intenrior	449.9	200
6	Rondana	999.9	100
7	Rampa	600	10
8	Red de carga	199.99	30

4) Llamamos al procedimiento de ventas, en donde vamos a vaciar el producto con id 4

```
CALL procedimientoVentas(4,50,1);
```

5) En la tabla de ventas verificamos que se haya realizado la compra

idVentas	nombre	fecha	nombre	cantidad	precioDeVenta
1	Kayro	2020-04-07 15:54:27	Bateria	50	232950

6) Como también se observan cambios en la tabla de productos

idProducto	nombre	precio	cantidad
1	Bujia	169	34
2	Filtro de aceite	174.9	100
3	Limpiaparabrisas	169.9	50
4	Bateria	4659	10
5	Luz intenrior	449.9	200
6	Rondana	999.9	100
7	Rampa	600	10
8	Red de carga	199.99	30

- **Escribir un proceso que cada 8 horas genere un resumen de ventas de las ultimas 8 horas**

1) Se crea la tabla bar que es la que guarda lo que hace el evento

```
CREATE TABLE bar
  SELECT idVentas, fecha, nombre, ventas.cantidad, precioDeVenta
  FROM ventas INNER JOIN producto ON producto_idProducto = idProducto WHERE ADDDATE(now(),INTERVAL -3 minute) <= fecha;
```

2) Se crea el schedule event, se le puso 1 minuto para las pruebas y no esperar 8 horas

```
DELIMITER //
CREATE EVENT ventas_8horas
ON SCHEDULE EVERY 1 MINUTE
STARTS CURRENT_TIMESTAMP
ON COMPLETION PRESERVE
DO
BEGIN
  DELETE FROM bar;
  INSERT INTO bar (idVentas, fecha, nombre, cantidad, precioDeVenta)
  SELECT idVentas, fecha, nombre, ventas.cantidad, precioDeVenta
  FROM ventas INNER JOIN producto ON producto_idProducto = idProducto
  WHERE ADDDATE(now(),INTERVAL -1 MINUTE) <= fecha;
END //
DELIMITER ;
```

3) Se ven las ventas

idVentas	nombre	fecha	nombre	cantidad	precioDeVenta
1	Kayro	2020-04-07 15:54:27	Bateria	50	232950
2	Kayro	2020-04-07 16:15:06	Bujia	4	676
3	Kayro	2020-04-07 16:15:06	Limpiaparabrisas	10	1699
4	Kayro	2020-04-07 16:15:07	Bateria	1	4659
5	Kayro	2020-04-07 16:15:09	Red de carga	5	999.95
10	Kayro	2020-04-07 16:24:48	Bujia	1	169
11	Kayro	2020-04-07 16:24:50	Rampa	1	600
12	Kayro	2020-04-07 16:33:33	Luz intenrior	2	899.8
13	Kayro	2020-04-07 16:33:33	Bujia	2	338
14	Kayro	2020-04-07 16:33:35	Limpiaparabrisas	5	849.5

4) Se crea las ventas

```
mysql> CALL procedimientoVentas(1,7,15);
Query OK, 1 row affected (0.33 sec)

mysql>
mysql> CALL procedimientoVentas(4,3,16);
Query OK, 1 row affected (0.28 sec)

mysql> CALL procedimientoVentas(8,5,17);
Query OK, 1 row affected (0.32 sec)
```

5) Se muestra la tabla bar

idVentas	fecha	nombre	cantidad	precioDeVenta
15	2020-04-07 16:41:39	Bujia	7	1183
16	2020-04-07 16:41:40	Bateria	3	13977
17	2020-04-07 16:41:41	Red de carga	5	999.95

6) Se ven las ventas

idVentas	nombre	fecha	nombre	cantidad	precioDeVenta
1	Kayro	2020-04-07 15:54:27	Bateria	50	232950
2	Kayro	2020-04-07 16:15:06	Bujia	4	676
3	Kayro	2020-04-07 16:15:06	Limpiaparabrisas	10	1699
4	Kayro	2020-04-07 16:15:07	Bateria	1	4659
5	Kayro	2020-04-07 16:15:09	Red de carga	5	999.95
10	Kayro	2020-04-07 16:24:48	Bujia	1	169
11	Kayro	2020-04-07 16:24:50	Rampa	1	600
12	Kayro	2020-04-07 16:33:33	Luz intenrior	2	899.8
13	Kayro	2020-04-07 16:33:33	Bujia	2	338
14	Kayro	2020-04-07 16:33:35	Limpiaparabrisas	5	849.5
15	Kayro	2020-04-07 16:41:39	Bujia	7	1183
16	Kayro	2020-04-07 16:41:40	Bateria	3	13977
17	Kayro	2020-04-07 16:41:41	Red de carga	5	999.95

- ## Conclusión

  Este trabajo fue muy enriquecedor ya que pudimos poner en práctica nuestros conocimientos teóricos adquiridos el parcial pasado de la clase de bases de datos avanzadas. Nos dimos a la tarea de realizar triggers, events, stored procedures y funciones los cuales nos van a servir para proyectos futuros destacando el proyecto final de la materia.

  Creemos firmemente que si continuamos realizando este tipo de actividades podremos mejorar nuestras capacidades en la materia y no resentir el cambio del modelo presencial al modelo virtual.