

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

CURSO: BASES DE DATOS

Grupo: 01

Serie SQL

Alumno:

Ortiz Valles Joaquín Rafael

Profesor:

Ing. Fernando Arreola Franco

Diciembre 2025

1. Configuración y Scripts (Ejercicio 1)

Enunciado: Editar el archivo script_bd2.sql teniendo en cuenta las siguientes consideraciones:

- Para la tabla cliente, agregar PK y que el atributo estado tome por defecto el valor cdmx.
- Para la tabla artículo agregar una restricción que no permita ingresar un valor menor que 0 en el atributo precio, así como la PK.
- Para la tabla orden, agregar PK, permitir que el atributo fecha tome por defecto la hora del sistema y considerar que hay una relación 1:M entre la tabla cliente y orden, por lo que debe agregar la FK donde corresponda.

DDL corregido y para PostgreSQL:

```
1  -- 1. Crear base de datos y conectar
2  CREATE DATABASE datos_clase;
3  \c datos_clase
4
5  -- 2. Tabla Cliente
6  -- Agregar PK y default 'cdmx' en estado
7  CREATE TABLE cliente (
8      id_Cliente varchar(13) PRIMARY KEY,    -- PK Agregada
9      nombre char(50),
10     ap_Pat char(50) not null,
11     ap_Mat char(50),
12     estado varchar(25) DEFAULT 'cdmx'      -- Default agregado
13 );
14
15 -- 3. Tabla Articulo
16 -- Restriccion precio >= 0 y PK. Se cambia char a numeric.
17 CREATE TABLE articulo (
18     num_Articulo int PRIMARY KEY,
19     nombre_Articulo varchar(30) not null,
20     precio numeric(10,2) not null CHECK (precio >= 0),
21     categoria varchar(35) not null
22 );
23
24 -- 4. Tabla Orden
25 -- PK, default fecha sistema, Relacion 1:M con Cliente
26 CREATE TABLE orden (
27     id_Orden int PRIMARY KEY,
28     fecha timestamp not null DEFAULT NOW(),   -- Default hora sistema
29     id_Cliente varchar(13),
30     CONSTRAINT fk_cliente FOREIGN KEY (id_Cliente) REFERENCES
31     cliente(id_Cliente)
);
```

Query History

```

1 -- 1. Crear base de datos y conectar
2 CREATE DATABASE serie_sql;
3
4 -- 2.Tabla Cliente
5 -- Agregar PK y default 'cdmx' en estado
6 CREATE TABLE cliente (
7     id_Cliente varchar(13) PRIMARY KEY, -- PK Agregada
8     nombre char(50),
9     ap_Pat char(50) not null,
10    ap_Mat char(50),
11    estado varchar(25) DEFAULT 'cdmx' -- Default agregado
12 );
13
14 -- 3. Tabla Articulo
15 -- Restricción precio >= 0 y PK
16 CREATE TABLE articulo (
17     num_Articulo int PRIMARY KEY, -- PK Agregada
18     nombre_Articulo varchar(30) not null, -- Corregido "varrchar"
19     precio numeric(10,2) not null CHECK (precio >= 0), -- Cambiado a numeric para validar >= 0
20     categoria varchar(35) not null
21 );
22
23 -- 4. Tabla Orden
24 -- PK, default fecha sistema, Relación 1:M con Cliente
25 CREATE TABLE orden (
26     id_Orden int PRIMARY KEY, -- PK Agregada
27     fecha timestamp not null DEFAULT NOW(), -- Default hora sistema
28     id_Cliente varchar(13), -- FK debe ser del mismo tipo que en tabla cliente
29     CONSTRAINT fk_cliente FOREIGN KEY (id_Cliente) REFERENCES cliente(id_Cliente)
30 );

```

CREATE TABLE

Query returned successfully in 59 msec.

Figura 1: Creación de las tablas

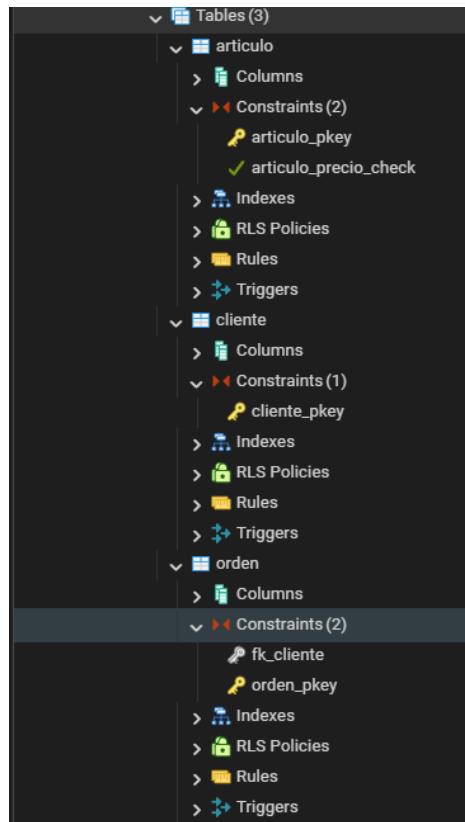


Figura 2: Tablas de nuestra base

2. Consultas de Vuelos (Ejercicios 2-13)

Nota: Todas las consultas incluyen la fecha del sistema y el usuario actual como se solicitó en las instrucciones generales.

Ejercicio 2

Enunciado: Indicar las ciudades que tienen más de un aeropuerto.

```
1  SELECT city, COUNT(*) AS cantidad_aeropuertos, CURRENT_USER, NOW()
2   FROM aeropuertos
3   GROUP BY city
4   HAVING COUNT(*) > 1;
```

```
40 --2. Ciudades con más de un aeropuerto
41 SELECT city, COUNT(*) AS cantidad_aeropuertos, CURRENT_USER, NOW()
42   FROM aeropuertos
43   GROUP BY city
44   HAVING COUNT(*) > 1;
45
46
47
48 --3. Aerolíneas que NO terminan en 'Inc.' ni 'Co' (Sin usar AND/OR)
49 SELECT airline, CURRENT_USER, NOW()
50   FROM aerolineas
51 WHERE airline !~ '(Inc.|Co)$';
52
53
54
55 --4. Aeropuertos implicados en el vuelo con mayor retraso de llegada
```

Data Output Messages Notifications

SQL

	city character varying	cantidad_aeropuertos bigint	current_user name	now timestamp with time zone
1	Jackson	2	postgres	2025-12-04 01:25:46.890293...
2	Springfield	2	postgres	2025-12-04 01:25:46.890293...
3	Albany	2	postgres	2025-12-04 01:25:46.890293...
4	Columbia	2	postgres	2025-12-04 01:25:46.890293...
5	New York	2	postgres	2025-12-04 01:25:46.890293...
6	Jacksonville	2	postgres	2025-12-04 01:25:46.890293...
7	Wilmington	2	postgres	2025-12-04 01:25:46.890293...
8	San Diego	2	postgres	2025-12-04 01:25:46.890293...
9	Chicago	2	postgres	2025-12-04 01:25:46.890293...
10	Houston	2	postgres	2025-12-04 01:25:46.890293...
11	Charleston	2	postgres	2025-12-04 01:25:46.890293...
12	Portland	2	postgres	2025-12-04 01:25:46.890293...
13	Columbus	2	postgres	2025-12-04 01:25:46.890293...
14	Rochester	2	postgres	2025-12-04 01:25:46.890293...

Total rows: 14 | Query complete 00:00:00.110

Figura 3: Ciudades con más de un aeropuerto

Ejercicio 3

Enunciado: Nombre de las aerolíneas que no terminan en Inc. ni en Co sin usar operadores AND y OR.

```
1 -- Se utiliza expresion regular para evitar el uso de AND/OR
2 SELECT airline, CURRENT_USER, NOW()
3 FROM aerolineas
4 WHERE airline !~ '(Inc\.|Co)$';
```

```
48 --3. Aerolineas que NO terminan en 'Inc.' ni 'Co' (Sin usar AND/OR)
49 SELECT airline, CURRENT_USER, NOW()
50 FROM aerolineas
51 WHERE airline !~ '(Inc\.|Co)$';
52
53 |
54
55 --4 Aeronuertos implicados en el vuelo con mayor retraso de llegada
```

Data Output Messages Notifications

	airline character varying	current_user name	now timestamp with time zone
1	JetBlue Airways	postgres	2025-12-04 01:29:45.039087...
2	Spirit Air Lines	postgres	2025-12-04 01:29:45.039087...
3	Southwest Airlines Co.	postgres	2025-12-04 01:29:45.039087...
4	Atlantic Southeast Airlin...	postgres	2025-12-04 01:29:45.039087...
5	Virgin America	postgres	2025-12-04 01:29:45.039087...

Total rows: 5 | Query complete 00:00:00.103

Figura 4: Aerolíneas que NO terminan en 'Inc.' ni 'Co'

Ejercicio 4

Enunciado: Indicar los nombres de los aeropuertos que estuvieron implicados en el vuelo que presentó el mayor retraso de llegada.

```
1  SELECT origin_airport, destination_airport, arrivale_delay,  
2      CURRENT_USER, NOW()  
3  FROM vuelos  
4  WHERE arrivale_delay = (SELECT MAX(arrivale_delay) FROM vuelos);
```

```
55  --4. Aeropuertos implicados en el vuelo con mayor retraso de llegada  
56  SELECT origin_airport, destination_airport, arrivale_delay, CURRENT_USER, NOW()  
57  FROM vuelos  
58  WHERE arrivale_delay = (SELECT MAX(arrivale_delay) FROM vuelos);  
59  
60
```

Data Output Messages Notifications

origin_airport character varying | destination_airport character varying | arrivale_delay smallint | current_user name | now timestamp with time zone

	origin_airport character varying	destination_airport character varying	arrivale_delay smallint	current_user name	now timestamp with time zone
1	BHM	DFW	1971	postgres	2025-12-04 01:32:50.323079...

Total rows: 1 | Query complete 00:00:00.835

Figura 5: Aeropuertos implicados en el vuelo con mayor retraso de llegada

Ejercicio 5

Enunciado: Mostrar aquella categoría (tabla artículo) que tiene el precio mínimo. La información debe estar agrupada.

```

1 SELECT categoria, MIN(precio) as precio_minimo, CURRENT_USER, NOW()
2 FROM articulo
3 GROUP BY categoria
4 ORDER BY precio_minimo ASC
5 LIMIT 1;

```

--5. Categoría con el precio mínimo (Tabla Artículo)

```

62 SELECT categoria, MIN(precio) as precio_minimo, CURRENT_USER, NOW()
63 FROM articulo
64 GROUP BY categoria
65 ORDER BY precio_minimo ASC
66 LIMIT 1;
67
68

```

	categoría character varying (35)	precio_minimo numeric	current_user name	now timestamp with time zone
1	accesorios	120.00	postgres	2025-12-04 02:02:49.073638...

Total rows: 1 | Query complete 00:00:00.099

Figura 6: Categoría precio mínimo

Ejercicio 6

Enunciado: Se desea conocer el nombre de aquellas aerolíneas cuyo segundo carácter del iata_code termina en X ó 9. Debe incluirse una columna que muestre dicha terminación.

```

1 SELECT
2     airline,
3     iata_code,

```

```

4     SUBSTRING(iata_code , 2, 1) as terminacion,
5     CURRENT_USER , NOW()
6 FROM aerolineas
7 WHERE SUBSTRING(iata_code , 2, 1) SIMILAR TO '[X9]';

```

The screenshot shows a PostgreSQL client interface with the following details:

- SQL Editor Content:**

```

70
71 --6. Aerolíneas (2do caracter IATA termina en X ó 9)
72 SELECT
73     airline,
74     iata_code,
75     SUBSTRING(iata_code , 2, 1) as terminacion,
76     CURRENT_USER, NOW()
77 FROM aerolineas
78 WHERE SUBSTRING(iata_code , 2, 1)SIMILAR TO '[X9]';
79
80

```
- Data Output:** A table showing the results of the query.
- Table Headers:**

	airline character varying	iata_code character varying	terminacion text	current_user name	now timestamp with time zone
--	------------------------------	--------------------------------	---------------------	----------------------	---------------------------------
- Table Data:**

1	Frontier Airlines I...	F9	9	postgres	2025-12-04 01:36:06.031774...
2	Virgin America	VX	X	postgres	2025-12-04 01:36:06.031774...
- Bottom Status:**

Total rows: 2 | Query complete 00:00:00.086

Figura 7: Aerolíneas (2do caracter IATA termina en X ó 9)

Ejercicio 7

Enunciado: Proporcionar el nombre de los aeropuertos cuya latitud se encuentre entre 40 y 41, y su longitud sea menor que el promedio de la longitud (tomando el promedio de aquellas observaciones cuya latitud esté entre 40 y 41).

```
1  SELECT airport, latitude, longitude, CURRENT_USER, NOW()
2  FROM aeropuertos
3  WHERE latitude BETWEEN 40 AND 41
4  AND longitude < (
5      SELECT AVG(longitude)
6      FROM aeropuertos
7      WHERE latitude BETWEEN 40 AND 41
8  );
```

```
82 --7. Aeropuertos (Lat 40-41) con Longitud < Promedio Longitud El promedio se calcula solo sobre los de Lat 40-41.
83 SELECT airport, latitude, longitude, CURRENT_USER, NOW()
84 FROM aeropuertos
85 WHERE latitude BETWEEN 40 AND 41
86 AND longitude < (
87     SELECT AVG(longitude)
88     FROM aeropuertos
89     WHERE latitude BETWEEN 40 AND 41
90 );
```

Data Output Messages Notifications

	airport character varying	latitude double precision	longitude double precision	current_user name	now timestamp with time zone
1	Arcata Airport	40.97812	-124.10862	postgres	2025-12-04 01:39:28.739458...
2	Elko Regional Airport	40.82493	-115.7917	postgres	2025-12-04 01:39:28.739458...
3	Central Nebraska Regional Airport	40.96747	-98.30861	postgres	2025-12-04 01:39:28.739458...
4	Yampa Valley AirportÂ (Yampa Valley Regio...	40.48118	-107.21766	postgres	2025-12-04 01:39:28.739458...
5	Lincoln AirportÂ (Lincoln Municipal)	40.85097	-96.75925	postgres	2025-12-04 01:39:28.739458...
6	Redding Municipal Airport	40.50898	-122.2934	postgres	2025-12-04 01:39:28.739458...
7	Salt Lake City International Airport	40.78839	-111.97777	postgres	2025-12-04 01:39:28.739458...
8	Valdez Airport	40.4409	-109.50992	postgres	2025-12-04 01:39:28.739458...

Total rows: 8 | Query complete 00:00:00.097

Figura 8: Latitud/Longitud

Ejercicio 8

Enunciado: ¿Cuántos aviones por aerolínea y día, fueron cancelados saliendo del aeropuerto de Honolulu?

```

1  SELECT airline, day, COUNT(*) as cancelados, CURRENT_USER, NOW()
2  FROM vuelos
3  WHERE origin_airport = 'HNL' AND cancelled = '1'
4  GROUP BY airline, day;

```

```

94 --8. Vuelos cancelados por aerolínea y día desde Honolulu
95 SELECT airline, day, COUNT(*) as cancelados, CURRENT_USER, NOW()
96 FROM vuelos
97 WHERE origin_airport = 'HNL' AND cancelled = '1'
98 GROUP BY airline, day;
99
100

```

Data Output Messages Notifications

The screenshot shows a PostgreSQL database interface with the following details:

- Query:** A SQL query to find canceled flights from Honolulu by airline and day.
- Results:** A table with 91 rows showing the count of canceled flights for each airline and day.
- Table Headers:** airline (character varying), day (smallint), cancelados (bigint), current_user (name), now (timestamp with time zone).
- Sample Data:** Rows show counts like (UA, 28, 3), (US, 1, 1), (VX, 2, 1) for various days of the month.
- Bottom Status:** Total rows: 91 | Query complete 00:00:00.506

	airline character varying	day smallint	cancelados bigint	current_user name	now timestamp with time zone
78	UA	28	3	postgres	2025-12-04 01:42:37.119851...
79	UA	29	3	postgres	2025-12-04 01:42:37.119851...
80	UA	30	6	postgres	2025-12-04 01:42:37.119851...
81	UA	31	1	postgres	2025-12-04 01:42:37.119851...
82	US	1	1	postgres	2025-12-04 01:42:37.119851...
83	US	3	1	postgres	2025-12-04 01:42:37.119851...
84	US	5	1	postgres	2025-12-04 01:42:37.119851...
85	US	6	1	postgres	2025-12-04 01:42:37.119851...
86	US	8	1	postgres	2025-12-04 01:42:37.119851...
87	US	17	1	postgres	2025-12-04 01:42:37.119851...
88	US	18	1	postgres	2025-12-04 01:42:37.119851...
89	US	23	1	postgres	2025-12-04 01:42:37.119851...
90	US	24	1	postgres	2025-12-04 01:42:37.119851...
91	VX	2	1	postgres	2025-12-04 01:42:37.119851...

Figura 9: Cancelados Honolulu

Ejercicio 9

Enunciado: Hacer un cross join entre la tabla cliente y la tabla aerolíneas.

```

1  -- Asumiendo permisos de lectura cruzada o dblink
2  SELECT c.nombre, a.airline, CURRENT_USER, NOW()
3  FROM datos_clase.public.cliente c
4  CROSS JOIN registro_vuelos.public.aerolineas a;

```

The screenshot shows a PostgreSQL client interface with the following details:

- SQL Editor:** Contains the following SQL code:

```

100
101 --9. Cross Join entre Cliente y Aerolíneas
102 SELECT
103     c.nombre,
104     a.airline,
105     CURRENT_USER as usuario,
106     NOW() as fecha
107     FROM cliente c
108     CROSS JOIN aerolineas_temp a;
109

```
- Data Output:** Shows the results of the query in a table format. The table has four columns: nombre, airline, usuario, and fecha.

	nombre character (50)	airline character varying (50)	usuario name	fecha timestamp with time zone
1	Jaime	American Airlines	postgres	2025-12-04 02:06:24.703068...
2	Luisa	American Airlines	postgres	2025-12-04 02:06:24.703068...
3	Mario	American Airlines	postgres	2025-12-04 02:06:24.703068...
4	Jaime	Aeromexico	postgres	2025-12-04 02:06:24.703068...
5	Luisa	Aeromexico	postgres	2025-12-04 02:06:24.703068...
6	Mario	Aeromexico	postgres	2025-12-04 02:06:24.703068...
7	Jaime	Delta Air Lines	postgres	2025-12-04 02:06:24.703068...
8	Luisa	Delta Air Lines	postgres	2025-12-04 02:06:24.703068...
9	Mario	Delta Air Lines	postgres	2025-12-04 02:06:24.703068...

- Status Bar:** Shows "Total rows: 9" and "Query complete 00:00:00.070".

Figura 10: Cross Join

Ejercicio 10

Enunciado: Cantidad de vuelos cancelados por día.

```

1  SELECT year, month, day, COUNT(*) as total_cancelados, CURRENT_USER ,
2      NOW()
2  FROM vuelos

```

```

3 WHERE cancelled = '1'
4 GROUP BY year, month, day;

```

```

108
109 --10. Cantidad de vuelos cancelados por día
110 SELECT year, month, day, COUNT(*) as total_cancelados, CURRENT_USER, NOW()
111 FROM vuelos
112 WHERE cancelled = '1'
113 GROUP BY year, month, day;
114
115

```

Data Output Messages Notifications

	year smallint	month smallint	day smallint	total_cancelados bigint	current_user name	now timestamp with time zone
352	2015	12	18	168	postgres	2025-12-04 01:47:27.426482...
353	2015	12	19	59	postgres	2025-12-04 01:47:27.426482...
354	2015	12	20	119	postgres	2025-12-04 01:47:27.426482...
355	2015	12	21	172	postgres	2025-12-04 01:47:27.426482...
356	2015	12	22	141	postgres	2025-12-04 01:47:27.426482...
357	2015	12	23	212	postgres	2025-12-04 01:47:27.426482...
358	2015	12	24	308	postgres	2025-12-04 01:47:27.426482...
359	2015	12	25	142	postgres	2025-12-04 01:47:27.426482...
360	2015	12	26	392	postgres	2025-12-04 01:47:27.426482...
361	2015	12	27	1158	postgres	2025-12-04 01:47:27.426482...
362	2015	12	28	2177	postgres	2025-12-04 01:47:27.426482...
363	2015	12	29	685	postgres	2025-12-04 01:47:27.426482...
364	2015	12	30	281	postgres	2025-12-04 01:47:27.426482...
365	2015	12	31	42	postgres	2025-12-04 01:47:27.426482...

Total rows: 365 | Query complete 00:00:00.494

Figura 11: Cancelados por día

Ejercicio 11

Enunciado: Seleccionar el nombre de los aeropuertos cuya segunda letra del iata code sea K ó X, sin usar operadores AND, NOT u OR.

```

1 SELECT airport, iata_code, CURRENT_USER, NOW()
2 FROM aeropuertos
3 WHERE SUBSTRING(iata_code, 2, 1) ~ '[KX]';

```

```

116
117 --11. Aeropuertos (2da letra IATA es K ó X) sin AND/NOT/OR
118 SELECT airport, iata_code, CURRENT_USER, NOW()
119 FROM aeropuertos
120 WHERE SUBSTRING(iata_code, 2, 1) ~ '[KX]';
121
122
123

```

Data Output Messages Notifications

The screenshot shows a PostgreSQL query interface with the following details:

- Query:**

```

SELECT airport, iata_code, CURRENT_USER, NOW()
FROM aeropuertos
WHERE SUBSTRING(iata_code, 2, 1) ~ '[KX]';

```
- Table Headers:** airport (character varying), iata_code (character varying), current_user (name), now (timestamp with time zone).
- Table Data:**

	airport	iata_code	current_user	now
1	King Salmon Airport	AKN	postgres	2025-12-04 01:49:47.768006...
2	Elko Regional Airport	EKO	postgres	2025-12-04 01:49:47.768006...
3	General Mitchell International Airport	MKE	postgres	2025-12-04 01:49:47.768006...
4	Muskegon County Airport	MKG	postgres	2025-12-04 01:49:47.768006...
5	Will Rogers World Airport	OKC	postgres	2025-12-04 01:49:47.768006...
6	Rock Springs-Sweetwater County Airport	RKS	postgres	2025-12-04 01:49:47.768006...
7	Texarkana Regional Airport (Webb Fie...	TXK	postgres	2025-12-04 01:49:47.768006...
- Bottom Status:** Total rows: 7 | Query complete 00:00:00.119

Figura 12: Aeropuertos K o X

Ejercicio 12

Enunciado: Indicar el nombre(s) de la aerolínea cuya distancia de vuelo es la mayor.

```
1 SELECT al.airline, v.distance, CURRENT_USER, NOW()
2 FROM vuelos v
3 JOIN aerolineas al ON v.airline = al.iata_code
4 ORDER BY distance DESC
5 LIMIT 1;
```

The screenshot shows a PostgreSQL query interface with the following details:

- SQL Editor Content:**

```
--12. Aerolínea con mayor distancia de vuelo
SELECT al.airline, v.distance, CURRENT_USER, NOW()
FROM vuelos v
JOIN aerolineas al ON v.airline = al.iata_code
ORDER BY distance DESC
LIMIT 1;
```
- Result Table:**

	airline	distance	current_user	now
1	Hawaiian Airlines I...	4983	postgres	2025-12-04 01:51:49.333863...
- Bottom Status Bar:** Total rows: 1 | Query complete 00:00:00.986

Figura 13: Aerolínea mayor distancia

Ejercicio 13

Enunciado: Indicar el nombre del aeropuerto de origen donde se presentó el mayor tiempo de vuelo.

```
1 SELECT origin_airport, air_time, CURRENT_USER, NOW()
2 FROM vuelos
3 WHERE air_time = (SELECT MAX(air_time) FROM vuelos);
```

```

132
133    --13. Aeropuerto de origen con mayor tiempo de vuelo
134    SELECT origin_airport, air_time, CURRENT_USER, NOW()
135    FROM vuelos
136    WHERE air_time = (SELECT MAX(air_time) FROM vuelos);

```

Data Output Messages Notifications

The screenshot shows a PostgreSQL query interface with the following details:

- Query:**

```

--13. Aeropuerto de origen con mayor tiempo de vuelo
SELECT origin_airport, air_time, CURRENT_USER, NOW()
FROM vuelos
WHERE air_time = (SELECT MAX(air_time) FROM vuelos);

```
- Data Output:** The results are displayed in a table with the following columns and data:

	origin_airport character varying	air_time smallint	current_user name	now timestamp with time zone
1	JFK	690	postgres	2025-12-04 01:53:14.563804...
2	JFK	690	postgres	2025-12-04 01:53:14.563804...

- Total rows: 2**
- Query complete 00:00:00.743**

Figura 14: Aeropuerto mayor tiempo vuelo

3. Modelado Entidad-Relación (Ejercicio 14)

Enunciado: Partiendo del MER proporcionado (Votante, Partido, Candidato), generar el mapeo a la representación intermedia de MR y generar el DDL. Para las FK debe establecerse la restricción cascade para borrado y actualización.

Solución - Modelo Relacional:

- **Partido** (siglas, nombre, año_fundacion)
- **Candidato** (rfc, nombre, años_trayectoria, siglas_partido_fk)
- **Candidato_Emails** (rfc_candidato, email) [Atributo multivalorado]
- **Votante** (id, nombre, dirección, edad, fecha_nacimiento, rfc_candidato_fk)

Solución - DDL:

```
1 CREATE TABLE Partido (
2     siglas varchar(10) PRIMARY KEY ,
3     nombre varchar(100) ,
4     anio_fundacion int
5 );
6
7 CREATE TABLE Candidato (
8     rfc varchar(13) PRIMARY KEY ,
9     nombre varchar(100) ,
10    anios_trayectoria int ,
11    siglas_partido varchar(10) ,
12    CONSTRAINT fk_partido FOREIGN KEY (siglas_partido)
13        REFERENCES Partido(siglas)
14        ON DELETE CASCADE ON UPDATE CASCADE
15 );
16
17 CREATE TABLE Candidato_Emails (
18     rfc_candidato varchar(13) ,
19     email varchar(100) ,
20     PRIMARY KEY (rfc_candidato, email),
21     CONSTRAINT fk_email_cand FOREIGN KEY (rfc_candidato)
22         REFERENCES Candidato(rfc)
23         ON DELETE CASCADE ON UPDATE CASCADE
24 );
25
26 CREATE TABLE Votante (
27     id int PRIMARY KEY ,
28     nombre varchar(100) ,
29     direccion varchar(150) ,
30     edad int ,
31     fecha_nacimiento date ,
32     rfc_candidato_voto varchar(13) ,
```

```

33     CONSTRAINT fk_voto FOREIGN KEY (rfc_candidato_voto)
34         REFERENCES Candidato(rfc)
35         ON DELETE CASCADE ON UPDATE CASCADE
36 );

```

Query History

```

1 CREATE TABLE Partido (
2     siglas varchar(10) PRIMARY KEY,
3     nombre varchar(100),
4     año_fundacion int
5 );
6
7 CREATE TABLE Candidato (
8     rfc varchar(13) PRIMARY KEY,
9     nombre varchar(100),
10    años_trayectoria int,
11    siglas_partido varchar(10),
12    CONSTRAINT fk_partido FOREIGN KEY (siglas_partido)
13        REFERENCES Partido(siglas)
14        ON DELETE CASCADE ON UPDATE CASCADE
15 );
16
17 -- Tabla para atributo multivalorado 'emails'
18 CREATE TABLE Candidato_Emails (
19     rfc_candidato varchar(13),
20     email varchar(100),
21     PRIMARY KEY (rfc_candidato, email),
22     CONSTRAINT fk_email_cand FOREIGN KEY (rfc_candidato)
23         REFERENCES Candidato(rfc)
24         ON DELETE CASCADE ON UPDATE CASCADE
25 );
26
27 CREATE TABLE Votante (
28     id int PRIMARY KEY,
29     nombre varchar(100),
30     direccion varchar(150),
31     edad int,
32     fecha_nacimiento date,

```

Data Output Messages Notifications

```

CREATE TABLE

Query returned successfully in 53 msec.

Total rows: | Query complete 00:00:00.053 |

```

Figura 15: script DDL

4. Procedimientos Almacenados (Ejercicio 15)

Enunciado: Genere una función o procedure que reciba el nombre de una categoría, itere sobre los registros (usando cursos), seleccione nombre y precio, y además incluya el precio mínimo y máximo de la categoría.

Solución:

```
1 CREATE OR REPLACE FUNCTION reporte_categoria(categoría_input varchar
)
2 RETURNS void AS $$ 
3 DECLARE
4     min_precio numeric;
5     max_precio numeric;
6     rec_articulo RECORD;
7     cur_articulos CURSOR FOR
8         SELECT nombre_Articulo, precio
9             FROM articulo WHERE categoria = categoría_input;
10 BEGIN
11     -- 1. Obtener Min y Max de la categoría
12     SELECT MIN(precio), MAX(precio) INTO min_precio, max_precio
13     FROM articulo WHERE categoria = categoría_input;
14
15     RAISE NOTICE 'Reporte para categoría: %', categoría_input;
16     RAISE NOTICE 'Rango Global: % - %', min_precio, max_precio;
17
18     -- 2. Abrir cursor e iterar
19     OPEN cur_articulos;
20 LOOP
21     FETCH cur_articulos INTO rec_articulo;
22     EXIT WHEN NOT FOUND;
23
24     RAISE NOTICE 'Artículo: %, Precio: %',
25             rec_articulo.nombre_Articulo, rec_articulo.precio;
26 END LOOP;
27 CLOSE cur_articulos;
28 END;
29 $$ LANGUAGE plpgsql;
```

Query Query History

```

64    -- 1. Obtener Min y Max de la categoría
65    SELECT MIN(precio), MAX(precio) INTO min_precio, max_precio
66    FROM articulo
67    WHERE categoria = categoria_input;
68
69    -- Cabecera del reporte
70    RAISE NOTICE 'Reporte para categoría: %', categoria_input;
71    RAISE NOTICE 'Precio Mínimo Global: %, Precio Máximo Global: %', min_precio, max_precio;
72    RAISE NOTICE '-----';
73
74    -- 2. Abrir cursor e iterar
75    OPEN cur_articulos;
76
77    LOOP
78        FETCH cur_articulos INTO rec_articulo;
79        EXIT WHEN NOT FOUND;
80
81        --Imprimir registro + estadísticos
82        RAISE NOTICE 'Artículo: %, Precio: % (Rango Cat: % - %)', 
83            rec_articulo.nombre_Articulo,
84            rec_articulo.precio,
85            min_precio,
86            max_precio;
87    END LOOP;
88
89    CLOSE cur_articulos;
90
91 END;
92 $$ LANGUAGE plpgsql;

```

Data Output Messages Notifications

```

NOTICE: Reporte para categoría: ropa
NOTICE: Precio Mínimo Global: 450.00, Precio Máximo Global: 650.00
NOTICE: -----
NOTICE: Artículo: Bermuda, Precio: 500.00 (Rango Cat: 450.00 - 650.00)
NOTICE: Artículo: Jeans, Precio: 650.00 (Rango Cat: 450.00 - 650.00)
NOTICE: Artículo: Falda, Precio: 450.00 (Rango Cat: 450.00 - 650.00)

Successfully run. Total query runtime: 70 msec.
1 rows affected.

```

Total rows: 1 Query complete 00:00:00.070

Figura 16: Procedimiento Almacenado

Referencias

- [1] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems*, 7th ed. Hoboken, NJ, USA: Pearson, 2016.
- [2] PostgreSQL Global Development Group, “PostgreSQL 16.1 Documentation,” [Online]. Available: <https://www.postgresql.org/docs/>. [Accessed: Dec. 03, 2025].
- [3] C. J. Date, *An Introduction to Database Systems*, 8th ed. Boston, MA, USA: Addison-Wesley, 2003.
- [4] A. Silberschatz, H. F. Korth, and S. Sudarshan, *Database System Concepts*, 7th ed. New York, NY, USA: McGraw-Hill Education, 2019.