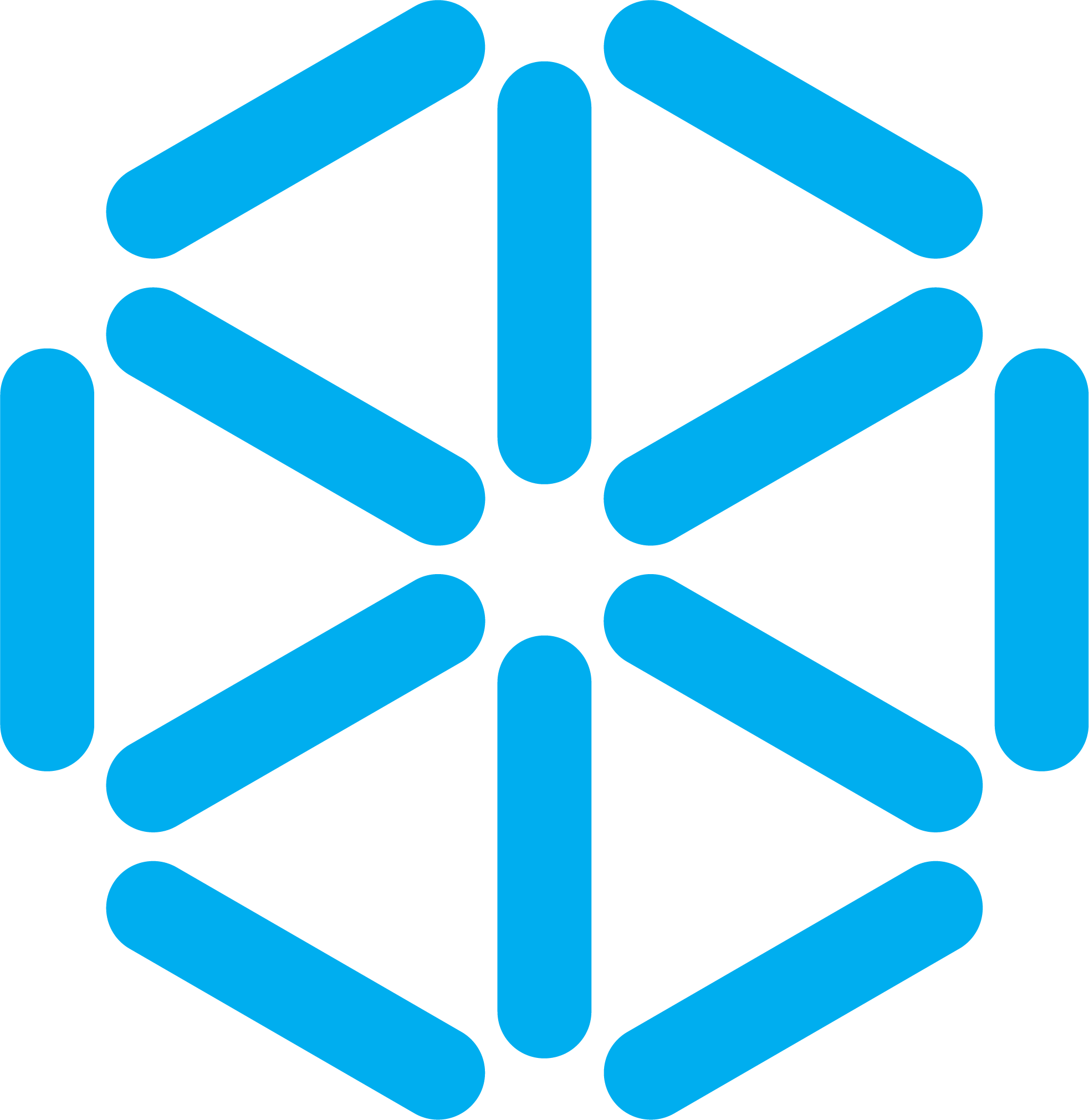
Laboratorio Base de Datos 2. (**REENTREGA**)



José Gabriel Hernández

Denís Barreto

Aparicio Oromí

Emiliano Langorta

9 de Mayo del 2025

Universidad Tecnológica del Uruguay.

Tecnólogo en Informática

Base de Datos 2 - Daniel Díaz

**Tabla de Contenidos.**

[Introducción 5](#_heading=h.gjdgxs)

[Modelo Entidad Relación (MER) 6](#_heading=h.gjdgxs)

[Modelo Relacional (MR) 7](#_heading=h.gjdgxs)

[DDL, DML y DCL 8](#_heading=h.gjdgxs)

[Ejercicios 9](#_heading=h.f6ge2846lxel)

[Query1 9](#_heading=h.1aec8npwc113)

[Letra 9](#_heading=h.khe7ipbjd78q)

[Resultado Final 9](#_heading=h.4uc8o37u9ixr)

[Query2 10](#_heading=h.9wwz8q9szin0)

[Letra 10](#_heading=h.k3k1i92skv8j)

[Resultado Final 10](#_heading=h.uu88wwgtn0t0)

[Query3 11](#_heading=h.e5n9o4g6dull)

[Letra 11](#_heading=h.r161xyffh9vq)

[Resultado Final 11](#_heading=h.xbjcn7qlhro)

[Query4 13](#_heading=h.7lv0erevmruw)

[Letra 13](#_heading=h.ss5x5gu2f5xi)

[Resultado Final 13](#_heading=h.1mfnctd31vaw)

[Query5 15](#_heading=h.a75dnowi8jo)

[Letra 15](#_heading=h.rvr1oun1jfgp)

[Resultado Final 15](#_heading=h.2ho8bnht6pla)

[Query6 17](#_heading=h.o8wze4ib4pjo)

[Letra 17](#_heading=h.pmfj3l1zv6b2)

[Resultado Final 17](#_heading=h.fi5xf6e3c1ni)

[Query7 19](#_heading=h.jucroyvbcia8)

[Letra 19](#_heading=h.gvawxhf68zco)

[Resultado Final 19](#_heading=h.4v28nh7dqxgu)

[Query8 22](#_heading=h.z06ohjz174ob)

[Letra 22](#_heading=h.9diwbnqq71mu)

[Datos de Prueba 22](#_heading=h.9w20xhs5jpb6)

[Resultado Final 22](#_heading=h.35srfyojuj3)

# 

# 

# Introducción

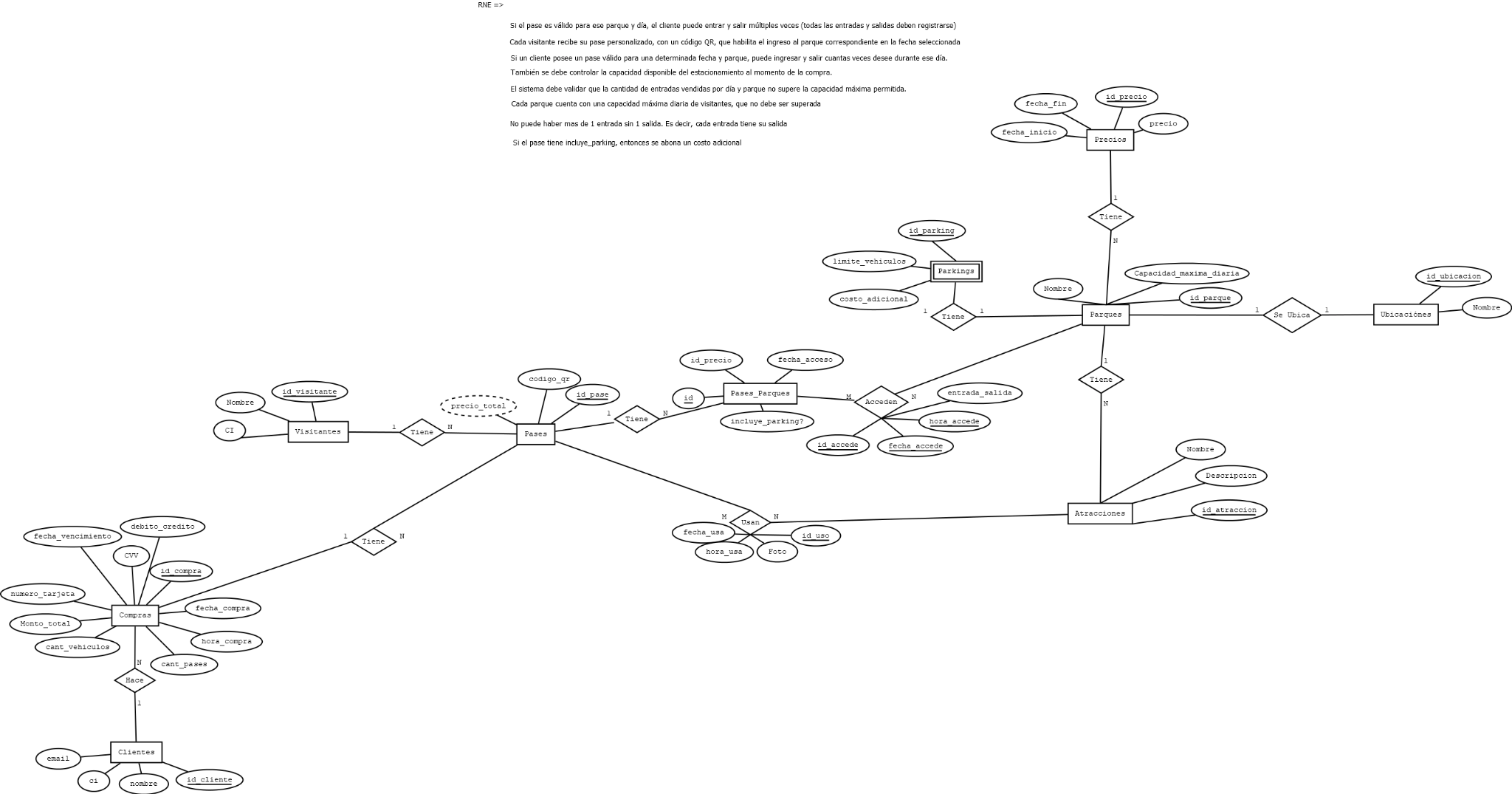
Este trabajo corresponde al Laboratorio Obligatorio 1 de la materia Bases de Datos 2. El objetivo principal es diseñar una base de datos que permita gestionar de forma eficiente las operaciones de un parque de diversiones con varias ubicaciones, distintos tipos de juegos y servicios como venta de entradas, estacionamiento, control de accesos y gestión de clientes.

A lo largo del trabajo se realizó un análisis del problema, se diseñó el Modelo Entidad Relación (MER) y su respectiva versión normalizada, junto al Modelo Relacional (MR) y se implementaron los scripts necesarios para crear las tablas, definir restricciones y cargar datos de prueba. También se desarrollaron consultas SQL que permiten obtener información clave sobre el uso del sistema, ingresos, concurrencia y comportamiento de los visitantes.

La idea es aplicar los conocimientos vistos en clase de forma práctica, resolviendo una situación realista que involucra múltiples entidades, relaciones y reglas de negocio.

# Modelo Entidad Relación (MER)

MER



Se adjunta la imagen en la carpeta, junto con el .DIA

# Modelo Relacional (MR)

**PARQUE**(id\_parque,id\_ubicacion,nombre,capacidad\_maxima\_diaria)

**PK**=**CK**=id\_parque

**FK**=id\_ubicacion

**PRECIO**(id\_precio,id\_parque,precio,fecha\_inicio,fecha\_fin);

**PK**=id\_precio,precio,fecha\_inicio,fecha\_fin

**FK**=id\_parque

**PARKING**(id\_parking,id\_parque,limite\_vehiculos,costo\_adicional)

**PK**=**CK**=id\_parking

**FK**=id\_parque

**UBICACION**(id\_ubicacion,nombre)

**PK**=**CK**=id\_ubicacion

**ATRACCIONES**(nombre,descripcion,id\_parque,id\_atraccion)

**PK**=**CK**=id\_atraccion

**FK**=id\_parque

**USA**(id\_uso,id\_pase,id\_atraccion,fecha,hora,foto)

**PK**=**CK**=id\_uso

**FK**=id\_parque,id\_atraccion

**ACCEDE**(id\_accede,id\_pase\_parque,id\_parque,fecha,hora,entrada\_salida)

**PK**=**CK**=id\_accede,id\_pase\_parque,id\_parque,fecha,hora

**FK**=id\_pase\_parque,id\_parque

**PASE**(id\_pase,id\_visitante,id\_compra,codigo\_qr,precio\_total);

**PK**=**CK**=id\_pase

**FK**=id\_visitante,id\_compra,id\_pase\_parque

**PaseParque**(id\_pase\_parque,id\_parque,id\_pase,incluye\_parking?,fecha\_accesso)

**PK**=**CK**=id\_pase\_parque

**FK**=id\_parque,id\_pase

**VISITANTE**(id\_visitante,ci,nombre)

**PK**=id\_visitante

**CK**=ci

**COMPRA**(id\_compra,id\_cliente,fecha\_compra,hora\_compra,monto\_total,debito\_credito,numero\_tarjeta,fecha\_vencimiento,cvv,cant\_pases,cant\_vehiculos)

**PK**=**CK**=id\_compra

**FK**=id\_pase,id\_cliente

**CLIENTE**(id\_cliente,email,ci,nombre)

**PK**=id\_cliente

**CK**=id\_cliente,email,ci

# DDL, DML y DCL

Se adjuntan en la carpeta

# Ejercicios

**Nota**: Los datos de prueba son los que están en el DML.

## Query1

### **Letra**

Los 3 entretenimiento más utilizados en un periodo dado (tomando como referencia los alumnos inscriptos). Entregar la solución en un script "query01.sql".

### **Resultado Final**

SELECT

a.nombre,

COUNT(u.id\_uso) AS total\_usos

FROM

usan u

JOIN atracciones a ON u.id\_atraccion = a.id\_atraccion

WHERE

u.fecha\_usa BETWEEN '2025-01-01' AND '2025-05-28'

GROUP BY

a.id\_atraccion,

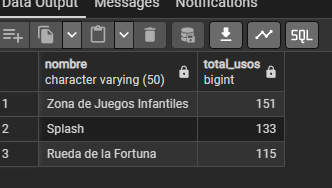
a.nombre

ORDER BY

total\_usos DESC

LIMIT

3;



## 

## Query2

### **Letra**

Total de ingresos por ventas de entradas en un periodo dado. Entregar la solución en un script "query02.sql".

### 

### 

### **Resultado Final**

### 

SELECT

SUM(p.precio\_total) AS total\_ingresos

FROM

pases p

JOIN

compras c ON p.id\_compra = c.id\_compra

WHERE

c.fecha\_compra BETWEEN '2025-01-01' AND '2025-12-31';

### 

## Query3

### **Letra**

Cantidad de clientes que ingresan al mismo entretenimiento en un período dado (**hacen uso del juego más de una vez**). Entregar la solución en un script "query03.sql".

### 

### **Resultado Final**

SELECT

a.id\_atraccion,

a.nombre AS nombre\_atraccion,

COUNT(\*) AS veces\_utilizado

FROM (

SELECT

p.id\_visitante,

u.id\_atraccion

FROM

usan u

JOIN pases p ON u.id\_pase = p.id\_pase

WHERE

u.fecha\_usa BETWEEN '2025-01-01' AND '2025-12-31'

GROUP BY

p.id\_visitante, u.id\_atraccion

HAVING

COUNT(\*) > 1

) sub

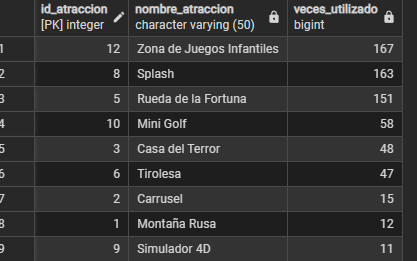
JOIN atracciones a ON sub.id\_atraccion = a.id\_atraccion

GROUP BY

a.id\_atraccion, a.nombre

ORDER BY

veces\_utilizado DESC;



## 

## 

## 

## Query4

### **Letra**

Dado el documento de un cliente devolver las fechas y parques a los que puede ingresar. Entregar la solución en un script "query04.sql"

### 

### **Resultado Final**

SELECT

cl.ci AS documento\_cliente,

pp.fecha\_acceso,

pa.nombre AS parque

FROM

clientes cl

JOIN

compras c ON cl.id\_cliente = c.id\_cliente

JOIN

pases p ON c.id\_compra = p.id\_compra

JOIN

pases\_parques pp ON p.id\_pase = pp.id\_pase

JOIN

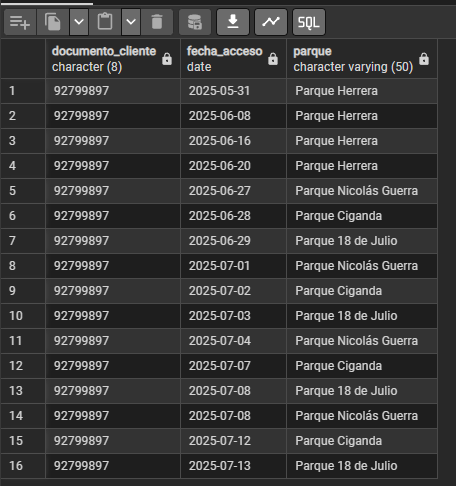
parques pa ON pp.id\_parque = pa.id\_parque

WHERE

cl.ci = '92799897'

ORDER BY

pp.fecha\_acceso, pa.nombre;



## 

## Query5

### **Letra**

Devolver la cantidad de entretenimientos utilizados por parques en una fecha dada. Entregar la solución en un script "query05.sql”

### 

### **Resultado Final**

SELECT

pa.nombre AS parque,

u.fecha\_usa,

COUNT(\*) AS cantidad\_entretenimientos\_usados

FROM

usan u

JOIN

atracciones a ON u.id\_atraccion = a.id\_atraccion

JOIN

parques pa ON a.id\_parque = pa.id\_parque

WHERE

u.fecha\_usa BETWEEN DATE '2025-01-01' AND DATE '2025-12-31'

GROUP BY

pa.nombre, u.fecha\_usa

ORDER BY

pa.nombre;



## Query6

### **Letra**

Retornar los dos días de la semana que el parque tiene mayor concurrencia, en un periodo dado. Entregar la solución en un script "query06.sql"

### 

### **Resultado Final**

SELECT

TO\_CHAR(a.fecha\_accede, 'FMDay') AS dia\_semana,

EXTRACT(ISODOW FROM a.fecha\_accede) AS orden\_dia,

COUNT(\*) AS total\_accesos

FROM acceden a

WHERE a.entrada\_salida = 'E'

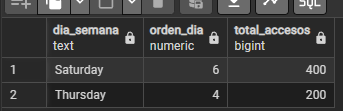
AND a.fecha\_accede BETWEEN DATE '2025-01-01' AND DATE '2025-12-31'

AND a.id\_parque = 2

GROUP BY TO\_CHAR(a.fecha\_accede, 'FMDay'), EXTRACT(ISODOW FROM a.fecha\_accede)

ORDER BY total\_accesos DESC

LIMIT 2;



**NOTA**:   
  
***EXTRACT***

Extract extrae una parte específica de la fecha, ya sea su año, mes, día. El DOW es el dia de la semana pero en numerico, 0=domingo, 6=sábado  
  
***TO\_CHAR***

Como precisamos que sea el DIA, usamos el TO\_CHAR que agarra una fecha y la traduce al formato que precisemos, en este caso solo el dia.

***ISODOW***

EXTRACT(ISODOW FROM fecha) obtiene el número del día de la semana según el estándar **ISO** 8601:

1 = Lunes

7 = Domingo

Esto permite ordenar los días correctamente empezando por lunes, a diferencia de **EXTRACT**(**DOW**) (donde 0 = domingo, 6 = sábado).

## 

## Query7

### **Letra**

Una vista materializada que informe en el año 2024 la cantidad de entradas vendidas por parque y el ingreso en USD. Entregar la solución en un script "vista\_query07.sql".

### 

### **Resultado Final**

CREATE MATERIALIZED VIEW vista\_ingresos\_2024 AS

SELECT

pa.id\_parque,

pa.nombre AS nombre\_parque,

COUNT(DISTINCT pp.id\_pase) AS cantidad\_entradas,

ROUND(

SUM(

pr.precio +

CASE

WHEN pp.incluye\_parking THEN COALESCE(park.costo\_adicional, 0)

ELSE 0

END

) / 40.0, 2

) AS ingreso\_usd

FROM

pases\_parques pp

JOIN parques pa ON pp.id\_parque = pa.id\_parque

JOIN precios pr ON pp.id\_precio = pr.id\_precio

JOIN pases p ON pp.id\_pase = p.id\_pase

JOIN compras c ON p.id\_compra = c.id\_compra

LEFT JOIN parkings park ON pa.id\_parque = park.id\_parque

WHERE

c.fecha\_compra BETWEEN '2025-01-01' AND '2025-12-31'

GROUP BY

pa.id\_parque, pa.nombre

ORDER BY

pa.id\_parque;

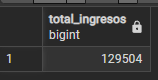
## 

SELECT \* FROM vista\_ingresos\_2024;



Nota: El ingreso, si es en un mismo rango de fecha que la Query 2, la sumatoria de INGRESO\_USD deberia dar el mismo resultado que la Query 2 pero modificando el campo

SUM(p.precio\_total) / 40 AS total\_ingresos

Con ese campo modificado, el resultado de la Query 2 es:  


Y, si sumamos el ingreso\_usd de los campos, entonces me va a dar el resultado. Para temas de prueba, se realizó la consulta agregando

ROUND(

SUM(

SUM(

pr.precio + CASE

WHEN pp.incluye\_parking THEN COALESCE(park.costo\_adicional, 0)

ELSE 0

END

)

) OVER () / 40.0,

2

) AS ingreso\_total\_usd

al select, lo que dio el siguiente resultado



que si lo comparamos con el resultado de la query 2 (en usd) es lo mismo (pero redondeado)

## 

## Query8

### **Letra**

El archivo denominado "dcl.sql" deberá entregar las sentencias SQL necesarias para crear:

1. Un usuario “**gerente**” que tenga **permisos totales** sobre la **base de datos del parque** (solo sobre esa).
2. Un usuario “**contador**” que **únicamente pueda ejecutar la vista del punto 7**.
3. Un usuario “**operador**” que pueda acceder a **visualizar e insertar y editar las entradas vendidas**, así como a los **datos de los clientes**

### **Datos de Prueba**

* No

### **Resultado Final**

-- Crear usuario gerente con permisos totales sobre todas las tablas y vistas del parque

CREATE USER gerente WITH PASSWORD 'gerente';

GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO gerente;

-- Crear usuario contador con acceso solo a la vista materializada

CREATE USER contador WITH PASSWORD 'contador';

GRANT SELECT ON vista\_ingresos\_2024 TO contador;

-- Crear usuario operador con acceso a datos de entradas vendidas y clientes

CREATE USER operador WITH PASSWORD 'operador';

-- Entradas = tabla pase, y compra

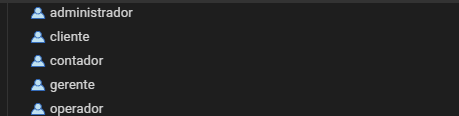
GRANT SELECT, INSERT, UPDATE ON pases TO operador;

GRANT SELECT, INSERT, UPDATE ON pases\_parques TO operador;

GRANT SELECT, INSERT, UPDATE ON compras TO operador;

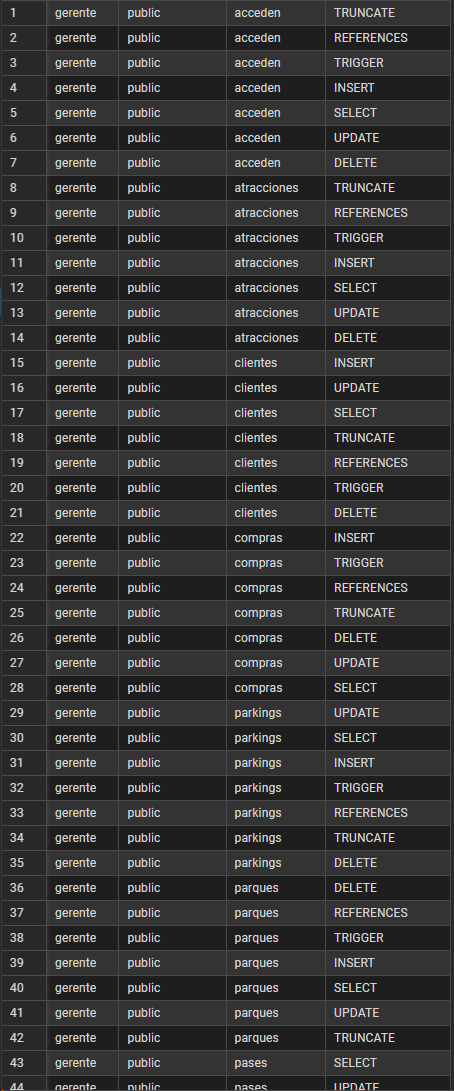
-- Datos de los clientes

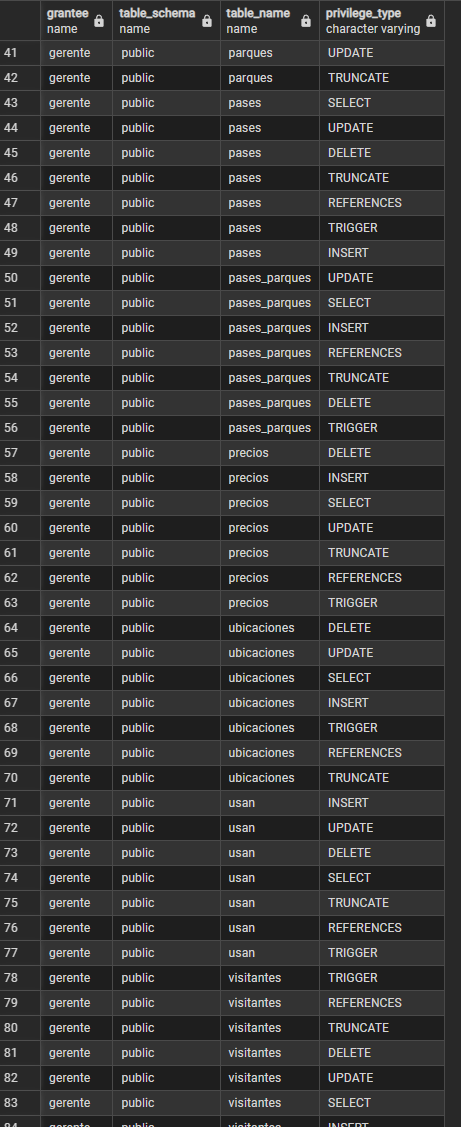
GRANT SELECT, INSERT, UPDATE ON clientes TO operador;

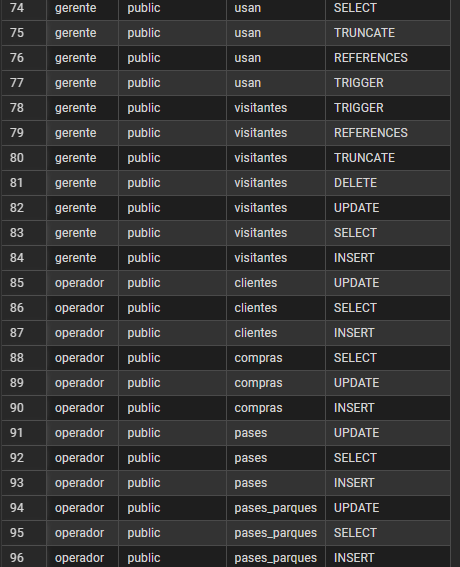


Como vemos, se crearon bien los usuarios “Gerente, contador y operador”

Con los campos que les asignamos:







Para ver los privilegios de esos usuarios se usó la siguiente consulta:

SELECT

grantee, table\_schema, table\_name, privilege\_type

FROM

information\_schema.role\_table\_grants

WHERE

grantee IN ('gerente', 'contador', 'operador')

ORDER BY

grantee, table\_name;