

Nombre de la práctica	Manual operadores set, joins y subconsultas			No.	
Asignatura:	Taller de Base de Datos	Carrera:	Ing. Sistemas Computacionales	Duración de la práctica (Hrs)	

Raúl Ciriaco Castillo 3501 ISIC

I. Competencia(s) específica(s):

II. Lugar de realización de la práctica (laboratorio, taller, aula u otro): Aula

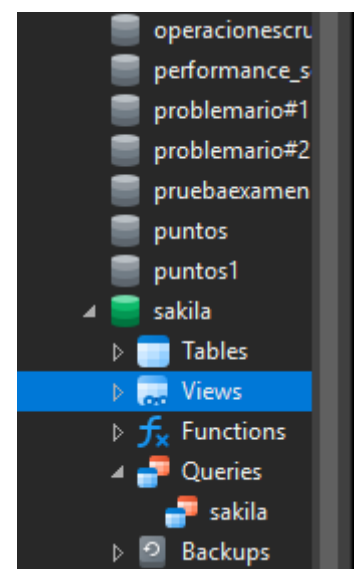
III. Material empleado:

- Computadora
- Software Navicat proporcionado por el docente

IV. Desarrollo de la práctica:

Para el desarrollo de esta práctica usaremos la base de datos que ya tenemos por defecto en nuestro software Navicat llamada Sakila, que se encuentra en el siguiente apartado.

Vamos a crear un query en el cual iremos realizando consultas subconsultas, unión de tablas como se mostrará a continuación anexando su respectivo resultado final.



1. Operadores

A continuación, se presentan las consultas con su resultado final de cada una:

```
# OPERADORES SET, JOINS Y SUBCONSULTAS

-- Calcula la longitud promedio de los nombres de los actores y la devuelve
SELECT AVG(LENGTH(first_name)) FROM actor;

-- Selecciona todos los actores cuyo nombre tiene una longitud mayor a 5.3050
SELECT * FROM actor WHERE LENGTH(first_name) > 5.3050;

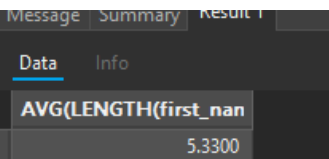
-- Selecciona todos los actores cuyo nombre tiene una longitud mayor a la longitud promedio de los nombres de los actores
SELECT * FROM actor WHERE LENGTH(first_name) > (SELECT AVG(LENGTH(first_name)) FROM actor);
```

Consulta 1:

```
# OPERADORES SET, JOINS Y SUBCONSULTAS

-- Calcula la longitud promedio de los nombres de los actores y la devuelve
SELECT AVG(LENGTH(first_name)) FROM actor;

-- Selecciona todos los actores cuyo nombre tiene una longitud mayor a 5.3050
```



Consulta 2:

```

6
7 -- Selecciona todos los actores cuyo nombre tiene una longitud mayor a 5.3050
8 SELECT * FROM actor WHERE LENGTH(first_name) > 5.3050;
9
10 -- Selecciona todos los actores cuyo nombre tiene una longitud mayor a la longitud pr

```

actor_id	first_name	last_name	last_update
1	PENELOPE	GUINNESS	2006-02-15 04:34:33
4	JENNIFER	DAVIS	2006-02-15 04:34:33
5	JOHNNY	LOLLOBRIGIDA	2006-02-15 04:34:33
8	MATTHEW	JOHANSSON	2006-02-15 04:34:33
10	CHRISTIAN	GABLE	2006-02-15 04:34:33
14	VIVIEN	BERGEN	2006-02-15 04:34:33
20	LUCILLE	TRACY	2006-02-15 04:34:33
21	KIRSTEN	PALTROW	2006-02-15 04:34:33
23	SANDRA	KILMER	2006-02-15 04:34:33
24	CAMERON	STREEP	2006-02-15 04:34:33
30	SANDRA	PECK	2006-02-15 04:34:33
34	AUDREY	OLIVIER	2006-02-15 04:34:33
39	GOLDIE	BRODY	2006-02-15 04:34:33

Consulta 3:

```

4 -- Calcula la longitud promedio de los nombres de los actores y la devuelve
5 SELECT AVG(LENGTH(first_name)) FROM actor;
6
7 -- Selecciona todos los actores cuyo nombre tiene una longitud mayor a 5.3050
8 SELECT * FROM actor WHERE LENGTH(first_name) > 5.3050;
9
10 -- Selecciona todos los actores cuyo nombre tiene una longitud mayor a la longitud promedio de los nombres de los actores
11 SELECT * FROM actor WHERE LENGTH(first_name) > (SELECT AVG(LENGTH(first_name)) FROM actor);
12
13

```

actor_id	first_name	last_name	last_update
1	PENELOPE	GUINNESS	2006-02-15 04:34:33
4	JENNIFER	DAVIS	2006-02-15 04:34:33
5	JOHNNY	LOLLOBRIGIDA	2006-02-15 04:34:33
8	MATTHEW	JOHANSSON	2006-02-15 04:34:33
10	CHRISTIAN	GABLE	2006-02-15 04:34:33
14	VIVIEN	BERGEN	2006-02-15 04:34:33
20	LUCILLE	TRACY	2006-02-15 04:34:33
21	KIRSTEN	PALTROW	2006-02-15 04:34:33
23	SANDRA	KILMER	2006-02-15 04:34:33
24	CAMERON	STREEP	2006-02-15 04:34:33
30	SANDRA	PECK	2006-02-15 04:34:33
34	AUDREY	OLIVIER	2006-02-15 04:34:33
39	GOLDIE	BRODY	2006-02-15 04:34:33

2. Relación de Tablas

De igual manera se mostrarán algunas consultas referentes a la relación de tablas con su respectivo funcionamiento y ejecución final:

```

14 -- RELACION DE TABLAS --
15 -- Selecciona el id de la categoría cuyo nombre es "Comedy"
16 SELECT category_id FROM category WHERE name="Comedy";
17
18 -- Selecciona el id de las películas que pertenecen a la categoría con id 5
19 SELECT film_id FROM film_category WHERE category_id = 5;
20
21 -- Selecciona el id de las películas que pertenecen a la categoría cuyo nombre es "Comedy"
22 SELECT film_id FROM film_category WHERE category_id = (SELECT category_id FROM category WHERE name="Comedy");
23
24 -- Selecciona el id de los actores que participan en las películas con los id 7, 28, 99
25 SELECT actor_id FROM film_actor WHERE film_id IN (7, 28, 99);
26
27

```

- Consulta 1:

```

-- RELACION DE TABLAS --
-- Selecciona el id de la categoría cuyo nombre es "Comedy"
SELECT category_id FROM category WHERE name="Comedy";

```

Data	Info
category_id	
5	

- Consulta 2:

```

18 -- Selecciona el id de las películas que pertenecen a la categoría con id 5
19 SELECT film_id FROM film_category WHERE category_id = 5;
20
21 -- Selecciona el id de las películas que pertenecen a la categoría cuyo nombre es "Comedy"
22 SELECT film_id FROM film_category WHERE category_id = (SELECT category_id FROM category WHERE name="Comedy");
23

```

film_id
7
28
99
119
127
159
176
182
188
202
242
247
265
276

- Consulta 3:

```

-- Selecciona el id de las películas que pertenecen a la categoría cuyo nombre es "Comedy"
SELECT film_id FROM film_category WHERE category_id = (SELECT category_id FROM category WHERE name="Comedy");

-- Selecciona el id de los actores que participan en las películas con los id 7, 28, 99
SELECT actor_id FROM film_actor WHERE film_id IN (7, 28, 99);

-- COMBINACION COMPLETA --

# Ejercicio 1:
-- Selecciona el id de los actores que participan en películas de la categoría "Comedy"
SELECT actor_id FROM film_actor WHERE film_id IN (SELECT film_id FROM film_category WHERE category_id = (SELECT category_id FROM category WHERE name="Comedy"));

# Ejercicio 2:
-- Selecciona el nombre y apellido de los actores que participan en películas de la categoría "Comedy"
SELECT first_name, last_name FROM actor WHERE actor_id IN (SELECT actor_id FROM film_actor WHERE film_id IN (SELECT film_id FROM film_category WHERE category_id = (SELECT category_id FROM category WHERE name="Comedy")));

```

film_id
7
28
99
119
127
159
176
182
188
202
242
247
265
276

- Ejercicio 1:

	Data	Info
30	# Ejercicio 1:	
31	-- Selecciona el id de los actores que participan en películas de la categoría "Comedy"	
32	SELECT actor_id FROM film_actor WHERE film_id IN (SELECT film_id FROM film_category WHERE category_id = (SELECT category_id FROM category WHERE name="Comedy"));	actor_id
33		99
34	# Ejercicio 2:	133
35	-- Selecciona el nombre y apellido de los actores que participan en películas de la categoría "Comedy"	162
36	SELECT first_name, last_name FROM actor WHERE actor_id IN (SELECT actor_id FROM film_actor WHERE film_id IN (SELECT film_id FROM film_category WHERE category_id = (SELECT category_id FROM category WHERE name = "Comedy")));	170
37		185
38	# Ejercicio 3:	74
39	-- Encuentra los títulos de las películas que nunca han sido alquiladas --	162
40	SELECT title FROM film WHERE film_id NOT IN (SELECT film_id FROM inventory WHERE inventory_id IN (SELECT inventory_id FROM rental WHERE rental_id));	48
41		144
		17
		34
		37

- Ejercicio 2:

	Data	Info
	# Ejercicio 1:	
	-- Selecciona el id de los actores que participan en películas de la categoría "Comedy"	
	SELECT actor_id FROM film_actor WHERE film_id IN (SELECT film_id FROM film_category WHERE category_id = (SELECT category_id FROM category WHERE name="Comedy"));	first_name last_name
		JIM MOSTEL
		RICHARD PENN
		OPRAH KILMER
		MENA HOPPER
		MICHAEL BOLGER
		MILLA KEITEL
		FRANCES DAY-LEWIS
		ANGELA WITHERSPOON
		HELEN VOIGHT
		AUDREY OLIVIER
		VAL BOLGER
		CHRISTIAN AKROYD

- Ejercicio 3:

	Data	Info
	# Ejercicio 1:	
	-- Selecciona el id de los actores que participan en películas de la categoría "Comedy"	
	SELECT actor_id FROM film_actor WHERE film_id IN (SELECT film_id FROM film_category WHERE category_id = (SELECT category_id FROM category WHERE name="Comedy"));	title
		ALICE FANTASIA
		APOLLO TEEN
		ARGONAUTS TO
		ARK RIDGEMONT
		ARSENIC INDEPE
		BOONDOCK BAL
		BUTCH PANTHEI
		CATCH AMISTAI
		CHINATOWN GL
		CHOCOLATE DU
		COMMANDMEN
		CROSSING DIVOF

4. Joins

Posteriormente conocimos los Joins y en breve su definición con un ejemplo como se muestra en pantalla:

```
-- JOINS EXISTENTES --
/*
INNER JOIN:
Devuelve las filas que tienen coincidencias en ambas tablas.
Ejemplo: SELECT * FROM A INNER JOIN B ON A.id = B.id;

FULL JOIN:
Devuelve todas las filas cuando hay una coincidencia en una de las tablas. Si no hay coincidencia, el resultado contiene NULL.
MySQL no soporta directamente FULL OUTER JOIN, pero se puede simular con una combinación de LEFT JOIN y RIGHT JOIN con UNION.
Ejemplo: SELECT * FROM A LEFT JOIN B ON A.id = B.id UNION SELECT * FROM A RIGHT JOIN B ON A.id = B.id;

CROSS JOIN:
Devuelve el producto cartesiano de las dos tablas, es decir, todas las combinaciones posibles de filas.
Ejemplo: SELECT * FROM A CROSS JOIN B;

LEFT JOIN (o LEFT OUTER JOIN):
Devuelve todas las filas de la tabla izquierda y las filas coincidentes de la tabla derecha. Si no hay coincidencias, el resultado contiene NULL para las columnas de la tabla derecha.
Ejemplo: SELECT * FROM A LEFT JOIN B ON A.id = B.id;

RIGHT JOIN (o RIGHT OUTER JOIN):
Devuelve todas las filas de la tabla derecha y las filas coincidentes de la tabla izquierda. Si no hay coincidencias, el resultado contiene NULL para las columnas de la tabla izquierda.
Ejemplo: SELECT * FROM A RIGHT JOIN B ON A.id = B.id;
*/
```

Realizamos algunos ejercicios de ejemplo como se muestra a continuación:

```
# .. CONSULTAS Y FUNCIONAMIENTO ..

# Ejemplo 1: INNER JOIN:
SELECT f.title, c.name as category_name -- Selecciona el título de la película y el nombre de la categoría, renombrando el nombre de la categoría como 'category_name'
FROM film as f -- Utiliza la tabla 'film' con el alias 'f'
INNER JOIN film_category as fc -- Realiza un INNER JOIN con la tabla 'film_category' usando el alias 'fc'
ON f.film_id = fc.film_id -- Condición de unión: los 'film_id' en ambas tablas deben coincidir
INNER JOIN category as c -- Realiza un INNER JOIN con la tabla 'category' usando el alias 'c'
ON fc.category_id = c.category_id; -- Condición de unión: los 'category_id' en ambas tablas deben coincidir

# Ejemplo 2: INNER JOIN:
SELECT * FROM film -- Selecciona todas las columnas de la tabla 'film'
INNER JOIN film_category ON film.film_id = film_category.film_id -- Realiza un INNER JOIN con la tabla 'film_category' basado en 'film_id'
JOIN category AS c ON film_category.category_id = c.category_id; -- Realiza un JOIN con la tabla 'category' usando el alias 'c' basado en 'category_id'

# Ejemplo 3: INNER JOIN:
SELECT film.title, c.name -- Selecciona el título de la película y el nombre de la categoría
FROM film -- Utiliza la tabla 'film'
INNER JOIN film_category ON film.film_id = film_category.film_id -- Realiza un INNER JOIN con la tabla 'film_category' basado en 'film_id'
JOIN category AS c ON film_category.category_id = c.category_id; -- Realiza un JOIN con la tabla 'category' usando el alias 'c' basado en 'category_id'
```

```
# Ejemplo 4: LEFT JOIN:
-- Encuentra el título de las películas y las veces que han sido alquiladas --

SELECT f.title, COUNT(r.rental_id) AS cantidad_rentas
FROM film AS f
LEFT JOIN inventory AS i ON f.film_id = i.film_id
LEFT JOIN rental AS r ON i.inventory_id=r.inventory_id
GROUP BY f.title;

# Ejemplo 5: LEFT JOIN:
-- Muestra los nombres de los actores y los nombres de las películas en las que han participado --
-- ( first_name, last_name, nombre_pelicula ) --
SELECT a.first_name, a.last_name, f.title AS nombre_pelicula
FROM film AS f
LEFT JOIN film_actor AS fa ON f.film_id = fa.film_id
LEFT JOIN actor AS a ON fa.actor_id = a.actor_id
ORDER BY f.title;
```


1. Consulta:

```

67
68 # Ejemplo 1: INNER JOIN:
69 SELECT f.title, c.name AS category_name -- Selecciona el título de la película y el nombre de la categoría,
70 FROM film AS f -- Utiliza la tabla 'film' con el alias 'f'
71 INNER JOIN film_category AS fc -- Realiza un INNER JOIN con la tabla 'film_category' usando el alias 'fc'
72 ON f.film_id = fc.film_id -- Condición de unión: los 'film_id' en ambas tablas deben coincidir
73 INNER JOIN category AS c -- Realiza un INNER JOIN con la tabla 'category' usando el alias 'c'
74 ON fc.category_id = c.category_id; -- Condición de unión: los 'category_id' en ambas tablas deben coincidir
75

```

Message Summary Result 1

title	category_name
AMADEUS HOLY	Action
AMERICAN CIRC	Action
ANTITRUST TOM	Action
ARK RIDGEMONT	Action
BAREFOOT MAN	Action
BERETS AGENT	Action
BRIDE INTRIGUE	Action
BULL SHAWSHAI	Action
CADDYSHACK JE	Action
CAMPUS REMEM	Action

2. Consulta:

```

76 # Ejemplo 2: INNER JOIN:
77 SELECT * FROM film -- Selecciona todas las columnas de la tabla 'film'
78 INNER JOIN film_category ON film.film_id = film_category.film_id -- Realiza un INNER JOIN con la tabla 'film_category' basado en 'film_id'
79 INNER JOIN category AS c ON film_category.category_id = c.category_id; -- Realiza un JOIN con la tabla 'category' usando el alias 'c' basado en 'category_id'
80
81 # Ejemplo 3: INNER JOIN:

```

Message Summary Result 1

film_id	title	description	release_year	language_id	original_language_id	rental_duration	rental_rate	length	replacement_cost	rating	special_features	last_up
19	AMADEUS HOLY	A Emotional Display of a P	2008	1	1	8	0.99	113	20.99	PG	Commentaries, Deleted Sci	2008-02
21	AMERICAN CIRC	A Insightful Drama of a Ge	2008	1	1	8	4.99	129	11.99	R	Commentaries, Behind the	2008-02
29	ANTITRUST TOM	A Fateful Yarn of a Woman	2008	1	1	5	2.99	180	11.99	NC-17	Trailers, Commentaries, Del	2008-02
38	ARK RIDGEMONT	A Beautiful Yarn of a Poem	2008	1	1	6	0.99	88	25.99	NC-17	Trailers, Commentaries, Del	2008-02
50	BAREFOOT MAN	A Intrepid Story of a Cat A	2008	1	1	8	2.99	129	15.99	G	Trailers, Commentaries	2008-02
67	BERETS AGENT	A Taut Saga of a Crocodile	2008	1	1	5	2.99	77	24.99	PG-13	Deleted Scenes	2008-02
97	BRIDE INTRIGUE	A Epic Tale of a Robot And	2008	1	1	7	0.99	98	24.99	G	Trailers, Commentaries, Del	2008-02
109	BULL SHAWSHAI	A Fanciful Drama of a Mo	2008	1	1	6	0.99	125	21.99	NC-17	Deleted Scenes	2008-02
111	CADDYSHACK JE	A Awe-Inspiring Episode of	2008	1	1	3	0.99	52	17.99	NC-17	Commentaries, Deleted Sci	2008-02
115	CAMPUS REMEM	A Astounding Drama of a	2008	1	1	5	2.99	167	27.99	R	Behind the Scenes	2008-02
126	CASUALTIES ENC	A Insightful Yarn of a A Sh	2008	1	1	3	4.99	179	16.99	G	Trailers	2008-02
130	CELEBRITY HOPR	A Amazing Documentary i	2008	1	1	7	0.99	110	24.99	PG-13	Deleted Scenes, Behind the	2008-02
162	CLUELESS BUCKE	A Taut Tale of a Car And a	2008	1	1	4	2.99	95	13.99	R	Trailers, Deleted Scenes, Del	2008-02
194	CROWD URASE	A Awe-Inspiring Document	2008	1	1	8	0.99	154	22.99	PG	Trailers, Commentaries, Del	2008-02
205	DANCES HOME	A Insightful Reflection of a	2008	1	1	8	0.99	90	22.99	NC-17	Trailers, Commentaries, Del	2008-02

3. Consulta:

```

81 # Ejemplo 3: INNER JOIN:
82 SELECT film.title, c.name -- Selecciona el título de la película y el nombre de la categoría
83 FROM film -- Utiliza la tabla 'film'
84 INNER JOIN film_category ON film.film_id = film_category.film_id -- Realiza un INNER JOIN con la
85 JOIN category AS c ON film_category.category_id = c.category_id; -- Realiza un JOIN con la tabla
86

```

Message Summary Result 1

title	name
AMADEUS HOLY	Action
AMERICAN CIRC	Action
ANTITRUST TOM	Action
ARK RIDGEMONT	Action
BAREFOOT MAN	Action
BERETS AGENT	Action
BRIDE INTRIGUE	Action
BULL SHAWSHAI	Action
CADDYSHACK JE	Action
CAMPUS REMEM	Action
CASUALTIES ENC	Action
CELEBRITY HOPR	Action
CLUELESS BUCKE	Action

4. Consulta:

```

86
87 # Ejemplo 4: LEFT JOIN:
88 -- Encuentra el título de las películas y las veces que han sido alquiladas --
89
90 SELECT f.title, COUNT(r.rental_id) AS cantidad_rentas
91 FROM film AS f
92 LEFT JOIN inventory AS i ON f.film_id = i.film_id
93 LEFT JOIN rental AS r ON i.inventory_id=r.inventory_id
94 GROUP BY F.title;
95

```

Message Summary Result 1

Data Info

title	cantidad_rentas
ACADEMY DINO:	23
ACE GOLDFINGER	7
ADAPTATION HO	12
AFFAIR PREJUDIC	23
AFRICAN EGG	12
AGENT TRUMAN	21
AIRPLANE SIERRA	15
AIRPORT POLLO	18
ALABAMA DEVIL	12
ALADDIN CALEN	23
ALAMO VIDEOTA	24
ALASKA PHANTO	26
ALI FOREVER	9

5. Consulta:

```

96 # Ejemplo 5: LEFT JOIN:
97 -- Muestra los nombres de los actores y los nombres de las películas en las que han participado --
98 -- ( first_name, last_name, nombre_pelicula ) --
99
100 SELECT a.first_name, a.last_name, f.title AS nombre_pelicula
101 FROM film AS f
102 LEFT JOIN film_actor AS fa ON f.film_id = fa.film_id
103 LEFT JOIN actor AS a ON fa.actor_id = a.actor_id
104 ORDER BY f.title;

```

Message Summary Result 1

Data Info

first_name	last_name	nombre_pelicula
PENELOPE	GUINNESS	ACADEMY DINOSAUR
CHRISTIAN	GABLE	ACADEMY DINOSAUR
LUCILLE	TRACY	ACADEMY DINOSAUR
SANDRA	PECK	ACADEMY DINOSAUR
JOHNNY	CAGE	ACADEMY DINOSAUR
MENA	TEMPLE	ACADEMY DINOSAUR
WARREN	NOLTE	ACADEMY DINOSAUR
OPRAH	KILMER	ACADEMY DINOSAUR
ROCK	DUKAKIS	ACADEMY DINOSAUR
MARY	KEITEL	ACADEMY DINOSAUR
BOB	FAWCETT	ACE GOLDFINGER
MINNIE	ZELLWEGER	ACE GOLDFINGER
SEAN	GUINNESS	ACE GOLDFINGER
CHRIS	DEPP	ACE GOLDFINGER
NICK	WAHLBERG	ADAPTATION HOLES

6. Consulta:

5. Otras Consultas (UNION, EXCEPT, DISTINCT)

- Las consultas de UNION combinan resultados de múltiples selecciones y eliminan duplicados.
- Las consultas de EXCEPT devuelven filas que están en el primer conjunto, pero no en el segundo.
- DISTINCT se utiliza para eliminar duplicados y devolver solo valores únicos.

A continuación, se muestran algunos ejemplos de uso para estas consultas con su respectiva captura de ejecución:

```
-- Seleccionar todos los nombres de los actores
SELECT first_name FROM actor;

-- Seleccionar todos los nombres de los clientes
SELECT first_name FROM customer;

-- Combinar ambas selecciones y eliminar duplicados
SELECT first_name FROM actor
UNION
SELECT first_name FROM customer;

-- Encuentra las películas que no han sido alquiladas
SELECT title FROM film
EXCEPT
SELECT f.title FROM film AS f -- Selecciona el título de las películas
JOIN inventory AS i ON f.film_id = i.film_id -- Realiza un JOIN con la tabla 'inventory' usando 'film_id'
JOIN rental AS r ON i.inventory_id = r.inventory_id; -- Realiza un JOIN con la tabla 'rental' usando 'inventory_id'

-- Selecciona las ciudades distintas asociadas con las direcciones del personal
SELECT DISTINCT city FROM city -- Selecciona las ciudades distintas de la tabla 'city'
JOIN address ON city.city_id = address.city_id -- Realiza un JOIN con la tabla 'address' usando 'city_id'
JOIN staff ON address.address_id = staff.address_id; -- Realiza un JOIN con la tabla 'staff' usando 'address_id'
```

- Consulta 1:

```
112
113 -- OTRA MANERA USANDO (UNION, EXCEPT, DISTINCT) --
114 -- Seleccionar todos los nombres de los actores
115 SELECT first_name FROM actor;
116
```

Message	Summary	Result 1
Data	Info	
first_name		
PENELOPE		
NICK		
ED		
JENNIFER		
JOHNNY		
BETTE		
GRACE		
MATTHEW		
JOE		
CHRISTIAN		
ZERO		
KARL		
UMA		
VIVIEN		

- Consulta 2:

```

117 -- Seleccionar todos los nombres de los clientes
118 SELECT first_name FROM customer;
119
120 -- Combinar ambas selecciones y eliminar duplicados

```

first_name
MARY
PATRICIA
LINDA
BARBARA
ELIZABETH
JENNIFER
MARIA
SUSAN
MARGARET
DOROTHY
LISA

- Consulta 3:

```

119
120 -- Combinar ambas selecciones y eliminar duplicados
121 SELECT first_name FROM actor
122 UNION
123 SELECT first_name FROM customer;
124
125 -- Encuentra las películas que no han sido alquiladas

```

first_name
PENELOPE
NICK
ED
JENNIFER
JOHNNY
BETTE
GRACE
MATTHEW
JOE
CHRISTIAN
ZERO
KARL
UMA

- Consulta 4:

```

124
125 -- Encuentra las películas que no han sido alquiladas
126 SELECT title FROM film
127 EXCEPT
128 SELECT f.title FROM film AS f -- Selecciona el título de film
129 JOIN inventory AS i ON f.film_id = i.film_id -- Realiza un
130 JOIN rental AS r ON i.inventory_id = r.inventory_id; -- Real
131

```

title
ALICE FANTASIA
APOLLO TEEN
ARGONAUTS TO
ARK RIDGEMONT
ARSENIC INDEPE
BOONDOCK BAL
BUTCH PANTHEI
CATCH AMSTAI
CHINATOWN GL
CHOCOLATE DU
COMMANDMEN
CROSSING DIVOF
CROWDS TELEM
CRYSTAL BREAK
DAZED PUNK
DELIVERANCE M

- Consulta 5:

```

132 -- Selecciona las ciudades distintas asociadas con las
133 SELECT DISTINCT city FROM city -- Selecciona las ciudad
134 JOIN address ON city.city_id = address.city_id -- Real
135 JOIN staff ON address.address_id = staff.address_id; --
136
137

```

city
Lethbridge
Woodridge

6. CODIGO UTILIZADO Y REALIZADO EN CLASE PARA ESTA PRACTICA:

OPERADORES SET, JOINS Y SUBCONSULTAS

-- Calcula la longitud promedio de los nombres de los actores y la devuelve

```
SELECT AVG(LENGTH(first_name)) FROM actor;
```

-- Selecciona todos los actores cuyo nombre tiene una longitud mayor a 5.3050

```
SELECT * FROM actor WHERE LENGTH(first_name) > 5.3050;
```

-- Selecciona todos los actores cuyo nombre tiene una longitud mayor a la longitud promedio de los nombres de los actores

```
SELECT * FROM actor WHERE LENGTH(first_name) > (SELECT AVG(LENGTH(first_name)) FROM actor);
```

-- RELACION DE TABLAS --

-- Selecciona el id de la categoría cuyo nombre es "Comedy"

```
SELECT category_id FROM category WHERE name="Comedy";
```

-- Selecciona el id de las películas que pertenecen a la categoría con id 5

```
SELECT film_id FROM film_category WHERE category_id = 5;
```

-- Selecciona el id de las películas que pertenecen a la categoría cuyo nombre es "Comedy"

```
SELECT film_id FROM film_category WHERE category_id = (SELECT category_id FROM category WHERE name="Comedy");
```

-- Selecciona el id de los actores que participan en las películas con los id 7, 28, 99

```
SELECT actor_id FROM film_actor WHERE film_id IN (7, 28, 99);
```

-- SUBCONSULTAS --

Ejercicio 1:

-- Selecciona el id de los actores que participan en películas de la categoría "Comedy"

```
SELECT actor_id FROM film_actor WHERE film_id IN (SELECT film_id FROM film_category WHERE category_id = (SELECT category_id FROM category WHERE name="Comedy"));
```

Ejercicio 2:

-- Selecciona el nombre y apellido de los actores que participan en películas de la categoría "Comedy"

```
SELECT first_name, last_name FROM actor WHERE actor_id IN (SELECT actor_id FROM film_actor WHERE film_id IN (SELECT film_id FROM film_category WHERE category_id = (SELECT category_id FROM category WHERE name = "Comedy")));
```

Ejercicio 3:

-- Encuentra los títulos de las películas que nunca han sido alquiladas --

```
SELECT title FROM film WHERE film_id NOT IN (SELECT film_id FROM inventory WHERE inventory_id IN(SELECT inventory_id FROM rental WHERE rental_id));
```

-- JOINS EXISTENTES --

/*
INNER JOIN:
Devuelve las filas que tienen coincidencias en ambas tablas.
Ejemplo: SELECT * FROM A INNER JOIN B ON A.id = B.id;

FULL JOIN:
Devuelve todas las filas cuando hay una coincidencia en una de las tablas. Si no hay coincidencia, el resultado contiene NULL.
MySQL no soporta directamente FULL OUTER JOIN, pero se puede simular con una combinación de LEFT JOIN y RIGHT JOIN con UNION.
Ejemplo: SELECT * FROM A LEFT JOIN B ON A.id = B.id UNION SELECT * FROM A RIGHT JOIN B ON A.id = B.id;

CROSS JOIN:
Devuelve el producto cartesiano de las dos tablas, es decir, todas las combinaciones posibles de filas.
Ejemplo: SELECT * FROM A CROSS JOIN B;

LEFT JOIN (o LEFT OUTER JOIN):
Devuelve todas las filas de la tabla izquierda y las filas coincidentes de la tabla derecha. Si no hay coincidencias, el resultado contiene NULL para las columnas de la tabla derecha.
Ejemplo: SELECT * FROM A LEFT JOIN B ON A.id = B.id;

RIGHT JOIN (o RIGHT OUTER JOIN):
Devuelve todas las filas de la tabla derecha y las filas coincidentes de la tabla izquierda. Si no hay coincidencias, el resultado contiene NULL para las columnas de la tabla izquierda.
Ejemplo: SELECT * FROM A RIGHT JOIN B ON A.id = B.id;

*/

.. CONSULTAS Y FUNCIONAMIENTO ..

Ejemplo 1: INNER JOIN:
SELECT f.title, c.name as category_name -- Selecciona el título de la película y el nombre de la categoría, renombrando el nombre de la categoría como 'category_name'
FROM film as f -- Utiliza la tabla 'film' con el alias 'f'
INNER JOIN film_category as fc -- Realiza un INNER JOIN con la tabla 'film_category' usando el alias 'fc'
ON f.film_id = fc.film_id -- Condición de unión: los 'film_id' en ambas tablas deben coincidir
INNER JOIN category as c -- Realiza un INNER JOIN con la tabla 'category' usando el alias 'c'
ON fc.category_id = c.category_id; -- Condición de unión: los 'category_id' en ambas tablas deben coincidir

Ejemplo 2: INNER JOIN:
SELECT * FROM film -- Selecciona todas las columnas de la tabla 'film'
INNER JOIN film_category ON film.film_id = film_category.film_id -- Realiza un INNER JOIN con la tabla 'film_category' basado en 'film_id'
JOIN category AS c ON film_category.category_id = c.category_id; -- Realiza un JOIN con la tabla 'category' usando el alias 'c' basado en 'category_id'

Ejemplo 3: INNER JOIN:
SELECT film.title, c.name -- Selecciona el título de la película y el nombre de la categoría
FROM film -- Utiliza la tabla 'film'

INNER JOIN film_category ON film.film_id = film_category.film_id -- Realiza un INNER JOIN con la tabla 'film_category' basado en 'film_id'

JOIN category AS c ON film_category.category_id = c.category_id; -- Realiza un JOIN con la tabla 'category' usando el alias 'c' basado en 'category_id'

Ejemplo 4: LEFT JOIN:

-- Encuentra el título de las películas y las veces que han sido alquiladas --

```
SELECT f.title, COUNT(r.rental_id) AS cantidad_rentas
FROM film AS f
LEFT JOIN inventory AS i ON f.film_id = i.film_id
LEFT JOIN rental AS r ON i.inventory_id=r.inventory_id
GROUP BY F.title;
```

Ejemplo 5: LEFT JOIN:

-- Muestra los nombres de los actores y los nombres de las películas en las que han participado --

-- (first_name, last_name, nombre_pelicula) --

```
SELECT a.first_name, a.last_name, f.title AS nombre_pelicula
FROM film AS f
LEFT JOIN film_actor AS fa ON f.film_id = fa.film_id
LEFT JOIN actor AS a ON fa.actor_id = a.actor_id
ORDER BY f.title;
```

-- OTRAS MANERAS (CONCATENACION) --

-- Concatenar el nombre y apellido de los actores y mostrar el título de la película

SELECT CONCAT(a.first_name, "", a.last_name) AS full_name, f.title -- Concatenar first_name y last_name, y renombrarlo como full_name

FROM actor AS a -- Utiliza la tabla 'actor' con alias 'a'

JOIN film_actor AS fa ON a.actor_id = fa.actor_id -- Realiza un JOIN con la tabla 'film_actor' usando 'actor_id'

JOIN film AS f ON fa.film_id = f.film_id; -- Realiza un JOIN con la tabla 'film' usando 'film_id'

-- OTRA MANERA USANDO (UNION, EXCEPT, DISTINCT) --

-- Seleccionar todos los nombres de los actores

SELECT first_name FROM actor;

-- Seleccionar todos los nombres de los clientes

SELECT first_name FROM customer;

-- Combinar ambas selecciones y eliminar duplicados

SELECT first_name FROM actor

UNION

SELECT first_name FROM customer;

-- Encuentra las películas que no han sido alquiladas

SELECT title FROM film

EXCEPT

SELECT f.title FROM film AS f -- Selecciona el título de las películas

JOIN inventory AS i ON f.film_id = i.film_id -- Realiza un JOIN con la tabla 'inventory' usando 'film_id'

JOIN rental AS r ON i.inventory_id = r.inventory_id; -- Realiza un JOIN con la tabla 'rental' usando 'inventory_id'

-- Selecciona las ciudades distintas asociadas con las direcciones del personal

SELECT DISTINCT city FROM city -- Selecciona las ciudades distintas de la tabla 'city'

JOIN address ON city.city_id = address.city_id -- Realiza un JOIN con la tabla 'address' usando 'city_id'

JOIN staff ON address.address_id = staff.address_id; -- Realiza un JOIN con la tabla 'staff' usando 'address_id'

-- (TAREA CON SUBCONSULTAS) --

```
SELECT city FROM city WHERE city_id IN (  
SELECT city_id FROM address WHERE address_id IN(  
SELECT address_id FROM staff));
```

NOTAS:

-- UNION & DISTINCT - MUESTRA SOLO UN NOMBRE

-- UNION ALL - MUESTRA TODOS LOS NOMBRES ESPECIFICOS

INTRODUCCION PARA LA CREACIÓN DE VISTAS

/* PARA CREAR UNA VISTA SE USA LO SIGUIENTE: */

CREATE VIEW prueba AS

SELECT first_name, last_name FROM actor

WHERE YEAR(last_update)>2020;

-- Muestra el primero y segundo nombre del actor cuando el año de actualización es mayor a 2020.--

SELECT * FROM prueba; /* Mostrar el contenido de nuestra VISTA "Prueba" */

INSERT actor VALUES(DEFAULT, "Raulito", "Ciriaco", NOW()); /* INSERTA LOS DATOS EN LA VISTA */

SELECT * FROM prueba;

V. Conclusiones:

Adentrarse en el tema de vistas, operadores SET, JOINS y subconsultas en MySQL puede parecer complicado al principio. La combinación de tablas, la comprensión de las uniones y la definición de variables puede resultar abrumadora. Sin embargo, con práctica, apoyo de compañeros y orientación de los profesores, es posible dominar estos conceptos. Este manual tiene como objetivo facilitarte el aprendizaje de estos elementos cruciales para el manejo de bases de datos.

Aunque al principio puede resultar complicado, con práctica y la ayuda adecuada, es posible dominar estos conceptos y aplicarlos para desarrollar bases de datos de manera eficiente y resolver problemas del mundo real. Continuar practicando y realizando ejercicios ayudará a fortalecer nuestro conocimiento en este ámbito de nuestra carrera.