

Sprint 3

Tasca S3.01. Manipulació de taules

En este sprint se simula una situación empresarial en la que deberás realizar diversas manipulaciones en las tablas de una base de datos. Además, trabajarás con índices y vistas para optimizar consultas y organizar la información.

Continuarás trabajando con la base de datos que contiene información de un marketplace, un entorno similar a Amazon donde varias empresas venden sus productos a través de un canal online. En esta actividad, empezarás a trabajar con datos relacionados con tarjetas de crédito.

Añade las tablas al modelo según corresponda:

Nivel 1: Tabla "credit_card"

Nivel 3: Tabla "user"

Nivel 1

Ejercicio 1:

Tu tarea es diseñar y crear una tabla llamada "credit_card" que almacene detalles cruciales sobre las tarjetas de crédito. La nueva tabla debe ser capaz de identificar de forma única cada tarjeta y establecer una relación adecuada con las otras dos tablas ("transaction" y "company"). Después de crear la tabla será necesario que ingreses la información del documento denominado "datos_introducir_credit". Recuerda mostrar el diagrama y realizar una breve descripción de este.

Respuesta: Los pasos que seguí para resolver este ejercicio fueron:

1. Crear la tabla "credit_card" con las columnas que indica el archivo "datos_introducir_sprint3_credit", estas columnas se pueden ver en la Figura 1.

```
INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2938', 'TR301950312213576817638  
INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2945', 'D0268547637485374752165  
INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2952', 'B6451VQL52710525608255'  
INSERT INTO credit card (id, iban, pan, pin, cvv, expiring date) VALUES ('CcU-2959', 'CR7242477244335841535'.
```

Figura 1 información del archivo "datos_introducir_sprint3_credit"

2. Una vez identificadas las columnas cree la tabla "credit_card" como se indica en la Figura 2.

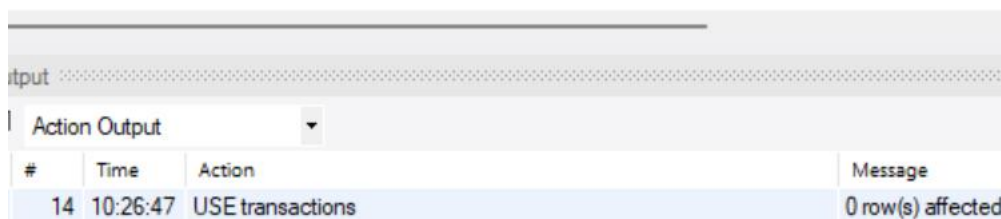
Tasca S3.01. Manipulació de taules

Telesforo Sol Campuzano

P2P: [Gabriel Pérez Santana](#)

Observación: Definí el tipo de dato de la columna *expiring_date* como VARCHAR(10) para que sea compatible con el formato de las fechas en el archivo *datos_introducir_sprint3_credit*.

```
1 • USE transactions;
2
3 -- Creamos la tabla credit_card
4 • CREATE TABLE IF NOT EXISTS credit_card (
5     id VARCHAR(20) PRIMARY KEY,
6     iban VARCHAR(34),
7     pan VARCHAR(16),
8     pin VARCHAR(4),
9     cvv VARCHAR(3),
10    expiring_date VARCHAR(10)
11 );
12
```



#	Time	Action	Message
14	10:26:47	USE transactions	0 row(s) affected

Figura 2 Creación de la tabla "credit_card"

3. Cargué los datos del archivo *expiring_date* en la tabla *credit_card*:

Observación: En la definición de la tabla *credit_card* la columna *pan* estaba como VARCHAR(16). Sin embargo, al cargar los datos se produjo un error, ya que uno de los valores superaba dicha longitud. Por ello, modifiqué la definición de la columna *pan* a VARCHAR(255), como se muestra en la Figura 3 (en la parte comentada del script). Esta modificación me permitió cargar los datos en la tabla *credit_card* (ver Figura 4).

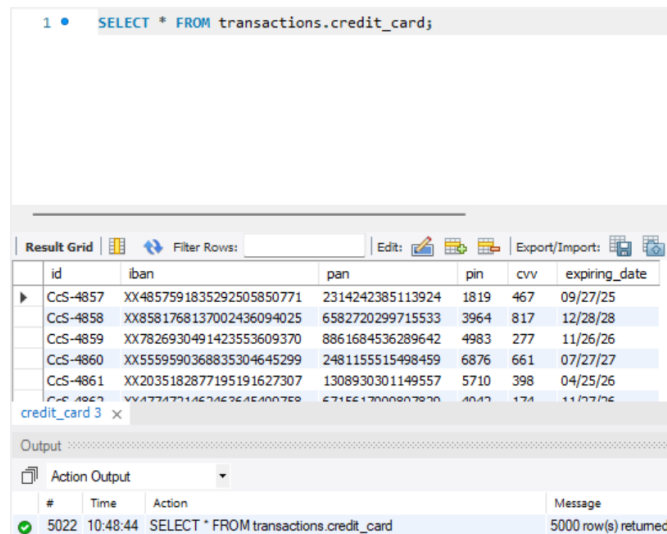
Tasca S3.01. Manipulació de taules
Telesforo Sol Campuzano
P2P: Gabriel Pérez Santana

```
-- Creamos la tabla credit_card
CREATE TABLE IF NOT EXISTS credit_card (
    id VARCHAR(20) PRIMARY KEY,
    iban VARCHAR(34),
    pan VARCHAR(225),
    pin VARCHAR(4),
    cvv VARCHAR(3),
    expiring_date VARCHAR(10)
);

ALTER TABLE credit_card
MODIFY COLUMN pan VARCHAR(255); Esta modificación la hice porque algunos datos son mayores
```

Figura 3 Modificación de la columna pan.

1 • `SELECT * FROM transactions.credit_card;`



	id	iban	pan	pin	cvv	expiring_date
▶	CcS-4857	XX4857591835292505850771	2314242385113924	1819	467	09/27/25
	CcS-4858	XX8581768137002436094025	6582720299715533	3964	817	12/28/28
	CcS-4859	XX7826930491423553609370	8861684536289642	4983	277	11/26/26
	CcS-4860	XX5559590368835304645299	2481155515498459	6876	661	07/27/27
	CcS-4861	XX2035182877195191627307	1308930301149557	5710	398	04/25/26
	CcS-4862	XX17371467467645400759	6715617000907970	4047	174	11/27/26

credit_card 3 x

Output

Action Output

#	Time	Action	Message
5022	10:48:44	SELECT * FROM transactions.credit_card	5000 row(s) returned

Figura 4 Datos de tabla credit_card.

4. Para establecer una relación adecuada entre la tabla credit_card con las otras dos tablas ("transaction" y "company"). Tuve en cuenta los siguientes puntos:

- La tabla transaction tiene información de la tabla company y de la tabla credit_card, con lo cual pensando en un modelo estrella, la tabla transaction será la tabla de hechos y las tablas company y credit_card serán tablas de dimensiones.
- La tabla transaction ya tiene una foreign key (company_id) referenciada a la tabla company(id).

Por los dos puntos anteriores, agregué una foreign key a la tabla de transaction, como se muestra en la Figura 5.

Observación: En la figura aparece un error porque hice la captura de pantalla después de haber creado la foreign key y al volver a correr la solicitud me apareció el error porque ya existía la foreign key.

Tasca S3.01. Manipulació de taules

Telesforo Sol Campuzano

P2P: Gabriel Pérez Santana

```
16 -- Relacionar la tabla credit_card con las tablas transaction y company.
17 -- Esta relación es adecuada porque la tabla transaction ya tiene un foreign key que la relaciona con la tabla company.
18 -- Ahora creo una foreign key que la relaciona con la tabla credit card.
19 ALTER TABLE transaction
20 ADD CONSTRAINT fk_credit_card_id_transaction
21 FOREIGN KEY (credit_card_id) REFERENCES credit_card(id);
```

Output

#	Time	Action	Message
3	13:18:10	ALTER TABLE transaction ADD CONSTRAINT fk_credit_card_id_tra...	Error Code: 1826. Duplicate foreign key constraint name 'fk_credit_ca

Figura 5 Creación de foreign key para relacionar la tabla transaction con la tabla credit_card.

5. Finalmente generé el diagrama (ver la Figura 6). Considerando un modelo en estrella, la tabla de hechos corresponde a la tabla *transaction*, mientras que las tablas *company* y *credit_card* funcionan como tablas de dimensiones. En la imagen se observa que las columnas de *transaction* están acompañadas por tres tipos de iconos. A partir de ello se deduce que la columna *id* constituye la primary key, mientras que *credit_card_id* y *company_id* son las foreign keys que relacionan la tabla de hechos con las tablas *credit_card* y *company*, respectivamente. Asimismo, se observa que múltiples transacciones pueden asociarse a una tarjeta de crédito y, del mismo modo, múltiples transacciones pueden estar vinculadas a una compañía.

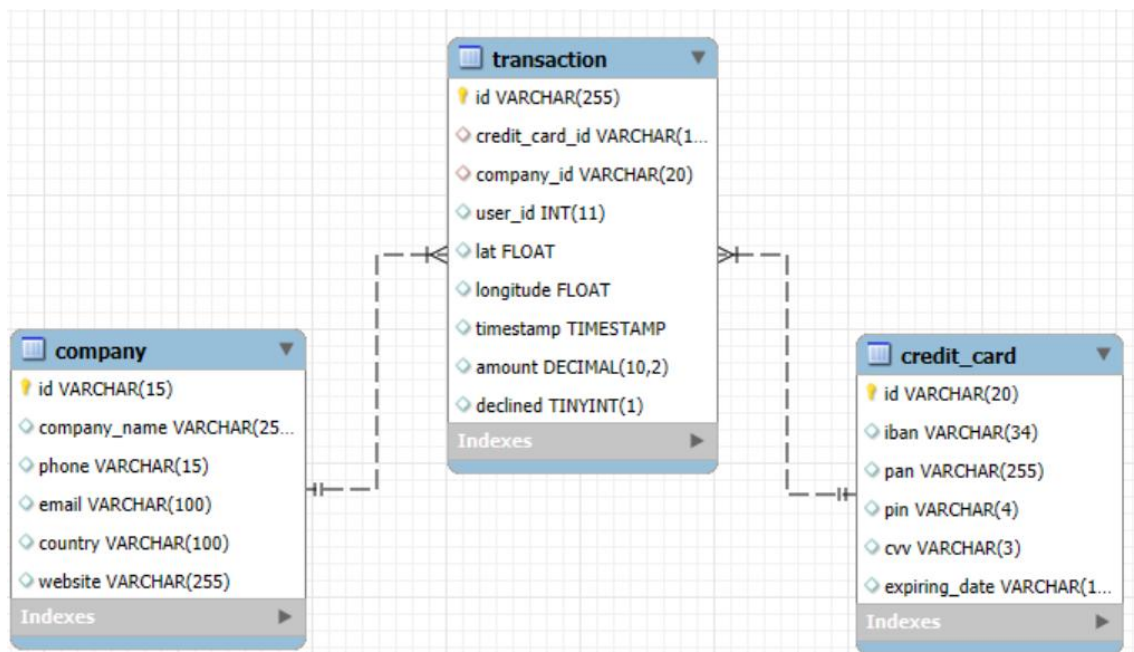


Figura 6 Diagrama

Ejercicio 2:

El departamento de Recursos Humanos ha identificado un error en el número de cuenta asociado a su tarjeta de crédito con ID CcU-2938. La información que debe mostrarse para este registro es: TR323456312213576817699999. Recuerda mostrar que el cambio se realizó.

Respuesta: La consulta para hacer la actualización se muestra en la Figura 7. En esta consulta utilice la función (UPDATE, SET y WHERE) para actualizar la tabla, también se puede observar que la modificación se realizó con éxito.

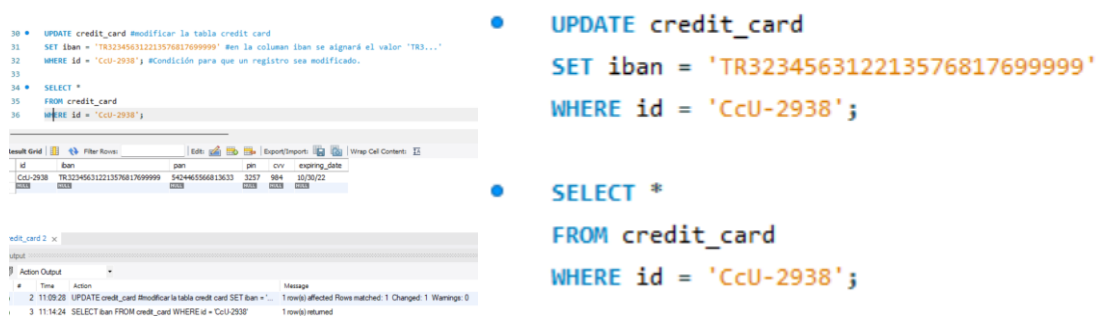


Figura 7 Actualización de registro en la tabla credit_card.

Ejercicio 3:

En la tabla "transaction" ingresa una nueva transacción con la siguiente información:

d	108B1D1D-5B23-A76C-55EF-C568E49A99DD
credit_card_id	CcU-9999
company_id	b-9999
user_id	9999
lat	829.999
longitude	-117.999
amount	111.11
declined	0

Respuesta: Para ingresar una nueva transacción en la tabla transaction, deben existir previamente las claves foráneas correspondientes en las tablas company y credit_card. Para ello, primero verifiqué que exista alguna compañía en la tabla company con id = 'b-9999'. Y también verifiqué si existe un registro en la tabla credit_card con id = 'CcU-9999'. Como se observa en la Figura 8, no existe ningún

Tasca S3.01. Manipulació de taules

Telesforo Sol Campuzano

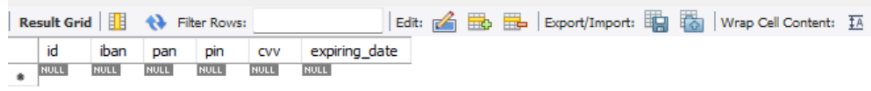
P2P: [Gabriel Pérez Santana](#)

registro en la tabla company con id = 'b-9999', ni en la tabla credit_card con id = 'CcU-9999'.

```

41  -- verificación de existencia de alguna compañía con id= 'b-9999'
42  ● SELECT *
43  FROM company
44  WHERE id = 'b-9999';
45  -- verificación de existencia de alguna tarjeta de credito con id= 'CcU-9999'
46  ● SELECT *
47  FROM credit_card
48  WHERE id = 'CcU-9999';

```



credit_card 5 x

Output

Action Output

#	Time	Action	Message
9	11:44:51	SELECT * FROM transactions.company	100 row(s) returned
10	11:45:33	SELECT * FROM company WHERE id = 'b-9999'	0 row(s) returned
11	11:52:23	SELECT * FROM credit_card WHERE id = 'CcU-9999'	0 row(s) returned

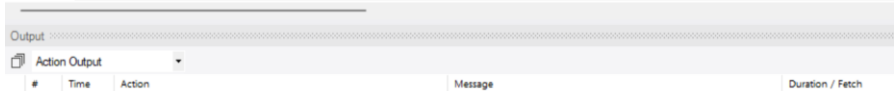
Figura 8 verificación de existencia de compañía con id='b-9999'.

Por lo anterior, primero se debía crear dichos registros en ambas tablas (company y credit_card) con los identificadores mencionados y para después crear la nueva transacción en la tabla transaction con los datos que se han dado. Esto se puede observar en la Figura 9.

```

42  ● SELECT *
43  FROM company
44  WHERE id = 'b-9999';
45  -- verificación de existencia de alguna tarjeta de credito con id= 'CcU-9999'
46  ● SELECT *
47  FROM credit_card
48  WHERE id = 'CcU-9999';
49  -- crear un registro en company con id='b-9999'
50  ● INSERT INTO company (id)
51  VALUES ('b-9999');
52  -- crear un registro en credit_card con id=
53  ● INSERT INTO credit_card (id)
54  VALUES ('CcU-9999');
55  -- crear registro en la tabla transaction
56  ● INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined) VALUES ('108810
57

```



Output

Action Output

#	Time	Action	Message	Duration / Fetch
16	12:16:48	INSERT INTO credit_card (id) VALUES ('CcU-9999')	1 row(s) affected	0.000 sec
17	12:21:10	INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined)	1 row(s) affected	0.000 sec
18	12:21:26	SELECT * FROM transactions.transaction	100001 row(s) returned	0.000 sec / 0.234 sec

Figura 9 Creación de registro en tabla transactions.

Ejercicio 4:

Desde recursos humanos te solicitan eliminar la columna "pan" de la tabla credit_card. Recuerda mostrar el cambio realizado.

Respuesta: La eliminación de la columna se puede observar en la Figura 10.

Tasca S3.01. Manipulació de taules
Telesforo Sol Campuzano
P2P: Gabriel Pérez Santana

```
58 | Ejercicio 4:
59 | Desde recursos humanos te solicitan eliminar la columna "pan" de la tabla credit_card.
60 | Recuerda mostrar el cambio realizado.
61 | */
62 |
63 | ALTER TABLE credit_card
64 | DROP COLUMN pan;
```

Output

Action Output

#	Time	Action	Message
✓ 1	13:45:56	SELECT * FROM transactions.credit_card	5001 row(s) returned
✓ 2	13:46:32	ALTER TABLE credit_card DROP COLUMN pan	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
✓ 3	13:46:41	SELECT * FROM transactions.credit_card	5001 row(s) returned

Figura 10 Eliminación de columna pan en la tabla credit_card.

Nivel 2

Ejercicio 1: Elimina de la tabla transaction el registro con ID 000447FE-B650-4DCF-85DE-C7ED0EE1CAAD de la base de datos.

Respuesta: La Figura 11 muestra la consulta utilizada para eliminar el registro.

```
66 | /* Nivel 2
67 | Ejercicio 1
68 | Elimina de la tabla transacción el registro con ID 000447FE-B650-4DCF-85DE-C7ED0EE1CAAD de la base de datos.
69 | */
70 | DELETE FROM transaction
71 | WHERE ID='000447FE-B650-4DCF-85DE-C7ED0EE1CAAD';
```

Output

Action Output

#	Time	Action	Message	Duration
✓ 1	15:38:27	SELECT * FROM transactions.transaction	100001 row(s) returned	0.015 sec
✓ 2	15:39:11	DELETE FROM transaction WHERE ID='000447FE-B650-4DCF-8...	1 row(s) affected	0.016 sec

Figura 11 Eliminación de registro en la tabla transaction.

Ejercicio 2

La sección de marketing desea tener acceso a información específica para realizar análisis y estrategias efectivas. Se ha solicitado crear una vista que proporcione detalles clave sobre las compañías y sus transacciones. Será necesaria que crees una vista llamada VistaMarketing que contenga la siguiente información: Nombre de la compañía. Teléfono de contacto. País de residencia. Media de compra realizado por cada compañía. Presenta la vista creada, ordenando los datos de mayor a menor promedio de compra.

Tasca S3.01. Manipulació de taules
Telesforo Sol Campuzano
P2P: [Gabriel Pérez Santana](#)

Respuesta: En la Figura 12 se muestra la consulta para crear la vista VistaMarketing con la siguiente información: nombre de la compañía, teléfono de contacto, país de residencia y media de compra realizado por cada compañía, en la vista los datos están ordenados de mayor a menor promedio de compra. En la Figura 13 se muestra la consulta para mostrar toda la información de la vista creada.

```
82 • CREATE VIEW VistaMarketing AS
83 SELECT c.company_name, c.phone, c.country, ROUND(AVG(t.amount),2) AS avg_amount
84 FROM company c
85 INNER JOIN transaction t ON c.id =t.company_id
86 GROUP BY c.id
87 ORDER BY avg_amount DESC;
```

Output

#	Time	Action	Message
12	16:45:52	DROP VIEW IF EXISTS VistaMarketing	0 row(s) affected, 1 warning(s): 1051 Unknown table 'transactions...' (
13	16:46:17	CREATE VIEW VistaMarketing AS SELECT c.company_name, c...	0 row(s) affected

Figura 12 Consulta para crear la vista VistaMarketing

Query 1 sprint3* transaction company vistamarketing x

```
1 • SELECT * FROM transactions.vistamarketing;
```

Result Grid

company_name	phone	country	avg_amount
Ac Fermentum Incorporated	06 85 56 52 33	Germany	284.87
Pretium Neque Corp.	07 77 48 55 28	Australia	276.16
Urna Convallis Associates	06 01 24 77 04	United States	274.24
At Associates	09 56 61 10 65	New Zealand	272.21
Metus Vitae Associates	08 25 44 40 66	Australia	270.08
Aliquam Dignam Limited	07 76 61 47 46	United States	269.60

Output

#	Time	Action	Message
12	16:45:52	DROP VIEW IF EXISTS VistaMarketing	0 row(s) affected, 1 warning(s): 1051 Unknown table 'transactions...' (
13	16:46:17	CREATE VIEW VistaMarketing AS SELECT c.company_name, c...	0 row(s) affected
14	16:46:27	SELECT * FROM transactions.vistamarketing	101 row(s) returned

Figura 13 Consulta para mostrar la vista

Ejercicio 3

Filtra la vista VistaMarketing para mostrar sólo las compañías que tienen su país de residencia en "Germany".

Tasca S3.01. Manipulació de taules
Telesforo Sol Campuzano
P2P: Gabriel Pérez Santana

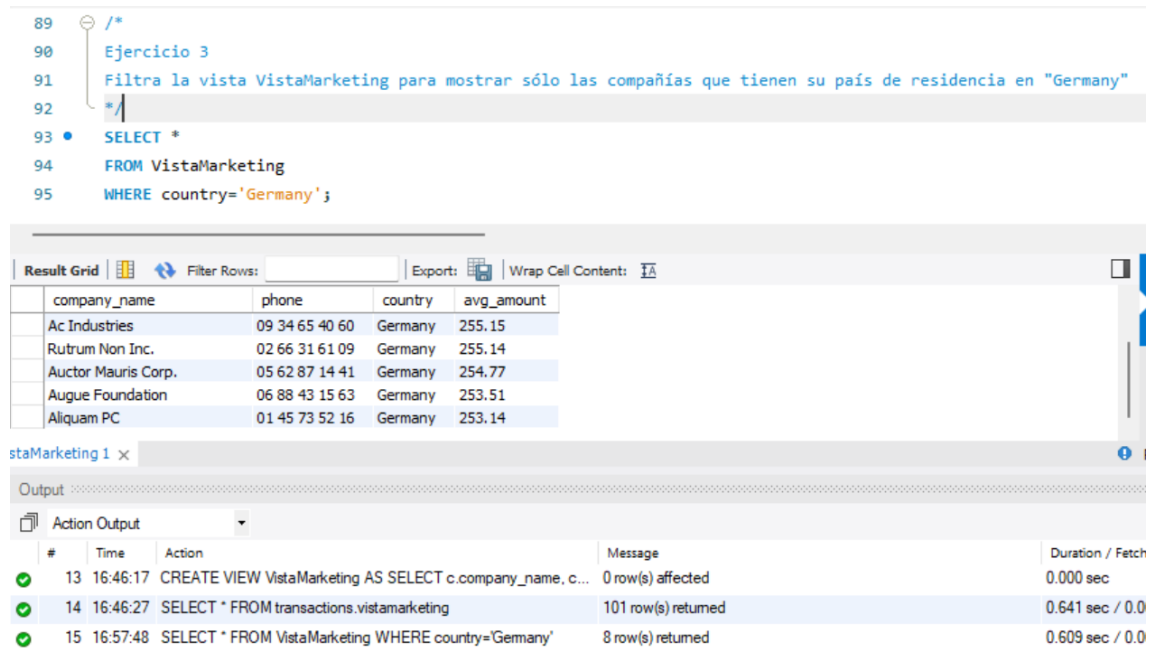
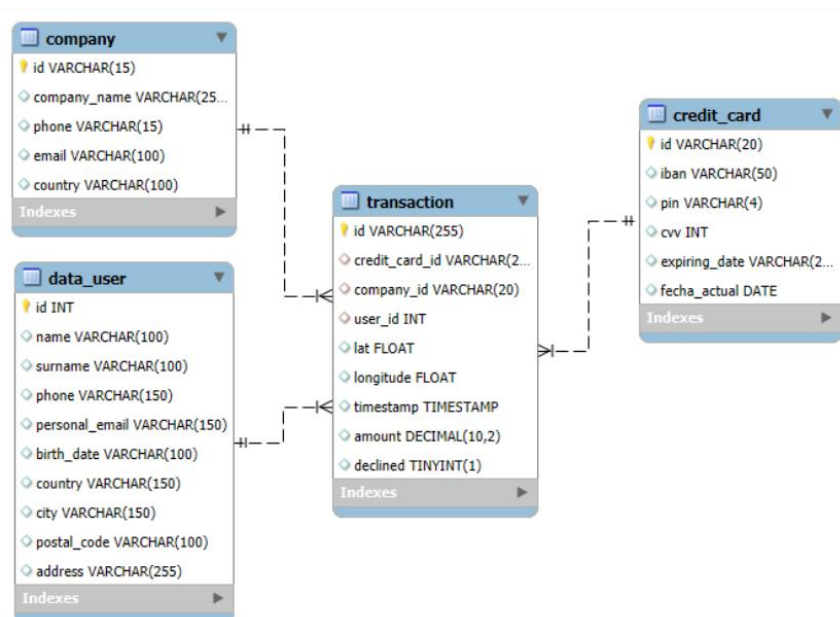


Figura 14 Filtro en vista.

Nivel 3

Ejercicio 1:

La próxima semana tendrás una nueva reunión con los gerentes de marketing. Un compañero de tu equipo realizó modificaciones en la base de datos, pero no recuerda cómo las realizó. Te pide que le ayudes a dejar los comandos ejecutados para obtener el siguiente diagrama:



Resposta: Para obtener el diagrama que se pide en el ejercicio, primero realicé los cambios en la tabla data_user, después en la tabla credit_card, y por último en la tabla company.

Así, los 10 cambios para obtener la tabla data_user son:

1. Crear la tabla “user” (usando el archivo “estructura datos user”) y cargar los datos de esa tabla (usando el archivo “datos introducir sprint3 user”). Esto se puede ver en la Figura 15 y en la Figura 16, respectivamente.

```
105 • CREATE TABLE IF NOT EXISTS user (  
106     id CHAR(10) PRIMARY KEY,  
107     name VARCHAR(100),  
108     surname VARCHAR(100),  
109     phone VARCHAR(150),  
110     email VARCHAR(150),  
111     birth_date VARCHAR(100),  
112     country VARCHAR(150),  
113     city VARCHAR(150),  
114     postal_code VARCHAR(100),  
115     address VARCHAR(255)  
116 );
```

Output :

Action Output

#	Time	Action	Message
✓ 2	09:41:06	SELECT * FROM transactions.credit_card	5001 row(s) returned
✓ 3	09:41:11	SELECT * FROM transactions.company	101 row(s) returned
✓ 4	09:45:14	CREATE TABLE IF NOT EXISTS user (id CHAR(10) PRIMARY ...	0 row(s) affected

Figura 15 Creación de la tabla user.

Tasca S3.01. Manipulació de taules

Telesforo Sol Campuzano

P2P: Gabriel Pérez Santana

Query 1 sprint3* transaction credit_card company datos introducir sprint3 user* user

1 • SELECT * FROM transactions.user;

id	name	surname	phone	email	birth_date	country	city	postal_code	address
1	Zeus	Gamble	1-282-581-0551	interdum.enim@protonmail.edu	Nov 17, 1985	United States	New York	10001	348-7818 St
10	Robert	Mccarthy	(324) 746-6771	fermentum@protonmail.com	Apr 30, 1984	United States	San Jose	95101	P.O. Box 77
100	Melodie	Mclean	1-677-221-7152	risus.varius@google.ca	Sep 15, 1989	United States	San Jose	95101	Ap #644-84
1000	Amigrv	Qbulnxbp	+48-258-9936	amigrv.qbulnxbp@example.com	May 17, 1970	Germany	Stuttgart	70173	215 Qbulnxb

#	Time	Action	Message	Duration /
5004	09:50:06	INSERT INTO user (id, name, surname, phone, email, birth_date, ...)	1 row(s) affected	0.000 sec
5005	09:50:06	INSERT INTO user (id, name, surname, phone, email, birth_date, ...)	1 row(s) affected	0.000 sec
5006	09:50:42	SELECT * FROM transactions.user	5000 row(s) returned	0.000 sec

Figura 16 Datos de la tabla user.

- Relacionar la tabla transaction con la tabla user. En la tabla transaction la columna user_id será la clave foránea y se relacionará con la columna id de la tabla user. En el primer intento para hacer la relación salió un error, esto porque las columnas user_id, de la tabla transaction, e id de la tabla user, no son del mismo tipo. Para saber de qué tipo es la columna user_id realicé la siguiente consulta “DESCRIBE transaction;” y en el grid se observa que el tipo de datos de la columna user_id son int(11) ver la Figura 17.

117 • DESCRIBE transaction;

118

Field	Type	Null	Key	Default	Extra
id	varchar(255)	NO	PRI	NULL	
credit_card_id	varchar(15)	YES	MUL	NULL	
company_id	varchar(20)	YES	MUL	NULL	
user_id	int(11)	YES		NULL	
lat	float	YES		NULL	
longitude	float	YES		NULL	

#	Time	Action	Message
5006	09:50:42	SELECT * FROM transactions.user	5000 row(s) returned
5007	10:06:07	ALTER TABLE transaction ADD CONSTRAINT fk_user_id_trans...	Error Code: 1215. Cannot add foreign key constraint
5008	10:12:49	DESCRIBE transaction	9 row(s) returned

Figura 17 Descripción de la tabla transaction

- Igualar el tipo de datos de las columnas id y user_id de las tablas user y transaction respectivamente. Al tratarse de identificadores, considero que

Tasca S3.01. Manipulació de taules
Telesforo Sol Campuzano
P2P: [Gabriel Pérez Santana](#)

deben ser tipo de texto. Por ello, modifiqué user_id para que tenga el mismo tipo que id, es decir, CHAR(10). Ver la Figura 18.

Observación: Este cambio me generó algunas dificultades, por los siguientes puntos: en el diagrama solicitado los datos de la columna user_id son del tipo int, además user_id es la primary key y user_id es la columna a la que se hace referencia en la foreign key que relaciona las tablas transaction y user_id. Comenté este punto en la revisión P2P y nos pareció interesante dejar así la solución del ejercicio para mostrar como resolví esa situación.

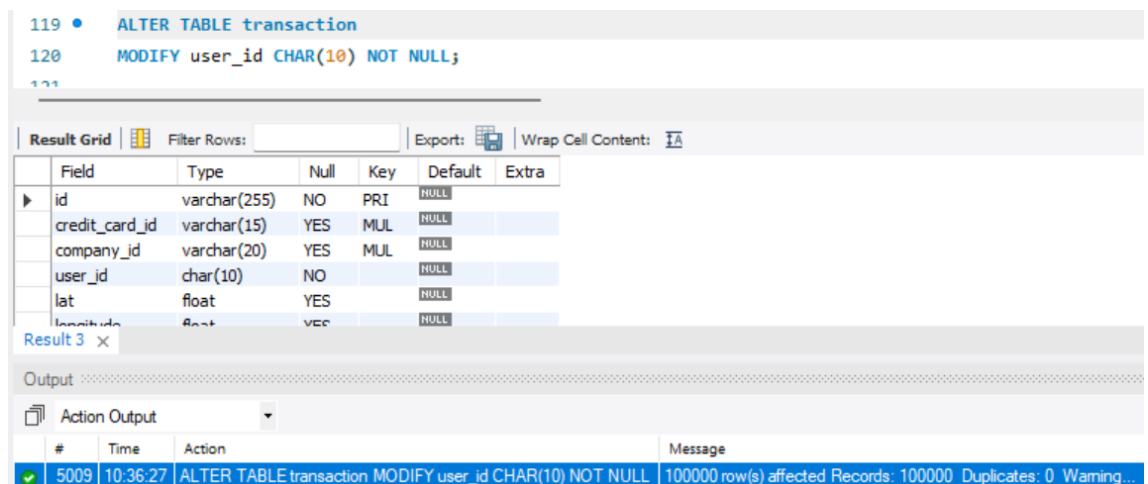


Figura 18 Cambio de tipos de datos de la columna user_id.

- Al intentar crear el foreign key para relacionar la tabla transaction con user, salió otro error. Este error se debe a que en la tabla transaction agregamos un nuevo registro, en el ejercicio 3 del nivel 1 del sprint3. Y este registro nuevo no está en la tabla user. Así que debemos agregar este registro en la tabla user antes de crear la foreign key. Esta agregación y la creación de la foreign key se pueden observar en la Figura 19.

Tasca S3.01. Manipulació de taules
Telesforo Sol Campuzano
P2P: [Gabriel Pérez Santana](#)

```

124 -- verificar existencia de id= '9999' en user
125 • SELECT *
126 FROM user
127 WHERE id ='9999';
128
129 -- crear un registro en user con id='9999'
130 • INSERT INTO user (id)
131 VALUES ('9999');
132
133 -- Relacionar la tabla transaction con la tabla user
134 • ALTER TABLE transaction
135 ADD CONSTRAINT fk_user_id_transaction
136 FOREIGN KEY (user_id) REFERENCES user(id);

```

#	Time	Action	Message
✓ 5018	10:57:19	SELECT * FROM user WHERE id ='9999'	0 row(s) returned
✓ 5019	11:01:09	INSERT INTO user (id) VALUES ('9999')	1 row(s) affected
✓ 5020	11:01:56	ALTER TABLE transaction ADD CONSTRAINT fk_user_id_trans...	100000 row(s) affected Records: 100000 Duplicates: 0 Warning...

Figura 19. Foreign key de la tabla transaction para relacionarla con la tabla user.

5. Ya está creada la relación, con esto el diagrama resultante es el que se observa en la Figura 20. Ahora se hacen los siguientes cambios:

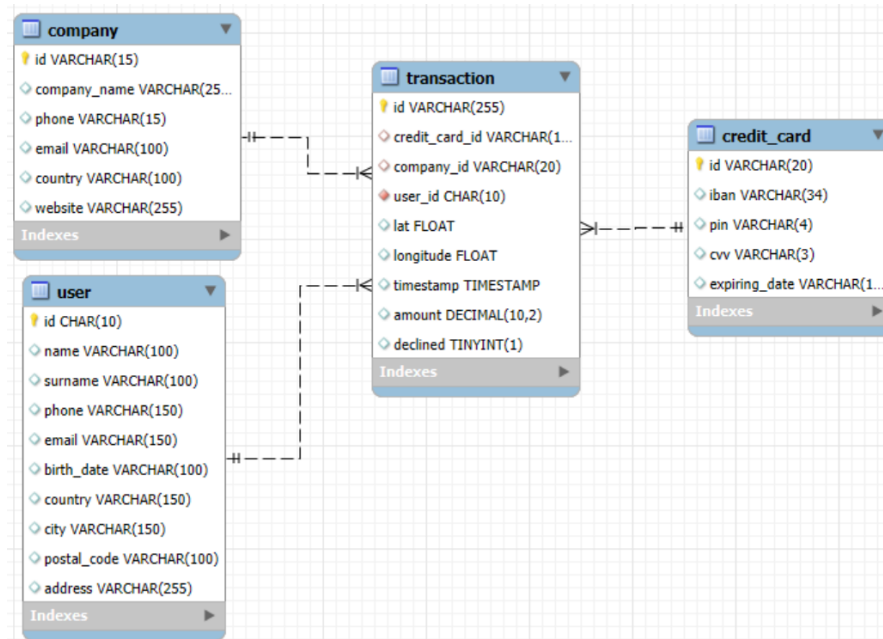


Figura 20 Diagrama inicial de la base de datos transactions.

- Eliminar la foreign key user_id de la tabla transactions para modificar la tabla user (ver Figura 21).
- Renombrar la tabla user a data_user (ver Figura 21).
- Crear una columna id2 de tipo entero (ver Figura 21). Los cambios de este punto hasta el punto 11 los hice porque no pude modificar directamente la

Tasca S3.01. Manipulació de taules
Telesforo Sol Campuzano
P2P: [Gabriel Pérez Santana](#)

columna id de tipo CHAR a tipo INT en la tabla data_user, por los puntos comentados en la observación del cambio 3.

```

139 • ALTER TABLE transaction
140     DROP FOREIGN KEY fk_user_id_transaction;
141     -- renombrar tabla
142 • RENAME TABLE user TO data_user;
143     -- creacion de nueva columna
144 • ALTER TABLE data_user
145     ADD id2 INT;
146
147     -- Desactivar el modo seguro

```

#	Time	Action	Message
5022	13:03:37	DROP INDEX 'fk_user_id_transaction' ON 'transactions'.transaction...	OK
5023	13:11:34	RENAME TABLE user TO data_user	0 row(s) affected
5024	13:38:45	ALTER TABLE data_user ADD id2 INT	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0

Figura 21

- Copiar y convertir los datos de id a id2, pero que sean enteros y sin espacios antes y al final de cada dato (esto lo hace TRIM). Para esto fue necesario desactivar el modo seguro, luego hacer el cambio y volver a activar el modo seguro (Ver Figura 22).

```

148 • SET SQL_SAFE_UPDATES = 0;
149     -- Copiar los valores de id a id2 convirtiéndolos a entero
150 • UPDATE data_user
151     SET id2 = CAST(TRIM(id) AS UNSIGNED)
152     WHERE id IS NOT NULL;
153     -- Volver a activar el modo seguro
154 • SET SQL_SAFE_UPDATES = 1;
155

```

#	Time	Action	Message
5045	14:41:02	SET SQL_SAFE_UPDATES = 0	0 row(s) affected
5046	14:41:09	UPDATE data_user SET id2 = CAST(TRIM(id) AS UNSIGNED) ...	5001 row(s) affected Rows matched: 5001 Changed: 5001
5047	14:41:18	SET SQL_SAFE_UPDATES = 1	0 row(s) affected
5048	14:41:29	SELECT * FROM transactions.data_user	5001 row(s) returned

Figura 22

Tasca S3.01. Manipulació de taules
Telesforo Sol Campuzano
P2P: [Gabriel Pérez Santana](#)

10. Renombrar la columna email como personal_email. En la tabla data_user, la clave primaria sigue siendo id de tipo *CHAR*, pero se requiere que la clave primaria sea la columna id2 de tipo *INT*. Para lograrlo, primero renombré id como user_id, luego renombré id2 como id, después eliminé la clave primaria de user_id y, finalmente, establecí la columna id como clave primaria (ver Figura 23).

```
154 • ALTER TABLE data_user
155     RENAME COLUMN email TO personal_email;
156     -- renombrar la columna id a user_id
157 • ALTER TABLE data_user
158     RENAME COLUMN id TO user_id;
159     -- renombrar la columna id2 a id
160 • ALTER TABLE data_user
161     RENAME COLUMN id2 TO id;
162     -- quitar primary key de user_id
163 • ALTER TABLE data_user
164     DROP PRIMARY KEY;
165     -- hacer primary key a nueva columna id
166 • ALTER TABLE data_user
167     ADD PRIMARY KEY (id);
168     -- relacionar la tabla transaction con la tabla user_id
169     -- cambiar
170
```

Output

#	Time	Action	Message
✓ 5055	16:01:06	ALTER TABLE data_user DROP PRIMARY KEY	5001 row(s) affected Records: 5001 Duplicates: 0 Warnings: 0
✓ 5056	16:01:51	ALTER TABLE data_user ADD PRIMARY KEY (id)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
✓ 5057	16:02:11	SELECT * FROM transactions.data_user	5001 row(s) returned

Figura 23

11. Por último, eliminé las columnas user_id y user_id_anterior de las tablas data_user y transaction respectivamente. Y cree la foreign key en transaction que relaciona la tabla transaction con la tabla data user mediante sus columnas user_id e id respectivamente (Ver la Figura 24).

```
217 • ALTER TABLE data_user
218     DROP COLUMN user_id;
219 • ALTER TABLE transaction
220     DROP COLUMN user_id_anterior;
221
222 • ALTER TABLE transaction
223     ADD CONSTRAINT fk_user_id_transaction
224     FOREIGN KEY (user_id) REFERENCES data_user(id);
```

Output

#	Time	Action	Message
✓ 16	19:51:41	ALTER TABLE data_user DROP COLUMN user_id	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
✓ 17	19:52:50	ALTER TABLE transaction DROP COLUMN user_id_anterior	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
✓ 18	19:56:51	ALTER TABLE transaction ADD CONSTRAINT fk_user_id_trans...	100000 row(s) affected Records: 100000 Duplicates: 0 Warning...

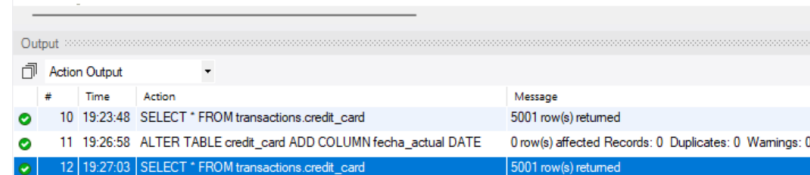
Figura 24

Tasca S3.01. Manipulació de taules
Telesforo Sol Campuzano
P2P: Gabriel Pérez Santana

Ahora enlisto los pasos para obtener la tabla credit_card solicitada.

1. Pasar la columna de iban VARCHAR (34) a iban VARCHAR (50) (Ver la Figura 25).
2. Pasar la columna cvv VARCHAR (3) a cvv INT (Ver la Figura 25).
3. Pasar la columna expiring_date VARCHAR(10) a expiring_date VARCHAR(20) (Ver la Figura 25).
4. Agregar la columna fecha_actual DATE (Ver la Figura 25).

```
194 -- Cambios para obtener la tabla credit_card
195 • ALTER TABLE credit_card
196   MODIFY iban VARCHAR(50) NULL;
197
198 • ALTER TABLE credit_card
199   MODIFY cvv INT NULL,
200   MODIFY expiring_date VARCHAR (20) NULL;
201
202 • ALTER TABLE credit_card
203   ADD COLUMN fecha_actual DATE;
204
```



The screenshot shows the output of SQL commands in a table with columns: #, Time, Action, and Message. It contains three rows of successful operations on the credit_card table.

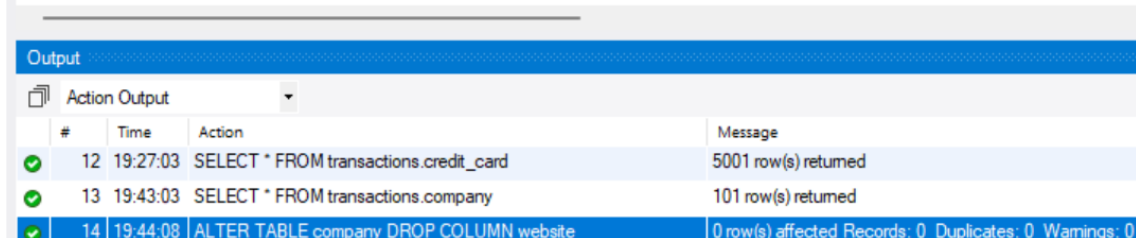
#	Time	Action	Message
✓ 10	19:23:48	SELECT * FROM transactions.credit_card	5001 row(s) returned
✓ 11	19:26:58	ALTER TABLE credit_card ADD COLUMN fecha_actual DATE	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
✓ 12	19:27:03	SELECT * FROM transactions.credit_card	5001 row(s) returned

Figura 25 Cambios en la tabla credit_card.

Ahora se muestra el paso para obtener la tabla company solicitada.

1. Eliminar la columna website de la tabla company (Ver la Figura 26).

```
214 • ALTER TABLE company
215   DROP COLUMN website;
```



The screenshot shows the output of SQL commands in a table with columns: #, Time, Action, and Message. It contains three rows of operations, including the successful dropping of the website column from the company table.

#	Time	Action	Message
✓ 12	19:27:03	SELECT * FROM transactions.credit_card	5001 row(s) returned
✓ 13	19:43:03	SELECT * FROM transactions.company	101 row(s) returned
✓ 14	19:44:08	ALTER TABLE company DROP COLUMN website	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0

Figura 26 Eliminación de columna website.

A continuación, muestro el diagrama obtenido (ver la Figura 27):

Tasca S3.01. Manipulació de taules
Telesforo Sol Campuzano
P2P: Gabriel Pérez Santana

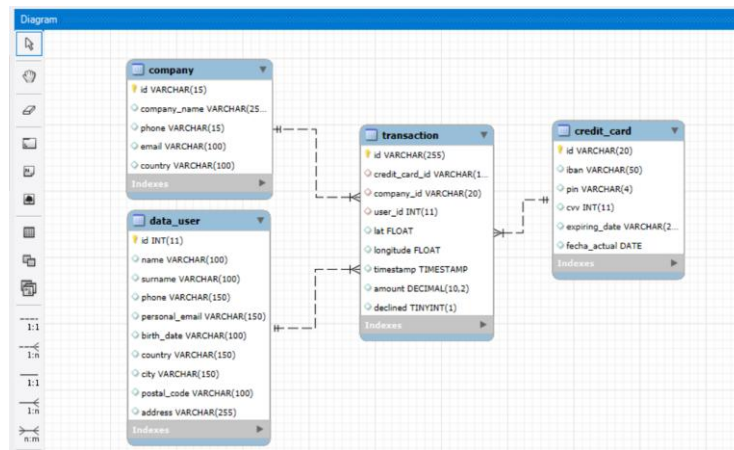


Figura 27 Diagrama solicitado.

Ejercicio 2:

La empresa también le pide crear una vista llamada "InformeTecnico" que contenga la siguiente información:

- ID de la transacción
- Nombre del usuario/a
- Apellido del usuario/a
- IBAN de la tarjeta de crédito usada.
- Nombre de la compañía de la transacción realizada.

Asegúrese de incluir información relevante de las tablas que conocerá y utilice alias para cambiar de nombre columnas según sea necesario.

Muestra los resultados de la vista, ordena los resultados de forma descendente en función de la variable ID de transacción.

Respuesta: En la Figura 28 se observa la consulta para generar la vista solicitada.

```

240 • CREATE VIEW InformeTecnico AS
241 SELECT t.id AS id_transaccion, d.name AS nombre_usuario, d.surname AS apellido, cc.iban AS IBAN, c.company_name AS
242 FROM transaction t
243 INNER JOIN data_user d ON t.user_id = d.id
244 INNER JOIN credit_card cc ON t.credit_card_id = cc.id
245 INNER JOIN company c ON t.company_id = c.id
246 ORDER BY id_transaccion DESC;

```

Output

#	Time	Action	Message	Duration / Fetch
2	13:35:54	CREATE VIEW InformeTecnico AS SELECT t.id AS id_transacci...	0 row(s) affected	0.016 sec.

Figura 28 creación de la vista informetecnico.

En la Figura 29 se muestran los resultados de la vista de forma descendente en función de la variable ID de transacción.

Tasca S3.01. Manipulació de taules
Telesforo Sol Campuzano
P2P: Gabriel Pérez Santana

1 • `SELECT * FROM transactions.infometecnico;`

id_transaccion	nombre_usuario	apellido	IBAN	compania
FFFD31D6-9495-47CE-B54A-7DB8E1CC274B	Bmrgli	Tprvvmrc	XX794814451211289182490922	Turpis Company
FFFCF76D-ECF0-4985-A2D0-B2A7B75998FC	Dfrled	Vlqcdl	XX636251701647892036676034	Amet Nulla Donec Corporation
FFFC9E8D-27C7-4ADE-98F2-7533EF4DF126	Securp	Faofvqfy	XX162677143304223631437567	Nunc Interdum Incorporated
FFFB270D-F53A-4D5D-9666-E5307C53CC84	Ggzjpa	Uirzjulh	XX395114267082019952567052	Viverra Donec Foundation
FFF9E3CE-234E-408C-A8EF-F9CAD577224A	Yshimq	Zpsjsleed	XX8845462156537570367941	Convallis In Incorporated
FFF9E178-6CD2-4DF9-99B0-49AE068809B1	Jevepx	Xwcwzwnm	XX321405515711654384711481	Mus Aenean Eget Foundation
FFF867C9-17B5-4B1F-AFD9-F8023AAA449E	Fqlngd	Lvhfqyxi	XX278446342932680979729426	Cras Vehicula Aliquet Industries

informetecnico 2 x

Output

#	Time	Action	Message
4	13:36:21	SELECT * FROM transactions.infometecnico	100000 row(s) returned

Figura 29 Resultados de la vista en orden descendente en función de la variable id_transacción.