SPRINT S3.01 – MANIPULACIÓN DE TABLAS

NIVEL 1

Ejercicio 1.1

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 ingresar la información del documento denominado "datos_introducir_credit". Recordar mostrar el diagrama y realizar una breve descripción del mismo.

Creando la tabla **"credit_card"** siguiendo las indicaciones y estableciendo el campo **"id"** como primary key además de definir la relación con la tabla **transaction** a través de la **foreign key** desde la creación.

*__Actualización:__ Se ha definido como **primary key** a **id** desde su línea de código. Se ha modificado el tipo de variable de los campos **PIN** y **CVV** a **CHAR(4)** y **CHAR(3)**, respectivamente. Anteriormente los habiamos definido como INT.

```
 9    USE transactions;
10
11    CREATE TABLE credit_card (
12        id VARCHAR(15) primary key,
13        iban VARCHAR(50),
14        pan VARCHAR(50),
15        pin CHAR(4),
16        cvv CHAR(3),
17        expiring_date VARCHAR(10)
18        );
```

Averiguamos el nombre de la **foreign key** en **transaction** para eliminarla ya que luego necesitaremos crear las relaciones a través de las **foreign keys**.

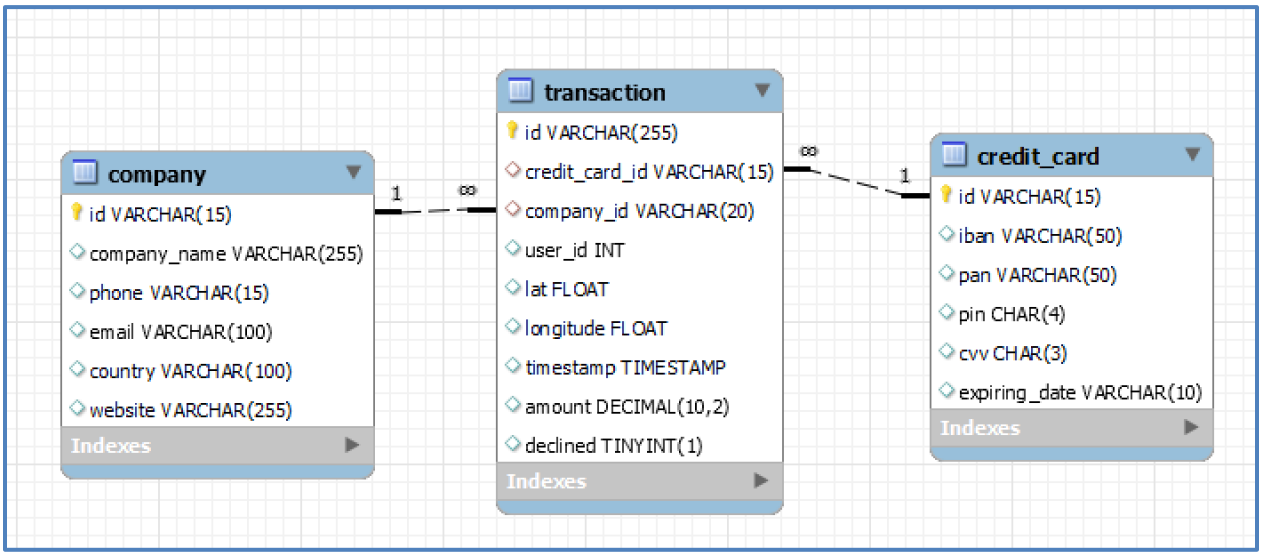
```
24
25 •    SELECT constraint_name
26      FROM information_schema.key_column_usage
27      WHERE table_name = 'transaction'
28      AND constraint_schema = 'transactions';
29
30 •    ALTER table transaction
31      DROP foreign key transaction_ibfk_1;
```

Creamos las **`Foreign Keys`** en **`transaction`** que referencian a las **`primary keys`** de las tablas **`credit_card`** y **`company`**:

```
34
35 •    SET FOREIGN_KEY_CHECKS = 0;
36 •    ALTER TABLE transaction
37      ADD FOREIGN KEY(credit_card_id) REFERENCES credit_card(id);
38 •    ALTER TABLE transaction
39      ADD FOREIGN KEY(company_id) REFERENCES company(id);
40 •    SET FOREIGN_KEY_CHECKS = 1;
```

Una vez cargados los datos en nuestra nueva tabla tenemos ya las 3 tablas relacionadas, como mostramos a continuación:

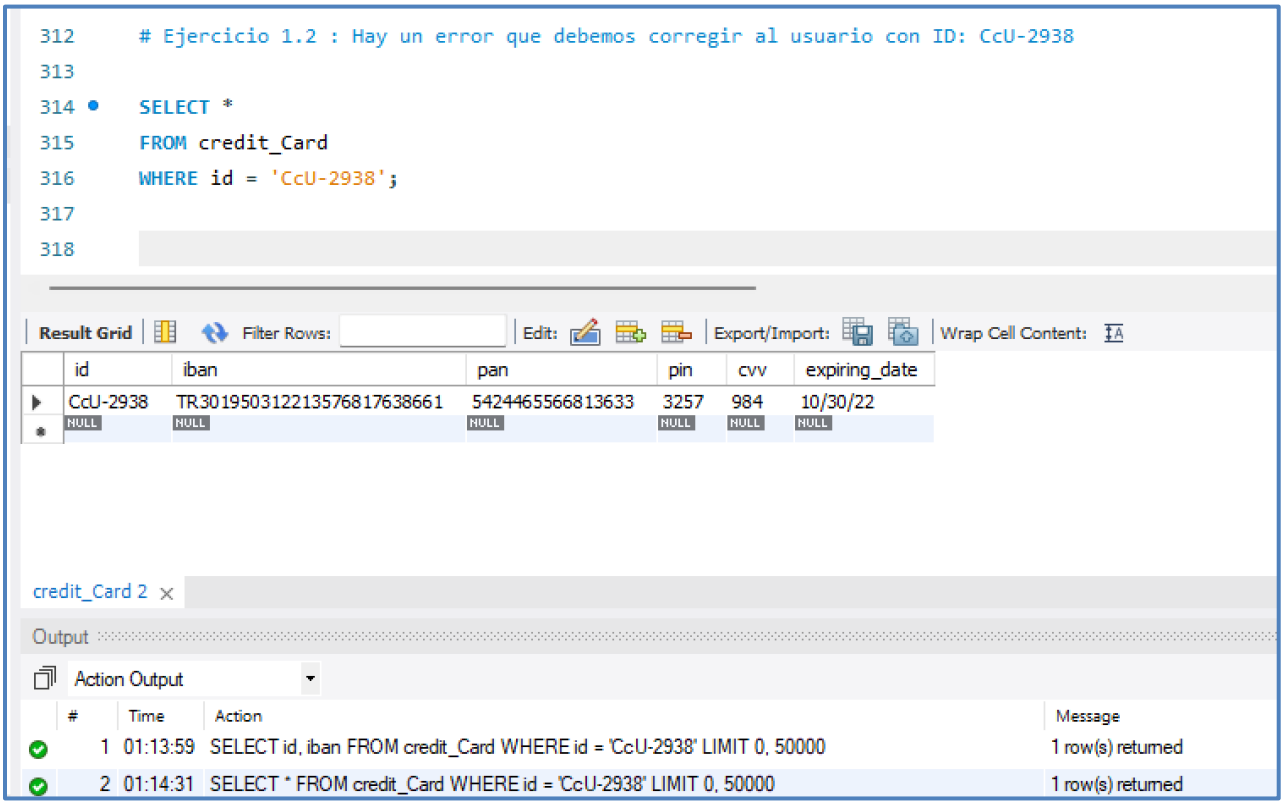
***Actualización:** Ahora la cardinalidad entre las tablas es la correcta, como se observa en la imagen. Siendo **transaction** la *tabla de hechos* y **company** y **credit_card** las *tablas de dimensiones*.



Ejercicio 1.2

El departamento de Recursos Humanos ha identificado un error en el número de cuenta del usuario con ID CcU-2938. La información que debe mostrarse para este registro es: R323456312213576817699999. Recordar mostrar que el cambio se realizó.

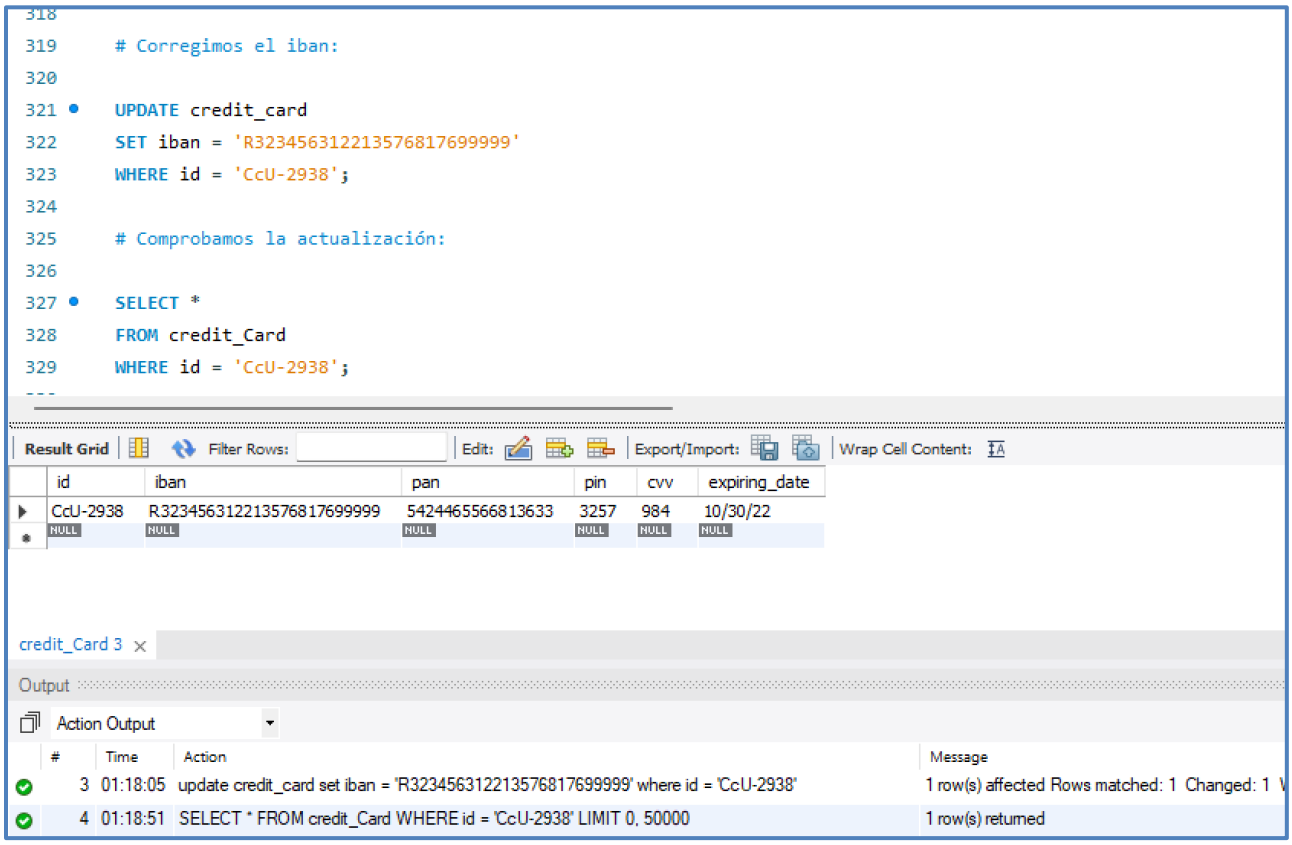
Debemos corregir el Iban:



El iban por defecto sería: **TR30195031221357638661**

Se corrige el `iban` anterior por el nuevo `iban`: **"R323456312213576817699999"** :

A continuación se comprueba la correción:



Ejercicio 1.3

En la tabla "transaction" ingresar un nuevo usuario con la siguiente información:

Id	108B1D1D-5B23-A76C-55EF-C568E49A99DD
credit_card_id	CcU-9999
company_id	b-9999
user_id	9999
lato	829.999
longitud	-117.999
amunt	111.11
declined	0

***Actualización:** Ya no se utilizará la vía de desactivar las `Foreign Key` para realizar modificaciones. En su lugar; y para mantener la integridad referencial; ingresaremos los datos en las correspondientes tablas siguiendo el siguiente orden: **credit_card** y **company** . Luego de ello recién se procede a ingresar los datos

de la tabla **transaction**. Para las tablas **credit_card** y **company** basta con ingresar sus respectivas `Primary Keys`; para la tabla **transaction** se ingresan los datos de todos los campos a excepción de timestamp, pues no se proporciona y no es obligatorio, por lo que quedaría reflejado como `NULL`.

```
361
362 •  INSERT INTO company (id) VALUES ('b-9999');
363
364 •  INSERT INTO credit_card (id) VALUES ('CcU-9999');
365
366 •  INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined) VALUES
367        ('108B1D1D-5B23-A76C-55EF-C568E49A99DD', 'CcU-9999', 'b-9999', '9999', '829.999', '-117.999', '111.11', '0')
368        ;
369
```

Se verifica que se ingresó correctamente el nuevo registro:

```
371    # Se verifica que se creó correctamente el nuevo registro:
372
373 •  select * from transaction
374    where id = '108B1D1D-5B23-A76C-55EF-C568E49A99DD';
375
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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

transaction 17

Output

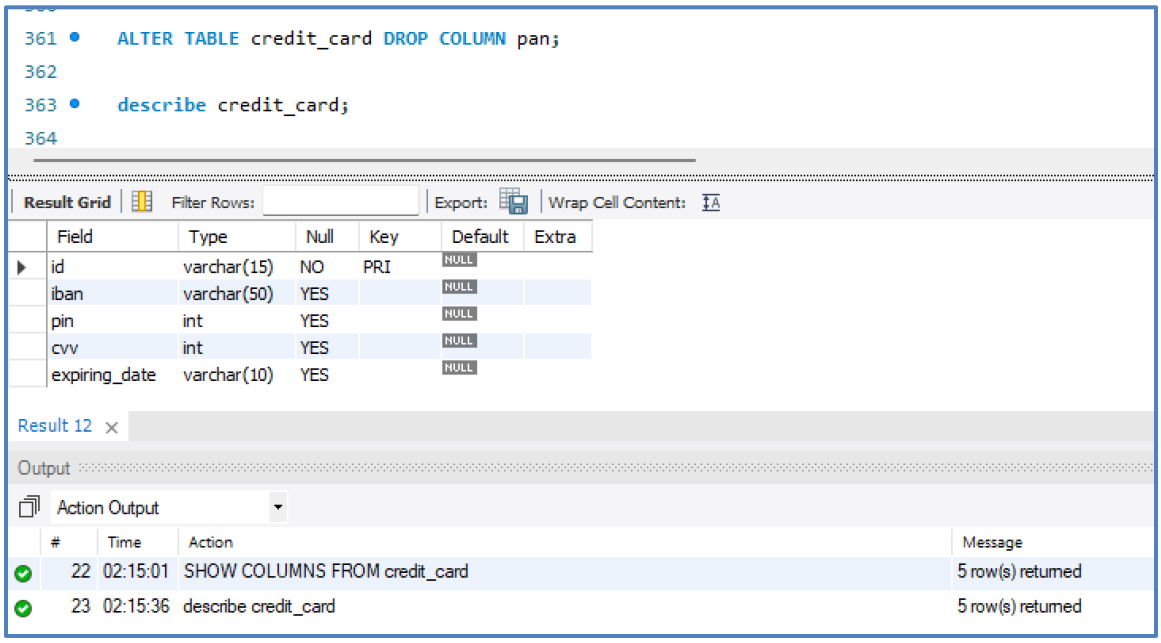
Action Output

#	Time	Action	Message
22	03:39:52	select * from transaction where id = '108B1D1D-5B23-A76C-55EF-C568E49A99DD' LIMIT 0, 500...	1 row(s) returned

Ejercicio 1.4

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

Borramos la columna "`pan`" y mostramos como esta ya no aparece en la descripción de la tabla `credit_card`.

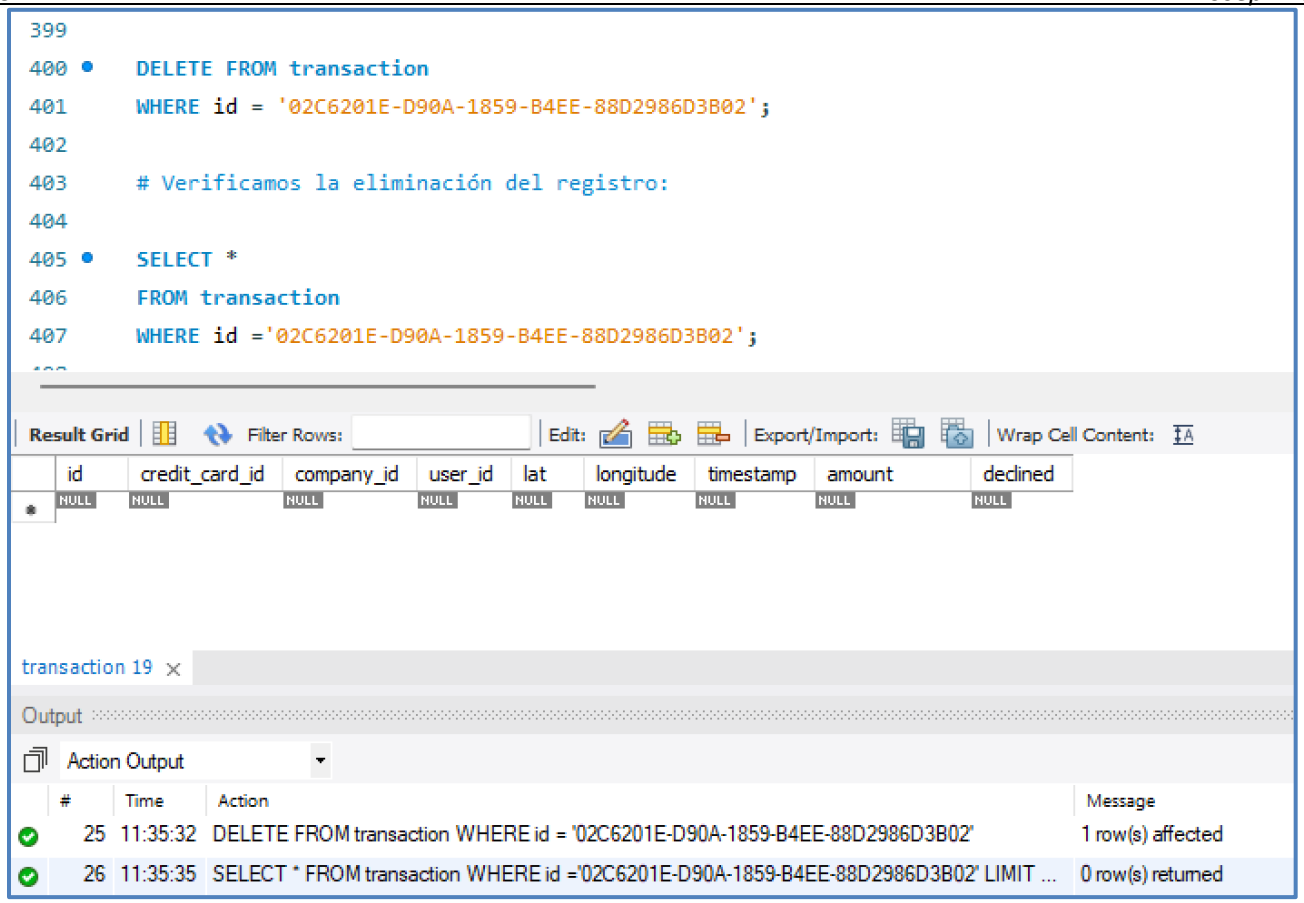


NIVEL 2

Ejercicio 2.1

Eliminar de la tabla "transaction" el registro con ID "02C6201E-D90A-1859-B4EE-88D2986D3B02" de la base de datos.

***Actualización:** Ya no se realiza la eliminación del registro a través de la *desactivación de la clave foránea*. Ahora que hemos establecido correctamente la relación y cardinalidad entre las tablas de hechos y dimensiones, la eliminación del registro en **transaction** se ejecuta directamente. Paso seguido, verificamos la supresión del registro.



Ejercicio 2.2

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 necesario crear 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. Presentar la vista creada, ordenando los datos de mayor a menor promedio de compra.

La query solicitada entraría dentro de la categoría de **vista compleja** dado que utilizariamos sentencias como `join`, `group by`, etc. Al mismo tiempo, considerando solo las compras (`declined = 0`) efectuadas y redondeando las cantidades, obtenemos la siguiente vista.

*** Actualización:** Se utilizará la sentencia `ORDER BY` de manera externa (es decir, al consultar la **vista**) y no dentro de la creación de la **vista**, como se había realizado anteriormente.

```
413 •   CREATE VIEW VistaMarketing AS
414     SELECT company_name, phone, country, ROUND(AVG(amount), 2) AS compra_media
415     FROM company
416     JOIN transaction ON company.id = transaction.company_id
417     WHERE declined = 0
418     GROUP BY company_name, phone, country;
419
420 •   SELECT *
421     FROM VistaMarketing
422     ORDER BY compra_media DESC;
423
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

company_name	phone	country	compra_media
Eget Ipsum Ltd	03 67 44 56 72	United States	481.86
Sed Id Limited	07 28 18 18 13	United States	477.51
Neque Tellus Incorporated	04 43 18 34 19	Ireland	477.10
Nunc Sit Incorporated	07 28 42 63 63	Norway	461.83
Non Magna LLC	06 71 73 13 17	United Kingdom	458.74
Maecenas Malesuada Fringilla Inc.	09 38 53 76 61	Netherlands	451.29

VistaMarketing 23

Output

Action Output

#	Time	Action	Message
33	13:57:01	CREATE VIEW VistaMarketing AS SELECT company_name, phone, country, ROUND(AVG(amo...	0 row(s) affected
34	13:57:04	SELECT * FROM VistaMarketing ORDER BY compra_media DESC LIMIT 0, 50000	101 row(s) returned

Ejercicio 2.3

Filtrar la vista "VistaMarketing" para mostrar solo las compañías que tienen su país de residencia en Alemania.

Aplicaremos el filtro mediante el uso de la clausula **`WHERE`** a la vista ya creada. En la imagen apreciamos el script para invocar a la vista y su resultado.

***Actualización:** Decidimos agregar la sentencia para dar orden al resultado de mayor a menor compra media.

```
422 •   SELECT *
423     FROM VistaMarketing
424     WHERE country ='Germany'
425     ORDER BY compra_media DESC;
426
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

company_name	phone	country	compra_media
Ac Industries	09 34 65 40 60	Germany	396.15
Auctor Mauris Corp.	05 62 87 14 41	Germany	308.99
Ac Fermentum Incorporated	06 85 56 52 33	Germany	293.57
Aliquam PC	01 45 73 52 16	Germany	280.34
Rutrum Non Inc.	02 66 31 61 09	Germany	266.90
Nunc Interdum Incorporated	05 18 15 48 13	Germany	242.95

VistaMarketing 25

Output

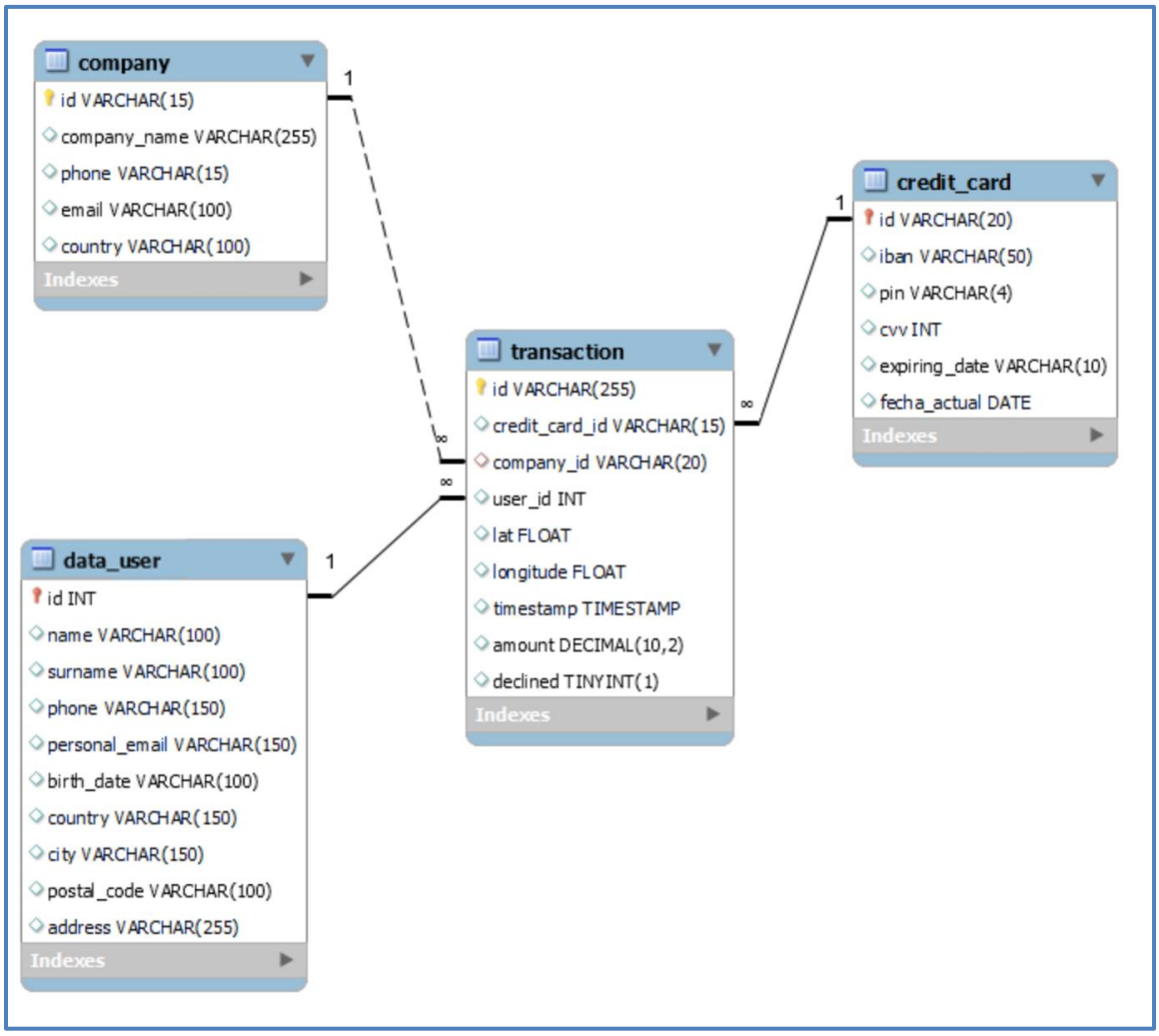
Action Output

#	Time	Action	Message
35	14:00:00	SELECT * FROM VistaMarketing WHERE country ='Germany' LIMIT 0, 50000	8 row(s) returned

NIVEL 3

Ejercicio 3.1

La próxima semana se tendrá una nueva reunión con los gerentes de marketing. Un compañero del equipo realizó modificaciones en la base de datos, pero no recuerda cómo las realizó. Pide ayudarle a dejar los comandos ejecutados para obtener el siguiente diagrama:



Para esta actividad es necesario describir el "paso a paso" de las tareas realizadas. Es importante realizar descripciones sencillas, simples y fáciles de comprender. Para realizar esta actividad se debe trabajar con los archivos denominados "estructura_datos_user" y "datos_introducir_user".

Observando el diagrama de relación de tablas podemos notar que hay una serie de comandos que se han ejecutado para llegar a tal situación y los describiremos en los pasos siguientes.

 # creación la tabla "user" y subida de datos

Paso 1: Crear la tabla **data_user** con los comandos mostrados en la imagen siguiente:

```
405     #Paso 1: Se ha creado la tabla "user" siguiendo la instrucción del script "estructura_datos_user":
406
407 •   create index idx_user_id on transaction(user_id);
408
409 •   create table if not exists user (
410         id INT primary key,
411         name varchar(100),
412         surname varchar(100),
413         phone varchar(150),
414         email varchar(150),
415         birth_date varchar(100),
416         country varchar(150),
417         city varchar(150),
418         postal_code varchar(100),
419         address varchar(255),
420         foreign key(id) references transaction(user_id)
421         );
```

Paso 2: Se carga toda la data del archivo **dades_introduir_user**.

 # Alteraciones en la tabla "company":

Paso 3: Se elimina el campo `website` de la tabla `company`

```
712 •    alter table company
713       drop website;
714
```

 # Alteraciones en la tabla "credit_card":

Paso 4: Se modifica la longitud del tipo de dato del campo id de la tabla `credit_card`, de `varchar(15)` a `varchar(20)`

```
717 •   alter table credit_card
718     modify column id varchar(20);
```

Paso 5: Se modifica el tipo de dato del campo `pin` de la tabla `credit_card`, de `int` a `varchar(4)`

```
722 •   alter table credit_card
723     modify pin varchar(4);
```

Paso 6: Se crea el campo `fecha_actual` siendo del tipo `DATE`

```
727 •   alter table credit_card
728     add column fecha_actual date;
```

 # Alteraciones en la tabla "user":

Paso 7: En la tabla `user`, se modifica el nombre de la tabla a `data_user:`

```
732 •   rename table user to data_user;
```

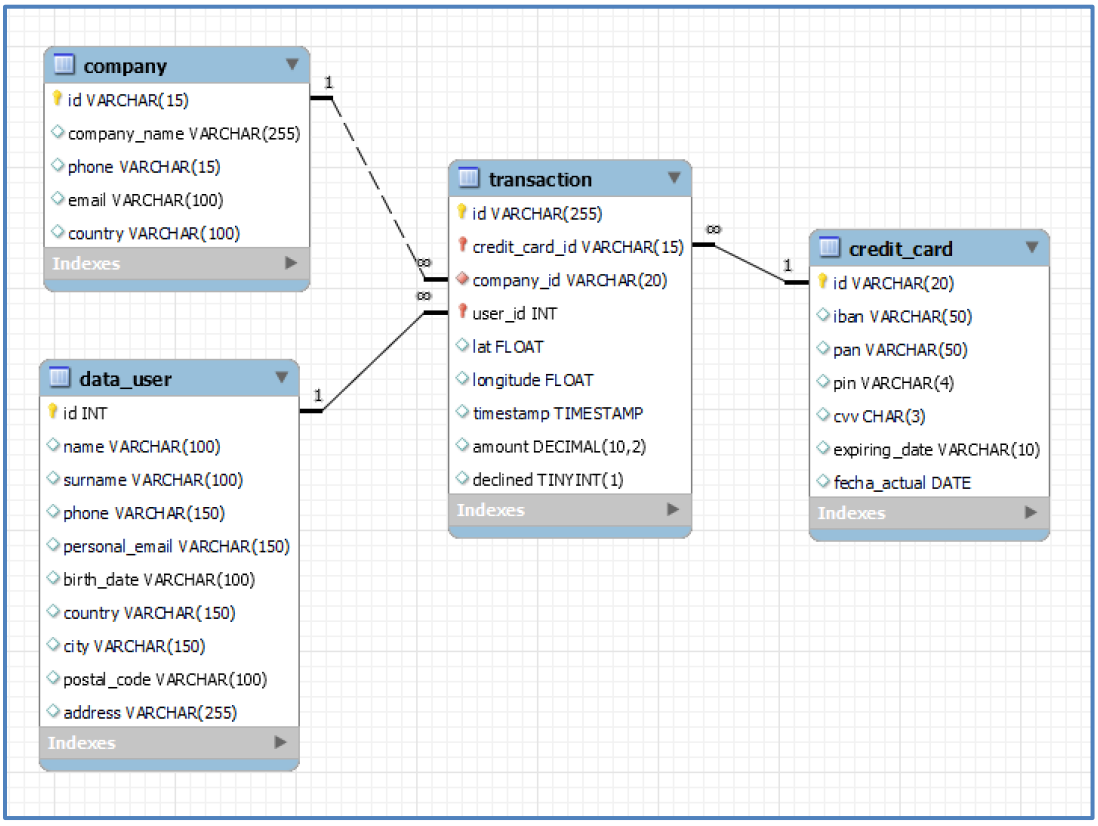
Paso 8: En tabla `data_user`, se modifica el nombre del campo `email` a `personal_email`:

```
736 •   alter table data_user
737     change column email personal_email varchar(150);
```

Como consecuencia de aplicar todas los pasos obtenemos la misma tabla como la mostrada en el enunciado; con lo que se le podrá orientar al compañero del equipo los comandos que tuvo que ejecutar paso a paso, tanto para volver a ejecutarlos como para poder revertirlos, de ser necesario.

Diagrama final, luego de ejecutar todos los pasos:

***Actualización:** Para el diagrama se ha utilizado la notación *connect to columns* en lugar de *crow's foot*.



Ejercicio 3.2

La empresa también solicita 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
- Asegurarse de incluir información relevante de ambas tablas y utilizar alias para cambiar de nombre columnas según sea necesario.

Mostrar los resultados de la vista, ordenar los resultados de forma descendente en función de la variable ID de transacción.

Hemos creado una vista o **`view`** según los datos solicitados, adicionándole el monto de la transacción. El script ejecutado y su respectivo resultado se muestran a continuación:

* **Actualización:** Se utilizará la clausula ORDER BY cuando se consulte la vista y no en su definición.

```
786 •   CREATE OR REPLACE VIEW InformeTecnico AS
787     SELECT transaction.id as id_transaccion, data_user.name AS nombre_usuario, data_user.surname AS apellido_usuario,
788     credit_card.iban AS iban_tarjeta_credito, transaction.amount AS monto_transaccion, company.company_name AS nombre_compania
789     FROM transaction
790     JOIN company ON company.id = transaction.company_id
791     JOIN credit_card on credit_card.id = transaction.credit_card_id
792     JOIN data_user on data_user.id = transaction.user_id;
793
794 •   SELECT * FROM InformeTecnico
795     ORDER BY id_transaccion DESC;
796
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id_transaccion	nombre_usuario	apellido_usuario	iban_tarjeta_credito	monto_transaccion	nombre_compania
FE96CE47-BD59-381C-4E18-E3CA3D44E8FF	Kenyon	Hartman	DO26854763748537475216568689	480.13	Magna A Neque Industries
FE809ED4-2DB6-55AC-C915-929516E4646B	Molly	Gilliam	SE2813123487163628531121	219.83	Nunc Interdum Incorporated
FD9CBCCD-8E1E-8DA1-4606-7E3A6F3A5A65	Linus	Willis	KW9485332754781757886242955643	42.32	Nunc Interdum Incorporated
FD89D51B-AE8D-77DC-E450-B8083FBD3187	Hilda	Levy	LT053237077744561475	200.72	Malesuada PC
FD2E8957-414B-BEEC-E9AD-59AA7A8A6290	Hedwig	Gilbert	GE84848451582810541526	78.29	Neque Tellus Imperdiet Corp.

InformeTecnico 28

Output

Action Output

#	Time	Action	Message
332	14:44:18	CREATE OR REPLACE VIEW InformeTecnico AS SELECT transaction.id as id_transaccion, dat...	0 row(s) affected
333	14:44:20	SELECT * FROM InformeTecnico ORDER BY id_transaccion DESC LIMIT 0, 50000	586 row(s) returned