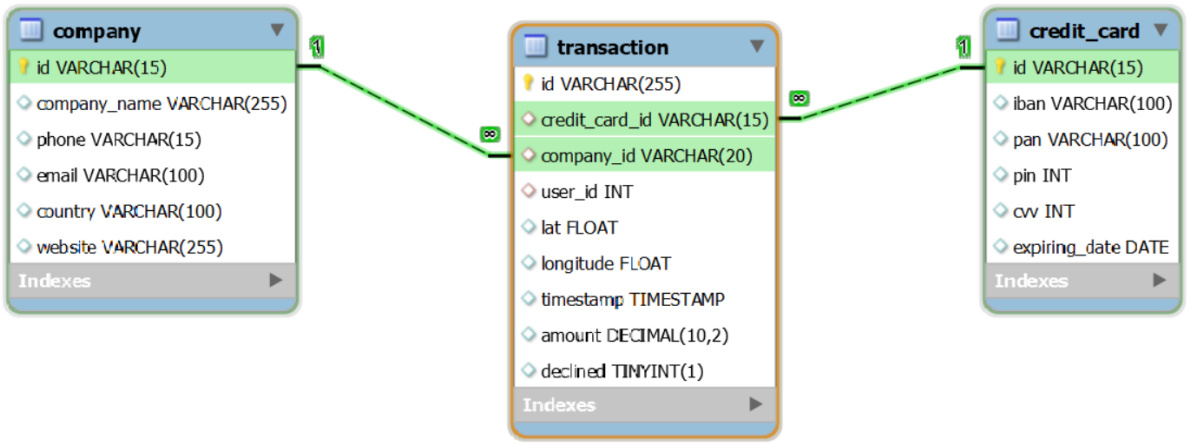
SPRINT 3: GESTIÓN DE TABLAS, ÍNDICES Y VISTAS

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 "dades_introduir_credit". Recuerda mostrar el diagrama y realizar una breve descripción de este.

En la base de datos *Transactions* se ha añadido una nueva tabla dimensión credit_card, que recopila los detalles de las tarjetas de crédito usadas en las transacciones registradas. El diagrama actualizado de este modelo en estrella se muestra a continuación:



La tabla credit_card se ha creado con las siguientes variables:

- Id (primary key): identificador único y no nulo para cada tarjeta de crédito; tipo de dato utilizado: VARCHAR (15), de acuerdo con cómo estaba definida la columna credit_card_id en la tabla transaction
- iban: identificador del número IBAN asociado a la tarjeta de crédito; VARCHAR (100), para poder incluir letras y números
- pan: número de cuenta principal de la tarjeta; VARCHAR (100) para poder cargar correctamente los datos proporcionados, ya que algunos registros tenían espacios en blanco y otros no
- pin: PIN de la tarjeta; INT
- cvv: código de seguridad CVV de la tarjeta; INT
- expiring_date: fecha de vencimiento de la tarjeta; VARCHAR (100) para poder cargar correctamente los datos proporcionados

NOTA: se ha adaptado el tipo de dato al formato de datos que teníamos que introducir; una vez cargados los datos, he modificado el formato de los datos en la columna expiring_date (mm/dd/yy) a YYYY-MM-DD, pudiendo cambiar el tipo de dato a DATE (véase los pasos explicados en el script Sprint3_NM.sql). He realizado este paso para practicar el cambio de formato y de datatype.

Es importante tener en cuenta este cambio dado que, si siguiéramos extrayendo los datos de origen en formato string, daría error; para solucionarlo, se podría cambiar el formato de fecha antes de cargar los datos en la tabla o pedir que los datos de origen tengan el formato de DATE. De lo contrario, sería

mejor mantener los datos en VARCHAR, tratando las fechas como si fuesen string en los análisis posteriores.

```
CREATE TABLE IF NOT EXISTS credit_card (
    id VARCHAR(15) PRIMARY KEY,
    iban VARCHAR(100),
    pan VARCHAR(100),
    pin INT,
    cvv INT,
    expiring_date VARCHAR(100)
);
```

A continuación, se ha creado una foreign key (FK) en la tabla de transaction para relacionar el id de tarjetas de crédito. Con esto, obtenemos la relación N-1 entre las tablas transaction y credit_card (transaction N – 1 credit_card).

Asimismo, he creado un índice en la tabla de hechos para el id de tarjeta de crédito (FK). La creación de este índice podría optimizar las operaciones de JOIN y acelerar la recuperación de datos relacionados entre tablas.

```
CREATE INDEX idx_credit_card_id ON transaction(credit_card_id);
ALTER TABLE transaction
ADD FOREIGN KEY (credit_card_id) REFERENCES credit_card(id);
```

[CORRECCIÓN: después de la corrección P2P, he modificado el tipo de dato de la columna cvv de INT a VARCHAR; la columna PIN se modificará en el ejercicio 3.1]

```
ALTER TABLE credit_card
MODIFY COLUMN cvv VARCHAR(3);
```

<u>Ejercicio 2.</u> El departamento de Recursos Humanos ha identificado un error en el número de cuenta del usuario con ID CcU-2938. Se requiere actualizar la información que identifica una cuenta bancaria a nivel internacional (identificado como "IBAN"): TR323456312213576817699999. Recuerda mostrar que el cambio se realizó.

Actualizamos los datos de la cuenta IBAN del usuario CcU-2938 mediante UPDATE.

```
UPDATE credit_card
SET iban = 'TR323456312213576817699999'
WHERE id='CcU-2938';
```

Antes de realizar el cambio, el usuario tenía IBAN TR301950312213576817638661.

Después del cambio, los datos de IBAN se han modificado por los correctos, manteniendo intactos los demás datos registrados para esa tarjeta de crédito.

id	iban	pan	pin	cvv	expiring_date
CcU-2938	TR323456312213576817699999	5424465566813633	3257	984	2022-10-30

<u>Ejercicio 3.</u> En la tabla "transaction" ingresa 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
lat	829.999
longitude	-117.999
amount	111.11
declined	0

Después de escribir el código para insertar los valores especificados, aparece un error debido a la restricción generada por Foreign Keys definidas. Como queremos añadir registros de credit_card_id, company_id y user_id, salta error al no tener datos de esos id's en sus respectivas tablas dimensión.

Para resolverlo, he añadido en cada tabla dimensión el identificador que toca. Los demás campos de cada tabla dimensión tendrán valores NULL, que se podrán actualizar más adelante con la información pertinente.

```
INSERT INTO company (id)
VALUES ('b-9999');

INSERT INTO credit_card (id)
VALUES ('CcU-9999');

INSERT INTO user (id)
VALUES ('9999');

#Ya no habría problemas para cargar los datos en la tabla de transacciones
INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined)
VALUES ('108B1D1D-5B23-A76C-55EF-C568E49A99DD', 'CcU-9999', 'b-9999', '9999', '829.999', '-117.999', '111.11', '0');
```

Como resultado, obtenemos 588 registros de transacciones, uno más respecto los que teníamos antes. El registro insertado aparece en la tabla a continuación, con registro NULL para timestamp ya no se nos proporcionó ningún valor.

id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined
108B1D1D-5B23-A76C-55EF-C568E49A99...	CcU-9999	b-9999	9999	829.999	-117.999	NULL	111.11	0

<u>Ejercicio 4.</u> Desde recursos humanos te solicitan eliminar la columna "pan" de la tabla credit_card. Recuerda mostrar el cambio realizado.

Para eliminar la columna 'pan' tenemos que cambiar la tabla credit_card (ALTER TABLE) y especificar que el cambio que queremos hacer en esta tabla es DROP COLUMN.

```
ALTER TABLE credit_card
DROP COLUMN pan;
```

Como resultado, con la consulta SHOW COLUMNS FROM credit_card vemos que ya no aparece la columna 'pan'.

Field	Type	Null	Key	Default	Extra
id	varchar(15)	NO	PRI	NULL	
iban	varchar(100)	YES		NULL	
pin	int	YES		NULL	
cvv	varchar(3)	YES		NULL	
expiring_date	date	YES		NULL	

NIVEL 2

<u>Ejercicio 1.</u> Elimina el registro con ID 02C6201E-D90A-1859-B4EE-88D2986D3B02 de la base de datos.

Para eliminar el registro de la tabla transaction usamos DELETE, especificando el id del registro que queremos borrar.

```
DELETE FROM transaction
WHERE id='02C6201E-D90A-1859-B4EE-88D2986D3B02';

SELECT *
FROM transaction
WHERE id='02C6201E-D90A-1859-B4EE-88D2986D3B02';
```

Después de ejecutar la query para eliminar el registro, si hacemos la consulta de ese registro ya no nos aparece nada como resultado. Por tanto, después del cambio la tabla transaction tiene un total de 587 registros.

id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

<u>Ejercicio 2.</u> 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 que crees una vista llamada VistaMarketing que contenga la siguiente información: Nombre de la compañía. Teléfono de contacto. País de residencia. Promedio de compra realizado por cada compañía. Presenta la vista creada, ordenando los datos de mayor a menor promedio de compra.

Para crear la vista necesitamos calcular el promedio de compra realizado por cada compañía (GROUP BY company.id).

CORRECCIÓN P2P: Dado que la pregunta especifica que el promedio sea de compras, considero las transacciones aceptadas (declined = 0).

Seleccionamos todos los campos que nos piden de la tabla company y lo relacionamos con la tabla transaction para calcular el promedio.

NOTA: dado que la vista es para la sección de marketing, he añadido alias a los encabezados de la vista para que sea más claro qué información se muestra en cada columna.

```sql
CREATE VIEW VistaMarketing AS
(
SELECT  c.company_name AS nombre_compañia,
        c.phone AS telefono_contacto,
        c.country AS pais_residencia,
        ROUND(AVG(t.amount),2) AS promedio_compras
FROM company c
JOIN transaction t
ON c.id=t.company_id
WHERE declined=0
GROUP BY c.id
);

SELECT *
FROM VistaMarketing
ORDER BY promedio_compras DESC;
```

Como resultado obtenemos una vista con 101 registros de compañías con los campos solicitados, ordenado de mayor a menor promedio. Hay un registro con todos los valores NULL excepto el promedio de compras, que corresponde a la compañía b-9999.

nombre_compañia	telefono_contacto	pais_residencia	promedio_compras
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

<u>Ejercicio 3.</u> Filtra la vista VistaMarketing para mostrar sólo las compañías que tienen su país de residencia en "Germany".

Para obtener datos de las compañías cuyo país de residencia es Alemania, tratamos la vista VistaMarketing creada como una tabla y la filtramos por el país.

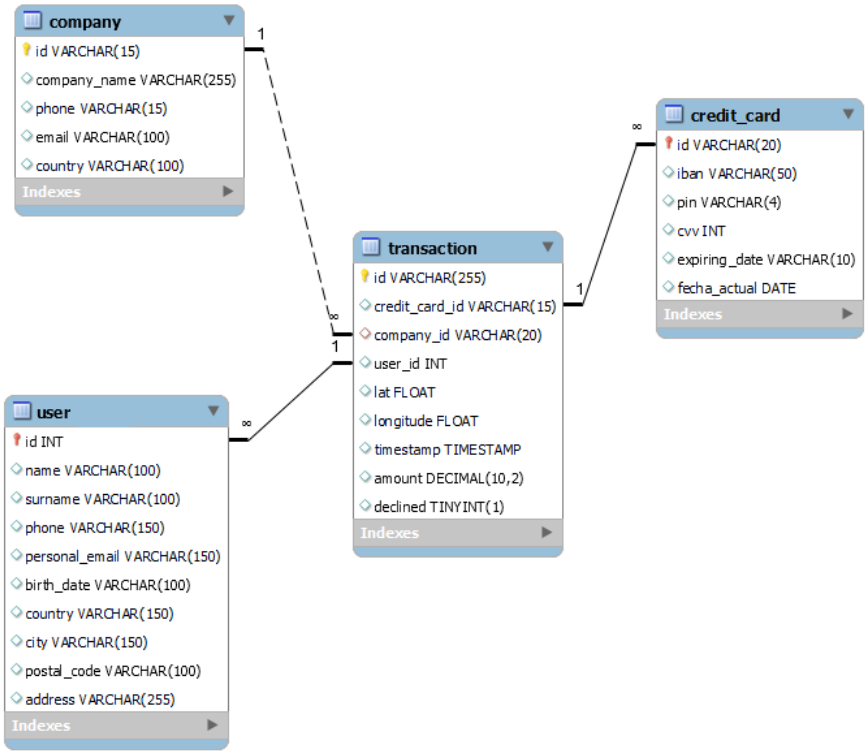
```sql
SELECT *
FROM VistaMarketing
WHERE pais_residencia = 'Germany'
ORDER BY promedio_compras DESC;
```

Como resultado, obtenemos 8 compañías con país de residencia en Alemania: Ac Industries, Auctor Mauris Corp., Ac Fermentum Incorporated, Aliquam PC, Rutrum Non Inc., Nunc Interdum Incorporated, Convallis In Incorporated y Augue Foundation.

nombre_compañia	telefono_contacto	pais_residencia	promedio_compras
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
Convallis In Incorporated	06 66 57 29 50	Germany	60.99
Augue Foundation	06 88 43 15 63	Germany	15.05

NIVEL 3

<u>Ejercicio 1.</u> 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:



En esta actividad, es necesario que describas el "paso a paso" de las tareas realizadas.

Hasta ahora, la base de datos *Transactions* tenía tres tablas: tabla de hechos transaction y dos tablas dimensión, company y credit_card.

Se ha añadido una nueva tabla dimensión al modelo: user. Contiene variables que proporcionan detalles sobre los usuarios que realizaron las transacciones: nombre y apellido, teléfono, correo electrónico, fecha de nacimiento, país, ciudad, código postal y la dirección. Todos tienen el tipo de dato VARCHAR.

Además, se realizaron los siguientes cambios en las tablas de la base de datos:

- Tabla company: se ha borrado la columna website

- Tabla user: se ha modificado el nombre de la columna email (script original) a personal_email (diagrama proporcionado)

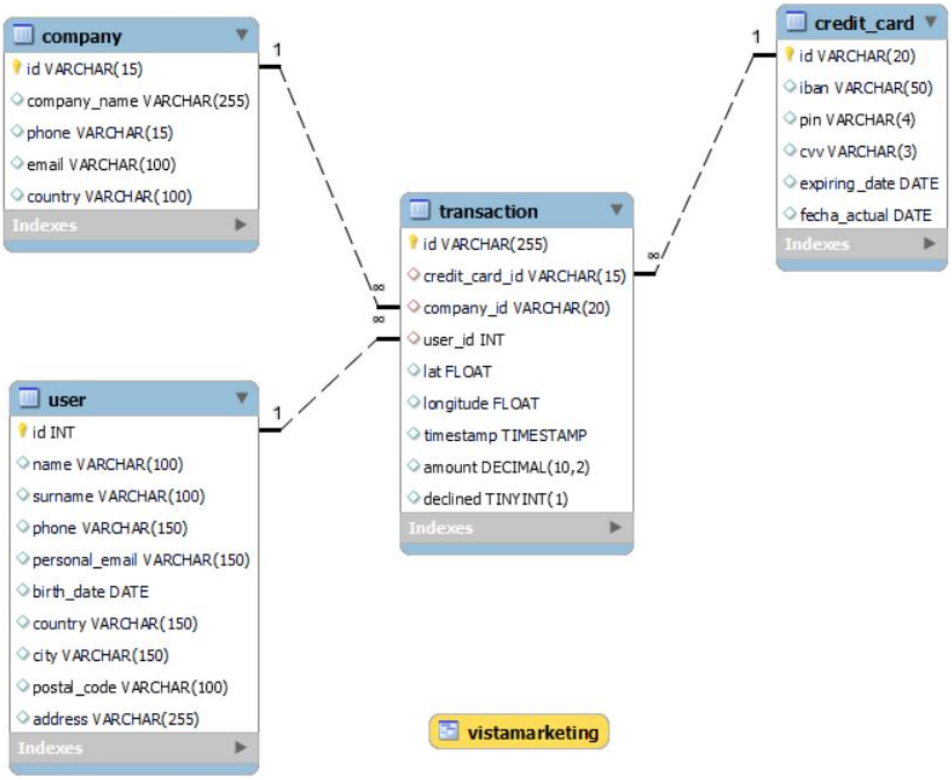
- Tabla credit_card:
    - se ha añadido una columna llamada fecha_actual con tipo de dato DATE

        - Únicamente con el nombre de la columna no podemos saber si contiene datos y, en caso de los contenga, qué datos son; únicamente sabemos que serán de tipo DATE
        - Por el nombre, se puede intuir que sea la fecha actual, que se puede añadir con la función CURRENT_DATE (); puede servir para comprobar cuánto queda para la fecha de vencimiento de las tarjetas o si ya caducaron
    - se ha cambiado el tipo de dato de la columna pin a VARCHAR
        - Es posible que esto sea uno de los cambios realizados; sin embargo, al ser una tabla que hemos diseñado y creado de cero, quizás desde el inicio la columna pin estaba definida como VARCHAR; en mi caso, estaba definida como INT, por lo que lo considero como un cambio
    - CORRECION P2P. Otros posibles cambios:
        - credit_card.id de VARCHAR (15) a VARCHAR (20)
        - credit_card.iban de VARCHAR (100) a VARCHAR (50)

- No aparece la VistaMarketing creada para la sección de marketing
    - Es posible que esto no sea un cambio, sino que para hacer el diagrama únicamente se tuvieron en cuenta las dimensiones y la tabla de hechos del modelo
        - De lo contrario, el compañero eliminó la vista con este comando: DROP VIEW VistaMarketing;

```sql
-- cambio nombre de columna email en la tabla user
ALTER TABLE user
RENAME COLUMN email TO personal_email;

-- borrar columna website de la tabla company
ALTER TABLE company
DROP COLUMN website;

-- añadir columna fecha_actual en la tabla credit_card
ALTER TABLE credit_card
ADD COLUMN fecha_actual DATE;

UPDATE credit_card
SET fecha_actual = CURRENT_DATE()
WHERE id BETWEEN 'CcU-2938' AND 'CcU-9999';

-- modificar tipo de dato de columna pin de la tabla credit_card
ALTER TABLE credit_card
MODIFY COLUMN pin VARCHAR(4);

-- otros posibles cambios en la tabal credit_card
ALTER TABLE credit_card
MODIFY COLUMN id VARCHAR(20);

ALTER TABLE credit_card
MODIFY COLUMN iban VARCHAR(50);
```

Una vez realizados estos cambios, obtenemos el siguiente diagrama del modelo: ***



*** NOTA 1: las relaciones entre las tablas dimensión y la tabla de hechos no coincide con el diagrama planteado en el ejercicio; esto es debido a que en el script estuctura_datos_user hay un código (FOREIGN KEY(id) REFERENCES transaction(user_id)) que relaciona la tabla de hechos con la tabla user de 1 a N (Transaction 1 – N user); no he ejecutado esa parte del script dado que la relación real es la inversa (Transaction N – 1 user), es decir, un mismo usuario puede realizar varias transacciones; además, los id de usuarios de la tabla user son PK, lo cual quiere decir que son únicos y unívocos.

Por tanto, en su lugar definí la FK en la tabla de hechos, ya que a través de la columna transaction.user_id relacionamos ambas tablas.

```
CREATE INDEX idx_user_id ON transaction(user_id);
ALTER TABLE transaction
ADD FOREIGN KEY (user_id) REFERENCES user(id);
```

*** NOTA 2: tal y como he explicado en el primer ejercicio, he modificado credit_card.expiring_date de VARCHAR a DATE para practicar el cambio de formato y por si interesaba más trabajar con formato fecha; por el mismo motivo, he modificado user.birth_date de VARCHAR a DATE.

```
UPDATE user
SET birth_date=STR_TO_DATE(birth_date, '%b %d, %Y')
WHERE id > 0;

ALTER TABLE user
MODIFY birth_date DATE;
```

<u>Ejercicio 2.</u> La empresa también te 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**

Asegúrate de incluir información relevante de ambas tablas y utiliza 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.

Para crear vista con la información solicitada, tenemos que recopilar información de todas las tablas del modelo. Por tanto, las relacionamos con la tabla Transaction mediante JOIN.

NOTA: he añadido alias a los encabezados de la vista para que sea más claro qué información se muestra en cada columna.

```
CREATE VIEW InformeTecnico AS
(
SELECT  t.id AS ID_transaccion,
        u.name AS Nombre_usuario,
        u.surname AS Apellido_usuario,
        cred.iban AS IBAN_tarjeta,
        c.company_name AS Nombre_compañia
FROM transaction t
JOIN user u
    ON t.user_id=u.id
JOIN credit_card cred
    ON t.credit_card_id=cred.id
JOIN company c
    ON t.company_id=c.id
);

SELECT *
FROM InformeTecnico
ORDER BY ID_transaccion DESC;
```

Como resultado, obtenemos una vista con 587 registros de transacciones y los detalles solicitados de éstas. El ID_transaccion = 108B1D1D-5B23-A76C-55EF-C568E49A99DD (añadida en 1.3) tiene valores NULL para el resto de campos. Los cinco primeros resultados se muestran en la siguiente tabla:

ID_transaccion	Nombre_usuario	Apellido_usuario	IBAN_tajeta	Nombre_compañia
FE96CE47-BD59-381C-4E18-E3CA3D44E8FF	Kenyon	Hartman	DO26854763748537475216568689	Magna A Neque Industries
FE809ED4-2DB6-55AC-C915-929516E4646B	Molly	Gilliam	SE281312348716362853112l	Nunc Interdum Incorporated
FD9CBCCD-8E1E-8DA1-4606-7E3A6F3A5A65	Linus	Willis	KW9485332754781757886242955643	Nunc Interdum Incorporated
FD89D51B-AE8D-77DC-E450-B8083FBD3187	Hilda	Levy	LT053237077744561475	Malesuada PC
FD2E8957-414B-BEEC-E9AD-59AA7A8A6290	Hedwig	Gilbert	GE84848451582810541526	Neque Tellus Imperdiet Corp.