

# DATA ANALYTICS

IT ACADEMY | BCN ACTIVA

## SPRINT 03

Manipulación de tablas  
30 de abril de 2025



Letiane Benincá  
benincalf@gmail.com

IT ACADEMY



## DESCRIPCIÓN

En este sprint, se simula una situación empresarial en la que debes realizar varias manipulaciones en las tablas de la base de datos. A su vez, tendrás que trabajar con índices y vistas. En esta actividad, continuarás trabajando con la base de datos que contiene información de una empresa dedicada a la venta de productos en línea. En esta tarea, comenzarás a trabajar con información relacionada con tarjetas de crédito.

### NIVEL 1

#### EJERCICIO 01

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 manera 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.

##### Creación de la tabla credit\_card:

La tabla fue creada de acuerdo con la lógica de los datos a introducir de crédito. Los campos creados fueron:

- **id VARCHAR(255):** clave primaria, alfanuméricos de hasta 15 caracteres. ['CcU-2938']
- **Iban VARCHAR(255):** International Bank Account Number, alfanuméricos de hasta 50 caracteres. ['TR301950312213576817638661']
- **pan VARCHAR(255):** Primary Account Number, almacena el número de la tarjeta, alfanuméricos de hasta 20 caracteres. ['5424465566813633']
- **pin VARCHAR(255):** Código secreto de la tarjeta, alfanuméricos de hasta 4 caracteres. ['3257']
- **cvv VARCHAR(255):** Card Verification Value, alfanuméricos de hasta 3 caracteres. ['984']
- **expiring\_date VARCHAR(255):** la fecha de caducidad de la tarjeta en formato alfanuméricos de hasta 10 caracteres. ['10/30/22'] # He intentado utilizar el formato DATE, pero el default de MySQL es XX/XX/XXXX, lo que generaba un error a utilizarlo.

*\*PS: Siempre iniciar con VARCHAR(255), cuando no conocemos los datos para evitar errores de carga. Luego si puede adaptar.*

Añadí una clave foránea en la tabla "transaction" que vincula el campo "credit\_card\_id" con el "id" de la tabla "credit\_card".

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'SCHEMAS' pane shows a tree view with 'transacciones' expanded, showing 'credit\_card' and 'transaction' tables. The main pane displays the following SQL script:

```

30
31 -- Nivel 01: Ejercicio 01:
32 -- Diseñar y crear una tabla llamada "credit_card" que almacene detalles cruciales sobre las tarjetas de crédito.
33 CREATE TABLE IF NOT EXISTS credit_card (
34     id VARCHAR(255) PRIMARY KEY,
35     iban VARCHAR(255),
36     pan VARCHAR(255),
37     pin VARCHAR(255),
38     cvv VARCHAR(255),
39     expiring_date VARCHAR(255)
40 );
41
42 ALTER TABLE transaction
43 ADD FOREIGN KEY (credit_card_id)
44 REFERENCES credit_card(id);
45

```

The 'Output' pane shows the execution results:

#	Time	Action	Message	Duration / Fetch
710	15:44:09	CREATE TABLE IF NOT EXISTS credit_card ( id VARCHAR(255) PRIMA...	0 row(s) affected	0.032 sec
711	15:44:24	ALTER TABLE transaction ADD FOREIGN KEY (credit_card_id) REFEREN...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.078 sec

Query Completed

Finalmente, inserté los datos del archivo "datos\_introducir\_credit".

The screenshot shows the SQL Server Enterprise Manager interface. The main pane displays the following SQL script:

```

40 );
41
42 ALTER TABLE transaction
43 ADD FOREIGN KEY (credit_card_id)
44 REFERENCES credit_card(id);
45
46 SELECT *
47 FROM credit_card;

```

The 'Result Grid' shows the data inserted into the 'credit\_card' table:

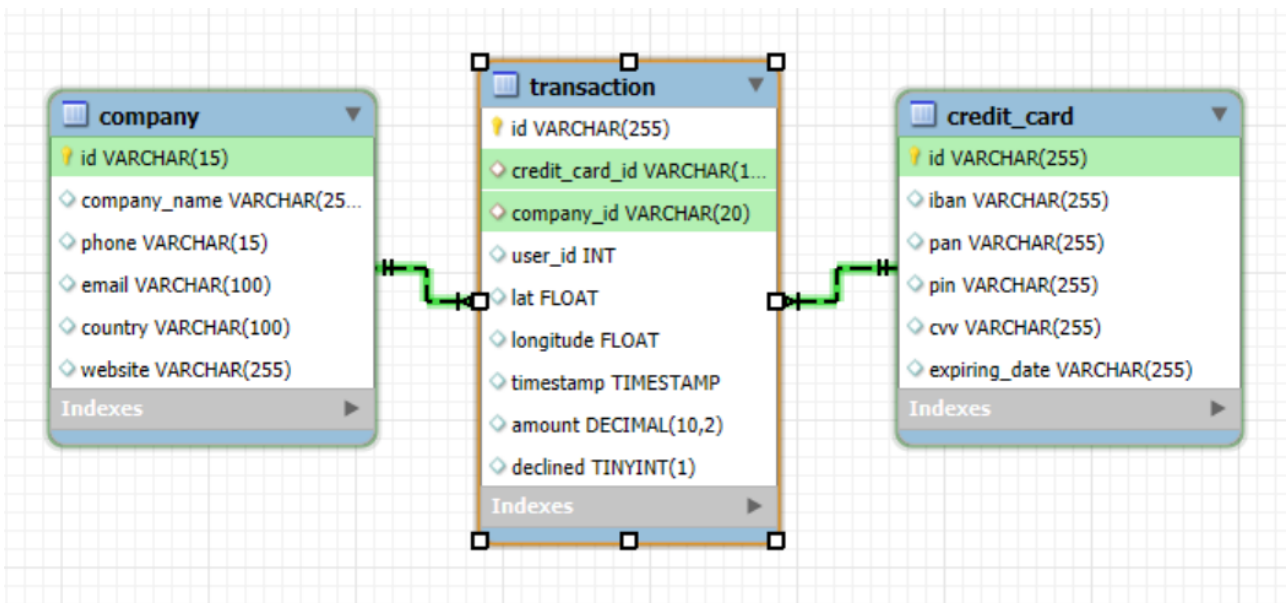
id	iban	pan	pin	cvv	expiring_date
CdU-2938	TR301950312213576817638661	5424465566813633	3257	984	10/30/22
CdU-2945	DO26854763748537475216568689	5142423821948828	9080	887	08/24/23
CdU-2952	BG451VQL52710525608255	4556 453 55 5287	4598	438	06/29/21
CdU-2959	CR7242477244335841535	372461377349375	3583	667	02/24/23
CdU-2966	BG72LKTQ15627628377363	448566 886747 7265	4900	130	10/29/24
CdU-2973	PT87806228135092429456346	544 58654 54343 384	8760	887	01/30/25

The 'Output' pane shows the execution results:

#	Time	Action	Message	Duration / Fetch
986	15:45:38	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ( ...	1 row(s) affected	0.000 sec
987	15:45:59	SELECT * FROM credit_card	275 row(s) returned	0.000 sec / 0.000 sec

Query Completed

Utilizando el reverse engineer, verificamos las cardinalidades de las tablas. Inicialmente la clave foránea de la nueva tabla no estaba bien configurada y para esto si ha añadido una restricción para la misma. Presentamos el diagrama final:



La base de datos es relacional con un esquema en estrella, donde la tabla **transaction** es la tabla de hechos y **company** y **credit\_card** son tablas dimensionales. Las relaciones entre las tablas son de tipo 1 a N, ya que una empresa o tarjeta de crédito pueden estar asociadas a múltiples transacciones, pero cada transacción es única. Las claves primarias (PK) de las tablas dimensionales (id) se vinculan como claves foráneas (FK) en la tabla de hechos.

## EJERCICIO 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. Recuerda demostrar que el cambio se ha realizado.

En este caso, primero conferimos los datos existentes:

SCHEMAS

Filter objects

- energia\_db
- hospitales
- nens\_nenes
- olympics
- sakila
- sys
- tienda\_online
- transactions**
  - company
  - credit\_card
  - transaction
  - Views
  - Stored Procedures

Administration Schemas

Information

Schema: transactions

Object Info Session

```

47 FROM credit_card;
48
49 -- Nivel 01: Ejercicio 02:
50 -- El departamento de Recursos Humanos ha identificado un error en el número de cuenta del usuario con ID CcU-2938.
51 -- La información que debe mostrarse para este registro es: R323456312213576817699999.
52 * SELECT *
53 FROM credit_card
54 WHERE id = 'CcU-2938';

```

Result Grid

id	iban	pan	pin	cvv	expiring_date
CcU-2938	TR30195031221357681763861	5424465566813633	3257	984	10/30/22

credit\_card 2 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
987	15:45:59	SELECT * FROM credit_card	275 row(s) returned	0.000 sec / 0.000 sec
988	15:48:58	SELECT * FROM credit_card WHERE id = 'CcU-2938'	1 row(s) returned	0.000 sec / 0.000 sec

Query Completed

Por medio del uso del UPDATE TABLE, hacemos el cambio de información del usuario 'CcU-2938'.

SCHEMAS

Filter objects

- energia\_db
- hospitales
- nens\_nenes
- olympics
- sakila
- sys
- tienda\_online
- transactions**
  - company
  - credit\_card
  - transaction
  - Views
  - Stored Procedures

Administration Schemas

Information

Schema: transactions

Object Info Session

```

55
56 * UPDATE credit_card
57 SET iban = 'R32345631221357681769999'
58 WHERE id = 'CcU-2938';
59
60 * SELECT *
61 FROM credit_card
62 WHERE id = 'CcU-2938';

```

Result Grid

id	iban	pan	pin	cvv	expiring_date
CcU-2938	R32345631221357681769999	5424465566813633	3257	984	10/30/22

credit\_card 3 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
989	15:51:24	UPDATE credit_card SET iban = 'R32345631221357681769999' WHERE id = 'CcU-2938'	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
990	15:51:28	SELECT * FROM credit_card WHERE id = 'CcU-2938'	1 row(s) returned	0.000 sec / 0.000 sec

Query Completed

## EJERCICIO 3

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

Inicialmente, verificamos se no existen los datos en la tabla:

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'SCHEMAS' pane shows the 'transactions' schema. The main query window contains the following SQL code:

```

64 -- Nivel 01: Ejercicio 03:
65 -- En la tabla "transaction" ingresa un nuevo usuario:
66 SELECT *
67 FROM transaction
68 WHERE id = '108B1D1D-5B23-A76C-55EF-C568E49A990D';
69
70
71

```

The 'Result Grid' shows the output of the query, which is empty. The 'Output' pane shows the execution results:

#	Time	Action	Message	Duration / Fetch
990	15:51:28	SELECT * FROM credit_card WHERE id = 'CcU-2938'	1 row(s) returned	0.000 sec / 0.000 sec
991	15:53:39	SELECT * FROM transaction WHERE id = '108B1D1D-5B23-A76C-55EF-C5...	0 row(s) returned	0.016 sec / 0.000 sec

Query Completed

Para añadir los datos de nuevo usuario fue utilizado el INSERT INTO:

- Inicialmente en las tablas **company** y **credit\_card**, luego inserimos los datos en la tabla **transaction**, quedando fuera el timestamp que no estaba en los datos, siendo que este quedará NULL.

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'SCHEMAS' pane shows the 'transactions' schema. The main query window contains the following SQL code:

```

68 WHERE id = '108B1D1D-5B23-A76C-55EF-C568E49A990D';
69
70 INSERT INTO company (id)
71 VALUES ('b-9999');
72
73 INSERT INTO credit_card (id)
74 VALUES ('CcU-9999');
75
76 INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined)
77 VALUES ('108B1D1D-5B23-A76C-55EF-C568E49A990D', 'CcU-9999', 'b-9999', 9999, 829.999, -117.999, 111.11, 0);
78
79 SELECT *
80 FROM transaction
81 WHERE id = '108B1D1D-5B23-A76C-55EF-C568E49A990D';
82
83

```

The 'Output' pane shows the execution results:

#	Time	Action	Message	Duration / Fetch
993	15:54:53	INSERT INTO credit_card (id) VALUES (CcU-9999)	1 row(s) affected	0.016 sec
994	15:55:06	INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longit...	1 row(s) affected	0.000 sec

Query Completed



Verificamos la inserción de los datos:

The screenshot shows a database management tool interface. On the left, the 'SCHEMAS' pane shows a tree view with 'transactions' selected. The main query editor contains the following SQL code:

```

76 • INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined)
77 VALUES ('108B1D1D-5B23-A76C-55EF-C568E49A99DD', 'ccu-9999', 'b-9999', 9999, 829.999, -117.999, 111.11, 0);
78
79 • SELECT *
80 FROM transaction
81 WHERE id = '108B1D1D-5B23-A76C-55EF-C568E49A99DD';
82
83

```

The 'Result Grid' shows the following data:

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	111.11	0	

The 'Output' pane shows the following messages:

#	Time	Action	Message	Duration / Fetch
994	15:55:06	INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined)	1 row(s) affected	0.000 sec
995	15:55:38	SELECT * FROM transaction WHERE id = '108B1D1D-5B23-A76C-55EF-C568E49A99DD'	1 row(s) returned	0.000 sec / 0.000 sec

Query Completed

## EJERCICIO 4

Desde Recursos Humanos solicitan eliminar la columna "pan" de la tabla *credit\_card*. Recuerda demostrar que el cambio se ha realizado.

Verificamos la tabla *credit\_card* la columna pan:

The screenshot shows the same database management tool interface. The 'SCHEMAS' pane shows 'credit\_card' selected. The main query editor contains the following SQL code:

```

79 • SELECT *
80 FROM transaction
81 WHERE id = '108B1D1D-5B23-A76C-55EF-C568E49A99DD';
82
83 -- Nivel 01: Ejercicio 04:
84 -- Desde Recursos Humanos solicitan eliminar la columna "pan" de la tabla credit_card.
85 • SELECT pan
86 FROM credit_card;
87

```

The 'Result Grid' shows the following data:

pan
5424465566813633
5142423821948828
4556 453 55 5287
372461377349375
448566 886747 7265
544 58654 54343 384

The 'Output' pane shows the following messages:

#	Time	Action	Message	Duration / Fetch
995	15:55:38	SELECT * FROM transaction WHERE id = '108B1D1D-5B23-A76C-55EF-C568E49A99DD'	1 row(s) returned	0.000 sec / 0.000 sec
996	15:57:26	SELECT pan FROM credit_card	276 row(s) returned	0.000 sec / 0.000 sec

Query Completed

Para alterar la tabla, utilizamos el `ALTER TABLE credit_card DROP COLUMN pan` eliminamos la columna. Así queda el resultado, la tabla *credit\_card* ya no tiene la columna pan.

**SCHEMAS**

Filter objects

- energia\_db
- hospitales
- nens\_nenes
- olympics
- sakila
- sys
- tienda\_online
- transactions**
  - Tables
    - company
    - credit\_card
    - transaction
  - Views
- Stored Procedures

Administration Schemas

Information

Schema: **transactions**

Object Info Session

Query Completed

87

88 • ALTER TABLE credit\_card

89 DROP COLUMN pan;

90

91 • SELECT \*

92 FROM credit\_card;

93

94

Result Grid

	id	iban	pin	cvv	expiring_date
▶	CcU-2938	R32345631221357681769999	3257	984	10/30/22
	CcU-2945	DO26854763748537475216568689	9080	887	08/24/23
	CcU-2952	BG45IVQL52710525608255	4598	438	06/29/21
	CcU-2959	CR7242477244335841535	3583	667	02/24/23
	CcU-2966	BG72LKTQ15627628377363	4900	130	10/29/24
	CcU-2973	PT87806228135092429456346	8760	887	01/30/25

credit\_card 7 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
997	15:58:40	ALTER TABLE credit_card DROP COLUMN pan	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.032 sec
998	15:58:43	SELECT * FROM credit_card	276 row(s) returned	0.000 sec / 0.000 sec



## NIVEL 2

### EJERCICIO 01

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

Primero verificamos el registro:

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'SCHEMAS' pane shows the 'transactions' schema. The main pane displays a query plan for a SELECT statement. The query is as follows:

```

93
94 -- Nivel 02: Ejercicio 01:
95 -- Elimina de la tabla transaction el registro con ID 02C6201E-D90A-1859-B4EE-88D2986D3B02 de la base de datos.
96 • SELECT *
97 FROM transaction
98 WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02';
99
100

```

The 'Result Grid' shows the following columns: id, credit\_card\_id, company\_id, user\_id, lat, longitude, timestamp, amount, declined. The 'Output' pane shows the execution results:

#	Time	Action	Message	Duration / Fetch
998	15:58:43	SELECT * FROM credit_card	276 row(s) returned	0.000 sec / 0.000 sec
999	15:59:38	SELECT * FROM transaction WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02';	0 row(s) returned	0.000 sec / 0.000 sec

Para eliminarlo utilizamos el DELETE FROM con el filtro WHERE con el código del registro:

Una vez eliminado, volvemos a seleccionar para comprobar que se haya eliminado correctamente.

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'SCHEMAS' pane shows the 'transactions' schema. The main pane displays a query plan for a DELETE statement. The query is as follows:

```

99
100 • DELETE FROM transaction
101 WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02';
102
103 • SELECT *
104 FROM transaction
105 WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02';
106

```

The 'Result Grid' shows the following columns: id, credit\_card\_id, company\_id, user\_id, lat, longitude, timestamp, amount, declined. The 'Output' pane shows the execution results:

#	Time	Action	Message	Duration / Fetch
1000	16:00:52	DELETE FROM transaction WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02';	0 row(s) affected	0.000 sec
1001	16:00:57	SELECT * FROM transaction WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02';	0 row(s) returned	0.000 sec / 0.000 sec

## EJERCICIO 2

El departamento 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 realizada por cada compañía.

Presenta la vista creada, ordenando los datos de mayor a menor promedio de compra.

Usamos el comando **CREATE VIEW AS** para crear la vista. Seleccionamos el nombre de la compañía, el teléfono y el país, además de calcular el promedio de compras a partir de las tablas **company** y **transaction**, unidas mediante un JOIN por el id de la compañía. Agrupamos los resultados por los datos que deseamos mostrar y los ordenamos de forma descendente por el promedio.

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'SCHEMAS' pane shows the 'transactions' schema selected. The main pane displays the SQL script for creating the view:

```

111 FROM company co
112 JOIN transaction tr
113 ON tr.company_id = co.id
114 GROUP BY 1, 2, 3
115 ORDER BY 4 DESC;
116
117 SELECT *
118 FROM VistaMarketing;

```

Below the script, the 'Result Grid' shows the data returned by the query. The columns are: company\_name, phone, country, and Promedio\_Compras. The data is sorted by Promedio\_Compras in descending order.

company_name	phone	country	Promedio_Compras
Eget Ipsum Ltd	03 67 44 56 72	United States	473.08
Non Magna LLC	06 71 73 13 17	United Kingdom	468.35
Sed Id Limited	07 28 18 18 13	United States	461.21
Justo Eu Arcu Ltd	08 42 56 71 52	Italy	443.64
Eget Tincidunt Dui Institute	05 35 93 32 44	Netherlands	442.52
Viverra Donec Foundation	03 33 12 32 73	United Kingdom	442.28
Vestibulum Lorem PC	02 02 87 33 40	Belgium	434.06

At the bottom, the 'Output' pane shows the execution results of the CREATE VIEW and SELECT statements.

Como resultados, tenemos la vista creada, y a la lateral izquierda aparece la creación de la vista.

\* En la entrega anterior, faltaba el ON en el JOIN, y los promedios estaban igual. Ahora se ha solucionado.

## EJERCICIO 3

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

En este caso, la vista funciona como una nueva tabla y hacemos una consulta directamente en ella, utilizando el filtro **WHERE** para encontrar el país deseado.

**SCHEMAS**

- Filter objects
- olympics
- sakila
- sys
- tienda\_online
- transactions
  - Tables
    - company
    - credit\_card
    - transaction
  - Views
    - vistamarketing
  - Stored Procedures
  - Functions
- world

Administration Schemas Information

Schema: transactions

```

119
120 -- Nivel 02: Ejercicio 03:
121 -- Filtra la vista VistaMarketing para mostrar solo las compañías que tienen su país de residencia en "Germany".
122 • SELECT *
123 FROM vistamarketing
124 WHERE country = 'Germany';
125
126
  
```

Result Grid

company_name	phone	country	Promedio_Compras
Aliquam PC	01 45 73 52 16	Germany	385.27
Ac Industries	09 34 65 40 60	Germany	289.65
Rutrum Non Inc.	02 66 31 61 09	Germany	266.90
Nunc Interdum Incorporated	05 18 15 48 13	Germany	244.03
Augue Foundation	06 88 43 15 63	Germany	240.80
Ac Fermentum Incorporated	06 85 56 52 33	Germany	206.47
Auctor Mauris Corp.	05 62 87 14 41	Germany	184.31

vistamarketing 18 x

Output

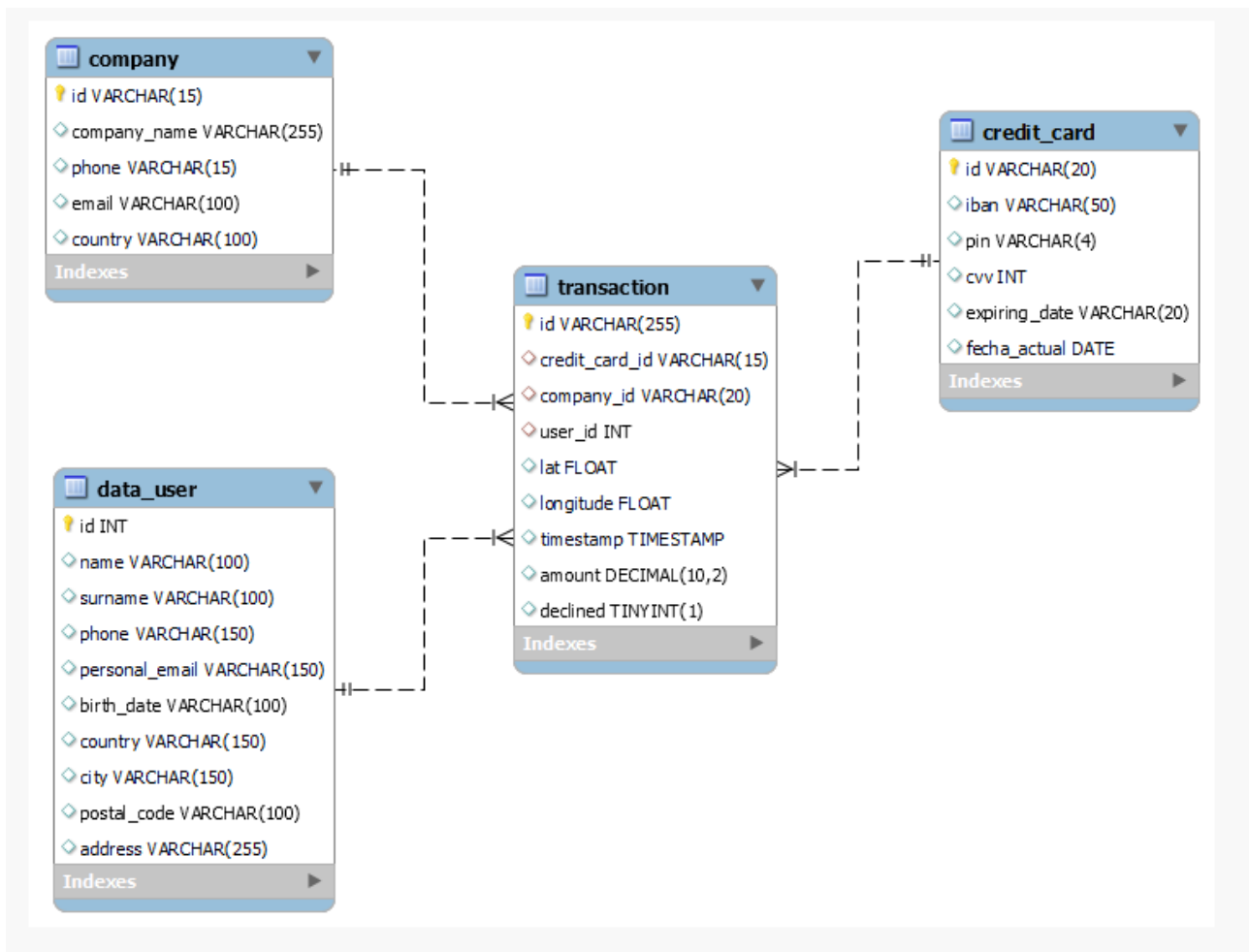
Action Output

#	Time	Action	Message	Duration / Fetch
1003	16:27:23	SELECT * FROM VistaMarketing	101 row(s) returned	0.016 sec / 0.000 sec
1004	16:29:36	SELECT * FROM vistamarketing WHERE country = 'Germany'	8 row(s) returned	0.016 sec / 0.000 sec

## NIVEL 3

### EJERCICIO 01

La próxima semana tendrás una 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 lo ayudes a documentar los comandos ejecutados para obtener el siguiente diagrama:



#### Recordatorio

En esta actividad, es necesario que describas el "paso a paso" de las tareas realizadas. Es importante realizar descripciones sencillas, simples y fáciles de comprender. Para llevar a cabo esta actividad, deberás trabajar con los archivos denominados "estructura\_dades\_user" y "dades\_introduir\_user".

Para ejecutar esta tarea, inicialmente, creamos la tabla user, que ya tenemos la estructura ordenada:

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'SCHEMAS' pane is open, showing the 'transactions' schema. The main pane displays the following SQL script:

```

127 -- documentar los comandos ejecutados para obtener el diagrama dado:
128 -- Primero creamos la tabla user:
129 CREATE INDEX idx_user_id ON transaction(user_id);
130
131 CREATE TABLE IF NOT EXISTS user (
132     id INT PRIMARY KEY,
133     name VARCHAR(100),
134     surname VARCHAR(100),
135     phone VARCHAR(150),
136     email VARCHAR(150),
137     birth_date VARCHAR(100),
138     country VARCHAR(150),
139     city VARCHAR(150),
140     postal_code VARCHAR(100),
141     address VARCHAR(255),
142     FOREIGN KEY(id) REFERENCES transaction(user_id)
143 );

```

The 'Output' pane at the bottom shows the execution results:

#	Time	Action	Message	Duration / Fetch
1005	16:32:19	CREATE INDEX idx_user_id ON transaction(user_id)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.046 sec
1006	16:32:25	CREATE TABLE IF NOT EXISTS user ( id INT PRIMARY KEY, na...	na... 0 row(s) affected	0.031 sec

Luego, inserimos los datos:

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'SCHEMAS' pane is open, showing the 'transactions' schema. The main pane displays the following SQL script:

```

1 SET foreign_key_checks = 0;
2
3 -- Insertamos datos de user
4 INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
5 INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
6 INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
7 INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
8 INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
9 INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
10 INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
11 INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
12 INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
13 INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
14 INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
15 INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
16 INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
17 INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
18 INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (

```

The 'Output' pane at the bottom shows the execution results:

#	Time	Action	Message	Duration / Fetch
1282	16:33:32	INSERT INTO user (id, name, surname, phone, email, birth_date, country, ci...	1 row(s) affected	0.000 sec
1283	16:33:32	SET foreign_key_checks = 1	0 row(s) affected	0.000 sec

Para adaptar las tablas con el modelo referenciado inicialmente es necesario ajustar algunas columnas y tablas:

## En la tabla user y company:

Schema: transactions

```

143  });
144
145  -- Cambios necesarios para adaptarse al modelo dado:
146  -- En la tabla user:
147  -- Cambiar el nombre tabla para data_user:
148  RENAME TABLE user
149  TO data_user;
150
151  -- Cambiar el nombre de la columna email para personal_email:
152  ALTER TABLE data_user
153  CHANGE email personal_email VARCHAR(150);
154
155  -- En la tabla company:
156  -- Eliminar la columna website:
157  ALTER TABLE company
158  DROP COLUMN website;
159

```

Output

#	Time	Action	Message	Duration / Fetch
1285	16:35:42	ALTER TABLE data_user CHANGE email personal_email VARCHAR(150)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.031 sec
1286	16:36:01	ALTER TABLE company DROP COLUMN website	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.032 sec

## En la tabla credit\_card:

Schema: transactions

```

159
160  -- En la tabla credit_card:
161  -- Creamos la columna fecha_actual:
162  ALTER TABLE credit_card
163  ADD COLUMN fecha_actual DATE;
164
165  -- Cambiar el tipo de datos del cvv para INT:
166  ALTER TABLE credit_card
167  MODIFY COLUMN cvv INT;
168
169  -- Cambiar el tipo de datos en expiring_date:
170  ALTER TABLE credit_card
171  MODIFY COLUMN expiring_date VARCHAR(20);
172
173  -- Cambiar el tipo de datos en pin:
174  ALTER TABLE credit_card
175  MODIFY COLUMN pin VARCHAR(4);

```

Output

#	Time	Action	Message	Duration / Fetch
1287	16:39:39	ALTER TABLE credit_card ADD COLUMN fecha_actual DATE	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.032 sec
1288	16:39:48	ALTER TABLE credit_card MODIFY COLUMN cvv INT	276 row(s) affected Records: 276 Duplicates: 0 Warnings: 0	0.078 sec
1289	16:40:24	ALTER TABLE credit_card MODIFY COLUMN expiring_date VARCHAR(20)	276 row(s) affected Records: 276 Duplicates: 0 Warnings: 0	0.078 sec



**Script SQL:**

```

173 -- Cambiar el tipo de datos en pin:
174 ALTER TABLE credit_card
175 MODIFY COLUMN pin VARCHAR(4);
176
177 -- Cambiar el tipo de datos en iban:
178 ALTER TABLE credit_card
179 MODIFY COLUMN iban VARCHAR(50);
180
181 -- Eliminar la clave foranea:
182 SET foreign_key_checks = 0;
183
184 -- Cambiar el tipo de datos en id:
185 ALTER TABLE credit_card
186 MODIFY COLUMN id VARCHAR(20);
187
188 -- Activar las claves:
189 SET foreign_key_checks = 1;

```

**Output:**

#	Time	Action	Message	Duration / Fetch
1290	16:40:52	ALTER TABLE credit_card MODIFY COLUMN expiring_date VARCHAR(4)	Error Code: 1265. Data truncated for column 'expiring_date' at row 1	0.031 sec
1291	16:41:04	ALTER TABLE credit_card MODIFY COLUMN pin VARCHAR(4)	276 row(s) affected Records: 276 Duplicates: 0 Warnings: 0	0.079 sec
1292	16:41:35	ALTER TABLE credit_card MODIFY COLUMN iban VARCHAR(50)	276 row(s) affected Records: 276 Duplicates: 0 Warnings: 0	0.078 sec

Falta ajustar la **foreign key** de **user\_id** en **transaction** e **id** en **data\_user**. Para corregir la relación de tipo 1 a N, será necesario eliminar la relación actual y agregar el registro previamente creado en la tabla **transaction** con **user\_id = 9999**. Si los datos no coinciden y existen más registros en **transaction** que no tienen correspondencia en la tabla **data\_user**, se generará una relación inversa, como la que tenemos ahora, donde la cardinalidad es de 1 a N de **transaction** hacia **data\_user**.

**Script SQL:**

```

185 ALTER TABLE credit_card
186 MODIFY COLUMN id VARCHAR(20);
187
188 -- Activar las claves:
189 SET foreign_key_checks = 1;
190
191 -- Eliminar la clave foranea:
192 ALTER TABLE data_user
193 DROP FOREIGN KEY data_user_ibfk_1;
194
195 -- Registramos el usuario 9999:
196 INSERT INTO data_user (id)
197 VALUES (9999);
198
199 -- Crear la foreign key entre las tablas de hecho y dimension:
200 ALTER TABLE transaction
201 ADD FOREIGN KEY (user_id) REFERENCES data_user(id);

```

**Table: transaction**

Column	Data Type
id	varchar(255) PK
credit_card_id	varchar(15)
company_id	varchar(20)
user_id	int
lat	float
longitude	float
timestamp	timestamp
amount	decimal(10,2)
declined	tinyint(1)

**Output:**

#	Time	Action	Message	Duration / Fetch
1298	16:49:05	ALTER TABLE data_user DROP FOREIGN KEY data_user_ibfk_1	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.031 sec
1299	16:51:48	INSERT INTO data_user (id) VALUES (9999)	1 row(s) affected	0.015 sec
1300	16:52:22	ALTER TABLE transaction ADD FOREIGN KEY (user_id) REFERENCES data_user(id)	587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0	0.140 sec

Con estos cambios, comparamos los diagramas:

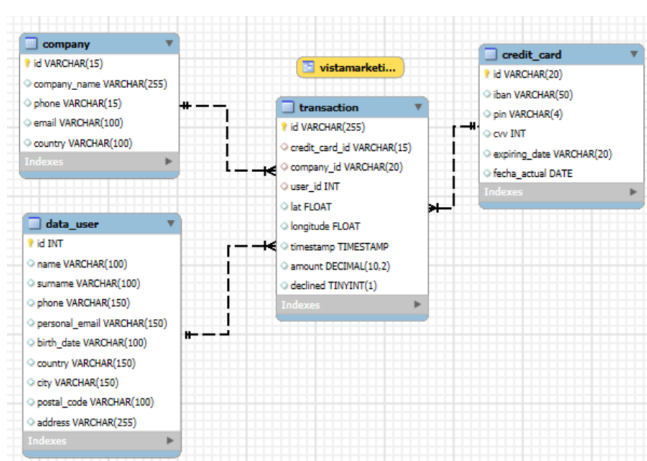


Diagrama creado

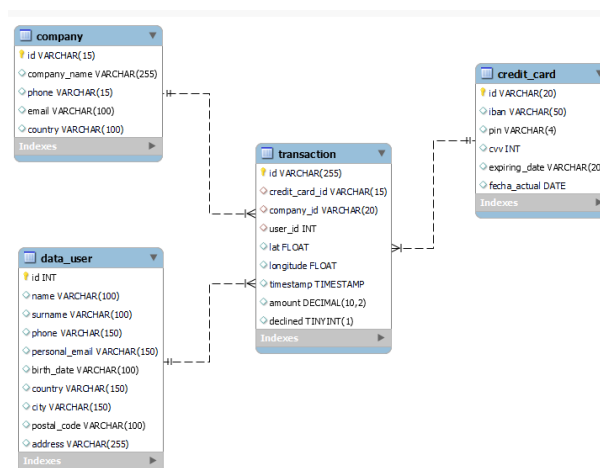
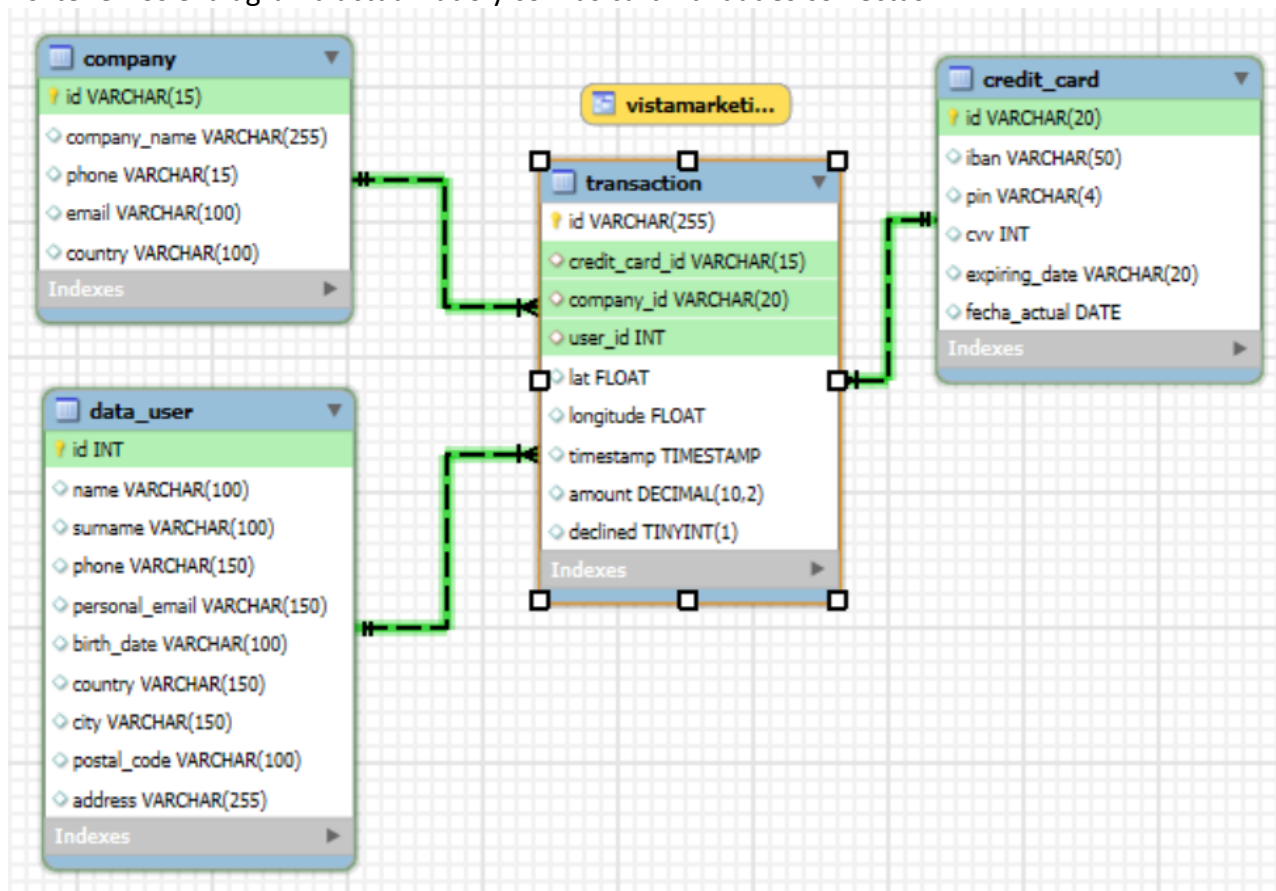


Diagrama dado para el ejercicio

Así tenemos el diagrama actualizado y con las cardinalidades correctas:



Así, al final tenemos un diagrama, donde la tabla de hechos es transaction y las de dimensiones company, credit\_card y data\_user, que fue creada nueva.

## EJERCICIO 02

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.

Asegúrate de incluir información relevante de ambas tablas y utiliza alias para renombrar columnas según sea necesario.

Muestra los resultados de la vista y ordena los resultados de manera descendente en función de la variable *ID de transaction*.

Generamos la vista utilizando **CREATE VIEW** e incluimos todas las variables solicitadas, para conectar las tablas **data\_user**, **credit\_card** y **company** con **transaction**, realizamos tres uniones (JOIN).

The screenshot shows a database management interface with a left sidebar displaying a schema tree. The 'transaction' table is selected, and its columns are listed: id, name, surname, iban, and company\_name. The main window displays the SQL code for creating the view 'InformeTecnico' and the output of the query.

```

202
203 -- Nivel 03: Ejercicio 02:
204 -- La empresa también solicita crear una vista llamada "InformeTecnico":
205 CREATE VIEW InformeTecnico AS
206 SELECT tr.id, us.name, us.surname, cr.iban, co.company_name
207 FROM transaction tr
208 JOIN data_user us
209 ON tr.user_id = us.id
210 JOIN credit_card cr
211 ON tr.credit_card_id = cr.id
212 JOIN company co
213 ON tr.company_id = co.id
214 ORDER BY 1;
215
216
217
  
```

The output window shows the execution of the query, with the following results:

#	Time	Action	Message	Duration / Fetch
1299	16:51:48	INSERT INTO data_user (id) VALUES (9999)	1 row(s) affected	0.015 sec
1300	16:52:22	ALTER TABLE transaction ADD FOREIGN KEY (user_id) REFERENCES da...	587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0	0.140 sec
1301	16:58:54	CREATE VIEW InformeTecnico AS SELECT tr.id, us.name, us.surname, cr.i...	0 row(s) affected	0.016 sec

Luego, visualizamos el contenido de la vista ejecutando **SELECT \* FROM InformeTecnico**, organizamos los resultados de forma descendente (usando DESC) según el id de la tabla **transaction**, para mostrar los valores de mayor a menor.

**SCHEMAS**

Filter objects

- sys
- tienda\_online
- transactions
  - Tables
    - company
    - credit\_card
    - data\_user
    - transaction
  - Views
    - informetecnico
    - vistamarketing
  - Stored Procedures
  - Functions

Administration Schemas

**Information**

**View: informetecnico**

**Columns:**

id	varchar(255)
name	varchar(100)
surname	varchar(100)
iban	varchar(50)
company_name	varchar(255)

Object Info Session

Query Completed

212 JOIN company co  
213 ON tr.company\_id = co.id  
214 ORDER BY 1;  
215  
216 SELECT \*  
217 FROM InformeTecnico  
218 ORDER BY id Desc;  
219

Result Grid

id	name	surname	iban	company_name
FE96CE47-8D59-381C-4E18-E3CA3D4E8FF	Kenyon	Hartman	DO26854763748537475216568689	Magna A Neque Industries
FE809ED4-2D86-55AC-C915-929516E46468	Molly	Gillam	SE2813123487163628531121	Nunc Interdum Incorporated
FD9CB0CD-8E1E-8DA1-4606-7E3A6F3A5A65	Linus	Willis	KW9485332754781757886242955643	Nunc Interdum Incorporated
FD89D51B-AE8D-77DC-E450-88083FBD3187	Hilda	Levy	LT053237077744561475	Malesuada PC
FD2E8957-4148-8EEC-E9AD-59AA7A8A6290	Hedwig	Gilbert	GE84848451582810541526	Neque Tellus Imperdiet Corp.
FCE2AB9A-271D-2BDC-9E49-8DD92A373391	Hakeem	Alford	MD1234119525145401270486	Nunc Interdum Incorporated

InformeTecnico 19 x

Output

#	Time	Action	Message	Duration / Fetch
1300	16:52:22	ALTER TABLE transaction ADD FOREIGN KEY (user_id) REFERENCES da...	587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0	0.140 sec
1301	16:58:54	CREATE VIEW InformeTecnico AS SELECT tr.id, us.name, us.surname, cr.i...	0 row(s) affected	0.016 sec
1302	17:00:19	SELECT * FROM InformeTecnico ORDER BY id Desc	587 row(s) returned	0.015 sec / 0.000 sec