

Sprint 3: Gestió de taules, índex i vistes

Tasca S3.01. Manipulació de taules



Autor

Fede Labate



P2P correcció

Damian Molini



Descripció

En aquest sprint, es simula una situació empresarial en la qual has de realitzar diverses manipulacions en les taules de la base de dades. Al seu torn, hauràs de treballar amb índexs i vistes. En aquesta activitat, continuaràs treballant amb la base de dades que conté informació d'una empresa dedicada a la venda de productes en línia. En aquesta tasca, començaràs a treballar amb informació relacionada amb targetes de crèdit.



Objectius

- Manipulació de dades.
- Treballar amb vistes i índexs



Durada

6 dies.
















Lliurament

Emmagatzema en un repositori del teu GitHub una carpeta que contingui:

- **L'arxiu .sql** que contingui tots els scripts.
- Un **PDF** que contingui una captura de pantalla del workbench on es pugui observar el script de la consulta que vas fer i el resultat obtingut per a cada exercici.

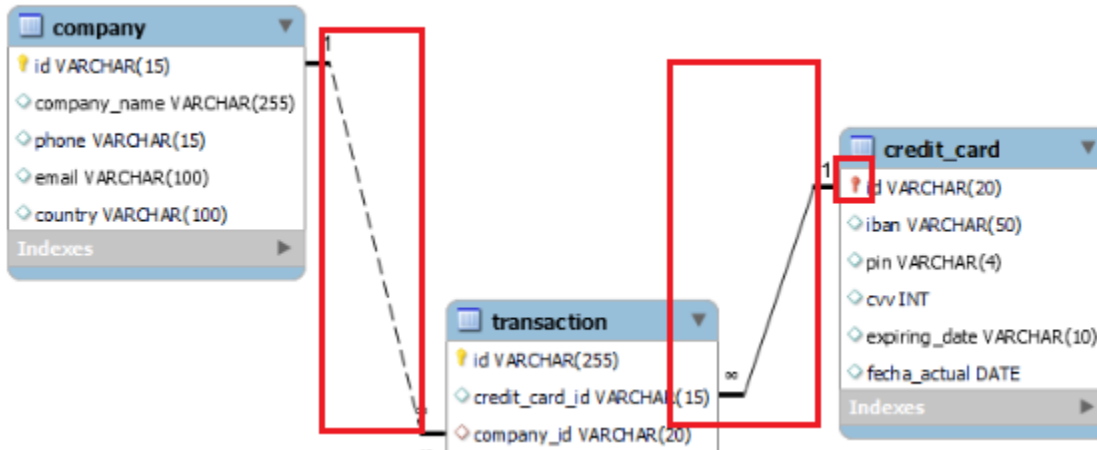
En el lliurament, col·loca el link al repositori.

Tasca S3.01. Manipulació de taules.....	1
 Autor.....	1
 P2P correcció.....	1
 Descripció.....	1
 Objectius.....	1
 Durada.....	1
 Lliurament.....	1
 Aclariments.....	3
★ Nivell 1.....	5
Exercici 1.....	5
Exercici 2.....	8
Exercici 3.....	10
Exercici 4.....	11
 He après.....	11
 He recordat.....	12
★★ Nivell 2.....	12
Exercici 1.....	12
Exercici 2.....	13
Exercici 3.....	14
 He après.....	14
★★★ Nivell 3.....	15
Exercici 1.....	15
Exercici 2.....	25
 He après.....	26
 Recursos.....	26
 Correcció.....	27

● Aclariments

Me surgieron dos dudas muy puntuales del [ejercicio 1](#) del [nivel 3](#):

1. El significado de la llave en color rojo.
2. El significado de la línea sólida versus la línea punteada



La máxima aproximación que tengo es que la **llave roja** indica que hay una advertencia o un problema con la definición de las claves.

En cuanto a la **línea sólida** y la **línea punteada**, he encontrado la siguiente información:

[mysql - Entity Relationships: Difference between solid line and dotted line - Stack Overflow](#)

SOLID LINE => IDENTIFYING relationship

Definition from mySQL docs: An identifying relationship is one where the child table (*tabla de hechos* -> *transactions*) cannot be uniquely identified without its parent. Typically this occurs where an intermediary table is created to resolve a many-to-many relationship. In such cases, the primary key is usually a composite key made up of the primary keys from the two original tables.

Example: We have an application that registers the arrival time of the employees with this model:

```
user { id_user, name, department, job }
arrival_log { id_user, arrival_time, department }
```

Each row of arrival_log requires the user_id to be specified. Without the user_id, we wouldn't know who reached the offices. The entity arrival_log is a weak one because it depends on other entities' existence (user) to work.

DOTTED LINE => NON-IDENTIFYING relationship.

Definition: A non-identifying relationship is one where the child can be identified independently of the parent.

Example:

```
flower( flower_id, flower_latin_name, flower_type_id)
flower_type( flower_type_id, name, description )
```

The relationship between flower and flower_type is non-identifying because each flower_type can be identified without having to exist in the flower table.

Entonces la conclusión a la que he llegado, es que las relaciones de la tablas *transacction* con *credit_card*, *data_user* y *company* han de ser sólidas (IDENTIFYING relationship), dado que cada transacción necesita de las otras tablas para ser más específica.

★ Nivell 1

Exercici 1

La teva tasca és dissenyar i crear una taula anomenada "credit_card" que emmagatzemi detalls crucials sobre les targetes de crèdit. La nova taula ha de ser capaç d'identificar de manera única cada targeta i establir una relació adequada amb les altres dues taules ("transaction" i "company").

Després de crear la taula serà necessari que ingressis la informació del document denominat "dades_introduir_credit". Recorda mostrar el diagrama i realitzar una breu descripció d'aquest.

Para la creación de la tabla *credit_card* partiré de la siguiente estructura:

```

7 CREATE TABLE credit_card (
8     id VARCHAR(255) PRIMARY KEY NOT NULL,
9     iban VARCHAR(255) NULL,
10    pan VARCHAR(255) NULL,
11    pin VARCHAR(255) NULL,
12    cvv VARCHAR(255) NULL,
13    expiring_date VARCHAR(255) NULL
14 );

```

Figura 1.1.1 - Instrucción de creación de la tabla *credit_card*

Ingreso los datos provistos para la tabla *credit_card*. (275 registros)

The screenshot shows a database management interface with a SQL editor and a result grid. The SQL editor contains several INSERT statements for the *credit_card* table. The result grid displays the first 10 rows of the data inserted.

id	iban	pan	pin	cvv	expiring_date
CcU-2938	TR301950312213576817638661	5424465566813633	3257	984	2022-10-30
CcU-2945	DO26854763748537475216568689	5142423821948828	9080	887	2023-08-24
CcU-2952	BG45IVQL52710525608255	4556 453 55 5287	4598	438	2021-06-29
CcU-2959	CR7242477244335841535	372461377349375	3583	667	2023-02-24
CcU-2966	CR7242477244335841535	372461377349375	3583	667	2023-02-24
CcU-2966	BG72LKTQ15627628377363	448566 886747 7265	4900	130	2024-10-29
CcU-2973	PT87806228135092429456346	544 58654 54343 384	8760	887	2025-01-30
CcU-2980	DE39241881883086277136	402400 7145845969	5075	596	2022-07-24
CcU-2987	GE89681434837748781813	3763 747687 76666	2298	797	2023-10-31
CcU-2994	BH62714428368066765294	344283273252593	7545	595	2022-02-28

The bottom of the screenshot shows the 'Output' tab with a message: "SELECT * FROM credit_card LIMIT 0, 10000" and "275 row(s) returned".

Figura 1.1.2 - Instrucción de inserción de datos en la tabla *credit_card*

Para mejorar el rendimiento y la disponibilidad de recursos, redefino los tipos de variables de la tabla *credit_card*:

```

16  ## Redefinición de tipos de variables en la tabla credit_card:
17  #id -> Cambio a VARCHAR(8)
18  ● ALTER TABLE credit_card MODIFY id VARCHAR(8) NOT NULL;
19  #iban -> Utilizo VARCHAR(32)caracteres dado que es la cantidad de los IBANs más largos del mundo (Santa Lucía & Nicaragua). Fuente: https://es.iban.com/estructura
20  ● ALTER TABLE credit_card MODIFY iban VARCHAR(32);
21  #pan -> 19 caracteres era suficiente pero coloco VARCHAR(20) por seguridad (No coloco INT por los espacios).
22  ● ALTER TABLE credit_card MODIFY pan VARCHAR(20);
23  #pin -> Número de 4 cifras -> SMALLINT: Enteros de -32768 a 32767.
24  ● ALTER TABLE credit_card MODIFY pin SMALLINT;
25  #cvv -> Número de 3 cifras -> SMALLINT: Enteros de -32768 a 32767.
26  ● ALTER TABLE credit_card MODIFY cvv SMALLINT;
27  #expiring_date -> Tipo DATE previa transformación:
28  ## Paso 1: Añadir una nueva columna de tipo DATE.
29  ● ALTER TABLE credit_card
30  ADD COLUMN expiring_date_new DATE;
31  ## Paso 2: Actualizar los valores del nuevo campo. [Xm: Mes (01-12)] | [Xd: Día del mes (01-31)] | [XY: Año con cuatro dígitos]
32  ● UPDATE credit_card
33  SET expiring_date_new = STR_TO_DATE(expiring_date, "%m/%d/%Y");
34  ## Paso 3: Eliminar el campo original (expiring_date).
35  ● ALTER TABLE credit_card
36  DROP COLUMN expiring_date;
37  ## Paso 4: Cambio el nombre de la columna nuevamente a expiring_date
38  ● ALTER TABLE credit_card CHANGE expiring_date_new expiring_date DATE;

```

Figura 1.1.3 - Instrucciones de redefinición de tipos de datos en la tabla *credit_card*

Confirmo que los tipos de datos sean correctos:

```

42  -- Visualizo el total de las columnas con sus Types correctos.
43  ● SHOW COLUMNS FROM credit_card;
44
45
46

```

Field	Type	Null	Key	Default	Extra
id	varchar(8)	NO	PRI	NULL	
iban	varchar(32)	YES		NULL	
pan	varchar(20)	YES		NULL	
pin	smallint	YES		NULL	
cvv	smallint	YES		NULL	
expiring_date	date	YES		NULL	

Figura 1.1.4 - Instrucción + Visualización final de tipos de datos en la tabla *credit_card*

Visualizo la tabla para confirmar que no haya valores truncados.

id	iban	pan	pin	cvv	expiring_date
CcU-2938	TR301950312213576817638661	5424465566813633	3257	984	2022-10-30
CcU-2945	DO26854763748537475216568689	5142423821948828	9080	887	2023-08-24
CcU-2952	BG45IVQL52710525608255	4556 453 55 5287	4598	438	2021-06-29
CcU-2959	CR7242477244335841535	372461377349375	3583	667	2023-02-24
CcU-2966	BG72LKTQ15627628377363	448566 886747 7265	4900	130	2024-10-29
CcU-2973	PT87806228135092429456346	544 58654 54343 384	8760	887	2025-01-30

Figura 1.1.5 - Visualización de datos en la tabla *credit_card*

Por último, debo vincular **credit_card_id** de la tabla *transaction* con **id** de la tabla *credit_card*. Para ello, primero formatearé el valor de **credit_card_id** al tipo VARCHAR(8) y luego crearé la relación a través de una FK:

```

39  #Vinculación de transaction.credit_card_id con credit_card.id:
40  ## Cambio el tipo de dato para que sean iguales: VARCHAR(8)
41  • ALTER TABLE transaction CHANGE credit_card_id credit_card_id VARCHAR(8) NOT NULL;
42  ## Creo la Foreign Key:
43  • ALTER TABLE transaction ADD CONSTRAINT fk_transaction_credit_card FOREIGN KEY (credit_card_id) REFERENCES credit_card(id);

```

Figura 1.1.6 - Instrucciones de vinculación de *transaction.credit_card_id* con *credit_card.id*

Confirmo que la FK creada en la tabla *transaction* esté correcta:

```

45  -- Visualizo el total de las columnas con sus Types correctos.
46  • SHOW COLUMNS FROM transaction;
47
48
49

```

Field	Type	Null	Key	Default	Extra
id	varchar(255)	NO	PRI	NULL	
credit_card_id	varchar(8)	NO	MUL	NULL	
company_id	varchar(20)	YES	MUL	NULL	
user_id	int	YES		NULL	
lat	float	YES		NULL	
longitude	float	YES		NULL	
timestamp	timestamp	YES		NULL	
amount	decimal(10,2)	YES		NULL	
declined	tinyint(1)	YES		NULL	

Figura 1.1.7 - Instrucción + Visualización final de tipos de datos en la tabla *transaction*

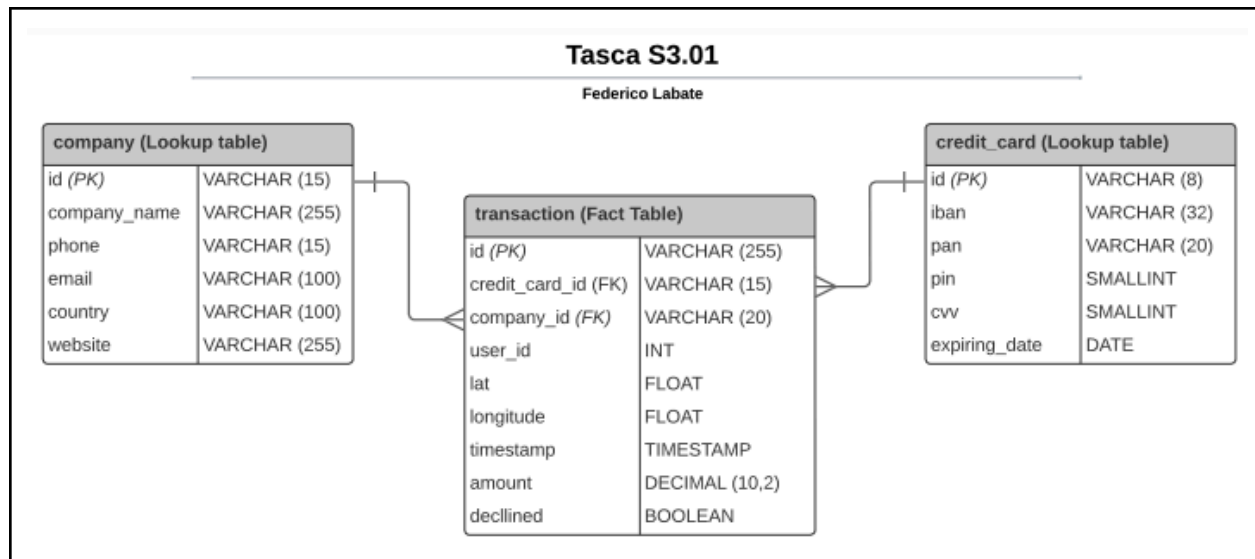


Figura 1.1.8 - Diagrama E/R

En la figura 1.1.8 podemos apreciar el diagrama de entidad relación con *transaction* como tabla de hechos. La misma contiene un campo de identificador único llamado **id**, el cual será su clave primaria y granularidad a nivel de transacción. Se relaciona con las tablas tablas de dimensiones *company* y *credit_card* a través de las claves foráneas **company_id** y **credit_card_id** respectivamente, ambas con una cardinalidad N-1. Lo que significa que habrá muchos registros en la tabla *transaction* que se relacionarán a un solo registro de las tablas *company* y *credit_card*.

Exercici 2

El departament de Recursos Humans ha identificat un error en el número de compte de l'usuari amb ID CcU-2938. La informació que ha de mostrar-se per a aquest registre és: R323456312213576817699999. Recorda mostrar que el canvi es va realitzar.

Sprint 3 Gestió de taules, índex i...

Limit to 10000 rows

```

50 -- Exercici 2
51 -- El departament de Recursos Humans ha identificat un error en el número de compte de l'usuari
52 #Consulta inicial:
53 • SELECT *
54 FROM credit_card
55 WHERE id = "CcU-2938";

```

Result Grid

	id	iban	pan	pin	cvv	expiring_date
▶	CcU-2938	TR301950312213576817638661	5424465566813633	3257	984	2022-10-30
*	NULL	NULL	NULL	NULL	NULL	NULL

Output

Action Output

#	Time	Action	Message	Duration / Fetch
✓ 356	12:35:48	SELECT * FROM credit_card WHERE id = "CcU-2938";	1 row(s) returned	0.000 sec / 0.000 sec

Figura 1.2.1 - Query + resultado de la consulta inicial.

Sprint 3 Gestió de taules, índex i...

Limit to 10000 rows

```

57 #Cambio:
58 • UPDATE credit_card set
59 iban = 'R323456312213576817699999'
60 WHERE id = "CcU-2938";
61 #Confirmación del cambio:
62 • SELECT *
63 FROM credit_card
64 WHERE id = "CcU-2938";

```

Result Grid

	id	iban	pan	pin	cvv	expiring_date
▶	CcU-2938	R323456312213576817699999	5424465566813633	3257	984	2022-10-30
*	NULL	NULL	NULL	NULL	NULL	NULL

credit_card 15 x Apply Revert

Output

Action Output

#	Time	Action	Message	Duration / Fetch
✓ 12	19:52:30	SELECT * FROM credit_card WHERE id = "CcU-2938" LI...	1 row(s) returned	0.000 sec / 0.000 sec

Figura 1.2.2 - Instrucción + Query + resultado del cambio realizado.

Exercici 3

En la taula "transaction" ingressa un nou usuari amb la següent informació:

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

The screenshot shows a database management interface with the following components:

- SQL Editor:** Contains the following SQL code:


```

66 -- Exercici 3
67 -- En la taula "transaction" ingressa un nou usuari amb la següent informació:
68 -- Id: 108B1D1D-5B23-A76C-55EF-C568E49A99DD | credit_card_id: CcU-9999 | company_id: b-9999 | user_id: 9999 | lat: 829.999 | longitude: -117.999 |
69 • SELECT * FROM company WHERE id = "b-9999";
70 #Debo crear primero los registros company.id = "b-9999" y credit_card.id="CcU-9999" por ser claves externas a la tabla transaction:
71 • INSERT INTO company (id)
72   VALUES ("b-9999");
73 • INSERT INTO credit_card (id)
74   VALUES ("CcU-9999");
75 #Creo el nuevo usuario
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 #Visualizo la nueva transacción
      
```
- Result Grid:** Displays the result of the SQL query. The table has 9 columns: id, credit_card_id, company_id, user_id, lat, longitude, timestamp, amount, and declined. The first row shows the inserted data, and the rest are NULL.

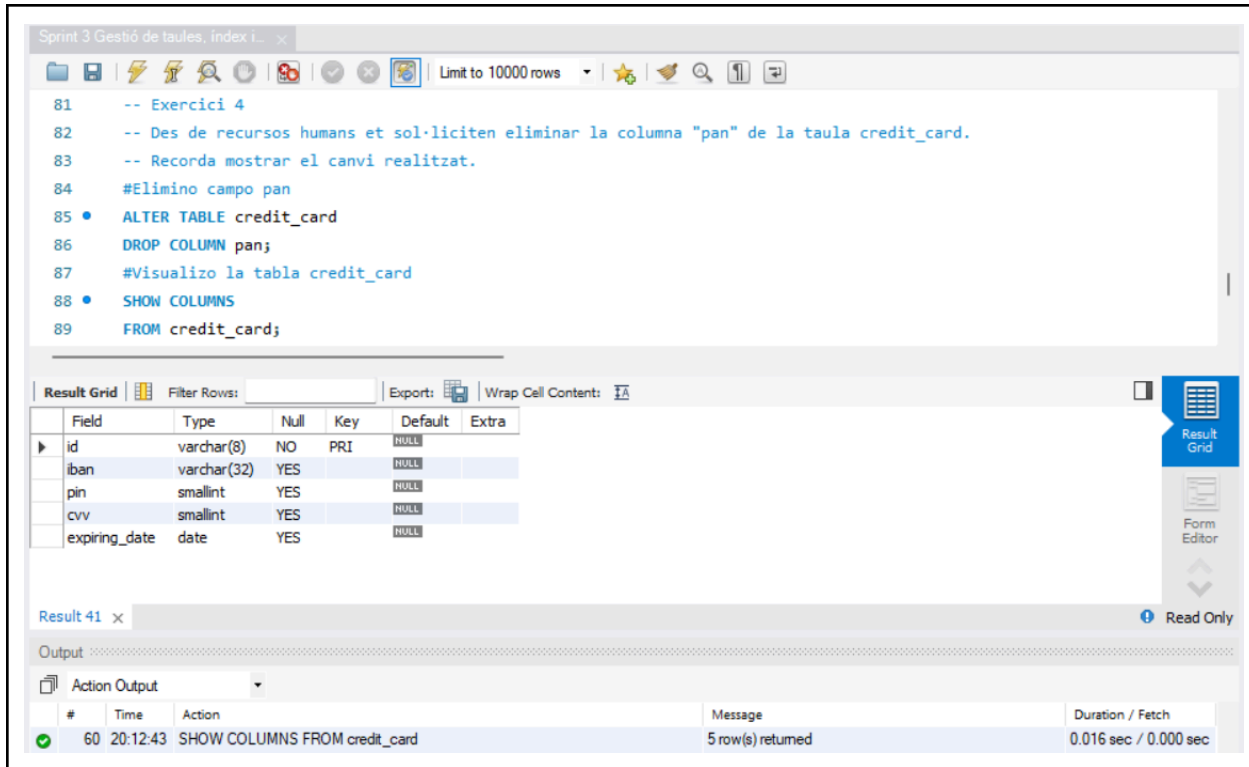
id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined
108B1D1D-5B23-A76C-55EF-C568E49A99DD	CcU-9999	b-9999	9999	829.999	-117.999	NULL	111.11	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
- Output Panel:** Shows the execution details of the SQL query.

#	Time	Action	Message	Duration / Fetch
49	20:05:07	SELECT * FROM transaction WHERE id = "108B1D1D-5B23-A76C-55EF-C568E49A99DD" LIMIT 0, 100...	1 row(s) returned	0.000 sec / 0.000 sec

Figura 1.3.1 - Instrucció + resultat.

Exercici 4

Des de recursos humans et sol·liciten eliminar la columna "pan" de la taula credit_card. Recorda mostrar el canvi realitzat.



The screenshot shows a database management interface with the following SQL commands in the editor:

```

81 -- Exercici 4
82 -- Des de recursos humans et sol·liciten eliminar la columna "pan" de la taula credit_card.
83 -- Recorda mostrar el canvi realitzat.
84 #Elimino campo pan
85 • ALTER TABLE credit_card
86   DROP COLUMN pan;
87 #Visualizo la tabla credit_card
88 • SHOW COLUMNS
89   FROM credit_card;

```

Below the editor, the 'Result Grid' shows the structure of the 'credit_card' table:

Field	Type	Null	Key	Default	Extra
id	varchar(8)	NO	PRI	NULL	
iban	varchar(32)	YES		NULL	
pin	smallint	YES		NULL	
cvv	smallint	YES		NULL	
expiring_date	date	YES		NULL	

The 'Action Output' section shows the execution of the 'SHOW COLUMNS FROM credit_card' command, which returned 5 rows in 0.016 seconds.

Figura 1.4.1 - Instrucció + resultado.

He après

- A crear una taula desde los tipos VARCHAR(255) e ir ajustando en base al contenido de los distintos campos.
- A formatear una fecha con la función **STR_TO_DATE()**. [MySQL STR_TO_DATE\(\) Function](#)
- A crear una **restricció** y por consiguiente una **Foreign Key** en dos tablas ya creadas. [SQL ADD CONSTRAINT Keyword](#)
- La diferencia entre los comandos **DDL** (*Data Definition Language*) [**CREATE, ALTER, DROP, TRUNCATE, RENAME**] conocidos también como **instrucciones** y los **DML** (*Data Manipulation Language*) [**SELECT, INSERT, UPDATE, DELETE**] más popularmente conocido como **queries**.
- Las distintas visualizaciones de tablas con el comando **SHOW**.
- Al ingresar nuevos registros en una tabla, si alguno de los campos es una clave externa a otra tabla, el valor indicado en el insert deberá contener uno existente en el campo identificador de la tabla destino.
- Obviamente no puedo ingresar un registro con datos NULL si el mismo campo no los permite 😊

He recordat

- A generar un diagrama E/R a través de la herramienta [Reverse Engineer](#).

★★ Nivell 2

Exercici 1

Elimina de la taula transaction el registre amb ID 02C6201E-D90A-1859-B4EE-88D2986D3B02 de la base de dades.

The screenshot shows a database management interface with a SQL editor and a results pane. The SQL editor contains the following code:

```

91  -- ##### Nivell 2 #####
92  -- Exercici 1
93  -- Elimina de la taula transaction el registre amb ID 02C6201E-D90A-1859-B4EE-88D2986D3B02 de la base de dades.
94  • DELETE
95  FROM transaction
96  WHERE id = "02C6201E-D90A-1859-B4EE-88D2986D3B02";
97  #Visualizo el id eliminado.
98  • SELECT * FROM transaction WHERE id = "02C6201E-D90A-1859-B4EE-88D2986D3B02";
  
```

The results pane shows a table with the following columns: id, credit_card_id, company_id, user_id, lat, longitude, timestamp, amount, and declined. The table is currently empty, with all cells showing NULL.

The output pane shows the execution results of the SQL queries:

#	Time	Action	Message	Duration / Fetch
102	09:27:38	DELETE FROM transaction WHERE id = "02C6201E-D90A-1859-B4EE-88D2986D3B02";	1 row(s) affected	0.000 sec
103	09:27:40	SELECT * FROM transaction WHERE id = "02C6201E-D90A-1859-B4EE-88D2986D3B02";	0 row(s) returned	0.000 sec / 0.000 sec

Figura 2.1.1 - Query + resultado.

Exercici 2

La secció de màrqueting desitja tenir accés a informació específica per a realitzar anàlisi i estratègies efectives. S'ha sol·licitat crear una **vista** que proporcioni detalls clau sobre les companyies i les seves transaccions. Serà necessària que creïs una vista anomenada **VistaMarketing** que contingui la següent informació:
Nom de la companyia. Telèfon de contacte. País de residència. Mitjana de compra realitzat per cada companyia. Presenta la vista creada, ordenant les dades de major a menor mitjana de compra.



Figura 2.2.1 - Instrucció de creació de vista VistaMarketing.

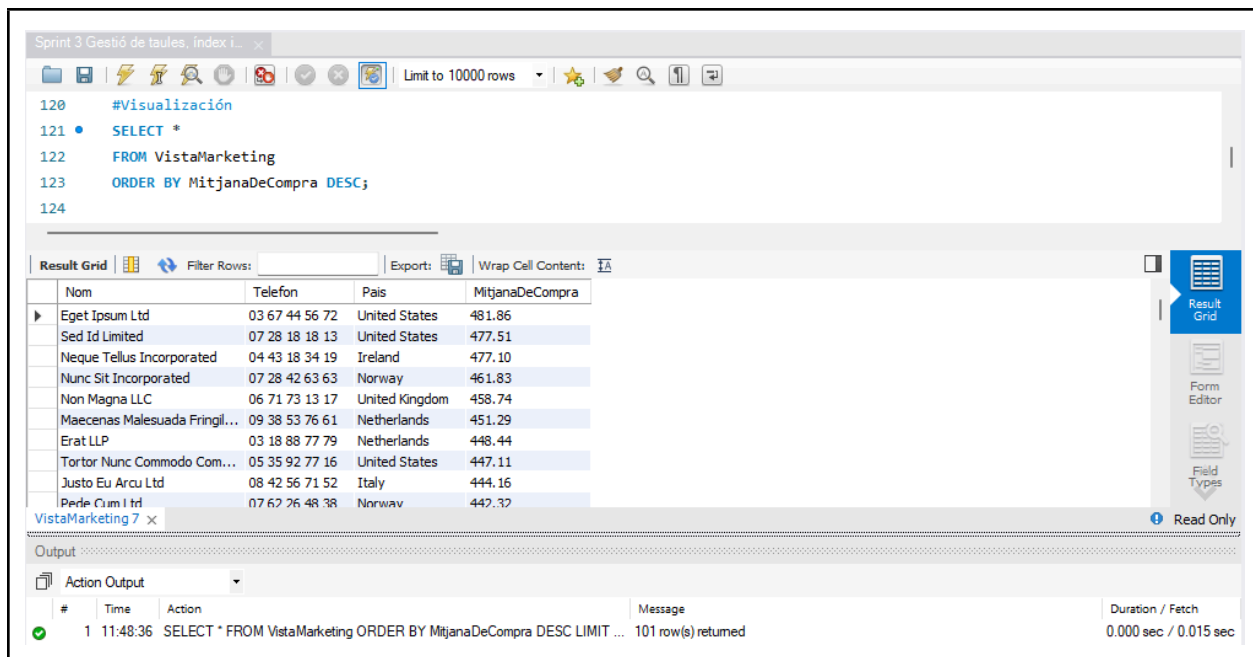


Figura 2.2.2 - Query + resultado.

Exercici 3

Filtra la vista VistaMarketing per a mostrar només les companyies que tenen el seu país de residència en "Germany"

The screenshot shows a SQL query editor with the following query:

```

121 -- Exercici 3
122 -- Filtra la vista VistaMarketing per a mostrar només les companyies que tenen el seu país de residència en "Germany"
123 • SELECT *
124 FROM VistaMarketing
125 WHERE País = "Germany";

```

The result grid displays the following data:

Nom de la companyia	Telèfon de contacte	País	MitjanaDeCompra
Nunc Interdum Incorporated	05 18 15 48 13	Germany	242.95
Ac Fermentum Incorporated	06 85 56 52 33	Germany	293.57
Rutrum Non Inc.	02 66 31 61 09	Germany	266.90
Ac Industries	09 34 65 40 60	Germany	396.15
Convallis In Incorporated	06 66 57 29 50	Germany	60.99
Auctor Mauris Corp.	05 62 87 14 41	Germany	308.99
Augue Foundation	06 88 43 15 63	Germany	15.05
Aliquam PC	01 45 73 52 16	Germany	280.34

The bottom section shows the output of the query:

```

Output
Action Output
# Time Action Message Duration / Fetch
1 13:46:24 SELECT * FROM VistaMarketing WHERE País = "Germany" LIMIT 0, 10... 8 row(s) returned 0.000 sec / 0.000 sec

```

Figura 2.3.1 - Query + resultado.

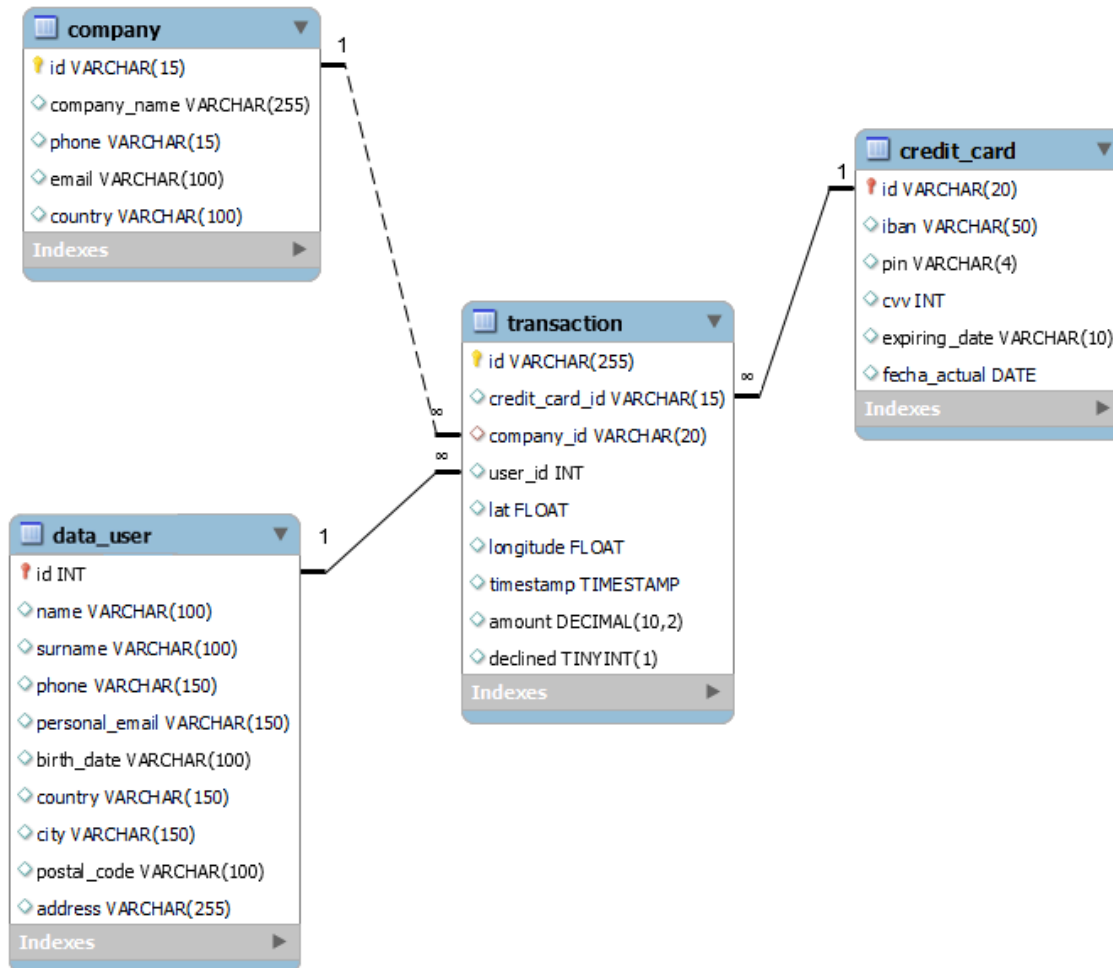
He après

- A eliminar un registro con el comando **DELETE**. [SQL DELETE Statement](#)
- A crear una vista con el comando **CREATE VIEW** nombreLista **AS**. [SQL CREATE VIEW, REPLACE VIEW, DROP VIEW Statements](#)
- A eliminar una vista con el comando **DROP VIEW** nombreLista.
- Que los alias deben ser de una sola palabra si luego se quiere llamar a ellos. Por eso la necesidad de usar CamelCase o "_". Los "["]" no se toman como continuos. Ejemplo: Quería utilizar el alias **MitjanaDeCompra[€]** pero el mismo luego no lo reconocía a la hora de hacer el **ORDER BY MitjanaDeCompra[€] DESC**, por lo que tuve que reducirlo solamente a "MitjanaDeCompra"
- A agrupar con GROUP BY siempre por **IDs** únicos (ejemplo: Nombre de las empresas)

☆☆☆ Nivell 3

Exercici 1

La setmana vinent tindràs una nova reunió amb els gerents de màrqueting. Un company del teu equip va realitzar modificacions en la base de dades, però no recorda com les va realitzar. Et demana que l'ajudis a deixar els comandos executats per a obtenir el següent diagrama:



Recordatori

En aquesta activitat, és necessari que descriguis el "pas a pas" de les tasques realitzades. És important realitzar descripcions senzilles, simples i fàcils de comprendre. Per a realitzar aquesta activitat hauràs de treballar amb els arxius denominats "estructura_dades_user" i "dades_introduir_user".

- 1) Inserto estructura_datoss_user.sql y datos_introducir_user (1).sql
 - a) Open SQL Scrip... (Ctrl+Mayús+O) > estructura_datoss_user.sql & datos_introducir_user (1).sql
- 2) Visualizo el diagrama E/R actual con el Reverse Enginer (Ctrl + R)
- 3) Procedo a hacer los cambios marcados en rojo:

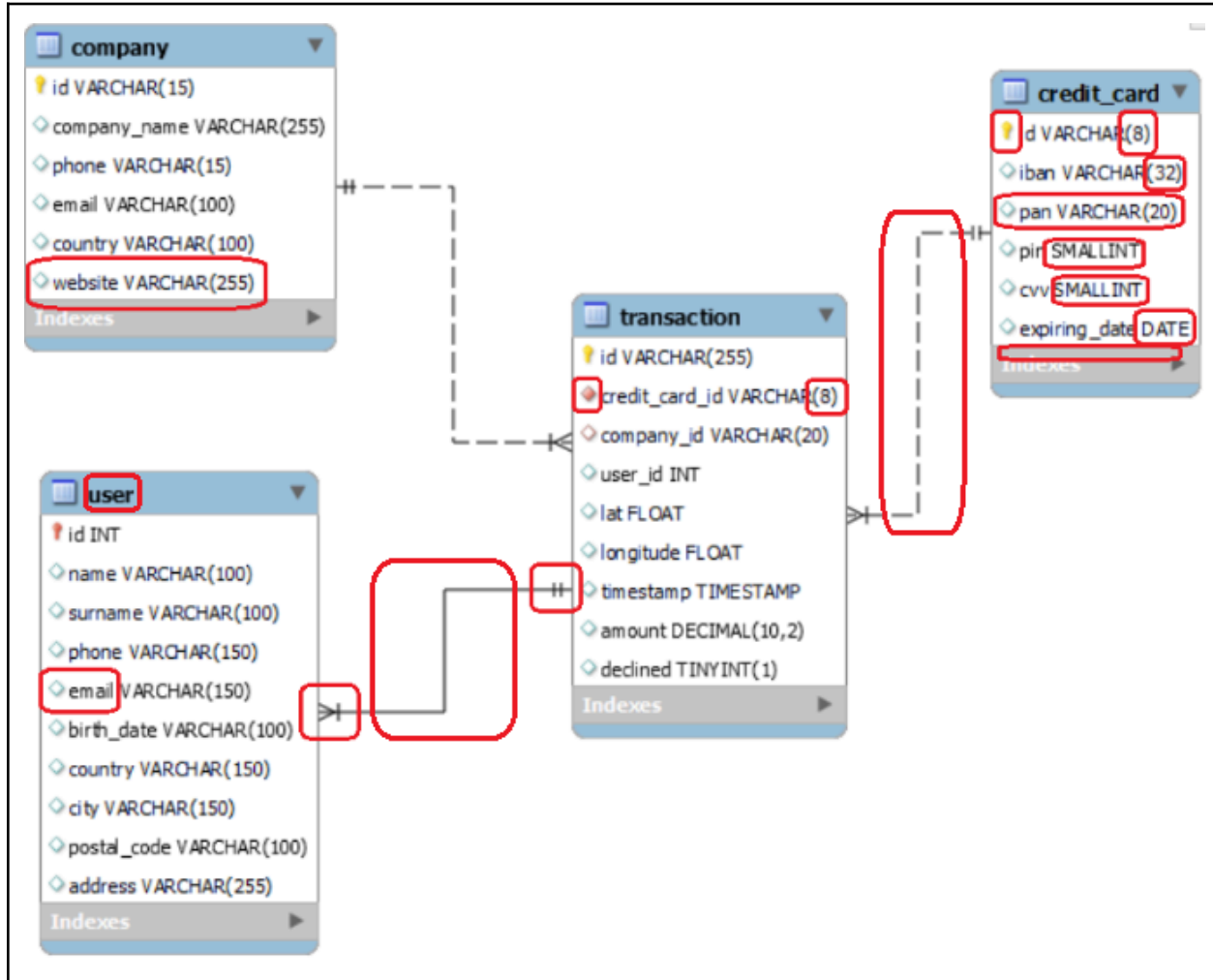


Figura 3.1.1 - Diagrama ER a modificar.

- 4) Tabla **company**:
a) Eliminar el campo *website*

The screenshot shows a database management tool interface. The SQL editor contains the following commands:

```

137  ##Tabla company:
138  #Eliminar campo "website"
139  • ALTER TABLE company DROP COLUMN website;
140  #Visualización
141  • SHOW COLUMNS
142  FROM company;
    
```

The Result Grid shows the structure of the 'company' table:

Field	Type	Null	Key	Default	Extra
id	varchar(15)	NO	PRI	NUL	
company_name	varchar(255)	YES		NUL	
phone	varchar(15)	YES		NUL	
email	varchar(100)	YES		NUL	
country	varchar(100)	YES		NUL	

The Output section shows the execution results:

#	Time	Action	Message	Duration / Fetch
1	16:01:48	ALTER TABLE company DROP COLUMN website	0 row(s) affected Records: 0 Duplica...	0.063 sec
2	16:02:03	SHOW COLUMNS FROM company	5 row(s) returned	0.000 sec / 0.000 sec

Figura 3.1.2 - Instrucción + resultado tabla *company*.

- 5) Tabla **data_user**:
a) Renombrar la tabla *user* por *data_user*

The screenshot shows a database management tool interface. The SQL editor contains the following commands:

```

144  ##Tabla data_user:
145  #Renombrar el campo email por personal_email
146  • RENAME TABLE user to data_user;
147  #Visualización
148  • SHOW tables;
149
    
```

The Result Grid shows the list of tables in the database:

Tables_in_transactions
company
credit_card
data_user
transaction

The Output section shows the execution results:

#	Time	Action	Message	Duration / Fetch
1	16:16:12	RENAME TABLE user to data_user	0 row(s) affected	0.032 sec
2	16:16:14	SHOW tables	4 row(s) returned	0.000 sec / 0.000 sec

Figura 3.1.3 - Instrucción + resultado tabla *data_user*.

b) Renombrar el campo email por personal_email

Sprint 3 Gestió de taules, índex i...

Limit to 10000 rows

```

150  ##Tabla data_user:
151  #Renombrar el campo email por personal_email
152  • ALTER TABLE data_user CHANGE email personal_email VARCHAR(150);
153  #Visualizació
154  • SHOW COLUMNS
155  FROM data_user;

```

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

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	
name	varchar(100)	YES		NULL	
surname	varchar(100)	YES		NULL	
phone	varchar(150)	YES		NULL	
personal_email	varchar(150)	YES		NULL	
birth_date	varchar(100)	YES		NULL	
country	varchar(150)	YES		NULL	
city	varchar(150)	YES		NULL	
postal_code	varchar(100)	YES		NULL	
address	varchar(255)	YES		NULL	

Result 9 x | Read Only

Output:

Action Output

#	Time	Action	Message	Duration / Fetch
✓ 1	16:25:28	ALTER TABLE data_user CHANGE email personal_email VARCH...	0 row(s) affected Records: 0 Duplicat...	0.016 sec
✓ 2	16:25:30	SHOW COLUMNS FROM data_user	10 row(s) returned	0.015 sec / 0.000 sec

Figura 3.1.4 - Instrucció + resultat taula data_user.

c) Renombrar el campo email por personal_email

Sprint 3 Gestió de taules, índex i...

Limit to 10000 rows

```

150  ##Tabla data_user:
151  #Renombrar el campo email por personal_email
152  • ALTER TABLE data_user CHANGE email personal_email VARCHAR(150);
153  #Visualizació
154  • SHOW COLUMNS
155  FROM data_user;

```

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

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	
name	varchar(100)	YES		NULL	
surname	varchar(100)	YES		NULL	
phone	varchar(150)	YES		NULL	
personal_email	varchar(150)	YES		NULL	
birth_date	varchar(100)	YES		NULL	
country	varchar(150)	YES		NULL	
city	varchar(150)	YES		NULL	
postal_code	varchar(100)	YES		NULL	
address	varchar(255)	YES		NULL	

Result 9 x | Read Only

Output:

Action Output

#	Time	Action	Message	Duration / Fetch
1	16:25:28	ALTER TABLE data_user CHANGE email personal_email VARCH...	0 row(s) affected Records: 0 Duplicat...	0.016 sec
2	16:25:30	SHOW COLUMNS FROM data_user	10 row(s) returned	0.015 sec / 0.000 sec

Figura 3.1.4 - Instrucció + resultat taula data_user.

- d) Invertir la relación de la FK *user.id* de la tabla **transaction**.
Para ello debo averiguar el nombre de la FK existente, eliminarla y luego crear una nueva desde **transaction** hacia **data_user**

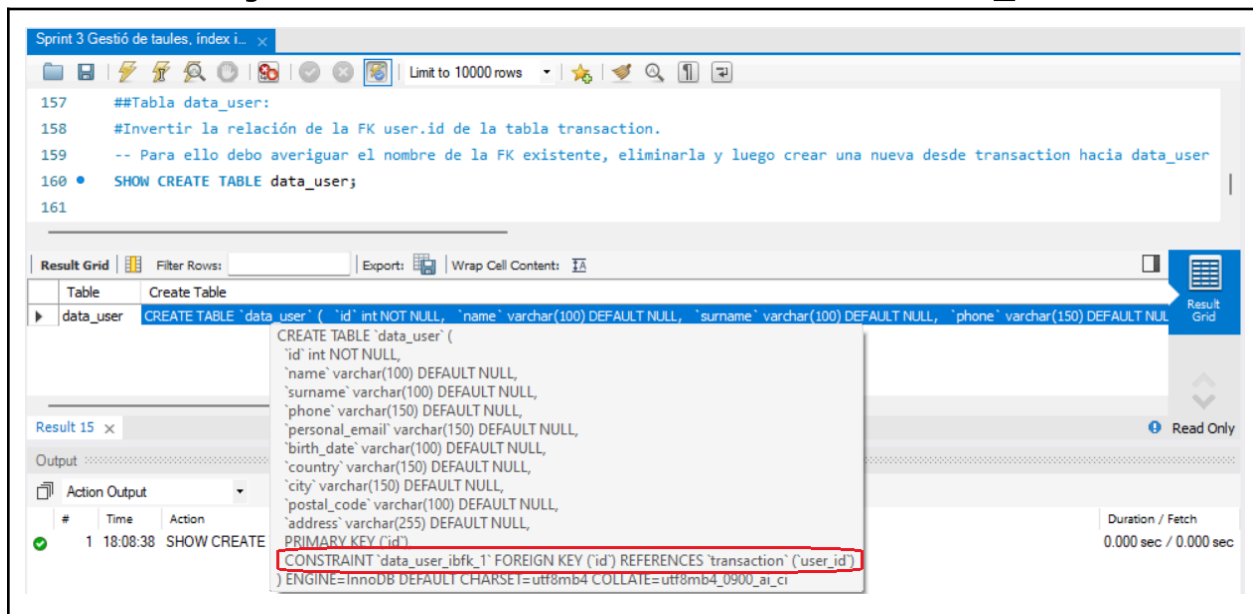


Figura 3.1.5 - Instrucción + resultado nombre de la FK existente.

- e) Eliminar la FK *data_user_ibfk_1*. Luego crearé la nueva FK desde la tabla **transaction**

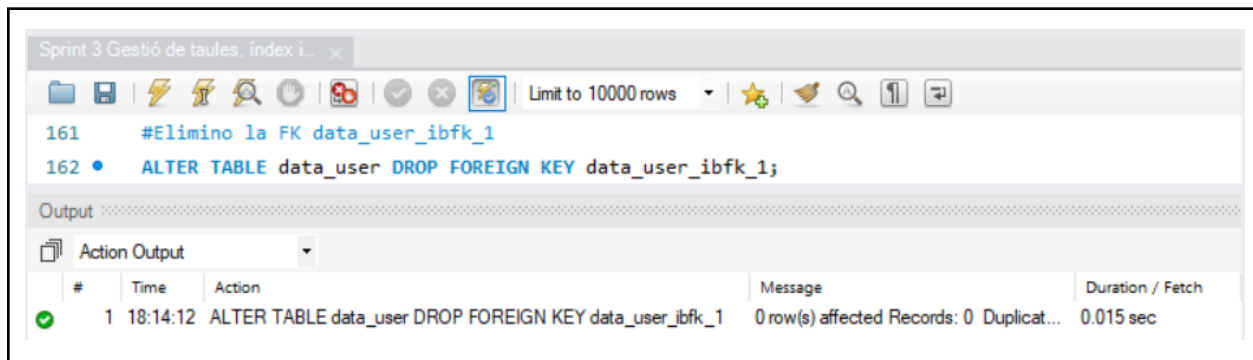


Figura 3.1.6 - Instrucción + resultado de la eliminación de la FK *data_user_ibfk_1*

6) Tabla **transaction**:

- a) Cambiar *credit_card_id* de NOT NULL a que permita valores nulos y que el tipo de dato sea VARCHAR(15)

The screenshot shows a database management tool interface. The SQL editor contains the following commands:

```

164 ##Tabla transaction:
165 #Cambiar credit_card_id de NOT NULL a que permita valores nulos y que el tipo de dato sea VARCHAR(15)
166 • ALTER TABLE transaction CHANGE credit_card_id credit_card_id VARCHAR (15);
167 #Visualización
168 • SHOW COLUMNS
169 FROM transaction;
  
```

The Result Grid shows the structure of the 'transaction' table:

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	YES	MUL	NULL	
lat	float	YES		NULL	
longitude	float	YES		NULL	
timestamp	timestamp	YES		NULL	
amount	decimal(10,2)	YES		NULL	
declined	tinyint(1)	YES		NULL	

The Output section shows the execution results:

#	Time	Action	Message	Duration / Fetch
1	09:38:30	ALTER TABLE transaction CHANGE credit_card_id credit_card_id V...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.015 sec
2	09:38:31	SHOW COLUMNS FROM transaction	9 row(s) returned	0.000 sec / 0.000 sec

Figura 3.1.7 - Instrucción + resultado campo *credit_card_id*

b) Elimino la Foreign Key *fk_transaction_credit_card*

The screenshot shows the SQL editor with the following command:

```

171 #Elimino la Foreign Key fk_transaction_credit_card
172 • ALTER TABLE transaction DROP FOREIGN KEY fk_transaction_credit_card;
  
```

The Output section shows the execution result:

#	Time	Action	Message	Duration / Fetch
1	18:34:23	ALTER TABLE transaction DROP FOREIGN KEY fk_transaction_c...	0 row(s) affected Records: 0 Duplicat...	0.032 sec

Figura 3.1.8 - Instrucción + resultado de la eliminación de la FK *fk_transaction_credit_card*

7) Tabla **credit_card**:

- a) Cambiar el tipo de los siguientes campos:
- id* a VARCHAR(20)
 - iban* a VARCHAR(50)

- iii) *pin* a VARCHAR(4)
- iv) *cvv* a INT
- v) *expiring_date* a VARCHAR (10)
- vi) Eliminar el campo *pan*.
- vii) Agregar el campo *fecha_actual* con tipo DATE

Sprint 3 Gestió de taules, index i...

Limit to 10000 rows

```

174  ##Tabla credit_card:
175  #Cambiar el tipo de los siguientes campos:
176  -- id a VARCHAR(20) | iban a VARCHAR(50) | pin a VARCHAR(4) | cvv a INT | expiring_date a VARCHAR (10).
177  • ALTER TABLE credit_card MODIFY COLUMN id VARCHAR(20);
178  • ALTER TABLE credit_card MODIFY COLUMN iban VARCHAR(50);
179  • ALTER TABLE credit_card MODIFY COLUMN pin VARCHAR(4);
180  • ALTER TABLE credit_card MODIFY COLUMN cvv INT;
181  • ALTER TABLE credit_card MODIFY COLUMN expiring_date VARCHAR(10);
182  -- Eliminar el campo pan.
183  • ALTER TABLE credit_card DROP COLUMN pan;
184  -- Agrego el campo fecha_actual con tipo DATE
185  • ALTER TABLE credit_card ADD fecha_actual DATE;
186  #Visualización:
187  • SHOW COLUMNS
188  FROM credit_card;
  
```

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

Field	Type	Null	Key	Default	Extra
id	varchar(20)	NO	PRI	NULL	
iban	varchar(50)	YES		NULL	
pin	varchar(4)	YES		NULL	
cvv	int	YES		NULL	
expiring_date	varchar(10)	YES		NULL	
fecha_actual	date	YES		NULL	

Result 1 x | Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	08:10:03	ALTER TABLE credit_card MODIFY COLUMN id VARCHAR(20)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.031 sec
2	08:31:21	ALTER TABLE credit_card MODIFY COLUMN iban VARCHAR(50)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.015 sec
3	08:31:23	ALTER TABLE credit_card MODIFY COLUMN pin VARCHAR(4)	276 row(s) affected Records: 276 Duplicates: 0 Warnings: 0	0.078 sec

Figura 3.1.9 - Instrucción + resultado de la tabla *credit_card*

8) Foreign Keys

- Agregar FK: *transaction* -> *credit_card* (N a 1)
- Agregar FK: *transaction* -> *data_user* (N a 1)
- Cambiar nombre *transaction_ibfk_1* a *fk_transaction_company* (a fin de mantener la coherencia)

```

193  #Foreign Keys
194  -- Agregar FK: transaction -> credit_card (N a 1)
195  ALTER TABLE transaction
196  ADD CONSTRAINT fk_transaction_credit_card
197  FOREIGN KEY (credit_card_id)
198  REFERENCES credit_card (id);
199
200  -- Agregar FK: transaction -> data_user (N a 1)
201  ALTER TABLE transaction
202  ADD CONSTRAINT fk_transaction_data_user
203  FOREIGN KEY (user_id)
204  REFERENCES data_user (id);
205
206  -- Cambiar nombre transaction_ibfk_1 a fk_transaction_company
207  #Eliminar transaction_ibfk_1
208  ALTER TABLE transaction
209  DROP FOREIGN KEY transaction_ibfk_1;
210  #Crearla nuevamente con el nombre fk_transaction_company
211  ALTER TABLE transaction
212  ADD CONSTRAINT fk_transaction_company
213  FOREIGN KEY (company_id)
214  REFERENCES company (id);
  
```

#	Time	Message	Duration / Fetch
1	21:36:54	587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0	0.125 sec
2	21:36:56	587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0	0.125 sec
3	21:36:58	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.125 sec
4	21:37:01	587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0	0.125 sec

Figura 3.1.10 - Instrucciones de creación de foreign keys

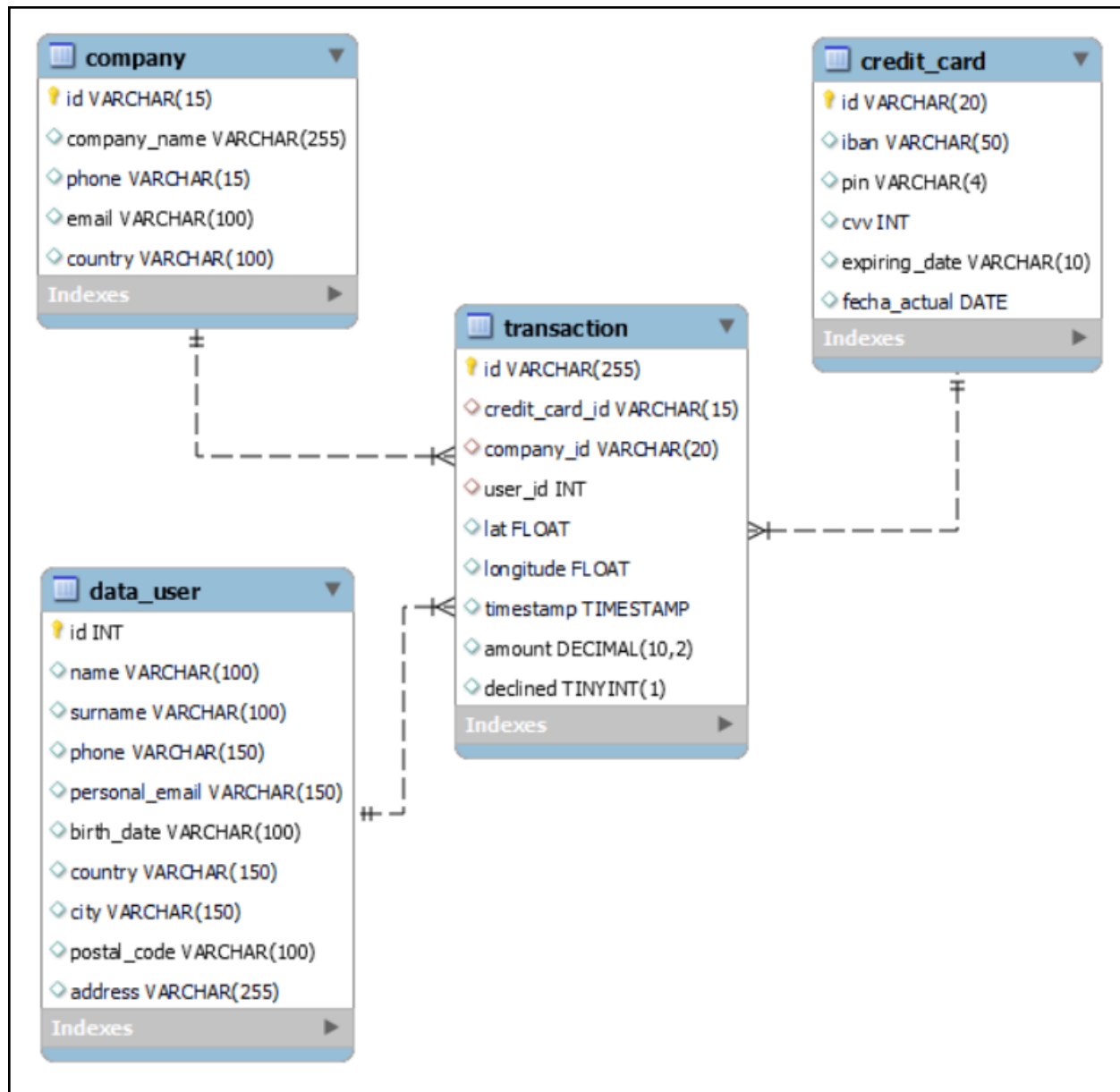


Figura 3.1.11 - Diagrama ER final

Cómo debería quedar el diagrama ER final según mi hipótesis de [Aclariments](#):

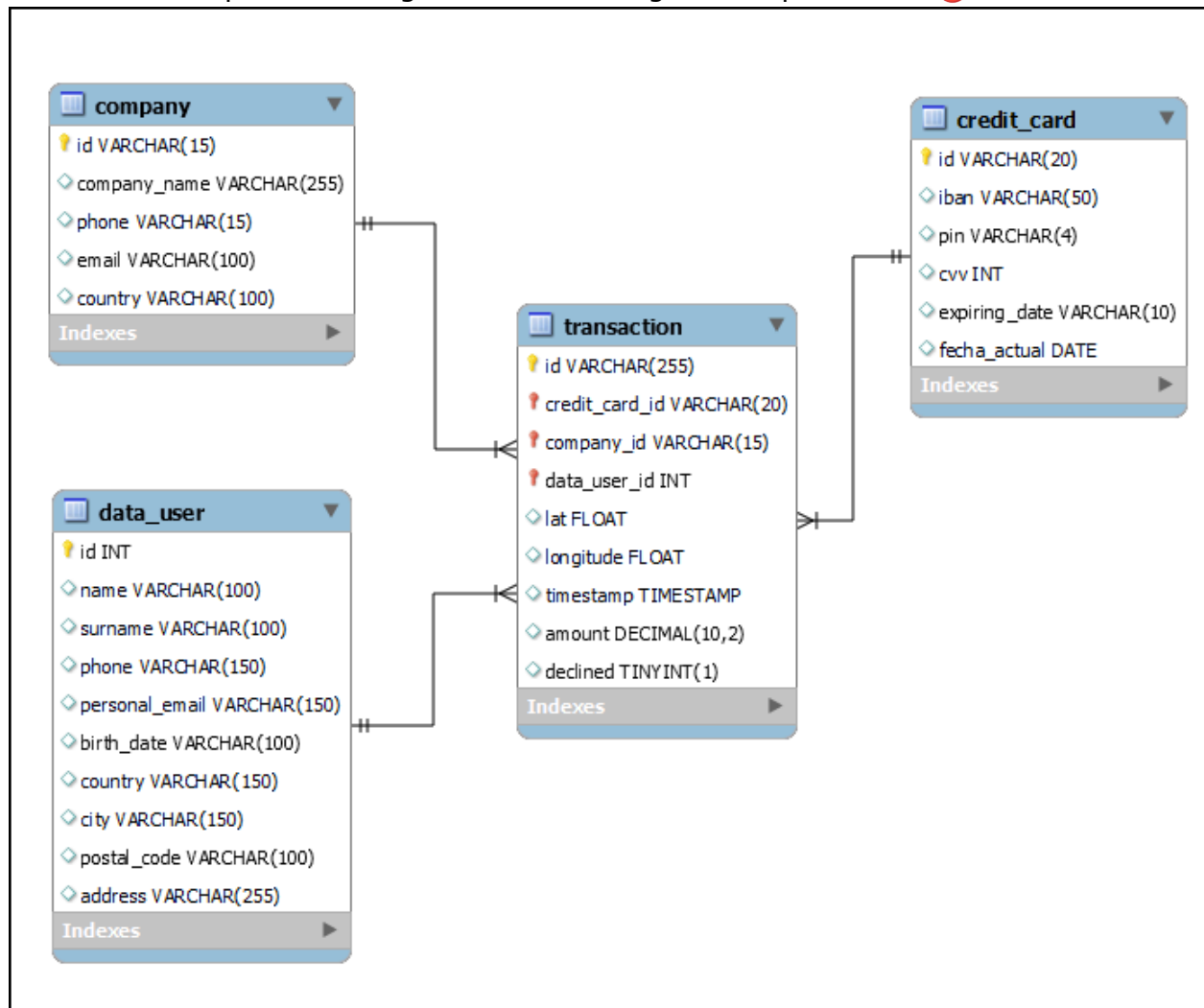


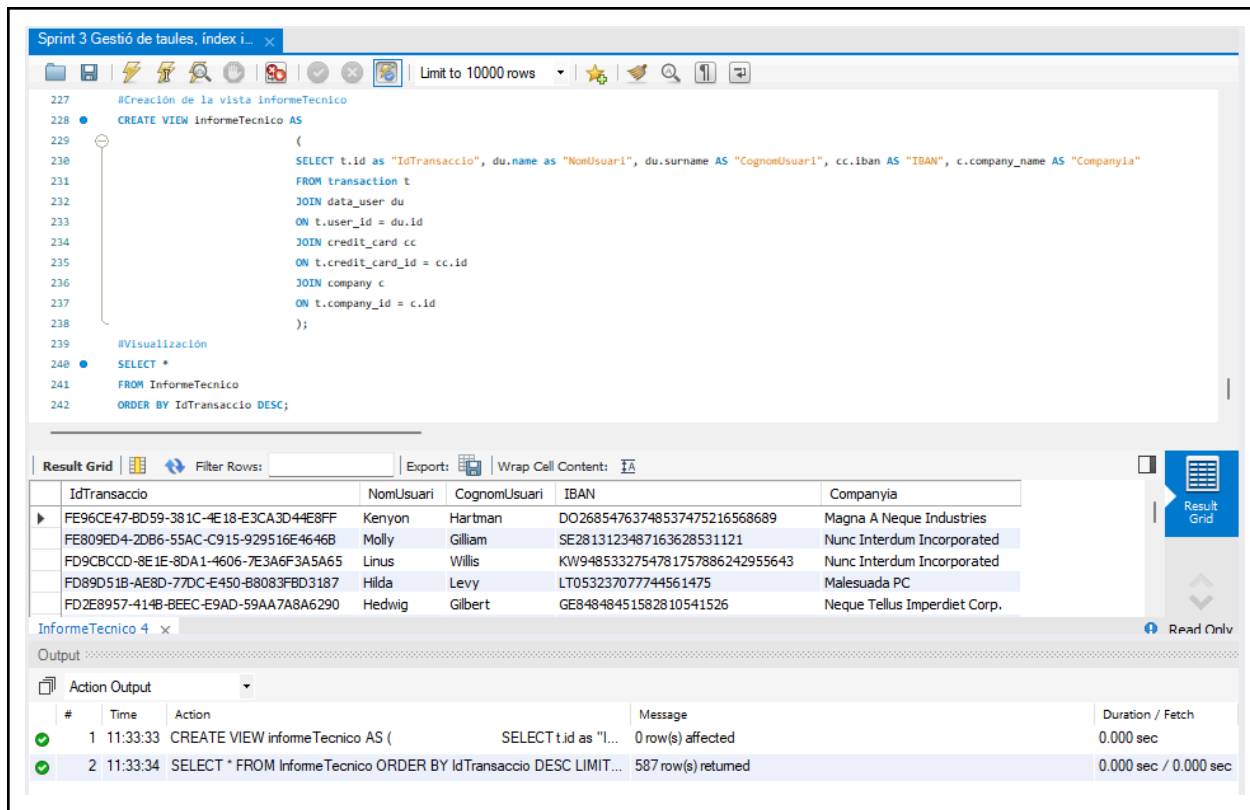
Figura 3.1.12 - Diagrama ER final

Exercici 2

L'empresa també et sol·licita crear una **vista** anomenada "InformeTecnico" que contingui la següent informació:

- ID de la transacció
- Nom de l'usuari/ària
- Cognom de l'usuari/ària
- IBAN de la targeta de crèdit usada.
- Nom de la companyia de la transacció realitzada.
- Assegura't d'incloure informació rellevant de totes dues taules i utilitza àlies per a canviar de nom columnes segons sigui necessari.

Mostra els resultats de la vista, ordena els resultats de manera descendent en funció de la variable ID de transaction.



```

227 #Creación de la vista InformeTecnico
228 CREATE VIEW InformeTecnico AS
229 (
230     SELECT t.id as "IdTransaccio", du.name as "NomUsuari", du.surname AS "CognomUsuari", cc.iban AS "IBAN", c.company_name AS "Companyia"
231     FROM transaction t
232     JOIN data_user du
233     ON t.user_id = du.id
234     JOIN credit_card cc
235     ON t.credit_card_id = cc.id
236     JOIN company c
237     ON t.company_id = c.id
238 );
239
240 #Visualización
241 SELECT *
242 FROM InformeTecnico
243 ORDER BY IdTransaccio DESC;
    
```

IdTransaccio	NomUsuari	CognomUsuari	IBAN	Companyia
FE96CE47-8D59-381C-4E18-E3CA3D44E8FF	Kenyon	Hartman	DO26854763748537475216568689	Magna A Neque Industries
FE809ED4-2DB6-55AC-C915-929516E46468	Molly	Gilliam	SE2813123487163628531121	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.

InformeTecnico 4 x

Output

#	Time	Action	Message	Duration / Fetch
1	11:33:33	CREATE VIEW InformeTecnico AS (SELECT t.id as "I...	0 row(s) affected 0.000 sec
2	11:33:34	SELECT * FROM InformeTecnico ORDER BY IdTransaccio DESC LIMIT...	587 row(s) returned	0.000 sec / 0.000 sec

Figura 3.2.1 - Instrucción + Query + resultado.

He après

- La diferencia entre **CHANGE COLUMN**, **MODIFY COLUMN** y **RENAME COLUMN**:
 - CHANGE COLUMN**: Requiere tanto el nombre actual como el nuevo nombre de la columna
 - MODIFY COLUMN**: Solo necesita el nombre actual de la columna. (Se utiliza solo cuando quiero modificar el tipo de dato sin alterar el nombre de la misma).
 - RENAME COLUMN**: No utilizado en este Sprint. Como su nombre indica, solo cambia el nombre:
 - ALTER TABLE** table_name
RENAME COLUMN old_name **TO** new_name;

Recursos

[datos introducir credit.sql](#)
[datos introducir user \(1\).sql](#)
[estructura datos user.sql](#)

Correcció