

# Tasca S3.01. Manipulació de taules

## Timan(Nakta) Samipour

Compañero para revisión peer-to-peer : Federico Urbina

### 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\_introducir\_credit". Recorda mostrar el diagrama i realitzar una breu descripció d'aquest.

The screenshot shows the MySQL Workbench interface with two main panes. The left pane displays the database schema, showing the 'transactions' schema containing the 'credit\_card' table. The right pane shows the SQL editor and results.

**SQL Editor (Top):**

```
1 ##Sprint 3
2 ## Nivell 1
3 ## Exercici 1
4 ## La teva tasca és dissenyar i crear una taula anomenada "credit_card" que emmagatzemi detalls crucials sobre les targetes de crèdit.
5 ## La nova taula ha de ser capaç d'identificar de manera única cada targeta i establir una relació adequada amb les altres dues taules
6 ## ("transaction" i "company"). Després de crear la taula serà necessari que ingressis la informació del document denominat
7 ## "dades_introducir_credit". Recorda mostrar el diagrama i realitzar una breu descripció d'aquest.
8
9 • CREATE TABLE IF NOT EXISTS credit_card (
10    id          VARCHAR(20)  NOT NULL,
11    iban        VARCHAR(34)  NOT NULL,
12    pan         CHAR(19)    NOT NULL,
13    pin         CHAR(4)     NOT NULL,
14    cvv         CHAR(4)     NOT NULL,
15    expiring_date VARCHAR(10) NOT NULL,
16    PRIMARY KEY (id)
17 );
18
```

**Action Output:**

Time	Action	Response	Duration / Fetch T
16:20:10	CREATE TABLE IF NOT EXISTS credit_card ( id VARCHAR(20) NOT NULL, iban VARCHAR(34) ... )	0 row(s) affected	0.011 sec

**SQL Editor (Bottom):**

```
19 • SHOW CREATE TABLE transaction;
20
```

**Form Editor:**

Table: transaction

Create Table:

```
CREATE TABLE `transaction` (
  `id` varchar(255) NOT NULL,
  `credit_card_id` varchar(15) DEFAULT NULL,
  `company_id` varchar(20) DEFAULT NULL,
  `user_id` int DEFAULT NULL,
  `lat` float DEFAULT NULL,
  `longitude` float DEFAULT NULL,
  `timestamp` timestamp NULL DEFAULT NULL,
  `amount` decimal(10,2) DEFAULT NULL,
  `declined` tinyint(1) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `company_id` (`company_id`),
  CONSTRAINT `transaction_ibfk_1` FOREIGN KEY (`company_id`) REFERENCES `company` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

**Action Output:**

Time	Action	Response	Duration / Fetch Time
11:01:33	SHOW CREATE TABLE transaction	1 row(s) returned	0.0011 sec / 0.00001...

```

-- 
23 •  SELECT COUNT(*) AS total_credit_cards
24   FROM credit_card;
25

100%  ◊  18:24 | Result Grid Filter Rows: Search Export: 
total_credit_car...
5000

25
26 •  SELECT COUNT(*) AS invalid.refs
27   FROM transaction t
28   LEFT JOIN credit_card c ON c.id = t.credit_card_id
29   WHERE t.credit_card_id IS NOT NULL
30   AND c.id IS NULL;|
31

100%  ◊  20:30 | Result Grid Filter Rows: Search Export: 
inv...
0

-- 
33  -- Se crea la clave foránea entre transaction y credit_card una vez verificada
34  -- la integridad de los datos, garantizando la coherencia del modelo relacional.
35 •  ALTER TABLE transaction
36   ADD CONSTRAINT fk_transaction_credit_card
37   FOREIGN KEY (credit_card_id) REFERENCES credit_card(id);
38
39 •  SHOW CREATE TABLE transaction;
40
41 •  ALTER TABLE transaction
42   MODIFY credit_card_id VARCHAR(20);
43

100%  ◊  31:39 | Form Editor Navigate: 1/1
Form Editor
transaction
Table:
Create Table:
Result 13
Action Output
Catalog
Tables
Views
Routine Groups
Transactions
transaction
credit_card
company

```

The screenshot shows a database management interface with several panes:

- Code Editor:** Displays SQL queries for selecting total credit cards (5000), counting invalid references (0), and altering the transaction table to add a foreign key constraint and modify the credit\_card\_id column.
- Result Grid:** Shows the results of the first query: a single row with 'total\_credit\_cards' set to 5000.
- Result Grid:** Shows the results of the second query: a single row with 'invalid.refs' set to 0.
- Form Editor:** Shows the 'transaction' table definition, including columns like id, credit\_card\_id, company\_id, user\_id, lat, longitude, timestamp, amount, and declined, along with constraints and engine information.
- Action Output:** Shows the creation of the 'transaction' table with a timestamp of 11:20:41, one row returned, and a duration of 0.00082 sec / 0.00000.
- Diagram:** A diagram showing relationships between three tables: transaction, credit\_card, and company. The transaction table has a 1:n relationship with credit\_card and a 1:n relationship with company. The credit\_card table has a 1:n relationship with transaction. The company table has a 1:n relationship with transaction.

## Solución:

La solución implementada consiste en la creación de la tabla `credit_card`, definiendo el campo `id` como clave primaria para garantizar la identificación única de cada tarjeta de crédito. Los tipos de datos de los campos (`iban`, `pan`, `pin`, `cvv` y `expiring_date`) se han seleccionado de forma coherente con el formato y las restricciones del conjunto de datos proporcionado.

La relación entre `credit_card` y `transaction` se establece mediante el campo

`transaction.credit_card_id`, permitiendo asociar cada transacción con la tarjeta utilizada.

Previamente a la creación de la clave foránea, se ha verificado la integridad de los datos existentes, comprobando que no existen referencias inválidas entre ambas tablas.

La relación con la tabla `company` se mantiene de forma indirecta a través de la tabla `transaction`, preservando un diseño normalizado y evitando redundancias innecesarias en el modelo. Una vez validada la coherencia de los datos, se crea la clave foránea a nivel físico entre `transaction` y `credit_card`, garantizando la integridad referencial del sistema.

Finalmente, se cargan los datos correspondientes a las tarjetas de crédito y se presenta el diagrama entidad-relación que refleja correctamente las relaciones establecidas entre las tablas del modelo.

## Exercici 2

El departament de Recursos Humans ha identificat un error en el número de compte associat a la targeta de crèdit amb ID CcU-2938. La informació que ha de mostrar-se per a aquest registre és: TR323456312213576817699999. Recorda mostrar que el canvi es va realitzar.

```
43
44  ##Exercici 2
45  ##El departament de Recursos Humans ha identificat un error en el número de compte associat a la targeta de crèdit amb ID CcU-2938.
46  ## La informació que ha de mostrar-se per a aquest registre és: TR323456312213576817699999. Recorda mostrar que el canvi es va realitzar.
47 •  SELECT id, iban
48  FROM credit_card
49  WHERE id = 'CcU-2938';
50
```

Result Grid   Filter Rows: Search   Edit: Export/Import:   Resu  
Grid

id	iban
CcU-2938	TR301950312213576817638661
HULL	HULL

Time Action Response Duration / Fetch Time

1 11:19:41 SELECT COUNT(DISTINCT country) AS total\_countries FROM company AS c JOIN transaction ON t.comp... 1 row(s) returned 0.070 sec / 0.000005...

```
44  ##Exercici 2
45  ##El departament de Recursos Humans ha identificat un error en el número de compte associat a la targeta de crèdit amb ID CcU-2938.
46  ## La informació que ha de mostrar-se per a aquest registre és: TR323456312213576817699999. Recorda mostrar que el canvi es va realitzar.
47 •  UPDATE credit_card
48  SET iban = 'TR323456312213576817699999'
49  WHERE id = 'CcU-2938';
50
51 •  SELECT id, iban
52  FROM credit_card
53  WHERE id = 'CcU-2938';
54
```

Action Output   Response   Duration / Fetch Time

Time Action Response Duration / Fetch Time

1 13:56:23 SELECT id, iban FROM credit\_card WHERE id = 'CcU-2938' 1 row(s) returned 0.00076 sec / 0.0000...
2 13:57:38 UPDATE credit\_card SET iban = 'TR323456312213576817699999' WHERE id = 'CcU-2938' 1 row(s) affected Rows matched: 1 Changed: 1 Warni... 0.0023 sec

```

50
51 •  SELECT id, iban
52   FROM credit_card
53   WHERE id = 'CcU-2938';
54

100%  23:53

Result Grid  Filter Rows: Search Edit: Export/Import:
id iban
CcU-2938 TR323456312213576817699999
NULL NULL

2 13:57:38 UPDATE credit_card SET iban = 'TR323456312213576817699999' WHERE id = 'CcU-2938'
3 13:57:54 SELECT id, iban FROM credit_card WHERE id = 'CcU-2938'

1 row(s) affected Rows matched: 1 Changed: 1 Warni... 0.0023 sec
1 row(s) returned 0.00078 sec / 0.0000...

```

### Solución:

Para resolver el problema, se realiza una actualización directa sobre la tabla **credit\_card**, modificando el campo `iban` del registro identificado por su clave primaria (`id = 'CcU-2938'`). Esta operación permite corregir el dato erróneo sin afectar a la estructura del modelo ni a las relaciones existentes con otras tablas.

Tras la actualización, se ejecuta una consulta de verificación para confirmar que el valor del IBAN ha sido modificado correctamente, garantizando la coherencia y trazabilidad del cambio solicitado por el departamento de Recursos Humanos.

## Exercici 3

Utilitzant només subconsultes (sense utilitzar JOIN):

- o Mostra totes les transaccions realitzades per

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

```

57
58 • INSERT INTO transaction (
59   id,
60   credit_card_id,
61   company_id,
62   user_id,
63   lat,
64   longitude,
65   amount,
66   declined
67 )
68 VALUES (
69   '108B1D1D-5B23-A76C-55EF-C568E49A99DD',
70   'CcU-9999',
71   'b-9999',
72   '9999',
73   '829.999',
74   '-117.999',
75   '111.11',
76   '0'
77 );
78
79

```

Action Output	Time	Action	Response	Duration / Fetch Time
1 14:44:42	INSERT INTO transaction ( id, credit_card_id, company_id, user_id, lat, longitude, amount,...		Error Code: 1452. Cannot add or update a child row: a...	0.0018 sec

```
79 • SELECT *
80   FROM company
81   WHERE id = 'b-9999';
82
83 • INSERT INTO company (id, company_name)
84   VALUES ('b-9999', 'Test Company');
85
86
87
100% 35:84

Action Output
Time Action Response Duration / Fetch Time
① 1 14:44:42 INSERT INTO transaction ( id, credit_card_id, company_id, user_id, lat, longitude, amount,... Error Code: 1452. Cannot add or update a child row: a... 0.0018 sec
② 2 14:44:53 SELECT * FROM company WHERE id = 'b-9999' 0 row(s) returned 0.00092 sec / 0.0000...
③ 3 14:50:10 INSERT INTO company (id, company_name) VALUES ('b-9999', 'Test Company') 1 row(s) affected 0.0014 sec

91 • SELECT id
92   FROM credit_card
93   WHERE id = "CcU-9999";
94
95 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date)
96   VALUES ('CcU-9999', 'TR323456312213576817699999', '1111222233334444',
97           '1234', '123', '2030-12-31');
98
99 #check
100 • SELECT id, iban
101   FROM credit_card
102   WHERE id = 'CcU-9999';
103

100% 1:103

Result Grid Filter Rows: Search Edit: Export/Import: Result Grid Form Editor
id iban
CcU-9999 TR323456312213576817699999
NULL NULL

credit_card 7
Action Output
Time Action Response Duration / Fetch Time
④ 1 15:14:03 SELECT id FROM credit_card WHERE id = 'CcU-9999' 0 row(s) returned 0.0010 sec / 0.0000...
⑤ 2 16:06:09 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-9999', 'TR3234563122135768... 1 row(s) affected 0.0022 sec
⑥ 3 16:07:04 SELECT id, iban FROM credit_card WHERE id = 'CcU-9999' 1 row(s) returned 0.00078 sec / 0.0000...

80 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date)
81   VALUES ('CcU-9999', 'TR323456312213576817699999', '1111222233334444',
82           '1234', '123', '2030-12-31');
83
84 #check
85 • SELECT id, iban
86   FROM credit_card
87   WHERE id = 'CcU-9999';
88
89 #control
90 • SELECT
91   id, credit_card_id, company_id, user_id, lat, longitude, amount, declined
92   FROM transaction
93   WHERE id = '108B1D1D-5B23-A76C-55EF-C568E49A99DD';
94

100% 51:93

Result Grid Filter Rows: Search Edit: Export/Import: Result Grid Form Editor
id credit_card_id company_id user_id lat longitude amount declined
108B1D1D-5B23-A76C-55EF-C568E49A99DD CcU-9999 b-9999 9999 829.999 -117.999 111.11 0
NULL NULL NULL NULL NULL NULL NULL NULL NULL

transaction 8
Action Output
Time Action Response Duration / Fetch Time
⑦ 1 16:15:08 SELECT id, credit_card_id, company_id, user_id, lat, longitude, amount, declined FROM transaction WHERE ... 1 row(s) returned 0.0014 sec / 0.0000...
```

### Solución:

Para insertar la nueva transacción es necesario respetar las restricciones de integridad referencial definidas en el modelo de datos. La tabla `transaction` contiene claves foráneas que referencian a las tablas `company` y `credit_card`, por lo que los valores utilizados en los campos `company_id` y `credit_card_id` deben existir previamente en sus tablas correspondientes.

En primer lugar, se comprobó la existencia del registro con `company_id = 'b-9999'` en la tabla `company`. Al no existir inicialmente, se insertó dicho registro y se verificó su creación mediante una consulta de comprobación.

A continuación, se comprobó la existencia del registro con `credit_card_id = 'CcU-9999'` en la tabla `credit_card`. Al no existir, se insertó el registro correspondiente y se validó su inserción.

Una vez garantizada la existencia de ambos registros referenciados, se procedió a insertar la nueva transacción en la tabla `transaction` con los valores indicados en el enunciado. Finalmente, se ejecutó una consulta de control filtrada por el identificador de

## Exercici 4

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

```

98 • ALTER TABLE credit_card
99   DROP COLUMN pan;
100
101 • DESCRIBE credit_card;
102
100%  22:101 |
```

**Result Grid** Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
id	varchar(20)	NO	PRI	HULL	
iban	varchar(34)	NO		HULL	
pin	char(4)	NO		HULL	
cvv	char(4)	NO		HULL	
expiring_date	varchar(10)	NO		HULL	

**Action Output**

Time	Action	Response	Duration / Fetch Time
10:35:04	ALTER TABLE credit_card DROP COLUMN pan	0 row(s) affected Records: 0 Duplicates: 0 Warnings...	0.022 sec
10:35:25	DESCRIBE credit_card	5 row(s) returned	0.0021 sec / 0.00003...

### Solución:

Para eliminar la columna solicitada se utiliza la instrucción `ALTER TABLE`, que permite modificar la estructura de una tabla existente. En este caso, se elimina la columna `pan` de la tabla `credit_card`, ya que ya no es necesaria para los requerimientos actuales del sistema.

Tras ejecutar la modificación estructural, se realiza una consulta de comprobación (`DESCRIBE`) sobre la tabla `credit_card` para verificar que la columna ha sido eliminada correctamente y dejar evidencia del cambio realizado.

## Nivell 2

### Exercici 1

Elimina de la taula `transaction` el registre amb ID 000447FE-B650-4DCF-85DE-C7ED0EE1CAAD de la base de dades.

```

105 • SELECT *
106   FROM transaction
107   WHERE id = '000447FE-B650-4DCF-85DE-C7ED0EE1CAAD';
108
109 • DELETE
110   FROM transaction
111   WHERE id = '000447FE-B650-4DCF-85DE-C7ED0EE1CAAD';
112
100%  51:107 |
```

**Result Grid** Filter Rows: Search Edit: Export/Import:

id	credit_card...	company_id	user_id	lat	longitude	timestamp	amount	declined
HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

**Action Output**

Time	Action	Response	Duration / Fetch Time
10:56:58	SELECT * FROM transaction WHERE id = '000447FE-B650-4DCF-85DE-C7ED0EE1CAAD'	1 row(s) returned	0.0048 sec / 0.00001...
10:57:09	DELETE FROM transaction WHERE id = '000447FE-B650-4DCF-85DE-C7ED0EE1CAAD'	1 row(s) affected	0.0036 sec
10:57:16	SELECT * FROM transaction WHERE id = '000447FE-B650-4DCF-85DE-C7ED0EE1CAAD'	0 row(s) returned	0.00069 sec / 0.000...

## Solución:

Para eliminar el registro indicado se utiliza la instrucción `DELETE`, que permite borrar filas específicas de una tabla.

Antes de realizar la eliminación, se ejecuta una consulta `SELECT` filtrada por el identificador de la transacción para verificar que el registro existe y confirmar que es el correcto.

Posteriormente, se ejecuta el `DELETE` utilizando la condición `WHERE id = '000447FE-B650-4DCF-85DE-C7ED0EE1CAAD'`, garantizando que únicamente se elimine el registro solicitado.

Finalmente, se realiza una consulta de comprobación para evidenciar que la transacción

## Exercici 2

La secció de màrqueting desitja tenir accés a informació específica per a realitzar ànalisi 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.

```
118 • CREATE VIEW VistaMarketing AS
119   SELECT c.company_name      AS nombre_compania,
120         c.phone              AS telefono,
121         c.country            AS pais,
122         AVG(t.amount)        AS media_compra
123   FROM company
124   JOIN transaction          AS t ON t.company_id = c.id
125   WHERE t.declined = 0
126   GROUP BY c.id, c.company_name, c.phone, c.country;
127
128 • SELECT *
129   FROM VistaMarketing
130   ORDER BY media_compra DESC;
131
```

The screenshot shows a database interface with the following details:

- Code Area:** Displays the SQL code for creating the `VistaMarketing` view and executing a query against it.
- Result Grid:** Shows the results of the query, which lists company names, phone numbers, countries, and average purchase amounts. The data includes:

nombre_compania	telefono	pais	media_compra
Ac Fermentum Incorporated	06 85 56 52 33	Germany	284.911333
Pretium Neque Corp.	07 77 48 55 23	Australia	275.578064
Urna Convallis Associates	06 01 24 77 04	United States	273.566925
At Associates	09 56 61 10 65	New Zealand	272.736985
Metus Vitae Associates	08 25 44 40 66	Australia	270.051876
Aliquet Diam Limited	02 76 61 47 46	United States	269.291145
Nec Luctus LLC	02 14 71 75 73	Norway	268.604837
Neque Tellus Incorporated	04 43 18 34 19	Ireland	267.563349
Cras Consulting	07 50 10 85 63	Belgium	267.382797

- Action Output:** Shows the log of actions taken, including the creation of the view and the execution of the query.

## Solución:

Para dar respuesta a la solicitud del departamento de marketing, se crea una vista utilizando la instrucción `CREATE VIEW`, encapsulando la lógica de agregación y relación entre tablas en una estructura reutilizable.

La vista se construye a partir de un `JOIN` entre las tablas `company` y `transaction`, lo que permite asociar cada empresa con sus transacciones. Se utiliza la función de agregación `AVG(t.amount)` para calcular la media de compra por compañía, aplicando `GROUP BY` a nivel de empresa.

Además, se filtran únicamente las transacciones válidas mediante la condición `t.declined = 0`, garantizando que el cálculo de la media se base solo en compras efectivas.

Una vez creada la vista, se consulta mediante un `SELECT` ordenando los resultados de forma descendente según la media de compra, cumpliendo con los requisitos indicados

## Exercici 3

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

```
134 •  SELECT *
135   FROM VistaMarketing
136   WHERE pais = 'Germany'
137   ORDER BY media_compra DESC;
```

Result Grid | Filter Rows: Search Export:

nombre_compania	telefono	pais	media_compra
Ac Fermentum Incorporated	06 85 56 52 33	Germany	284.911333
Nunc Interdum Incorporated	05 18 15 48 13	Germany	259.319156
Convallis In Incorporated	06 66 57 29 50	Germany	257.693651
Ac Industries	09 34 65 40 60	Germany	255.169777
Rutrum Non Inc.	02 66 31 61 09	Germany	255.137959
Auctor Mauris Corp.	05 62 87 14 41	Germany	254.675099
Augue Foundation	06 88 43 15 63	Germany	253.564644
Aliquam PC	01 45 73 52 16	Germany	252.958601

VistaMarketing 5 |

Action Output | Time Action Response Duration / Fetch Time

1 11:51:01 SELECT \* FROM VistaMarketing WHERE pais = 'Germany' ORDER BY media\_compra DESC 8 row(s) returned 0.076 sec / 0.000011...

### Solució:

Dado que la información requerida ya se encuentra encapsulada en la vista VistaMarketing, no es necesario redefinir la lógica de agregación ni realizar nuevos JOIN.

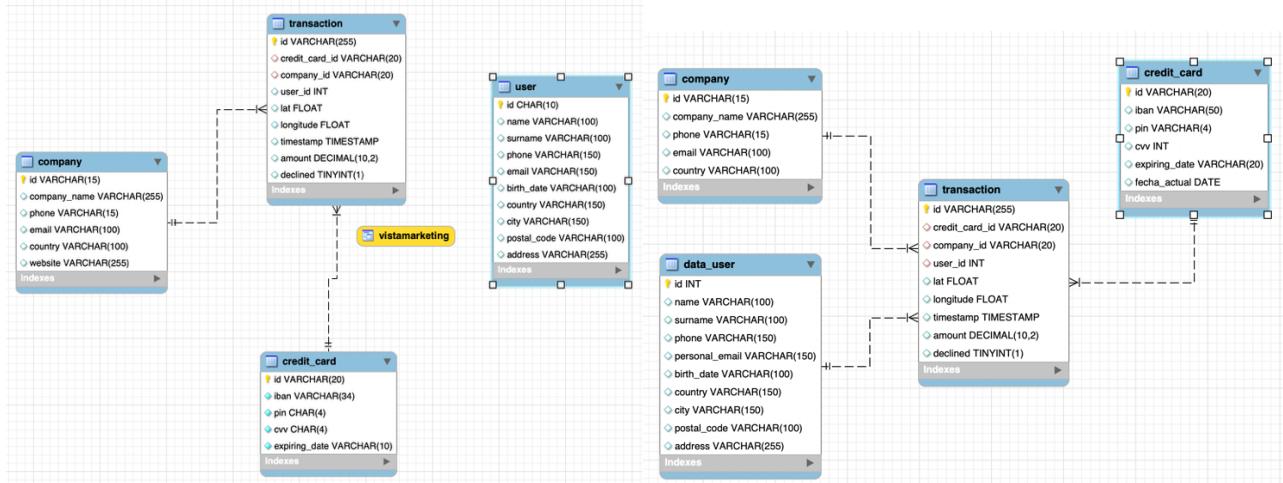
Para cumplir con el requisito, se realiza una consulta SELECT sobre la vista existente, aplicando una condición WHERE sobre el campo pais para filtrar únicamente las compañías con país de residencia "Germany".

De este modo, se reutiliza la vista previamente creada y se obtiene un resultado filtrado.

## Nivell 3

### Exercici 1

Presenta el nom, telèfon, país, data i amount, d'aquelles empreses que van realitzar transaccions amb un valor comprès entre 350 i 400 euros i en alguna d'aquestes dates: 29 d'abril del 2015, 20 de juliol del 2018 i 13 de març del 2024. Ordena els resultats de major a menor quantitat.



Antes



Después

```

145 •  SELECT kcu.CONSTRAINT_NAME, kcu.COLUMN_NAME,
146      kcu.REFERENCED_TABLE_NAME, kcu.REFERENCED_COLUMN_NAME
147  FROM information_schema.KEY_COLUMN_USAGE AS kcu
148 WHERE kcu.TABLE_SCHEMA = DATABASE()
149   AND kcu.TABLE_NAME = 'transaction'
150   AND kcu.REFERENCED_TABLE_NAME IS NOT NULL;
151
100% 45:150 |
```

**Result Grid** Filter Rows: Search Export:

CONSTRAINT_NAME	COLUMN_NAME	REFERENCED_TABLE_NAME	REFERENCED_COLUMN_NAME
fk_transaction_credit_card	credit_card_id	credit_card	id
transaction_ibfk_1	company_id	company	id

KEY\_COLUMN\_USAGE 8

**Action Output**

Time	Action	Response	Duration / Fetch Time
13:18:21	SHOW TABLES	5 row(s) returned	0.0020 sec / 0.00001...
13:21:17	SELECT kcu.CONSTRAINT_NAME, kcu.COLUMN_NAME, kcu.REFERENCED_TABLE_NAME, kcu.REFERENC...	2 row(s) returned	0.0035 sec / 0.00001...

Read Only

### Antes

```

144
145 •  SELECT kcu.CONSTRAINT_NAME, kcu.COLUMN_NAME,
146      kcu.REFERENCED_TABLE_NAME, kcu.REFERENCED_COLUMN_NAME
147  FROM information_schema.KEY_COLUMN_USAGE AS kcu
148 WHERE kcu.TABLE_SCHEMA = DATABASE()
149   AND kcu.TABLE_NAME = 'transaction'
150   AND kcu.REFERENCED_TABLE_NAME IS NOT NULL;
151
152 •  ALTER TABLE user
153   RENAME TO data_user;
154
100% 45:150 |
```

**Result Grid** Filter Rows: Search Export:

CONSTRAINT_NAME	COLUMN_NAME	REFERENCED_TABLE_NAME	REFERENCED_COLUMN_NAME
fk_transaction_credit_card	credit_card_id	credit_card	id
fk_transaction_user	user_id	data_user	id
transaction_ibfk_1	company_id	company	id

KEY\_COLUMN\_USAGE 24

**Action Output**

Time	Action	Response	Duration / Fetch Time
15:19:22	INSERT INTO data_user (id) VALUES ('9999')	1 row(s) affected	0.0028 sec
15:20:14	SELECT COUNT(*) AS orphan_user_ids FROM transaction AS t LEFT JOIN data_user AS du ON du.id = t.u...	1 row(s) returned	0.099 sec / 0.000015...
15:20:34	ALTER TABLE transaction ADD CONSTRAINT fk_transaction_user FOREIGN KEY (user_id) REFERENCES data_...	100000 row(s) affected Records: 100000 Duplicates: 0.688 sec	
15:22:29	SELECT kcu.CONSTRAINT_NAME, kcu.COLUMN_NAME, kcu.REFERENCED_TABLE_NAME, kcu.REFERENC...	3 row(s) returned	0.0011 sec / 0.00003...

Read Only

### Después

Don't Limit

```

SELECT kcu.CONSTRAINT_NAME, kcu.COLUMN_NAME,
       kcu.REFERENCED_TABLE_NAME, kcu.REFERENCED_COLUMN_NAME
  FROM information_schema.KEY_COLUMN_USAGE AS kcu
 WHERE kcu.TABLE_SCHEMA = DATABASE()
   AND kcu.TABLE_NAME = 'transaction'
   AND kcu.REFERENCED_TABLE_NAME IS NOT NULL;

152 •  ALTER TABLE user
153   RENAME TO data_user;
154
100% 20:162 |
```

**Result Grid** Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
id	char(10)	NO	PRI		
name	varchar(100)	YES			
surname	varchar(100)	YES			
phone	varchar(150)	YES			
email	varchar(150)	YES			
birth_date	varchar(100)	YES			
country	varchar(150)	YES			
city	varchar(150)	YES			
postal_code	varchar(100)	YES			
address	varchar(255)	YES			

Result 18

**Action Output**

Time	Action	Response	Duration / Fetch Time
14:00:15	DESCRIBE transaction	9 row(s) returned	0.0030 sec / 0.00001...
14:00:34	DESCRIBE data_user	10 row(s) returned	0.0023 sec / 0.00001...

Read Only

```

159
160    -- check Comprobación de la consistencia de los datos y tipos de columnas
161 •  DESCRIBE transaction;
162 •  DESCRIBE data_user;
163
164    -- Consistencia de los datos y tipos de columnas
165 •  ALTER TABLE data_user
166        MODIFY id INT NOT NULL;
167
168 •  ALTER TABLE transaction
169        MODIFY user_id INT;
170
171 •  ALTER TABLE transaction
172        ADD CONSTRAINT fk_transaction_user
173            FOREIGN KEY (user_id)
174            REFERENCES data_user(id);
175

```

### Corrección de tipos según el modelo objetivo:

```

168 •  ALTER TABLE transaction
169        ADD CONSTRAINT fk_transaction_user
170            FOREIGN KEY (user_id)
171            REFERENCES data_user(id);
172
173    #Buscar Orphan user_id
174 •  SELECT COUNT(*) AS orphan_user_ids
175        FROM transaction AS t
176        LEFT JOIN data_user AS du ON du.id = t.user_id
177        WHERE t.user_id IS NOT NULL
178        AND du.id IS NULL;
179
180 •  SELECT t.id, t.user_id
181        FROM transaction AS t
182        LEFT JOIN data_user AS du ON du.id = t.user_id
183        WHERE t.user_id IS NOT NULL
184        AND du.id IS NULL;
185
186 •  INSERT INTO data_user (id)
187        VALUES ('9999');
188
189
190

```

Action Output			
	Time	Action	Response
✓ 1	15:19:22	INSERT INTO data_user (id) VALUES ('9999')	1 row(s) affected
✓ 2	15:20:14	SELECT COUNT(*) AS orphan_user_ids FROM transaction AS t LEFT JOIN data_user AS du ON du.id = t.user_id WHERE t.user_id IS NOT NULL AND du.id IS NULL;	1 row(s) returned
✓ 3	15:20:34	ALTER TABLE transaction ADD CONSTRAINT fk_transaction_user FOREIGN KEY (user_id) REFERENCES data_user(id);	100000 row(s) affected Records: 100000 Duplicates:... 0.688 sec

### Detección y resolución de orfandad (Error 1452)

```

-- Verificar cambios a realizar en Tabla Company
DESCRIBE company;
ALTER TABLE company DROP website;

-- Verificar cambios a realizar en Tabla data user
DESCRIBE data_user;
ALTER TABLE data_user
    RENAME COLUMN email TO personal_email,
    MODIFY COLUMN id INT;

201
202    -- Verificar cambios a realizar en Tabla Credit_card
203 •  DESCRIBE credit_card;
204 •  ALTER TABLE credit_card
205        MODIFY COLUMN iban VARCHAR(50),
206        MODIFY COLUMN pin VARCHAR(4),
207        MODIFY COLUMN cvv INT,
208        MODIFY COLUMN expiring_date VARCHAR(20),
209        ADD COLUMN fecha_actual DATE;
210
211
212

```

Action Output			
	Time	Action	Response
✓ 1	22:23:17	DESCRIBE credit_card	5 row(s) returned
✓ 2	22:25:50	ALTER TABLE credit_card MODIFY COLUMN iban VARCHAR(50), MODIFY COLUMN pin VARCHAR(4), MO...	5001 row(s) affected Records: 5001 Duplicates: 0 W... 0.054 sec

```

210
211      -- Mostrar cambios en la tabla
212 • SHOW CREATE TABLE data_user;
213 • SHOW CREATE TABLE company;
214 • SHOW CREATE TABLE credit_card;
215 • SHOW CREATE TABLE transaction;
216
100%  1:212 | Form Editor Navigate: ⟲ ⟳ 1 / 1 ⟷

```

**Form Editor** Navigate: ⟲ ⟳ 1 / 1 ⟷

Table: `data_user`

Create Table:

```

CREATE TABLE `data_user` (
  `id` int NOT NULL,
  `name` varchar(100) DEFAULT NULL,
  `surname` varchar(100) DEFAULT NULL,
  `phone` varchar(150) DEFAULT NULL,
  `personal_email` varchar(150) DEFAULT NULL,
  `birth_date` varchar(100) DEFAULT NULL,
  `country` varchar(150) DEFAULT NULL,
  `city` varchar(150) DEFAULT NULL,
  `postal_code` varchar(100) DEFAULT NULL,
  `address` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
)

```

Result 10 Read Only

Action Output

Action	Time	Response	Duration / Fetch Time
SELECT * FROM InformeTecnico ORDER BY transaction_id DESC	22:35:25	100000 row(s) returned	0.0027 sec / 0.195 sec
SELECT kcu.CONSTRAINT_NAME, kcu.COLUMN_NAME, kcu.REFERENCED_TABLE_NAME, kcu.REFERE...	22:38:06	3 row(s) returned	0.0023 sec / 0.00000...
SHOW CREATE TABLE data_user	22:45:10	1 row(s) returned	0.0021 sec / 0.00001...

```

213 • SHOW CREATE TABLE company;
214 • SHOW CREATE TABLE credit_card;
215 • SHOW CREATE TABLE transaction;
216
100%  27:213 | Form Editor Navigate: ⟲ ⟳ 1 / 1 ⟷

```

**Form Editor** Navigate: ⟲ ⟳ 1 / 1 ⟷

Table: `company`

Create Table:

```

CREATE TABLE `company` (
  `id` varchar(15) NOT NULL,
  `company_name` varchar(255) DEFAULT NULL,
  `phone` varchar(15) DEFAULT NULL,
  `email` varchar(100) DEFAULT NULL,
  `country` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci

```

Result 11 Read Only

Action Output

Action	Time	Response	Duration / Fetch Time
SELECT kcu.CONSTRAINT_NAME, kcu.COLUMN_NAME, kcu.REFERENCED_TABLE_NAME, kcu.REFERE...	22:38:06	3 row(s) returned	0.0023 sec / 0.00000...
SHOW CREATE TABLE data_user	22:45:10	1 row(s) returned	0.0021 sec / 0.00001...
SHOW CREATE TABLE company	22:46:19	1 row(s) returned	0.00097 sec / 0.00000...

```

214 • SHOW CREATE TABLE credit_card;
215 • SHOW CREATE TABLE transaction;
216
100%  31:215 | Form Editor Navigate: ⟲ ⟳ 1 / 1 ⟷

```

**Form Editor** Navigate: ⟲ ⟳ 1 / 1 ⟷

Table: `transaction`

Create Table:

```

CREATE TABLE `transaction` (
  `id` varchar(255) NOT NULL,
  `credit_card_id` varchar(20) DEFAULT NULL,
  `company_id` varchar(20) DEFAULT NULL,
  `user_id` int DEFAULT NULL,
  `lat` float DEFAULT NULL,
  `longitude` float DEFAULT NULL,
  `timestamp` timestamp NULL DEFAULT NULL,
  `amount` decimal(10,2) DEFAULT NULL,
  `declined` tinyint(1) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `company_id` (`company_id`),
  KEY `fk_transaction_credit_card` (`credit_card_id`),
  KEY `fk_transaction_user` (`user_id`),
  CONSTRAINT `fk_transaction_credit_card` FOREIGN KEY (`credit_card_id`) REFERENCES `credit_card` (`id`),
)

```

Result 13 Read Only

Action Output

Action	Time	Response	Duration / Fetch Time
SHOW CREATE TABLE credit_card	22:46:30	1 row(s) returned	0.00098 sec / 0.00000...
SHOW CREATE TABLE transaction	22:46:38	1 row(s) returned	0.00074 sec / 0.00000...

```

215 • SHOW CREATE TABLE transaction;
216
100% 31:214 | 1 / 1
Form Editor Navigate: <> << >> >>
Table: credit_card
Create Table: CREATE TABLE `credit_card` (
  `id` varchar(20) NOT NULL,
  `iban` varchar(50) DEFAULT NULL,
  `pin` varchar(4) DEFAULT NULL,
  `cvv` int DEFAULT NULL,
  `expiring_date` varchar(20) DEFAULT NULL,
  `fecha_actual` date DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
Result 12
Action Output
Time Action Response Duration / Fetch Time
4 22:45:10 SHOW CREATE TABLE data_user 1 row(s) returned 0.0021 sec / 0.00001...
5 22:46:19 SHOW CREATE TABLE company 1 row(s) returned 0.00097 sec / 0.0000...
6 22:46:30 SHOW CREATE TABLE credit_card 1 row(s) returned 0.00098 sec / 0.000...

```

### Solució:

En una primera versió de la pràctica, es va intentar crear la relació entre la taula `transaction` i la taula d'usuaris amb una definició de **foreign key** que no quedava totalment alineada amb el diagrama objectiu. Després de la revisió **peer-to-peer**, es va detectar el punt clau: el diagrama final requeria que la relació es fes amb camps de tipus **INT** (tant `data_user.id` com `transaction.user_id`). A partir d'aquí, es va iniciar un procés de correcció per reconstruir el model de manera neta i traçable.

Primer, es va fer una revisió de l'estat del model per evitar canvis a cegues: es van llistar les taules existents (`SHOW TABLES`), es va revisar l'estructura de les taules implicades (`DESCRIBE`) i es van consultar les claus foranes actives a `transaction` mitjançant `information_schema.KEY_COLUMN_USAGE`. Això va permetre confirmar quines relacions ja existien (per exemple amb `company` i `credit_card`) i detectar que faltava/estava mal definida la relació amb els usuaris.

A continuació, per alinear el model amb el diagrama del professor, es va renombrar la taula `user` a `data_user`. Aquest canvi també evita confusions perquè `USER` pot actuar com a paraula reservada en MySQL, i millora la consistència semàntica del model. Un cop renombrada, es van fer comprovacions bàsiques (`SHOW TABLES LIKE 'data_user'` i `SELECT COUNT(*)`) per assegurar que la taula existia i que les dades seguien disponibles.

Després, es va verificar la compatibilitat de tipus entre els camps relacionats. En aquest punt (i segons el feedback del peer-to-peer), es va corregir l'estructura modificant els tipus de `data_user.id` i `transaction.user_id` a **INT**, ja que una foreign key requereix que els camps relacionats siguin compatibles i, idealment, idèntics en tipus i definició. Un cop ajustats els tipus, es va procedir a crear de nou la foreign key `fk_transaction_user`.

Durant la creació de la FK, MySQL va retornar l'error **1452**, indicant un problema d'integritat referencial: existien transaccions amb `user_id` que no tenien registre corresponent a `data_user`. Per resoldre-ho de manera controlada, es van identificar els valors "orfes" amb un `LEFT JOIN`. El resultat va mostrar que només hi havia **un únic cas** (`user_id = 9999`). Tot i que es podria haver fet una generació automàtica de múltiples usuaris "dummy" si hi hagués molts casos, en aquest escenari es va aplicar una correcció mínima: inserir un registre *placeholder* a `data_user` amb l'`id = 9999` per restaurar la integritat referencial sense alterar l'historic de `transaction`.

Finalment, es van aplicar ajustos addicionals per fer que l'estructura coincidís exactament amb el diagrama final: eliminació de camps no requerits (com `website` a `company`), canvi de nom de columna (`email` → `personal_email` a `data_user`), i adequació de tipus/longituds a `credit_card` (incloent l'addició de `fecha_actual`). Per tancar el procés, es va documentar i validar l'estat final

amb evidències objectives (`SHOW CREATE TABLE` i consultes a `information_schema`), confirmant que les claus foranes i l'estructura final concorden amb el diagrama objectiu.

## Exercici 2

L'empresa també us demana 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.

Assegureu-vos d'incloure informació rellevant de les taules que coneixereu i utilitzeu àlies per canviar de nom columnes segons calgui.

Mostra els resultats de la vista, ordena els resultats de forma descendente en funció de la variable ID de transacció.

```

195 •  CREATE OR REPLACE VIEW InformeTecnico AS
196   SELECT t.id          AS transaction_id,
197         du.name        AS user_name,
198         du.surname     AS user_surname,
199         cc.iban        AS credit_card_ibanc,
200         c.company_name AS company_name
201   FROM transaction    AS t
202   JOIN data_user      AS du ON du.id = t.user_id
203   JOIN credit_card    AS cc ON cc.id = t.credit_card_id
204   JOIN company        AS c  ON c.id = t.company_id;
205
206 •  SELECT *
207   FROM InformeTecnico
208   ORDER BY transaction_id DESC;
209

```

The screenshot shows the MySQL Workbench interface with the following details:

- Code Area:** Displays the SQL code for creating the view and executing it.
- Result Grid:** Shows the results of the query, displaying columns: transaction\_id, user\_name, user\_surname, credit\_card\_iban, and company\_name. The results list various transactions with their corresponding users, companies, and credit card details.
- Action Output:** Shows the log of actions taken, including the creation of the view and the execution of the select statement.

### Solución:

#### Criterio de unión (JOIN) y consistencia del resultado

Se utilizaron `JOIN` (inner join) entre las tablas para asegurar que el informe técnico muestre únicamente transacciones con información completa y consistente (usuario, tarjeta y compañía existentes). Este enfoque es apropiado para un informe técnico porque evita filas con valores nulos debidos a relaciones incompletas.

*(Si el objetivo fuese incluir también transacciones incompletas, podría considerarse `LEFT JOIN`, pero no es el caso prioritario en este informe.)*

#### Ordenación de resultados

Por buenas prácticas, la ordenación (`ORDER BY`) se aplica al **consultar** la vista, no dentro de su definición, ya que una vista representa un conjunto de datos y el orden final debe controlarse desde la query que consume la vista.

Por ello, se ejecutó un `SELECT * FROM InformeTecnico ORDER BY transaction_id DESC;` para cumplir el requisito del enunciado.