

Diego Mere

Data Analytics

Tasca Tasca S4.01. Creació de Base de Dades – Sprint4

Nivell 1

Descàrrega els arxius CSV, estudia'ls i dissenya una base de dades amb un esquema d'estrella que contingui, almenys 4 taules de les quals puguis realitzar les següents consultes

En este caso, tuve que crear todas las tablas (incluyendo productos aunque por ahora, esta desconectada del esquema), para permitir la carga local de archivos hubo que agregar **OPT_LOCAL_INFILE=1** en connections y **SET GLOBAL local_infile = 1** para activar y permitir la carga de archivos en local.

Primero creando la base de datos “Negocio” y luego las tablas:

```
4 • create database negocio;
5 • use negocio;
6
7 • create table IF NOT EXISTS companies(
8     company_id varchar(15) not null primary key,
9     company_name varchar(255),
10    phone varchar(20),
11    email varchar(50),
12    country varchar(50),
13    website varchar(100));
14
15
16 • LOAD DATA LOCAL INFILE 'C:/Users/diego/Desktop/Especializacion/SQL/Sprint 4/companies.csv'
17 INTO TABLE companies
18 FIELDS TERMINATED BY ','
19 ENCLOSED BY '"'
20 LINES TERMINATED BY '\r\n'
21 IGNORE 1 LINES;
```

```

24 • create table IF NOT EXISTS credit_cards(
25     id varchar(15) not null primary key,
26     user_id varchar(15),
27     iban varchar(50),
28     pan varchar(50),
29     pin varchar(6),
30     cvv varchar(3),
31     track1 varchar(50),
32     track2 varchar(50),
33     expiring_date varchar(50));
34
35 • LOAD DATA LOCAL INFILE 'C:/Users/diego/Desktop/Especializacion/SQL/Sprint 4/credit_cards.csv'
36 INTO TABLE credit_cards
37 FIELDS TERMINATED BY ','
38 LINES TERMINATED BY '\n'
39 IGNORE 1 LINES;
40
41 • create table IF NOT EXISTS products(
42     id varchar(15) not null primary key,
43     product_name varchar(50),
44     price varchar(15),
45     colour varchar(50),
46     weight float,
47     warehouse_id varchar(15));
48
49 • LOAD DATA LOCAL INFILE 'C:/Users/diego/Desktop/Especializacion/SQL/Sprint 4/products.csv'
50 INTO TABLE products FIELDS TERMINATED BY ','
51 LINES TERMINATED BY '\n'
52 IGNORE 1 LINES;
53
54
55 • create table IF NOT EXISTS user(
56     id varchar(15) not null primary key,
57     name varchar(50),
58     surname varchar(50),
59     phone varchar(50),
60     email varchar(50),
61     birth_date varchar(50),
62     country varchar(50),
63     city varchar(50),
64     postal_code varchar(20),
65     address varchar(100));
66
67
68 • LOAD DATA LOCAL INFILE 'C:/Users/diego/Desktop/Especializacion/SQL/Sprint 4/users_ca.csv'
69 INTO TABLE user
70 FIELDS TERMINATED BY ','
71 ENCLOSED BY '"'
72 LINES TERMINATED BY '\r\n'
73 IGNORE 1 LINES;
74

```

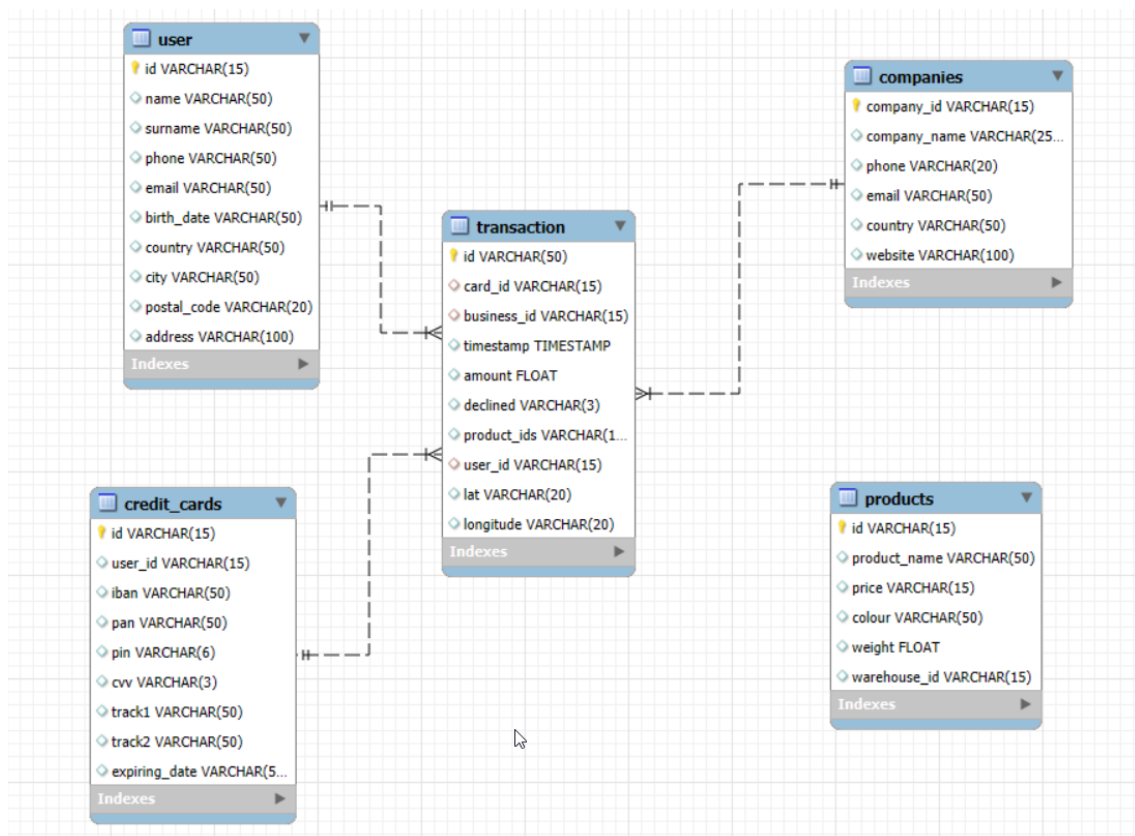
```

90 • create table IF NOT EXISTS transaction(
91   id varchar(50) not null primary key,
92   card_id varchar(15) ,
93   business_id varchar(15),
94   timestamp timestamp,
95   amount float,
96   declined varchar(3),
97   product_ids varchar(15),
98   user_id varchar(15),
99   lat varchar(20),
100  longitude varchar(20),
101  FOREIGN KEY(card_id) REFERENCES credit_cards(id),
102  FOREIGN KEY(business_id) REFERENCES companies(company_id),
103  FOREIGN KEY(user_id) REFERENCES user(id)
104 );
105
106
107 • LOAD DATA LOCAL INFILE 'C:/Users/diego/Desktop/Especializacion/SQL/Sprint 4/transactions.csv'
108 INTO TABLE transaction
109 FIELDS TERMINATED BY ';'
110 ENCLOSED BY '"'
111 LINES TERMINATED BY '\r\n'

```

En la tabla “Users” cargamos los 3 archivos, de los 3 países ya que tienen los mismos datos.

Cada carga contempla variables como el tipo de separador, o el final de línea. En la última tabla (“De hechos”) se crean las FKs que la relacionan con el resto, creando un esquema en estrella:



Exercici 1

Realitza una subconsulta que mostri tots els usuaris amb més de 30 transaccions utilitzant almenys 2 taules.

```
7 • SELECT name
8 FROM user
9 WHERE id IN (SELECT user_id
10 FROM transaction
11 GROUP BY user_id
12 HAVING COUNT(*) > 30
13 );
```

Usando subconsultas como solicitado, pero podría ser mas eficiente con joins

Exercici 2

Mostra la mitjana d'amount per IBAN de les targetes de crèdit a la companyia Donec Ltd, utilitza almenys 2 taules.

```
18 • SELECT credit_cards.iban, ROUND(AVG(transaction.amount), 2) AS Media
19 FROM credit_cards
20 JOIN transaction ON credit_cards.id = transaction.card_id
21 JOIN companies ON companies.company_id = transaction.business_id
22 WHERE companies.company_name = 'Donec Ltd'
23 GROUP BY credit_cards.iban;
24
25
```

Solo una tarjeta (con 2 gastos) para esta compa;ia

Nivell 2

Crea una nova taula que reflecteixi l'estat de les targetes de crèdit basat en si les últimes tres transaccions van ser declinades i genera la següent consulta:

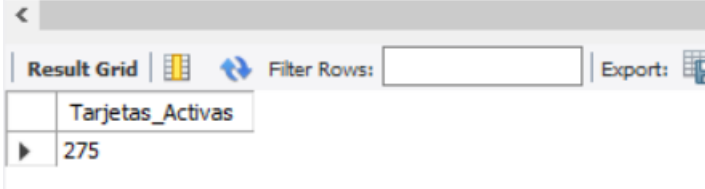
```
4 • create table Estado_Tarjetas
5 SELECT card_id, CASE
6 WHEN SUM(declined) = 3 THEN 'Desactivada'
7 ELSE 'Activa'
8 END AS Estado
9 FROM (SELECT card_id, declined
10 FROM (SELECT card_id, declined, ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY timestamp DESC) AS Ultimas
11 FROM transaction) t
12 WHERE t.Ultimas <= 3
13 ) x
14 GROUP BY card_id;
```

Para crear la tabla, al tener que considerar las ultimas 3 transacciones de cada tarjeta, no podemos crear un group by y además limitar a 3, por lo que podemos usar “**Window Functions**” que nos permite, en este caso, agrupar la tabla por card_id, ordenar por fecha (descendiente) y solo tomar las ultimas 3 de cada una. Uniendo esto como subconsulta a la tabla de tarjetas, podemos mostrar los card_ids y con un case, cuando la suma de las ultimas 3 transacciones sea 3 (o sea, 3 declined) la tarjeta se guardará como desactivada, en caso contrario, como activada. Y con esto creamos la tabla Estado_Tarjetas.

Exercici 1

Quantes targetes estan actives?

```
18 • select count(Estado) Tarjetas_Activas
19 from Estado_Tarjetas
20 where Estado = "Activa";
21
```



Tarjetas_Activas
275

Están activas todas las 275 tarjetas.

Nivell 3

Crea una taula amb la qual puguem unir les dades del nou arxiu products.csv amb la base de dades creada, tenint en compte que des de transaction tens product_ids. Genera la següent consulta:

```
5 • CREATE TABLE Trans_Prod (
6     id_Trans_Prod INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
7     id_transaction VARCHAR(50),
8     id_product VARCHAR(15)
9 );
```

Tabla intermedia que contenga cada producto de cada transacción individual, con solo 3 columnas, un id único de transacción y producto, creado al momento, el ID de la transacción dueña y cada uno de los products id de la transacción, por lo que hay que separarlos cuando haya mas de uno.

Creemos una tabla temporal, con 10 números (asumiendo que cada transacción tendrá un máximo de 10 productos, en caso de necesitar mas, se pueden añadir:

```
12 • CREATE TEMPORARY TABLE numbers (  
13     num INT);  
14  
15 • INSERT INTO numbers (num) VALUES (1), (2), (3), (4), (5), (6), (7), (8), (9), (10);  
16
```

Lo complicado es al querer dividir los products ids del campo de transacciones

```
18 • INSERT INTO Trans_Prod (id_transaction, id_product)  
19 SELECT t.id AS id_transaction, TRIM(SUBSTRING_INDEX(SUBSTRING_INDEX(t.product_ids, ',', n.num), ',', -1)) AS id_product  
20 /* t.id = id de la transaccion. SUBSTRING_INDEX(t.product_ids, ',', n.num) regresa n.num (numero) de products ID, lo llamaremos Resto  
21 SUBSTRING_INDEX(Resto, ',', -1) extrae el ultimo product ID del resto del paso anterior  
22 Trim elimina los espacios necesarios antes o despues de una ,  
23 */  
24 FROM transaction t JOIN numbers n -- join con la tabla temporal numbers, asumiendo que no haya transaction con mas de 10 productos, si la hay, hay que crear mas numeros  
25 ON n.num <= LENGTH(t.product_ids) - LENGTH(REPLACE(t.product_ids, ',', '')) + 1; -- LENGTH(t.product_ids) calcula tamaño de string con comas; LENGTH(REPLACE(t.product_ids,  
26
```

Este código tiene varias partes, primero:

SELECT t.id AS id_transaction,
TRIM(SUBSTRING_INDEX(SUBSTRING_INDEX(t.product_ids, ',', n.num), ',', -1)) AS
id_product

t.id = id de la transaccion.

SUBSTRING_INDEX(t.product_ids, ',', n.num) regresa n.num (numero) de products ID, lo llamaremos Resto

SUBSTRING_INDEX(Resto, ',', -1) extrae el ultimo product ID del resto del paso anterior

Trim elimina los espacios necesarios antes o después de una “,”

```
29 • ALTER TABLE Trans_Prod  
30 ADD CONSTRAINT fk_id_transaction  
31 FOREIGN KEY (id_transaction)  
32 REFERENCES transaction(id)  
33 ON DELETE CASCADE;  
34  
35 • ALTER TABLE Trans_Prod  
36 ADD CONSTRAINT fk_id_product  
37 FOREIGN KEY (id_product)  
38 REFERENCES products(id)  
39 ON DELETE CASCADE;  
40
```

Creemos las FK de productos y de transacciones en la tabla intermedia

Exercici 1

Necessitem conèixer el nombre de vegades que s'ha venut cada producte

```
43
44 • SELECT p.id, p.product_name, COUNT(tp.id_product) AS sales_count
45 FROM products p JOIN Trans_Prod tp ON p.id = tp.id_product
46 GROUP BY p.id, p.product_name
47 ORDER BY sales_count DESC;
```

Este código muestra, el id del producto, su nombre, cuenta las veces que aparece el producto en la tabla intermedia de transacciones y productos y ordena de mayor a menos, haciendo un join entre productos y la tabla intermedia Trans_Prod

	id	product_name	sales_count
▶	23	riverlands north	68
	67	Winterfell	68
	79	Direwolf riverlands the	66
	2	Tarly Stark	65
	43	duel	65

Esquema final:

