

SPRINT 4

Creació de Base de Dades

Borja Munitiz

Previ

Creació de la base de dates a partir de la investigació dels arxius CSV aportats.

Primer crearem les taules amb les seves PRIMARY KEYS i FOREIGN KEYS, basant-nos en les columnes i dades de cadaquè dels fitxers CSV, primer creem les taules PARENTS (COMPANIES, PRODUCTS i DATA_USER), a continuació les CHILD (CREDIT_CARDS i TRANSACTIONS), per assegurar-nos que les FOREIGN KEYS estan correctament connectades.

Un cop tenim totes les taules creades intèrim les dades dels fitxers CSV amb la funció LOAD DATA LOCAL INFILE, com no tots els fitxers tenen els mateixos separadors hem hagut d'explicitar-lo per a cada inserció. He decidit inserir totes les dades dels usuaris a la mateixa taula (DATA_USER) per fer les consultes més eficients i connectar aquesta taula directament amb la taula CREDIT_CARDS. (En intentar inserir les dades he tingut un error en MYSQL Workbench, l'he solucionat canviant la SYSTEM VALUE local_infile a ON, d'aquesta manera el sistema m'ha permès introduir dades dels arxius CSV)

Un cop tenim les taules amb les seves dades podem refer les relacions entre les taules i eliminar aquells camps que no necessitem, en aquest he eliminat la columna USER_ID de la taula TRANSACTIONS, ja que es pot accedir a aquesta informació a través de la taula CREDIT_CARDS, així tindrem un esquema de floc de neu.

Farem la relació entre PRODUCTS i TRANSACTIONS, pel fet que es dona una relació N:M i hem de crear una taula intermèdia per relacionar-les, per poder obtenir els ID dels productes de cada transacció utilitzarem una funció RECURSIVE per extreure de cada filera els ID que conté i els inserim a la taula intermèdia. Un cop fet això creem les claus necessàries per connectar les taules PRODUCTS i TRANSACTIONS.

Creació de l'esquema i les taules

- `CREATE SCHEMA modeling;`
- `USE modeling;`

```
• ⊕ CREATE TABLE companies(  
company_id VARCHAR (30) PRIMARY KEY,  
company_name VARCHAR (255),  
phone VARCHAR (30),  
email VARCHAR (50),  
country VARCHAR (50),  
website VARCHAR (255)  
);
```

```
• ⊕ CREATE TABLE products(  
id VARCHAR (30) PRIMARY KEY,  
product_name VARCHAR (255),  
price VARCHAR (30),  
colour VARCHAR (30),  
weight VARCHAR (50),  
warehouse_id VARCHAR (50)  
);
```

```
• ⊕ CREATE TABLE data_user(  
id VARCHAR (30) PRIMARY KEY,  
name VARCHAR (30),  
surname VARCHAR (30),  
phone VARCHAR (30),  
email VARCHAR (50),  
birth_date VARCHAR (50),  
country VARCHAR (50),  
city VARCHAR (50),  
postal_code VARCHAR (50),  
address VARCHAR (255)  
);
```

```
• ⊕ CREATE TABLE credit_cards(  
id VARCHAR (30) PRIMARY KEY,  
user_id VARCHAR (30) ,  
iban VARCHAR (200),  
pan VARCHAR (200),  
pin VARCHAR (200),  
cvv INT,  
track1 VARCHAR (255),  
track2 VARCHAR (255),  
expiring_date VARCHAR (100),  
CONSTRAINT user_id FOREIGN KEY(user_id) REFERENCES data_user(id)  
);
```

```
• ⊕ CREATE TABLE transactions(  
id VARCHAR (255) PRIMARY KEY,  
card_id VARCHAR (30),  
business_id VARCHAR (30),  
timestamp TIMESTAMP,  
amount DECIMAL(10,2),  
declined TINYINT (1),  
product_ids VARCHAR (30),  
user_id VARCHAR (30),  
lat FLOAT,  
longitude FLOAT,  
CONSTRAINT card_id FOREIGN KEY(card_id) REFERENCES credit_cards(id),  
CONSTRAINT business_id FOREIGN KEY(business_id) REFERENCES companies(company_id),  
CONSTRAINT t_user_id FOREIGN KEY(user_id) REFERENCES data_user(id)  
);
```

Inserció de les dades

- ```
LOAD DATA LOCAL INFILE '/Users/borja/Desktop/IT Academy/Especialització/SQL/Sprint 4/companies.csv'
INTO TABLE companies
FIELDS TERMINATED BY ','
ENCLOSED BY ""
LINES TERMINATED BY '\n'
IGNORE 1 LINES;
```

- ```
LOAD DATA LOCAL INFILE '/Users/borja/Desktop/IT Academy/Especialització/SQL/Sprint 4/credit_cards.csv'
INTO TABLE credit_cards
FIELDS TERMINATED BY ','
ENCLOSED BY ""
LINES TERMINATED BY '\n'
IGNORE 1 LINES;
```

- ```
LOAD DATA LOCAL INFILE '/Users/borja/Desktop/IT Academy/Especialització/SQL/Sprint 4/products.csv'
INTO TABLE products
FIELDS TERMINATED BY ','
ENCLOSED BY ""
LINES TERMINATED BY '\n'
IGNORE 1 LINES;
```

- ```
LOAD DATA LOCAL INFILE '/Users/borja/Desktop/IT Academy/Especialització/SQL/Sprint 4/transactions.csv'
INTO TABLE transactions
FIELDS TERMINATED BY ';'
ENCLOSED BY ""
LINES TERMINATED BY '\r\n'
IGNORE 1 LINES;
```

- ```
LOAD DATA LOCAL INFILE '/Users/borja/Desktop/IT Academy/Especialització/SQL/Sprint 4/users_ca.csv'
INTO TABLE data_user
FIELDS TERMINATED BY ','
ENCLOSED BY ""
LINES TERMINATED BY '\r\n'
IGNORE 1 LINES;
```

- ```
LOAD DATA LOCAL INFILE '/Users/borja/Desktop/IT Academy/Especialització/SQL/Sprint 4/users_uk.csv'
INTO TABLE data_user
FIELDS TERMINATED BY ','
ENCLOSED BY ""
LINES TERMINATED BY '\r\n'
IGNORE 1 LINES;
```

- ```
LOAD DATA LOCAL INFILE '/Users/borja/Desktop/IT Academy/Especialització/SQL/Sprint 4/users_usa_1.csv'
INTO TABLE data_user
FIELDS TERMINATED BY ';'
ENCLOSED BY ""
LINES TERMINATED BY '\n'
IGNORE 1 LINES;
```

# Arreglem les conexions entre les taules

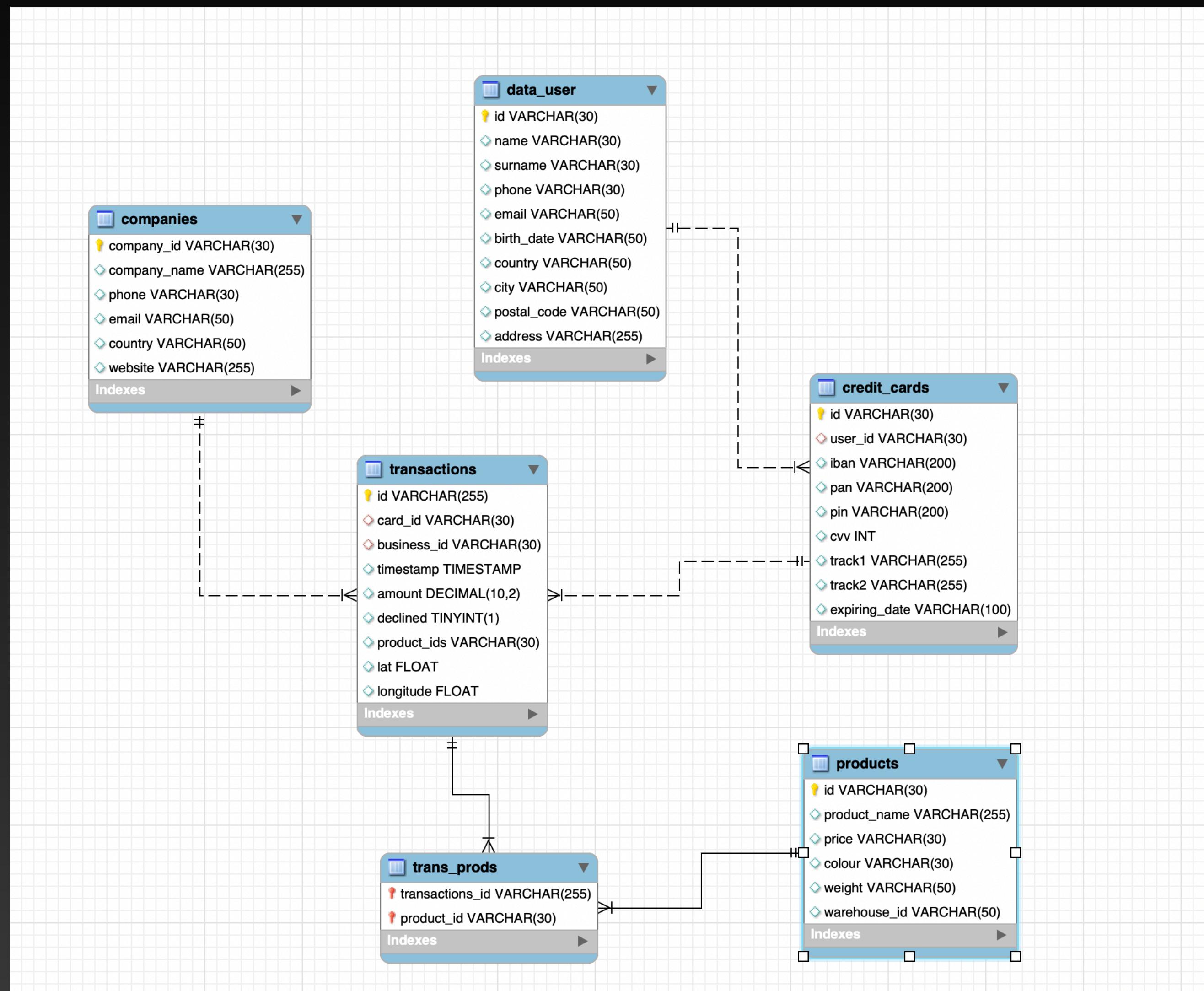
- `ALTER TABLE transactions  
DROP CONSTRAINT t_user_id;`
- `ALTER TABLE transactions  
DROP COLUMN user_id;`

- `CREATE TABLE trans_prods(  
transactions_id VARCHAR (255),  
product_id INT  
);`

```
INSERT INTO trans_prods
WITH RECURSIVE trans_prod AS (
SELECT transactions.id AS trans_id,
CAST(SUBSTRING_INDEX(SUBSTRING_INDEX(transactions.product_ids, ',', n), ',', -1) AS UNSIGNED) AS prod_id,
1 AS n
FROM transactions
JOIN (
SELECT 1 AS n
UNION ALL SELECT 2
UNION ALL SELECT 3
UNION ALL SELECT 4
UNION ALL SELECT 5
) AS numbers
ON n <= 1 + (LENGTH(transactions.product_ids)-LENGTH(REPLACE(transactions.product_ids, ',', ',')))
)
SELECT trans_id, prod_id
FROM trans_prod;
```

- `ALTER TABLE trans_prods  
ADD PRIMARY KEY (transactions_id, product_id),  
ADD FOREIGN KEY (transactions_id) REFERENCES transactions(id),  
ADD FOREIGN KEY (product_id) REFERENCES products(id);`

# Esquema final de la base de dades



# Exercici 1

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

Per a saber els usuaris amb més de 30 transaccions fem un conteig de les transaccions de cada usuari unint les talles DATA\_USER, CREDIT\_CARDS I TRANSACTIONS, agrupem pel cognom i filtrem només aquella que tenen més de 30 transaccions.

- ```
SELECT name, surname, COUNT(transactions.id)AS quant_trans
FROM data_user
INNER JOIN credit_cards
ON data_user.id = user_id
LEFT JOIN transactions
ON credit_cards.id = card_id
GROUP BY name, surname
HAVING quant_trans > 30;
```

	name	surname	quant_trans
	Ocean	Nelson	39
	Hedwig	Gilbert	38
	Lynn	Riddle	39

Exercici 2

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

Per a saber la mitjana de AMOUNT de les targetes per IBAN de la companyia "Donec Ltd" realitzem la mitjana de AMOUNT arrodonint el resultat i mostrant 2 decimals, unim les taules TRANSACTIONS, COMPANIES i CREDIT_CARDS, filtrem pel nom de la companyia i agrupem per nom de la companyia i IBAN

```
207 •   SELECT company_name, iban, ROUND(AVG(amount),2) AS mitjana
208   FROM transactions
209   INNER JOIN companies
210   ON (business_id = company_id)
211   INNER JOIN credit_cards
212   ON card_id = credit_cards.id
213   WHERE company_name = "Donec Ltd"
214   GROUP BY company_name, iban;
215
216
```

100% 1:203 |

Result Grid Filter Rows: Search Export:

company_name	iban	mitjana
Donec Ltd	PT87806228135092429456346	203.72

Result 51

Action Output

Time	Action	Response
259 10:03:15	SELECT company_name, iban, ROUND(AVG(amount),2) AS mitjana FROM transactions INNER JOIN companies ON (busine... 1 row(s) returned	

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:

Per a crear la taula on sapiguem si les últimes 3 transaccions de cada targeta han estat declinades primer creuem una SUQUERY on extraguem les últimes 3 transaccions que han estat declinades, en una SUBQUERY exterior usem la SUBQUERY anterior per a extreure les últimes transaccions declinades per a cada targeta, i adjuntem la selecció amb les taules CREDIT_CARDS i TRANSACTIONS, usem el CASE per a definir si les targetes estan inactives o no en funció de si les últimes 3 transaccions han estat declinades, al final vam ordenar per CARD_ID, amb la QUERY completa fem la taula amb CREATE TABLE ... AS.

```
• CREATE TABLE targetes_decl AS
  SELECT DISTINCT(transactions.card_id), times_decl AS decl_mes_de_3, STR_TO_DATE(expiring_date, "%m/%d/%y") AS expired,
CASE
    WHEN times_decl >= 3 THEN "Targeta inactiva"
    ELSE "Targeta activa"
END estat
FROM transactions
LEFT JOIN credit_cards
ON transactions.card_id = credit_cards.id
LEFT JOIN (SELECT card_id, COUNT(card_id) AS times_decl
          FROM transactions
          WHERE transactions.timestamp IN (SELECT * FROM (
              SELECT t.timestamp
              FROM transactions t
              WHERE t.card_id = transactions.card_id AND t.declined = 1
              ORDER BY timestamp DESC
              LIMIT 3
          ) AS time
          GROUP BY card_id
          HAVING COUNT(card_id)>=3
      ) AS last_decl
      ON transactions.card_id = last_decl.card_id
      ORDER BY card_id;

• SELECT *
  FROM targetes_decl;
```

Exercici 1

Quantes targetes estan actives?

Amb la taula creada podem fer una QUERY on comptem les targetes que estan actives filtrant pel camp ESTAT i si la data de caducitat ja ha passat.

```
264 •   SELECT COUNT(*) as targetes_actives
265   FROM targetes_decl
266   WHERE estat = "Targeta activa" AND expired > CURRENT_DATE;
267
268
```

100% 18:259

Result Grid Filter Rows: Search Export:

targetes_activ...
32

Result 71

Action Output

	Time	Action	Response
--	------	--------	----------

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:

En haver realitzat aquesta taula de relació intermèdia en la fase de creació de l'esquema ja estem preparats per a fer la consulta, en tot cas el mètode per a crear-la ha estat el següent:

"Farem la relació entre PRODUCTS i TRANSACTIONS, pel fet que es dona una relació N:M i hem de crear una taula intermèdia per relacionar-les, per poder obtenir els ID dels productes de cada transacció utilitzarem una funció RECURSIVE per extreure de cada filera els ID que conté i els inserim a la taula intermèdia. Un cop fet això creem les claus necessàries per connectar les taules PRODUCTS i TRANSACTIONS."

```
CREATE TABLE trans_prods(
    transactions_id VARCHAR (255),
    product_id INT
);
```

```
INSERT INTO trans_prods
WITH RECURSIVE trans_prod AS (
    SELECT transactions.id AS trans_id,
    CAST(SUBSTRING_INDEX(SUBSTRING_INDEX(transactions.product_ids, ',', n), ',', -1) AS UNSIGNED) AS prod_id,
    1 AS n
    FROM transactions
    JOIN (
        SELECT 1 AS n
        UNION ALL SELECT 2
        UNION ALL SELECT 3
        UNION ALL SELECT 4
        UNION ALL SELECT 5
    ) AS numbers
    ON n <= 1 + (LENGTH(transactions.product_ids)-LENGTH(REPLACE(transactions.product_ids, ',', '')))
)
SELECT trans_id, prod_id
FROM trans_prod;
```

Exercici 1

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

Per a saber la quantitat de vegades que s'ha venut un producte el fem comptant la quantitat de transaccions per producte, unint les taules TRANS_PRODS i PRODUCTS, agrupem pel nom del producte i el seu ID, ordenem en funció del comptatge de transaccions.

```
271 •   SELECT products.id, product_name, COUNT(transactions_id) AS vegades_venut
272   FROM trans_prods
273   RIGHT JOIN products
274     ON products.id = product_id
275   GROUP BY product_name, products.id
276   ORDER BY vegades_venut DESC;
```

100% 1:267

Result Grid Filter Rows: Search Export:

id	product_name	vegades_venut
79	Direwolf riverlands the	66
43	duel	65
2	Tarly Stark	65

Result 53

Action Output

Time	Action	Response
10:13:29	SELECT products.id, product_name, COUNT(transactions_id) AS vegades_venut FROM trans_prods RIGHT JOIN products...	100 row(s) returned

Esquema final

Podem observar que la taula TARGETES_DECL està separada de la resta de taules ja que conté totes les dades necessàries per a realitzar les consultes, en cas necessari es podria declarar com FOREIGN KEY el camp CARD_ID i usar-se com a taula accessòria amb una relació de 1:1 amb la taula CREDIT_CARDS

