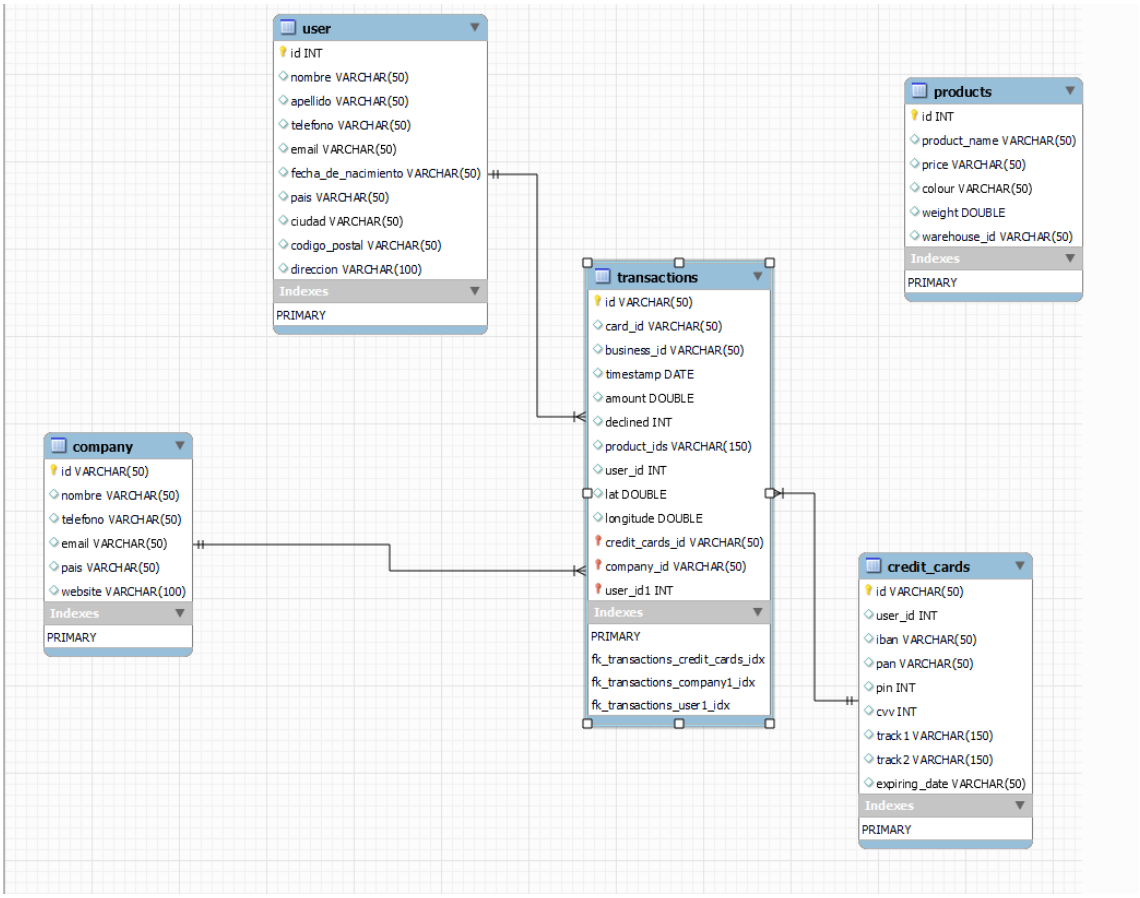
Tarea S4.01. Modelaje SQL

Nivel 1

Se crea el esquema de trabajo SP4, y luego se procede a crear la tabla user teniendo en cuenta la distribución de los datos en los archivos csv "users_ca", "users_uk" y "users_usa"

manteniendo la característica de autoincremento del campo id que es la PRIMARY KEY

del mismo modo se crean las tablas restantes dentro del modelo importando la data desde los archivos csv.



En este momento la tabla “Products” aún no ha sido integrada al esquema pues en la tabla “transactions” el identificador de productos relacionado a las transacciones agrupa en un mismo campo los productos vendidos impidiendo su relación directa con la tabla “Products”

Tarea S4.01. Modelaje SQL

Ejercicio 1

Realiza una subconsulta que muestre a todos los usuarios con más de 30 transacciones utilizando al menos 2 tablas. En la imagen se muestra el script se realiza una consulta (no Subconsulta) que muestra los resultados de manera efectiva.

```
83
84 •  SELECT u.id AS Usuario, count(t.id) AS No_operaciones
85    FROM user u
86    JOIN transactions t ON u.id = t.user_id
87    GROUP BY Usuario
88    HAVING No_operaciones > 30;
89
```

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

Usuario	No_operaciones
92	39
267	52
272	76
275	48

Ejercicio 2

Muestra el promedio de la suma de transacciones por IBAN de las tarjetas de crédito en la compañía “Donec Ltd.” utilizando al menos 2 tablas.

```
91
92 •  SELECT AVG(t.amount) AS Media_de_gasto, cc.iban AS Iban
93    FROM credit_cards cc
94    JOIN transactions t ON t.card_id = cc.id
95    JOIN company c ON t.business_id = c.id
96    WHERE c.nombre IN ("Donec Ltd")
97    GROUP BY Iban;
98
```

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

Media_de_gasto	Iban
203.715	PT87806228135092429456346

Tarea S4.01. Modelaje SQL

Ejercicio

Nivel 2

Este script de MySQL crea una tabla llamada Status basada en una consulta que determina el estado de activación de las tarjetas de crédito según las últimas tres transacciones.

Subconsulta Interna:

La subconsulta interna está anidada dentro de otra consulta. Esta subconsulta se encarga de seleccionar las últimas tres transacciones para cada tarjeta de crédito:

La consulta selecciona las columnas “card_id”, “declined”, y “timestamp” de la tabla “transactions”, utilizando las variables de usuario “@rown” y “@target” para llevar un seguimiento del número de transacciones por tarjeta de crédito.

Ordena las filas por “card_id”, “timestamp” en orden descendente y “declined”, numerando las filas para cada tarjeta de crédito según la fecha de cada operación y muestra si la transacción fue rechazada, con un límite de 3 transacciones por tarjeta.

Consulta Externa:

La consulta externa se aplica a los resultados de la subconsulta interna.

Agrupa los resultados por “card_id”, y utilizando una función de agregación (SUM) calcula la suma de los valores de “declined” (0 aceptada, 1 rechazada) para cada tarjeta de crédito.

Luego, utiliza una expresión CASE para determinar el estado de la tarjeta de crédito:

Si la suma de “declined” para las últimas tres transacciones es igual a 3, entonces la tarjeta se considera "Inactiva".

De lo contrario, se considera "Activa".

El resultado de esta evaluación se almacena en una columna llamada Status.

Esta tabla podría ser integrada al esquema teniendo en cuenta que el campo “Card_id” contiene valores únicos que pueden relacionarse con la tabla “transactions” referenciado al campo del mismo nombre el cual es una FOREIGN KEY.

Tarea S4.01. Modelaje SQL

Tarea S4.01. Modelaje SQL NV... | transactions | transactions | status

Limit to 1000 rows

```sql
CREATE TABLE Status (
SELECT last3.Card_id, (
CASE
    WHEN sum(last3.declined) = 3 THEN 'Inactiva'
    ELSE 'Activa'
END) as Status
FROM (SELECT card_id, declined, timestamp
 FROM ( SELECT t.declined, t.card_id, t.timestamp,
     @rown := IF(@target = t.card_id, @rown + 1, 1) AS rown, @target := t.card_id
        FROM transactions t JOIN (SELECT @target := NULL, @rown := 0)
AS Bucle ORDER BY t.card_id, t.timestamp DESC, t.declined ) AS T1 WHERE rown <= 3) AS last3
GROUP BY card_id);

SELECT *
FROM status;
```

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

Card_id	Status
CcU-2938	Activa
CcU-2945	Activa
CcU-2952	Activa
CcU-2959	Activa
CcU-2966	Activa
CcU-2973	Activa
CcU-2980	Activa
CcU-2987	Activa
CcU-2994	Activa
CcU-3001	Activa
CcU-3008	Activa
CcU-3015	Activa
CcU-3022	Activa
CcU-3029	Activa
CcU-3036	Activa
CcU-3043	Activa

status 29

Tarea S4.01. Modelaje SQL

Ejercicio

Nivel 3

Se crea una nueva tabla denominada “Prods_Transaction”, en esta tabla se han separado los id de producto que estaban agrupados en un solo campo, de forma que en esta nueva tabla se muestra el id de la transacción y el producto vendido uno por uno. Valiéndome de la instrucción SUBSTRING_INDEX, y declarando una variable de usuario que determina el numero de productos vendidos en cada transacción, he logrado separar estos productos a fin de normalizar los datos y poder relacionar la nueva tabla creada con el esquema.

```
#Ejercicio Nv3

CREATE TABLE Prods_Transaction (
SELECT id, pid
FROM (
    SELECT id, @num := 1 + LENGTH(product_ids) - LENGTH(REPLACE(product_ids, ',', '')) AS num,
        IF(@num >= 1, SUBSTRING_INDEX(product_ids, ',', 1), NULL) AS PID
    FROM transactions t
    where t.declined = 0) AS pdi1
UNION ALL
SELECT id, pid
FROM (
    SELECT id, @num := 1 + LENGTH(product_ids) - LENGTH(REPLACE(product_ids, ',', '')) AS num,
        IF(@num > 1, SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', 2), ',', -1), NULL) AS PID
    FROM transactions t
    where t.declined = 0) AS pdi2
UNION ALL
SELECT id, pid
FROM (
    SELECT id, @num := 1 + LENGTH(product_ids) - LENGTH(REPLACE(product_ids, ',', '')) AS num,
        IF(@num > 2, SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', 3), ',', -1), NULL) AS PID
    FROM transactions t
    where t.declined = 0) AS pdi3
UNION ALL
SELECT id, pid
FROM (
    SELECT id, @num := 1 + LENGTH(product_ids) - LENGTH(REPLACE(product_ids, ',', '')) AS num,
        IF(@num > 3, SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', 4), ',', -1), NULL) AS PID
    FROM transactions t
    where t.declined = 0) AS pdi4
);
```

Luego se han agregado las FOREIGN KEYS, a fin de relacionar la nueva tabla con el esquema.

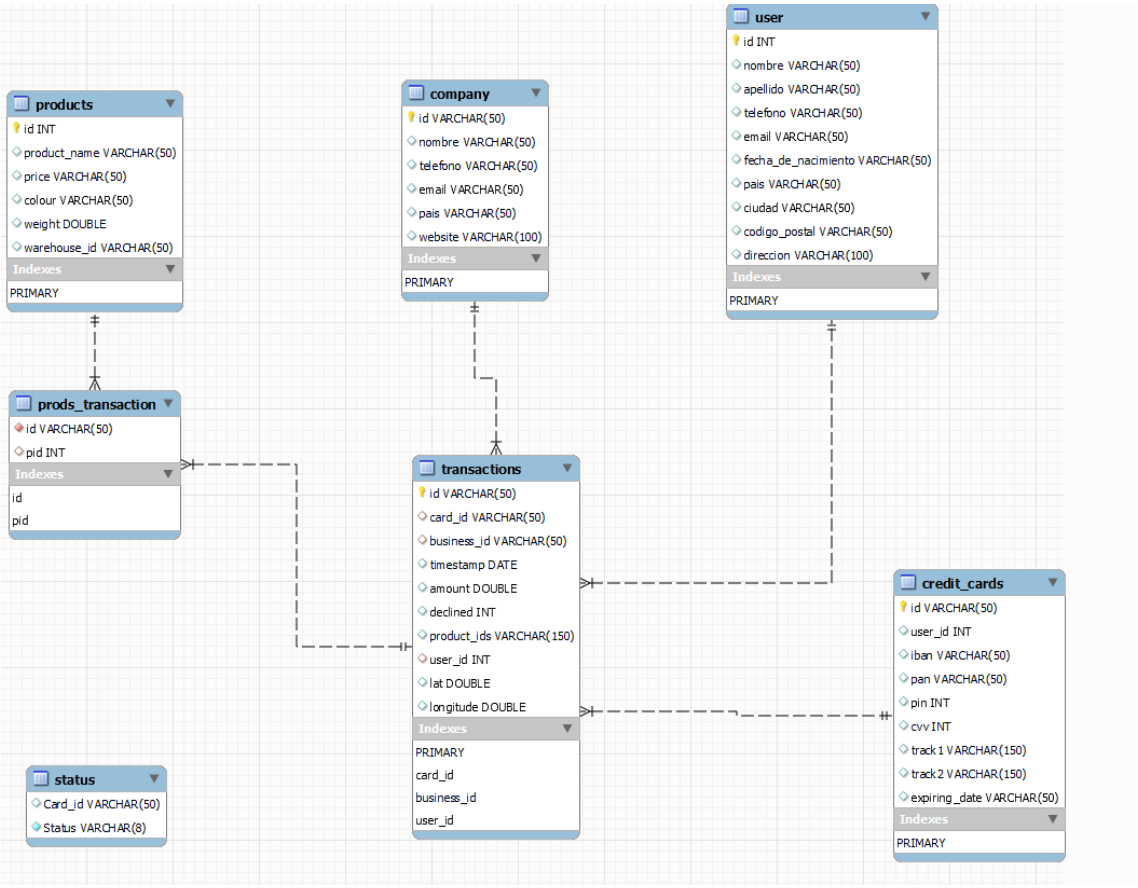
Vista de la nueva tabla “Prods_Transaction”, se aprecia que resultado del script las transacciones en las cuales se comercializa menos de 4 productos distintos (este es el máximo de productos por transacción en los datos que se tienen) se muestran registros donde el Product ID es NULL. Se han tenido en cuenta solo las operaciones efectivamente realizadas.

Tarea S4.01. Modelaje SQL

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

id	pid
02C6201E-D90A-1859-B4EE-88D2986D3B02	71
02C6201E-D90A-1859-B4EE-88D2986D3B02	1
02C6201E-D90A-1859-B4EE-88D2986D3B02	19
02C6201E-D90A-1859-B4EE-88D2986D3B02	NULL
0466A42E-47CF-8D24-FD01-C0B689713128	47
0466A42E-47CF-8D24-FD01-C0B689713128	97
0466A42E-47CF-8D24-FD01-C0B689713128	43
0466A42E-47CF-8D24-FD01-C0B689713128	NULL
063FBA79-99EC-66FB-29F7-25726D1764A5	47
063FBA79-99EC-66FB-29F7-25726D1764A5	67
063FBA79-99EC-66FB-29F7-25726D1764A5	31
063FBA79-99EC-66FB-29F7-25726D1764A5	5
0668296C-CDB9-A883-76BC-2E4C44F8C8AE	89
0668296C-CDB9-A883-76BC-2E4C44F8C8AE	83
0668296C-CDB9-A883-76BC-2E4C44F8C8AE	79
0668296C-CDB9-A883-76BC-2E4C44F8C8AE	NULL
06CD9AA5-9B42-D684-DDDD-A5E394FEBA99	43
06CD9AA5-9B42-D684-DDDD-A5E394FEBA99	31
06CD9AA5-9B42-D684-DDDD-A5E394FEBA99	NULL

La nueva tabla contiene un campo “id” FOREIGN KEY referenciado al campo “id” de la Tabla “transactions” (PRIMARY KEY), contiene así mismo un campo “pid” FOREIGN KEY referenciado al campo “id” de la Tabla “products” (PRIMARY KEY), permitiendo de esta manera la integración de la tabla “products” al modelo:



Tarea S4.01. Modelaje SQL

Finalmente se realiza la consulta según se ha requerido en el ejercicio mostrándose el total de productos vendidos, según su id.

```
153 •   SELECT p.id AS Producto, COUNT(pt.pid) Total_Vendido
154     FROM products p
155     JOIN prods_transaction pt ON pt.pid = p.id
156     GROUP BY Producto
157     ORDER BY Producto ASC;
```

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

Producto	Total_Vendido
1	51
2	56
3	43
5	42
7	44
11	40
13	51
17	54
19	44
23	60
29	43
31	40
37	45
41	48
43	54
47	50
53	47
59	35
61	50
67	59
71	44
73	39
79	52
83	46
89	46
97	53