

# “Diseño físico y optimización de consultas en el caso SuperAndes”

Juan Sebastián Alegría Zúñiga, Jaime Andrés Torres Bermejo  
ISIS2304 – Iteración 4 – Proyecto de curso - Caso SuperAndes  
Universidad de los Andes, Bogotá, Colombia  
[j.alegria@uniandes.edu.co](mailto:j.alegria@uniandes.edu.co), [j.torres16@uniandes.edu.co](mailto:j.torres16@uniandes.edu.co)  
Fecha de presentación: diciembre 4 de 2022

1	Introducción .....	1
2	Resultados .....	1
2.1	Modelos.....	1
2.2	Diseño físico .....	2
2.3	Resultados logrados .....	3
2.4	Estructura del proyecto .....	3
2.5	Resultados no logrados .....	4
2.6	Balance de pruebas .....	4
3	Planes de ejecución .....	4
3.1	RFC10 .....	4
3.2	RFC11 .....	5
3.3	RFC12 .....	5
3.4	RFC13 .....	6
3.5	Efectos de la selectividad durante la consulta.....	7
4	Consultas SQL .....	9
4.1	RFC10 .....	9
4.2	RFC11 .....	10
4.3	RFC12 .....	10
4.4	RFC13 .....	11
5	Diseño de datos .....	12
6	Planes de optimización .....	15
6.1	Análisis de ejecución de consultas entre aplicaciones .....	15
7	Conclusiones .....	15
8	Bibliografía .....	15

## 1 Introducción

Este documento presenta los resultados obtenidos para la cuarta iteración del proyecto SuperAndes en el curso Sistemas Transaccionales de la Universidad de Los Andes.

## 2 Resultados

### 2.1 Modelos

A continuación, se presenta tanto el modelo conceptual como relacional, luego de haberlo normalizado y agregado atributos útiles para la implementación de los requerimientos del caso:

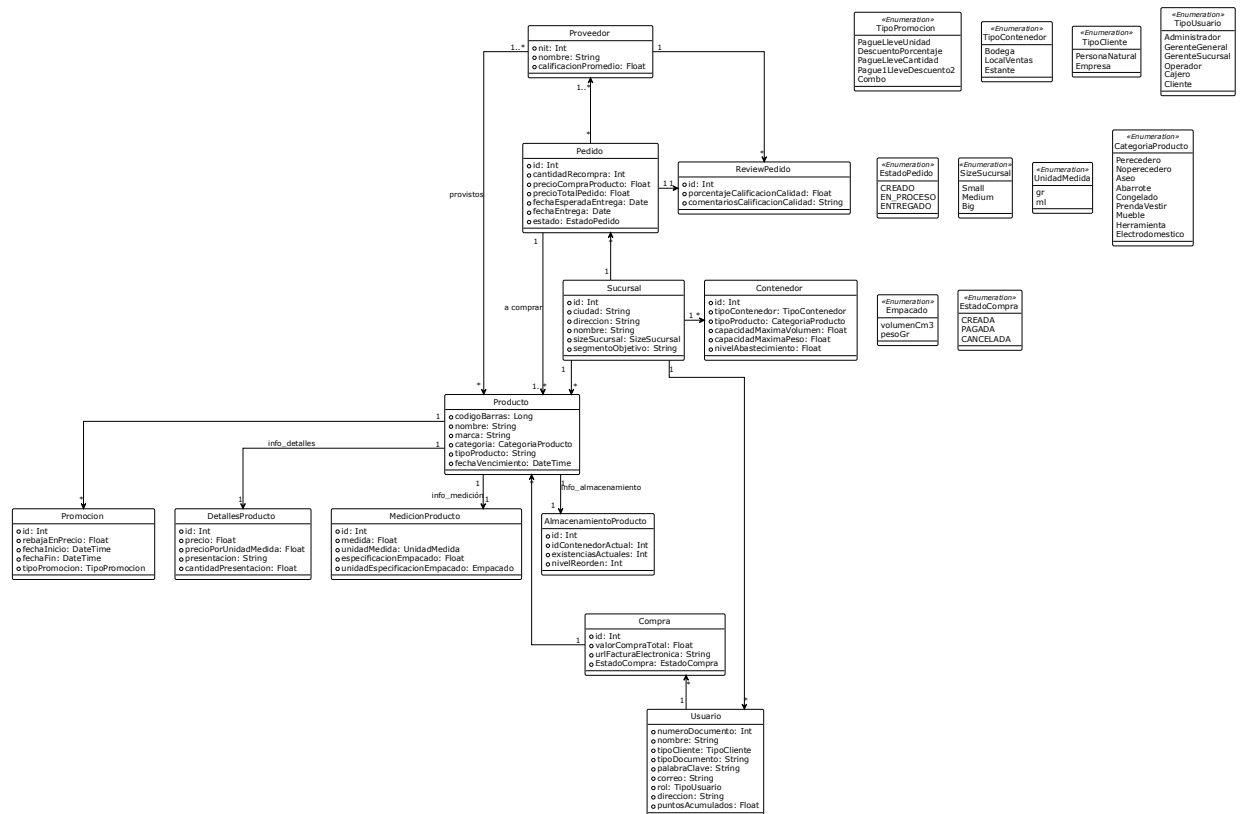


Figura 1. Modelo conceptual

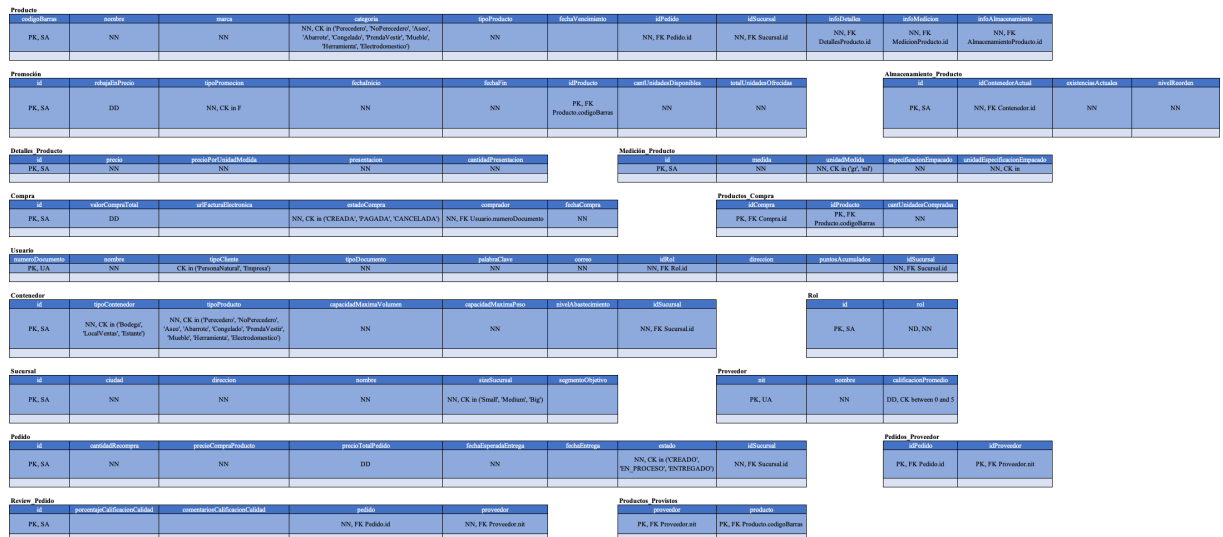


Figura 2. Modelo relacional

## 2.2 Diseño físico

A continuación, se presenta una captura de pantalla en la que se evidencia los índices generados automáticamente por Oracle. En su mayoría son llaves primarias de las tablas, creados a partir de las restricciones. Posiblemente, Oracle los creo para realizar las queries y joins de manera eficiente. Estos son en su mayoría índices primarios, que son de gran valor para encontrar rápidamente los registros deseados y contribuyen a los requerimientos funcionales de consulta.

The screenshot shows a SQL query result in a database client. The query is: `SELECT * FROM ALL_INDEXES WHERE TABLE_OWNER = 'ISIS2304B03202220';`. The result displays 27 rows of index information. The columns are: OWNER, INDEX\_NAME, INDEX\_TYPE, TABLE\_OWNER, TABLE\_NAME, TABLE\_TYPE, UNIQUENESS, COMPRESSION, and PRE. All entries have the same OWNER and TABLE\_OWNER, and most are NORMAL or UNIQUE indexes on various tables.

	OWNER	INDEX_NAME	INDEX_TYPE	TABLE_OWNER	TABLE_NAME	TABLE_TYPE	UNIQUENESS	COMPRESSION	PRE
1	ISIS2304B03202220	A_BEBIDA_PK	NORMAL	ISIS2304B03202220	A_BEBIDA	TABLE	UNIQUE	DISABLED	
2	ISIS2304B03202220	A_GUSTAN_PK	NORMAL	ISIS2304B03202220	A_GUSTAN	TABLE	UNIQUE	DISABLED	
3	ISIS2304B03202220	A_SIRVEN_PK	NORMAL	ISIS2304B03202220	A_SIRVEN	TABLE	UNIQUE	DISABLED	
4	ISIS2304B03202220	SYS_C00756370	NORMAL	ISIS2304B03202220	PRODUCTOS_PROVISTOS	TABLE	UNIQUE	DISABLED	
5	ISIS2304B03202220	SYS_C00756337	NORMAL	ISIS2304B03202220	USUARIO	TABLE	UNIQUE	DISABLED	
6	ISIS2304B03202220	PK	NORMAL	ISIS2304B03202220	PRODUCTOS_COMPRA	TABLE	UNIQUE	DISABLED	
7	ISIS2304B03202220	SYS_C00756308	NORMAL	ISIS2304B03202220	PEDIDO	TABLE	UNIQUE	DISABLED	
8	ISIS2304B03202220	A_BAR_PK	NORMAL	ISIS2304B03202220	A_BAR	TABLE	UNIQUE	DISABLED	
9	ISIS2304B03202220	SYS_C00756318	NORMAL	ISIS2304B03202220	CONTENEDOR	TABLE	UNIQUE	DISABLED	
10	ISIS2304B03202220	UN_TIPOBEB_NOMBRE	NORMAL	ISIS2304B03202220	A_TIPOBEBIDA	TABLE	UNIQUE	DISABLED	
11	ISIS2304B03202220	SYS_C00756355	NORMAL	ISIS2304B03202220	PRODUCTO	TABLE	UNIQUE	DISABLED	
12	ISIS2304B03202220	SYS_C00756287	NORMAL	ISIS2304B03202220	ROL	TABLE	UNIQUE	DISABLED	
13	ISIS2304B03202220	A_VISITAN_PK	NORMAL	ISIS2304B03202220	A_VISITAN	TABLE	UNIQUE	DISABLED	
14	ISIS2304B03202220	A_TIPOBEBIDA_PK	NORMAL	ISIS2304B03202220	A_TIPOBEBIDA	TABLE	UNIQUE	DISABLED	
15	ISIS2304B03202220	A_BEBEDOR_PK	NORMAL	ISIS2304B03202220	A_BEBEDOR	TABLE	UNIQUE	DISABLED	
16	ISIS2304B03202220	PK_SILLASRESERVAS	NORMAL	ISIS2304B03202220	SILLASRESERVAS	TABLE	UNIQUE	DISABLED	
17	ISIS2304B03202220	SYS_C00756284	NORMAL	ISIS2304B03202220	DETALLES_PRODUCTO	TABLE	UNIQUE	DISABLED	
18	ISIS2304B03202220	SYS_C00756286	NORMAL	ISIS2304B03202220	ROL	TABLE	UNIQUE	DISABLED	
19	ISIS2304B03202220	SYS_C00769960	NORMAL	ISIS2304B03202220	PEDIDOS_PROVEEDOR	TABLE	UNIQUE	DISABLED	
20	ISIS2304B03202220	SYS_C00772118	NORMAL	ISIS2304B03202220	PROMOCION	TABLE	UNIQUE	DISABLED	
21	ISIS2304B03202220	SYS_C00756294	NORMAL	ISIS2304B03202220	MEDICION_PRODUCTO	TABLE	UNIQUE	DISABLED	
22	ISIS2304B03202220	SYS_C00771007	NORMAL	ISIS2304B03202220	COMPRA	TABLE	UNIQUE	DISABLED	
23	ISIS2304B03202220	SYS_C00756302	NORMAL	ISIS2304B03202220	PROVEEDOR	TABLE	UNIQUE	DISABLED	
24	ISIS2304B03202220	SYS_C00756328	NORMAL	ISIS2304B03202220	REVIEW_PEDIDO	TABLE	UNIQUE	DISABLED	
25	ISIS2304B03202220	SYS_C00756323	NORMAL	ISIS2304B03202220	ALMACENAMIENTO_PRODUCTO	TABLE	UNIQUE	DISABLED	
26	ISIS2304B03202220	SYS_C00756299	NORMAL	ISIS2304B03202220	SUCURSAL	TABLE	UNIQUE	DISABLED	
27	ISIS2304B03202220	PK_RESERVAS	NORMAL	ISIS2304B03202220	RESERVAS	TABLE	UNIQUE	DISABLED	

Para los requerimientos con rangos de fechas nos hubiera gustado utilizar índices secundarios que permitan filtrar por buckets de información los registros, pero descartamos la idea por el espacio que conllevaría esto. De resto, nos parece que Oracle, al ser un lenguaje declarativo, nos otorgó índices bastante útiles para nuestros requerimientos funcionales de consulta, por lo que no requerimos la creación de nuevos índices.

## 2.3 Resultados logrados

Se lograron generar no solamente un script de generación de datos el cual nos diese resultados en la forma de tuplas utilizando varias librerías de Python, y junto con eso y SQL Loader, se expandió considerablemente el número de registros en la base de datos. Dado que esto nos permitía revisar la ejecución en base de datos de las consultas, a la hora de crear RFC 10, 11, y 13 pudimos comprobar su eficiencia dadas tablas que, en conjunto, estaban rondando el millón de registros. Con ninguna sobrepasando los 30 segundos de ejecución sobre el tiempo de prueba dado. En general el propósito principal de esta iteración se logró cumplir satisfactoriamente.

## 2.4 Estructura del proyecto

- En `docs/javadocs/` podrá encontrar la documentación autogenerada del proyecto JDO.
- En `docs/modelos/` podrá encontrar el modelo relacional y conceptual utilizado para este proyecto.
- En `docs/sql/create-tables/` podrá encontrar todas las sentencias de creación de tablas.
- En `docs/sql/data` podrá encontrar sentencias de limpieza de base de datos y scripts de creación del esquema completo.
- En `docs/sql/data-population` podrá encontrar sentencias de inserción de registros de prueba para poblar la base de datos y soportar los requerimientos funcionales.

- En `docs/sql/rfc` podrá encontrar sentencias de consulta para soportar los requerimientos funcionales de consulta.
- En `src/` podrá encontrar el código fuente del proyecto.
- En `docs/sql-generator/src` podrá encontrar el script de generación de datos masivos.
- En `docs/sql-generator/build` podrá encontrar archivos CSV generados para poblar la base de datos con alrededor 1 millón de datos.

## 2.5 Resultados no logrados

A pesar de que se realizaron todas las consultas SQL necesarias para ejecutar en su totalidad los requerimientos de esta iteración, no se logró crear todos los métodos necesarios para visualizar su funcionamiento en la interfaz gráfica. Se crearon muchos de los métodos requeridos para satisfacer los requerimientos en la interfaz, interfazSuperAndes, negocio/SuperAndes, la mayoría en PersistenciaSuperAndes, y únicamente quedó faltando los SQL<Entidad> que se encargaban de ejecutar las sentencias SQL ya creadas en docs. Además de esto, aunque se lograron consultas parciales de RFC12 las cuáles respondían partes de la pregunta formulada, el sistema de queries de Oracle no nos permitió unir dichas consultas en una sola query, por lo que en el documento sql y en las query, quedaron separados.

## 2.6 Balance de pruebas

A pesar de que no se realizaron pruebas unitarias o manejo de QA como se habría esperado del enunciado, se pudo generar una base de datos la cual nos permitiera correr los query especificados con datos relativamente realistas, y a la hora de probar los datos generados por esta mock data y compararlos con lo que nos debería dar una consulta de este tipo, se estableció que las consultas SQL realizadas funcionaban de forma adecuada para solucionar los problemas establecidos. Y esta carga de datos nos implicó el manejo de tuplas con los datos y condiciones establecidas, a partir de los cuales se puede establecer que las pruebas de integridad, de ser construidas formalmente, funcionarían prácticamente a la perfección.

## 3 Planes de ejecución

### 3.1 RFC10

Resultado de la Consulta x Explicación del Plan x

SQL 0,27 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			4319	386
HASH			4319	386
HASH JOIN		GROUP BY	4319	385
Access Predicates				
PRODUCTOS_COMPRA.ID_COMPRA=COMPRA.ID				
NESTED LOOPS			4319	385
NESTED LOOPS				
STATISTICS COLLECTOR				
HASH JOIN			1245	380
Access Predicates				
USUARIO.NUMERO_DOCUMENTO=COMPRA.COMPRADOR				
NESTED LOOPS			1245	380
STATISTICS COLLECTOR				
TABLE ACCESS COMPRA		FULL	1245	166
Filter Predicates				
AND				
COMPRA.FECHA_COMPRA>=TO_DATE(' 2020-01-01 00:00:00', 'yyyy-mm-dd hh24:mi:ss')				
COMPRA.ESTADO_COMPRA='PAGADA'				
COMPRA.FECHA_COMPRA<=TO_DATE(' 2022-01-01 00:00:00', 'yyyy-mm-dd hh24:mi:ss')				
TABLE ACCESS USUARIO		BY INDEX ROWID	1	214
INDEX SYS_C00756332		UNIQUE SCAN		
Access Predicates				
USUARIO.NUMERO_DOCUMENTO=COMPRA.COMPRADOR				
TABLE ACCESS USUARIO		FULL	63049	214
INDEX PK		RANGE SCAN		
Access Predicates				
PRODUCTOS_COMPRA.ID_COMPRA=COMPRA.ID				
TABLE ACCESS PRODUCTOS_COMPRA		BY INDEX ROWID	3	5
TABLE ACCESS PRODUCTOS_COMPRA		FULL	4320	5

### 3.2 RFC11

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			4320	386
HASH JOIN			4320	386
Access Predicates				
USUARIO.NUMERO_DOCUMENTO=COMPRA.COMPRADOR				
NESTED LOOPS			4320	386
STATISTICS COLLECTOR				
HASH JOIN			4320	171
Access Predicates				
PRODUCTOS_COMPRA.ID_COMPRA=COMPRA.ID				
NESTED LOOPS			4320	171
STATISTICS COLLECTOR				
TABLE ACCESS	PRODUCTOS_COMPRA	FULL	4320	5
CQMPBA		BY INDEX ROWID		
INDEX	SYS_C00771007	UNIQUE SCAN	1	166
Access Predicates				
PRODUCTOS_COMPRA.ID_COMPRA=COMPRA.ID				
TABLE ACCESS	COMPRA	FULL	98006	166
SYS_C00756337		UNIQUE SCAN		
Access Predicates				
USUARIO.NUMERO_DOCUMENTO=COMPRA.COMPRADOR				
TABLE ACCESS	USUARIO	BY INDEX ROWID	1	214
USUARIO		FULL	63049	214

### 3.3 RFC12

sistrans-connection   Import-productos-csv-bad_2022.12.04-16.49.32.sql   Import-productos-csv-bad_2022.12.04-16.56.58.sql   A_BAR...				
4.11600018 seconds				
Worksheet   Query Builder				
<pre>SELECT NUM_COMPRAS AS VALORES_MINIMOS, ID_PRODUCTO AS PRODUCTO_MINIMO, ID_SUCURSAL FROM (   SELECT SEMANA AS SEMANA_CONSULTA,   MIN(COMPRAS) AS NUM_COMPRAS,   ID_SUCURSAL AS ID   FROM PRODUCTOS_COMPRADOS   GROUP BY SEMANA,   ID_SUCURSAL   ORDER BY SEMANA ) INNER JOIN (   SELECT *   FROM PRODUCTOS_COMPRADOS ) ON NUM_COMPRAS = COMPRAS AND ID_SUCURSAL = ID ;</pre>				
Script Output   Query Result   Explain Plan				
SQL   4.116 seconds				
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			43	778
HASH JOIN			43	778
Access Predicates				
AND				
NUM_COMPRAS=COMPRAS				
ID_SUCURSAL=ID				
VIEW			1000	390
SORT		ORDER BY	1000	390
HASH		GROUP BY	1000	390
VIEW	SYS.VM_NWWW_0		4320	388
HASH		GROUP BY	4320	388
NESTED LOOPS		SEMI	4320	387
VIEW			4320	387
HASH		GROUP BY	4320	387
HASH JOIN			4320	386
Access Predicates				
USUARIO.NUMERO_DOCUMENTO=COMPRA.COMPRADOR				
HASH			4320	171
Access Predicates				

sistrans-connection | Import-productos-csv-bad\_2022.12.04-16.49.32.sql | Import-productos-csv-bad\_2022.12.04-16.56.58.sql | A\_BAR... | 3.11400008 seconds

Worksheet: Explain Plan... (F10)

```

SELECT SEMANA_CONSULTA,
NUM_COMPRAS AS VALORES_MAXIMOS,
ID_PRODUCTO AS PRODUCTO_MAXIMO,
ID_SUCURSAL
FROM (

```

Script Output | Query Result | Explain Plan | 3.114 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				778
HASH JOIN				778
Access Predicates				
AND				
NUM_COMPRAS=COMPRAS				
ID_SUCURSAL=ID				
VIEW				390
SORT		ORDER BY		390
HASH		GROUP BY		390
VIEW	SYS.VM_NWVW_0		4320	388
HASH		GROUP BY	4320	388
NESTED LOOPS		SEMI	4320	387
VIEW			4320	387
HASH		GROUP BY	4320	387
HASH JOIN			4320	386
Access Predicates				
USUARIO.NUMERO_DOCUMENTO=COMPRA.COMPRADOR				
HASH			4320	171
Access Predicates				
PRODUCTOS_COMPRA.ID_COMPRA=COMPRA.ID				
TA PRODUCTOS_COMPRA		FULL	4320	5
TA COMPRA		FULL	98006	166
TABLE USUARIO		FULL	63049	214
INDEX	SYS_C00756355	UNIQUE SCAN	98006	0
Access Predicates				
CODIGO_BARRAS=ID_PRODUCTO				
VIEW			4320	388

### 3.4 RFC13

Resultado de la Consulta | Explicación del Plan | 0,305 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				670
HASH JOIN				670
Access Predicates				
USUARIO.NUMERO_DOCUMENTO=COMPRA.COMPRADOR				
HASH JOIN				456
Access Predicates				
PRODUCTOS_COMPRA.ID_PRODUCTO=PRODUCTO.CODIGO_BARRAS				
Filter Predicates				
OR				
COMPRA.VALOR_COMPRA_TOTAL >= 100000				
PRODUCTO.CATEGORIA='Herramienta'				
PRODUCTO.CATEGORIA='Tecnologia'				
NESTED LOOPS				456
NESTED LOOPS				
STATISTICS COLLECTOR				
HASH JOIN			4320	171
Access Predicates				
PRODUCTOS_COMPRA.ID_COMPRA=COMPRA.ID				
TABLE ACCESS	PRODUCTOS_COMPRA	FULL	4320	5
TABLE ACCESS	COMPRA	FULL	32824	166
Filter Predicates				
COMPRA.ESTADO_COMPRA='PAGADA'				
INDEX	SYS_C00756355	UNIQUE SCAN		
Access Predicates				
PRODUCTOS_COMPRA.ID_PRODUCTO=PRODUCTO.CODIGO_BARRAS				
TABLE ACCESS	PRODUCTO	BY INDEX ROWID	1	284
Filter Predicates				
OR				
COMPRA.VALOR_COMPRA_TOTAL >= 100000				
PRODUCTO.CATEGORIA='Herramienta'				
PRODUCTO.CATEGORIA='Tecnologia'				
TABLE ACCESS	PRODUCTO	FULL	98006	284
TABLE ACCESS	USUARIO	FULL	63049	214

### 3.5 Efectos de la selectividad durante la consulta.

Página de bienvenida x RFC10.sql x RFC11.sql x RFC13.sql x

Hoja de Trabajo de SQL Historial

Hoja de Trabajo Generador de Consultas

```

) ON NUMERO_DOCUMENTO = COMPRADOR INNER JOIN (
  SELECT *
  FROM PRODUCTOS_COMPRA
) ON ID_COMPRA = ID )
WHERE FECHA_COMPRA BETWEEN '01-01-2020' AND
'01-01-2022' AND
ESTADO_COMPRA = 'PAGADA'
GROUP BY NUMERO_DOCUMENTO,
NOMBRE,
ID_COMPRA,
ID_PRODUCTO,
CANT_UNIDADES_COMPRADAS
/*
/SELECT
NUMERO_DOCUMENTO

```

Explicación del Plan x Resultado de la Consulta x

SQL | Se han recuperado 50 filas en 0,552 segundos

	NUMERO_DOCUMENTO	NOMBRE	COMPRAS_POR_USUARIO	ID_COMPRA	ID_PRODUCTO	CANT_UNIDADES_COMPRADAS
1	3896442104	Dyanna Dowthwaite	1	82	6372105410289965	1
2	1278995099	Giles Loftin	1	159	6372105410289965	4
3	628195931	Jannel Blincow	1	248	6372105410289965	2
4	9031101052	Sonnie Goodspeed	1	350	6372105410289965	4
5	1278995099	Giles Loftin	1	502	6372105410289965	2
6	8792260616	Gusty Ciani	1	568	6372105410289965	2
7	3896442104	Dyanna Dowthwaite	1	582	6372105410289965	3
8	9031101052	Sonnie Goodspeed	1	661	6372105410289965	4
9	1278995099	Giles Loftin	1	500	2296617651768	10

Página de bienvenida x RFC10.sql x RFC11.sql x RFC13.sql x

Hoja de Trabajo de SQL Historial

Hoja de Trabajo Generador de Consultas

```

SELECT *
FROM COMPRA
) ON NUMERO_DOCUMENTO = COMPRADOR INNER JOIN (
  SELECT *
  FROM PRODUCTOS_COMPRA
) ON ID_COMPRA = ID )
WHERE FECHA_COMPRA BETWEEN '01-01-2021' AND
'01-01-2022' AND
ESTADO_COMPRA = 'PAGADA'
GROUP BY NUMERO_DOCUMENTO,
NOMBRE,
ID_COMPRA,
ID_PRODUCTO,
CANT_UNIDADES_COMPRADAS
/*
/SELECT
NUMERO_DOCUMENTO

```

Explicación del Plan x Resultado de la Consulta x

SQL | Se han recuperado 50 filas en 1,713 segundos

	NUMERO_DOCUMENTO	NOMBRE	COMPRAS_POR_USUARIO	ID_COMPRA	ID_PRODUCTO	CANT_UNIDADES_COMPRADAS
1	3896442104	Dyanna Dowthwaite	1	82	6372105410289965	1
2	1278995099	Giles Loftin	1	159	6372105410289965	4
3	628195931	Jannel Blincow	1	248	6372105410289965	2
4	9031101052	Sonnie Goodspeed	1	350	6372105410289965	4
5	1278995099	Giles Loftin	1	502	6372105410289965	2
6	8792260616	Gusty Ciani	1	568	6372105410289965	2
7	3896442104	Dyanna Dowthwaite	1	582	6372105410289965	3
8	9031101052	Sonnie Goodspeed	1	661	6372105410289965	4
9	1278995099	Giles Loftin	1	500	2296617651768	10
10	1278995099	Giles Loftin	1	209	7166595780628	1
11	8792260616	Gusty Ciani	1	518	3758229693462	7
12	3896442104	Dyanna Dowthwaite	1	84	5947382929268	18
13	3896442104	Dyanna Dowthwaite	1	377	1065366470842	20
14	1278995099	Giles Loftin	1	370	7656221265105	19
15	9031101052	Sonnie Goodspeed	1	108	3441131596838	7



Hoja de Trabajo    Generador de Consultas <pre> ) ON NUMERO_DOCUMENTO = COMPRADOR INNER JOIN (   SELECT *   FROM PRODUCTOS_COMPRA ) ON ID_COMPRA = ID ) WHERE FECHA_COMPRA BETWEEN '01-06-2021' AND '01-01-2022' AND ESTADO_COMPRA = 'PAGADA' GROUP BY NUMERO_DOCUMENTO, NOMBRE, ID_COMPRA, ID_PRODUCTO, CANT_UNIDADES_COMPRADAS /* SELECT   NUMERO_DOCUMENTO, </pre>						
Explicación del Plan x    Resultado de la Consulta x SQL   Se han recuperado 50 filas en 2,423 segundos						
	NUMERO_DOCUMENTO	NOMBRE	COMPRAS_POR_USUARIO	ID_COMPRA	ID_PRODUCTO	CANT_UNIDADES_COMPRADAS
1	3896442104	Dyanna Dowthwaite	1	82	6372105410289965	1
2	1278995099	Giles Loftin	1	159	6372105410289965	4
3	628195931	Jannel Blincow	1	248	6372105410289965	2
4	9031101052	Sonnie Goodspeed	1	350	6372105410289965	4
5	1278995099	Giles Loftin	1	502	6372105410289965	2
6	8792260616	Gusty Ciani	1	568	6372105410289965	2
7	3896442104	Dyanna Dowthwaite	1	582	6372105410289965	3
8	9031101052	Sonnie Goodspeed	1	661	6372105410289965	4
9	1278995099	Giles Loftin	1	500	2296617651768	10
10	1278995099	Giles Loftin	1	209	7166595780628	1
11	8792260616	Gusty Ciani	1	518	3758229693462	7
12	3896442104	Dyanna Dowthwaite	1	84	5947382929268	18
13	3896442104	Dyanna Dowthwaite	1	377	1065366470842	20
14	1278995099	Giles Loftin	1	370	7656221265105	19
15	9031101052	Sonnie Goodspeed	1	108	3441131596838	7
16	9031101052	Sonnie Goodspeed	1	280	5823446479798	11
17	1278995099	Giles Loftin	1	239	4482841616675	18

A la hora de establecer el impacto de la selectividad con respecto a la ejecución, utilizamos el RFC10 para medir el impacto que una mayor o menor selectividad sobre los datos puede tener durante el tiempo de ejecución.

Utilizando las fechas de parámetro requeridas para la ejecución de la consulta como la variable a modificar, tomando un rango de fechas mayor (que conduce a que la selectividad disminuya) o menor (que conduce a que la selectividad aumente).

Dados estos resultados, concluimos que entre mayor la selectividad del rango, mayor el tiempo de consulta, esto lo podemos explicar a partir del algoritmo interno de consulta usado por Oracle. Pues este parece estar haciendo una mayor cantidad de comparaciones al aumentar la selectividad, intentando conseguir el rango de fechas estipulado.

Sin embargo, es posible que estos datos también se vean afectados por registros en caché del computador o directamente por ejecuciones hechas en sistemas operativos o entornos de desarrollo diferentes



## 4 Consultas SQL

### 4.1 RFC10

```
SELECT
    NUMERO_DOCUMENTO,
    NOMBRE,
    COUNT(NOMBRE) AS COMPRAS_POR_USUARIO,
    ID_COMPRA,
    ID_PRODUCTO,
    CANT_UNIDADES_COMPRADAS
FROM
    ( (
        SELECT
            *
        FROM
            USUARIO
    ) INNER JOIN (
        SELECT
            *
        FROM
            COMPRA
    ) ON NUMERO_DOCUMENTO = COMPRADOR INNER JOIN (
        SELECT
            *
        FROM
            PRODUCTOS_COMPRA
    ) ON ID_COMPRA = ID )
WHERE
    FECHA_COMPRA BETWEEN ?
    AND ?
    AND ESTADO_COMPRA = 'PAGADA'
    AND CANT_UNIDADES_COMPRADAS = ?
GROUP BY
    NUMERO_DOCUMENTO,
    NOMBRE,
    ID_COMPRA,
    ID_PRODUCTO,
```

CANT\_UNIDADES\_COMPRADAS

## 4.2 RFC11

```
SELECT
    *
FROM
    (
        SELECT
            *
        FROM
            COMPRAS_USUARIO
        WHERE
            NUMERO_DOCUMENTO NOT IN (
                SELECT
                    NUMERO_DOCUMENTO
                FROM
                    USUARIOS_CON_COMPRA
                WHERE
                    FECHA_COMPRA BETWEEN ?
                    AND ?
                    AND ESTADO_COMPRA = 'PAGADA'
            )
    );
```

## 4.3 RFC12

```
CREATE VIEW PRODUCTOS_COMPRADOS AS (
    SELECT SEMANA,
        ID_PRODUCTO,
        COMPRAS,
        ID_SUCURSAL FROM ( (SELECT TRUNC(FECHA_COMPRA, 'IW') AS SEMANA,
ID_PRODUCTO, COUNT(ID_PRODUCTO) AS COMPRAS, ID_SUCURSAL FROM
USUARIOS_CON_COMPRA GROUP BY FECHA_COMPRA, ID_PRODUCTO, ID_SUCURSAL
ORDER BY FECHA_COMPRA) INNER JOIN (SELECT CODIGO_BARRAS, NOMBRE FROM
PRODUCTO) ON CODIGO_BARRAS = ID_PRODUCTO ) GROUP BY SEMANA,
        ID_PRODUCTO,
        COMPRAS,
        ID_SUCURSAL
    );

SELECT *
FROM PRODUCTOS_COMPRADOS;
```

```

SELECT SEMANA_CONSULTA,
       NUM_COMPRAS AS VALORES_MAXIMOS,
       ID_PRODUCTO AS PRODUCTO_MAXIMO,
       ID_SUCURSAL
FROM ( (
        SELECT SEMANA          AS SEMANA_CONSULTA,
               MAX(COMPRAS) AS NUM_COMPRAS,
               ID_SUCURSAL AS ID
        FROM PRODUCTOS_COMPRADOS
        GROUP BY SEMANA,
               ID_SUCURSAL
        ORDER BY SEMANA
    ) INNER JOIN (
        SELECT *
        FROM PRODUCTOS_COMPRADOS
    ) ON NUM_COMPRAS = COMPRAS AND
       ID_SUCURSAL = ID );

```

```

SELECT NUM_COMPRAS AS VALORES_MINIMOS,
       ID_PRODUCTO AS PRODUCTO_MINIMO,
       ID_SUCURSAL
FROM ( (
        SELECT SEMANA          AS SEMANA_CONSULTA,
               MIN(COMPRAS) AS NUM_COMPRAS,
               ID_SUCURSAL AS ID
        FROM PRODUCTOS_COMPRADOS
        GROUP BY SEMANA,
               ID_SUCURSAL
        ORDER BY SEMANA
    ) INNER JOIN (
        SELECT *
        FROM PRODUCTOS_COMPRADOS
    ) ON NUM_COMPRAS = COMPRAS AND
       ID_SUCURSAL = ID );

```

#### 4.4 RFC13

```

SELECT
    *

```

```

FROM
    (
        SELECT
            *
        FROM
            (
                SELECT
                    *
                FROM
                    USUARIOS_CON_COMPRA
            )
        INNER JOIN (
            SELECT
                *
            FROM
                PRODUCTO
        )
        ON ID_PRODUCTO = CODIGO_BARRAS
    WHERE
        ESTADO_COMPRA = 'PAGADA'
    )
WHERE
    VALOR_COMPRA_TOTAL >= 100000
    OR CATEGORIA IN ('Tecnologia',
        'Herramienta')

```

## 5 Diseño de datos

En iteraciones anteriores, utilizábamos la herramienta de Mockaroo para generar alrededor de 1000 registros por tabla, pero dado las limitaciones de las cuentas gratuitas, decidimos crear nuestro propio generador de datos para esta iteración.

Para generar un volumen de datos que nos permita verificar en un escenario real la eficiencia de nuestros requerimientos, se creó un script de Python en el que se indica el número de registros a crear por tabla para que se generen datos aleatorios utilizando la librería de Faker.

Cada tabla tiene alrededor de 100 000 registros para que de este modo nuestra base de datos tuviese alrededor de 1 millón de registros.

En un inicio, se establecieron las columnas o encabezados de la tabla en un array, y luego, por cada iteración se generaba un array de información falsa pero coherente a la tabla, representando un nuevo registro. A continuación se puede evidenciar un ejemplo de esto:

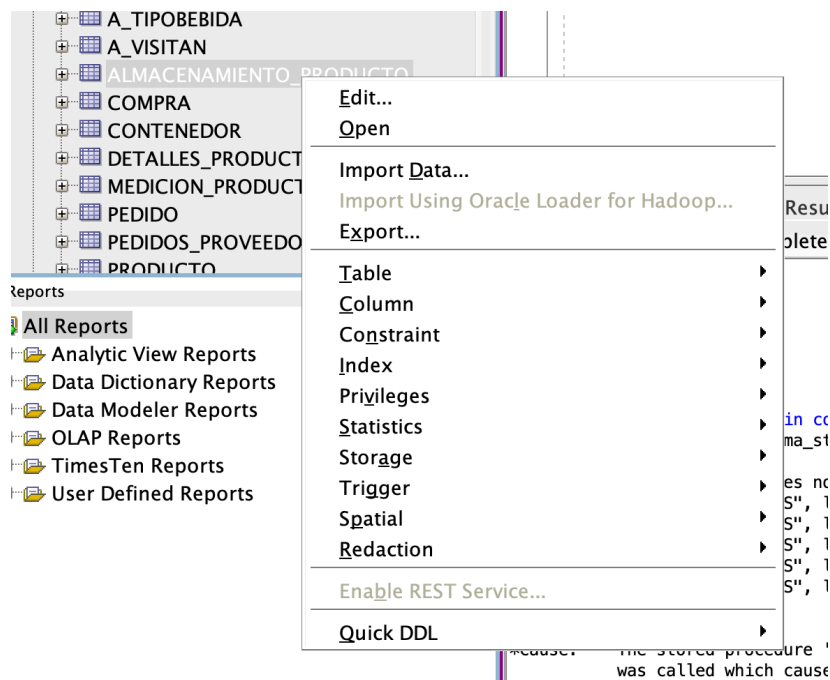
```
def generate_sucursal():
    ret = []
    ret.append([
        "ID",
        "CIUDAD",
        "DIRECCION",
        "NOMBRE",
        "SIZE_SUCURSAL",
        "SEGMENTO_OBJETIVO"
    ])
    for i in range(NUM_ENTRIES):
        ret.append([
            str(i),
            fake.city(),
            fake.address().replace('\n', ' '),
            fake.street_name(),
            random.choice(["Small", "Medium", "Big"]),
            fake.text(max_nb_chars=10)
        ])
    return numpy.array(ret, dtype=object)
```

Después de esto, convertíamos los arrays nativos de Python a unos de numpy para que, cuando todas las tablas hayan generado sus datos, se escriba fácilmente a archivos CSV, desde un diccionario que contiene los datos de todas las tablas y los nombres de los archivos.

```
def array_to_csv(matrix):
    for item, array in matrix.items():
        print(f"Writing {item}...")
        with open(f'../build/{item}.csv', 'w+', newline='') as file:
            mywriter = csv.writer(file, delimiter=',')
            mywriter.writerows(array)
```

Estos archivos se exportaban a el directorio `sql-generator/build/` y fueron los que posteriormente utilizamos para cargar la información a nuestra base de datos de Oracle.

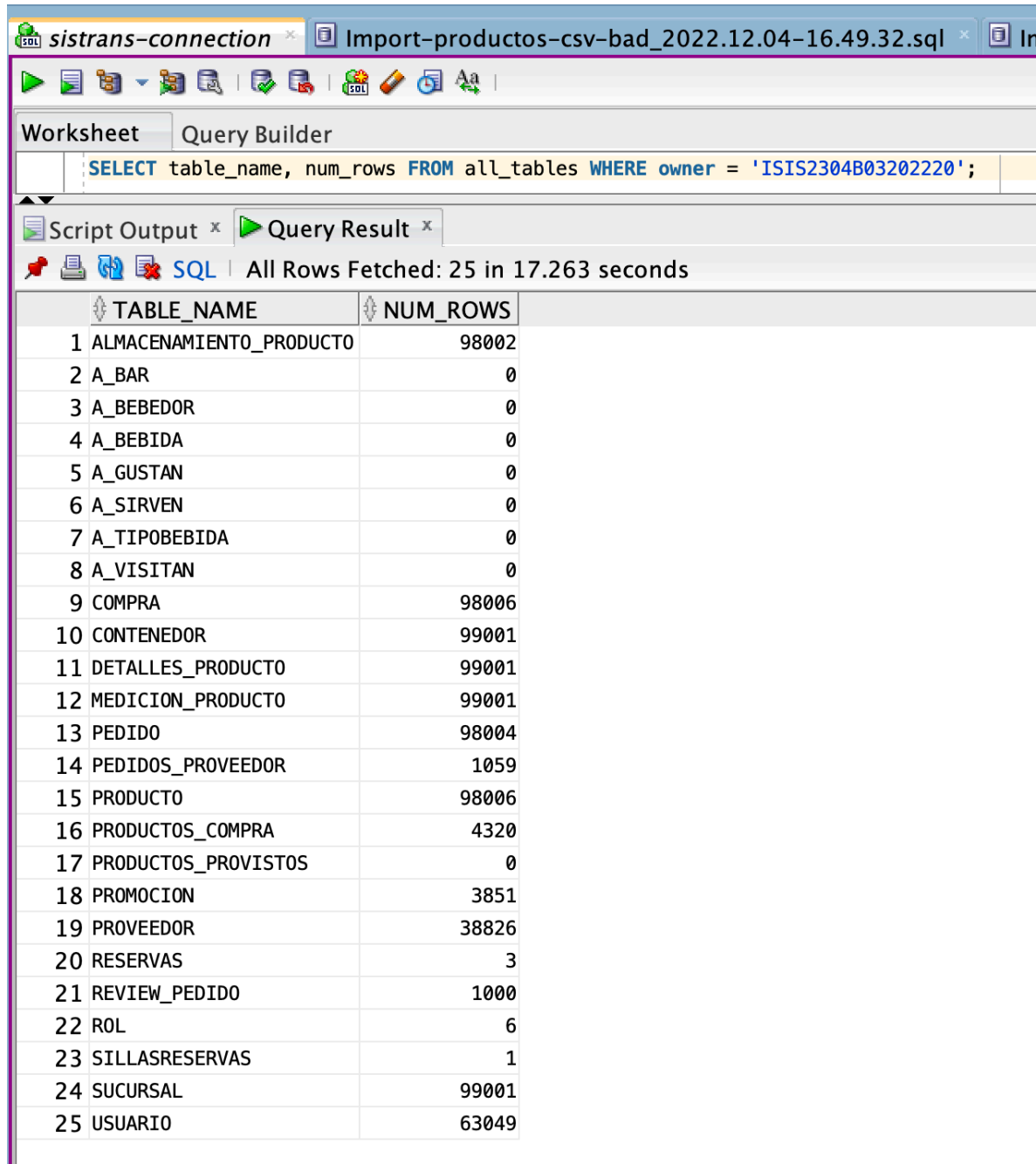
En SQL Developer, utilizamos la opción de importar datos desde un archivo local, y por cada tabla, se insertaron los archivos CSV generados anteriormente.



Finalmente, nuestra base de datos quedo con más de 900 000 registros en total, lo que es un número apropiado para las pruebas de eficiencia, sin llegar a sobrecargar nuestro sistema.

A medida que se insertaban más datos respecto al 20%, se demoraba más la inserción de los datos y por lo general aumentaban los conflictos.

En algunas tablas, fue un poco complicado sincronizar las llaves foráneas generadas con otras tablas que lo requieran, pero lo solucionamos parcialmente al generar las llaves primarias de estas a nivel global del generador de datos. A continuación, se muestran las estadísticas de las tablas relacionadas a nuestro usuario.



Worksheet Query Builder

```
SELECT table_name, num_rows FROM all_tables WHERE owner = 'ISIS2304B03202220';
```

Script Output x Query Result x

SQL | All Rows Fetched: 25 in 17.263 seconds

TABLE_NAME	NUM_ROWS
1 ALMACENAMIENTO_PRODUCTO	98002
2 A_BAR	0
3 A_BEBEDOR	0
4 A_BEBIDA	0
5 A_GUSTAN	0
6 A_SIRVEN	0
7 A_TIPOBEBIDA	0
8 A_VISITAN	0
9 COMPRA	98006
10 CONTENEDOR	99001
11 DETALLES_PRODUCTO	99001
12 MEDICION_PRODUCTO	99001
13 PEDIDO	98004
14 PEDIDOS_PROVEEDOR	1059
15 PRODUCTO	98006
16 PRODUCTOS_COMPRA	4320
17 PRODUCTOS_PROVISTOS	0
18 PROMOCION	3851
19 PROVEEDOR	38826
20 RESERVAS	3
21 REVIEW_PEDIDO	1000
22 ROL	6
23 SILLASRESERVAS	1
24 SUCURSAL	99001
25 USUARIO	63049

## 6 Planes de optimización

### 6.1 Análisis de ejecución de consultas entre aplicaciones

Analice la diferencia entre la ejecución de consultas delegada al manejador de bases de datos como Oracle y compárelo con una ejecución donde la aplicación trae los datos a memoria principal y resuelve con instrucciones de control (if, while, etc.), los operadores involucrados en las consultas como joins, selecciones y proyecciones.

R// La principal información entregada en este resultado es que la ejecución del programa se haría con un lenguaje de programación funcional, no declarativo.

Esto es importante porque implica que esta diferencia de ejecución se verá afectada no solamente por este hecho, sino también porque tendríamos un control mucho más granular sobre la forma en la cual manejamos estas consultas, es importante entender que Oracle es de código cerrado, por lo que no podemos hacer comparaciones exactas sobre que podríamos mejorar sobre su codebase, pero es posible que con optimizaciones como el manejo de concurrencia, la escritura de código elegante y bastante más especializado con respecto a estas consultas o incluso, con la opción de utilizar un código de relativamente bajo nivel, como Rust o C++, podamos mejorar los tiempos de consulta sobre Oracle.

Además, podemos ser mucho más específicos en nuestras optimizaciones de acuerdo con el caso de uso o situación problema en la que nos encontremos.

## 7 Conclusiones

Durante esta iteración, aprendimos a trabajar con grandes volúmenes de datos para reflexionar sobre la eficiencia y optimización de los requerimientos de nuestro negocio, ya sea utilizando índices, planes de ejecución o distribuciones de datos en específicos.

También, observamos como la selectividad de nuestras consultas influye en el rendimiento de la ejecución de operaciones, y el impacto que tiene el uso de memoria secundaria con mayor frecuencia.

En general, aunque hubo algunos problemas uniendo requerimientos funcionales de consulta y manejos de algunas partes de la ejecución, en general gran parte de los requerimientos necesarios y de las consultas pedidas fueron generadas correctamente.

No hubo mayores cambios a la estructura de la base de datos debido a el enfoque en mantenibilidad y flexibilidad que se tuvo durante las entregas de anteriores iteraciones, y entonces, aunque no todas las partes de los entregables están necesariamente pulidas al nivel de anteriores iteraciones, en general esta fue una iteración exitosa del proyecto.

## 8 Bibliografía

1. *Universidad de los Andes*. [En línea] [Citado el: 1 de Septiembre de 2022.] [https://bloqueneon.uniandes.edu.co//content/enforced/140479-UN\\_202220\\_ISIS2304\\_I/Proyecto/ST-Pry-DocMarco-It1.pdf?isCourseFile=true&\\_&d2lSessionVal=ZXW7mHJ8YHBaf9ZZ0mPgH8UUe&ou=140479](https://bloqueneon.uniandes.edu.co//content/enforced/140479-UN_202220_ISIS2304_I/Proyecto/ST-Pry-DocMarco-It1.pdf?isCourseFile=true&_&d2lSessionVal=ZXW7mHJ8YHBaf9ZZ0mPgH8UUe&ou=140479).