

Tarea 1 - Consultas

IIC2440 Procesamiento de Datos Masivos

Maria Jose Ortega Rosales y Pedro Pablo Zavala Tejos

1. Verificación de nuestro modelo de base de datos

Para asegurar que nuestro modelo está bien implementado verificamos directamente en el archivo *json* con un ejemplo en específico. En este caso, consultamos lo siguiente:

```
SELECT *
FROM `iic2440.ordenes_compra`
WHERE id_orden = 181 -- "compra_id: 181"
AND id_region = 1 -- "Region 2 (ordenes_r2.json)"
AND id_usuario = 1 -- "usuario_id: 1"
AND EXTRACT(MONTH FROM fecha) = 1 -- "Ordenes en enero"
```

Donde la respuesta de la consulta fue que para el usuario 1, el 1 de enero del 2023 compro 10 manzanas (id = 1) y 7 melones (id = 4).

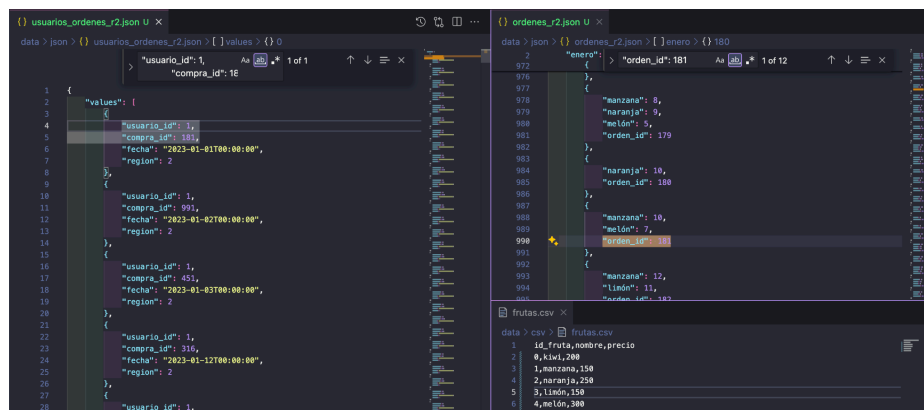


Resultados de la consulta

INFORMACIÓN DEL TRABAJO		RESULTADOS		GRÁFICO	JSON	DETALLES DE LA EJECUCIÓN		GRÁFICO DE EJECUCIÓN
Fila	id_usuario	id_orden	id_region	fecha	id_fruta	cantidad		
1	1	181	1	2023-01-01	1	10		
2	1	181	1	2023-01-01	4	7		

Figura 4. Ejemplo de Testing del modelo

Si analizamos los archivos *usuarios_ordenes_r2.json* y *ordenes_r2.json*, podemos ver claramente que al buscar el usuario con id igual a 1, su compra realizada el 1 de enero contiene exactamente las mismas frutas mostradas en la tabla de la consulta. Con esto, logramos comprobar que nuestro modelo de base de datos está bien implementado y los datos fueron bien procesados.



usuarios_ordenes_r2.json

```
{
  "values": [
    {
      "usuario_id": 1,
      "compra_id": 181,
      "fecha": "2023-01-01T00:00:00",
      "region": 2
    },
    {
      "usuario_id": 1,
      "compra_id": 991,
      "fecha": "2023-01-02T00:00:00",
      "region": 2
    },
    {
      "usuario_id": 1,
      "compra_id": 451,
      "fecha": "2023-01-03T00:00:00",
      "region": 2
    },
    {
      "usuario_id": 1,
      "compra_id": 316,
      "fecha": "2023-01-12T00:00:00",
      "region": 2
    }
  ]
}
```

ordenes_r2.json

```
{
  "enero": [
    {
      "manzana": 8,
      "naranja": 9,
      "melón": 5,
      "orden_id": 179
    },
    {
      "manzana": 10,
      "orden_id": 180
    },
    {
      "manzana": 10,
      "melón": 7,
      "orden_id": 181
    },
    {
      "manzana": 12,
      "limón": 11,
      "orden_id": 182
    }
  ]
}
```

frutas.csv

id_fruta	nombre	precio
0	kiwi	200
1	manzana	150
2	narana	250
3	limón	150
4	melón	300

Figura 5. Comprobación de nuestro modelos en archivos json.

2. Consultas analíticas

2.1 Consultas y respuestas

2.1.1 Consulta 1

```
/*
Dada una region, entrega el numero de unidades vendidas al mes para cada fruta.
*/

WITH ordenes_reg_1 AS (
  SELECT id_usuario,
         id_orden,
         id_region,
         id_fruta,
         EXTRACT(MONTH FROM fecha) as mes,
         cantidad
  FROM `iic2440.ordenes_compra`
  WHERE id_region = 0 -- Considerando la region 1 (ID = 0)
),
frutas_por_mes AS (
  SELECT mes, id_fruta,
         SUM(cantidad) OVER (item_window) AS unidades_vendidas,
         ROW_NUMBER() OVER (item_window) AS ranking
  FROM ordenes_reg_1
  QUALIFY ranking = 1
  WINDOW item_window AS (PARTITION BY mes, id_fruta ORDER BY mes ASC)
)

SELECT mes, id_fruta, unidades_vendidas
FROM frutas_por_mes
```

2.1.2 Consulta 2

```
/*  
Dado un cliente, entrega como ha evolucionado el dinero gastado por el cliente en  
el tiempo.  
*/
```

```
WITH gastos_por_orden AS (  
  SELECT  
    ordenes_compra.id_usuario,  
    ordenes_compra.fecha,  
    Frutas.id_fruta,  
    ordenes_compra.cantidad * frutas.precio AS dinero_gastado  
  FROM `iic2440.ordenes_compra` AS ordenes_compra  
  JOIN `iic2440.Frutas` AS frutas  
  ON ordenes_compra.id_fruta = frutas.id_fruta  
  WHERE id_usuario = 1 -- Para el cliente con ID = 1  
)  
evolucion_gastos AS (  
  SELECT *,  
  SUM(dinero_gastado) OVER (item_window) AS gasto_diario,  
  ROW_NUMBER() OVER (item_window) AS ranking  
  FROM gastos_por_orden  
  QUALIFY ranking = 1  
  WINDOW item_window AS (PARTITION BY fecha ORDER BY fecha ASC )  
)
```

```
/*  
Hacemos un qualify, ya que obtenemos la suma del dinero gastado por dia.  
Al hacer el SUM, obtenemos tuplas repetidas, con los mismos valores.  
Por lo tanto, escogemos la primera para evitar duplicados.  
*/
```

```
SELECT id_usuario, fecha, gasto_diario  
FROM evolucion_gastos
```

2.1.3 Consulta 3

```
/*Para cada region y mes, entrega la fruta mas vendida.*/
WITH ordenes_compra AS (
  SELECT id_region, frutas.id_fruta, nombre, precio, cantidad,
    EXTRACT(MONTH FROM fecha) AS mes
  FROM `iic2440.ordenes_compra`
  JOIN `iic2440.Frutas` AS frutas
  ON `iic2440.ordenes_compra`.id_fruta = frutas.id_fruta
),
cantidad_fruta_mes AS (
  SELECT *,
    SUM(cantidad) OVER (item_window) as cantidad_total,
  FROM ordenes_compra
  WINDOW item_window AS (PARTITION BY id_region, mes, id_fruta ORDER BY id_fruta DESC)
),
ranking_fruta_mes AS (
  SELECT *,
    ROW_NUMBER() OVER (
      PARTITION BY id_region, mes ORDER BY cantidad_total DESC
    ) AS ranking
  FROM cantidad_fruta_mes
  QUALIFY ranking = 1
)

SELECT id_region, mes, nombre, cantidad_total
FROM ranking_fruta_mes
```

2.1.4 Consulta 4

```
/*Calcula para cada semana el dinero que ha entrado a la tienda dicha semana. Luego
entrega el dinero entrante acumulado.*/
/*
Truncamos la semana al primer dia de la semana (asume domingo como primer dia).
Por ejemplo:
- Fecha original: 2023-12-23 (Sabado) -> Fecha truncada: 2023-12-17 (Domingo anterior)
*/
WITH ordenes_compra AS (
    SELECT
        DATE_TRUNC(fecha, WEEK) AS semana,
        precio * cantidad AS dinero_entrante
    FROM `iic2440.ordenes_compra`
    JOIN `iic2440.Frutas` AS frutas
    ON `iic2440.ordenes_compra`.id_fruta = frutas.id_fruta
),
dinero_por_semana AS (
    SELECT semana,
    SUM(dinero_entrante) OVER (item_window) AS dinero_entrante,
    FROM ordenes_compra
    QUALIFY (ROW_NUMBER() OVER (item_window) = 1)
    WINDOW item_window AS (PARTITION BY semana ORDER BY semana)
)

SELECT *,
    SUM(dinero_entrante) OVER (
        ORDER BY semana ASC
        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
    ) as dinero_acumulado
FROM dinero_por_semana
```

2.2 Resultados de cada consulta

2.2.1 Resultado consulta 1

🔍

2_3_consulta_1

▶

EJECUTAR

📄

GUARDAR LA CONSULTA (VERSIÓN CLÁSICA)

▼

2

Dada una region, entrega el numero de unidades vendidas al mes para cada fruta.

3

*/

4

5

WITH ordenes_reg_1 AS (

6

SELECT id_usuario,

7

id_orden,

8

id_region,

9

id_fruta,

Resultados de la consulta

INFORMACIÓN DEL TRABAJO

RESULTADOS

GRÁFICO

JSON

DETALLES DE LA E.

Fila	mes ▼	id_fruta ▼	unidades_vendidas
1	1	0	423
2	1	1	537
3	1	2	1185
4	1	3	1993
5	1	4	3792
6	2	0	413
7	2	1	546
8	2	2	1192
9	2	3	2012
10	2	4	2771

2.2.2 Resultado consulta 2

🔍	2_3_consulta_2	EJECUTAR	GUARDAR LA CONSULTA (VERSIÓN CLÁSICA) ▾	COMPARTIR ▾
1	/*			
2	Dado un cliente, entrega como ha evolucionado el dinero gastado por el cliente en el tiempo.			
3	*/			
4				
5	WITH gastos_por_orden AS (
6	SELECT			
7	ordenes_compra.id_usuario,			
8	ordenes_compra.fecha,			
9	Frutas.id_fruta,			
10	ordenes_compra.cantidad * frutas.precio AS dinero_gastado			
11	FROM `iic2440.ordenes_compra` AS ordenes_compra			
12	JOIN `iic2440.Frutas` AS frutas			
13	ON ordenes_compra.id_fruta = frutas.id_fruta			

Resultados de la consulta

INFORMACIÓN DEL TRABAJO		RESULTADOS	GRÁFICO	JSON	DETALLES DE LA EJECUCIÓN	GF
Fila	id_usuario ▾	fecha ▾	gasto_diario ▾			
1	1	2023-01-01	3600			
2	1	2023-01-02	2100			
3	1	2023-01-03	4650			
4	1	2023-01-07	2400			
5	1	2023-01-09	3750			
6	1	2023-01-12	4200			

2.2.3 Resultado consulta 3

🔍

2_3_consulta_3

EJECUTAR

GUARDAR LA CONSULTA (VERSIÓN CLÁSICA) ▾

```

1  /*Para cada region y mes, entrega la fruta mas vendida.*/
2  WITH ordenes_compra AS (
3    SELECT id_region, frutas.id_fruta, nombre, precio, cantidad,
4           | EXTRACT(MONTH FROM fecha) AS mes
5    FROM `iic2440.ordenes_compra`
6    JOIN `iic2440.Frutas` AS frutas
7    ON `iic2440.ordenes_compra`.id_fruta = frutas.id_fruta
8  ),
9  cantidad_fruta_mes AS (

```

Resultados de la consulta

INFORMACIÓN DEL TRABAJO

RESULTADOS

GRÁFICO

JSON

DETALLES DE LA E

Fila	id_region ▾	mes ▾	nombre ▾	cantidad_total ▾
1	1	4	manzana	6645
2	0	2	melón	3771
3	1	3	melón	6664
4	0	12	melón	3779
5	1	10	naranja	12606
6	0	6	naranja	4376
7	0	1	melón	3792
8	1	12	naranja	13693
9	0	8	naranja	4413

2.2.4 Resultado consulta 4

2_3_consulta_4

EJECUTAR

GUARDAR LA CONSULTA (VERSIÓN CLÁSICA)

```
1 /*Calcula para cada semana el dinero que ha entrado a la tienda dicha semana. Luego
2 /*
3 Truncamos la semana al primer dia de la semana (asume domingo como primer dia).
4 Por ejemplo:
5 - Fecha original: 2023-12-23 (Sabado) -> Fecha truncada: 2023-12-17 (Domingo anterior)
6 */
7 WITH ordenes_compra AS (
8   SELECT
9     DATE_TRUNC(fecha, WEEK) AS semana,
10    precio * cantidad AS dinero_entrante
11   FROM `iic2440.ordenes_compra`
12   JOIN `iic2440.Frutas` AS frutas
13   ON `iic2440.ordenes_compra`.id_fruta = frutas.id_fruta
```

Resultados de la consulta

INFORMACIÓN DEL TRABAJO

RESULTADOS

GRÁFICO

JSON

DETALLES DE LA

Fila	semana	dinero_entrante	dinero_acumulado
1	2023-04-30	1923550	25603700
2	2023-05-28	2127250	33283600
3	2023-06-11	2222000	37990750
4	2023-03-26	1457200	17491700
5	2023-11-12	2544700	91242350
6	2023-09-03	2686700	66355300

3. Aclaración

- La tarea se encuentra en el siguiente repositorio de github: <https://github.com/pedrozavalat/iic2440-t01>