**Análisis de resultados – Trabajo 4 Bases de Datos 2**

**Elizabeth Álzate Zapata**

**Cristian Mejía Martínez**

**PUNTO 1**

**Primer Experimento**

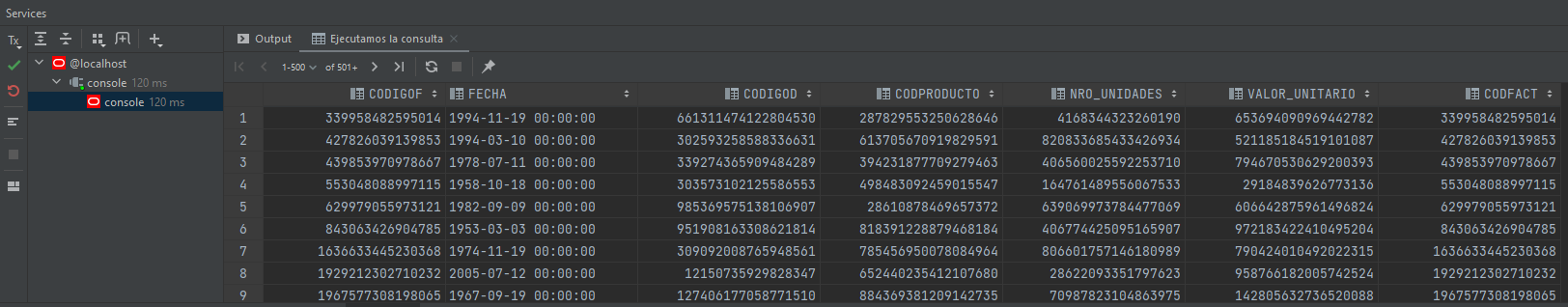
**Volumen bajo de datos**

Para el primer experimento se llevarán a cabo ejecuciones de las 4 propuestas sugeridas en el enunciado, teniendo 5000 entradas en la tabla factura y 20000 entradas en la tabla detalles con un límite de 7 detalles por factura, tal y como se puede ver en la ejecución del programa:

/\* Llenar las tablas con pocos datos \*/  
BEGIN llenado\_aleatorio\_factura(5000); END;  
BEGIN llenado\_aleatorio\_detalle(20000, 7); END;

***Propuesta A)***

Al ejecutar la propuesta A obtenemos un tiempo de ejecución de 120 milisegundos para la consulta solicitada, obteniendo una tabla con 20000 entradas correspondientes a cada una de las facturas junto con sus detalles, en donde cada factura tiene exactamente 4 detalles.

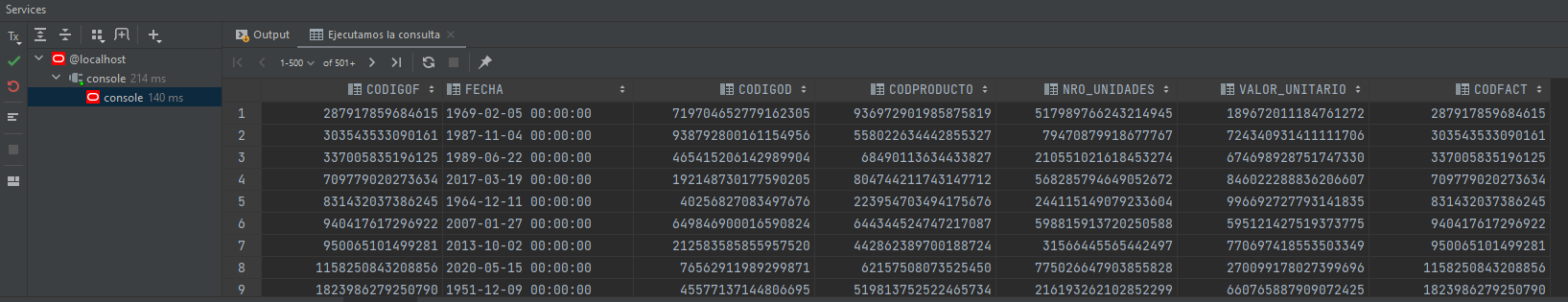


Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

***Propuesta B)***

Al ejecutar la propuesta B obtenemos un tiempo de ejecución de 214 milisegundos, tiempo que es mayor al de la consulta anterior, a pesar de estar obteniendo los mismos resultados, esto indica que para esta consulta en específico con una cantidad baja de entradas en ambas tablas resulta ser contraproducente en cuestión de rendimiento.

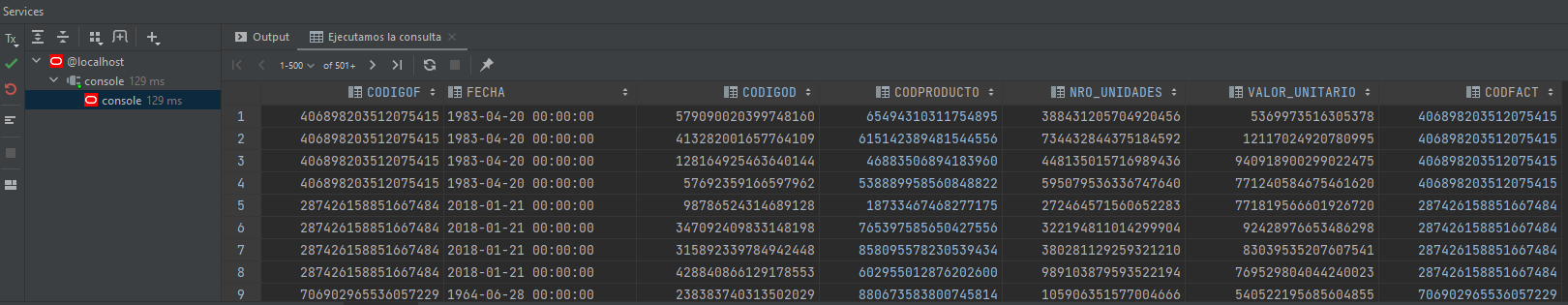


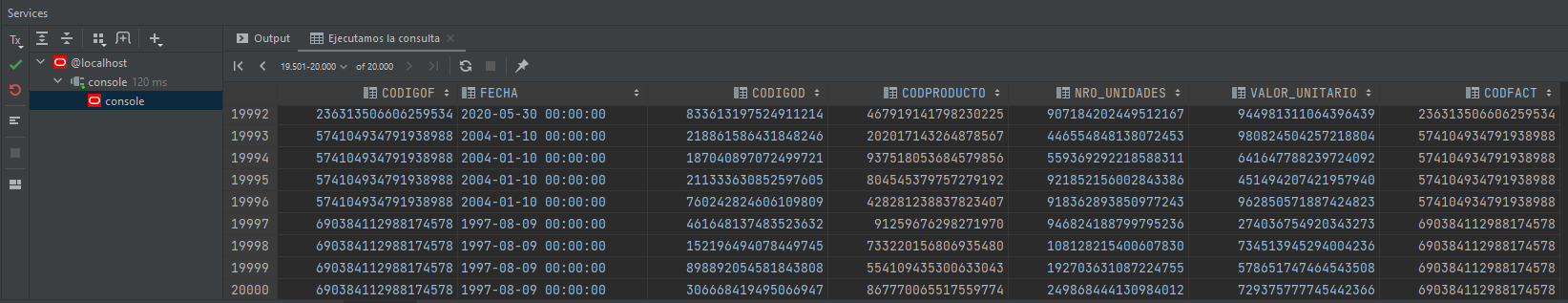
Texto

Descripción generada automáticamente

***Propuesta C)***

Al ejecutar la propuesta C obtenemos un tiempo de ejecución de 129 milisegundos con el mismo resultado de A y B, tiempo que, a pesar de ser más rápido que la propuesta B, requiere más esfuerzo que la A, por lo que también resulta contraproducente con una cantidad baja de datos en ambas tablas





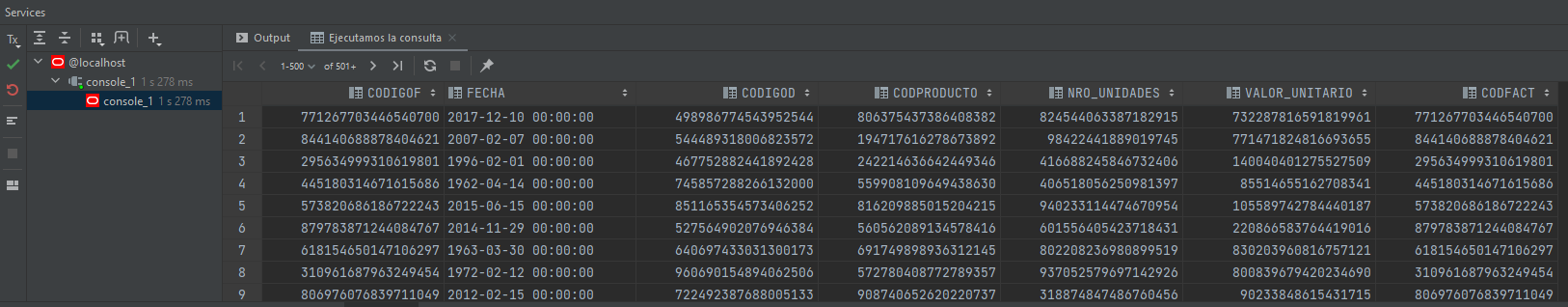
**Volumen alto de datos**

Se repetirá la ejecución del primer experimento teniendo 25000 entradas en la tabla factura y 100000 entradas en la tabla detalles con un límite de 7 detalles por factura, tal y como se puede ver en la ejecución del programa:

/\* Llenar las tablas con datos medianos \*/  
BEGIN *llenado\_aleatorio\_factura*(25000); END;  
BEGIN *llenado\_aleatorio\_detalle*(100000, 7); END;

***Propuesta A)***

Al ejecutar esta consulta se obtiene una tabla con 100000 entradas correspondiente a las facturas con sus respectivos detalles, en donde cada factura posee exactamente 4 detalles, esta consulta tuvo un tiempo de ejecución de 278 milisegundos, el cual es mayor al obtenido con un bajo volumen de datos.

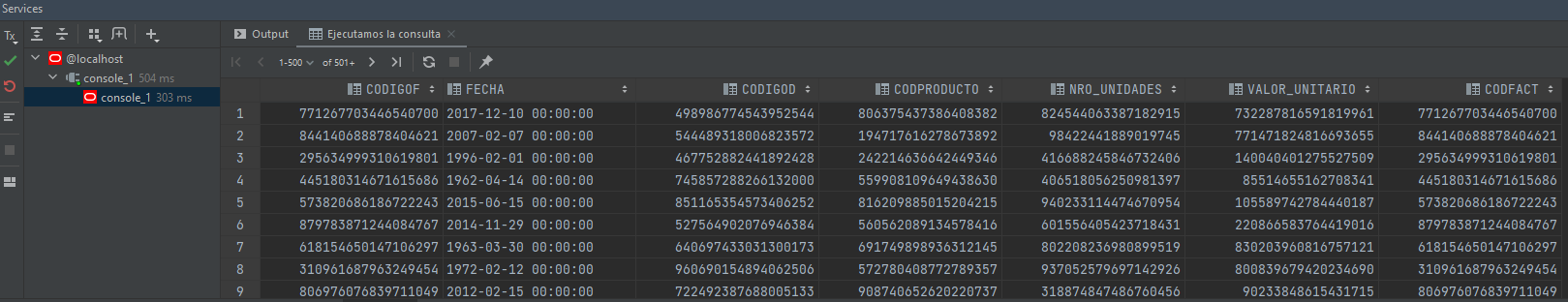


Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

***Propuesta B)***

Al ejecutar esta consulta se obtiene el mismo resultado que A, con un tiempo de ejecución de 504 milisegundos, lo que lo hace más ineficiente que la consulta A

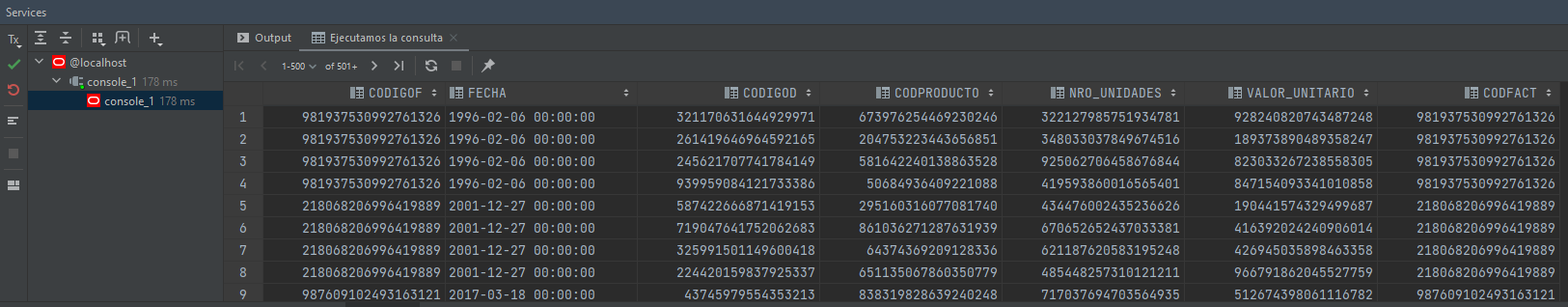


Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

***Propuesta C)***

Al ejecutar esta consulta obtenemos el mismo resultado que A con un tiempo de ejecución de 178 milisegundos, lo que lo hace más eficiente que A y B



Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**Segundo Experimento**

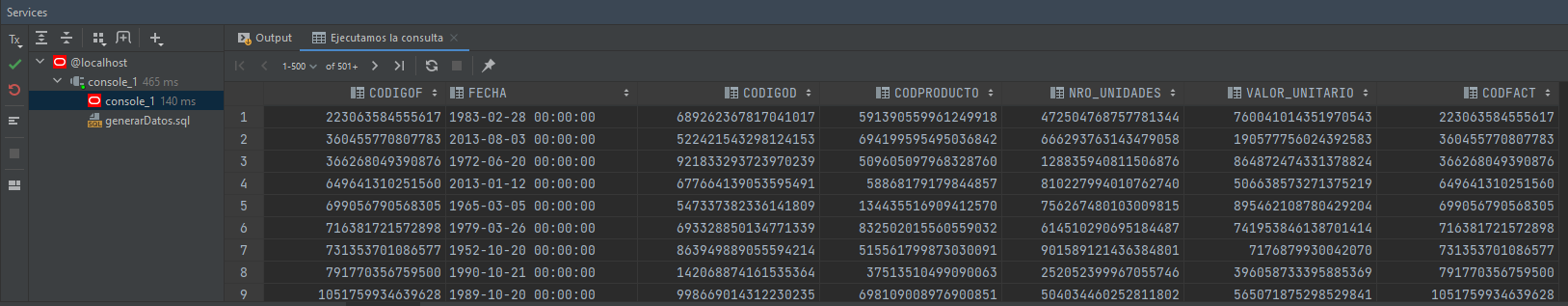
**Volumen bajo de datos**

Para el segundo experimento se llevarán a cabo ejecuciones de las 4 propuestas sugeridas en el enunciado, teniendo 5000 entradas en la tabla factura y 10000 entradas en la tabla detalles con un límite de 3 detalles por factura, tal y como se puede ver en la ejecución del programa:

/\* Llenar las tablas con pocos datos \*/  
BEGIN *llenado\_aleatorio\_factura*(5000); END;  
BEGIN *llenado\_aleatorio\_detalle*(10000, 3); END;

***Propuesta A)***

Al ejecutar esta consulta se obtiene una tabla con 10000 entradas en un tiempo de ejecución de 465 milisegundos, lo cual es relativamente lento para una cantidad de datos baja

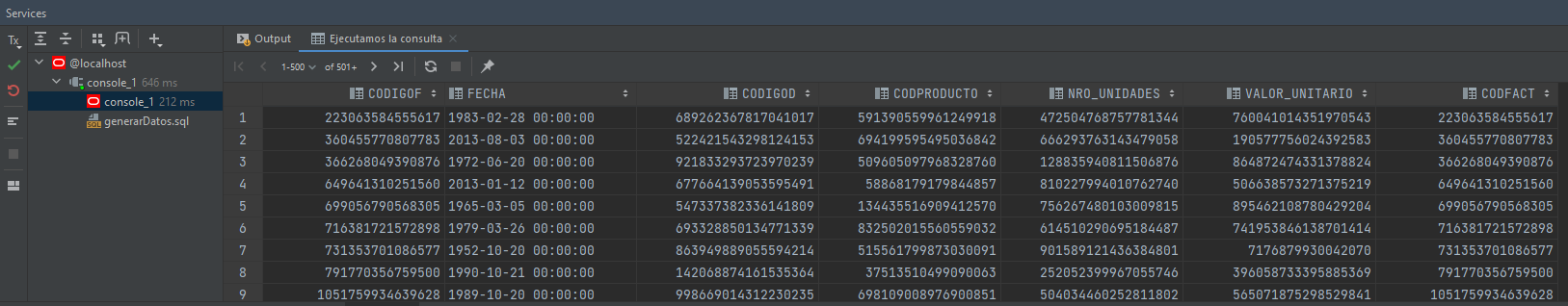


Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

***Propuesta B)***

Al ejecutar esta consulta obtenemos los mismos resultados de A pero con un tiempo de ejecución de 646 milisegundos, lo cual reafirma la hipótesis de que esta consulta es menos eficiente que la consulta A.

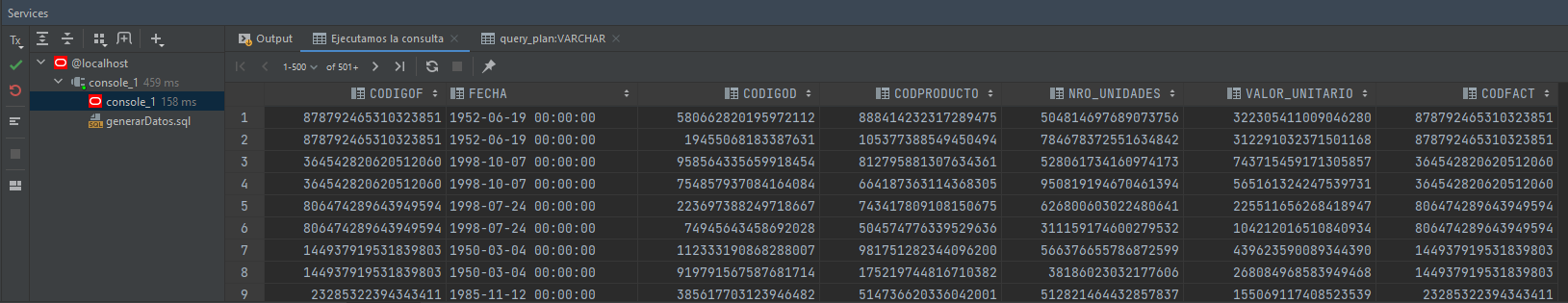


Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

***Propuesta C)***

Al ejecutar esta consulta obtenemos los mismos resultados de A pero con un tiempo de ejecución de 459 milisegundos, lo que lo hace la consulta más eficiente, además de poseer una estructura que permite llenar la tabla de datos de manera mucho más rápida.



Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

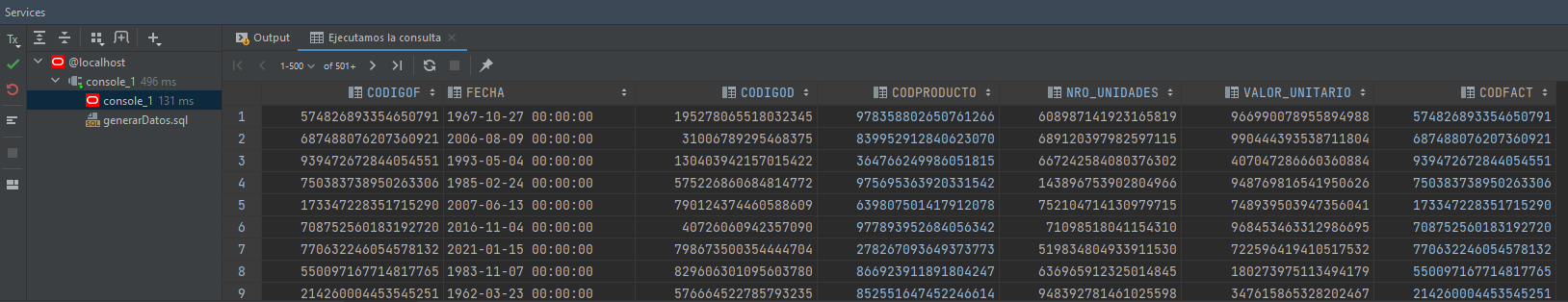
**Volumen alto de datos**

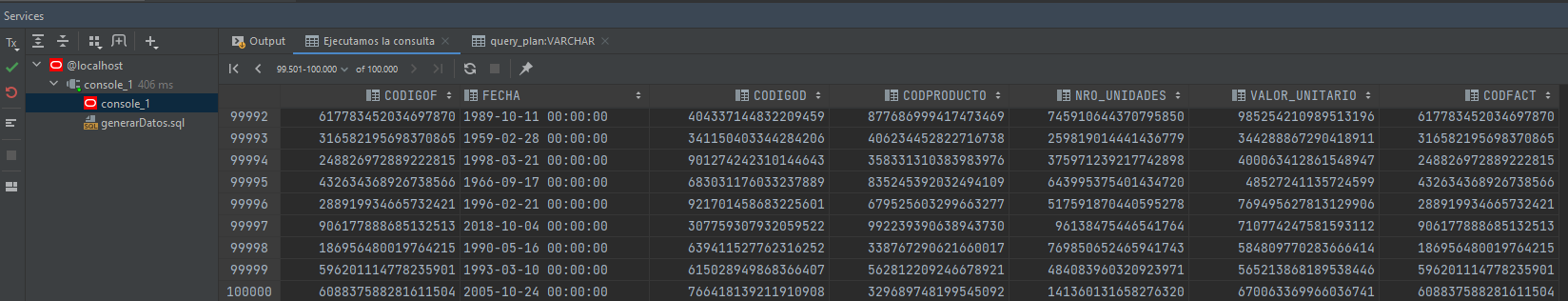
Se repetirá la ejecución del segundo experimento teniendo 50000 entradas en la tabla factura y 100000 entradas en la tabla detalles con un límite de 3 detalles por factura, tal y como se puede ver en la ejecución del programa:

/\* Llenar las tablas con muchos datos \*/  
BEGIN *llenado\_aleatorio\_factura*(50000); END;  
BEGIN *llenado\_aleatorio\_detalle*(100000, 3); END;

***Propuesta A)***

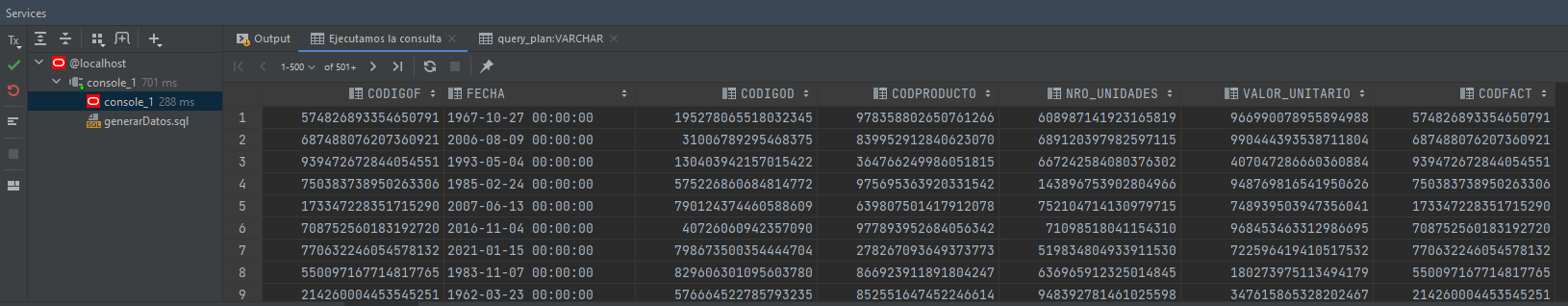
Al ejecutar esta consulta se obtienen 100000 entradas que corresponden a las facturas con sus detalles, con un tiempo de ejecución de 496 milisegundos, similar al tiempo que se obtuvo con una baja cantidad de datos.

******

******

***Propuesta B)***

Al ejecutar esta consulta obtenemos los mismos resultados que en A pero con un tiempo de ejecución de 701 milisegundos, el cual, nuevamente, es más lento que la consulta A.

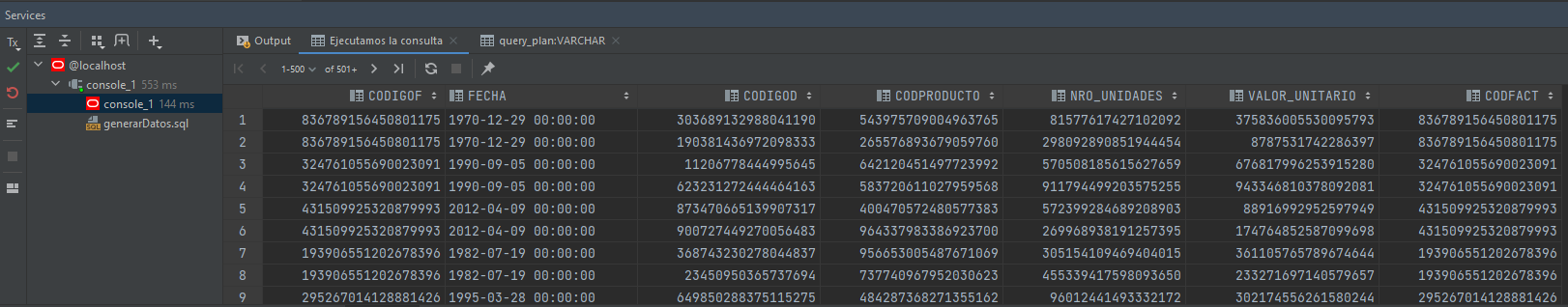


Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

***Propuesta C)***

Al ejecutar esta consulta obtenemos el mismo resultado de A pero con un tiempo de ejecución de 553 milisegundos, el cual es más rápido que la consulta B pero más lento que la consulta A



Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**Tercer Experimento**

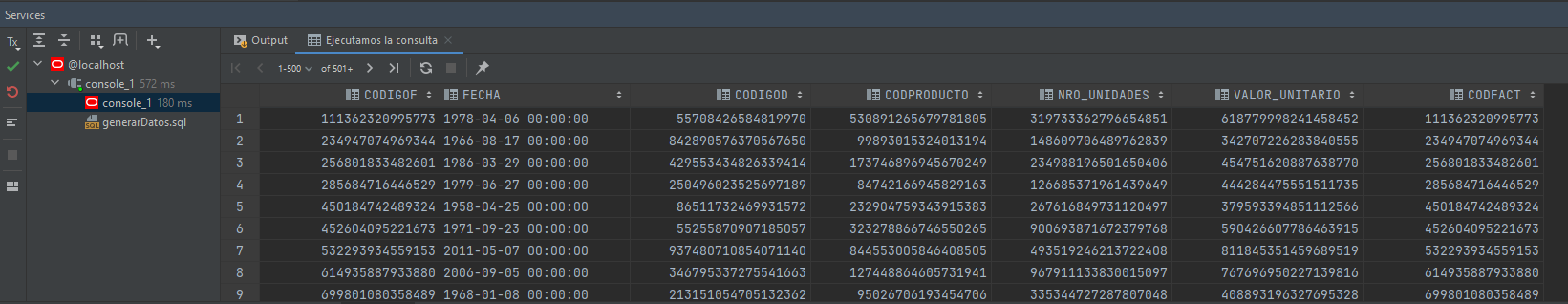
**Volumen bajo de datos**

Para el tercer experimento se llevarán a cabo ejecuciones de las 4 propuestas sugeridas en el enunciado, teniendo 5000 entradas en la tabla factura y 5000 entradas en la tabla detalles con un límite de 1 detalle por factura, tal y como se puede ver en la ejecución del programa:

/\* Llenar las tablas con pocos datos \*/  
BEGIN *llenado\_aleatorio\_factura*(5000); END;  
BEGIN *llenado\_aleatorio\_detalle*(5000, 1); END;

***Propuesta A)***

Al ejecutar esta propuesta se obtiene una tabla con 5000 registros que corresponden a cada una de las facturas con su respectivo detalle con un tiempo de ejecución de 572 milisegundos

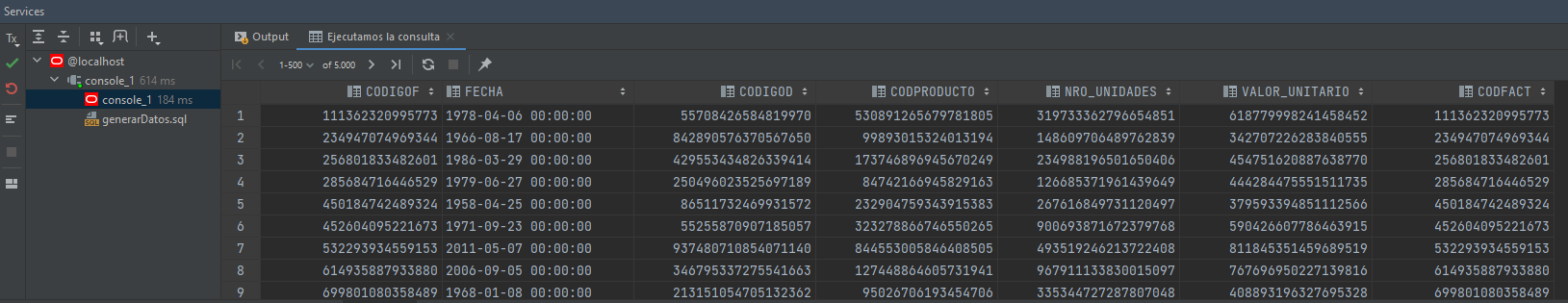


***Interfaz de usuario gráfica, Texto

Descripción generada automáticamente***

***Propuesta B)***

Al ejecutar esta consulta obtenemos el mismo resultado que obtuvimos con A pero con un tiempo de ejecución de 614 milisegundos, siendo menos eficiente que la consulta A

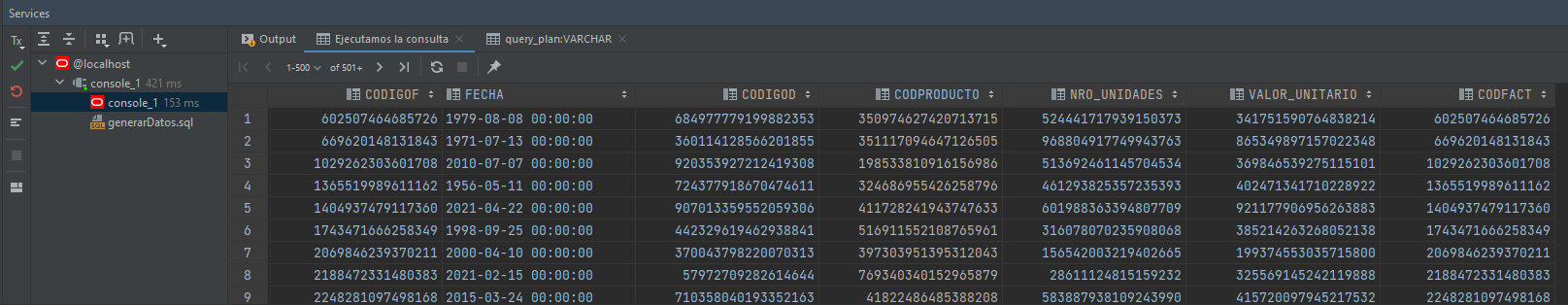


Captura de pantalla de computadora

Descripción generada automáticamente

***Propuesta C)***

Al ejecutar esta consulta obtenemos el mismo resultado que obtuvimos con A pero con un tiempo de ejecución de 421 milisegundos, siendo más eficiente que las consultas A y B.



Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

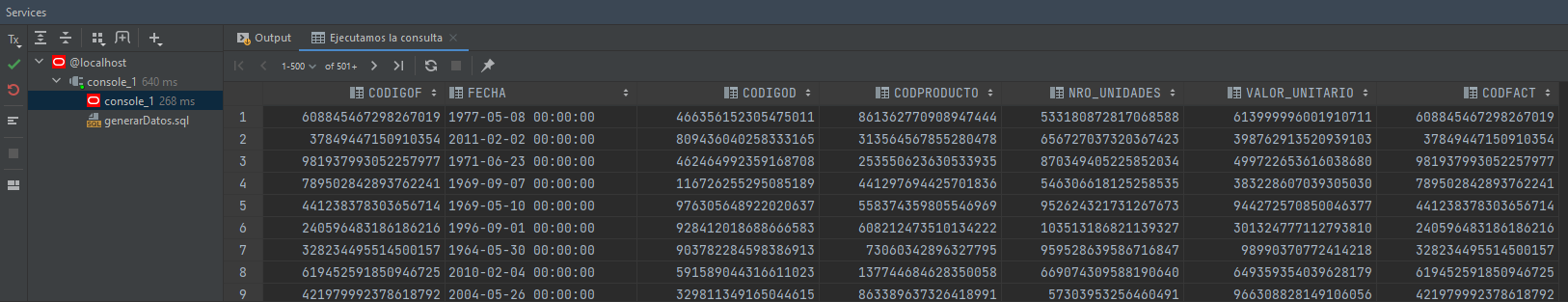
**Volumen alto de datos**

Se repetirá la ejecución del tercer experimento teniendo 50000 entradas en la tabla factura y 50000 entradas en la tabla detalles con un límite de 1 detalle por factura, tal y como se puede ver en la ejecución del programa:

/\* Llenar las tablas con muchos datos \*/  
BEGIN *llenado\_aleatorio\_factura*(50000); END;  
BEGIN *llenado\_aleatorio\_detalle*(50000, 1); END;

***Propuesta A)***

Se obtiene una tabla con 50000 entradas correspondientes a cada una de las facturas con su respectivo detalle con un tiempo de ejecución de 640 milisegundos.

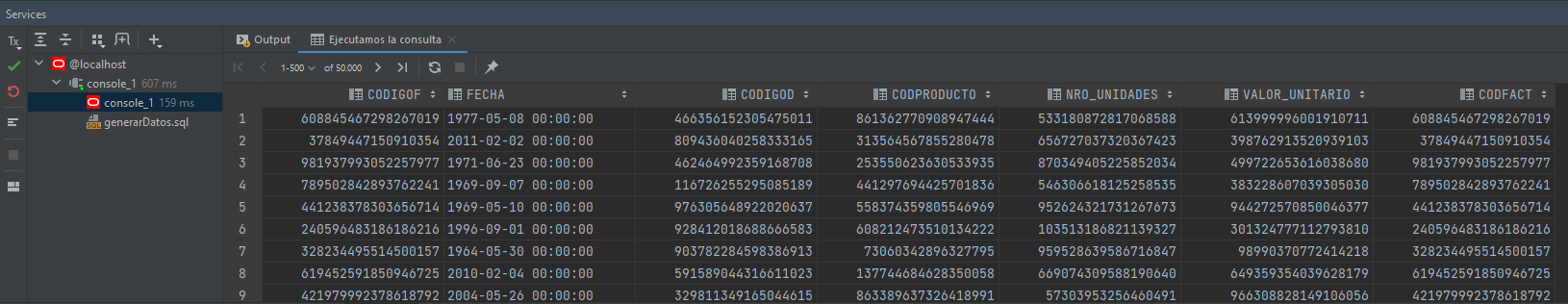
******

***Interfaz de usuario gráfica, Texto

Descripción generada automáticamente***

***Propuesta B)***

Se obtiene el mismo resultado que con la consulta A pero con un tiempo de ejecución de 607 milisegundos, siendo más eficiente que la consulta A

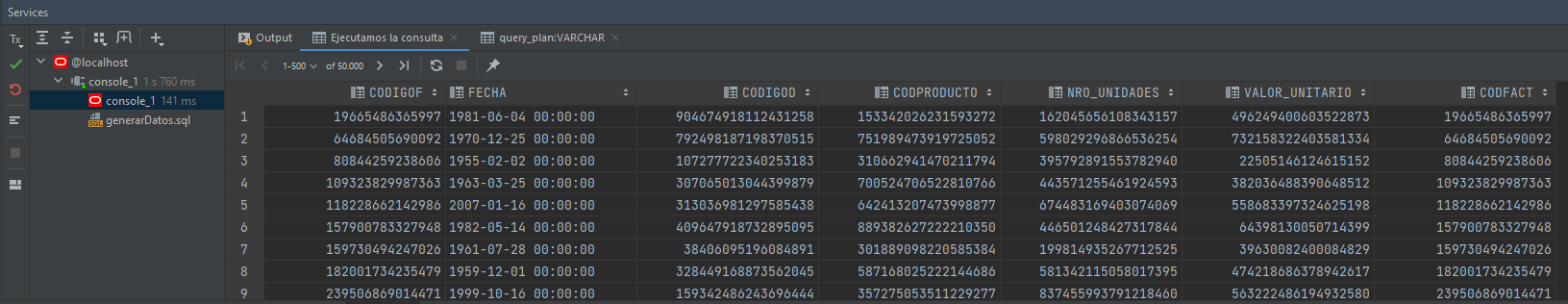


Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

***Propuesta C)***

Se obtiene el mismo resultado que con la consulta A pero con un tiempo de ejecución de 1 segundo y 760 milisegundos



Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**CONCLUSIONES**

En los tres experimentos, las consultas A, B y C mostraron un comportamiento similar sin importar la cantidad de entradas que tuvieran las tablas, siendo las tendencias las siguientes:

* Consulta A: Tiende a mantener un rendimiento constante, el cual no es el óptimo, pero tampoco es el más ineficiente
* Consulta B: Fue la consulta más lenta en la mayoría de los casos, mostrando que la creación del índice pudo no ayudar mucho en estos experimentos en específico
* Consulta C: Fue la consulta más eficiente en la mayoría de los casos, obteniendo un resultado más rápido pero similar a la consulta A, y, principalmente, es la alternativa que permite llenar de datos más rápidamente a ambas tablas, siendo la alternativa que más beneficio brinda entre las demás