**ITERACIÓN 4 – MODELAJE DE UNA APLICACIÓN TRANSACCIONAL**

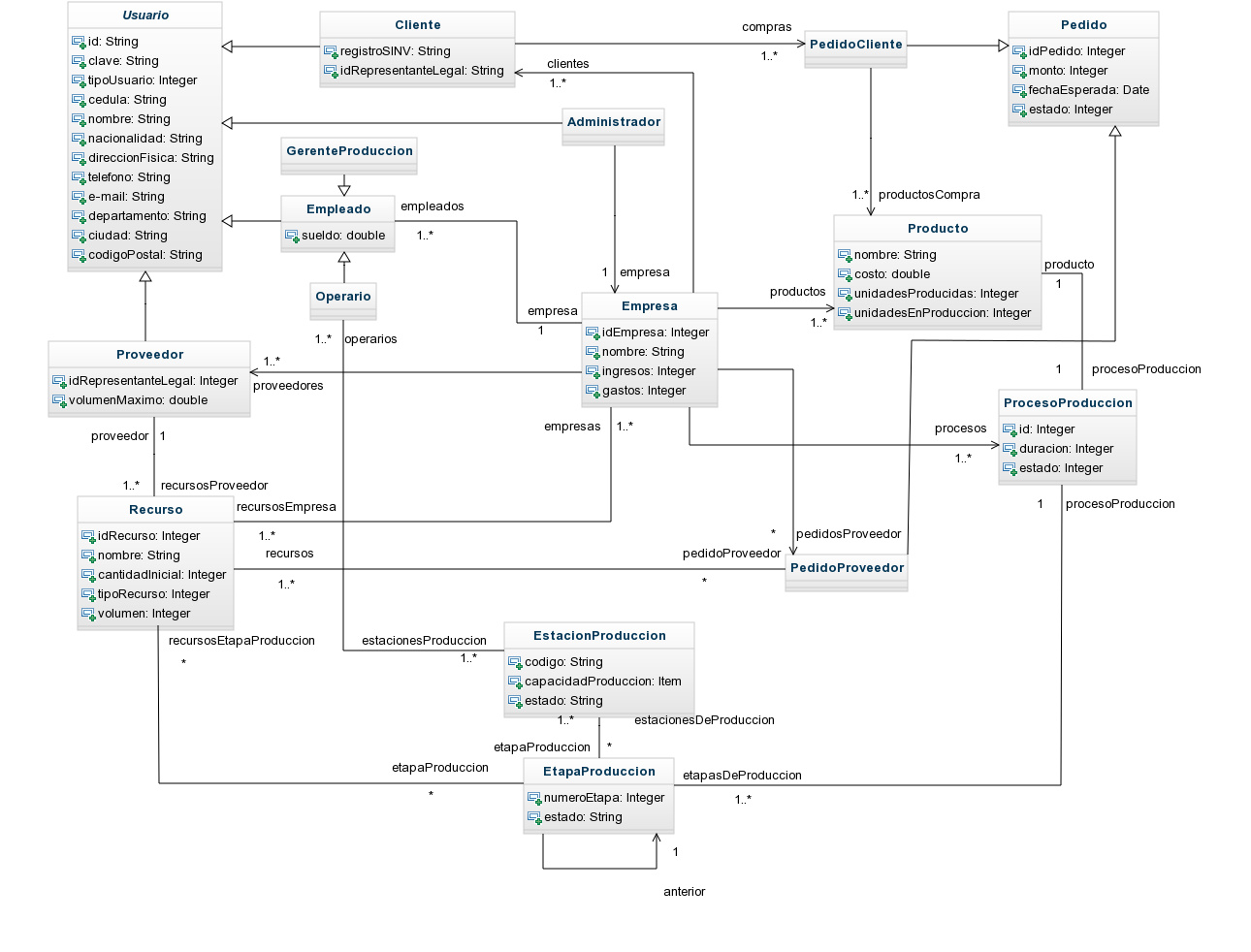
**Grupo 3**

**Carlos Felipe Agudelo Ospina 201328150**

**Sergio Yodeb Velásquez Yepes 201315851**

1. **Análisis**

Con respecto a la entrega pasada no se realizó ningún cambio al modelo conceptual.



Con respecto al modelo conceptual se remplazó la tabla operan con la tabla ejecutaron. Para que se pudiera tener la relación de ejecución de etapas con respecto a una estación de producción, porque en un principio no se había analizado bien esta realción.

|  |  |  |
| --- | --- | --- |
| **Operan** |  |  |
| idOperario | idEtapaProduccion | fechaEjecucion |
| PK1 | PK2, FK(EtapasProduccion.idEtapa) | NN |
| duracion | idPedido |  |
| NN | PK3(Pedido.idPedido) |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ejecutaron** | |  |  |  |
| idEtacionProduccion | | idEtapaProduccion | idPedido |
| PK1, FK(EstacionesProduccion.idEstacionProduccion) | | PK2, FK(EtapasProduccion.idEtapa) | PK3,FK(Pedido.idPedido) |
| TiempoEjecucion | | idOperario | fechaEjecucion |  |
| NN | | PK4, FK(Operarios.idOperario) | NN |  |
|  |

Por esta misma razón se agregó la columna IDPEDIDO a la tabla de ejecutan para que hubiera una relación de cada ejecución con un pedido específico.

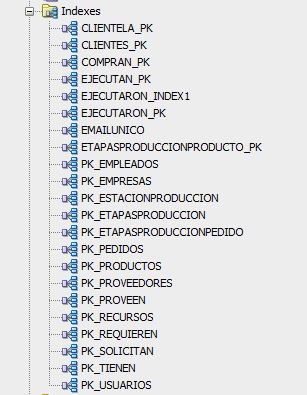
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ejecutan** |  |  |  |  |
| IdEstacionProduccion | IdEtapaProduccion | IdPedido |  |
| PK1, FK(EstacionesProduccion.idEs) | PK2, FK(EtapasProduccion.idEtapa) | PK3, FK(Pedidos.idPedido) |  |

1. **Diseño físico**
   1. **Documentación del diseño físico**

Los índices que manejamos son principalmente los primary key de todas las tablas que son generados por Oracle al nosotros asignarles el valor de primary key. Debido a la diversidad de las consultas crear índices para hacer más eficiente cada una de estas terminaría siendo crear índices para todas las columnas de la base de datos. Por lo cual nosotros miramos que columnas eran las más usadas en todos los requerimientos en general, encontramos que las que más se usaban en nuestro modelo relacional eran la FECHAPEDIDO, la FECHALLEGADA de la tabla Pedidos, el COSTO de la tabla recursos y de la tabla productos, . Puesto que la mayoría de requerimientos tienen como parámetro de búsqueda estas columnas. Y específicamente para esta iteración se usaron índices en la FECHAEJECUCION de la tabla ejecutaron, en TIPORECURSO en la tabla recursos.

Pero además hicimos 2 métodos en la clase DAO que nos permiten crear y eliminar índices cuando sea necesario.

Los índices creados de forma automática por Oracle son:



Los índices creados son las primary key de las tablas y ayudan al rendimiento en la medida de que una búsqueda sobre las tuplas de la tabla se hace buscando por las columnas que la diferencian, que justamente son las primary key.

* 1. **Documentación del análisis**
     1. **Documentación del escenario de pruebas**

**RFC8. CONSULTAR EJECUCIÓN DE ETAPAS DE PRODUCCIÓN 1**

**Consulta SQL**

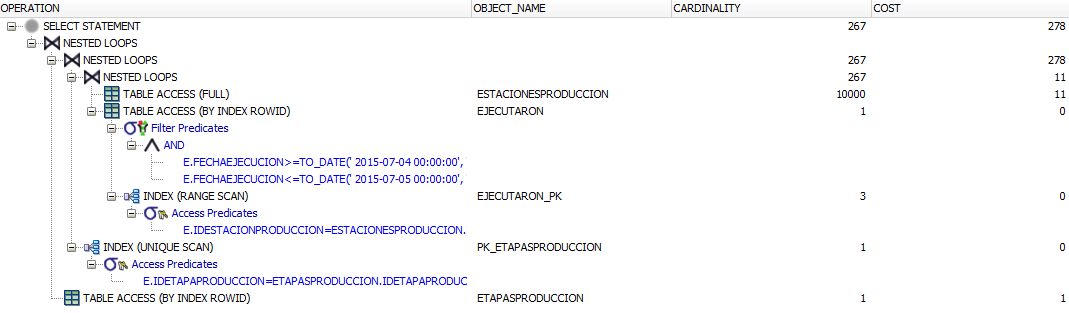
SELECT \* FROM Ejecutaron e NATURAL INNER JOIN EstacionesProduccion NATURAL INNER JOIN EtapasProduccion WHERE e.FechaEjecucion BETWEEN TO\_DATE(<<fecha #1>>,'YYYY-MM-DD') AND TO\_DATE(<<fecha #2>>,'YYYY-MM-DD');

**Distribución de los datos**

Se han ingresado en la base de datos la información de 10000 pedidos. Estos pedidos aleatoriamente fueron entregados o no; aproximadamente el 50% de estos fueron entregados. Por cada pedido entregado se generaron 10 entradas en la tabla Ejecutaron, dado que cada producto en la base de datos requiere de 10 pasos aleatorios entre 10000 posibles etapas de producción para ser completado. Cada una de estas ejecuciones fue realizada en una estación de producción aleatoria entre 10000 posibles estaciones. Las ejecuciones fueron también realizadas por un empleado aleatorio entre 100, en un tiempo aleatorio de ejecución, en un rango de fechas de 6 meses (entre Junio y Diciembre del 2015).

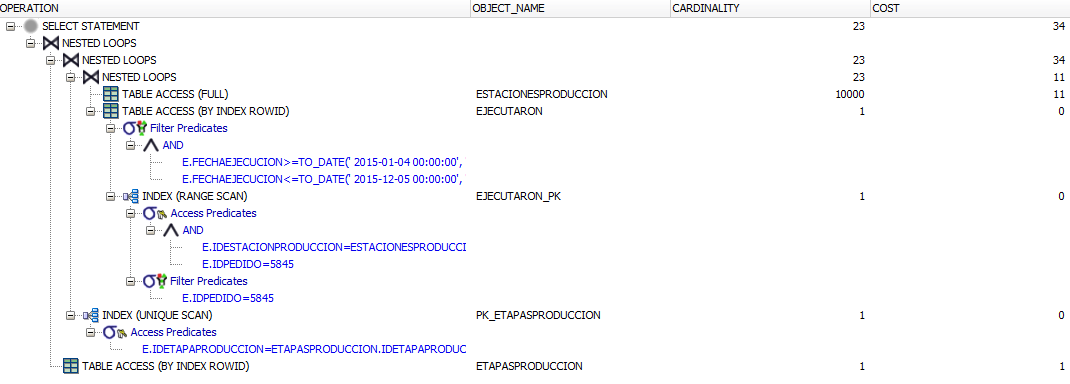
Se espera pues que aproximadamente, para un día dado dentro del rango, se hayan realizado 275 ejecuciones. Esto corresponde al 0.55% de ejecuciones totales. Si se agregan parámetros adicionales a la búsqueda, el porcentaje de ejecuciones recuperadas cambian significativamente. Por ejemplo, si se restringen las ejecuciones a las correspondientes a un pedido específico se recuperaría el 0.02% de las ejecuciones.

**Plan de ejecución y ejecución**



En el plan de ejecución, se utiliza el índice de las llaves primarias de ambas tablas para agilizar el proceso de JOIN entre las tablas. Ante la consulta de las ejecuciones entre el 4 y 5 de julio de 2015 se obtuvieron 294 ejecuciones en 0.788 segundos. Al utilizar un índice en la fecha de ejecución se reduce el tiempo de ejecución a 0.386 segundos. Ante esta consulta no es necesario agregar más índices además del posible índice en fecha de ejecución.

Si se realiza la consulta filtrando por id de un pedido, se logra obtener las filas requeridas en 0.103 segundos para un pedido aleatorio elegido. A continuación se puede ver el plan de ejecución obtenido al aplicar el filtro.



Se aprecia que se hace uso del índice correspondiente a la llave primaria en la tabla correspondiente al id del pedido para agilizar la respuesta.

**RFC9. CONSULTAR EJECUCIÓN DE ETAPAS DE PRODUCCIÓN 2**

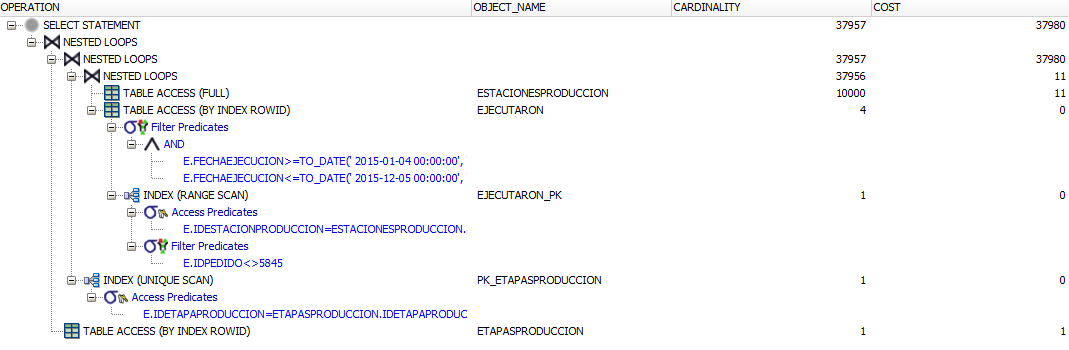
**Consulta SQL**

SELECT \* FROM Ejecutaron e NATURAL INNER JOIN EstacionesProduccion NATURAL INNER JOIN EtapasProduccion WHERE e.FechaEjecucion BETWEEN TO\_DATE(<<fecha #1>>,'YYYY-MM-DD') AND TO\_DATE(<<fecha #2>>,'YYYY-MM-DD') AND e.idPedido != <<idPedido>>;

**Distribución de los datos**

La cantidad y distribución de datos es la misma que la asignada para la consulta anterior. Debe tenerse en cuenta que en este caso la selectividad puede ser bastante alta. Por ejemplo, si se niega la búsqueda por un id de pedido, se obtiene el 99.98% de las ejecuciones totales, lo cual constituye una alta selectividad.

**Plan de ejecución y ejecución**



En el presente plan de ejecución se aprecia que se hace uso de los índices aplicados por Oracle a las llaves primarias de las tablas. Se tiene además que el costo de ejecución viene principalmente del proceso de JOIN entre las tablas involucradas en la consulta. Esto se debe a la gran cantidad de datos que deben ser unidos en este caso.

En cuanto al tiempo de ejecución, se tiene que 1500 filas pueden ser recuperadas en 1.045 segundos. Dado que SQL Developer realiza paginación de sus resultados, se espera que, sabiendo que los resultados serían 49990 filas, para obtener todos los resultados se tendría que esperar 33.3 segundos. No debe olvidarse que una consulta de este tipo no tiene mucho sentido, y no arroja información relevante para la aplicación.

**RFC10. CONSULTAR PEDIDOS 2**

**Consulta SQL**

SELECT \* FROM Pedidos NATURAL INNER JOIN Compran NATURAL INNER JOIN Productos NATURAL INNER JOIN (SELECT DISTINCT idProducto FROM EtapasProduccionProducto NATURAL INNER JOIN Requieren NATURAL INNER JOIN Recursos r WHERE r.costo><<costo>> AND r.tipoRecurso='<<tipo Recurso>>');

**Distribución de los datos**

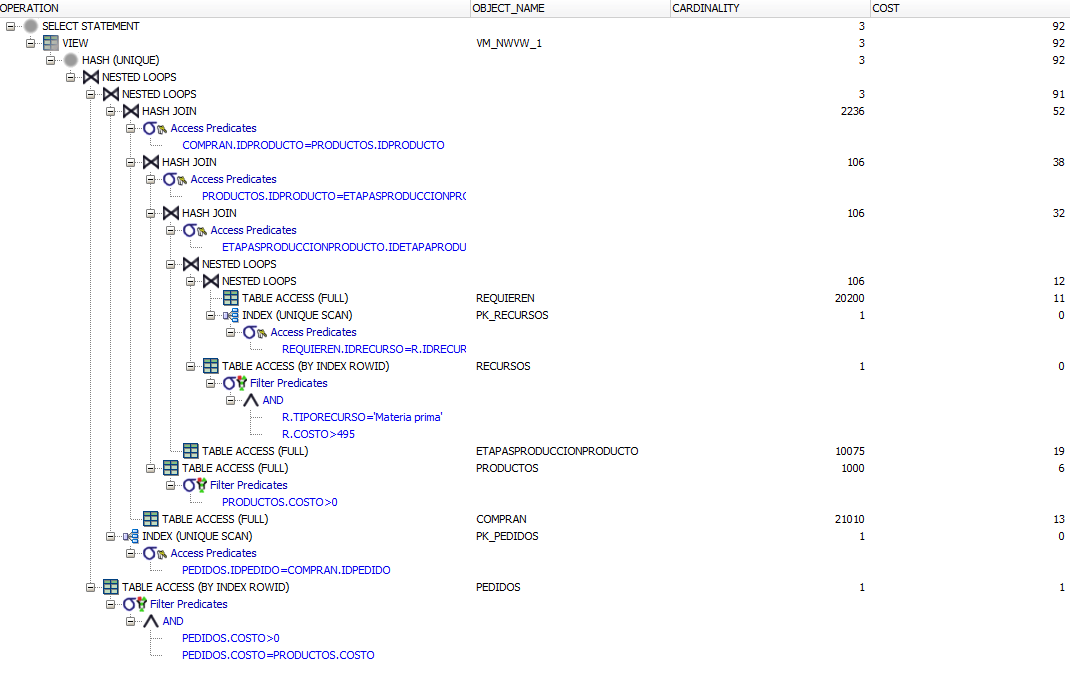
Se han ingresado a la base de datos la información de 10000 recursos. A cada una de las 10000 etapas de producción que se encontraban en el programa se le agregaron además la necesidad de dos recursos aleatorios entre los presentes para su ejecución. El costo de los recursos agregados fue aleatorio, entre $10 y $500. Se debe resaltar que a cada recurso se le asignó un tipo aleatorio, por lo que aproximadamente 50% de estos son materias primas y el otro 50% de recursos corresponde componentes.

Ante la consulta planteada es un poco complicado calcular la selectividad. Si se toma un costo de 10 como parámetro la selectividad sería de 50%, si se toma un costo de aproximadamente 260 la selectividad sería de 25%. Se tiene pues que la selectividad depende únicamente del costo seleccionado para la consulta.

**Plan de ejecución y ejecución**

El plan de ejecución aparece sumamente complicado, dado que consiste en gran parte de JOIN, puede apreciarse que buena cantidad de JOINS consisten en Hash Join. La ejecución de esta consulta, que para un costo con selectividad baja (495 de costo) arroja como resultado 146 tuplas, toma 1.05 segundos en ejecución.

Debe notarse que no hay muchas más posibilidades de variación a través del uso de índices, dado que la mayoría de JOINS ya se hacen sobre columnas que tienen índices sobre llaves primarias y, sobre el tipo de recurso no vale la pena tener un índice sobre el tipo de recurso dado que la selectividad es de 50% bajo este caso.



**RFC11. CONSULTAR MATERIAL 2**

**Consulta SQL**

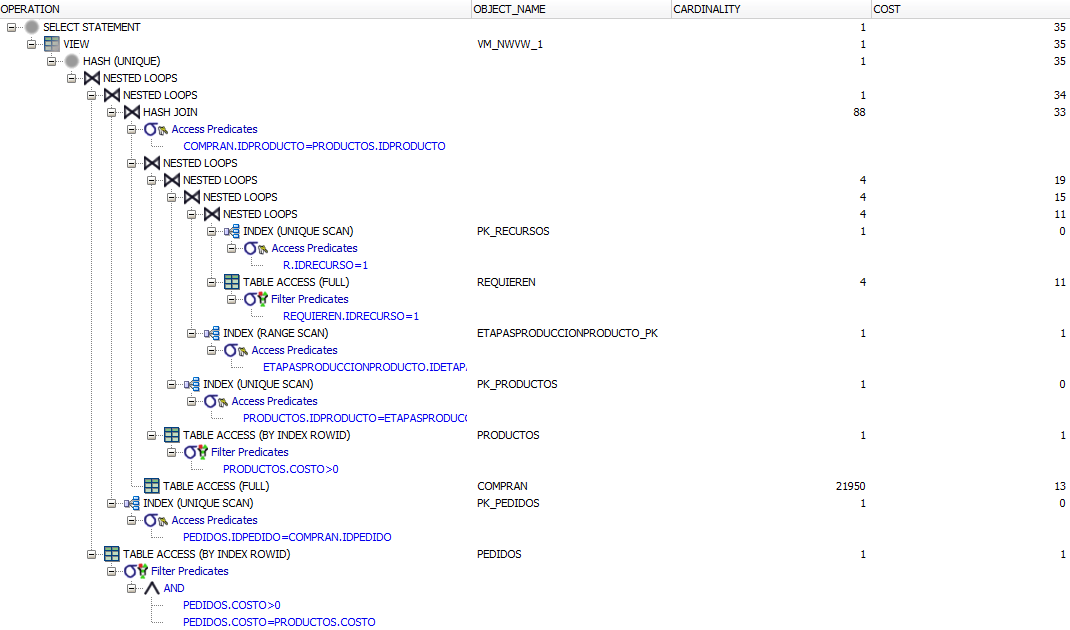
SELECT \* FROM Pedidos NATURAL INNER JOIN Compran NATURAL INNER JOIN Productos NATURAL INNER JOIN (SELECT DISTINCT idProducto FROM EtapasProduccionProducto NATURAL INNER JOIN Requieren NATURAL INNER JOIN Recursos r WHERE r.idRecurso=<<idRecurso>>);

**Distribución de los datos**

La distribución de datos es la misma que la del requerimiento anterior. Se tiene que en este caso la selectividad es distinta. Si se trata de encontrar todos los pedidos en los que un recurso dado se encuentra se tiene que probablemente se seleccione 2 pedidos entre 10000, dada la forma en la que los datos fueron agregados en la base de datos.

**Plan de ejecución y ejecución**

El plan de ejecución es en buena parte parecido al del anterior requerimiento, dado que los JOIN realizados son básicamente los mismos. En esta consulta los índices de las llaves primarias son utilizados más dado que, además de los JOIN, resultan útiles en el filtro de la consulta. Debe destacarse también que en este caso no hay necesidad de agregar índices adicionales en alguna columna. La ejecución de esta consulta, para un id de recurso dado que resulta en 5 tuplas toma 1.957 segundos.



* + 1. **Análisis de eficiencia**
* **RFC8. Consultar ejecución de etapas de producción 1**
  + **Escenarios con diferentes selectividades**

La selectividad de este requerimiento se puede dividir en la selectividad de las fechas y la selectividad del criterio de búsqueda asociado. Por lo tanto los escenarios serían:

Hay una dispersión muy baja en las fechas, todas son de un mismo periodo de tiempo, esto implicaría que la selectividad de las fechas es

* + **Diseño de un plan de ejecución**
  + **Comparar con el de Oracle**
* **RFC9. Consultar ejecución de etapas de producción 2**
* **RFC10. Consultar pedidos 2**
* **RFC11. Consultar material 2**

1. **Construcción de la aplicación y análisis de resultados**
   1. **Documentación del proceso de la carga de datos**
   2. **Análisis del proceso de optimización y el modelo de ejecución de consultas**

La diferencia principal es que el manejador de bases de datos tiene desarrollada una serie de algoritmos que le permiten encontrar de forma muy eficiente los datos, así como una forma de organizar físicamente la información para encontrarla de forma más efectiva, reduciendo los tiempos de la consulta a tiempos muy cortos. Habitualmente tienen los datos organizados en arboles b+ o tablas hashing expandibles, lo que les permite hacer operaciones de selección de una manera muy rápida. Además normalmente cuando uno hace la consulta el manejador no carga todos los datos inmediatamente sino que a medida que uno va pidiendo más datos él los va buscando y trayendo a la memoria principal.

Mientras que si uno quisiera hacer esto en la aplicación le tomaría mucho tiempo, puesto que si quisiera primero organizar los datos para hacer búsquedas sobre ellos tendría que sumarle al tiempo de búsqueda el tiempo de organizar los datos y para cada búsqueda con datos distintos tendría que volver a hacer el mismo proceso. O si intentara hacer la búsqueda sin ordenar el tiempo sería que para cada operación tendría que recorrer todos los datos que tiene.