Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ciencias de la Computación
CC42A/CC55A - Bases de Datos

Sistema de Base de Datos para Servicios y Facturación de Socios de COTEL-Bolivia

Informe 4: Diseño y Optimización de Consultas

Profesor : Claudio Gutiérrez G.
Profesor Auxiliar : Diego Diaz E.
Alumnos : Jorge Jara W.
Luis Silvestre Q.
Omar Larré V.
Fecha : 01/07/2009

Índice

1. Presentación

El presente informe describe el desarrollo de la última etapa del proyecto del curso Bases de Datos, en su versión del primer semestre del año 2009. El objetivo general de esta etapa es el diseño y la optimización de consultas sobre la base de datos implementada para almacenar información de socios, servicios y facturación de la Cooperativa de Telecomunicaciones de la Paz (COTEL, Bolivia). El trabajo consiste en la definición de índices y pruebas de rendimiento de la base de datos para comprobar la ganancia en tiempo de los planes de consulta con uso de índices.

2. Objetivos de la Etapa

Se plantean dos objetivos principales:

- Implementación de la base de datos en un sistema administrador de bases de datos. En este caso el sistema de administración es PostgreSQL, instalado sobre un computador de arquitectura Intel x86 y sistema operativo Linux.
- Selección de un conjunto de operaciones de consulta relevantes en cuanto a eficiencia y optimización en la base de datos implementada. En particular se requiere determinar los índices y campos adecuados para mejorar los tiempos de acceso a los datos, mediante la medición de tiempos y comprobación de los planes de consulta escogidos por el sistema administrador.

3. Desarrollo

3.1. Implementación de la Base de Datos

La implementación del esquema relacional se había realizado ya con la entrega de la etapa anterior del proyecto. Restaba la carga de datos en las tablas a ser utilizadas para las consultas.

Se realizaron cambios menores al esquema relacional, relativos al largo de algunos campos clave para identificar órdenes de trabajo.

3.1.1. Carga de Datos Desde Archivos de Datos Originales

Los archivos de referencia implementados en un sistema de base de datos fueron hechos en FOX-PRO, mediante software con interfaces en C, con formatos tales como DBF, CDX, FRT y otros. Se proporcionaron para la implementación del proyecto datos de socios, además de archivos con detalles de la acomodación física y lógica de los números telefónicos en la sub-central de San Pedro.

Para ser utilizados, estos archivos tuvieron que ser migrados a formato de base de datos DB2 para luego ser convertidos a formato Access, pudiendo manipular, verificarse y rescatar los datos que serían utilizados más adelante en la implementación del proyecto. Vale la pena destacar que las tablas no estaban normalizadas asi que se tuvo la urgencia de separalas cuidadosamente para ser compatibles con el modelo de datos relacional del proyecto; en este proceso de migración se tuvo que usar un método por etapas, pues las bases de datos antiguas poseen una estructura no compatible con el modelo relacional, y gracias a herramientas CASE se pudo migrar los datos a una base DB2.

En la mayoría de las tablas existían muchos vacíos y números sin asignación que provocaban polución a la estrutura de datos por tratarse de tuplas nulas en la mayoría de sus atributos, y fue necesario depurar estas tuplas nulas con el fin de tener datos que mantengan la consistencia de la base de datos imlpementada.

La Figura 1 es una vista de la interfaz del anterior sistema de COTEL (llamado MFDCO) con una Base de Datos en FOX-PRO, perteneciente a la subcentral de san Pedro (SPD). Este sistema sólo era de consulta histórica de registro, puesto que las transacciones de OT y de actualización de servicios y actividades se realizaban en plantillas Excel y Access para ser remitidas al Gran Centro.

La Figura 2 muestra archivos generados y utilizados por MFDCO (en formatos MBF, CDX, FRT, BAK), y varios archivos temporales. Notar la no existencia de ningún tipo de normalización de datos y que se trata de tablas en la mayoria de los casos con datos superfluos que violan todo concepto de normalización.

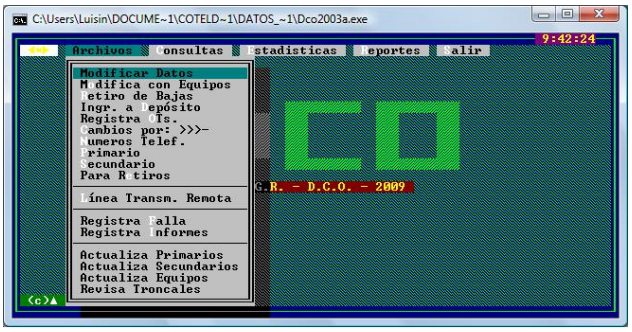

Figura 1: Interfaz del antiguo sistema de COTEL (MFDCO).

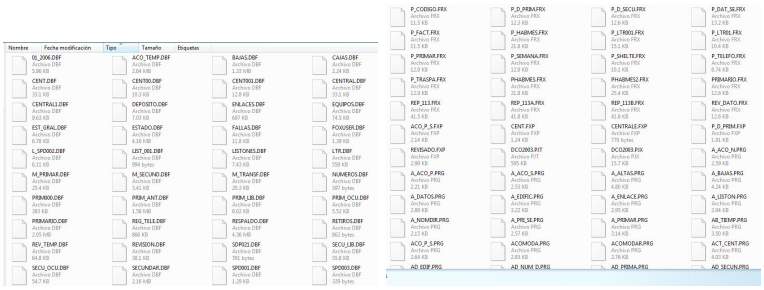

Figura 2: Archivos de base de datos FOX PRO.

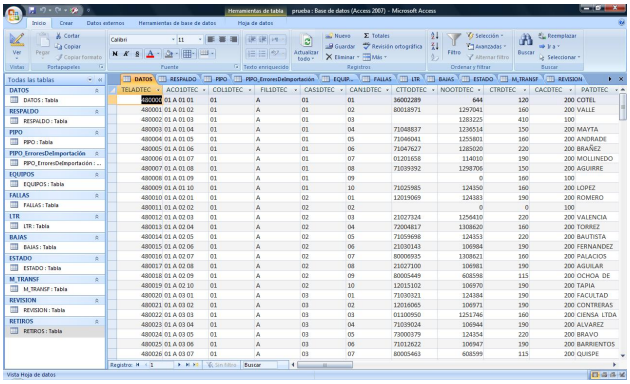

Figura 3: Importación de archivos a ACCESS.

La Figura 3 muestra la última etapa en la migración de datos desde FOX hasta Access. Como ya se mencionó, se debió pasar por varias etapas para lograr la integridad y conservación de los datos, es decir, se migraron las tablas a DB2 para luego ser importadas desde Access y luego a Excel para su apropiado manejo.

3.1.2. Simulación de Datos Faltantes

Pese a que los datos obtenidos son reales, una importante porción de datos relevantes no pudieron ser conseguido. En particular, sólo se dispone de estadísticas sobre las distribuciones de los datos relacionados con el consumo mensual de servicios de cada socio y con el detalle de la generación diaria de Órdenes de Trabajo, aunque cabe destacar que existe una fracción considerable de éste último (dispersa en archivos).

En este escenario fue necesario realizar simulaciones para generar estos datos, intentando aproximar el comportamiento descrito por las estadísticas de los datos reales. Para ello se desarrolló un programa en MATLAB que hiciera el trabajo de las simulaciones.

El consumo de minutos hablados mensuales de la línea telefónica se basa en el plan que todos los socios de COTEL tienen contratado, y que consta de 200 minutos libres de llamadas locales más un cobro extra por cada minuto excedido. Se sabe que el 80 % de los socios consume menos de los 200 minutos disponibles y que un 15 % del resto se exceden de los 200 minutos pero no en más que 23 minutos. Luego, el consumo de minutos mensuales, C, de un socio cualquiera en el mes t, se aproxima por

$$C = BU^{1/p} + (1 - B)(200 + X)$$

donde B es una variable aleatoria Bernoulli de parámetro 0,8, U es una variable uniforme en [0, 1] y X es una variable aleatoria exponencial de parámetro λ. Cada uno de los parámetros se ajustó para obtener los porcentajes que se observan en los datos reales. La Figura 4 muestra la distribución resultante luego de la simulación. Simulaciones similares se realizaron para el resto de las variables de consumo de cada socio.

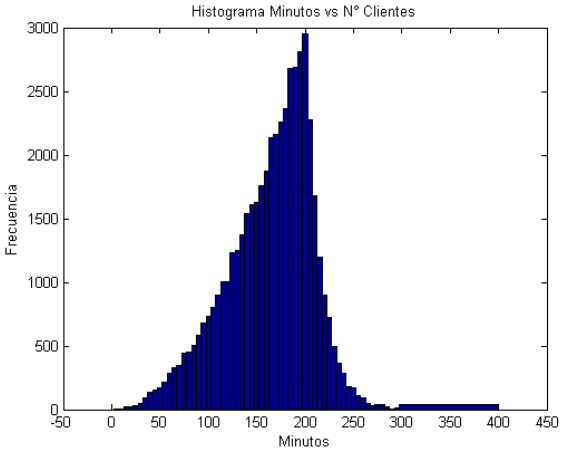

Figura 4: Distribución de la simulación de consumo mensual de minutos del plan de línea telefónica local.

3.2. Optimización de Consultas

Para el desarrollo de esta sección se presentan dos consultas sobre la base de datos que fueron sometidas a pruebas de optimización en su ejecución mediante el uso de índices, midiendo el tiempo de respuesta para cada caso.

3.2.1. Consultas Seleccionadas

Se identificaron dos consultas en que hay acceso a tablas con una gran cantidad de datos (cientos de miles de registros):

- Estado de facturación: facturas de servicios generadas en los últimos seis meses a socios de la Cooperativa y sus montos asociados.

  El objetivo de la consulta es establecer y presentar información que es de importancia para un ejecutivo de la cooperativa, siendo informacion estratégica para la evaluación y estimaciones de consumo de los socios y revisión de políticas sobre los ingresos y los comportamientos de éstos. En forma consolidada y a largo plazao esta información apoya la toma de desiciones sobre estrategias promocionales o bien políticas sobre los criterios de pago y consumo para el corte de una linea telefónica y/o servicios. En el mediano y corto, permite llevar el control a nivel operativo del estado de cuentas por cobrar de la Cooperativa. Debe tenerse en cuenta para un entorno real de aplicación que las tablas de facturación y generafacturación son en general poco suceptibles a modificaciones, pero a cambio su número de registros aumenta rápidamente (miles al mes).

- Estado de avance de órdenes de trabajo: detalle de estado y autorizaciones realizadas por funcionarios a las órdenes de trabajo que se encuentren sin terminar.

  El objetivo de la consulta es el seguimiento en el tiempo sobre las órdenes de trabajo solicitadas por un determinado socio o conjunto de socios, además de establecer e identificar a las administrativos en el flujo regular de autorización que requiere una OT, para ello también se requiere la asignación de un técnico para su atención, así como datos como la fecha y estado, este último motivo es importante para planificar con urgencia OT's con riesgo de caducidad o retraso en su atención, dependiendo de la gravedad y del tipo de atención que requiere. Este tipo de evaluaciones son necesarias también para registros historicos de responsabilidad administrativa y seguimiento de OTs.

  Si bien se generan en menos cantidad que los registros de facturación, las órdenes de trabajo están sujetas a modificaciones en períodos de días o semanas (usualmente). Los principales tipos de modificaciones son el cambio de estado. Para el caso de autorización por parte de administrativos, además de su ejecución por los técnicos.

3.2.2. Implementación en SQL

A continuación se muestra el código[1] en lenguaje SQL de las consultas detalladas en la sección anterior, sin nigún tipo de orden y optimización. Estas dos consultas se denominarán **consultas base** de aquí en adelante.

- **Consulta De Facturación**

```
SELECT   f.codigo                                    AS codigoDeFactura   ,
         f.fechadeemision                            AS fechaEmision      ,
         ltr.numero                                  AS numeroLTR         ,
         lt.numero                                   AS numeroFacturado   ,
         SUM (cs.tarifabase + (cs.tarifadia*consumodiacondserv) + (cs.tarifanoche*gf.consumodiacondserv))
         AS sumaConsumoDiaTemp,
         d.*
FROM     lineatelefonica     AS lt ,
         lineatelefonica     AS ltr,
         condiciondeservicio AS cs ,
         factura             AS f  ,
         generafacturacion   AS gf ,
         direccion           AS d
WHERE    d.callepasaje                     = lt.direccioncallepasaje
     AND d.numero                          = lt.direccionnumero
     AND d.zona                            = lt.direccionzona
     AND d.datosadicionales                = lt.direcciondatosadicionales
     AND gf.codigofactura                  = f.codigo
     AND gf.codigocondiciondeservicio      = cs.codigodeservicio
     AND gf.inicioperiodovigenciacondserv  = cs.inicioperiodovigencia
```

[1] Para formatear el aspecto del código se utilizó la herramienta online SQLinForm, ver http://www.sqlinform.com/.

```
    AND gf.finperiodovigenciacondserv      = cs.finperiodovigencia
    AND gf.numerofacturado                 = lt.numero
    AND gf.numerolineatelefoicaresidencial = ltr.numero
    AND f.fechadeemision                  >= '2008-06-01'
    AND f.fechadeemision                  <= '2008-12-01'
GROUP BY lt.numero         ,
         ltr.numero        ,
         f.codigo          ,
         d.callepasaje     ,
         d.numero          ,
         d.zona            ,
         d.datosadicionales,
         f.fechadeemision;
```

- **Consulta De Órdenes de Trabajo**

```
SELECT ot.cisocio                                    ,
       pp.nombre                                     ,
       pp.apellidop                                  ,
       ot.codigo AS codigoOT                         ,
       ot.numerolinea                                ,
       ot.tipo AS tipoOrden                          ,
       ot.descripcion                                ,
       autGerentes.niGerente AS codigoGerente        ,
       autGerentes.fechaAut                          ,
       autOperadores.niOperador AS codigoOperador    ,
       autOperadores.fechaAut                        ,
       autSupervisores.niSupervisor AS codigoSupervisor,
       autSupervisores.fechaAut                      ,
       exeTecnicos.niTecnico AS codigoTecnico        ,
       exeTecnicos.fechaEje  AS fechaEjecucion
FROM   socio                 AS s            ,
       persona               AS pp           ,
       ordendetrabajo        AS ot           ,
       (SELECT otg.codigo                                          AS gcodigo  ,
               agot.numerodeitem                                   AS niGerente,
               agot.fecha                                          AS fechaAut
       FROM    ordendetrabajo                                      AS otg
               LEFT OUTER JOIN autoriza_gerenteadjunto_ordendetrabajo AS agot
               ON      otg.codigo = agot.codigoordendetrabajo
       ) AS autGerentes                        ,
       (SELECT oto.codigo                                     AS ocodigo   ,
               aoot.numerodeitem                              AS niOperador,
               aoot.fecha                                     AS fechaAut
       FROM    ordendetrabajo                                 AS oto
               LEFT OUTER JOIN autoriza_operador_ordendetrabajo AS aoot
               ON      oto.codigo = aoot.codigoordendetrabajo
       ) AS autOperadores                        ,
       (SELECT ots.codigo                                       AS scodigo     ,
               asot.numerodeitem                                AS niSupervisor,
               asot.fecha                                       AS fechaAut
       FROM    ordendetrabajo                                   AS ots
               LEFT OUTER JOIN autoriza_supervisor_ordendetrabajo AS asot
               ON      ots.codigo = asot.codigoordendetrabajo
       ) AS autSupervisores                          ,
       (SELECT ote.codigo                                  AS tcodigo  ,
               eje.numerodeitemtecnico                     AS niTecnico,
               eje.fecha                                   AS fechaEje
```

```
        FROM    ordendetrabajo                                  AS ote
                LEFT OUTER JOIN ejecuta_tecnico_ordendetrabajo AS eje
                ON      ote.codigo = eje.codigoordendetrabajo
        ) AS exeTecnicos
WHERE  s.ceduladeidentidad = ot.cisocio
   AND pp.ci               = s.ceduladeidentidad
   AND ot.fechacreacion   >= '2008-09-01'
   AND ot.fechacreacion   <= '2008-12-31'
   AND (
            NOT (
                   ot.estado = 'Terminado'
            )
        )
   AND exeTecnicos.tcodigo     = ot.codigo
   AND autOperadores.ocodigo   = ot.codigo
   AND autSupervisores.scodigo = ot.codigo
   AND autGerentes.gcodigo     = ot.codigo;
```

3.3. Optimización de Consultas

Se probaron distintos planes de índices para las consultas escogidas, que se describen a continuación. En particular se construyeron planes con indexación por Hash y árboles B+, siendo *a priori* el hashing candidato en muchas situaciones de selección por igualdad, y B+ una alternativa para búsquedas por rango y selección de valores múltiples que satisfagan una igualdad. Las consultas fueron implementadas de SQL sin considerar optimizaciones. Se realizaron pruebas previas cambiando en el orden de los términos de las consultas, no encontrando diferencias apreciables en los tiempos ni en el plan resultante.

3.3.1. Planes de Índices Para Consulta de Facturación

- Abreviaturas para nombres de tablas:

  LT: LineaTelefonica

  LTR: LineaTelefonicaResidencial

  F: Factura

  GF: GeneraFacturacion

  D: Direccion

  CD: CondicionDeServicio

- Alternativa 0

  Sin índices especiales definidos (salvo los de clave primaria)

- Alternativa 1

  LT: B+ agrupado sobre la clave primaria (todos los campos, excepto en el consumo día/noche).

  LTR: B+ agrupado sobre la clave primaria ("número").

  CS: Cualquiera, que el tamaño de la tabla (cantidad de tuplas < 100) permite manejarla en memoria.

  D:

  - B+ agrupado sobre la clave primaria (callepasaje, numero, zona,...)

  - Hash sobre callepasaje

  GF: B+ agrupado sobre clave primaria

  F: B+ agrupado sobre clave primaria

- Alternativa 2

  B+ agrupado para todos las tablas. Además se tienen los siguientes índices

  D: Hash en "callepasaje"

  GF: Hash en "numerofacturado"

  F: B+ agrupado sobre fechadeemision en lugar de sobre la clave primaria ("codigo")

- Alternativa 3

  Igual a la alternativa 1, con excepción del índice de Hash removido de GF.numerofacturado

3.3.2. Planes de Índices Para Consulta de Órdenes de Trabajo

- Abreviaturas para nombres de tablas

  OT: ordendetrabajo

  AUTO: autoriza_operador_ordendetrabajo

  AUTG: autoriza_gerente_ordendetrabajo

  AUTS: autoriza_supervisor_ordendetrabajo

  ET: ejecuta_tecnico_ordendetrabajo

  P: persona

  S: socio

- Alternativa 0

  Sin índices especiales definidos (salvo los de clave primaria)

- Alternativa 1

  OT: B+ agrupado en "fechacreacion"

  AUTO: Hash sobre "codigoordendetrabajo"

  AUTS: Hash sobre "codigoordendetrabajo"

  AUTG: Hash sobre "codigoordendetrabajo"

  ET: Hash sobre "codigoordendetrabajo"

  P: archivo ordenado por "ci"

  S: archivo ordenado por "ci"

  Ante condiciones de selección por igualdad, el hashing suele ser la primera opción. Debe notarse, sin embargo, que en este caso el hashing debiera mapear a lo más un registro por cada entrada de la tabla, ya que para cada orden de trabajo sólo hay un operador que la autoriza, un supervisor, etc. (con excepción de los técnicos, que pueden ser 2 o más en algunos casos).

- Alternativa 2

  Similar a la alternativa 1, pero AUTO, AUTS, AUTG y ET indexados por árbol B+ (campo "codigo-ordendetrabajo") y agrupados. Se considera esta opción porque es probable que el índice Hash resulte ineficiente, tomando en cuenta que existe solamente un autorizador en cada tabla por orden de trabajo.

- Alternativa 3

  Igual a la alternativa 2 con la adición de índice Hash sobre el atributo "estado" de la tabla ordende-trabajo. Se considera esta opción porque la consulta considera el despliegue de las órdenes de trabajo que se encuentran pendientes, es decir aquéllas cuyo estado no es "Terminado".

3.4. Plataforma de Pruebas

- Hardware

  Computador personal de arquitectura x86.

  Procesador AMD Athlon XP 2600+ @2.0GHz, 512KB caché L2, bus de 266MHz.

  Memoria RAM de 1GB DDR @333MHz.

  Disco duro ATA Maxtor 6Y080P0, capacidad de 80GB, en interfaz IDE nForce2 (nVidia) de 32bits @66MHz, con partición de 13GB asignada para el sistema operativo Linux, en formato ext3.

- Software

  Sistema operativo Ubuntu Linux 8.04, kernel 2.6.24-18-generic.

  Sistema administrador de bases de datos: PostgreSQL 8.3.

  Se utilizó la aplicación Lappstack (Bitnami) para la administración de la base de datos mediante interfaz web.

3.5. Medición de tiempos

Se realizó mediante el uso de los comandos EXPLAIN y ANALYZE, que antepuestos a cada consulta permiten obtener el plan de ejecución escogido por el sistema de gestión de la base de datos. La opción ANALYZE permite la evaluación de la consulta, entregando el tiempo empleado en su ejecución.

Siguiendo la sugerencia de clases auxilires se realizaron n = 12 observaciones para cada prueba.

3.6. Resultados

Los tiempos de ejecución de cada una de las consultas, considerando uso del plan de índices y consultas alternativas se muestran a continuación. Adicionalmente, la información de salida de las optimizaciones generadas por el sistema de gestión de la base de datos se anexa al final de este informe.

3.6.1. Consulta Sobre Facturación

Tabla 1. Resultados obtenidos consulta sobre facturación.

Pruebas	Tiempos (milisegundos) para cada tipo de consulta			
	Alternativa 0	Alternativa 1	Alternativa 2	Alternativa 3
1	76952,978	71112,506	98877,518	102444,631
2	71961,973	72339,250	101982,446	92966,930
3	86550,268	72767,997	100378,658	94233,191
4	76462,765	70957,560	99218,176	92896,070
5	74491,506	71230,434	97106,000	93263,778
6	84917,537	72760,696	102839,369	90758,105
7	78788,026	77678,774	101322,978	90903,140
8	81958,255	73960,900	95408,192	96401,029
9	77176,793	70306,224	100862,857	92338,258
10	79046,581	77350,837	99172,593	96646,720
11	79406,931	70442,171	108096,835	92930,234
12	90941,041	70364,060	110605,755	92624,065
Promedio	79887,88783	72605,95075	101322,6148	94033,84592

El gráfico de estos resultados se encuentra en la Figura 5.

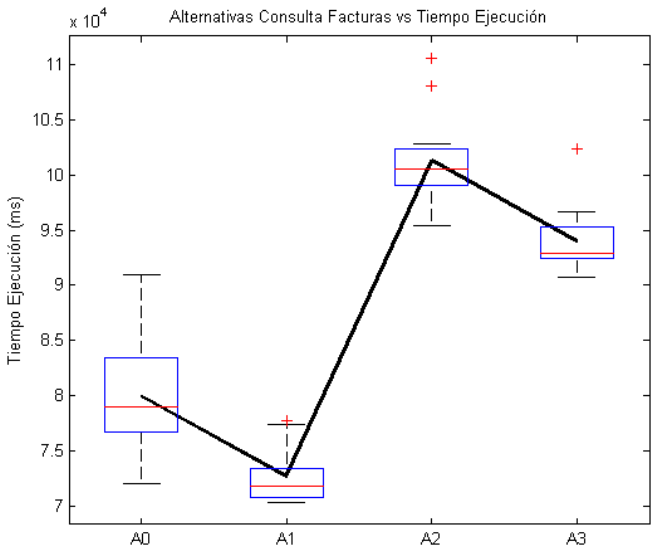

Figura 5: Resultados obtenidos de la consulta de facturas. En línea gruesa continua, los promedios de cada alternativa.

3.6.2. Consulta De Órdenes de Trabajo

Tabla 2. Resultados obtenidos consulta sobre OTs.

Pruebas	Tiempos (milisegundos) para cada tipo de consulta			
	Alternativa 0	Alternativa 1	Alternativa 2	Alternativa 3
1	0,680	0,748	0,712	0,624
2	0,711	0,789	0,553	0,729
3	0,782	0,781	0,736	0,782
4	0,721	0,840	0,773	0,758
5	0,751	0,750	0,662	0,655
6	0,747	0,763	0,555	0,714
7	0,722	0,777	0,725	0,655
8	0,746	0,706	0,673	0,544
9	0,644	0,788	0,695	0,732
10	0,701	0,790	0,706	0,699
11	0,763	0,752	0,720	0,725
12	0,685	0,724	0,750	0,691
Promedio	0,721083333	0,767333333	0,688333333	0,692333333

El gráfico de estos resultados se encuentra en la Figura 6.

3.7. Análisis y Discusión

- Consulta de Facturas

  La mejor alternativa de índices resultó ser la inclusión de un índice de Hash para el campo “direccioncalle” en la tabla Direccion. En este caso se realizaron operaciones de Merge-Join para la reunión entre Direccion y LineaTelefonica.

  Con cualquiera de los planes de índices probados el sistema administrador se vio forzado a realizar un barrido secuencial por la tabla GeneraFacturacion, en la que casi todos sus atributos forman parte de una condición de selección para reunión con otras tablas. El optimizador de consultas no fue capaz de

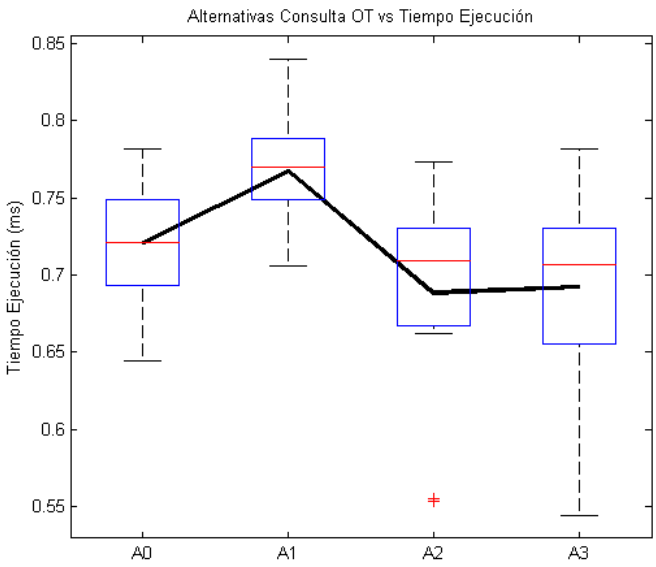

Figura 6: Resultados obtenidos de la consulta sobre OTs. En línea gruesa continua, los promedios de cada alternativa.

generar un plan de ejecución que tomara en cuenta adecuadamente la reducción del número de registros a revisar en la tabla GeneraFacturacion (aun agrupando e indexando por fecha de facturación). Además realizó siempre ciclos anidados sin índices. Esto conlleva una penalización en el procesamiento de la consulta, que es ostensiblemente notoria por cuanto se trata de una tabla que fácilmente supera el millón de registros. Si se comparan los gráficos de tiempo de las dos consultas hay una diferencia de varios órdenes de magnitud (menos de un segundo para OTs y más de un minuto para facturación).

Una posible explicación para esto el que la lógica empleada por el optimizador del SGBD no pueda determinar un orden apropiado para realizar la evaluación (por ejmeplo, utilizando como loop externo un barrido por índice de fecha en la tabla GeneraFacturacion). Para corroborar esta hipótesis se intentó cambiar el orden de las cláusulas de selección en la consulta, sin observar cambios (una prueba utilizando otro SGBD podría arrojar más evidencias para el caso). Otra posible causa de este problema puede ser el esquema normalizado para la tabla GeneraFacturacion. Dado que la consolidación de los datos en distintas tablas requiere de un proceso de reunión con muchas condiciones es inevitable que el gestor de BD (considerando la implementación particular de PostgreSQL) tenga que realizar un barrido secuencial por la tabla GeneraFacturacion. Como alternativa se podría incorporar al esquema de la tabla Factura el monto total facturado (atributo derivado) y el número de línea facturado. Esta alteración de esquema implica que la consulta podría definirse en términos de menos tablas (LT, F, D y eventualmente LTR), y podría realizarse un barrido por índice (de fecha) para acceder a los registros que correspondan, mejorando sustancialmente el tiempo de ejecución. En este caso habría un compromiso de las propiedades del modelo de datos en términos de su normalización, pero que puediera ser asumida en pos de una mejor eficiencia en la ejecución de este tipo de consultas.

Con todo, en el mejor de los casos (alternativa 1) Se logró un incremento en el tiempo de procesamiento de la consulta del orden del 10

- Consulta de Órdenes de Trabajo

  Esta consulta presentó tiempos de respuesta considerados breves (sobre todo comparados con la consulta anterior), manteniéndose para todos los planes de índices probados por debajo de un segundo. Un elemento favorable es el tamaño de las tablas de personal administrativo y técnico involucradas en la consulta, que por su reducido número de registros (del orden de decenas o centenas a lo sumo)

permiten realizar ciclos de modo muy eficiente. El mejor resultado promedio fue obtenido utilizando un índice de árbol B+ agrupado para las OTs. Sin embargo, la ganancia en tiempo de respuesta para los mejores planes fue de un 5 % promedio aprox. Came notar que los índices Hash fueron comparables a B+ (y aun con una ventaja marginal para B+) en el acceso a registros de autorización y asignación de personal.

4. Conclusiones

- La implementación de índices en la optimizacion de consultas suele ser la forma más eficiente de llevar a cabo una consulta relacional, dado que pueden emplearse distintos criterios en la definición de distintos índices para cada atributo en particular; cada propuesta de indexación puede ser más o menos eficiente, pero en general se mejoran los tiempos de procesamiento de consultas simples.
- Para el modelo relacional implementado en el proyecto, considerando los tipos de consulta y planes de optimización elaborados, existe la evidencia de que bajo el esquema de datos actual y los distintos índices considerados, el optimizador prefirió el barrido secuencial de las tuplas para ejecutar; lo que resultó en tiempos de ejecución bastante largos (más aún cconsiderando que en un entorno real un tiempo de espera del orden de minutos no es aceptable). En este caso para lograr una optimización (sin considerar cambio en la lógica del optimizador o del SGBD) se puede optar por relajar las restricciones de normalización en las tablas GeneraFacturacion y Factura, con el fin de tener almacenar reunidos a los datos relacionados por la consulta disminuyendo el número de reuniones y en consecuencia el tiempo de ejecución de la consulta; sería necesario en esta situación verificar la integridad de los datos mediante la implementacion de disparadores u otros mecanismos similares de comprobación de las restricciones relajadas en el esquema.
- En general, los objetivos definidos para esta etapa del proyecto se consideran cumplidos. Se logró implementar la base de datos y se realizaron pruebas de rendimiento. Respecto de la optimalidad de los planes y los resultados obtenidos quedan abiertas algunas interrogantes con respecto a la consulta de facturación, cuya exploración quedó inconclusa al término del tiempo presupuestado para esta etapa.
- Como un aspecto destacable del proyecto en general, está el desarrollo sistemático del trabajo realizado por el equipo, que permitió completar adecuadamente cada etapa y avanzar con mínimos inconvenientas en el proyecto. En términos de práctica y experiencia fue posible conocer y manejar mejor algunas funcionalidades y herramientas para diseño e implementación de bases de datos (CASE, PostgreSQL, Lappstack). Es valorable también la iniciativa de incluir como trabajo durante el curso un proyecto de estas caraterísticas, permitiendo vincular conceptos y práctica implicados en el diseño e implementación de bases de datos.

5. Anexo: algunas salidas de comando “EXPLAIN ANALYZE” del SGBD para consultas y planes de índices

5.1. Consulta Facturación

5.1.1. Alternativa 0

Indices de claves primarias solamente (sin ordenar tabla LT).

```
HashAggregate  (cost=26577.53..26580.03 rows=111 width=104) (actual time=74632.591..75395.811 rows=349069 loops=1)

  ->  Nested Loop  (cost=14497.61..26575.04 rows=111 width=104) (actual time=3244.626..68437.439 rows=634564 loops=1)

        ->  Nested Loop  (cost=14497.61..26204.91 rows=111 width=104) (actual time=3244.563..62372.334 rows=634564 loops=1)

              ->  Hash Join  (cost=14497.61..25420.06 rows=202 width=96) (actual time=3244.048..19742.173 rows=1087824 loops=1)

                    Hash Cond: (((gf.codigocondiciondeservicio)::text = (cs.codigodeservicio)::text) AND (gf.inicioperiodovigenciacondserv = cs.inicioperiodovigencia)
                    AND (gf.finperiodovigenciacondserv = cs.finperiodovigencia))

                    ->  Nested Loop  (cost=14496.31..25407.04 rows=862 width=102) (actual time=3243.892..16334.502 rows=1453572 loops=1)

                          ->  Merge Join  (cost=14492.81..15826.69 rows=36 width=48) (actual time=3243.702..6740.822 rows=60025 loops=1)

                                Merge Cond: (((lt.direccioncallepasaje)::text = (d.callepasaje)::text) AND ((lt.direccionnumero)::text = (d.numero)::text) AND
                                ((lt.direccionzona)::text = (d.zona)::text) AND ((lt.direcciondatosadicionales)::text = (d.datosadicionales)::text))

                                ->  Sort  (cost=8133.75..8287.54 rows=61515 width=47) (actual time=1528.692..3020.751 rows=61515 loops=1)

                                      Sort Key: lt.direccioncallepasaje, lt.direccionnumero, lt.direccionzona, lt.direcciondatosadicionales

                                      Sort Method:  external merge  Disk: 3640kB

                                      ->  Seq Scan on lineatelefonica lt  (cost=0.00..1347.15 rows=61515 width=47) (actual time=0.043..74.742 rows=61515 loops=1)

                                ->  Materialize  (cost=6311.77..6936.62 rows=49988 width=44) (actual time=1714.932..1923.185 rows=60138 loops=1)

                                      ->  Sort  (cost=6311.77..6436.74 rows=49988 width=44) (actual time=1714.907..1828.334 rows=49988 loops=1)

                                            Sort Key: d.callepasaje, d.numero, d.zona, d.datosadicionales

                                            Sort Method:  external sort  Disk: 2856kB

                                            ->  Seq Scan on direccion d  (cost=0.00..957.88 rows=49988 width=44) (actual time=0.082..45.121 rows=49988 loops=1)

                          ->  Bitmap Heap Scan on generafacturacion gf  (cost=3.50..264.36 rows=141 width=58) (actual time=0.105..0.123 rows=24 loops=60025)

                                Recheck Cond: (gf.numerofacturado = lt.numero)

                                ->  Bitmap Index Scan on idx_num_facturado  (cost=0.00..3.46 rows=141 width=0) (actual time=0.076..0.076 rows=24 loops=60025)

                                      Index Cond: (gf.numerofacturado = lt.numero)

                    ->  Hash  (cost=1.11..1.11 rows=11 width=43) (actual time=0.101..0.101 rows=11 loops=1)
```

```
                        ->  Seq Scan on condiciondeservicio cs  (cost=0.00..1.11 rows=11 width=43) (actual time=0.065..0.077 rows=11 loops=1)

              ->  Index Scan using factura_pkey on factura f  (cost=0.00..3.87 rows=1 width=27) (actual time=0.037..0.037 rows=1 loops=1087824)

                    Index Cond: ((f.codigo)::text = (gf.codigofactura)::text)

                    Filter: ((f.fechadeemision >= '2008-06-01 00:00:00'::timestamp without time zone) AND (f.fechadeemision <= '2008-12-01 00:00:00'::timestamp without time zone))

        ->  Index Scan using lineatelefonica_pkey on lineatelefonica ltr  (cost=0.00..3.32 rows=1 width=4) (actual time=0.006..0.007 rows=1 loops=634564)

              Index Cond: (ltr.numero = gf.numerolineatelefoicaresidencial)
```

5.1.2. Alternativa 1

Indices B+ agrupados

```
HashAggregate  (cost=55237.53..55240.02 rows=111 width=104) (actual time=71874.819..72771.363 rows=349069 loops=1)

  ->  Nested Loop  (cost=15831.20..55235.03 rows=111 width=104) (actual time=6355.392..65515.377 rows=634564 loops=1)

        ->  Nested Loop  (cost=15831.20..54864.91 rows=111 width=104) (actual time=6355.273..59726.132 rows=634564 loops=1)

              ->  Hash Join  (cost=15831.20..54080.06 rows=202 width=96) (actual time=6354.738..17278.377 rows=1087824 loops=1)

                    Hash Cond: (((gf.codigocondiciondeservicio)::text = (cs.codigodeservicio)::text) AND (gf.inicioperiodovigenciacondserv = cs.inicioperiodovigencia) AND
                    (gf.finperiodovigenciacondserv = cs.finperiodovigencia))

                    ->  Hash Join  (cost=15829.89..54067.04 rows=862 width=102) (actual time=6354.249..13952.478 rows=1453572 loops=1)

                          Hash Cond: (gf.numerofacturado = lt.numero)

                          ->  Seq Scan on generafacturacion gf  (cost=0.00..32703.20 rows=1473420 width=58) (actual time=0.038..1798.467 rows=1473420 loops=1)

                          ->  Hash  (cost=15829.44..15829.44 rows=36 width=48) (actual time=6354.045..6354.045 rows=60025 loops=1)

                                ->  Merge Join  (cost=14445.58..15829.44 rows=36 width=48) (actual time=3192.179..6217.530 rows=60025 loops=1)

                                      Merge Cond: (((lt.direccioncallepasaje)::text = (d.callepasaje)::text) AND ((lt.direccionnumero)::text = (d.numero)::text) AND
                                      ((lt.direccionzona)::text = (d.zona)::text) AND ((lt.direcciondatosadicionales)::text = (d.datosadicionales)::text))

                                      ->  Sort  (cost=8133.75..8287.54 rows=61515 width=48) (actual time=1519.120..2864.194 rows=61515 loops=1)

                                            Sort Key: lt.direccioncallepasaje, lt.direccionnumero, lt.direccionzona, lt.direcciondatosadicionales

                                            Sort Method:  external merge  Disk: 3640kB

                                            ->  Seq Scan on lineatelefonica lt  (cost=0.00..1347.15 rows=61515 width=48) (actual time=0.025..76.867 rows=61515 loops=1)

                                      ->  Materialize  (cost=6311.77..6936.62 rows=49988 width=44) (actual time=1672.979..1769.850 rows=60138 loops=1)

                                            ->  Sort  (cost=6311.77..6436.74 rows=49988 width=44) (actual time=1672.954..1719.629 rows=49988 loops=1)

                                                  Sort Key: d.callepasaje, d.numero, d.zona, d.datosadicionales

                                                  Sort Method:  external sort  Disk: 2856kB
```

```
                                        ->  Seq Scan on direccion d  (cost=0.00..957.88 rows=49988 width=44) (actual time=0.089..40.785 rows=49988 loops=1)

                  ->  Hash  (cost=1.11..1.11 rows=11 width=43) (actual time=0.098..0.098 rows=11 loops=1)

                        ->  Seq Scan on condiciondeservicio cs  (cost=0.00..1.11 rows=11 width=43) (actual time=0.064..0.070 rows=11 loops=1)

            ->  Index Scan using factura_pkey on factura f  (cost=0.00..3.87 rows=1 width=27) (actual time=0.037..0.037 rows=1 loops=1087824)

                  Index Cond: ((f.codigo)::text = (gf.codigofactura)::text)

                  Filter: ((f.fechadeemision >= '2008-06-01 00:00:00'::timestamp without time zone) AND
                  (f.fechadeemision <= '2008-12-01 00:00:00'::timestamp without time zone))

      ->  Index Scan using lineatelefonica_pkey on lineatelefonica ltr  (cost=0.00..3.32 rows=1 width=4)
      (actual time=0.005..0.007 rows=1 loops=634564)

            Index Cond: (ltr.numero = gf.numerolineatelefoicaresidencial)
```

5.1.3. Alternativa 2

Indice de Hash y B+ agrupado para dirección en LT

```
HashAggregate  (cost=27173.16..27175.80 rows=117 width=104) (actual time=73147.657..73907.229 rows=349069 loops=1)

  ->  Nested Loop  (cost=14498.13..27170.53 rows=117 width=104) (actual time=4030.026..66990.045 rows=634564 loops=1)

        ->  Nested Loop  (cost=14498.13..26780.40 rows=117 width=104) (actual time=4029.965..61365.131 rows=634564 loops=1)

              ->  Hash Join  (cost=14498.13..25952.81 rows=213 width=96) (actual time=4029.434..18855.020 rows=1087824 loops=1)

                    Hash Cond: (((gf.codigocondiciondeservicio)::text = (cs.codigodeservicio)::text) AND (gf.inicioperiodovigenciacondserv = cs.inicioperiodovigencia) AND
                    (gf.finperiodovigenciacondserv = cs.finperiodovigencia))

                    ->  Nested Loop  (cost=14496.83..25939.14 rows=910 width=102) (actual time=4029.283..15501.389 rows=1453572 loops=1)

                          ->  Merge Join  (cost=14493.33..15826.55 rows=38 width=48) (actual time=4029.084..6091.873 rows=60025 loops=1)

                                Merge Cond: (((d.callepasaje)::text = (lt.direccioncallepasaje)::text) AND ((d.numero)::text = (lt.direccionnumero)::text) AND
                                ((d.zona)::text = (lt.direccionzona)::text) AND ((d.datosadicionales)::text = (lt.direcciondatosadicionales)::text))

                                ->  Sort  (cost=6311.77..6436.74 rows=49988 width=44) (actual time=1682.364..1751.015 rows=49988 loops=1)

                                      Sort Key: d.callepasaje, d.numero, d.zona, d.datosadicionales

                                      Sort Method:  external sort  Disk: 2856kB

                                      ->  Seq Scan on direccion d  (cost=0.00..957.88 rows=49988 width=44) (actual time=0.038..39.001 rows=49988 loops=1)

                                ->  Materialize  (cost=8133.75..8902.69 rows=61515 width=47) (actual time=2346.654..2578.908 rows=61515 loops=1)

                                      ->  Sort  (cost=8133.75..8287.54 rows=61515 width=47) (actual time=2346.629..2475.643 rows=61515 loops=1)

                                            Sort Key: lt.direccioncallepasaje, lt.direccionnumero, lt.direccionzona, lt.direcciondatosadicionales

                                            Sort Method:  external sort  Disk: 3656kB
```

```
                                    ->  Seq Scan on lineatelefonica lt  (cost=0.00..1347.15 rows=61515 width=47) (actual time=0.074..77.713 rows=61515 loops=1)

                        ->  Bitmap Heap Scan on generafacturacion gf  (cost=3.50..264.36 rows=141 width=58)
                        (actual time=0.103..0.121 rows=24 loops=60025)

                              Recheck Cond: (gf.numerofacturado = lt.numero)

                              ->  Bitmap Index Scan on idx_num_facturado  (cost=0.00..3.46 rows=141 width=0)
                              (actual time=0.076..0.076 rows=24 loops=60025)

                                    Index Cond: (gf.numerofacturado = lt.numero)

                  ->  Hash  (cost=1.11..1.11 rows=11 width=43) (actual time=0.102..0.102 rows=11 loops=1)

                        ->  Seq Scan on condiciondeservicio cs  (cost=0.00..1.11 rows=11 width=43) (actual time=0.067..0.078 rows=11 loops=1)

            ->  Index Scan using factura_pkey on factura f  (cost=0.00..3.87 rows=1 width=27) (actual time=0.037..0.037 rows=1 loops=1087824)

                  Index Cond: ((f.codigo)::text = (gf.codigofactura)::text)

                  Filter: ((f.fechadeemision >= '2008-06-01 00:00:00'::timestamp without time zone) AND

                  (f.fechadeemision <= '2008-12-01 00:00:00'::timestamp without time zone))

      ->  Index Scan using idx_numero on lineatelefonica ltr  (cost=0.00..3.32 rows=1 width=4) (actual time=0.005..0.007 rows=1 loops=634564)

            Index Cond: (ltr.numero = gf.numerolineatelefoicaresidencial)
```


5.2. Consulta OT

5.2.1. Alternativa 0

Sin índices

```
Hash Left Join  (cost=3866.62..5972.72 rows=781 width=131) (actual time=463.076..591.448 rows=729 loops=1)

  Hash Cond: ((otg.codigo)::text = (agot.codigoordendetrabajo)::text)

  ->  Hash Join  (cost=3645.74..5736.05 rows=781 width=151) (actual time=450.013..576.172 rows=729 loops=1)

        Hash Cond: ((pp.ci)::text = (s.ceduladeidentidad)::text)

        ->  Seq Scan on persona pp  (cost=0.00..1895.00 rows=50000 width=25) (actual time=0.066..100.442 rows=50000 loops=1)

        ->  Hash  (cost=3635.98..3635.98 rows=781 width=142) (actual time=417.734..417.734 rows=729 loops=1)

              ->  Hash Join  (cost=2321.69..3635.98 rows=781 width=142) (actual time=306.206..416.024 rows=729 loops=1)

                    Hash Cond: ((s.ceduladeidentidad)::text = (ot.cisocio)::text)

                    ->  Seq Scan on socio s  (cost=0.00..700.10 rows=48510 width=8) (actual time=0.038..60.406 rows=48510 loops=1)

                    ->  Hash  (cost=2311.93..2311.93 rows=781 width=134) (actual time=306.076..306.076 rows=754 loops=1)

                          ->  Hash Join  (cost=1924.81..2311.93 rows=781 width=134) (actual time=284.080..304.567 rows=754 loops=1)
```

```
Hash Cond: ((ots.codigo)::text = (ote.codigo)::text)

->  Hash Left Join  (cost=228.10..578.04 rows=7833 width=44) (actual time=24.450..64.111 rows=7833 loops=1)

      Hash Cond: ((ots.codigo)::text = (asot.codigoordendetrabajo)::text)

      ->  Seq Scan on ordendetrabajo ots  (cost=0.00..189.33 rows=7833 width=32)
      (actual time=0.022..15.515 rows=7833 loops=1)

      ->  Hash  (cost=137.49..137.49 rows=7249 width=44) (actual time=24.378..24.378 rows=7249 loops=1)

            ->  Seq Scan on autoriza_supervisor_ordendetrabajo asot  (cost=0.00..137.49 rows=7249 width=44)
            (actual time=0.026..16.575 rows=7249 loops=1)

->  Hash  (cost=1686.94..1686.94 rows=781 width=186) (actual time=215.699..215.699 rows=754 loops=1)

      ->  Hash Join  (cost=1298.23..1686.94 rows=781 width=186) (actual time=194.587..214.055 rows=754 loops=1)

            Hash Cond: ((oto.codigo)::text = (ote.codigo)::text)

            ->  Hash Left Join  (cost=232.68..584.21 rows=7833 width=44) (actual time=28.733..91.879 rows=7833 loops=1)

                  Hash Cond: ((oto.codigo)::text = (aoot.codigoordendetrabajo)::text)

                  ->  Seq Scan on ordendetrabajo oto  (cost=0.00..189.33 rows=7833 width=32)
                  (actual time=0.019..14.234 rows=7833 loops=1)

                  ->  Hash  (cost=140.08..140.08 rows=7408 width=44) (actual time=28.672..28.672 rows=7408 loops=1)

                        ->  Seq Scan on autoriza_operador_ordendetrabajo aoot  (cost=0.00..140.08 rows=7408 width=44)
                        (actual time=0.025..5.754 rows=7408 loops=1)

            ->  Hash  (cost=1055.79..1055.79 rows=781 width=142) (actual time=116.540..116.540 rows=754 loops=1)

                  ->  Hash Join  (cost=688.84..1055.79 rows=781 width=142) (actual time=95.950..115.025 rows=754 loops=1)

                        Hash Cond: ((ote.codigo)::text = (ot.codigo)::text)

                        ->  Hash Left Join  (cost=194.72..524.49 rows=7833 width=44)
                        (actual time=27.931..57.912 rows=7833 loops=1)

                              Hash Cond: ((ote.codigo)::text = (eje.codigoordendetrabajo)::text)

                              ->  Seq Scan on ordendetrabajo ote  (cost=0.00..189.33 rows=7833 width=32)
                              (actual time=0.018..4.338 rows=7833 loops=1)

                              ->  Hash  (cost=117.10..117.10 rows=6210 width=44) (actual time=27.870..27.870 rows=6210 loops=1)

                                    ->  Seq Scan on ejecuta_tecnico_ordendetrabajo eje  (cost=0.00..117.10 rows=6210 width=44)
                                    (actual time=0.024..21.185 rows=6210 loops=1)

                        ->  Hash  (cost=484.35..484.35 rows=781 width=98) (actual time=39.965..39.965 rows=754 loops=1)

                              ->  Hash Join  (cost=257.84..484.35 rows=781 width=98)
                              (actual time=35.152..38.881 rows=754 loops=1)

                                    Hash Cond: ((otg.codigo)::text = (ot.codigo)::text)
```

```
                                                    ->  Seq Scan on ordendetrabajo otg  (cost=0.00..189.33 rows=7833 width=32)
                                                    (actual time=0.018..4.090 rows=7833 loops=1)

                                                    ->  Hash  (cost=248.08..248.08 rows=781 width=66)
                                                    (actual time=16.516..16.516 rows=754 loops=1)

                                                          ->  Seq Scan on ordendetrabajo ot  (cost=0.00..248.08 rows=781 width=66)
                                                          (actual time=12.251..15.456 rows=754 loops=1)

                                                                Filter: ((fechacreacion >= '2008-09-01 00:00:00'::timestamp without time zone) AND
                                                                (fechacreacion <= '2008-12-31 00:00:00'::timestamp without time zone) AND
                                                                ((estado)::text <> 'Terminado'::text))

  ->  Hash  (cost=133.17..133.17 rows=7017 width=44) (actual time=12.983..12.983 rows=7017 loops=1)

        ->  Seq Scan on autoriza_gerenteadjunto_ordendetrabajo agot  (cost=0.00..133.17 rows=7017 width=44)
        (actual time=0.026..5.245 rows=7017 loops=1)
```

5.2.2. Alternativa 1

Indices de Hash para códigos de OT y árbol B+ para fecha

```
Hash Left Join  (cost=3744.08..5850.18 rows=781 width=131) (actual time=440.496..530.800 rows=729 loops=1)

  Hash Cond: ((otg.codigo)::text = (agot.codigoordendetrabajo)::text)

  ->  Hash Join  (cost=3523.20..5613.51 rows=781 width=151) (actual time=414.292..502.398 rows=729 loops=1)

        Hash Cond: ((pp.ci)::text = (s.ceduladeidentidad)::text)

        ->  Seq Scan on persona pp  (cost=0.00..1895.00 rows=50000 width=25) (actual time=0.077..78.940 rows=50000 loops=1)

        ->  Hash  (cost=3513.44..3513.44 rows=781 width=142) (actual time=395.278..395.278 rows=729 loops=1)

              ->  Hash Join  (cost=2199.15..3513.44 rows=781 width=142) (actual time=302.573..393.597 rows=729 loops=1)

                    Hash Cond: ((s.ceduladeidentidad)::text = (ot.cisocio)::text)

                    ->  Seq Scan on socio s  (cost=0.00..700.10 rows=48510 width=8) (actual time=0.027..40.485 rows=48510 loops=1)

                    ->  Hash  (cost=2189.39..2189.39 rows=781 width=134) (actual time=302.459..302.459 rows=754 loops=1)

                          ->  Hash Join  (cost=1800.67..2189.39 rows=781 width=134) (actual time=291.599..300.918 rows=754 loops=1)

                                Hash Cond: ((oto.codigo)::text = (ote.codigo)::text)

                                ->  Hash Left Join  (cost=232.68..584.21 rows=7833 width=44) (actual time=34.408..58.251 rows=7833 loops=1)

                                      Hash Cond: ((oto.codigo)::text = (aoot.codigoordendetrabajo)::text)

                                      ->  Seq Scan on ordendetrabajo oto  (cost=0.00..189.33 rows=7833 width=32)
                                      (actual time=0.022..4.452 rows=7833 loops=1)

                                      ->  Hash  (cost=140.08..140.08 rows=7408 width=44) (actual time=34.340..34.340 rows=7408 loops=1)

                                            ->  Seq Scan on autoriza_operador_ordendetrabajo aoot  (cost=0.00..140.08 rows=7408 width=44)
```

```
            (actual time=0.030..5.765 rows=7408 loops=1)

->  Hash  (cost=1558.23..1558.23 rows=781 width=186) (actual time=219.336..219.336 rows=754 loops=1)

      ->  Hash Join  (cost=1171.11..1558.23 rows=781 width=186) (actual time=193.548..217.556 rows=754 loops=1)

            Hash Cond: ((ots.codigo)::text = (ote.codigo)::text)

            ->  Hash Left Join  (cost=228.10..578.04 rows=7833 width=44) (actual time=28.706..74.845 rows=7833 loops=1)

                  Hash Cond: ((ots.codigo)::text = (asot.codigoordendetrabajo)::text)

                  ->  Seq Scan on ordendetrabajo ots  (cost=0.00..189.33 rows=7833 width=32)
                  (actual time=0.019..26.254 rows=7833 loops=1)

                  ->  Hash  (cost=137.49..137.49 rows=7249 width=44) (actual time=28.644..28.644 rows=7249 loops=1)

                        ->  Seq Scan on autoriza_supervisor_ordendetrabajo asot  (cost=0.00..137.49 rows=7249 width=44)
                        (actual time=0.027..5.701 rows=7249 loops=1)

            ->  Hash  (cost=933.24..933.24 rows=781 width=142) (actual time=125.345..125.345 rows=754 loops=1)

                  ->  Hash Join  (cost=566.30..933.24 rows=781 width=142)
                  (actual time=99.056..123.810 rows=754 loops=1)

                        Hash Cond: ((ote.codigo)::text = (ot.codigo)::text)

                        ->  Hash Left Join  (cost=194.72..524.49 rows=7833 width=44)
                        (actual time=19.165..73.356 rows=7833 loops=1)

                              Hash Cond: ((ote.codigo)::text = (eje.codigoordendetrabajo)::text)

                              ->  Seq Scan on ordendetrabajo ote  (cost=0.00..189.33 rows=7833 width=32)
                               (actual time=0.018..16.627 rows=7833 loops=1)

                              ->  Hash  (cost=117.10..117.10 rows=6210 width=44)
                               (actual time=19.110..19.110 rows=6210 loops=1)

                                    ->  Seq Scan on ejecuta_tecnico_ordendetrabajo eje  (cost=0.00..117.10 rows=6210 width=44)
                                     (actual time=0.026..12.400 rows=6210 loops=1)

                        ->  Hash  (cost=361.81..361.81 rows=781 width=98) (actual time=45.234..45.234 rows=754 loops=1)

                              ->  Hash Join  (cost=135.30..361.81 rows=781 width=98)
                              (actual time=24.231..28.308 rows=754 loops=1)

                                    Hash Cond: ((otg.codigo)::text = (ot.codigo)::text)

                                    ->  Seq Scan on ordendetrabajo otg  (cost=0.00..189.33 rows=7833 width=32)
                                    (actual time=0.018..4.676 rows=7833 loops=1)

                                    ->  Hash  (cost=125.53..125.53 rows=781 width=66) (actual time=17.918..17.918 rows=754 loops=1)

                                          ->  Index Scan using idx_fechacreacion on ordendetrabajo ot  (cost=0.00..125.53 rows=781 width=66)
                                           (actual time=0.095..16.820 rows=754 loops=1)

                                                Index Cond: ((fechacreacion >= '2008-09-01 00:00:00'::timestamp without time zone) AND
                                                (fechacreacion <= '2008-12-31 00:00:00'::timestamp without time zone))
```

```
                                                                    Filter: ((estado)::text <> 'Terminado'::text)

  ->  Hash  (cost=133.17..133.17 rows=7017 width=44) (actual time=26.121..26.121 rows=7017 loops=1)

        ->  Seq Scan on autoriza_gerenteadjunto_ordendetrabajo agot  (cost=0.00..133.17 rows=7017 width=44)
        (actual time=0.028..5.832 rows=7017 loops=1)
```

5.2.3. Alternativa 2

Arboles B+

```
Nested Loop Left Join  (cost=3645.74..5999.21 rows=781 width=131) (actual time=430.324..596.370 rows=729 loops=1)

  ->  Hash Join  (cost=3645.74..5736.05 rows=781 width=151) (actual time=430.243..555.637 rows=729 loops=1)

        Hash Cond: ((pp.ci)::text = (s.ceduladeidentidad)::text)

        ->  Seq Scan on persona pp  (cost=0.00..1895.00 rows=50000 width=25) (actual time=0.028..66.387 rows=50000 loops=1)

        ->  Hash  (cost=3635.98..3635.98 rows=781 width=142) (actual time=427.448..427.448 rows=729 loops=1)

              ->  Hash Join  (cost=2321.69..3635.98 rows=781 width=142) (actual time=312.504..425.772 rows=729 loops=1)

                    Hash Cond: ((s.ceduladeidentidad)::text = (ot.cisocio)::text)

                    ->  Seq Scan on socio s  (cost=0.00..700.10 rows=48510 width=8) (actual time=0.028..58.840 rows=48510 loops=1)

                    ->  Hash  (cost=2311.93..2311.93 rows=781 width=134) (actual time=312.386..312.386 rows=754 loops=1)

                          ->  Hash Join  (cost=1924.81..2311.93 rows=781 width=134) (actual time=297.833..310.574 rows=754 loops=1)

                                Hash Cond: ((ots.codigo)::text = (ote.codigo)::text)

                                ->  Hash Left Join  (cost=228.10..578.04 rows=7833 width=44) (actual time=13.797..51.650 rows=7833 loops=1)

                                      Hash Cond: ((ots.codigo)::text = (asot.codigoordendetrabajo)::text)

                                      ->  Seq Scan on ordendetrabajo ots  (cost=0.00..189.33 rows=7833 width=32)
                                      (actual time=0.022..4.370 rows=7833 loops=1)

                                      ->  Hash  (cost=137.49..137.49 rows=7249 width=44) (actual time=13.730..13.730 rows=7249 loops=1)

                                            ->  Seq Scan on autoriza_supervisor_ordendetrabajo asot  (cost=0.00..137.49 rows=7249 width=44)
                                            (actual time=0.028..5.908 rows=7249 loops=1)

                                ->  Hash  (cost=1686.94..1686.94 rows=781 width=186) (actual time=253.361..253.361 rows=754 loops=1)

                                      ->  Hash Join  (cost=1298.23..1686.94 rows=781 width=186) (actual time=224.509..251.651 rows=754 loops=1)

                                            Hash Cond: ((oto.codigo)::text = (ote.codigo)::text)

                                            ->  Hash Left Join  (cost=232.68..584.21 rows=7833 width=44) (actual time=32.565..86.032 rows=7833 loops=1)

                                                  Hash Cond: ((oto.codigo)::text = (aoot.codigoordendetrabajo)::text)
```

```
                                    ->  Seq Scan on ordendetrabajo oto  (cost=0.00..189.33 rows=7833 width=32)
                                    (actual time=0.018..4.729 rows=7833 loops=1)

                                    ->  Hash  (cost=140.08..140.08 rows=7408 width=44) (actual time=32.506..32.506 rows=7408 loops=1)

                                          ->  Seq Scan on autoriza_operador_ordendetrabajo aoot  (cost=0.00..140.08 rows=7408 width=44)
                                          (actual time=0.028..5.803 rows=7408 loops=1)

                              ->  Hash  (cost=1055.79..1055.79 rows=781 width=142) (actual time=160.073..160.073 rows=754 loops=1)

                                    ->  Hash Join  (cost=688.84..1055.79 rows=781 width=142)
                                    (actual time=135.517..152.800 rows=754 loops=1)

                                          Hash Cond: ((ote.codigo)::text = (ot.codigo)::text)

                                          ->  Hash Left Join  (cost=194.72..524.49 rows=7833 width=44)
                                          (actual time=43.669..115.356 rows=7833 loops=1)

                                                Hash Cond: ((ote.codigo)::text = (eje.codigoordendetrabajo)::text)

                                                ->  Seq Scan on ordendetrabajo ote  (cost=0.00..189.33 rows=7833 width=32)
                                                (actual time=0.019..12.712 rows=7833 loops=1)

                                                ->  Hash  (cost=117.10..117.10 rows=6210 width=44) (actual time=43.616..43.616 rows=6210 loops=1)

                                                      ->  Seq Scan on ejecuta_tecnico_ordendetrabajo eje  (cost=0.00..117.10 rows=6210 width=44)
                                                      (actual time=0.030..35.482 rows=6210 loops=1)

                                          ->  Hash  (cost=484.35..484.35 rows=781 width=98) (actual time=32.225..32.225 rows=754 loops=1)

                                                ->  Hash Join  (cost=257.84..484.35 rows=781 width=98)
                                                (actual time=27.466..31.189 rows=754 loops=1)

                                                      Hash Cond: ((otg.codigo)::text = (ot.codigo)::text)

                                                      ->  Seq Scan on ordendetrabajo otg  (cost=0.00..189.33 rows=7833 width=32)
                                                      (actual time=0.018..4.097 rows=7833 loops=1)

                                                      ->  Hash  (cost=248.08..248.08 rows=781 width=66)
                                                      (actual time=7.604..7.604 rows=754 loops=1)

                                                            ->  Seq Scan on ordendetrabajo ot  (cost=0.00..248.08 rows=781 width=66)
                                                            (actual time=3.402..6.568 rows=754 loops=1)

                                                                  Filter: ((fechacreacion >= '2008-09-01 00:00:00'::timestamp without time zone) AND
                                                                  (fechacreacion <= '2008-12-31 00:00:00'::timestamp without time zone) AND
                                                                  ((estado)::text <> 'Terminado'::text))

->  Index Scan using idx_codigo_otg on autoriza_gerenteadjunto_ordendetrabajo agot  (cost=0.00..0.32 rows=1 width=44)
(actual time=0.053..0.054 rows=1 loops=729)

      Index Cond: ((otg.codigo)::text = (agot.codigoordendetrabajo)::text)
```

5.2.4. Alternativa 3

Arboles B+, Hash para estado de OT

```
Nested Loop Left Join  (cost=3645.74..5999.21 rows=781 width=131) (actual time=394.901..548.620 rows=729 loops=1)
  ->  Hash Join  (cost=3645.74..5736.05 rows=781 width=151) (actual time=394.820..508.218 rows=729 loops=1)
        Hash Cond: ((pp.ci)::text = (s.ceduladeidentidad)::text)
        ->  Seq Scan on persona pp  (cost=0.00..1895.00 rows=50000 width=25) (actual time=0.024..69.035 rows=50000 loops=1)
        ->  Hash  (cost=3635.98..3635.98 rows=781 width=142) (actual time=392.224..392.224 rows=729 loops=1)
              ->  Hash Join  (cost=2321.69..3635.98 rows=781 width=142) (actual time=308.600..387.973 rows=729 loops=1)
                    Hash Cond: ((s.ceduladeidentidad)::text = (ot.cisocio)::text)
                    ->  Seq Scan on socio s  (cost=0.00..700.10 rows=48510 width=8) (actual time=0.026..31.226 rows=48510 loops=1)
                    ->  Hash  (cost=2311.93..2311.93 rows=781 width=134) (actual time=308.478..308.478 rows=754 loops=1)
                          ->  Hash Join  (cost=1924.81..2311.93 rows=781 width=134) (actual time=291.248..306.911 rows=754 loops=1)
                                Hash Cond: ((ots.codigo)::text = (ote.codigo)::text)
                                ->  Hash Left Join  (cost=228.10..578.04 rows=7833 width=44) (actual time=30.414..88.424 rows=7833 loops=1)
                                      Hash Cond: ((ots.codigo)::text = (asot.codigoordendetrabajo)::text)
                                      ->  Seq Scan on ordendetrabajo ots  (cost=0.00..189.33 rows=7833 width=32)
                                      (actual time=0.023..5.806 rows=7833 loops=1)
                                      ->  Hash  (cost=137.49..137.49 rows=7249 width=44) (actual time=30.349..30.349 rows=7249 loops=1)
                                            ->  Seq Scan on autoriza_supervisor_ordendetrabajo asot  (cost=0.00..137.49 rows=7249 width=44)
                                            (actual time=0.027..22.641 rows=7249 loops=1)
                                ->  Hash  (cost=1686.94..1686.94 rows=781 width=186) (actual time=212.540..212.540 rows=754 loops=1)
                                      ->  Hash Join  (cost=1298.23..1686.94 rows=781 width=186) (actual time=188.413..210.889 rows=754 loops=1)
                                            Hash Cond: ((oto.codigo)::text = (ote.codigo)::text)
                                            ->  Hash Left Join  (cost=232.68..584.21 rows=7833 width=44) (actual time=24.102..63.736 rows=7833 loops=1)
                                                  Hash Cond: ((oto.codigo)::text = (aoot.codigoordendetrabajo)::text)
                                                  ->  Seq Scan on ordendetrabajo oto  (cost=0.00..189.33 rows=7833 width=32)
                                                  (actual time=0.019..4.421 rows=7833 loops=1)
                                                  ->  Hash  (cost=140.08..140.08 rows=7408 width=44) (actual time=24.041..24.041 rows=7408 loops=1)
                                                        ->  Seq Scan on autoriza_operador_ordendetrabajo aoot  (cost=0.00..140.08 rows=7408 width=44)
                                                        (actual time=0.028..5.785 rows=7408 loops=1)
                                            ->  Hash  (cost=1055.79..1055.79 rows=781 width=142) (actual time=131.284..131.284 rows=754 loops=1)
                                                  ->  Hash Join  (cost=688.84..1055.79 rows=781 width=142) (actual time=108.191..129.848 rows=754 loops=1)
                                                        Hash Cond: ((ote.codigo)::text = (ot.codigo)::text)
```

```
                                        ->  Hash Left Join  (cost=194.72..524.49 rows=7833 width=44)
                                        (actual time=37.229..73.557 rows=7833 loops=1)

                                              Hash Cond: ((ote.codigo)::text = (eje.codigoordendetrabajo)::text)

                                              ->  Seq Scan on ordendetrabajo ote  (cost=0.00..189.33 rows=7833 width=32)
                                              (actual time=0.018..5.345 rows=7833 loops=1)

                                              ->  Hash  (cost=117.10..117.10 rows=6210 width=44) (actual time=37.172..37.172 rows=6210 loops=1)

                                                    ->  Seq Scan on ejecuta_tecnico_ordendetrabajo eje  (cost=0.00..117.10 rows=6210 width=44)
                                                     (actual time=0.032..4.748 rows=6210 loops=1)

                                        ->  Hash  (cost=484.35..484.35 rows=781 width=98) (actual time=38.775..38.775 rows=754 loops=1)

                                              ->  Hash Join  (cost=257.84..484.35 rows=781 width=98) (actual time=33.965..37.687 rows=754 loops=1)

                                                    Hash Cond: ((otg.codigo)::text = (ot.codigo)::text)

                                                    ->  Seq Scan on ordendetrabajo otg  (cost=0.00..189.33 rows=7833 width=32)
                                                    (actual time=0.018..13.773 rows=7833 loops=1)

                                                    ->  Hash  (cost=248.08..248.08 rows=781 width=66)
                                                    (actual time=7.717..7.717 rows=754 loops=1)

                                                          ->  Seq Scan on ordendetrabajo ot  (cost=0.00..248.08 rows=781 width=66)
                                                          (actual time=3.398..6.674 rows=754 loops=1)

                                                                Filter: ((fechacreacion >= '2008-09-01 00:00:00'::timestamp without time zone) AND
                                                                (fechacreacion <= '2008-12-31 00:00:00'::timestamp without time zone) AND
                                                                ((estado)::text <> 'Terminado'::text))

->  Index Scan using idx_codigo_otg on autoriza_gerenteadjunto_ordendetrabajo agot  (cost=0.00..0.32 rows=1 width=44)
(actual time=0.038..0.039 rows=1 loops=729)

      Index Cond: ((otg.codigo)::text = (agot.codigoordendetrabajo)::text)
```