ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA EN SISTEMAS

“ANÁLISIS DE TÉCNICAS PARA TUNING DE UN DATA WAREHOUSE EN UN SISTEMA DE TOMA DE DECISIONES UTILIZANDO MICROSOFT SQL SERVER”

TESIS DE GRADO

Previa la obtención del título de

INGENIERO EN SISTEMAS INFORMÁTICOS

Presentado por:

María Isabel Uvidia Fassler

Riobamba - Abril

2011

Gracias Dios por haberme llenado de experiencias nuevas en esta etapa que culmina, al hacer que el desarrollo de mi tesis este acompañada de grandes mujeres que me demostraron que detrás de una mirada sutil y tierna hay una persona capaz, inteligente con pasos firmes a la meta, con muchas virtudes pero sobretodo por haber sido consejeras y amigas.

Un profundo agradecimiento al Tribunal de Tesis, Ing. Ivonne Rodríguez e Ing. Gloria Arcos que con el conocimiento y la capacidad supieron guiarme en los procesos más difíciles de esta tesis

A la Ing. Alejandra Oñate por todo el apoyo incondicional prestado durante el trabajo en la Unidad Técnica de Planificación.

Con todo mi cariño dedico esta tesis a mis padres, Fanny y John que con el sacrificio y perseverancia diaria me enseñaron que siempre tengo que luchar por mis ideales y sueños.

A mis hermanos Johncito y Johana por hacerme sentir que mis metas alcanzadas son su inspiración.

A David que con su amor, paciencia y apoyo me ayudó a caminar con pasos más firmes a la meta.

A mis amigos que siempre me acompañan en los buenos y malos momentos, demostrándome con una sonrisa la verdadera solución a un problema.

FIRMAS DE RESPONSABILIDAD

NOMBRES	**FIRMAS**	**FECHA**
Ing. Iván Menes **DECANO DE LA FACULTAD** **INFORMÁTICA Y ELECTRÓNICA**	_____	_____
Ing. Raúl Rosero **DIRECTOR DE ESCUELA** **INGENIERÍA EN SISTEMAS**	_____	_____
Ing. Ivonne Rodríguez **DIRECTOR DE TESIS**	_____	_____
Ing. Gloria Arcos **MIEMBRO DE TESIS**	_____	_____
Tlgo. Carlos Rodríguez **DIRECTOR DEL** **CENTRO DE** **DOCUMENTACIÓN**	_____	_____

NOTA: _____

FIRMA DE RESPONSABILIDAD DEL AUTOR

“Yo, María Isabel Uvidia Fassler, soy responsable de las ideas, doctrinas y resultados expuestos en esta Tesis; y, el patrimonio intelectual de la Tesis de Grado pertenece a la ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO”

_____

María Isabel Uvidia Fassler

INDICE GENERAL

AGRADECIMIENTO

DEDICATORIA

FIRMAS DE RESPONSABILIDAD

FIRMA DE RESPONSABILIDAD DEL AUTOR

INDICE GENERAL

ÍNDICE DE ABREVIATURAS Y ACRÓNIMOS

INDICE TABLAS

INDICE FIGURAS

INTRODUCCIÓN

CAPÍTULO I: MARCO REFERENCIAL

1.1. ANTECEDENTES ..... - 20 -
1.2. JUSTIFICACIÓN ..... - 21 -
1.2.1. JUSTIFICACIÓN TEÓRICA ..... - 21 -
1.2.2. JUSTIFICACIÓN METÓDOLÓGICA ..... - 21 -
1.2.3. JUSTIFICACIÓN APLICATIVA ..... - 21 -
1.3. OBJETIVOS ..... - 22 -
1.3.1. OBJETIVO GENERAL ..... - 22 -
1.3.2. OBJETIVOS ESPECÍFICOS ..... - 22 -
1.4. HIPÓTESIS ..... - 23 -

CAPÍTULO II: MARCO TEÓRICO

2.1. INTRODUCCIÓN ..... - 24 -
2.2. DATA WAREHOUSE ..... - 24 -
2.2.1. ¿QUÉ ES UN DATA WAREHOUSE? ..... - 25 -
2.2.2. CARACTERÍSTICAS ..... - 26 -
2.2.3. OBJETIVOS DEL DATA WAREHOUSE ..... - 28 -
2.2.4. ARQUITECTURA ..... - 29 -
2.2.4.1. ELEMENTOS CONSTITUYENTES DE UNA ARQUITECTURA DATA WAREHOUSE ..... - 30 -
2.2.5. TABLA DE HECHOS (FACT) ..... - 34 -
2.2.6. DIMENSIONES ..... - 35 -
2.2.7. CONSTRUCCIÓN DEL DATA WAREHOUSE ..... - 36 -
2.2.8. VENTAJAS ..... - 38 -
2.2.9. DESVENTAJAS ..... - 39 -
2.3. DATA MART ..... - 40 -
2.4. BASES DE DATOS OLTP Y OLAP ..... - 41 -
2.4.1. OLTP - On-Line Transactional Processing ..... - 41 -
2.4.2. OLAP - On-Line Analytical Processing ..... - 41 -
2.5. BUSINESS INTELLIGENCE ..... - 42 -
2.5.1. ¿QUÉ ES BUSINESS INTELLIGENCE? ..... - 42 -
2.5.2. BI COMO SOLUCIÓN TECNOLÓGICA ..... - 44 -

2.6. SISTEMAS DE TOMA DE DECISIONES ..... - 45 -
2.6.1. DEFINICIÓN DE SISTEMAS DE SOPORTE DE TOMA DE DECISIONES ... - 45 -
2.6.2. CARACTERÍSTICAS ..... - 46 -
2.6.3. CLASIFICACIÓN ..... - 47 -
2.6.4. TIPOS ..... - 48 -
2.6.5. DIFERENCIA CON OTRAS HERRAMIENTAS DE BUSINESS INTELLIGENCE . - 49 -
2.7. BUSINESS OBJECTS (BO) ..... - 50 -
2.8. TUNING ..... - 51 -
2.8.1. TUNING DEL DATA WAREHOUSE ..... - 52 -
2.9. RENDIMIENTO ..... - 53 -
2.9.1. CÓMO SE MIDE EL RENDIMIENTO ..... - 53 -
2.9.2. ANÁLISIS DE RENDIMIENTO ..... - 53 -
2.9.3. FACTORES QUE TIENEN IMPACTO EN EL RENDIMIENTO ..... - 54 -
2.9.4. OBJETIVOS DE RENDIMIENTO ..... - 55 -
2.10. SQL SERVER 2005 ..... - 56 -
2.10.1. RECURSOS DE SQL SERVER ..... - 56 -
2.10.2. INDICES EN SQL SERVER ..... - 57 -
2.10.2.1. ¿QUÉ ES UN INDICE? ..... - 58 -
2.10.2.2. INDICE CLUSTERED ..... - 58 -
2.10.2.3. INDICE NON-CLUSTERED ..... - 60 -
2.10.2.4. GUÍA GENERAL DE USO DE ÍNDICES ..... - 62 -
2.10.3. BLOQUEO DE CONSULTAS SQL ..... - 62 -

CAPÍTULO III: ANÁLISIS Y SELECCIÓN DE TÉCNICAS TUNING

3.1. INTRODUCCIÓN ..... - 65 -
3.2. PROCESO DE INGENIERÍA DE PERFORMANCE O RENDIMIENTO ..... - 66 -
3.2.1. CRITERIOS DE ANÁLISIS (CAPAS TUNING) ..... - 68 -
3.3. DEFINICIÓN Y ANALISIS DE PARÁMETROS ..... - 69 -
3.4. SELECCIÓN DE TÉCNICAS TUNING ..... - 70 -
3.4.1. DBMS (servidor de BD-Software) ..... - 71 -
3.4.2. DISEÑO DEL DATA WAREHOUSE ..... - 76 -
3.4.2.1. DESNORMALIZACIÓN ..... - 76 -
3.4.2.2. CREACIÓN DE FILEGROUPS ..... - 78 -
3.4.2.2.1. ESTRATEGIA DE ARCHIVOS Y GRUPOS DE ARCHIVOS ..... - 80 -
3.4.2.2.2. MEJORA DEL RENDIMIENTO DE BASES DE DATOS Y ALMACENES DE DATOS ..... - 81 -
3.4.2.2.3. REGLAS PARA EL DISEÑO DE ARCHIVOS Y GRUPOS DE ARCHIVOS - 81 -
3.4.2.2.4. RECOMENDACIONES ..... - 81 -
3.4.3. CONSULTAS ..... - 83 -
3.4.3.1. INDICES ..... - 83 -
3.4.3.1.1. INDICES EN SQL SERVER ..... - 84 -
3.4.3.1.2. CREACION DE ÍNDICES ..... - 85 -
3.4.3.1.3. MEJORAR EL RENDIMIENTO CON ÍNDICES CUBIERTOS ..... - 89 -
3.4.3.1.4. USO DE COLUMNAS INCLUIDAS (INCLUDED) Y LA REDUCCIÓN DE LA PROFUNDIDAD DE UN ÍNDICE ..... - 89 -
3.4.3.1.5. ÍNDICES COMPUESTOS ..... - 93 -
3.4.3.1.6. USO DE ÍNDICES AGRUPADOS ..... - 93 -
3.4.3.1.7. EL RENDIMIENTO DE LECTURA VS RENDIMIENTO DE ESCRITURA .... - 95 -

3.4.3.1.8. ANÁLISIS DE UTILIZACIÓN DE ÍNDICES - 96 -
3.4.3.1.9. RENDIMIENTO GLOBAL - 99 -
3.4.3.1.10. COMANDOS PARA ADMINISTRAR INDICES - 100 -
3.4.3.1.11. LOOKUP DE SQL SERVER - 103 -
3.4.3.2. PARTICIONAMIENTO DE TABLAS - 105 -
3.4.3.2.1. FUNCIONES DE PARTICIÓN - 105 -
3.4.3.2.2. ESQUEMAS DE PARTICIÓN - 107 -
3.4.3.2.3. CREACION DE UNA TABLA CON PARTICIONES - 107 -
3.4.3.2.4. LIMITACIONES DE MEMORIA E ÍNDICES CON PARTICIONES - 109 -
3.4.3.2.5. COMANDOS PARA ADMINISTRAR PARTICIONES - 110 -
3.4.3.3. CALIDAD DE SCRIPT SQL - 111 -
3.4.3.3.1. USANDO JOIN - 111 -
3.4.3.3.2. USANDO FUNCIONES AGREGADAS DE SQL SERVER - 111 -
3.4.3.3.3. SUBCONSULTAS - 112 -
3.4.3.3.4. ESCALAR UDF - 112 -
3.4.3.3.5. CURSORES - 113 -
3.4.3.3.6. UTILIZACIÓN DE ARGUMENTOS DE LA BÚSQUEDA - 113 -
3.4.3.3.7. ORDEN DE EJECUCION - 113 -
3.4.4. ESTADÍSTICAS SQL SERVER - 116 -
3.4.4.1. ACTUALIZACIÓN DE ESTADÍSTICAS - 117 -

CAPÍTULO IV: DESARROLLO DE TÉCNICAS DE TUNING EN EL DATA WAREHOUSE DEL PROYECTO SII-ESPOCH

4.1. INTRODUCCIÓN - 120 -
4.2. DESCRIPCIÓN DEL ESCENARIO HARDWARE - 121 -
4.2.1. JUSTIFICACIÓN DEL ESCENARIO - 123 -
4.2.1.1. EVIDENCIAS - 124 -
4.3. HERRAMIENTAS UTILIZADAS EN EL ESCENARIO - 128 -
4.4. DESCRIPCIÓN DE LA RED - 130 -
4.5. COMPARACIÓN DEL DBMS (SERVIDOR DE BD-SOFTWARE) - 131 -
4.5.1. AMBIENTE DE PRUEBA #1 - 131 -
4.5.2. OBJETIVO DEL AMBIENTE DE PRUEBA #1 - 131 -
4.5.3. ¿CÓMO SE VA A MEDIR? - 131 -
4.5.4. EXPERIMENTO - 133 -
4.5.4.1.1. SQL SERVER 2005 - 134 -
4.5.4.1.2. MYSQL 5.1 (MYISAM) - 138 -
MYSQL 5.1 (MYISAM) - 139 -
4.5.5. ANÁLISIS DE RESULTADOS DEL DBMS (servidor de BD-Software) - 142 -
4.5.5.1.1. TABLA FINAL DE RESULTADOS - 154 -
4.5.6. CONCLUSIÓN - 156 -
4.6. COMPARACIÓN DE DISEÑO DEL DATA WAREHOUSE - 156 -
4.5.7. TÉCNICA DE DESNORMALIZACIÓN - 156 -
4.5.7.1. AMBIENTE DE PRUEBA #2 - 157 -
4.5.7.2. OBJETIVO DEL AMBIENTE DE PRUEBA #2 - 157 -
4.5.7.3. ¿CÓMO SE VA A MEDIR? - 157 -
4.5.7.4. EXPERIMENTO - 157 -
4.5.7.4.1. TABLAS DE DATOS DE RESULTADO DE LA PRUEBA DE LA TÉCNICA DE DESNORMALIZACIÓN - 160 -
4.5.7.5. ANÁLISIS DE RESULTADOS DE DISEÑO DEL DATA WAREHOUSE – TÉCNICA DE DESNORMALIZACIÓN - 162 -
4.5.7.6. CONCLUSIÓN - 165 -

4.7. COMPARACIÓN DE CONSULTAS ..... - 165 -
4.7.1. TÉCNICA DE UTILIZACIÓN DE ÍNDICES ..... - 166 -
4.7.1.1. AMBIENTE DE PRUEBA #3 ..... - 166 -
4.7.1.2. OBJETIVO DEL AMBIENTE DE PRUEBA #3 ..... - 167 -
4.7.1.3. ¿CÓMO SE VA A MEDIR? ..... - 167 -
4.7.1.4. EXPERIMENTO ..... - 167 -
4.7.1.4.1. CASO 1 ..... - 169 -
4.7.1.4.2. CASO 2 ..... - 171 -
4.7.1.4.3. ÍNDICES UTILIZADOS PARA AMBOS CASOS ..... - 171 -
4.7.1.4.4. TABLAS DE DATOS DE RESULTADO DE LA PRUEBA DE LA TÉCNICA DE UTILIZACION DE ÍNDICES ..... - 172 -
4.7.1.5. ÁNALISIS DE RESULTADOS DE CONSULTAS – TÉCNICA DE UTILIZACION DE ÍNDICES ..... - 176 -
4.7.1.5.1. CASO 1 ..... - 176 -
4.7.1.5.2. CASO 2 ..... - 179 -
4.7.1.6. CONCLUSIÓN ..... - 182 -
4.7.2. TÉCNICA DE PARTICIONAMIENTO DE TABLAS ..... - 182 -
4.7.2.1. AMBIENTE DE PRUEBA #4 ..... - 183 -
4.7.2.2. OBJETIVO DEL AMBIENTE DE PRUEBA #4 ..... - 183 -
4.7.2.3. ¿CÓMO SE VA A MEDIR? ..... - 183 -
4.7.2.4. EXPERIMENTO ..... - 184 -
4.7.2.4.1. SCRIPT UTILIZADO PARA EL PARTICIONAMIENTO DE LA TABLA- 185 -
4.7.2.4.2. TABLAS DE DATOS DE RESULTADO DE LA PRUEBA DE LA TÉCNICA DE PARTICIONAMIENTO DE TABLAS ..... - 187 -
4.7.2.5. ÁNALISIS DE RESULTADOS DE CONSULTAS – TÉCNICA DE PARTICIONAMIENTO DE TABLAS ..... - 189 -
4.7.2.6. CONCLUSIÓN ..... - 192 -
4.7.3. TÉCNICA DE CALIDAD DE SCRIPT SQL ..... - 192 -
4.7.3.1. AMBIENTE DE PRUEBA #5 ..... - 193 -
4.7.3.2. OBJETIVO DEL AMBIENTE DE PRUEBA #5 ..... - 193 -
4.7.3.3. ¿CÓMO SE VA A MEDIR? ..... - 193 -
4.7.3.4. EXPERIMENTO ..... - 194 -
4.7.3.4.1. SENTENCIA SQL SELECT UTILIZADA EN EL ESCENARIO OLAP CON LA QUE SE REALIZAN LAS PRUEBAS DE LA TECNICA DE CALIDAD DE SCRIPT SQL ..... - 195 -
4.7.3.4.2. TABLAS DE DATOS DE RESULTADO DE LA PRUEBA DE LA TECNICA DE CALIDAD DE SCRIPT SQL ..... - 196 -
4.7.3.5. ÁNALISIS DE RESULTADOS DE CONSULTAS – TÉCNICA DE CALIDAD DE SCRIPT SQL ..... - 198 -
4.7.3.6. CONCLUSIÓN ..... - 201 -
4.8. RESULTADOS DE OPTIMIZACIÓN DE LAS TECNICAS ANALIZADAS ..... - 201 -

CAPÍTULO V: GUÍA DE APLICACIÓN DE TÉCNICAS TUNING

5.1. INTRODUCCIÓN ..... - 203 -
5.2. ANTECEDENTES DEL PROYECTO SII-ESPOCH ..... - 204 -
5.3. JUSTIFICACIÓN DEL PROYECTO SII-ESPOCH ..... - 204 -
5.4. INDICADORES DEL SISTEMA ACADÉMICO QUE SE OBTENDRÁN DEL SISTEMA DE TOMA DE DECISIONES ..... - 205 -
5.5. DESARROLLO DE LA GUÍA DE APLICACIÓN DE TÉCNICAS TUNING ..... - 206 -
5.3.1. TÉCNICA DE DESNORMALIZACIÓN ..... - 206 -

5.3.1.1. DATOS DUPLICADOS ..... - 207 -
5.3.1.2. TABLAS DESNORMALIZADAS ..... - 209 -
5.3.1.3. CLAVES SUBROGADAS ..... - 210 -
5.3.2. TÉCNICA DE CREACIÓN DE FILEGROUPS ..... - 211 -
5.3.2.1. CREACIÓN DE FILEGROUPS ..... - 212 -
5.3.3. TÉCNICA DE UTILIZACIÓN DE ÍNDICES ..... - 214 -
5.3.3.1. INDICES CREADOS EN EL DWH SII-ESPOCH ..... - 214 -
5.3.3.2. CRITERIOS DE OPTIMIZACIÓN PARA LA CREACIÓN DE ÍNDICES ..... - 216 -
5.3.3.3. SCRIPTS DE CREACIÓN DE ÍNDICES ..... - 216 -
5.3.3.4. CRITERIOS DE MANTENIMIENTO DE ÍNDICES ..... - 218 -
5.3.3.4.1. Reorganización de Índices ..... - 219 -
5.3.4. TÉCNICA DE CALIDAD DE SCRIPT SQL ..... - 221 -
5.3.4.1. CRITERIOS DE LA CALIDAD DE SCRIPT SQL ..... - 221 -
5.3.5. BLOQUEOS DE CONSULTAS SQL ..... - 222 -
5.3.6. CACHÉ DEL SERVIDOR SQL SERVER ..... - 223 -
5.3.7. ESTADÍSTICAS SQL SERVER ..... - 223 -
5.3.7.1. PLANES DE MANTENIMIENTO: ACTUALIZACIÓN DE ESTADÍSTICAS ..... - 226 -
5.3.7.1.1. Actualización de Estadísticas ..... - 227 -
5.3.8. TÉCNICA DE AGREGACIÓN (BO) ..... - 229 -
5.3.8.1. TABLAS AGREGADAS ..... - 229 -
5.3.8.2. CRITERIOS CONSIDERADOS EN LA CREACION DE TABLAS AGREGADAS ..... - 230 -
5.6. COMPROBACIÓN DE LA HIPÓTESIS ..... - 231 -
5.6.1. CONCLUSIÓN ..... - 232 -

CONCLUSIONES

RECOMENDACIONES

RESUMEN

SUMMARY

GLOSARIO DE TÉRMINOS

BIBLIOGRAFÍA

ANEXOS

ÍNDICE DE ABREVIATURAS Y ACRÓNIMOS

BI	Business Intelligence
BD	Base de Datos (Data Base)
BO	Business Intelligence
CMI	Cuadro de Mando Integral (BSC Balanced Scorecard)
CRM	Customer Relationship Management (Software para la administración de relación con los clientes)
DBMS	Database Management System (Gestor de Base de Datos)
DDL	Data Definition Language (Lenguaje de Definición de Datos)
DM	Data Mining
DML	Data Manipulation Language (Lenguaje de Manipulación de Datos)
DSA	Data Stage Area (Área de Datos Intermedia)
DSS	Decision Support System (Sistema de apoyo a las decisiones)
DWA	Data Warehouse Architecture
DWH	Data Warehouse (Almacén de Datos)
EIS	Executive Information Systems (Sistemas de Información para Directivos)
ERP	Enterprise Resource Planning (Sistemas de Información Gerenciales)
ESPOCH	Escuela Superior Politécnica de Chimborazo
ETL	Extract, transform and load (extracción, transformación y carga)

GDSS	Group Decision Support Systems
MOLAP	Multidimensional Online Analytical Processing (Procesamiento Analítico Multidimensional en Línea)
NN	Normal Null
ODS	Operational Data Store
OLAP	On-Line Analytical Processing (Procesamiento Analítico en Línea)
OLTP	On-Line Transacional Processing (Procesamiento Transaccional en Línea)
RID	Row Identified (ID)
SAI	Sun Academic Initiative
SAS	Business Analytics software
SII-ESPOCH	Sistema de Información Institucional – Escuela Superior Politécnica de Chimborazo
SQL	Structured Query Language (Lenguaje Declarativo de Acceso a Bases de Datos)
SSEE	Sistemas expertos basados en inteligencia artificial
TI	Tecnologías de la Información
TPS	Sistema de procesamiento de transacciones (Transaction Processing System)

INDICE TABLAS

TABLA I Comparación entre BD Operacional y Data Warehouse ..... - 25 -

TABLA II Criterios de Análisis ..... - 68 -

TABLA III Niveles de un índice de columnas no incluidas (NoIncludedColumns) .... - 90 -

TABLA IV Niveles de un índice de columnas incluidas (IncludedColumns) ..... - 91 -

TABLA V Propiedades de los Índices ..... - 100 -

TABLA VI Tabla de particiones creada con LEFT ..... - 106 -

TABLA VII Tabla de particiones creada con RIGHT ..... - 107 -

TABLA VIII Orden de Ejecución de clausulas SQL ..... - 114 -

TABLA IX Tabla resumen de criterios y parámetros de análisis ..... - 115 -

TABLA X Características de los servidores ..... - 121 -

TABLA XI Características de la PC de escritorio ..... - 122 -

TABLA XII Características de la laptop HP ..... - 122 -

TABLA XIII Ítems presentes en los planes de ejecución ..... - 129 -

TABLA XIV Criterios de medida ..... - 132 -

TABLA XV Comparación de SQL SERVER 2005 CON MYSQL 5.1 (MYISAM) ..... - 133 -

TABLA XVI Tabla de descripción de características de SQL SERVER 2005 ..... - 135 -

TABLA XVII Tabla de descripción de características de MYSQL 5.1(MYISAM) ..... - 139 -

TABLA XVIII Puntaje de Licenciamiento ..... - 142 -

TABLA XIX Puntaje de Multiplataforma ..... - 142 -

TABLA XX Puntaje de Soporte ..... - 143 -

TABLA XXI Puntaje de ACID ..... - 143 -

TABLA XXII Puntaje de Integridad Referencial ..... - 144 -

TABLA XXIII Puntaje de Consultas ..... - 144 -

TABLA XXIV Puntaje de Tablas Temporales ..... - 145 -

TABLA XXV Puntaje de Vistas ..... - 145 -
TABLA XXVI Puntaje de Cursores ..... - 146 -
TABLA XXVII Puntaje de Triggers ..... - 146 -
TABLA XXVIII Puntaje de Funciones ..... - 147 -
TABLA XXIX Puntaje de Procedimientos ..... - 147 -
TABLA XXX Puntaje de Índices ..... - 148 -
TABLA XXXI Puntaje de Particionamiento de Tablas ..... - 148 -
TABLA XXXII Puntaje de Replicación ..... - 149 -
TABLA XXXIII Puntaje de Seguridad ..... - 149 -
TABLA XXXIV Puntaje de Estabilidad ..... - 150 -
TABLA XXXV Puntaje de Herramientas GUI de administración ..... - 150 -
TABLA XXXVI Puntaje de Optimización de Base de Datos ..... - 151 -
TABLA XXXVII Puntaje de Importación/Exportación ..... - 151 -
TABLA XXXVIII Puntaje de Data Warehousing ..... - 152 -
TABLA XXXIX Puntaje de Servicios para cargas ETLs ..... - 152 -
TABLA XL Puntaje de Minería de Datos ..... - 153 -
TABLA XLI Puntaje de Business Intelligence ..... - 153 -
TABLA XLII Puntaje de Servicios para Procesamiento OLAP ..... - 154 -
TABLA XLIII Tabla de resultados de la comparación de DBMS ..... - 154 -
TABLA XLIV Tabla de resultados de la prueba de la técnica Desnormalización .... - 161 -
TABLA XLV Tabla de resultados de la prueba de la técnica utilización de índices (select) ..... - 173 -
TABLA XLVI Tabla de resultados de la prueba de la técnica utilización de índices (update) ..... - 175 -
TABLA XLVII Tabla de resultados de la prueba de la técnica de particionamiento - 188 -
TABLA XLVIII Tabla de resultados de la prueba de la técnica de calidad de script SQL- 197 -

TABLA XLIX Índices creados en el DWH SII-ESPOCH..... - 215 -

TABLA L: Estadística de la tabla DIM_ESTUDIANTE..... - 225 -

TABLA LI: Resultados de la aplicación de Técnicas Tuning en el DWH SII-ESPOCH .. - 231 -

INDICE FIGURAS

FIGURA II. 1: Arquitectura de un Data Warehouse ..... - 30 -

FIGURA II. 2: Tabla de Hecho ..... - 34 -

FIGURA II. 3: Dimensiones ..... - 36 -

FIGURA II. 4: Data Warehouse ..... - 38 -

FIGURA II. 5: Business Intelligence ..... - 42 -

FIGURA II. 6: Business Intelligence ..... - 44 -

FIGURA II. 7: Manejo de los datos desde un DSS ..... - 46 -

FIGURA II. 8: Informes del DSS ..... - 50 -

FIGURA II. 9: Recursos del Sistema ..... - 54 -

FIGURA II. 10: Rendimiento ..... - 55 -

FIGURA II. 11: Estructura interna de un índice ..... - 58 -

FIGURA II. 12: Búsqueda por clustered índice ..... - 59 -

FIGURA II. 13: Búsqueda por non-clustered índice ..... - 61 -

FIGURA III. 14: Performance en base de datos, en donde hay que focalizar esfuerzo .. - 67 -

FIGURA III. 15: Localizadores de fila ..... - 87 -

FIGURA III. 16: Estructura de un Índice SQL SERVER ..... - 92 -

FIGURA III. 17: Estructura de un Índice Agrupado ..... - 94 -

FIGURA III. 18: Uso de índices agrupados ..... - 97 -

FIGURA III. 19: Uso de índices no agrupados ..... - 98 -

FIGURA III. 20: Operación de Búsqueda (LookUp) de Índices agrupados y no agrupados ..... - 104 -

FIGURA IV. 21: Descripción del escenario utilizado para las pruebas ..... - 123 -

FIGURA IV. 22: Mensaje de advertencia al momento de la ejecución de la consulta ..... - 126 -

FIGURA IV. 23: Mensaje de advertencia, al encontrar falta de memoria durante la ejecución de la consulta ..... - 127 -

FIGURA IV. 24: Mensaje de advertencia por memoria insuficiente ..... - 127 -

FIGURA IV. 25: Descripción de la red del escenario ..... - 130 -

FIGURA IV. 26: Gráfico de resultado de la comparación de DBMS ..... - 156 -

FIGURA IV. 27: Diseño OLTP ..... - 158 -

FIGURA IV. 28: Diseño OLAP ..... - 158 -

FIGURA IV. 29: Gráfico de resultado de tiempo de ejecución ..... - 162 -

FIGURA IV. 30: Gráfico de resultado de la lecturas lógicas ..... - 163 -

FIGURA IV. 31: Gráfico de resultado de la costo de la consulta ..... - 164 -

FIGURA IV. 32: Gráfico de resultado de la utilización de la Técnica de Desnormalización ..... - 165 -

FIGURA IV. 33: Diseño OLAP ..... - 169 -

FIGURA IV. 34: Gráfico de resultado de tiempo de ejecución ..... - 176 -

FIGURA IV. 35: Gráfico de resultado de la lecturas lógicas ..... - 177 -

FIGURA IV. 36: Gráfico de resultado de la costo de la consulta ..... - 178 -

FIGURA IV. 37: Gráfico de resultado de tiempo de ejecución ..... - 179 -

FIGURA IV. 38: Gráfico de resultado de la lecturas lógicas ..... - 180 -

FIGURA IV. 39: Gráfico de resultado de la costo de la consulta ..... - 181 -

FIGURA IV. 40: Gráfico de resultado del uso de la Técnica de Utilización de Índices .... - 182 -

FIGURA IV. 41: Diseño OLAP ..... - 184 -

FIGURA IV. 42: Particionamiento de la tabla ..... - 187 -

FIGURA IV. 43: Gráfico de resultado de tiempo de ejecución ..... - 189 -

FIGURA IV. 44: Gráfico de resultado de la lecturas lógicas ..... - 190 -

FIGURA IV. 45: Gráfico de resultado de la costo de la consulta ..... - 191 -

FIGURA IV. 46: Gráfico de resultado del uso de la Técnica de Particionamiento ... - 192 -

FIGURA IV. 47: Diseño OLAP ..... - 194 -

FIGURA IV. 48: Gráfico de resultado de tiempo de ejecución ..... - 198 -

FIGURA IV. 49: Gráfico de resultado de la lecturas lógicas ..... - 199 -

FIGURA IV. 50: Gráfico de resultado de la costo de la consulta ..... - 200 -

FIGURA IV. 51: Gráfico de resultado del uso de la Técnica de Utilización de Índices .... - 201 -

FIGURA IV. 52: Gráfico de resultado final de optimización del uso de Técnicas Tuning - 202 -

FIGURA V. 53: Diagrama del Data Warehouse SII-ESPOCH ..... - 208 -

FIGURA V. 54: Dimensión Estudiante Desnormalizada ..... - 209 -

FIGURA V. 55: Dimensión Funcionario Desnormalizada ..... - 209 -

FIGURA V. 56: Dimensión Ubicación Geográfica Desnormalizada ..... - 210 -

FIGURA V. 57: Dimensión Institución Desnormalizada ..... - 210 -

FIGURA V. 58: Dimensión Fecha con clave subrogada (ID_FECHA) ..... - 211 -

FIGURA V. 59: Selección de FileGroups ..... - 212 -

FIGURA V. 60: FileGroups (Grupos de Archivos) ..... - 212 -

FIGURA V. 61: Selección de Files ..... - 213 -

FIGURA V. 62: Files (Archivos) ..... - 213 -

FIGURA V. 63: Reorganizar todo desde la interfaz gráfica ..... - 221 -

FIGURA V. 64: Gráfico final de resultado de optimización del uso de Técnicas Tuning en el DWH SII-ESPOCH ..... - 232 -

FIGURA V. 65: Ejemplo de árbol B ..... - 238 -

INTRODUCCIÓN

Debido al desarrollo de la tecnología a nivel estratégico, es importante analizar los Sistemas de toma de Decisiones que prestan información confiable a la hora de tomar una decisión, siendo apoyo sustentable para esta toma de decisiones. Además es importante conocer que estos sistemas trabajan con un Data Warehouse, que almacena la información precisa, muy útil para brindar información debido a que el Data Warehouse trabaja con datos históricos que tienen calidad e integridad de información.

Cuando se obtiene información de los Sistemas de tomas de decisiones es muy importante presentar al usuario final reportes que se generen en tiempos aceptables de respuesta, siendo las Técnicas Tuning que se desarrollan en esta tesis, la parte importante que contribuye a que el usuario obtenga esta información con rapidez, haciendo que los resultados se efectivicen y se logre la optimización del rendimiento, mediante Microsoft SQL Server 2005

Mediante el desarrollo de esta tesis, se busca optimizar el rendimiento en el Data Warehouse SII-ESPOCH, haciendo que mediante técnicas Tuning que se seleccionan y se detallan en las siguientes páginas se logre optimizar el tiempo de respuesta, haciendo que el usuario final obtenga la información con rapidez.

Este trabajo de investigación demuestra las técnicas analizadas y explicadas sobre escenarios OLAP, en los que se logra obtener optimización en la aplicación de cada técnica en el Data Warehouse.

CAPÍTULO I: MARCO REFERENCIAL

1.1. ANTECEDENTES

Mediante el desarrollo tecnológico, los sistemas de información que actualmente brindan mejores funciones y aporte a las empresas, se debe tomar en cuenta a los sistemas de apoyo en la toma de decisiones, que son sistemas que trabajan con un almacén de datos (Data Warehouse), los cuales poseen un factor fundamental, la calidad de información, ya que si se cuenta con datos deficientes, las decisiones que se puedan tomar en base a estos datos no serán las más acertadas.

Al obtener información de un sistema de toma de decisiones, es preciso considerar su eficiencia y rapidez, tomando en cuenta que toda la información entregada debe tener una presentación casi inmediata, es por eso que deben existir técnicas que permitan obtener mejor rendimiento (Tuning) para efectivizar resultados. Al mencionar Tuning se hace referencia a la optimización del rendimiento, que mediante Microsoft SQL Server 2005 se podrá identificar los problemas de rendimiento y de esta manera lograr optimizarlos y alcanzar su máximo desempeño.

Actualmente existen varias técnicas tuning que permiten optimizar el rendimiento, como utilización de índices, carga inicial eficiente, posibilidad de partición de tablas fact de gran tamaño, etc., siendo este trabajo de investigación el que determine las principales técnicas tuning, para evitar el bajo rendimiento el Data Warehouse de la ESPOCH.

1.2. JUSTIFICACIÓN

1.2.1. JUSTIFICACIÓN TEÓRICA

Actualmente las técnicas Tuning aplicadas a Data Warehouse son de gran relevancia ya que se puede comprobar la eficiencia y el rendimiento optimizando los tiempos de respuesta y de esta manera se pueda manejar la presentación de la información una mejor forma, sabiendo que dicha información manejada por el Data Warehouse debe ser integra y de alta calidad, ya que de esta información depende lo que presente en el sistema de toma de decisiones.

1.2.2. JUSTIFICACIÓN METÓDOLÓGICA

Se pretende aportar con el estudio de técnicas Tuning, la calidad y optimización del rendimiento aplicado en un Data Warehouse de un Sistema de toma de decisiones mediante Microsoft SQL Server 2005 y todas las herramientas de este, haciendo que mediante pruebas bajo diferentes parámetros, procedimientos y con el desarrollo de esta investigación se logre obtener técnicas de optimización de rendimiento en un almacén de datos. El desarrollo de la guía de técnicas pretende determinar los procesos y características de estas que logren la optimización de rendimiento en el Data Warehouse.

1.2.3. JUSTIFICACIÓN APLICATIVA

En la actualidad en la Escuela Superior Politécnica de Chimborazo en la Unidad Técnica de Planificación, existe el proyecto *SII-ESPOCH* que consiste en una solución BUSINESS INTELLIGENCE para la toma de decisiones, en el cual se implementará un Data Warehouse en SQL-SERVER y para la explotación de datos se utilizará BUSINESS OBJECTS, con el objetivo de gestionar información de relevancia,

de forma ágil, confiable, precisa, oportuna, que sirva de soporte para la toma de decisiones dentro de la institución y lograr una administración moderna y eficiente en el ámbito académico y administrativo.

En el Data Warehouse se aplicarán las técnicas Tuning, siendo toda esta información la base para las pruebas que demuestren optimización de rendimiento.

Estas técnicas Tuning mejorarán el rendimiento del Data Warehouse, permitiendo su máxima eficiencia.

1.3. OBJETIVOS

1.3.1. OBJETIVO GENERAL

Analizar las técnicas para realizar el Tuning en un Data Warehouse utilizando Microsoft SQL Server, y mejorar el rendimiento de un sistema de toma de decisiones.

1.3.2. OBJETIVOS ESPECÍFICOS

- Realizar el análisis de las técnicas Tuning de un Data Warehouse, comprendiendo el proceso y principales características.
- Determinar los parámetros y pruebas que se realizarán sobre el conjunto de datos del Data Warehouse.
- Comparar las técnicas tuning analizando la optimización de rendimiento del Data Warehouse.
- Desarrollar una guía de las técnicas de Tuning en el Data Warehouse del proyecto ***SII-ESPOCH***.
- Medir el rendimiento a través del sistema de toma de decisiones con BUSINESS OBJECTS.

1.4. HIPÓTESIS

La comparación de Técnicas de Tuning permitirá implementar un esquema de optimización de rendimiento del Data Warehouse del SII-ESPOCH

CAPÍTULO II: MARCO TEÓRICO

2.1. INTRODUCCIÓN

Durante el desarrollo de la optimización de rendimiento, es importante analizar todos los elementos que forman parte de este, empezando desde un Data Warehouse, sus principales características y ventajas, los sistemas de toma de decisiones, técnicas de optimización, prestaciones del DBMS, etc., por tal motivo este capítulo es el conjunto introductorio al desarrollo de la tesis.

2.2. DATA WAREHOUSE

Data warehouse es el centro de la arquitectura para los sistemas de información en la década de los '90. Soporta el procesamiento informático al proveer una plataforma sólida, a partir de los datos históricos para hacer el análisis. Facilita la integración de sistemas de aplicación no integrados. Organiza y almacena los datos que se necesitan para el procesamiento analítico, informático sobre una amplia perspectiva de tiempo.

Un Data Warehouse o Depósito de Datos es una colección de datos orientado a temas, integrado, no volátil, de tiempo variante, que se usa para el soporte del proceso de toma de decisiones gerenciales.

Se puede caracterizar un data warehouse haciendo un contraste de cómo los datos de un negocio almacenados en un data warehouse, difieren de los datos operacionales usados por las aplicaciones de producción.

TABLA

I Comparación entre BD Operacional y Data Warehouse

Base de Datos Operacional	**Data Warehouse**
Datos Operacionales	Datos del negocio para Información
Orientado a la aplicación	Orientado al sujeto
Actual	Actual + histórico
Detallada	Detallada + más resumida
Cambia continuamente	Estable

Fuente: SQLMAXCONNECTIONS, http://www.sqlmax.com/dataw1.asp

Diferentes tipos de información

El ingreso de datos en el data warehouse viene desde el ambiente operacional en casi todos los casos. El data warehouse es siempre un almacén de datos transformados y separados físicamente de la aplicación donde se encontraron los datos en el ambiente operacional. [21]

2.2.1. ¿QUÉ ES UN DATA WAREHOUSE?

Un Data Warehouse es una base de datos corporativa que se caracteriza por integrar y depurar información de una o más fuentes distintas, para luego procesarla permitiendo

su análisis desde infinidad de perspectivas y con grandes velocidades de respuesta. La creación de un Data Warehouse representa en la mayoría de las ocasiones el primer paso, desde el punto de vista técnico, para implantar una solución completa y fiable de Business Intelligence.[19]

Bill Inmon fue uno de los primeros autores en escribir sobre el tema de los Data Warehouse, define un Data Warehouse (almacén de datos) en términos de las características del repositorio de datos. Inmon defiende una metodología descendente (top-down) a la hora de diseñar un Data Warehouse, ya que de esta forma se considerarán mejor todos los datos corporativos. En esta metodología los Data marts se crearán después de haber terminado el Data Warehouse completo de la organización.

Ralph Kimball, es otro conocido autor en el tema de los Data Warehouse, define un almacén de datos como: "una copia de las transacciones de datos específicamente estructurada para la consulta y el análisis". También fue Kimball quien determinó que un data Warehouse no era más que: "la unión de todos los Data marts de una entidad". Defiende por tanto una metodología ascendente (bottom-up) a la hora de diseñar un almacén de datos. [25]

2.2.2. CARACTERÍSTICAS

Según definió el propio Bill Inmon, un data Warehouse se caracteriza por ser:

- **Integrado:** los datos almacenados en el data Warehouse deben integrarse en una estructura consistente, por lo que las inconsistencias existentes entre los diversos sistemas operacionales deben ser eliminadas. La información suele estructurarse también en distintos niveles de detalle para adecuarse a las distintas necesidades de los usuarios.
- **Temático:** sólo los datos necesarios para el proceso de generación del conocimiento del negocio se integran desde el entorno operacional. Los datos se organizan por temas para facilitar su acceso y entendimiento por parte de los usuarios finales. Por ejemplo, todos los datos sobre clientes pueden ser consolidados en una única tabla del Data Warehouse. De esta forma, las

peticiones de información sobre clientes serán más fáciles de responder dado que toda la información reside en el mismo lugar.

- **Histórico:** el tiempo es parte implícita de la información contenida en un Data Warehouse. En los sistemas operacionales, los datos siempre reflejan el estado de la actividad del negocio en el momento presente. Por el contrario, la información almacenada en el Data Warehouse sirve, entre otras cosas, para realizar análisis de tendencias. Por lo tanto, el Data Warehouse se carga con los distintos valores que toma una variable en el tiempo para permitir comparaciones.
- **No volátil:** el almacén de información de un Data Warehouse existe para ser leído, pero no modificado. La información es por tanto permanente, significando la actualización del Data Warehouse la incorporación de los últimos valores que tomaron las distintas variables contenidas en él sin ningún tipo de acción sobre lo que ya existía.

Otra característica del Data Warehouse es que contiene metadatos, es decir, datos sobre los datos. Los metadatos permiten saber la procedencia de la información, su periodicidad de refresco, su fiabilidad, forma de cálculo... etc.

Los metadatos serán los que permiten simplificar y automatizar la obtención de la información desde los sistemas operacionales a los sistemas informacionales.

Los objetivos que deben cumplir los metadatos, según el colectivo al que va dirigido, son:

- **Dar soporte al usuario final**, ayudándole a acceder al Data Warehouse con su propio lenguaje de negocio, indicando qué información hay y qué significado tiene. Ayudar a construir consultas, informes y análisis, mediante herramientas de Business Intelligence como DSS, EIS o CMI.
- **Dar soporte a los responsables técnicos del Data Warehouse en aspectos de auditoría**, gestión de la información histórica, administración del Data Warehouse, elaboración de programas de extracción de la información,

especificación de las interfaces para la realimentación a los sistemas operacionales de los resultados obtenidos... etc.

Por último, destacar que para comprender íntegramente el concepto de Data Warehouse, es importante entender cuál es el proceso de construcción del mismo, denominado ETL (Extracción, Transformación y Carga), a partir de los sistemas operaciones de una compañía:

- **Extracción:** obtención de información de las distintas fuentes tanto internas como externas.
- **Transformación:** filtrado, limpieza, depuración, homogeneización y agrupación de la información.
- **Carga:** organización y actualización de los datos y los metadatos en la base de datos.[19]

2.2.3. OBJETIVOS DEL DATA WAREHOUSE

Los objetivos fundamentales del Data Warehouse son los siguientes:

- Hace que la información de la organización sea accesible: los contenidos del Data Warehouse son entendibles y navegables, y el acceso a ellos son caracterizado por el rápido desempeño. Estos requerimientos no tienen fronteras y tampoco limites fijos. Cuando se habla de entendible significa, que los niveles de la información sean correctos y obvios. Y navegables significa el reconocer el destino en la pantalla y llegar a donde se quiera con sólo un clic. Rápido desempeño significa, cero tiempo de espera. Todo lo demás es un compromiso y por consiguiente algo que queremos mejorar.
- Hacer que la información de la organización sea consistente: la información de una parte de la organización puede hacerse coincidir con la información de la

otra parte de la organización. Si dos medidas de la organización tienen el mismo nombre, entonces deben significar la misma cosa. Y a la inversa, si dos medidas no significan la misma cosa, entonces son etiquetados diferentes. Información consistente significa, información de alta calidad. Significa que toda la información es contabilizada y completada. Todo lo demás es un compromiso y por consiguiente algo que se quiere mejorar.

- Es información adaptable y elástica: el Data Warehouse está diseñado para cambios continuos. Cuando se le hacen nuevas preguntas al Data Warehouse, los datos existentes y las tecnologías no cambian ni se corrompen. Cuando se agregan datos nuevos al Data Warehouse, los datos existentes y las tecnologías tampoco cambian ni se corrompen.

- Es la fundación de la toma de decisiones: el Data Warehouse tiene los datos correctos para soportar la toma de decisiones. Solo hay una salida verdadera del Data Warehouse: las decisiones que son hechas después de que el Data Warehouse haya presentado las evidencias. La original etiqueta que preside el Data Warehouse sigue siendo la mejor descripción de lo que se debe construir: un sistema de soporte a las decisiones.[15]

2.2.4. ARQUITECTURA

Una de las razones por las que el desarrollo de un data warehouse crece rápidamente, es que realmente es una tecnología muy entendible. De hecho, data warehousing puede representar mejor la estructura amplia de una empresa para administrar los datos informacionales dentro de la organización. A fin de comprender cómo se relacionan todos los componentes involucrados en una estrategia data warehousing, es esencial tener una Arquitectura Data Warehouse.[21]

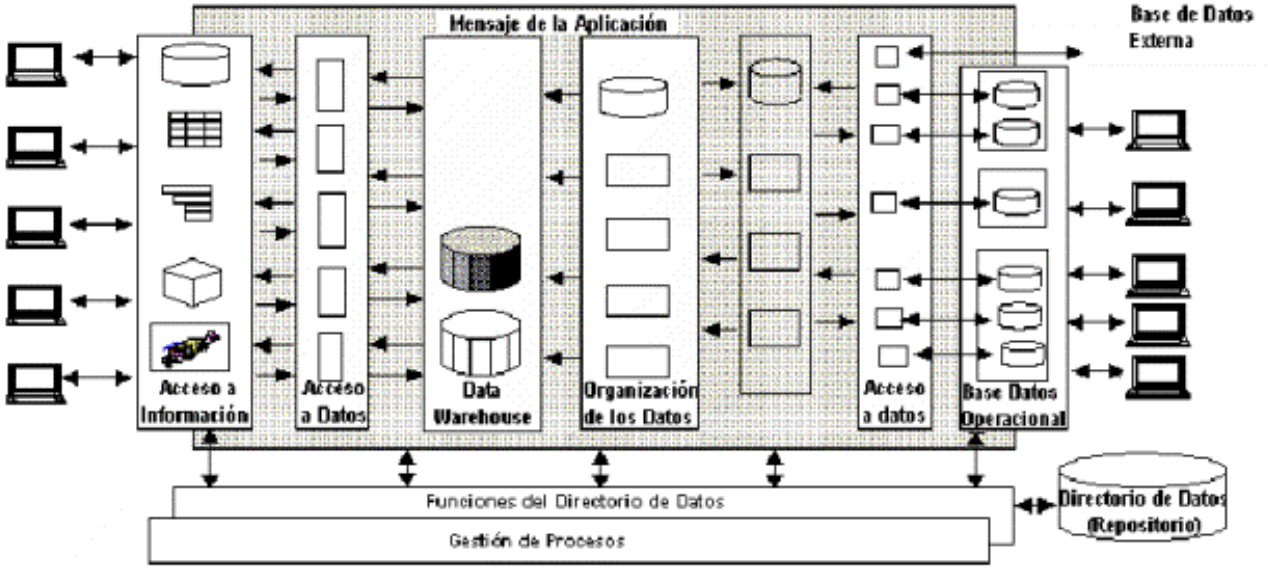


FIGURA II. 1: Arquitectura de un Data Warehouse

2.2.4.1. ELEMENTOS CONSTITUYENTES DE UNA ARQUITECTURA DATA WAREHOUSE

Una Arquitectura Data Warehouse (Data Warehouse Architecture - DWA) es una forma de representar la estructura total de datos, comunicación, procesamiento y presentación, que existe para los usuarios finales que disponen de una computadora dentro de la empresa.

La arquitectura se constituye de un número de partes interconectadas:

- Base de datos operacional / Nivel de base de datos externo
- Nivel de acceso a la información
- Nivel de acceso a los datos

- Nivel de directorio de datos (Meta data)
- Nivel de gestión de proceso
- Nivel de mensaje de la aplicación
- Nivel de data warehouse
- Nivel de organización de datos

Base de datos operacional / Nivel de base de datos externo

Los sistemas operacionales procesan datos para apoyar las necesidades operacionales críticas. Para hacer eso, se han creado las bases de datos operacionales históricas que proveen una estructura de procesamiento eficiente, para un número relativamente pequeño de transacciones comerciales bien definidas.

Ciertamente, la meta del data warehousing es liberar la información que es almacenada en bases de datos operacionales y combinarla con la información desde otra fuente de datos, generalmente externa.

Cada vez más, las organizaciones grandes adquieren datos adicionales desde bases de datos externas.

Nivel de acceso a la información

El nivel de acceso a la información de la arquitectura data warehouse, es el nivel del que el usuario final se encarga directamente. En particular, representa las herramientas que el usuario final normalmente usa día a día. Como por ejemplo: Excel, Lotus 1-2-3, Focus, Access, etc.

Este nivel también incluye el hardware y software involucrados en mostrar información en pantalla y emitir reportes de impresión, hojas de cálculo, gráficos y diagramas para el análisis y presentación.

Nivel de acceso a los datos

El nivel de acceso a los datos de la arquitectura data warehouse está involucrado con el nivel de acceso a la información para conversar en el nivel operacional. En la red mundial de hoy, el lenguaje de datos común que ha surgido es SQL. Originalmente, SQL fue desarrollado por IBM como un lenguaje de consulta, pero en los últimos veinte años ha llegado a ser el estándar para el intercambio de datos.

Uno de los adelantos claves de los últimos años ha sido el desarrollo de una serie de "filtros" de acceso a datos, tales como EDA/SQL para acceder a casi todo los Sistemas de Gestión de Base de Datos (Data Base Management Systems - DBMSs) y sistemas de archivos de datos, relacionales o no. Estos filtros permiten a las herramientas de acceso a la información, acceder también a la data almacenada en sistemas de gestión de base de datos que tienen años de antigüedad.

El nivel de acceso a los datos no solamente conecta DBMSs diferentes y sistemas de archivos sobre el mismo hardware, sino también a los fabricantes y protocolos de red. Una de las claves de una estrategia data warehousing es proveer a los usuarios finales con "acceso a datos universales".

El acceso a los datos universales significa que, teóricamente por lo menos, los usuarios finales sin tener en cuenta la herramienta de acceso a la información o ubicación, deberían ser capaces de acceder a cualquier o todos los datos en la empresa que es necesaria para ellos, para hacer su trabajo.

El nivel de acceso a los datos entonces es responsable de las interfaces entre las herramientas de acceso a la información y las bases de datos operacionales.

Nivel de Directorio de Datos (Meta data)

A fin de proveer el acceso a los datos universales, es absolutamente necesario mantener alguna forma de directorio de datos o repositorio de la información meta data. La meta data es la información alrededor de los datos dentro de la empresa.

Idealmente, los usuarios finales deberían de acceder a los datos desde el data warehouse (o desde las bases de datos operacionales), sin tener que conocer dónde residen los datos o la forma en que se han almacenado.

Nivel de Gestión de Procesos

El nivel de gestión de procesos tiene que ver con la programación de diversas tareas que deben realizarse para construir y mantener el data warehouse y la información del directorio de datos. Este nivel puede depender del alto nivel de control de trabajo para muchos procesos (procedimientos) que deben ocurrir para mantener el data warehouse actualizado.

Nivel de Mensaje de la Aplicación

El nivel de mensaje de la aplicación tiene que ver con el transporte de información alrededor de la red de la empresa. El mensaje de aplicación se refiere también como "subproducto", pero puede involucrar sólo protocolos de red. Puede usarse por ejemplo, para aislar aplicaciones operacionales o estratégicas a partir del formato de datos exacto, recolectar transacciones o los mensajes y entregarlos a una ubicación segura en un tiempo seguro.

Nivel Data Warehouse (Físico)

En el data warehouse (núcleo) es donde ocurre la data actual, usada principalmente para usos estratégicos. En algunos casos, se puede pensar del data warehouse simplemente como una vista lógica o virtual de datos. En muchos ejemplos, el data warehouse puede no involucrar almacenamiento de datos.

En un data warehouse físico, copias, en algunos casos, muchas copias de datos operacionales y/o externos, son almacenados realmente en una forma que es fácil de acceder y es altamente flexible. Cada vez más, los data warehouse son almacenados sobre plataformas cliente/servidor, pero por lo general se almacenan sobre mainframes.

Nivel de Organización de Datos

El componente final de la arquitectura data warehouse es la organización de los datos. Se llama también gestión de copia o réplica, pero de hecho, incluye todos los procesos

necesarios como seleccionar, editar, resumir, combinar y cargar datos en el depósito y acceder a la información desde bases de datos operacionales y/o externas.

La organización de datos involucra con frecuencia una programación compleja, pero cada vez más, están creándose las herramientas data warehousing para ayudar en este proceso. Involucra también programas de análisis de calidad de datos y filtros que identifican modelos y estructura de datos dentro de la data operacional existente.[21]

2.2.5. TABLA DE HECHOS (FACT)

Se denomina “hechos” a los indicadores de negocio. Por ejemplo, son “hechos” las ventas, los pedidos, los envíos, las reclamaciones, las compras, etc. Es decir, son todas aquellas medidas numéricas que se incluirán en el sistema Business Intelligence.

Técnicamente, una tabla de hecho es la tabla central de un modelo en estrella. En el siguiente diagrama, la tabla de ventas es la tabla de hechos:

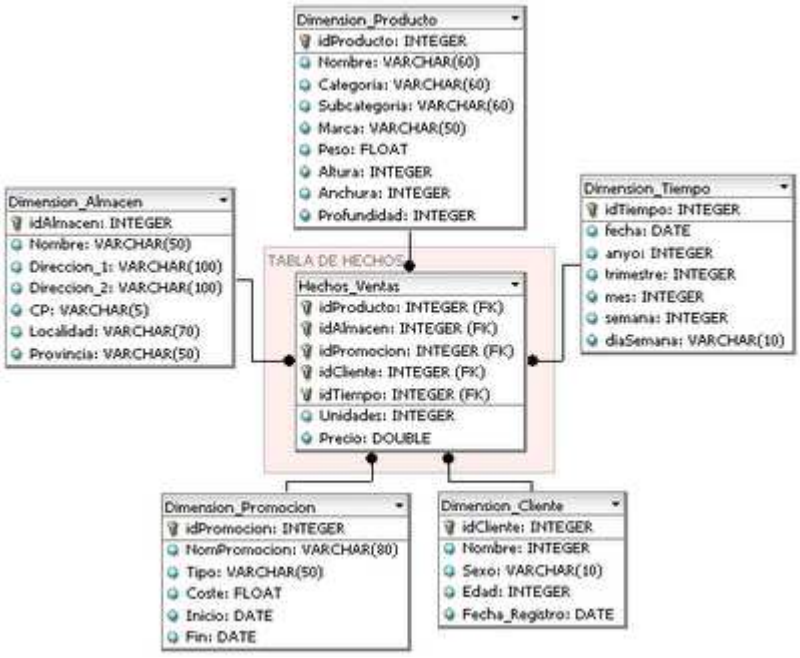


FIGURA II. 2: Tabla de Hecho

Una característica importante de las tablas de hecho es el “nivel de detalle” de la información que se almacena.

La tabla de hechos contiene las claves subrogadas de aquellas dimensiones que definen su nivel de detalle, y los indicadores. Nada más.

Por lo tanto, antes de crear la tabla de hechos debe entenderse perfectamente la información que se guardará, o se estaría cometiendo el error “Añadir dimensiones en una tabla de hechos antes de definir su granularidad.”

De hecho, la creación de una tabla de hechos es una tarea con poco margen a la imaginación. Antes que nada, debe localizarse el origen de la información que se quiere cargar, debe entenderse perfectamente el significado de estos indicadores, y debe determinarse el nivel de detalle de estos datos. Una vez hecho esto, la creación de la estructura de la tabla es inmediata. Tomando en cuenta que la tabla de hechos contiene las claves subrogadas de aquellas dimensiones que definen su nivel de detalle, y los indicadores, solamente eso.

En particular, es un error desnormalizar cualquier dimensión en la tabla de hechos. Por ejemplo, si la información está a nivel de cliente, no necesitamos poner la población o el país en la tabla de hechos. Resultaría redundante e impactaría directamente en el tamaño de la tabla (y en los tiempos de respuesta).

Hay que tomar en cuenta que las tablas de hecho contienen los indicadores numéricos provenientes de los orígenes transaccionales.[4]

2.2.6. DIMENSIONES

Se denominan dimensiones a aquellos datos que permiten filtrar, agrupar o seccionar la información. El término "dimensión" sigue teniendo una cierta connotación técnica, por lo que muchas personas lo denominan "atributo", "característica", "propiedad", "campo".

Algunas aplicaciones Business Intelligence utilizan el término "dimensión" como equivalente a "jerarquía" (especialmente en bases de datos multidimensionales). De

esta manera, se puede hablar de la dimensión geográfica que agrupa los diferentes niveles de continentes, países, regiones, provincias y localidades.

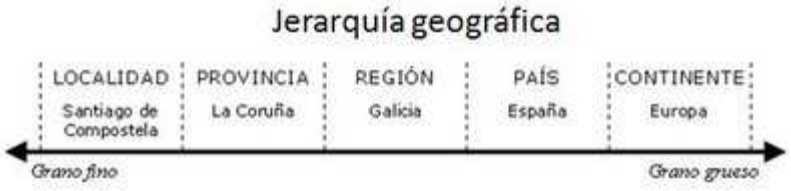


FIGURA II. 3: Dimensiones

En el modelo relacional del data warehouse las dimensiones se almacenan en las "tablas de dimensión". El espacio requerido por las tablas de dimensión es despreciable frente a lo que ocupan los hechos.

Por lo tanto, no se debe considerar el espacio como un aspecto determinante para modelizar las dimensiones. En particular, cada código debe tener su descripción. Las dimensiones son la interfaz que tendrán los usuarios para navegar por la información, por lo que conviene que sean lo más explícitas y claras posible.

Incluso se debe plantear la necesidad real de introducir los códigos en la capa de presentación a los usuarios y siempre son preferibles las descripciones también.

Es importante y se debe tomar en cuenta que las tablas de dimensión contienen los atributos (normalmente textuales) que permiten filtrar y agrupar los indicadores.[4]

2.2.7. CONSTRUCCIÓN DEL DATA WAREHOUSE

Para construir un Data Warehouse se necesitan herramientas para ayudar a la migración y a la transformación de los datos hacia el almacén. Una vez construido, se requieren medios para manejar grandes volúmenes de información. Se diseña su arquitectura dependiendo de la estructura interna de los datos del almacén y

especialmente del tipo de consultas a realizar. Para abordar un proyecto de data warehouse es necesario hacer un estudio de algunos temas generales de la organización o empresa, los cuales se describen a continuación:

- Situación actual de partida.- Cualquier solución propuesta de data warehouse debe estar muy orientada por las necesidades del negocio y debe ser compatible con la arquitectura técnica existente y planeada de la compañía.
- Tipo y características del negocio.- Es indispensable tener el conocimiento exacto sobre el tipo de negocios de la organización y el soporte que representa la información dentro de todo su proceso de toma de decisiones.
- Entorno técnico.- Se debe incluir tanto el aspecto del hardware (mainframes, servidores, redes,...) así como aplicaciones y herramientas. Se dará énfasis a los Sistemas de soporte a decisiones (DSS), si existen en la actualidad, cómo operan, etc.
- Expectativas de los usuarios.- Un proyecto de data warehouse no es únicamente un proyecto tecnológico, es una forma de vida de las organizaciones y como tal, tiene que contar con el apoyo de todos los usuarios y su convencimiento sobre su bondad.
- Etapas de desarrollo.- Con el conocimiento previo, ya se entra en el desarrollo de un modelo conceptual para la construcción del data warehouse.
- Prototipo.- Un prototipo es un esfuerzo designado a simular tanto como sea posible el producto final que será entregado a los usuarios.
- Piloto.- El piloto de un data warehouse es el primero, o cada uno de los primeros resultados generados de forma iterativa que se harán para llegar a la construcción del producto final deseado.
- Prueba del concepto tecnológico.- Es un paso opcional que se puede necesitar para determinar si la arquitectura especificada del data warehouse funcionará finalmente como se espera.

Una de las claves del éxito en la construcción de un Data Warehouse es el desarrollo de forma gradual, seleccionando a un departamento usuario como piloto y expandiendo progresivamente el almacén de datos a los demás usuarios. Por ello es importante elegir este usuario inicial o piloto, siendo importante que sea un departamento con pocos usuarios, en el que la necesidad de este tipo de sistemas es muy alta y se pueda obtener y medir resultados a corto plazo.[19]

2.2.8. VENTAJAS

Principales aportaciones de un Data Warehouse:

- La ventaja principal de este tipo de bases de datos radica en las estructuras en las que se almacena la información (modelos de tablas en estrella, en copo de nieve, cubos relacionales... etc). Este tipo de persistencia de la información es homogénea y fiable, y permite la consulta y el tratamiento jerarquizado de la misma (siempre en un entorno diferente a los sistemas operacionales).

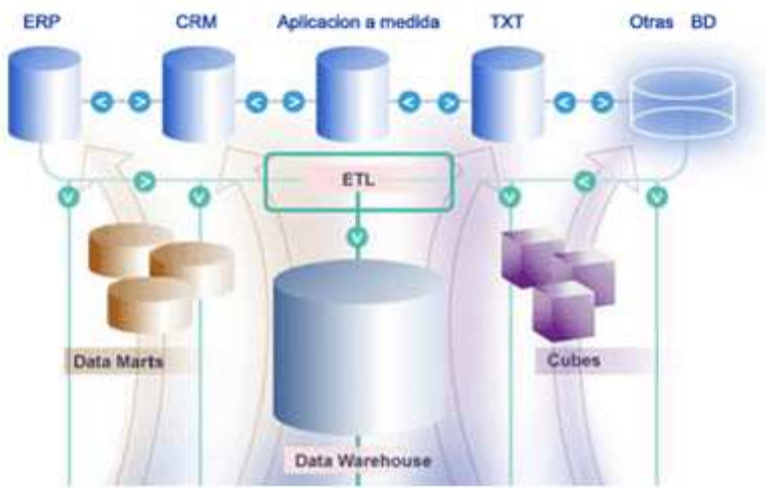


FIGURA II. 4: Data Warehouse

- Proporciona una herramienta para la toma de decisiones en cualquier área funcional, basándose en información integrada y global del negocio.
- Facilita la aplicación de técnicas estadísticas de análisis y modelización para encontrar relaciones ocultas entre los datos del almacén; obteniendo un valor añadido para el negocio de dicha información.
- Proporciona la capacidad de aprender de los datos del pasado y de predecir situaciones futuras en diversos escenarios.
- Simplifica dentro de la empresa la implantación de sistemas de gestión integral de la relación con el cliente.
- Supone una optimización tecnológica y económica en entornos de Centro de Información, estadística o de generación de informes con retornos de la inversión espectaculares. [19]

2.2.9. DESVENTAJAS

Utilizar almacenes de datos también plantea algunos inconvenientes, algunos de ellos son:

- A lo largo de su vida los almacenes de datos pueden suponer altos costos. El almacén de datos no suele ser estático. Los costos de mantenimiento son elevados.
- Los almacenes de datos se pueden quedar obsoletos relativamente pronto.
- A veces, ante una petición de información estos devuelven una información subóptima, que también supone una pérdida para la organización.
- A menudo existe una delgada línea entre los Data Warehouse y los sistemas operacionales. Hay que determinar qué funcionalidades de estos se pueden aprovechar y cuáles se deben implementar en el Data Warehouse, resultaría costoso implementar operaciones no necesarias o dejar de implementar alguna que sí vaya a necesitarse.[25]

2.3. DATA MART

Un Data mart es una versión especial del data warehouse (almacén de datos). Son subconjuntos de datos con el propósito de ayudar a que un área específica dentro del negocio pueda tomar mejores decisiones. Los datos existentes en este contexto pueden ser agrupados, explorados y propagados de múltiples formas para que diversos grupos de usuarios realicen la explotación de los mismos de la forma más conveniente según sus necesidades.

El Data Mart es un sistema orientado a la consulta, en el que se producen procesos batch de carga de datos (altas) con una frecuencia baja y conocida. Es consultado mediante herramientas OLAP (On line Analytical Processing - Procesamiento Analítico en Línea) que ofrecen una visión multidimensional de la información. Sobre estas bases de datos se pueden construir EIS (Executive Information Systems, Sistemas de Información para Directivos) y DSS (Decision Support Systems, Sistemas de Ayuda a la toma de Decisiones).

En síntesis, se puede decir que los data marts son pequeños data warehouse centrados en un tema o un área de negocio específico dentro de una organización.

Los Data marts son subconjuntos de datos de un data warehouse para áreas específicas.[25]

Entre las características de un data mart destacan:

- Usuarios limitados.
- Área específica.
- Tiene un propósito específico.

2.4. BASES DE DATOS OLTP Y OLAP

2.4.1. OLTP - On-Line Transactional Processing

Los sistemas OLTP son bases de datos orientadas al procesamiento de transacciones. Una transacción genera un proceso atómico (que debe ser validado con un commit, o invalidado con un rollback), y que puede involucrar operaciones de inserción, modificación y borrado de datos. El proceso transaccional es típico de las bases de datos operacionales.

- El acceso a los datos está optimizado para tareas frecuentes de lectura y escritura. (Por ejemplo, la enorme cantidad de transacciones que tienen que soportar las BD de bancos o hipermercados diariamente).
- Los datos se estructuran según el nivel aplicación (programa de gestión a medida, ERP o CRM implantado, sistema de información departamental).
- Los formatos de los datos no son necesariamente uniformes en los diferentes departamentos (es común la falta de compatibilidad y la existencia de islas de datos).
- El historial de datos suele limitarse a los datos actuales o recientes.

2.4.2. OLAP - On-Line Analytical Processing

Los sistemas OLAP son bases de datos orientadas al procesamiento analítico. Este análisis suele implicar, generalmente, la lectura de grandes cantidades de datos para llegar a extraer algún tipo de información útil: tendencias de ventas, patrones de comportamiento de los consumidores, elaboración de informes complejos, etc.

- El acceso a los datos suele ser de sólo lectura. La acción más común es la consulta, con muy pocas inserciones, actualizaciones o eliminaciones.

- Los datos se estructuran según las áreas de negocio, y los formatos de los datos están integrados de manera uniforme en toda la organización.
- El historial de datos es a largo plazo, normalmente de dos a cinco años.
- Las bases de datos OLAP se suelen alimentar de información procedente de los sistemas operacionales existentes, mediante un proceso de extracción, transformación y carga (ETL). [19]

2.5. BUSINESS INTELLIGENCE

2.5.1. ¿QUÉ ES BUSINESS INTELLIGENCE?

Business Intelligence es la habilidad para transformar los datos en información, y la información en conocimiento, de forma que se pueda optimizar el proceso de toma de decisiones en los negocios.

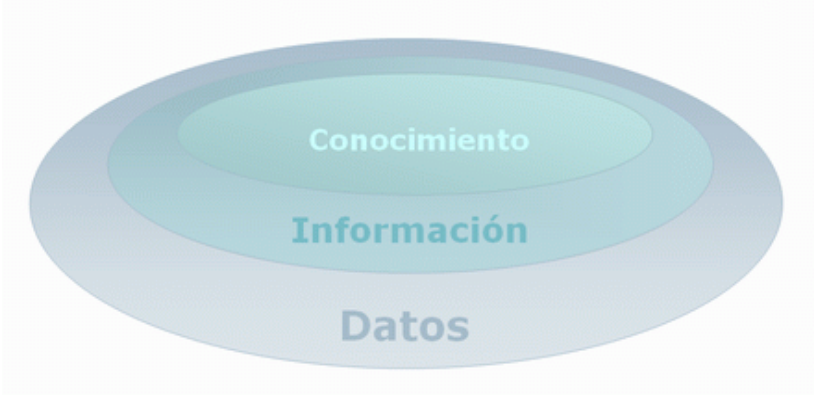


FIGURA II. 5: Business Intelligence

Desde un punto de vista más pragmático, y asociándolo directamente con las tecnologías de la información, podemos definir Business Intelligence como el conjunto de metodologías, aplicaciones y tecnologías que permiten reunir, depurar y transformar

datos de los sistemas transaccionales e información desestructurada (interna y externa a la compañía) en información estructurada, para su explotación directa (reporting, análisis OLTP / OLAP, alertas.) o para su análisis y conversión en conocimiento, dando así soporte a la toma de decisiones sobre el negocio.

La inteligencia de negocio actúa como un factor estratégico para una empresa u organización, generando una potencial ventaja competitiva, que no es otra que proporcionar información privilegiada para responder a los problemas de negocio: entrada a nuevos mercados, promociones u ofertas de productos, eliminación de islas de información, control financiero, optimización de costes, planificación de la producción, análisis de perfiles de clientes, rentabilidad de un producto concreto, etc...

Los principales productos de Business Intelligence que existen hoy en día son:

- Cuadros de Mando Integrales (CMI)
- Sistemas de Soporte a la Decisión (DSS)
- Sistemas de Información Ejecutiva (EIS)

Por otro lado, los principales componentes de orígenes de datos en el Business Intelligence que existen en la actualidad son:

- Datamart
- Datawarehouse

Los sistemas y componentes del BI se diferencian de los sistemas operacionales en que están optimizados para preguntar y divulgar sobre datos. Esto significa típicamente que, en un data warehouse, los datos están desnormalizados para apoyar consultas de alto rendimiento, mientras que en los sistemas operacionales suelen encontrarse normalizados para apoyar operaciones continuas de inserción, modificación y borrado de datos. En este sentido, los procesos ETL (extracción,

transformación y carga), que nutren los sistemas BI, tienen que traducir de uno o varios sistemas operacionales normalizados e independientes a un único sistema desnormalizado, cuyos datos estén completamente integrados.

En definitiva, una solución BI completa permite:

- Observar ¿qué está ocurriendo?
- Comprender ¿por qué ocurre?
- Predecir ¿qué ocurriría?
- Colaborar ¿qué debería hacer el equipo?
- Decidir ¿qué camino se debe seguir? [4]

FIGURA II. 6: Business Intelligence

2.5.2. BI COMO SOLUCIÓN TECNOLÓGICA

BI como solución tecnológica puede:

- **Centralizar, depurar y afianzar los datos.** Las tecnologías de BI permiten reunir, normalizar y centralizar toda la información de la empresa, mediante un almacén de datos, permitiendo así su explotación sin esfuerzo.

- **Descubrir información no evidente para las aplicaciones actuales.** En el día a día de las aplicaciones de gestión se pueden esconder pautas de comportamiento, tendencias, evoluciones del mercado, etc., que resulta prácticamente imposible reconocer sin el software adecuado. Es lo que se puede calificar como extraer información de los datos, y conocimiento de la información.
- **Optimizar el rendimiento de los sistemas.** Las plataformas de BI se diseñan para perfeccionar al máximo las consultas de alto nivel, realizando las transformaciones oportunas a cada sistema (OLTP - OLAP), y liberando los servidores operacionales.[20]

2.6. SISTEMAS DE TOMA DE DECISIONES

2.6.1. DEFINICIÓN DE SISTEMAS DE SOPORTE DE TOMA DE DECISIONES

Un Sistema de Soporte a la Decisión (DSS) es una herramienta de Business Intelligence enfocada al análisis de los datos de una organización.

En principio, puede parecer que el análisis de datos es un proceso sencillo, y fácil de conseguir mediante una aplicación hecha a medida o un ERP sofisticado. Sin embargo, no es así: estas aplicaciones suelen disponer de una serie de informes predefinidos en los que presentan la información de manera estática, pero no permiten profundizar en los datos, navegar entre ellos, manejarlos desde distintas perspectivas.

FIGURA II. 7: Manejo de los datos desde un DSS

El DSS es una de las herramientas más emblemáticas del Business Intelligence ya que, entre otras propiedades, permiten resolver gran parte de las limitaciones de los programas de gestión. [20]

2.6.2. CARACTERÍSTICAS

Estas son algunas de sus características principales del DSS:

- **Informes dinámicos, flexibles e interactivos**, de manera que el usuario no tenga que ceñirse a los listados predefinidos que se configuraron en el momento de la implantación, y que no siempre responden a sus dudas reales.
- **No requiere conocimientos técnicos**. Un usuario no técnico puede crear nuevos gráficos e informes y navegar entre ellos, haciendo drag&drop o drill through. Por tanto, para examinar la información disponible o crear nuevas métricas no es imprescindible buscar auxilio en el departamento de informática.
- **Rapidez en el tiempo de respuesta**, ya que la base de datos subyacente suele ser un data warehouse corporativo o un data mart, con modelos de datos en estrella o copo de nieve. Este tipo de bases de datos están optimizadas para el análisis de grandes volúmenes de información.

- **Integración entre todos los sistemas/departamentos de la compañía**. El proceso de ETL previo a la implantación de un Sistema de Soporte a la Decisión garantiza la calidad y la integración de los datos entre las diferentes unidades de la empresa. Existe lo que se llama: integridad referencial absoluta.
- **Cada usuario dispone de información adecuada a su perfil**. No se trata de que todo el mundo tenga acceso a toda la información, sino de que tenga acceso a la información que necesita para que su trabajo sea lo más eficiente posible.
- **Disponibilidad de información histórica**. En estos sistemas está a la orden del día comparar los datos actuales con información de otros períodos históricos de la compañía, con el fin de analizar tendencias, fijar la evolución de parámetros de negocio.[20]

2.6.3. CLASIFICACIÓN

La clasificación más habitual es la que hace referencia al **alcance de la decisión**. En este caso, se distinguen tres tipos:

- **Decisiones estratégicas**: Son aquellas que afectan a toda la empresa (o a una buena parte de la misma) durante un largo periodo de tiempo. Influyen, por lo tanto, en los objetivos generales de la empresa y en su modelo de negocio. Estas decisiones son tomadas por los máximos responsables de las compañías (CEO, presidentes, directores generales, comités de dirección, etc.).
- **Decisiones tácticas**: Afectan únicamente a parte de la empresa, o a parte de sus procesos, y generalmente se toman desde un sólo departamento (o de unos pocos). Tienen un impacto relevante a medio plazo (1 o 2 años, como máximo), y son tomadas por cargos intermedios (jefes de departamento, gerentes, etc.)
- **Decisiones operativas**: Afectan a actividades específicas, con un alcance muy claro, y su efecto es inmediato o muy limitado en el tiempo. Estas decisiones

son responsabilidad de los niveles bajos de la jerarquía empresarial (jefes de equipo, encargados de área, dependientes, etc.)

Merece la pena señalar que esta clasificación no dice nada sobre la importancia de las decisiones. Todas ellas son importantes y necesarias. Una mala decisión operativa puede costar millones (del mismo que una buena decisión puede suponer suculentos beneficios). También existen decisiones "estratégicas" que resultan ser irrelevantes desde el punto de vista económico.

2.6.4. TIPOS

Los tipos de Sistemas de Soporte a Decisiones son:

- **Sistemas de información gerencial (MIS)**

  Los sistemas de información gerencial (MIS, Management Information Systems), también llamados Sistemas de Información Administrativa (AIS) dan soporte a un espectro más amplio de tareas organizacionales, encontrándose a medio camino entre un DSS tradicional y una aplicación CRM/ERP implantada en la misma compañía.

- **Sistemas de información ejecutiva (EIS)**

  Los sistemas de información ejecutiva (EIS, Executive Information System) son el tipo de DSS que más se suele emplear en Business Intelligence, ya que proveen a los gerentes de un acceso sencillo a información interna y externa de su compañía, y que es relevante para sus factores clave de éxito.

- **Sistemas expertos basados en inteligencia artificial (SSEE)**

  Los sistemas expertos, también llamados sistemas basados en conocimiento, utilizan redes neuronales para simular el conocimiento de un experto y utilizarlo de forma efectiva para resolver un problema concreto. Este concepto está muy relacionado con el data mining o minería de datos.

- **Sistemas de apoyo a decisiones de grupo (GDSS)**

  Un sistema de apoyo a decisiones en grupos (GDSS, Group Decision Support Systems) es "un sistema basado en computadoras que apoya a grupos de personas que tienen una tarea (u objetivo) común, y que sirve como interfaz con un entorno compartido". El supuesto en que se basa el GDSS es que si se mejoran las comunicaciones se pueden mejorar las decisiones.

Como se sabe, en el día a día de las empresas se toman continuamente decisiones de muy distinto tipo, y de manera muy diferente. Existen decisiones que por su naturaleza se toman de manera racional y absolutamente informada, y otras que se toman de manera menos sistemática, casi por instinto.

Existen algunos DSS, los actuales DSS utilizan metodologías OLAP, y ofrecen un soporte pasivo a la toma de decisiones. Es decir, los sistemas DSS actuales ayudan a la toma de decisiones proporcionando información confiable y actualizada, pero raramente aportan valor añadido a la información y decisión resultante. Un soporte activo a la toma de decisiones requiere modelos matemáticos y estadísticos avanzados que descubran patrones ocultos en la información (para diseñar mejores campañas de marketing, para optimizar la cadena de suministro, para orientar mejor los productos a mercados específicos, etc.), y todo eso se está haciendo todavía muy poco, y de manera poco estructurada.[20]

2.6.5. DIFERENCIA CON OTRAS HERRAMIENTAS DE BUSINESS INTELLIGENCE

El principal objetivo de los Sistemas de Soporte a Decisiones es, a diferencia de otras herramientas como los Cuadros de Mando (CMI) o los Sistemas de Información Ejecutiva (EIS), explotar al máximo la información residente en una base de datos corporativa (data warehouse o data mart), mostrando informes muy dinámicos y con gran potencial de navegación, pero siempre con una interfaz gráfica amigable, vistosa y sencilla.

FIGURA II. 8: Informes del DSS

Otra diferencia fundamental radica en los usuarios a los que están destinadas las plataformas DSS: cualquier nivel gerencial dentro de una organización, tanto para situaciones estructuradas como no estructuradas. (En este sentido, por ejemplo, los CMI están más orientados a la alta dirección).

Por último, destacar que los DSS suelen requerir (aunque no es imprescindible) un motor OLAP subyacente, que facilite el análisis casi ilimitado de los datos para hallar las causas raíces de los problemas/pormenores de la compañía. [20]

2.7. BUSINESS OBJECTS (BO)

Las soluciones de Business Intelligence (BI) de SAP BusinessObjects proporcionan funciones completas de Business Intelligence que permiten a sus usuarios tomar decisiones efectivas y fundamentadas basadas en datos y análisis sólidos. Todos los usuarios, desde los analistas de nivel superior hasta los usuarios ocasionales de la empresa, tienen acceso a la información que necesitan con una dependencia mínima de los desarrolladores y los recursos de TI. [5]

Gracias a estas potentes soluciones, los usuarios de toda la empresa podrán acceder a la información, darle formato, analizarla, navegar por ella y compartirla a través de toda la corporación. Las soluciones de BI de SAP Business Objects permiten a los usuarios:

- Análisis avanzados
- Cuadros de mando y visualización
- Infraestructura de la información
- Consultas, informes y análisis
- Gestión de informes
- Búsqueda y exploración de datos

2.8. TUNING

Tuning de base de datos es la adecuación de procesos que se mantienen en memoria para dar acceso a esa base de datos.

Una base de datos, para estar servida en red requiere que se corra un proceso de servicio y algunos procesos adicionales, estos procesos se encargan de dar respuesta a las peticiones de los usuarios que se conectan a un data warehouse, de escribir y leer registros resolviendo interbloqueos, multilecturas y fallos en la conexión.

Estos procesos consumen recursos de CPU y de memoria, dependiendo de la base de datos, su volumen, cantidad de usuarios y cantidad de operaciones. El tuning implica configurar los procesos de base de datos para que sólo consuman los recursos necesarios sin que se vea afectada la velocidad de respuesta al hacer operaciones en la base de datos.

2.8.1. TUNING DEL DATA WAREHOUSE

El tuning de base de datos es configurar los elementos del servicio de base de datos para que su uso sea óptimo en velocidad y en uso de memoria. Una base de datos sin tuning puede ser lenta o consumir la memoria del equipo innecesariamente.

Los procesos de tuning dependen de la base de datos que se esté usando, puede ser Oracle, MySQL, Postgre, Sql Server, etc. Cada una de ellas proporciona herramientas específicas para ver los parámetros que está usando una base de datos y modificarlos si es preciso.

Se puede mejorar el rendimiento de una base de datos a través de la desnormalización y la utilización adecuada de índices.

El proceso de normalizar una base de datos es lo que garantiza una serie de propiedades tales como la no redundancia de datos y la simplificación y optimización del rendimiento del motor para operaciones transaccionales (agregar, eliminar, modificar y buscar una fila utilizando su clave primaria).

Pero en algunos casos se puede ver que muchas veces el rendimiento no se lleva de la mano con la normalización de las bases de datos para operaciones de consultas. Un ejemplo típico es el de querer obtener un total y este depende de la suma de una o varias columnas definidas en una serie de filas. Si resulta ser que esta operación se ejecuta lo suficientemente seguido se obtendría un mejor rendimiento si ya se tuviese ese total precalculado. Pero muchas veces las opciones de que desnormalizar no son tan obvias como la antedicha y generalmente son las que mayores beneficios brindan.

Hay casos en donde la desnormalización sumada a la utilización inteligente de los índices da una mejora del rendimiento en varios órdenes de magnitud. [28]

2.9. RENDIMIENTO

El rendimiento significa que el Data Warehouse no cause tiempos de respuesta poco razonables.

Tiempo de respuesta frente a rendimiento

El tiempo de respuesta mide la cantidad de tiempo necesario para que se devuelva el primer conjunto de resultados. Normalmente se hace referencia al tiempo de respuesta como el tiempo que tarda el usuario en recibir una afirmación visual de que una consulta está en proceso.

El rendimiento mide el número total de consultas que el servidor puede controlar durante un espacio de tiempo dado.

2.9.1. CÓMO SE MIDE EL RENDIMIENTO

El rendimiento se puede medir mediante la cantidad de entrada y salida necesaria para procesar una transacción, la cantidad de tiempo de CPU y el tiempo de respuesta.

2.9.2. ANÁLISIS DE RENDIMIENTO

El análisis de rendimiento consiste en determinar la duración las instrucciones realizadas en el data warehouse en base al tipo de operaciones que realiza y los objetos que este contiene.

Por ejemplo, una tabla cuando tarda en arrojar datos, cuánto dura en arrojar datos ordenados sin índices, cuánto dura con índices, si una consulta de varias tablas dura menos con join que con producto cruz, etc.

Sirve para ver si la estructura de datos puede ser optimizada (si es muy lenta) o para aplicar mejoras. [28]

2.9.3. FACTORES QUE TIENEN IMPACTO EN EL RENDIMIENTO

Recursos del sistema

FIGURA II. 9: Recursos del Sistema

Memoria

La existencia de una cantidad suficiente de RAM resulta vital para el rendimiento de SQL Server.

Procesador

El número de procesadores, así como su velocidad, tiene un impacto directo sobre el rendimiento general.

Disco

El número, velocidad y tipo de las unidades de disco, así como los tipos de controladores utilizados, afectan al rendimiento.

Red

La actividad de red simultánea puede tener impacto en el rendimiento de SQL Server. También tienen importancia el ancho de banda de la red y la velocidad de transferencia de los datos.

2.9.4. OBJETIVOS DE RENDIMIENTO

El objetivo del ajuste de rendimiento consiste en proporcionar un tiempo de respuesta aceptable en las consultas al, reducir la entrada y salida de disco y minimizar el tiempo de CPU para permitir el máximo rendimiento de los procesos de los usuarios. Este objetivo se logra mediante el análisis de los requisitos de las aplicaciones, la comprensión de la estructura física y lógica de los datos, y la capacidad de evaluar y negociar las contrapartidas entre usos conflictivos del data warehouse, frente a la ayuda a la toma de decisiones.

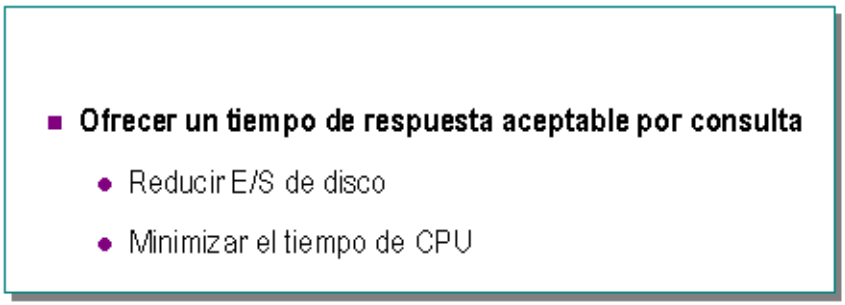


FIGURA II. 10: Rendimiento

Sistemas de ayuda a la toma de decisiones

Normalmente, la ayuda a la toma de decisiones implica varios argumentos de búsqueda y varias tablas. Las consultas pueden ser bastante complejas, ya que pueden utilizar funciones de agregado y agrupamiento. Algunas veces, también se hace referencia a las mismas como procesamiento analítico en línea (OLAP).

Las consultas pueden ser arbitrarias e impredecibles y pueden utilizar casi cualquier columna para especificar las filas que se desean.

La velocidad de recuperación y la devolución de los resultados son los aspectos más críticos de estos tipos de consultas. [3]

2.10.SQL SERVER 2005

2.10.1. RECURSOS DE SQL SERVER

Es importante conocer los recursos que tiene SQL SERVER con los que trabaja como administrador de base de datos.

MEMORIA

La memoria de SQL Server está dividida en dos espacios, el cache de procedimientos, y el cache de datos. El servidor SQL distribuye eficazmente la memoria entre el cache de procedimiento y el cache de datos usando los parámetros de configuración del cache de procedimiento. La distribución de la memoria restante entre procedimientos y datos, se encarga de mantener los objetos más usados en sus respectivos caches de memoria. Por consiguiente, los procedimientos almacenados mayormente usados deben estar en el cache de procedimientos, mientras que la mayoría de índices y tablas usadas más frecuentemente deben estar en el cache de datos.

CACHE DE DATOS

El cache de datos está compuesto de la memoria sobrante del Servidor de SQL una vez que los requisitos del cache de procedimiento han sido satisfechos. Lo mejor es tener bastante espacio del cache para contener la mayoría los índices usados y un porcentaje respetable de las tablas mas accedidas frecuentemente, reduciendo las entradas/salidas físicas.

CACHE DE PROCEDIMIENTOS

El Servidor de SQL distribuye 30 por ciento de memoria disponible al cache de procedimientos después que el kernel de SQL se ha asignado. Es importante recordar que el Servidor de SQL guardará una copia duplicada de cada procedimiento que se accede por más de un usuario.)

TEMPDB en RAM

Las consultas ejecutadas contra el Servidor de SQL que usan espacio de trabajo temporal como las operaciones, GROUP BY, tablas temporales, uniones de tablas (JOIN), estas son beneficiadas si existe suficiente espacio en RAM, además es importante mencionar que la RAM requerida por el Sistema Operativo debe tener más espacio disponible para poder ser utilizada por SQL SERVER.

2.10.2. INDICES EN SQL SERVER

Probablemente, la creación de índices útiles es la acción más importante que se puede realizar para mejorar el rendimiento. El tipo y número de índices, así como las columnas que se van a indizar se deben seleccionar cuidadosamente gracias a una profunda comprensión de las necesidades del usuario y de los propios datos. Los índices resultan útiles si sólo va a consultar la tabla o realizar modificaciones en los datos. En ambos casos, los índices pueden proporcionar un acceso más rápido a los datos, ya sea con propósito de leer o de escribir.

En SQL Server, se asigna espacio para los índices y tablas en trozos de páginas de 8 KB, la longitud máxima de una fila de índice es de 900 bytes y acepta 16 como el máximo número de columnas en un índice; el índice que almacena las filas de la tabla se denomina clúster, un índice clúster sólo se define si una tabla de índice requiere. Los otros índices, índices no agrupados (nonclustered)) apuntan al índice agrupado (clustered). [3]

2.10.2.1. ¿QUÉ ES UN INDICE?

Los índices son objetos de la bases de datos, cuya función es optimizar el acceso a datos. A medida que las tablas se van haciendo más grandes y se desea hacer consultar sobre estas tablas, los índices son indispensables.

Internamente un índice normal es una estructura de árbol-B, que cuenta con una página principal y luego esta con paginas hijas, que a su vez tiene más páginas hijas hasta llegar a la pagina final del índice (leaf level).

La clave del índice está repartida en las páginas del índice, de modo tal que la búsqueda se haga leyendo la menor cantidad posible de datos. [9]

Estructura interna de un índice:

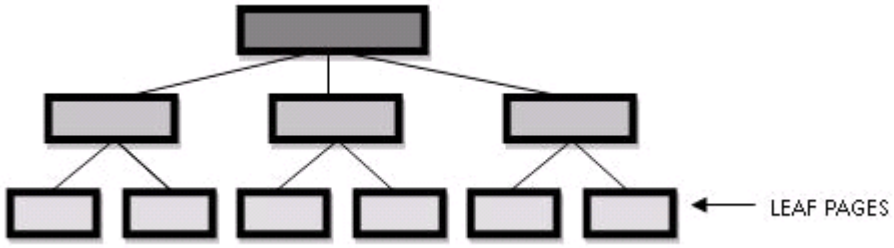


FIGURA II. 11: Estructura interna de un índice

Existen dos tipos de índices en SQL Server, estos son índices clustered e índices non-clustered, en realidad la diferencia de estos dos está en la leaf level (la última página) del índice. [15]

2.10.2.2. INDICE CLUSTERED

En un índice clustered, la leaf level es la página de datos. Con lo cual, el SQL SERVER, se ahorra hacer un salto para leer los datos del registro (Bookmark lookup). La diferencia es importante, ya que el uso de este tipo de índices al evitar tener que

hacer lecturas adicionales para traer el registro, son más óptimos para mejorar el rendimiento.

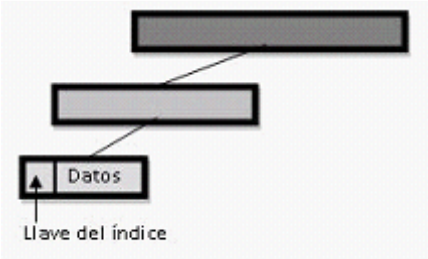


FIGURA II. 12: Búsqueda por clustered índice

Es importante analizar que no siempre se debe usar índices clustered, debido a que, lamentablemente solo puede haber un sólo índice clustered por tabla. La razón es muy sencilla y lógica: Los registros de la tabla físicamente son las páginas leaf-level del índice clustered. Los datos de la tabla esta ordenados según el índice. Y obviamente una tabla no puede simultáneamente estar físicamente ordenada de dos maneras diferentes.

Los índices agrupados, definen el orden en que almacenan las filas de la tabla (nodos hoja/página de datos de la imagen anterior). La clave del índice agrupado es el elemento clave para esta ordenación; el índice agrupado se implementa como una estructura de árbol B que ayuda a que la recuperación de las filas a partir de los valores de las claves del índice agrupado sea más rápida. Las páginas de cada nivel del índice, incluidas las páginas de datos del nivel hoja, se vinculan en una lista con vínculos dobles. Además, el desplazamiento de un nivel a otro se produce recorriendo los valores de claves.

Por lo tanto, en tablas grandes y muy consultadas, se debe tener cuidado y analizar a que campos se va a seleccionar para ser llaves del índice clustered. Se tiene un sólo índice de este tipo por tabla, por lo tanto no se debe desperdiciarlo. Este último punto

es importante para saber en qué situaciones y para que campos se debe utilizar un clustered index o un non-clustered. [9]

Se deben utilizar estos índices cuando existen:

- Columnas selectivas
- Columnas afectadas en consultas de rangos: BETWEEN, mayor que, menor que, etc.
- Columnas accedidas "secuencialmente"
- Columnas implicadas en JOIN, GROUP BY
- Acceso muy rápido a filas: lookups

2.10.2.3. INDICE NON-CLUSTERED

Los índices no agrupados (NON-CLUSTERED) tienen la misma estructura de árbol B que los índices agrupados, con algunos matices; en los índices agrupados, en el último nivel del índice (nivel de hoja) están los datos; en los índices no-agrupados, en el nivel de hoja del índice, hay un puntero a la localización física de la fila correspondiente en el índice agrupado. Además, la ordenación de las filas del índice está construida en base a la(s) columna(s) indexadas, lo cual no quiere decir (a diferencia de los índices agrupados), que la organización física de las páginas de datos corresponda con el índice.

En un índice non-clustered, la clave por la que se busca tiene un puntero a la página de datos donde se encuentra el registro.

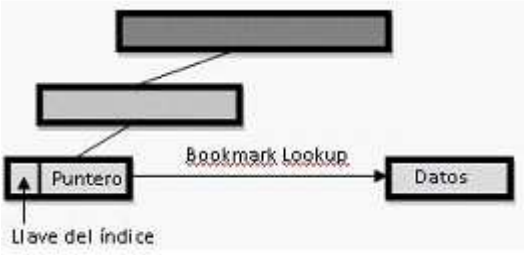


FIGURA II. 13: Búsqueda por non-clustered índice

Al utilizar este tipo de índices, se utiliza el lookup (lecturas adicionales para traer el registro) por eso es preciso utilizarlos de manera correcta, generalmente cuando son consultas selectivas. [9]

Se deben utilizar estos índices cuando existen:

- Columnas con datos muy selectivos
- Consultas que no devuelven muchas filas.
- Columnas en WHERE.
- Evitar acceso a páginas de datos realizando el acceso sólo por el índice.
- Covered queries (consultas cubiertas).

En SQL SERVER 2005, son nuevos los índices INCLUDE que son índices no-agrupados que en el nivel de hoja del índice (donde está el puntero al índice agrupado), se puede incluir más columnas; el objetivo de este nuevo tipo de índices es beneficiar el uso de las consultar cubiertas para evitar que se acceda a la página de datos del índice agrupado.

2.10.2.4. GUÍA GENERAL DE USO DE ÍNDICES

Se debe tomar en cuenta:

- Campos autoincrementales (Identitys, newsequentialid, etc), deben convenientemente ser del tipo clustered index. La razón es reducir el page split (fragmentación) de la tabla.
- Los clustered index son convenientes si se selecciona un rango de valores, ordenar (ORDER BY) o agrupar (GROUP BY).
- La PK es un buen candidato para un clustered index, pero no siempre. Por ejemplo, si se tiene una tabla de ventas, cuya PK es un identity en donde se efectúan muchas consultas por rangos de fecha, el campo Fecha sería un mejor candidato para el clustered que la PK.
- Para búsquedas de valores específicos, conviene utilizar un non-clustered index.
- Para índices compuestos, mejor utilizar non-clustered index (generalmente). [9]

2.10.3. BLOQUEO DE CONSULTAS SQL

En las versiones de SQL SERVER anteriores a la 2005 se usaba el control pesimista y basado en bloqueos. En la versión 2005 aparece el versionado de filas.

El proceso sigue de la siguiente manera: cuando se actualiza un registro de una tabla, al nuevo registro se le asigna el número de la transacción que realiza la actualización, la versión anterior del registro se almacena en una base de datos temporal, y el nuevo registro contiene un puntero a su versión anterior, que a su vez puede enlazar a su propia versión anterior, proporcionando así una lista enlazada. Las versiones de registros sólo deben almacenarse mientras haya operaciones que puedan requerirlos.

NIVELES DE AISLAMIENTO DE TRANSACCIONES

Para controlar como afectan los bloqueos a las transacciones, se puede modificar el nivel de aislamiento de las transacciones a través de la instrucción SET TRANSACTION ISOLATION LEVEL. Con esta instrucción se controla como interpreta la transacción los bloqueos existentes y como genera nuevos bloqueos con sus operaciones de lectura/escritura.

Su sintaxis general es la siguiente: SET TRANSACTION ISOLATION LEVEL <opción>.

Donde <opción> puede tomar estos valores:

- **READ COMMITTED:** La transacción no puede leer datos modificados por otras transacciones. Permite que otras transacciones puedan modificar los datos que se han leído. Esta opción es la predeterminada para SQL Server 2005.
- **READ UNCOMMITTED:** La transacción es capaz de leer los datos modificados por otras transacciones pero que aún no han sido confirmadas (pendientes de COMMIT).
- **REPEATABLE READ:** La transacción no puede leer datos modificados por otras transacciones y otras transacciones no pueden modificar los datos que se han leído.
- **SERIALIZABLE:** Las instrucciones no pueden leer datos que hayan sido modificados, pero aún no confirmados, por otras transacciones y ninguna otra transacción puede modificar los datos leídos por la transacción actual ni insertar filas nuevas con valores de clave que pudieran estar incluidos en el intervalo de claves hasta que la transacción actual finalice.

En SQL Server 2005 se ha incluido un nuevo nivel de aislamiento, el nivel SNAPSHOT que asegura la coherencia de los datos para toda la transacción:

- **SNAPSHOT:** Activa el versionado de fila. Las instrucciones que se ejecuten en la transacción no verán las modificaciones de datos efectuadas por otras transacciones, en su lugar reciben una "copia coherente" de cómo estaban los datos al comienzo de la transacción. De este modo actúan otros gestores de bases de datos muy populares, como por ejemplo, ORACLE.

  El nivel de aislamiento SNAPSHOT no está disponible de forma predeterminada. La opción Snapshot usa el versionado de filas para proporcionar consistencia de lectura a nivel de transacciones. Las operaciones de lectura no piden bloqueos. Cuando leen filas modificadas por otra transacción, recuperan la versión de la fila que existía antes de que la transacción comenzara. Se activa con la opción ALLOW_SNAPSHOT_ISOLATION, que por defecto está desactivada. [1]

CAPÍTULO III: ANÁLISIS Y SELECCIÓN DE TÉCNICAS TUNING

3.1. INTRODUCCIÓN

Para el análisis y selección de las técnicas tuning que se aplicarán en el desarrollo de la tesis, es importante definir el proceso mediante el cual se determinarán, ya que es primordial precisar los pasos a seguir. Tomando en cuenta de que se está hablando de requerimientos no funcionales relacionados entre sí (especifica criterios que pueden ser usados para juzgar la operación de un sistema, pero no su funcionalidad, estos criterios son las restricciones, atributos de calidad, calidad de servicio, objetivos de calidad. Estos requieren generalmente un entendimiento profundo de las tecnologías involucradas, especialmente cuando se trata de performance en el Data Warehouse) y que involucran a los motores de bases de datos en:

- Performance
- Capacidad
- Escalabilidad

3.2. PROCESO DE INGENIERÍA DE PERFORMANCE O RENDIMIENTO

El proceso de Ingeniería de Performance o rendimiento es el siguiente:

- Definir los objetivos de Performance (rendimiento)
- Diseñar para el rendimiento
- Medir y testear el rendimiento
- Tuning de Performance [12]

Mediante este proceso se determinarán las técnicas que se aplicarán en el Data Warehouse del SII-ESPOCH.

- **Definir los objetivos de Performance (rendimiento):** El objetivo del rendimiento consiste en proporcionar un tiempo de respuesta aceptable en las consultas al, reducir la entrada y salida de disco y minimizar el tiempo de CPU para permitir el máximo rendimiento de los procesos de los usuarios. Este objetivo se logra mediante el análisis de los requisitos de las aplicaciones, la comprensión de la estructura física y lógica de los datos, y la capacidad de evaluar y negociar las contrapartidas entre usos conflictivos del data warehouse, frente a la ayuda a la toma de decisiones.

  • Tiempos de Respuesta

  – Por transacción, emisión de reporte, etc

  • Rendimiento (Throughput)

  – Transacciones por segundo

  • Utilización de recursos

- CPU, Disk I/O, memoria

• Carga de trabajo

- Usuarios concurrentes, usuarios totales, volumen de datos, volumen de transacciones, etc.

Calidad de Servicio: Performance es uno de los atributos de calidad de servicio.

- **Diseñar para el rendimiento:** Desde el inicio es importante definir el diseño ya que es la base de la que se parte, tomando en cuenta que el diseño se mantendrá a lo largo de todo el proyecto y de este dependerá el rendimiento, escalabilidad y capacidad. El mal diseño afecta al modelado de la base de datos, tomando en cuenta que el diseño encierra el diseño físico y lógico del Data Warehouse

- **Medir y testear el rendimiento:** Este es el punto de partida en el que se analizan los requerimientos no funcionales y objetivos de performance. En esta etapa se obtiene la información de performance o rendimiento, para establecer los puntos de comparación y solucionar el problema, localizando queries más usados y los más lentos. [12]

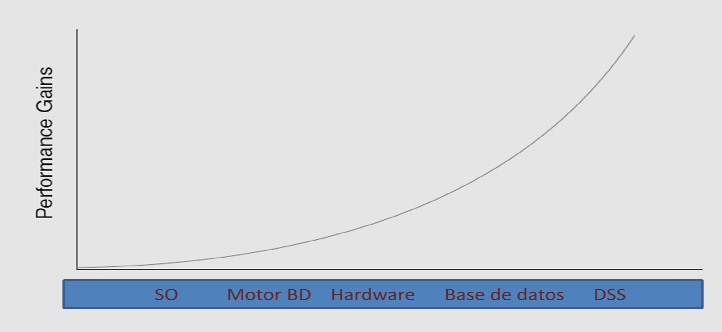


FIGURA III. 14: Performance en base de datos, en donde hay que focalizar esfuerzo

- **Tuning de Performance:** El tuning de performance es la parte final, en la que se aplican las técnicas tuning, permitiendo demostrar el mejoramiento del rendimiento del Data Warehouse.

3.2.1. CRITERIOS DE ANÁLISIS (CAPAS TUNING)

Para este análisis, se definen a continuación los criterios sobre los cuales se analizarán las técnicas tuning del Data Warehouse, varios son los criterios que se pueden enfocar en el mejoramiento del rendimiento, pero para este análisis, únicamente se hará hincapié en el DBMS, el DISEÑO DEL DATA WAREHOUSE, y las CONSULTAS, como se detallan a continuación:

TABLA II

Criterios de Análisis

CRITERIOS	
RED	DESCRIPCIÓN DE RED UTILIZADA
SERVIDOR (HARDWARE)	DESCRIPCIÓN DE HARDWARE DISPONIBLE
DBMS	PRESTACIONES DEL MOTOR DE BASE DE DATOS
DISEÑO DEL DATA WAREHOUSE	DESNORMALIZACION
	CREACION DE FILEGROUPS
CONSULTAS	INDICES
	PARTICIONAMIENTO DE TABLAS
	CALIDAD DE SQL SCRIPT

Elaborado por: Autora

3.3. DEFINICIÓN Y ANALISIS DE PARÁMETROS

El análisis de rendimiento se lo realizará mediante pruebas en el Data Warehouse del proyecto SII-ESPOCH, debido a que las medidas probarán mejoras en el rendimiento del data warehouse, estas medidas serán tomadas, dependiendo del criterio analizado, antes y después de las técnicas implementadas, pudiendo de esta manera conseguir el objetivo de rendimiento.

Para el análisis de las técnicas se manejarán parámetros para evaluar el rendimiento de consultas, utilizando el plan de ejecución real de SQL SERVER, siendo estos parámetros los siguientes:

- Tiempo de ejecución de la consulta
- Lectura lógica de páginas
- Costo de la consulta

TIEMPO DE EJECUCIÓN DE LA CONSULTA: El tiempo de ejecución de la consulta es la métrica más volátil. Es importante incluir siempre la métrica de tiempo de ejecución de la consulta en las comparaciones de rendimiento, ya que puede ayudar a detectar problemas de pérdidas de rendimiento de otros (Lectura de páginas y costo de la consulta).

LECTURA LÓGICA DE PÁGINAS: La lectura de páginas representa el número de 8 KB de páginas de datos que son accedidas por SQL Server en el motor de almacenamiento cuando se ejecuta una consulta. Se puede recuperar esta métrica ejecutando SET STATISTICS IO ON.

Al momento de la lectura de las páginas, se realizan lecturas lógicas, el total de las lecturas lógicas, es la suma de estas, la lectura lógica es el número de páginas leídas desde la memoria. Esta lectura representa el número total de páginas leídas desde un índice de la tabla determinada.

Los otros elementos que indican cómo muchas de las lecturas lógicas fueron leídas desde el disco duro (físico y lecuras read-ahead), el número de pasadas a través de un índice o lo que se tardó en responder a la consulta (scan count), y cómo muchas de las lecturas de páginas se utilizan para recuperar Objetos Grandes (LOB). Datos LOB se almacena fuera de la fila de los tipos de datos varchar (max), nvarchar (max), varbinary (max), text, imagen, ntext, y XML. La métrica leída en la página no tiene en cuenta la cantidad de recursos de CPU utilizados al ejecutar la consulta.

COSTO DE LA CONSULTA: El costo de la consulta es una métrica interna en SQL Server, en la que se maneja el CPU y recursos de E/S que son usados en la consulta. Mientras el costo de la consulta sea menor es mejor para el rendimiento. Este costo no es afectado por contención de recursos o por esperar bloqueos. [1]

Todos estos parámetros dependerán también de la cantidad de datos que se manejarán.

CANTIDAD DE DATOS: El conjunto de datos que se manejará durante las pruebas tendrá variación en la cantidad de datos, para de esta manera comprobar el rendimiento, siendo la cantidad de datos parte importante y de la que dependerán los tiempos de respuesta.

3.4. SELECCIÓN DE TÉCNICAS TUNING

Para la definición de parámetros y pruebas de análisis de técnicas tuning sobre el Data Warehouse del SII-ESPOCH, se van a tomar en cuenta los siguientes criterios:

- DBMS (servidor de BD-Software)
  - ✓ PRESTACIONES DEL MOTOR DE BASE DE DATOS
- DISEÑO DEL DATA WAREHOUSE
  - ✓ TÉCNICA DE DESNORMALIZACIÓN
  - ✓ CREACIÓN DE FILEGROUPS

- CONSULTAS
  - ✓ TÉCNICA DE UTILIZACIÓN DE ÍNDICES
  - ✓ TÉCNICA DE PARTICIONAMIENTO DE TABLAS
  - ✓ TÉCNICA DE CALIDAD DE SCRIPT SQL

3.4.1. DBMS (servidor de BD-Software)

Los sistemas de gestión de bases de datos (en inglés database management system, abreviado DBMS) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan, cuyo propósito general es el de manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante para una organización.

Existen distintos objetivos que deben cumplir los SGBD, siendo estas 25 características las que se analizarán, planteándose de acuerdo a los usos del DBMS en el proyecto SII-ESPOCH, por tal motivo las características analizadas en las que se dará mayor énfasis son optimización y Business Intelligence.

Las características son:

1. **LICENCIAMIENTO:** La licencia de software es un contrato entre el licenciante (autor/titular de los derechos de explotación/distribuidor) y el licenciatario del programa informático (usuario consumidor /usuario profesional o empresa), para utilizar el software cumpliendo una serie de términos y condiciones establecidas dentro de sus cláusulas. [18]

   Existen varios tipos de licencias, los que se analizarán son

   - Licencia de software de código cerrado o software propietario o privativo
   - Licencia de software de código abierto

2. **MULTIPLATAFORMA:** Multiplataforma se refiere a la compatibilidad que el DBMS tiene para funcionar con todos los Sistemas Operativos existentes, sean estos Propietarios u Open Source.

3. **SOPORTE:** El soporte técnico es un rango de servicios que proporcionan asistencia y tratan de ayudar al usuario a resolver determinados problemas con algún producto. La mayoría de las compañías que venden hardware o software ofrecen soporte técnico de manera telefónica o en línea.

4. **ACID:** Para asegurar la integridad de los datos se necesita que el sistema de base de datos mantenga las siguientes propiedades de las transacciones:

   A: Atomicidad

   O se ejecutan todas las operaciones o ninguna.

   C: Consistencia

   La ejecución aislada de las transacciones conserva la consistencia.

   I: (Isolation) Aislamiento

   Las Transacciones concurrentes no se afectan entre sí.

   D: Durabilidad

   Si la transacción finaliza con éxito, los cambios permanecen incluso si hay fallos.[16]

5. **INTEGRIDAD REFERENCIAL:** La integridad referencial es una propiedad deseable en las bases de datos. Gracias a la integridad referencial se garantiza que una entidad (fila o registro) siempre se relaciona con otras entidades válidas, es decir, que existen en la base de datos. Implica que en todo momento dichos datos sean correctos, sin repeticiones innecesarias, datos perdidos y relaciones mal resueltas. [16]

6. **CONSULTAS:** Al referirse a consultas se hace hincapié a la capacidad de manejar claves primarias y foráneas en las tablas de información.

7. **TABLAS TEMPORALES:** Son tablas que permanecen mientras el DBMS se está utilizando, facilitan ciertas operaciones pero una vez que el servidor deja de funcionar, se pierden.

   Los siguientes enunciados se refieren a las características básicas de un DBMS, siendo estas necesarias para manejar las transacciones y facilitar las necesidades del usuario:

8. **VISTAS**
9. **CURSORES**
10. **TRIGGER**
11. **FUNCIONES**
12. **PROCEDIMIENTOS**

Es importante mencionarlas porque algunos DBMS no los tienen implementados.

13. **INDICES:** Los índices de los sistemas de bases de datos sirven para agilizar la obtención de registros en respuesta a ciertas condiciones de búsqueda. Una estructura de índice permite el acceso rápido a los registros de un archivo, y está asociada a una clave de búsqueda concreta. [18]

14. **PARTICIONAMIENTO DE TABLAS:** El Particionamiento de tablas en índices se refiere a la capacidad de un DBMS para dividir una tabla o índice, en trozos mucho más pequeños. Esta capacidad ofrece ventajas en muchas áreas:

    - La copia de seguridad y recuperación es más sencilla y rápida, puesto que se puede realizar sobre particiones individuales en lugar de sobre toda la tabla.
    - Las operaciones de carga en un entorno de Date Warehouse son menos intrusivas: se pueden agregar datos a una partición y entonces agregar la partición a una tabla, lo que es una operación instantánea. De igual forma, eliminar una partición con datos obsoletos desde una tabla es muy sencillo en un almacén de datos que mantenga una ventana de datos históricos.

- El rendimiento de la consulta se mejora sustancialmente, puesto que el optimizador puede reconocer que solamente se tiene que acceder a un subconjunto de las particiones de una tabla con el fin de resolver la consulta.

15. **REPLICACIÓN:** La replicación de datos consiste en el transporte de datos entre dos o más servidores, permitiendo que ciertos datos de la base de datos estén almacenados en más de un sitio, y así aumentar la disponibilidad de los datos y mejorar el rendimiento de las consultas globales.

16. **SEGURIDAD:** Consiste en las acciones que el gestor de base de datos toma en el manejo la base de datos, tomando en cuenta el volumen de las transacciones y las restricciones que tiene que especificar en el acceso a los datos.

17. **ESTABILIDAD:** La estabilidad es un área importante porque se trata sobre la resistencia a la corrupción de los datos cuando los cierres inesperados del DBMS se realizan. [16]

18. **HERRAMIENTAS GUI DE ADMINISTRACIÓN:** Para facilitar las necesidades del usuario del DBMS es importante que este preste herramientas GUI (Interfaz Gráfica de Usuario)**,** permitiéndole de esta manera al usuario que logre administrar la base de datos o almacén de datos de forma fácil.

19. **ASESOR DE OPTIMIZACION DE BASE DE DATOS:** El asesor del DBMS sugiere automáticamente mejoras en la arquitectura de la base de datos para mejorar el rendimiento.

20. **IMPORTACION / EXPORTACION:** El DBMS debe permitir Importar y Exportar datos, ya que son procesos ETLs ocultos que se realizan automáticamente y permiten al usuario mediante una interfaz utilizar información y agregarla a diferentes DWH o BD.

21. **DATA WAREHOUSING:** Se entiende por Data Warehousing el proceso de extraer y filtrar datos de las operaciones comunes de la organización, procedentes de los distintos sistemas de información operacionales y/o sistemas

externos, para transformarlos, integrarlos y almacenarlos en un depósito o almacén de datos (Data Warehouse) con el fin de acceder a ellos para dar soporte en el proceso de toma de decisiones de una organización. Es decir, la finalidad es convertir los datos operacionales en información relacionada y estructurada, homogénea y de mayor calidad, identificada convenientemente y que se mantenga en el tiempo, es decir, los datos más recientes no sustituyen a los precedentes, pero tampoco se acumulan de cualquier manera, sino que se suelen mantener con un mayor nivel de detalle los datos actuales, y de manera más agregada los datos anteriores. Se pretende crear un círculo virtuoso para la información [8]

22. **SERVICIOS PARA CARGAS ETLs:** Estos servicios para cargas ETLs deben permitir la extracción, transformación y carga de datos, para facilidad del usuario deben ser capacidades gráficas del DBMS.

23. **SERVICIOS DE MINERIA DE DATOS:** Además de la minería de datos el DBMS debería permitir limpieza de datos y minería de texto. Básicamente esto se refiere a la extracción no trivial de información que reside de manera implícita en los datos. Dicha información era previamente desconocida y podrá resultar útil para algún proceso. Es decir, la minería de datos prepara, sondea y explora los datos para sacar la información oculta en ellos. Bajo el nombre de minería de datos se engloba todo un conjunto de técnicas encaminadas a la extracción de conocimiento procesable, implícito en las bases de datos.

24. **HERRAMIENTAS DE BUSINESS INTELLIGENCE:** Las herramientas de BI deben crear y depurar la integración de datos, OLAP, minería de datos y generación de informes.

25. **SERVICIOS DE ANALISIS PARA PROCESAMIENTO OLAP:** Al referirse a servicios de análisis del procesamiento OLAP se toma en cuenta todas las facilidades que el DBMS ofrece para que se utilice el campo de la Inteligencia empresarial (o Business Intelligence) cuyo objetivo es agilizar la consulta de grandes cantidades de datos. La razón de usar OLAP para las consultas es la velocidad de respuesta.

3.4.2. DISEÑO DEL DATA WAREHOUSE

En el criterio del diseño del Data Warehouse se analizarán dos puntos importantes, como es la desnormalización y la creación de filegroups (grupos de archivos)

3.4.2.1. DESNORMALIZACIÓN

El proceso de normalizar una base de datos es lo que nos garantiza una serie de propiedades tales como la no redundancia de datos, la simplificación y optimización del rendimiento del motor para operaciones transaccionales (OLTP – agregar, eliminar, modificar y buscar una fila utilizando su clave primaria). Pero en algunos casos es notable que el rendimiento no se lleve de la mano con la normalización de bases de datos para operaciones de consultas (OLAP - DWH). [6]

La Desnormalización consta de un conjunto de técnicas que pueden ser aplicadas al diseño para mejorar el rendimiento de las consultas. Todas las técnicas de desnormalización crean alguna forma de redundancia de datos.

Toda desnormalización será funcionalmente dependiente del proceso de consulta que se quiere mejorar, y por tanto, disminuirá la independencia entre procesos y datos. Si la consulta cambia, entonces el diseño del esquema tendrá que ser revisado y eventualmente cambiado. Por estas razones, la desnormalización no se debe aplicar sin un detallado estudio previo.

Hay cuatro técnicas en la desnormalización:

- **Datos duplicados:** Cuando atributos individuales se introducen de forma redundante para reducir el número de tuplas que deben ser revisadas en una consulta.
- **Datos derivados:** Cuando se introducen atributos para valores calculados, para reducir el número de tuplas accedidas al realizar un determinado cálculo.
- **Claves subrogadas:** Cuando se introducen claves artificiales en el diseño, para sustituir la clave original en el caso de que ésta sea grande o ineficiente.
- **Vector de datos:** Cuando un atributo multivaluado se implementa con un sólo atributo o conjunto de atributos en el esquema. [7]

Estas técnicas de desnormalización, se aplican dependiendo de la situación del diseño del data warehouse, de los datos que se almacenen y de las consultas que se requieran, es importante mencionar que una de las técnicas que más se utilizan en el diseño del Data Warehouse es el uso de claves subrogadas, ya que una clave subrogada es un identificador único que se asigna a cada registro de una tabla de dimensión. Esta clave, generalmente, no tiene ningún sentido específico de negocio. Son siempre de tipo numérico. Preferiblemente, un entero autoincremental.

Existen varios motivos por los que se deben crear claves subrogadas, siendo los principales los siguientes:

- **Fuentes heterogéneas.** El Data Warehouse suele alimentarse de diferentes fuentes, cada una de ellas con sus propias claves, por lo que es arriesgado asumir un código de alguna aplicación en particular. ¿Qué ocurriría si en el futuro se añade información de una aplicación que tiene su propio maestro de ciudades? Seguro que se generará un problema, aparecerán ciudades que no existían en el maestro. Es difícil gestionarlas, por esta razón lo mejor es crear propias claves subrogadas desde el inicio del proyecto.
- **Cambios en las aplicaciones origen.** Puede ocurrir que cambie la lógica operacional de alguna clave que se hubiera supuesto única, o que siempre debería estar informada. Lo mejor es crear propias claves subrogadas desde el

inicio del proyecto, porque no se sabe si todos los datos requeridos se obtendrán.

- **Rendimiento.** En la base de datos, ocupa menos espacio un entero que una cadena. Identificar una ciudad con 5 bytes, o una persona con 9 bytes es un desperdicio considerable de espacio. De hecho, no debe preocupar el espacio que ocupa, sino el tiempo que se pierde en leerlo. Es por eso que las claves subrogadas se las lleva a las tablas de hechos, por lo que cada código es susceptible de repetirse cientos de millones de veces. Conviene optimizarlo al máximo. Lo mejor es crear claves subrogadas desde el inicio del proyecto.[7]

Es importante mencionar que la desnormalización es un punto claro de optimización en un data warehouse, porque en resumen lo que hace es disminuir uniones de tablas (JOINs) para poder unir información que se encuentra relacionada.

3.4.2.2. CREACIÓN DE FILEGROUPS

SQL SERVER mapea una base de datos sobre un conjunto de archivos del sistema operativo. Los datos y la información del registro nunca se mezclan en el mismo archivo, y archivos individuales son solamente utilizados por una base de datos. [17]

Los grupos de archivos (FILEGROUPS) se denominan colecciones de archivos y se utilizan para simplificar la colocación de datos y tareas administrativas, tales como copia de seguridad y restauración. Las tablas y sus índices relacionados deben ser separados en archivos separados y los discos físicos para el funcionamiento óptimo de ejecución de consultas, pero no grupos de archivos por separado. Si están en grupos de archivos por separado, entonces no se puede realizar copias de seguridad y restauración de ellos como una sola unidad.

Hay tres tipos de archivos como:

- Primaria
- Secundaria
- Inicio de sesión

Cada base de datos se compone de al menos dos archivos: uno es un archivo de datos principal (por defecto, con la extensión mdf.), El otro es un archivo de registro (por defecto, con la extensión. ldf). Hay también archivos opcionales de datos secundarios (por defecto, con la extensión de la FDN).

Una base de datos sólo puede tener un archivo de datos principal, cero o más archivos de datos secundarios, y uno o más archivos de registro. Cada archivo de base de datos puede ser utilizado por una sola base de datos.

Los archivos de datos (. mdf y. FDN) se combinan en grupos de archivos. Un grupo de archivos es una colección de uno o más archivos de base de datos. Cada archivo de base de datos puede ser un miembro de un solo grupo de archivos. Los archivos de registro, por el contrario, no son miembros de grupos de archivos, pero se gestionan por separado.

Hay tres tipos de grupos de archivos:

- Primaria
- Definido por el usuario
- Por defecto

Cada base de datos tiene un sólo grupo de archivos principal, sólo un grupo de archivos predeterminado, y cero o más grupos de archivos definidos por el usuario. Si no se especifica los grupos de archivos definidos por el usuario, la base de datos contendrá sólo un grupo de archivos principal, que será también el grupo de archivos predeterminado. El grupo de archivos principal contiene el archivo de datos principal con todos los objetos del sistema en ella (tablas del sistema, procedimientos

almacenados del sistema, procedimientos almacenados extendidos y así sucesivamente). No se puede quitar objetos del sistema del grupo de archivos principal, pero se pueden crear objetos de usuario en los grupos de archivos definidos por el usuario para la asignación, el rendimiento y los propósitos de la administración. [24]

3.4.2.2.1. ESTRATEGIA DE ARCHIVOS Y GRUPOS DE ARCHIVOS

Los grupos de archivos utilizan una estrategia de relleno proporcional en todos los archivos dentro de cada grupo de archivos. Como los datos se escriben en el grupo de archivos, SQL SERVER Database Engine escribe una cantidad proporcional al espacio disponible en el archivo de cada archivo dentro del grupo de archivos, en lugar de escribir todos los datos para el primer archivo hasta que esté se llene. De esta manera, los archivos se llenan más o menos al mismo tiempo, y la creación de bandas simples se logra.

Tan pronto como todos los archivos en un grupo de archivos están llenos, el motor de base de datos expande automáticamente un archivo a la vez de un modo por turnos para permitir más datos, siempre que la base de datos está configurada para crecer automáticamente. Por ejemplo, un grupo de archivos se compone de tres archivos, todo listo para crecer de forma automática. Cuando el espacio en todos los archivos en el grupo de archivos se agota, sólo el primer archivo se expande. Cuando el primer archivo está lleno y no hay más datos se pueden escribir en el grupo de archivos, el segundo archivo se expande. Cuando el segundo archivo está lleno y no hay más datos se pueden escribir en el grupo de archivos, el tercer archivo se expande. Si el tercer archivo se llena y no hay más datos se pueden escribir en el grupo de archivos, el primer archivo se expande de nuevo, y así sucesivamente. [17]

3.4.2.2.2. MEJORA DEL RENDIMIENTO DE BASES DE DATOS Y ALMACENES DE DATOS

El uso de archivos y grupos de archivos mejora el rendimiento del almacén de datos, ya que permite que un almacén de datos que se creó en varios discos, varios controladores de disco, o RAID (matriz redundante de discos independientes) acceda a los datos en paralelo, al mismo tiempo. Esto acelera las operaciones el almacén de datos y mejora el rendimiento, ya que todas las E / S para una tabla específica puede ser dirigida a un disco específico.

3.4.2.2.3. REGLAS PARA EL DISEÑO DE ARCHIVOS Y GRUPOS DE ARCHIVOS

Las reglas siguientes se refieren a los archivos y grupos de archivos:

- Un archivo o grupo de archivos no puede ser utilizado por más de una base de datos.
- Un archivo puede ser miembro de un sólo grupo de archivos.
- Los archivos de registro de transacciones no son nunca parte de los grupos de archivos.

3.4.2.2.4. RECOMENDACIONES

Las siguientes son algunas recomendaciones generales cuando se trabaja con los archivos y grupos de archivos:

- La mayoría de las bases de datos funcionan bien con un único archivo de datos y un archivo de registro de transacciones (ldf).
- Si se utiliza varios archivos, crear un segundo grupo de archivos para el archivo adicional y hacer que el grupo de archivos sea un grupo de archivos

predeterminado. De esta manera, el archivo principal se incluirá en cuadros únicos del sistema y de los objetos.

- Para maximizar el rendimiento, crear archivos o grupos de archivos en diversos discos locales físicos disponibles es mejor. Coloque los objetos que compiten en gran medida por el espacio en grupos de archivos diferentes.
- Utilizar grupos de archivos para permitir la colocación de objetos en determinados discos físicos.
- Poner las diferentes tablas utilizadas que participan en las mismas consultas en grupos de archivos diferentes. Esto mejorará el rendimiento, a causa de que el disco en paralelo de E / S se une a la búsqueda de datos.
- Colocar tablas muy visitadas y los índices no agrupados que pertenecen a las tablas en grupos de archivos diferentes. Esto mejorará el rendimiento, ya que la E/S se realizará en paralelo si los archivos se encuentran en diferentes discos físicos.
- No se debe poner el archivo de registro de transacciones o los archivos en el mismo disco físico que tiene los otros archivos y grupos de archivos.

El propósito de crear un FILEGROUPS o Grupo de Archivos es poder acceder a datos e índices de forma paralela, esto optimiza las consultas lo cual mejora el tiempo de respuesta.

NOTA: Para obtener un rendimiento óptimo mediante creación de diferentes FILEGROUPS se debe por lo menos conseguir que el espacio asignado a DATOS y a INDICES se encuentre en discos físicos distintos

3.4.3. CONSULTAS

En este criterio de consultas se van a analizar los puntos más importantes del tuning del almacén de datos, siendo estos el uso de índices, particionamiento de tablas y la calidad del script SQL.

3.4.3.1. INDICES

Probablemente, la creación de índices útiles es la acción más importante que se puede realizar para mejorar el rendimiento. El tipo y número de índices, así como las columnas que se van a indizar se deben seleccionar cuidadosamente gracias a una profunda comprensión de las necesidades del usuario y de los propios datos. Los índices resultan útiles si sólo va a consultar la tabla o realizar modificaciones en los datos. En ambos casos, los índices pueden proporcionar un acceso más rápido a los datos, ya sea con propósito de leer o de escribir.

La optimación de índices es una faceta importante en cualquier base de datos o almacén de datos. Su función principal es reducir el tiempo de respuesta a las operaciones DML al disminuir la actividad de entrada y salida, tanto a disco como a memoria. Se emplean tanto en la búsqueda como en el ordenamiento de datos. No sólo porque haya una cláusula ORDER BY se debe ordenar, sino porque partes del plan de la consulta se pueden aprovechar al momento que los datos vengan ordenados. De todos los aspectos de optimación de un almacén de datos, la optimación de índices es lo único que puede mejorar el rendimiento en múltiplos en vez de en porcentajes. Es por eso que es tan importante.

Solamente la falta de índices, sino los índices mal diseñados constituyen las principales fuentes de atascos en aplicaciones de un almacén de datos. El diseño eficaz de los índices tiene gran importancia para conseguir un buen rendimiento de un almacén de datos, tanto en búsquedas de datos como en escritura. La selección de los índices apropiados para la estructura de una base de datos y su carga de trabajo, es

una compleja operación que busca el equilibrio entre la velocidad de la consulta y el costo de actualización. Los índices estrechos, o con pocas columnas en la clave de índice, necesitan menos espacio en el disco y son menos susceptibles de provocar sobrecargas debido a su mantenimiento. Por otra parte, la ventaja de los índices anchos es que pueden llegar a cubrir las consultas. Hay que experimentar con distintos diseños hasta encontrar el índice más eficaz. Se debe recordar que agregar, modificar y quitar índices no afecta al esquema o al diseño del almacén de datos, por lo que es más seguro experimentar.

3.4.3.1.1. INDICES EN SQL SERVER

En SQL SERVER, los índices se organizan como árboles B. Las páginas de un árbol B de índice se llaman nodos del índice. El nodo superior del árbol B se llama nodo raíz (root node) y contiene la entrada al índice. Es decir, cualquier búsqueda, sea ordenada o desordenada, debe pasar por este nodo. Incluso un recorrido completo debe encontrar la primera página de datos partiendo del nodo raíz. El nivel inferior de los nodos (leaf level) del índice se denomina nodos de hoja (Leaf pages). Los niveles del índice entre el nodo raíz y los nodos hoja se conocen en conjunto como niveles intermedios (Intermediate level).

En un índice agrupado, los nodos hoja contienen las páginas de datos de la tabla subyacente. Por tanto, la tabla completa está ordenada según el índice. También, todo acceso a la tabla se hace desde el nodo raíz.

En un índice no agrupado, los nodos hoja tienen una referencia al índice agrupado (si existe) o al índice montón (si la tabla no tiene ningún índice agrupado). Por tanto, si la tabla tiene un índice agrupado, sus columnas clave figuran siempre dentro del índice –como una especie de columna incluida–, pues son la referencia para encontrar las filas en la tabla en SQL SERVER 2005, la funcionalidad de los índices no agrupados puede ampliarse si se agregan columnas incluidas, denominadas columnas sin clave, en el nivel hoja del índice.

Las columnas de clave se almacenan en todos los niveles de índice no agrupado, mientras que las columnas sin clave sólo se almacenan en el nivel hoja. El nodo raíz y los nodos intermedios incluyen páginas de índice que contienen filas de índice. Cada fila de índice contiene un valor clave y un puntero a una página de nivel inferior en el árbol B, o bien a una fila de datos en el nivel hoja del índice (clave del índice agrupado o RID del montón). Las páginas del nivel hoja del índice se vinculan en una lista con vínculos dobles, por lo que en SQL SERVER 2005 es tan eficiente hacer un recorrido ascendente como descendente. Por esto casi nunca se indica cómo se quieren las columnas, excepto en índices compuestos (varias columnas) en los que unas son ascendentes y las otras descendentes. [26]

3.4.3.1.2. CREACION DE ÍNDICES

SQL SERVER admite dos tipos básicos de índices: agrupados y no agrupados. Ambos índices se implementan como un árbol balanceado, donde leaf level (última página) es el nivel inferior de la estructura. La diferencia entre estos tipos de índices es que el índice agrupado es la tabla real, es decir, el nivel inferior de un índice agrupado contiene las filas reales, incluyendo todas las columnas de la tabla. Un índice no agrupado, por el contrario, sólo contiene las columnas incluidas en la clave del índice, más un indicador que señala a la fila de datos reales. Si una tabla no tiene un índice agrupado definido en él, se le llama montón (Heap), o una tabla sin clasificar. También se puede decir que una tabla puede tener uno de dos formas: o bien es un montón (Heap) (sin clasificar) o un índice agrupado (clasificado). [1]

UNICIDAD

Cuando un índice es definido como UNIQUE, la clave del índice y sus correspondientes valores de la clave serán únicos. Un índice UNIQUE puede ser aplicado a cualquier columna si todos los valores de la columna son únicos. Un índice UNIQUE se puede definir sobre un conjunto de columnas mediante un índice compuesto. Por ejemplo, un índice UNIQUE puede ser definido sobre las columnas Apellido y NumDocumento, ninguna de ambas columnas deberá tener valores nulos y

las combinaciones de los valores de ambas columnas para los registros deberán ser únicas.

SQL SERVER automáticamente crea un índice UNIQUE para una columna o columnas definidas con las restricciones PRIMARY KEY o UNIQUE. Por lo tanto, se debe utilizar sólo las restricciones para forzar unicidad en vez de aplicar la característica UNIQUE al índice. SQL SERVER no permite crear un índice UNIQUE sobre una columna que contenga valores de la clave repetida.

LOCALIZADORES DE FILA

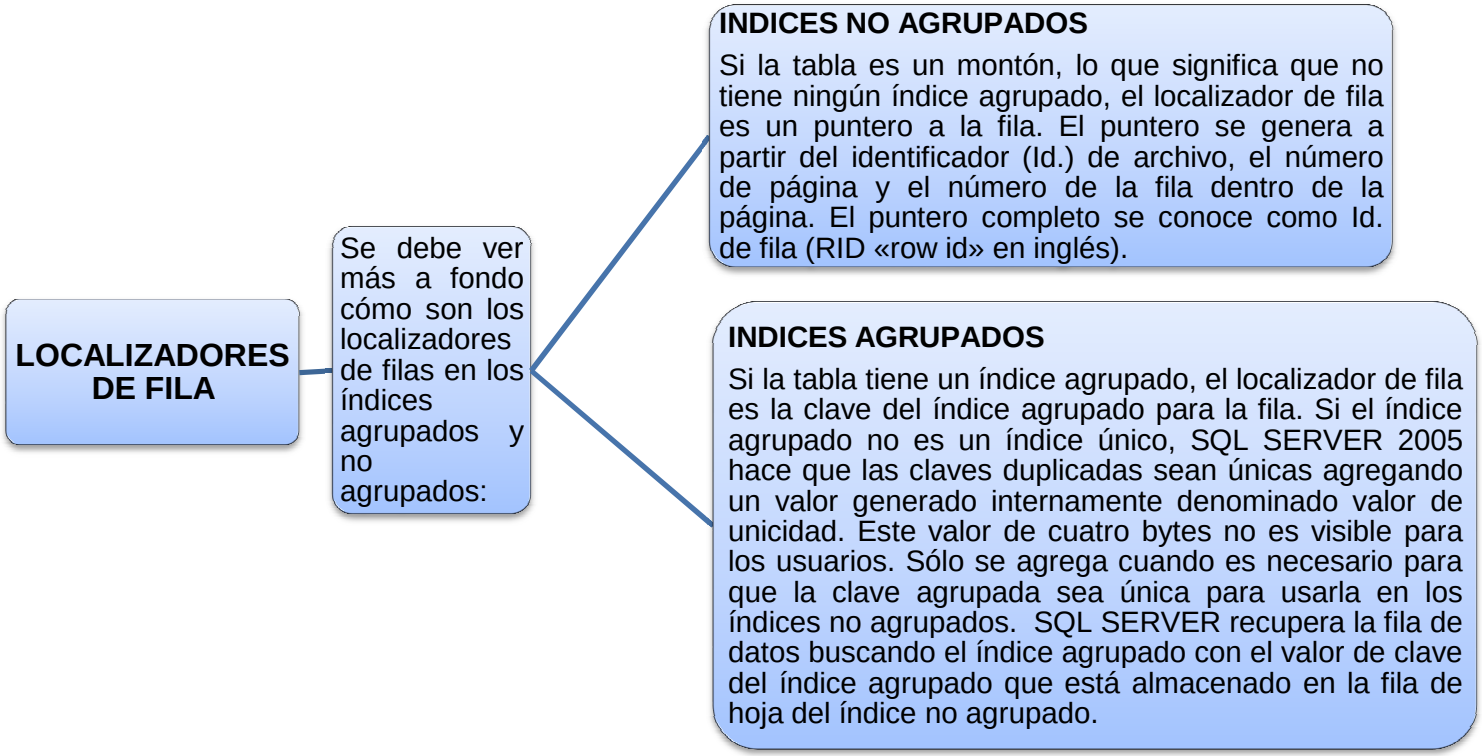


FIGURA III. 15: Localizadores de fila

Elaborado por: Autora

De este último punto se pueden extraer varias conclusiones:

- Cuando más ancha sea la clave de un índice agrupado, más ocuparán sus índices no agrupados pues siempre la incluyen.
- Si el índice agrupado no es único, añade 4 bytes a cada entrada de los índices no agrupados.

Por esto en muchos diseños se encuentra una clave artificial de las claves naturales que pueda tener una tabla, que es de tipo IDENTITY, sobre el que se crea el índice agrupado. Esto puede hacer más eficientes los índices porque:

1. Las comparaciones de enteros son las más rápidas por ser tipos nativos de cualquier procesador aunque al crear un campo nuevo hace más grande la tabla y aumenta las lecturas físicas, pero en la actualidad un campo INT añadido no se nota.
2. Los índices no agrupados salen más pequeños al tener el campo de tipo INT y por tanto hay menos lecturas físicas. Aunque estos datos pueden estar desordenados y por lo tanto aumentan las lecturas físicas aleatorias que son las más costosas, es por eso que se debe ordenar para evitar el aumento de lecturas físicas.
3. Se incluye un campo en todos los índices que nunca se va a usar para cubrir consultas pues es una clave artificial, a diferencia de las naturales que es mucho más fácil que aparezcan en las consultas. En todo caso sí se recomienda utilizar una longitud corta en la clave de los índices agrupados. Los índices agrupados también mejoran si se crean en columnas únicas o que no admitan valores NULL. Hay que examinar la unicidad de las columnas, no sólo porque el índice será más rápido al no tener el contador, sino porque un índice único en lugar de un índice no único con la misma combinación de columnas, proporciona información adicional para optimizar las consultas y, por tanto, resulta más útil. [26]

3.4.3.1.3. MEJORAR EL RENDIMIENTO CON ÍNDICES CUBIERTOS

La noción de un índice cubierto es que SQL Server no necesita usar las búsquedas entre el índice no agrupado y la tabla para devolver los resultados de la consulta.

Debido a que un índice agrupado es la tabla real, índices agrupados siempre cubren las consultas. Para considerar el índice cubierto, debe contener todas las columnas de referencia en la consulta (en cualquier cláusula SELECT, JOIN, WHERE, GROUP BY, HAVING, y así sucesivamente.

La ventaja de rendimiento obtenido mediante el uso de un índice cubierto suele ser ideal para las consultas que devuelven un gran número de filas (una consulta no selectiva) y menor para las consultas que devuelven pocas filas (una consulta selectiva). Se debe recordar que un pequeño número de filas es un término relativo, que podría significar 10 para una tabla con un par de cientos de filas y 1.000 para una tabla con millones de filas. Para las consultas que devuelven un número muy limitado de filas, un índice no cubierto también funciona muy bien. [1]

3.4.3.1.4. USO DE COLUMNAS INCLUIDAS (INCLUDED) Y LA REDUCCIÓN DE LA PROFUNDIDAD DE UN ÍNDICE

En las versiones de SQL Server anteriores a SQL Server 2005, la creación de índices no agrupados cubiertos a menudo puede resultar imposible debido a que un índice podría contener más de 16 columnas, o ser de más de 900 bytes. La nueva característica de columnas incluidas (Included) permite agregar columnas a un índice sin necesidad de añadir a las llaves de su índice. Las columnas incluidas no se pueden usar para tareas tales como el filtrado o la clasificación, su único beneficio es la reducción de lecturas de páginas que cubren a través de consultas, evitando búsquedas de la tabla (lookups).

Un índice puede tener un máximo de 1.023 columnas incluidas, así como una tabla puede tener un máximo de 1.024 columnas, por lo que es posible crear un índice no agrupado que cubre toda la tabla, que es casi como tener un segundo índice agrupado. Además, las columnas que utilice uno de los tipos de datos de gran tamaño [VARCHAR (max), NVAR-CHAR (max), VARBINARY (max), XML, TEXT, NTEXT E IMAGE] pueden ser incluidos en un índice de columna incluida.

Sólo las columnas que se utilizan para el filtrado, agrupación o clasificación debe ser parte de la clave de índice, todas las demás columnas incluidas en el índice debe ser columnas incluidas. Además de permitir más columnas en el índice, las columnas incluidas tienen otros beneficios.

Al utilizar índices con columnas incluidas y sin estas columnas incluidas, se puede ampliar o reducir la profundidad de los índices, los niveles no hoja (nivel que se encuentra antes del último nodo, nivel hoja) del índice con columnas incluidas contienen sólo la columna que está en la clave del índice (punteros más a un nivel superior), mientras que, por el otro índice, todas las columnas son parte de la clave de índice, haciendo que cada fila en el nivel no hoja más o menos tenga el mismo tamaño que la del nivel hoja. En las siguientes tablas se muestra la disposición de cada nivel de un índice de columnas no incluidas (NoIncludedColumns) y de un índice de columnas incluidas (IncludedColumns). [1]

TABLA III

Niveles de un índice de columnas no incluidas (NoIncludedColumns)

NIVEL	**CONTENIDO**
NODO RAIZ	1 página con 4 columnas señalando al siguiente nivel
PRIMER NIVEL INTERMEDIO	4 páginas con un total de 72 columnas señalando al siguiente nivel
SEGUNDO NIVEL INTERMEDIO	70 páginas con un total de 1668 columnas señalando al siguiente nivel
TERCER NIVEL INTERMEDIO	1668 páginas con un total de 40000 columnas

	señalando al siguiente nivel
NODOS DE HOJA (ULTIMO NIVEL)	40000 páginas que contienen 1000000 con todas las columnas del índice.

Fuente: Microsoft SQL Server 2008-Database Development.

TABLA IV

Niveles de un índice de columnas incluidas (IncludedColumns)

NIVEL	**CONTENIDO**
NODO RAIZ	1 página con 145 columnas señalando al siguiente nivel
NIVEL INTERMEDIO	145 páginas con un total de 40003 columnas señalando al siguiente nivel
NODOS DE HOJA (ULTIMO NIVEL)	40000 páginas que contienen 1000000 con todas las columnas del índice.

Fuente: Microsoft SQL Server 2008-Database Development.

Debido a las filas de las páginas de nivel no hoja del índice NoIncludedColumns son sustancialmente mayores que los del índice IncludedColumns, hay más páginas (y por tanto más niveles) son necesarios para crear el árbol balanceado para el índice.

Mientras el índice de NoIncludedColumns tenga más niveles que el índice de IncludedColumns, cada búsqueda a través del índice NoIncludedColumns las lecturas serán mayores y por lo tanto los niveles adicionales ocasionan una degradación del rendimiento. [2]

Profundidad del índice

Para estimar el rendimiento de un índice hay que ver el número de niveles que tiene, o lo que es lo mismo, el número de páginas leídas que tiene un acceso al índice. Recordando que en SQL SERVER 2005 los índices son árboles B balanceados, por lo

que siempre leerá el mismo número de páginas independientemente del dato que se busque.

Hay que estimar el número de niveles del índice siendo la tarea compleja y depende mucho del tipo de índices que se haga y de si la tabla tiene un índice agrupado y si éste es único. Respecto a la optimación de índices, se puede decir que cuanto mayor sea la profundidad, más lecturas lógicas hay que hacer. Y que si la tabla tiene un índice agrupado y el índice no agrupado no cubre la consulta, a la profundidad del índice no agrupado, habrá que añadir la búsqueda de las filas en el índice agrupado. Esto no quiere decir que una tabla con un índice agrupado tenga un acceso más lento, pues las páginas de los nodos raíz e intermedios suelen estar en memoria y por tanto son lecturas lógicas. Si está bien diseñado, puede disminuir las lecturas físicas o hacer que éstas sean ordenadas, lo que le daría una ventaja sustancial frente a una tabla con un índice montón.

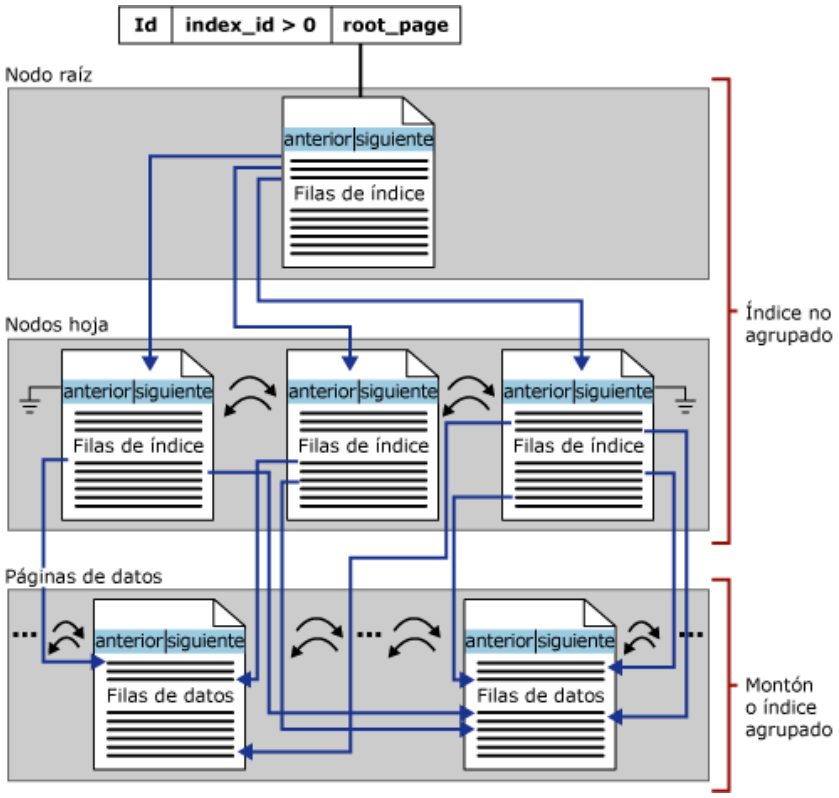


FIGURA III. 16: Estructura de un Índice SQL SERVER

Es importante distinguir entre bases de datos OLTP y almacenes de datos (OLAP). En estos últimos, lo normal es usar un proceso nocturno o en fin de semana para quitar los índices, actualizar las tablas y recrear dichos índices. Por tanto, suelen tener bastantes índices porque lo único importante es la velocidad de lectura; siempre que no llenen el espacio en disco. [26]

3.4.3.1.5. ÍNDICES COMPUESTOS

Un índice compuesto es cualquier índice que use más de una columna como clave. Los índices compuestos pueden mejorar el rendimiento de las consultas al reducir el número de operaciones de entrada/salida, porque una consulta sobre una combinación de columnas contenidas en el índice será ubicada completamente en el índice. Cuando el resultado de una consulta se obtiene completamente desde el índice sin tener que consultar los registros de la tabla, se puede decir que hay un recubrimiento de índice, esto tiene como resultado una extracción más rápida de los datos, ya que sólo se consultan las páginas del índice. Esto se produce cuando todas las columnas indicadas en las cláusulas SELECT y WHERE se encuentran dentro de la clave del índice o dentro de la clave del índice agrupado (si este existe). Es importante recordar que los valores de la clave del índice agrupado se encuentran también en las páginas de los índices no agrupados para poder encontrar los registros en la tabla.

3.4.3.1.6. USO DE ÍNDICES AGRUPADOS

Debido a que un índice agrupado es la tabla real, la lectura del índice agrupado no da lugar a búsquedas (lookup). Por lo tanto, un índice agrupado en general debe ser definido en las columnas que se han consultado y, normalmente, devuelven un montón de datos. El uso de un índice agrupado evita el problema de las búsquedas y obtener una gran cantidad de filas. Dos buenos candidatos para el índice agrupado son los más frecuentemente consultados, la columna de clave externa de la tabla (una

búsqueda en una clave externa normalmente devuelve filas) o la columna de la fecha más buscada de manera frecuente. (Citar las búsquedas devuelve un gran número de filas también.)

Otra consideración importante al seleccionar la columna o columnas sobre las que crear el índice agrupado es que el tamaño de la clave del índice clúster, esta debe ser lo más pequeño posible. Si existe un índice agrupado en una tabla, todos los índices no agrupados en esa tabla, usan la clave del índice agrupado como el puntero de fila del índice no agrupado en la tabla. Si un índice agrupado no existe, el identificador de fila se utiliza, que ocupa ocho bytes de almacenamiento en cada fila de cada índice no agrupado. Esto puede aumentar significativamente el tamaño del índice para las grandes tablas. [2]

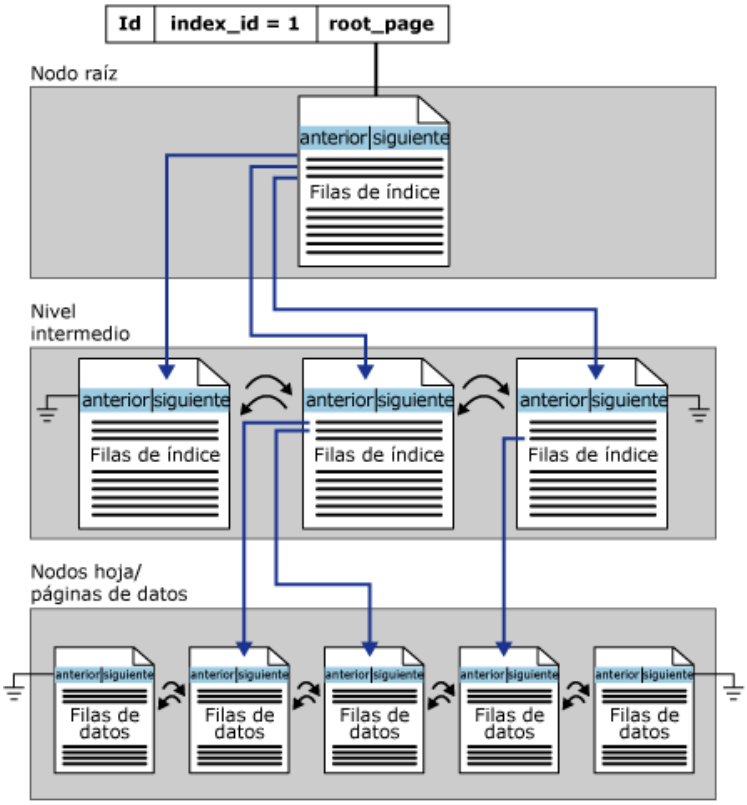


FIGURA III. 17: Estructura de un Índice Agrupado

3.4.3.1.7. EL RENDIMIENTO DE LECTURA VS RENDIMIENTO DE ESCRITURA

La adición de los índices sólo ayuda a aumentar el rendimiento de lectura. El rendimiento de escritura es normalmente degradada debido a los índices deben mantenerse al día con los datos de la tabla.

Si una tabla tiene cinco índices no agrupados definidos en él, un INSERT en la tabla es realmente seis INSERT: una para la tabla y uno para cada índice. Lo mismo ocurre con DELETE.

Con las instrucciones UPDATE, los índices que contienen sólo las columnas que se actualizan por la sentencia deben ser tocados.

Cuando se actualizan las claves de índice, la fila en el índice debe ser movido a la posición apropiada en el índice (a menos que la actualización modifica los datos sólo en las columnas incluidas). El resultado es que la actualización está dividida en un DELETE seguida de una instrucción INSERT. Dependiendo de la fragmentación interna de las páginas de índice, esto también puede causar divisiones de página.

Debido a que la relación entre las operaciones de lectura y escritura es muy variable entre los sistemas (e incluso tablas), es generalmente una buena idea para crear índices para optimizar el rendimiento de lectura y luego probar el efecto de que los índices han creado en el rendimiento de escritura. Siempre y cuando el rendimiento de escritura es aceptable (y se tiene suficiente espacio en disco para manejar los índices de creación), se puede mantener los índices creados. Normalmente se recomienda también para ejecutar una prueba de vez en cuando para comprobar que la lectura frente a la relación de escritura de la tabla no ha cambiado.

También se debe tener en cuenta que tanto las sentencias UPDATE y DELETE se benefician de ciertos índices para localizar las filas de la tabla que necesitan actualizar o eliminar. [1]

3.4.3.1.8. ANÁLISIS DE UTILIZACIÓN DE ÍNDICES

Debido a que los índices incurren en un costo (para el espacio de almacenamiento y para mantenerlas al día cuando se ejecutan instrucciones DML), es importante hacer un seguimiento de los índices que en realidad están siendo utilizados en las aplicaciones. Si un índice no se usa nunca, es probable que se pueda colocar tanto para ahorrar espacio de almacenamiento y reducir el costo de las operaciones de escritura. Sin embargo, se debe tener en cuenta que algunos de los índices se crean para un fin específico, por ejemplo, para optimizar los informes. Por lo tanto, se debe tener cuidado al retirar los índices utilizados o que rara vez se utiliza.

Al colocar un índice que rara vez se utiliza, debe documentar sus acciones para que el índice que se redujo se pueda volver a crear si es necesario más tarde. [1]

CONSIDERACIONES GENERALES

- Para que una consulta que se realiza en una tabla grande se ejecute rápidamente, resulta útil tener un índice en las columnas de la cláusula WHERE.
- Para determinar un número razonable de índices, se deberá tener en cuenta la frecuencia de las actualizaciones frente a las recuperaciones.
- Se deben elegir los índices según los tipos de cláusulas WHERE o combinaciones que realizará.
- La clave para seleccionar índices consiste en ser cuidadoso al elegir el tipo y número de índices. Determine el número mínimo que debe crear para que el

rendimiento no se vea afectado debido al mantenimiento. Se debe determinar los índices más útiles.

- No crear un índice si no se va a utilizar.
- Campos autoincrementales, deben convenientemente ser del tipo clustered index. La razón es reducir el page split (fragmentación) de la tabla.
- Los clustered index son convenientes si se va seleccionar un rango de valores, ordenar (ORDER BY) o agrupar (GROUP BY).
- La PK es un buen candidato para un clustered index. Pero no siempre. Por ejemplo, si tenemos una tabla de ventas, cuya PK es un identity en donde se efectúan muchas consultar por rangos de fecha, el campo Fecha seria un mejor candidato para el clustered que la PK.
- Para búsquedas de valores específicos, conviene utilizar un non-clustered index.
- Para índices compuestos, mejor utilizar non-clustered index (generalmente).

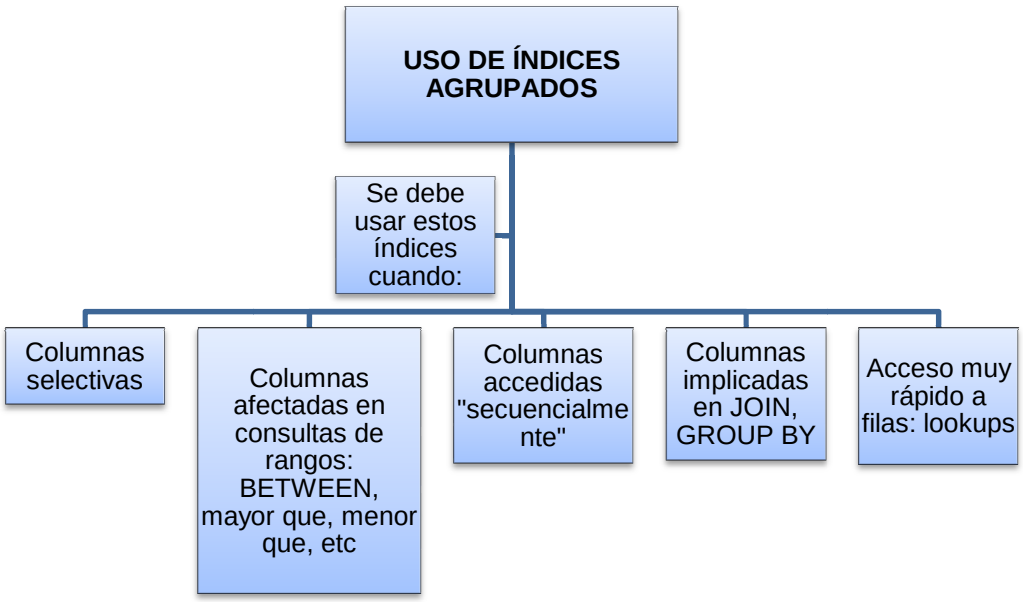


FIGURA III. 18: Uso de índices agrupados

Elaborado por: Autora

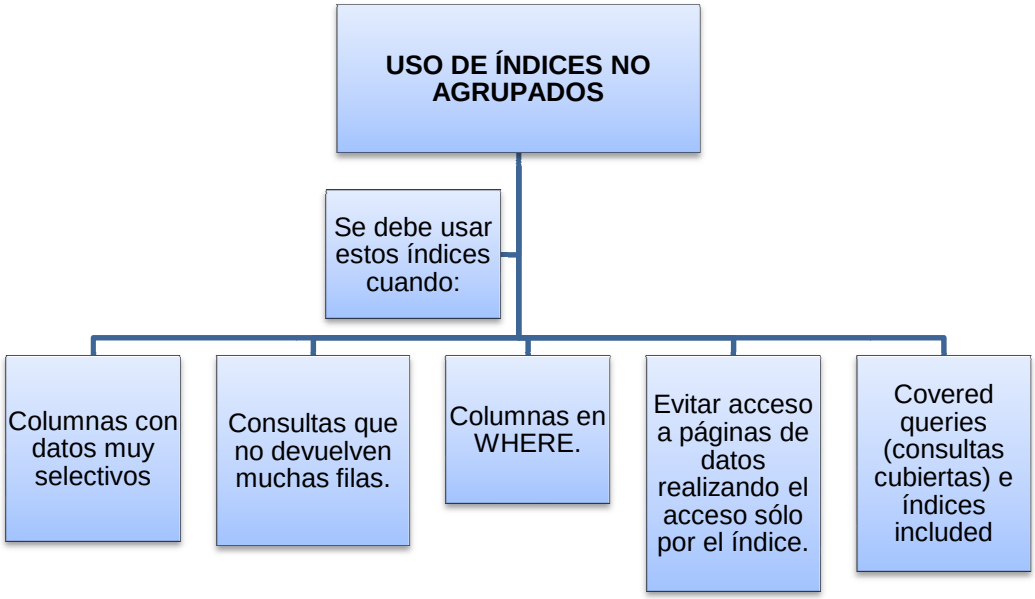


FIGURA III. 19: Uso de índices no agrupados

Elaborado por: Autora

PROBAR EL RENDIMIENTO DE LAS CONSULTAS

- Después de crear los índices, se debe probar el rendimiento de las consultas de prioridad más alta.
- Ejecutar SET SHOWPLAN ON, SET STATISTICS IO ON, SET STATISTICS TIME ON y, a continuación, se puede ejecutar cada consulta.

CUÁNDO NO SE DEBEN UTILIZAR ÍNDICES

Hay situaciones en que no deseará indizar. Entre estas situaciones se incluyen:

- Si el optimizador nunca utiliza el índice.
- Si se va a devolver más de un 10 a 20% de las filas.
- Si la columna contiene solamente uno, dos o tres valores únicos (selectividad baja).
- Si la columna que va a indizar es grande (> 20 bytes).
- Si la sobrecarga de mantener el índice es mayor que sus ventajas.
- Si la tabla es muy pequeña. [1]

3.4.3.1.9. RENDIMIENTO GLOBAL

Con todo lo analizado, se debe evitar agregar columnas que no sean necesarias. El hecho de agregar demasiadas columnas de índice, con o sin clave, puede tener las siguientes consecuencias en el rendimiento:

- Cabrán menos filas de índice en una página. Esto crea incrementos de entrada / salida y una reducción de la eficacia de la caché.
- Se necesitará más espacio en disco para almacenar el índice. En concreto, al agregar los tipos de datos varchar(max), nvarchar(max), varbinary(max) o xml como columnas de índice sin clave, aumentan significativamente los requisitos de espacio en disco. Esto se debe a que los valores de columnas se copian en el nivel hoja del índice. Por lo tanto, residen en el índice y en la tabla base.
- El mantenimiento del índice hace aumentar el tiempo necesario para realizar operaciones de modificación, inserción, actualización o eliminación en la tabla subyacente o la vista indizada.

Por tanto se debe determinar si la mejora del rendimiento de las consultas compensa el efecto en el rendimiento durante la modificación de datos y en los requisitos de espacio en disco adicionales. [26]

Antes de crear un índice, hay que evaluar no sólo la mejora que va a producir en las consultas, sino también el impacto global que va a tener en el rendimiento del almacén de datos. No siempre añadir un índice hace que mejore dicho rendimiento, por lo que hay que evaluar la relación del índice con la tabla, con las consultas, con el resto de los índices, el tipo de almacén de datos, etc. De esta manera se asegura que el rendimiento global, que es lo que interesa en definitiva, del almacén de datos, sea óptimo.

3.4.3.1.10. COMANDOS PARA ADMINISTRAR INDICES

Para administrar índices se pueden utilizar varios comandos, esta administración se la puede realizar a nivel de creación, eliminación, reconstrucción y uso de los índices.

SINTAXIS PARA CREAR UN INDICE

```
CREATE
[UNIQUE] [CLUSTERED |
NONCLUSTERED] INDEX
nombre_indice
ON [nombre_tabla |
nombre_IVsta] (nombre_columna
[,...n])
[WITH [propiedad_indice [,...n] ]
[ON grupo_archivos ]
```

PROPIEDADES DE LOS INDICES

Se pueden utilizar las siguientes características de los índices:

TABLA V

Propiedades de los Índices

TIPO DE ÍNDICE	OPCIONES
Agrupado	**ignore_dup_row** | **allow_dup_row**
Agrupado único	**ignore_dup_key**
No agrupado	Ninguna
No agrupado único	**ignore_dup_key**
No agrupado único	**ignore_dup_row**

Fuente: Microsoft SQL Server 2008-Database Development.

Uso de la opción ignore_dup_key

Si se intenta insertar un valor duplicado en una columna que tiene un índice único, el comando se cancela. Es posible evitar que se cancele una transacción grande incluyendo la opción ignore_dup_key con un índice unique.

El índice unique puede ser agrupado o no agrupado. Cuando se comienza la introducción de datos, cada intento de inserción de una clave duplicada se cancela, con un mensaje de error. Las claves no duplicadas se insertan de la forma habitual.

Uso de las opciones ignore_dup_row y allow_dup_row

La opción ignore_dup_row se utiliza para eliminar los duplicados de un lote de datos. Cuando se introduce una fila duplicada, esta fila se ignora y el comando insert en cuestión se cancela, con un mensaje de error informativo. Las filas no duplicadas se insertan de la forma habitual.

La opción ignore_dup_row se aplica sólo a las tablas con índices no únicos: no puede usar esta palabra clave si hay un índice único en alguna de las columnas de la tabla.

Uso de la opción sorted_data

La opción sorted_data acelera la creación de un índice cuando los datos de la tabla ya están clasificados según criterios de ordenación. Esta opción puede utilizarse con cualquier otra opción create index sin ningún efecto sobre su funcionamiento.

Si se especifica sorted_data , pero los datos no están ordenados, aparece un mensaje de error y el comando se aborta.

Esta opción agiliza la creación de índices sólo para índices agrupados o índices no agrupados únicos. Sin embargo, la creación de un índice no agrupado y no único tendrá éxito siempre que no haya filas con claves duplicadas. Si existen filas con claves duplicadas, aparece un mensaje de error y el comando se aborta.

SINTAXIS PARA ELIMINAR UN INDICE

```
DROP INDEX nombre_tabla.nombre_indice
```

SINTAXIS PARA RECONSTRUIR UN INDICE

```
DBCC DBREINDEX [nombre_tabla.nombre_indice]
```

El comando DBCC DBREINDEX reconstruye, a través de un solo comando, uno o más índices sobre una tabla o IVsta. Esta capacidad eIVta tener que utilizar múltiples comandos DROP INDEX y CREATE INDEX para reconstruir múltiples índices.

OBSERVAR LOS INDICES UTILIZADOS EN LA BASE DE DATOS

```
USE base de datos
GO
SELECT name, rows, rowcnt, keycnt from sysindexes
WHERE name NOT LIKE '%sys%'
ORDER BY keycnt
```

O también

```
sp_helpindex 'nombre_tabla'
```

El resultado que se retorna del sp_helpindex incluye el nombre del índice, el tipo de índice, el archivo de base de datos, y la o las columnas contenidas por el índice.

3.4.3.1.11. LOOKUP DE SQL SERVER

LookUp, traducido al español, significa operación de búsqueda, y representa exactamente eso. Realiza una operación de búsqueda, con un "identificador" de una fila, para traer más información, de la fila específica, desde la tabla.

La búsqueda se hace, utilizando el RID si la tabla no tiene un índice agrupado, o utilizando la llave del índice agrupado, si la tabla cuenta con uno de ese tipo como se muestra en la siguiente figura:

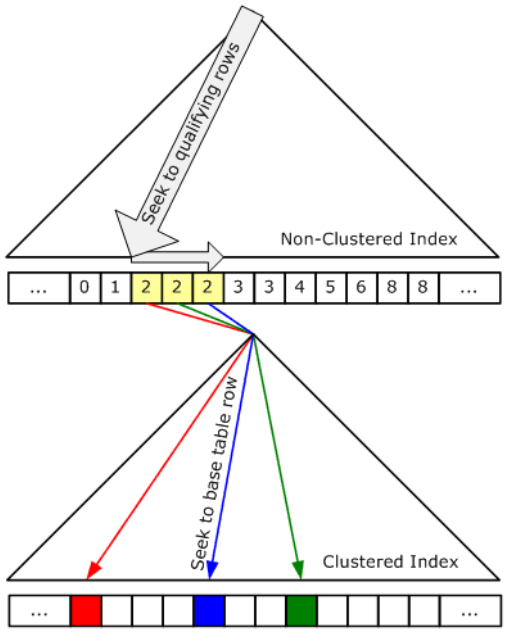


FIGURA III. 20: Operación de Búsqueda (LookUp) de Índices agrupados y no agrupados

En la figura se analizan los índices utilizando el índice no agrupado, primero se identifica las filas que cumplen con la condición de filtrado.

Resultado: Celdas amarillas.

Se usan los tres identificadores de filas (rojo, verde y azul) y se realiza la operación de búsqueda en el índice agrupado.

Si la tabla no tiene índice agrupado, la operación es similar, pero usa el RID.

Se retornan los datos faltantes.

Cuando existe una operación de búsqueda (lookup) muy costosa, es importante menorar esta carga, hay algunas maneras de hacerlo, una de ellas, y que no tienen ninguna gracia, es agregar todas las columnas necesarias al índice, y con eso, evitar la operación LookUp. Si se tiene toda la información en el índice, no se necesita ir a buscar más columnas a la tabla. Esta no falla nunca, pero se debe evitar incluir muchas columnas en los índices.

Otra solución, y que es mucho más elegante, y depende del escenario en el que se aplique, lo que se debe analizar es aumentar una columna al índice, observando el filtro que tiene la consulta, de esta manera disminuirá el lookUp.

Lo importante es minimizar el impacto de la operación LookUp en SQL Server, no eliminarla ya que para lograr eso, el camino es directo y simple. A veces no es necesario eliminarlo para mejorar drásticamente el rendimiento de una consulta. [13]

3.4.3.2. PARTICIONAMIENTO DE TABLAS

A partir de SQL SERVER 2005, se puede elegir particionar tablas e índices horizontal (es decir, por filas) en trozos más pequeños. El particionamiento puede mejorar el rendimiento de la consulta, pero la mejor opción para el rendimiento de consultas es la utilización de índices en lugar de repartir, pero al manejar gran cantidad de datos este particionamiento si mejorará el rendimiento.

Para poder tener tablas de particiones e índices, primero tiene que crear dos objetos: una función de partición y un esquema de partición. La función de partición, simplemente define los puntos (o más bien, los valores), donde cada partición termina. El esquema de partición define en que grupo de archivos va cada partición, se debe tener en cuenta que también puede definir un grupo de archivos para mantener todas las particiones. [1]

3.4.3.2.1. FUNCIONES DE PARTICIÓN

Funciones de partición se crean mediante la instrucción CREATE PARTITION FUNCTION. Una función de partición es simplemente una lista de hasta 999 valores que definen los divisores entre las particiones. Se puede decidir si los valores se interpretan como "menor o igual a" (<=) o "menor que"(<) mediante la definición de la función de partición como LEFT o RIGHT.

El siguiente código crea una función de partición LEFT.

```
CREATE PARTITION FUNCTION PF (INT)
AS RANGE LEFT
FOR VALUES (10,20,30);
```

Las particiones LEFT y RIGHT sirven para indicar qué partición deseamos que contenga cada valor frontera (la partición que queda a su izquierda o la que queda a su derecha). Las particiones resultantes de LEFT se muestran en la siguiente tabla:

TABLA VI

Tabla de particiones creada con LEFT

NÚMERO DE PARTICIÓN	RANGO DE LA PARTICIÓN
1	<= 10
2	> 10 AND <= 20
3	> 20 AND <= 30
4	> 30

Fuente: Microsoft SQL Server 2008-Database Development.

Se considera ahora el siguiente código, que crea virtualmente la misma función de partición sino que se define como RIGHT en vez de la LEFT.

```
CREATE PARTITION FUNCTION pf(INT)
AS RANGE RIGHT
FOR VALUES(10,20,30);
```

TABLA VII

Tabla de particiones creada con RIGHT

NÚMERO DE PARTICIÓN	RANGO DE LA PARTICIÓN
1	< 10
2	>= 10 AND < 20
3	>= 20 AND < 30
4	>= 30

Fuente: Microsoft SQL Server 2008-Database Development.

3.4.3.2.2. ESQUEMAS DE PARTICIÓN

Se define una función de partición utilizando la declaración CREATE PARTITION SCHEME. El esquema de partición es un mapa sencillo entre particiones para una partición en particular y los grupos de archivos.

La razón para el uso de grupos de archivos diferentes para las particiones diferentes suele ser capaz de almacenar diferentes partes de una tabla sobre los diferentes tipos de dispositivos de almacenamiento. [1]

3.4.3.2.3. CREACION DE UNA TABLA CON PARTICIONES

Después de haber creado la función de partición y el esquema de partición, se puede crear tablas e índices en el esquema de partición usando la cláusula ON de la sentencia CREATE TABLE y CREATE INDEX. Aunque se puede tener una tabla en un esquema de partición y sus índices en esquemas de partición diferentes (o una en un esquema de partición y otra no), se recomienda que todos ellos se creen en el mismo esquema de partición para apoyar a añadir y eliminar particiones cuando sea

necesario, sin tener que cerrar todas las aplicaciones. Una tabla con todos los índices definidos en el mismo esquema de partición se dice que son "particiones alineadas."

No únicamente, los índices no agrupados se alinean automáticamente con el esquema de partición de la tabla cuando se crea, es decir, que ni siquiera es necesario especificar la cláusula ON para la instrucción CREATE INDEX. Para índices únicos, sin embargo, se debe incluir la columna de partición en la clave de índice para que sea alineada con la tabla. Normalmente, esto contradice el objetivo de tener un índice único. Por lo general debe tener un índice único definido como no alineado, y si existe la necesidad de añadir o eliminar una partición, se debe borrar el índice no alineado, añadir o eliminar la partición, y luego volver a crear el índice no alineado. Al no realizar de esta manera el propósito de la partición no se elimina, porque si se tiene cinco índices alineados y uno no alineado, sólo se tiene que volver a crear el índice no alineado, que, sin particiones, posiblemente de igual manera se debería volver a crear los seis índices.

De esta manera se puede mejorar potencialmente el rendimiento de las consultas, sobre todo si la columna con particiones se consulta más. Si es así, SQL Server puede hacer la eliminación de la partición durante la optimización de modo que tiene que hacer buscar o explorar las operaciones sólo en ciertas particiones en lugar de en toda la tabla. Sin embargo, comparando el rendimiento de las consultas de una tabla sin un índice apropiado con una tabla sin particiones con un índice apropiado, se ve que el índice rinde mucho mejor que la partición (sin índice).

Se debe tener en cuenta que cuando ningún índice está disponible, la consulta en la tabla con particiones se comporta mejor. Pero cuando un índice está disponible, la consulta en la tabla sin particiones funciona mejor. Esto sucede porque en la tabla de particiones, la consulta realiza una operación de búsqueda en contra de dos índices (uno para cada partición), mientras que la consulta en la tabla sin particiones realiza una operación de búsqueda, frente a sólo un índice. [1]

Server no agrega la columna de partición al índice si ya está presente en el índice.

3.4.3.2.4. LIMITACIONES DE MEMORIA E ÍNDICES CON PARTICIONES

Las limitaciones de memoria pueden afectar al rendimiento o capacidad de SQL SERVER para crear un índice con particiones. Esto sucede especialmente cuando el índice no está alineado con su tabla base o no está alineado con su índice agrupado si ya se ha aplicado un índice agrupado a la tabla.

Cuando SQL Server realiza la ordenación para crear índices con particiones, primero crea una tabla de orden para cada partición. A continuación, crea las tablas de orden en el grupo de archivos respectivo de cada partición o en tempdb si se ha especificado la opción de índice SORT_IN_TEMPDB.

Cada tabla de orden requiere una cantidad mínima de memoria para su creación. Cuando crea un índice con particiones que está alineado con su tabla base, las tablas de orden se crean de una en una con menos memoria. Sin embargo, cuando crea un índice con particiones no alineado, las tablas de orden se crean al mismo tiempo.

En consecuencia, debe haber disponible memoria suficiente para permitir la realización de estas ordenaciones simultáneas. Cuanto mayor es el número de particiones, mayor es la cantidad de memoria necesaria. El tamaño mínimo para cada tabla de orden y para cada partición es de 40 páginas y 8 kilobytes por página. Por ejemplo, un índice con particiones no alineado con 100 particiones necesita memoria suficiente para ordenar en serie 4.000 (40 * 100) páginas al mismo tiempo. Si esta memoria está disponible, la operación de creación será satisfactoria, aunque ello afectará negativamente al rendimiento. Si esta memoria no está disponible, se producirá un error durante la operación de creación. De forma alternativa, un índice con particiones alineado con 100 particiones sólo necesita memoria suficiente para ordenar 40 páginas, ya que las ordenaciones no se realizan al mismo tiempo.

Tanto para los índices alineados como para los no alineados, el requisito de memoria puede ser mayor si SQL SERVER está aplicando grados de paralelismo a la operación de creación en un equipo con varios procesadores. Esto es así porque cuanto mayor son los grados de paralelismo, mayor es también el requisito de memoria. Por ejemplo,

si SQL SERVER establece los grados de paralelismo en 4, un índice no alineado con 100 particiones necesitará memoria suficiente para que cuatro procesadores puedan ordenar 4.000 páginas al mismo tiempo o 16.000 páginas. Si el índice con particiones está alineado, el requisito de memoria se reduce a cuatro procesadores que ordenan 40 páginas o 160 (4 * 40) páginas. Puede utilizar la opción de índice MAXDOP para reducir manualmente los grados de paralelismo. [10]

3.4.3.2.5. COMANDOS PARA ADMINISTRAR PARTICIONES

Para administrar las particiones se pueden utilizar varios comandos, esta administración consiste en la creación de funciones y esquemas de partición.

SINTAXIS PARA CREAR UNA FUNCIÓN DE PARTICIÓN

```
CREATE PARTITION FUNCTION PF(INT)
AS RANGE LEFT
FOR VALUES(1000,100000,1000000);
```

SINTAXIS PARA CREAR UN ESQUEMA DE PARTICIÓN

```
CREATE PARTITION SCHEME PS
AS PARTITION PF ALL TO([PRIMARY]);
```

SENTENCIA PARA OBSERVAR LAS PARTICIONES CREADAS

```
SELECT index_id, partition_number,rows FROM sys.partitions
WHERE object_id= OBJECT_ID('DWH.FAC_INSCRIPCION_GRADO')
ORDER BY index_id,partition_number;
```

3.4.3.3. CALIDAD DE SCRIPT SQL

Es importante analizar la calidad del SCRIPT SQL ya que si se tienen sentencias bien elaboradas es notable la mejora del rendimiento en el tiempo de respuesta al ejecutar las consultas.

Para estudiar el concepto de calidad de script es importante conocer los posibles elementos que forman parte del script.

3.4.3.3.1. USANDO JOIN

Para optimizar las consultas, una de las estrategias básicas es reducir al mínimo el número de JOIN a las cláusulas utilizadas. Otra consideración es que los JOINS externos incurren en más costos que los INNER JOINS por el trabajo extra necesario para encontrar las filas no coincidentes. Si sólo INNER JOINS se utilizan en una consulta, el comportamiento de las cláusulas ON y WHERE es el mismo, no importa si se pone una expresión en ON o en la cláusula WHERE. Además es importante ordenar los JOIN cuando existen tablas con distinta cantidad de datos, es preciso primero ir colocando las tablas que contengan la mayor cantidad de datos. [1]

3.4.3.3.2. USANDO FUNCIONES AGREGADAS DE SQL SERVER

Existen varias funciones agregadas que SQL SERVER posee para facilitar la elaboración de consultas, es importante saber que es preciso utilizar estas funciones ya que permiten mejorar el rendimiento durante la ejecución de las consultas. Como cuando queremos obtener columnas que se encuentren en un rango, se puede utilizar signos como < >, pero es mejor utilizar la función BETWEEN, este es uno de los ejemplos pero existen varias funciones, siendo las más utilizadas AVG, SUM, COUNT, ROW_NUMBER.

3.4.3.3.3. SUBCONSULTAS

Una subconsulta correlacionada se ejecuta sólo una vez a la ejecución de la consulta y devuelve un solo valor. Estas consultas suelen incurrir en pocos gastos. Subconsultas correlacionadas que incluyen una referencia a la consulta externa, normalmente, esta referencia se utiliza para filtrar la subconsulta correlacionada. Una subconsulta correlacionada es generalmente igual en rendimiento en comparación a la utilización de un JOIN cuando se utiliza en combinación con el operador EXISTS para filtrar la consulta externa.

Debido a que cada subconsulta se ejecuta por separado, esta podría terminar con diferentes índices. Esto significa que estas consultas no pueden devolver los valores de la misma fila (que probablemente están destinados a devolver).

Hay varias maneras de escribir esta consulta, la más común en SQL Server 2008 es, probablemente, utilizar la nueva cláusula de APPLY. Si la subconsulta se utiliza en el FROM, JOIN, o cláusulas APPLY, también puede ser contemplado como una tabla derivada. La cláusula APPLY básicamente le da la oportunidad de combinar dos subconsultas en una sola, reduciendo el número de ejecuciones de la subconsulta a la mitad. Un OUTER APPLY funciona de manera similar a la de su combinación externa izquierda, y la contraparte, CROSS APPLY, se comporta como un INNER JOIN.

3.4.3.3.4. ESCALAR UDF

Un UDF escalar es una función que devuelve un solo valor (no es un conjunto de resultados). Este tipo de función se utiliza con frecuencia en las consultas y puede degradar significativamente el rendimiento. La razón de esto es que estas funciones no se expanden y optimizan en el plan de consulta principal por el optimizador, sino que se acaba de llamar desde el plan de ejecución sin ningún tipo de optimización basada en el contexto en el que se inserta en el plan.

En ocasiones es mejor realizar una subconsulta correlacionada en lugar de la UDF.

3.4.3.3.5. CURSORES

En general, se debería evitar el uso de cursores, debido a los efectos negativos en el rendimiento.

3.4.3.3.6. UTILIZACIÓN DE ARGUMENTOS DE LA BÚSQUEDA

Un argumento de búsqueda (SARG) es una expresión de filtro que se utiliza para limitar el número de filas devueltas por una consulta y que se puede utilizar un índice de operación de búsqueda que mejora sustancialmente el rendimiento de la consulta.

3.4.3.3.7. ORDEN DE EJECUCION

En la siguiente tabla se muestra el orden de ejecución de los elementos en un script SQL

TABLA VIII

Orden de Ejecución de clausulas SQL

CLÁUSULAS DE RESULTADOS	
1. FROM, JOIN, APPLY, and ON	El JOIN se ejecuta y el primer filtro de la consulta (Cláusula ON) se aplica
2. WHERE	El segundo filtro de la consulta se aplica
3. GROUP BY y funciones agregadas como (SUM, AVG y así sucesivamente) están incluidas en la consulta	Agrupación y cálculos de agregación se realizan
4. HAVING	El tercer filtro de la consulta (filtrado de resultados o funciones de agregación) se aplican
5. SELECT	Las columnas que se deben retornar en la consulta son seleccionadas
6. ORDER BY	Los resultados son ordenados
7. TOP	El cuarto (y el ultimo) filtro de la consulta se aplica; esto hace que la consulta devuelva sólo los primeras X filas de los resultados hasta el momento
8. FOR XML	El resultado tabular devuelto por la instrucción SELECT se convierte en lenguaje de marcado extensible (XML).

Fuente: Microsoft SQL Server 2008-Database Development.

TABLA IX

Tabla resumen de criterios y parámetros de análisis

CRITERIOS	TÉCNICAS	PARÁMETROS		ESCENARIO	
DBMS (servidor de BD-Software)	ANALISIS DEL GESTOR DE BASE DE DATOS	OPTIMIZACIÓN	BI	MYSQL 5.1 (MyISAM) (SW LIBRE)	SQL SERVER 2005 STANDART (PROPIETARIO)
DISEÑO DEL DATA WAREHOUSE	DESNORMALIZACION	TIEMPO DE RESPUESTA	CANTIDAD DE DATOS	OLTP	OLAP
	CREACION DE FILEGROUPS	TIEMPO DE RESPUESTA	CANTIDAD DE DATOS	OLAP(sin aplicar técnicas)	OLAP(aplicando técnicas)
CONSULTAS	INDICES	TIEMPO DE RESPUESTA	CANTIDAD DE DATOS	OLAP(sin aplicar técnicas)	OLAP(aplicando técnicas)
	PARTICIONAMIENTO DE TABLAS	TIEMPO DE RESPUESTA	CANTIDAD DE DATOS	OLAP(sin aplicar técnicas)	OLAP(aplicando técnicas)
	CALIDAD DE SCRIPT SQL	TIEMPO DE RESPUESTA	CANTIDAD DE DATOS	OLAP(sin aplicar técnicas)	OLAP(aplicando técnicas)

Elaborado por: Autora

3.4.4. ESTADÍSTICAS SQL SERVER

Las estadísticas de SQL Server son información sobre la distribución de los datos existentes en las columnas de las tablas de la base de datos. A través de las estadísticas, el servidor conoce como es la información de una columna, como por ejemplo, si varía mucho, si todos los datos son iguales y los niveles de variación que hay. Estas le permiten al servidor "conocer" los datos de las columnas sin necesidad de tener que leerlas a menudo. Realmente el no conoce todos los datos, pero con la información que obtiene le basta para tomar buenas decisiones.

Esta información la utiliza cuando se solicita datos de ciertas tablas que cumplan con ciertas condiciones (select…from...where). Las condiciones que se especifican en el where de una consulta son analizadas por el optimizador de consultas para determinar cuál es la forma más rápida de obtener la información requerida. Para lograr esto, suponiendo que se tiene una consulta con varias condiciones en el where, el servidor examina las estadísticas asociadas a las columnas referenciadas en el where, como también los índices existentes en las tablas y donde participan esas columnas. Para el caso de los índices, SQL SERVER maneja un conjunto de estadísticas de forma similar que para una columna de una tabla, independiente del tipo de índice (agrupado o no agrupado).

En el caso en que la consulta tiene una sola condición, no hay muchas soluciones posibles. Si hay un índice sobre la columna que se está buscando, en la mayoría de los casos lo usará (dependerá de las estadísticas y de otros factores) y en caso contrario, realizará un SCAN sobre la tabla o el índice agrupado (en caso de tener).

Existen dos formas tradicionales de ver las estadísticas. Una de ellas es consultando el catálogo interno de SQL Server o la otra es a través de la interfaz gráfica asociada al plan de ejecución de una consulta. Antes de ver como se hace, se debe conocer como se consultan los catálogos internos. Algunas de las tablas a las que se hace referencia en las consultas no necesariamente existen físicamente y muchas de ellas, o son

vistas solamente o se construyen al momento de ejecutar. Además, los objetos (tablas, procedimientos, etc.) rara vez están almacenados por el nombre que se le da, sino que por un identificador interno. Para obtener el identificador interno de un objeto, existe una función llamada object_id('objeto') que lo retorna, pudiendo usarse en una consulta o una instrucción set. El catálogo donde se almacena la información de los índices y estadísticas de una tabla se llama sysindexes, este puede ser consultado con un simple select.

Cuando se realizan consultas al servidor las estadísticas se crean automáticamente. Se puede mostrar la estadística con el comando:

```
dbcc showstatistics (tabla, indice|estadística)
```

Toda esta información le permite saber al optimizador de consultas como es la información de la columna o índice, sin necesidad de "tocar" los datos. Y por el mismo motivo, si se quiere que el analizador siempre encuentre la mejor opción y el servidor responda al máximo, se debe proveer a éste de las estadísticas actualizadas.

Se puede mencionar que las estadísticas pueden actualizarse o eliminarse manualmente a través de la interfaz gráfica o de consultas SQL (drop statistics). Además, conviene saber que SQL Server se encarga de actualizarlas y eliminarlas cuando estima que es necesario, pero también puede agregarse una tarea de mantención que las actualice cada cierto tiempo.

3.4.4.1. ACTUALIZACIÓN DE ESTADÍSTICAS

Previamente se mencionó que SQL SERVER actualiza las estadísticas automáticamente, pero ¿cuándo sabe que tiene que actualizarlas?. Como no existen procesos mágicos, la única forma es darse cuenta de que la estadística ya no refleja la realidad y es porque los registros han sido modificados, y para eso el servidor lleva la

cuenta de cuantos registros se han modificado, insertado y eliminado. Es posible que haya nuevos datos y que se generen nuevos rangos (RANGE_HI_KEY), como también más valores repetidos (EQ_ROWS). Puede incluso que la densidad de la estadística sea menor y ahora si sea útil para las búsquedas.

Para actualizar las estadísticas manualmente se pueden ejecutar cualquiera de estas dos opciones:

```
update statistics
sp_updatestats
```

La diferencia principal entre estos métodos es que con el primero se puede controlar que tabla se quiere actualizar, y si se quiere para todos sus índices y estadísticas o para alguno específico, o para todos los índices o todas las estadísticas de la tabla. Además se pueden definir otros parámetros como la cantidad de filas de la tabla a revisar (numero, porcentaje, etc.) como también si se quiere utilizar el mismo valor que se utilizó la primera vez. Por otra parte, el procedimiento almacenado actualiza las estadísticas de todas las tablas de la base de datos que tengan cambios.

Para la creación de estadísticas se pueden utilizar las instrucciones

```
create statistics
sp_createstats
```

El control de create statistics es completo comparado con sp_createstats, pero este último tiene la ventaja de que crea estadísticas en todas las columnas de todas las tablas que poseen las siguientes características:

- No son columnas computadas (formulas)
- No son de tipo text, ntext e image
- No poseen estadísticas
- No son la primera columna de un índice.

Con estas instrucciones se puede controlar de forma completa las estadísticas de SQL Server. [14]

CAPÍTULO IV: DESARROLLO DE TÉCNICAS DE TUNING EN EL DATA WAREHOUSE DEL PROYECTO SII-ESPOCH

4.1. INTRODUCCIÓN

Para la realización de las pruebas, es importante describir el escenario sobre los que varias técnicas tuning (bajo diferentes criterios), probarán la optimización de rendimiento en el data warehouse del SII-ESPOCH. Es importante tomar en cuenta de que al hablar de procesamiento OLAP, es decir de una solución utilizada en el campo de la Inteligencia empresarial (o Business Intelligence) y en aplicaciones de toma de decisiones, el objetivo primordial es agilizar las consultas de grandes cantidades de datos, ya que la razón de usar OLAP para las consultas es la velocidad de respuesta. Es por eso que la principal característica que potencia a OLAP, es que es lo más rápido a la hora de ejecutar sentencias SQL de tipo SELECT, en contraposición con OLTP que es la mejor opción para operaciones de tipo INSERT, UPDATE Y DELETE. Por tal motivo las sentencias SQL que se realizarán en las pruebas son **SELECT**.

4.2. DESCRIPCIÓN DEL ESCENARIO HARDWARE

El escenario sobre el que se realizarán las pruebas consta de:

- Dos servidores virtuales con las siguientes características:

TABLA X

Características de los servidores

CARACTERISTICAS DE LOS SERVIDORES	
SERVIDOR	SERVIDOR BLADE HP EVA 4400, montados sobre un chasis HP C7000
PROCESADORES	Intel(R) Xeon (R) CPU X5460 @3.16GHz, 3.17GHz (2 procesadores)
RAM	8 GB
DISCO DURO	120 GB
SISTEMA OPERATIVO	WINDOWS SERVER 2008 ENTERPRISE SP 2 de 32 bits

Elaborado por: Autora

Estos servidores alojan:

- El primero almacena el DATA WAREHOUSE SII-ESPOCH, en SQL SERVER 2005, con la dirección IP:712.30.60.111
- El segundo servidor aloja la aplicación Business Objects (B.O.), con la dirección IP:712.30.60.112

Además es importante mencionar que el acceso se realiza desde la Unidad Técnica de Planificación, mediante la conexión de escritorio remoto desde los equipos de esta

Unidad, este acceso se realiza para la carga, transformación y manipulación de la información, igualmente para la implementación de técnicas tuning y su mejoramiento de rendimiento, permitiendo de esta manera el acceso directo sin tener que dirigirse al lugar donde se encuentran físicamente los servidores.

- Los equipos desde los que se realiza la conexión de escritorio remoto (PC de escritorio, laptop), tienen las siguientes características:

TABLA XI

Características de la PC de escritorio

CARACTERÍSTICAS DE LA PC DE ESCRITORIO	
PROCESADOR	Intel Core 2 Quad 3 GHz
RAM	3 GB
DISCO DURO	400 GB
SISTEMA OPERATIVO	Windows XP de 32 bits
IP	172.30.10.104

Elaborado por: Autora

TABLA XII

Características de la laptop HP

CARACTERÍSTICAS DE LA LAPTOP HP	
PROCESADOR	INTEL CORE i5 2.40GHz
RAM	4 GB
DISCO DURO	500 GB
SISTEMA OPERATIVO	Windows 7 Ultimate de 64 bits
IP	172.30.10.114

Elaborado por: Autora

En la siguiente figura se muestra de mejor manera el escenario utilizado.

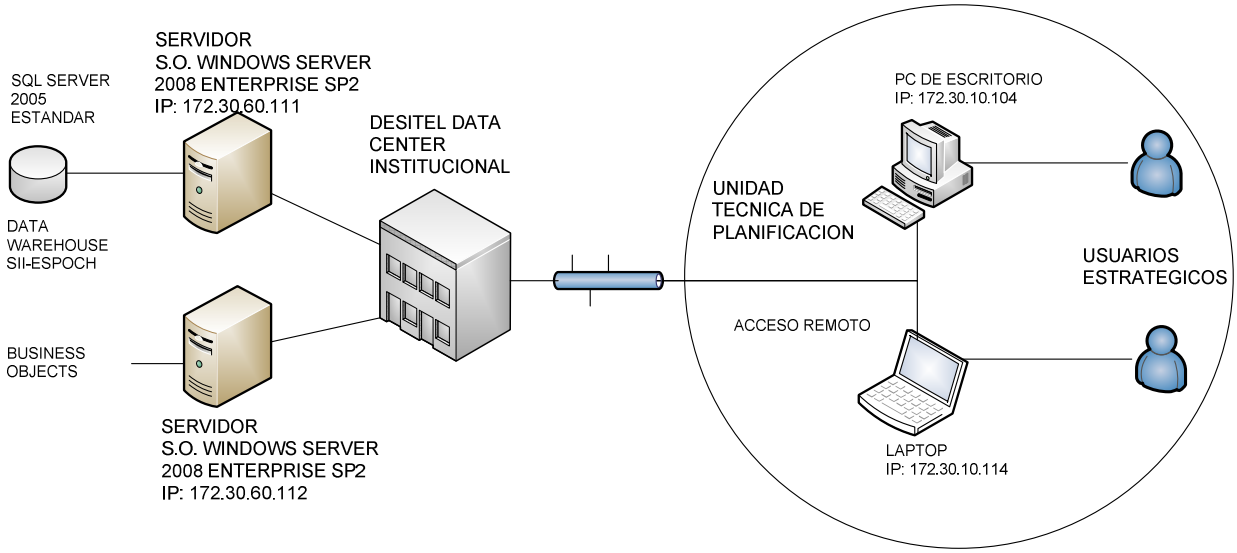


FIGURA IV. 21: Descripción del escenario utilizado para las pruebas

Elaborado por: Autora

4.2.1. JUSTIFICACIÓN DEL ESCENARIO

Es importante justificar el escenario sobre el que se realizarán todas las pruebas, de esta manera se respaldan las decisiones tomadas a nivel de hardware (servidores).

Para comprobar que el escenario elegido es apto para la realización de las pruebas, se realizan algunas pruebas en una laptop con las siguientes características:

- Procesador Intel CORE i5
- Memoria RAM de 4GB

- Almacenamiento en disco de 500 GB

Mediante la ejecución de una sentencia SQL con 1.000.000 de datos se obtienen las siguientes evidencias.

4.2.1.1. EVIDENCIAS

Se utiliza una sentencia SQL de prueba, la misma que en el servidor se ejecuta sin problema, la laptop de prueba emite el siguiente mensaje antes de concluir con la ejecución de la sentencia.

SENTENCIA SQL

```
select  DISTINCT  B.ID_ESTUDIANTE

    ,D.ID_UBICACION_GEOGRAFICA

    ,cast(COALESCE(A.dtFecha,'01/01/1900') as int) AS ID_FECHA

    ,0 AS ID_TITULO_ESPOCH

    ,CASE WHEN E.strCodInt IS NOT NULL THEN F.ID_INSTITUCION

        ELSE 0 END AS ID_INSTITUCION

    ,CASE WHEN A.blnConfirmada=1 THEN 1

       ELSE 2 END AS ID_ESTADO_ESTUDIANTE

    ,G.ID_PERACAD

    ,CASE WHEN (E.strCodTit IS NOT NULL AND H.ID_TITULO_INSTITUCION IS NOT NULL)  THEN
H.ID_TITULO_INSTITUCION

        WHEN (E.strCodTit IS NOT NULL AND H.ID_TITULO_INSTITUCION IS NULL) THEN
J.ID_TITULO_ESPOCH

        ELSE 0 END AS ID_TITULO_INSTITUCION

   ,G.ID_UBICACION_ACADEMICA  AS ID_UBICACION_ACADEMICA --
,I.ID_UBICACION_ACADEMICA

    ,0 AS VALOR

    ,1 AS CANTIDAD

    ,GETDATE() AS ETL_FECHA_CARGA

from   DSA.Oasis_Inscripciones A INNER

  JOIN  DWH.DIM_ESTUDIANTE B    ON A.strCedEstud=B.CI

  JOIN DWH.DIM_UBICACION_ACADEMICA I ON I.COD_CARRERA_PROGRAMA=A.strCodCarrera

  JOIN DSA.Oasis_Estudiantes C ON A.strCedEstud=C.strCedula  AND C.strCedula=B.CI

  JOIN DWH.DIM_UBICACION_GEOGRAFICA D ON
LTRIM(RTRIM(C.strCodCiudadProc))=D.COD_CANTON

  JOIN DWH.DIM_PERIODO_ACADEMICO G ON G.COD_PERIODO=A.strCodPeriodo AND
G.COD_CARRERA_PROGRAMA=A.strCodCarrera

  LEFT JOIN DSA.Oasis_Grados E ON A.strCedEstud=E.strCedEstud

  LEFT JOIN DWH.DIM_INSTITUCION F ON E.strCodInt=F.CODIGO_INSTITUCION

  LEFT JOIN DWH.DIM_TITULO_INSTITUCION H ON H.CODIGO_TITULO=E.strCodTit

  LEFT JOIN DWH.DIM_TITULO_ESPOCH J ON J.CODIGO_TITULO=E.strCodTit
```

Durante la ejecución de la consulta aparece el icono de mensaje de advertencia, mostrado en la siguiente figura:

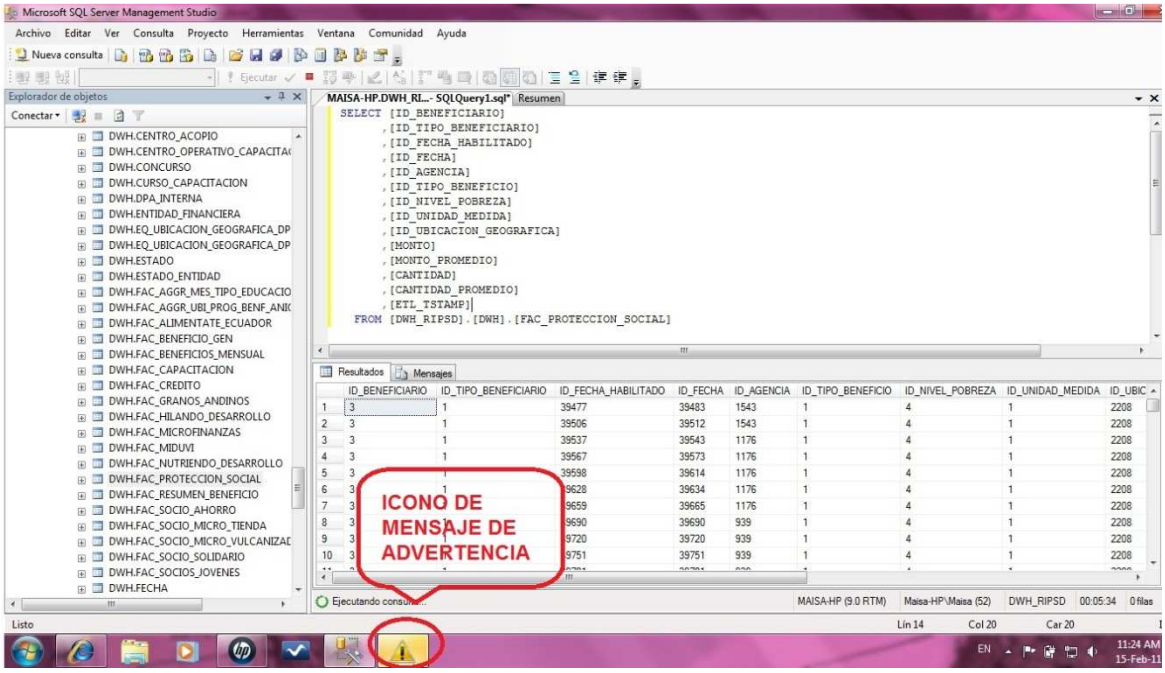


FIGURA IV. 22: Mensaje de advertencia al momento de la ejecución de la consulta

Esta sentencia SQL devuelve 73914 filas en el servidor, mientras que en el otro equipo (laptop) emite un mensaje de advertencia.

MENSAJES DE ADVERTENCIA

Al ejecutar la consulta aparecen las siguientes advertencias, debido a la falta de memoria durante la ejecución de la consulta.

FIGURA IV. 23: Mensaje de advertencia, al encontrar falta de memoria durante la ejecución de la consulta

El equipo notifica memoria insuficiente y para poder continuar con el funcionamiento normal de este es necesario reiniciarlo, cabe indicar que el equipo no puede generar la consulta SQL, debido a la falta de hardware.

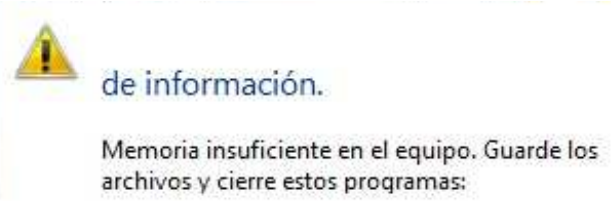


FIGURA IV. 24: Mensaje de advertencia por memoria insuficiente

De esta manera se evidencia que el escenario sobre el que se realizarán las pruebas (SERVIDOR) es apto para alojar el DWH_SIIESPOCH.

4.3. HERRAMIENTAS UTILIZADAS EN EL ESCENARIO

Para poder medir el rendimiento en las pruebas realizadas, es preciso prescindir de los **planes de ejecución reales** de SQL SERVER, estos planes se basan en el costo. Cada plan de ejecución posible tiene asociado un costo en términos de la cantidad de recursos del equipo que se utilizan. Se deben analizar los planes posibles y elegir el de menor costo estimado. La estimación se basa en las estadísticas de distribución de datos que están disponibles cuando se evalúa cada tabla implicada en la consulta.

Frecuentemente se asume que el costo estimado de ejecución es un buen indicador de cuánto tiempo tardará la consulta en ejecutarse y que esta estimación permite distinguir los planes buenos de los que no lo son.

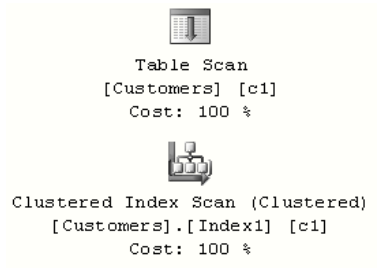
Puesto que se trata de una estimación, podría ser errado, en ocasiones los planes con costos estimados más altos pueden ser mucho más efectivos en términos de CPU, E/S y del tiempo de ejecución, a pesar de que sus estimaciones sean más altas. Algunas instrucciones SELECT complejas tienen miles de planes de ejecución posibles, por lo que se debe encontrar un plan de ejecución “lo suficientemente bueno”, que devuelva resultados al usuario con un costo razonable de recursos y que devuelva los resultados lo más rápido posible.

ITEMS QUE PERMITEN NOTAR LA NECESIDAD DE OPTIMIZACIÓN

En la siguiente tabla se muestran los ítems que al existir en un plan de ejecución, permitirán analizar la utilización de técnicas tuning como la utilización de índices, estos además de aparecer en el plan de ejecución, tendrán altos porcentajes.[1]

TABLA XIII

Ítems presentes en los planes de ejecución

ITEMS CON POSIBLES IMPLICACIONES	
Thick arrows 	Una flecha gruesa representa un gran número de filas en movimiento de una operación a otra en la ejecución del plan. Mientras mayor sea el número de filas a transferirse de una operación a otra, más gruesa va a ser la flecha.
Hash operations 	Si una operación de hash se utiliza para controlar las cláusulas como GROUP BY y JOIN, significa a menudo que no existe un índice apropiado para optimizar la consulta.
Sorts 	Un Sort no es necesariamente malo, pero si tiene un alto porcentaje del costo de la consulta, se debe considerar si un índice puede ser construido para eliminar la necesidad de la operación de ordenación.
Table or clustered index scans 	Un índice agrupado (Clustered index) y un recorrido de tabla (Tabla scan) indican que no hay índice adecuado que se utiliza para optimizar la consulta.
Large plans	El plan con menor número de operaciones suele ser el mejor plan optimizado.

Fuente: Microsoft SQL Server 2008-Database Development.

4.4. DESCRIPCIÓN DE LA RED

El escenario real sobre el que esta ejecutándose el proyecto SII-ESPOCH, tiene las siguientes características de red:

- Sobre el Departamento de Sistemas y Telemática se encuentran los servidores que almacenan el Data Warehouse y Business Objects.
- A continuación se encuentra el Switch de Distribución CISCO 3560g, IP de VLAN 172.30.80.65, con puertos GigaEthernet Trunk 28 y 27, que se conectan con un Switch de Acceso CISCO 2960 24 tc-s (puerto GigaEthernet Trunk 1) con IP 172.30.80.66, mediante el cual se permite el acceso de las máquinas de la Unidad Técnica de Planificación a los servidores ubicados en el DESITEL.

En la siguiente figura se muestra de mejor manera el escenario:

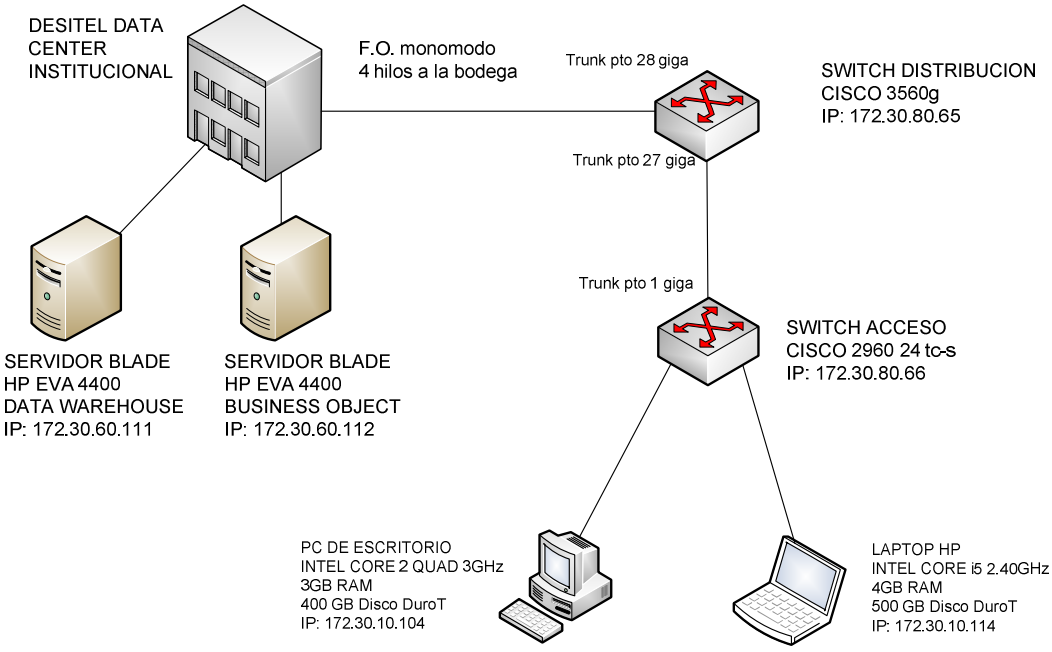


FIGURA IV. 25: Descripción de la red del escenario

Elaborado por: Autora

4.5. COMPARACIÓN DEL DBMS (servidor de BD-Software)

Este criterio de DBMS es un punto importante sobre el que se trabaja, ya que es primordial justificar la utilización del actual gestor de base de datos, para lo cual se realizará una comparación con un DBMS Open Source, basados bajo diferentes lineamientos de análisis, siendo la optimización y Business Inteligence las principales características que se van a comparar.

4.5.1. AMBIENTE DE PRUEBA #1

Los DBMS que se van a comparar son SQL SERVER 2005 (Propietario) con MYSQL 5.1 (MYISAM) (Open Source) el motor de almacenamiento sobre el que se establecen las comparaciones es MYISAM, debido a que es el motor de almacenamiento que viene por defecto en MYSQL.

4.5.2. OBJETIVO DEL AMBIENTE DE PRUEBA #1

Comparar los DBMS Propietario SQL SERVER 2005 y DBMS Open Source MYSQL 5.1

4.5.3. ¿CÓMO SE VA A MEDIR?

Los criterios de medida que se van a realizar sobre esta prueba van a tener un valor sobre 100, como se muestra en la siguiente tabla:

TABLA XIV

Criterios de medida

CARACTERÍSTICAS	CALIFICACIÓN SOBRE 100
LICENCIAMIENTO	6
MULTIPLATAFORMA	2
SOPORTE	2
ACID	2
INTEGRIDAD REFERENCIAL	2
CONSULTAS	12
TABLAS TEMPORALES	3
VISTAS	2
CURSORES	2
TRIGGER	2
FUNCIONES	2
PROCEDIMIENTOS	2
INDICES	10
PARTICIONAMIENTO DE TABLAS	8
REPLICACIÓN	2
SEGURIDAD	4
ESTABILIDAD	5
HERRAMIENTAS GUI DE ADMINISTRACIÓN	3
ASESOR DE OPTIMIZACION DE BASE DE DATOS	4
IMPORTACION / EXPORTACION	2
DATA WAREHOUSING	4
SERIVCIOS PARA CARGAS ETLs:	6
MINERIA DE DATOS:	4
HERRAMIENTAS DE BUSINESS INTELLIGENCE	4
SERIVCIOS PARA PROCESAMIENTO OLAP	5
TOTAL	**100**

Elaborado por: Autora

4.5.4. EXPERIMENTO

En la siguiente tabla se realiza la comparación entre el DBMS Propietario y Open Source, de acuerdo a las características definidas.

TABLA XV

Comparación de SQL SERVER 2005 CON MYSQL 5.1 (MYISAM)

CARACTERÍSTICAS	SQL SERVER 2005	MYSQL 5.1(MYISAM)
LICENCIAMIENTO	LICENCIADO	LICENCIA GPL
MULTIPLATAFORMA	NO. SÓLO WINDOWS Y TODAS SUS VERSIONES	SI. MICROSOFT WINDOWS, UNIX, GNU LINUX, SOLARIS
SOPORTE	SI	SI
ACID	SI	SI
INTEGRIDAD REFERENCIAL	SI	SI
CONSULTAS	SI	SÓLO CLAVES PRIMARIAS
TABLAS TEMPORALES	SI	SI
VISTAS	SI	Vistas actualizables
CURSORES	SI	SI
TRIGGER	SI	Rudimentarias
FUNCIONES	SI	SI
PROCEDIMIENTOS	SI	SI
INDICES	Árbol-B	Árbol R-/R+
PARTICIONAMIENTO DE TABLAS	NO	NO
REPLICACIÓN	Instantáneas, transaccional, mezcla (Snapshot, Transactional, Merge)	De un sólo sentido (One-way only)
SEGURIDAD	Alta	Alta
ESTABILIDAD	Alta	Baja

HERRAMIENTAS GUI DE ADMINISTRACIÓN	SI	SI
ASESOR DE OPTIMIZACION DE BASE DE DATOS	SI	NO
IMPORTACION / EXPORTACION	SI	SI
DATA WAREHOUSING	SI	NO
SERIVCIOS PARA CARGAS ETLs:	SI	NO
MINERIA DE DATOS	SI	NO
HERRAMIENTAS DE BUSINESS INTELLIGENCE	SI	NO
SERIVCIOS PARA PROCESAMIENTO OLAP	SI	NO

Elaborado por: Autora

A continuación se van a describir las características de los DBMS analizados.

4.5.4.1.1. SQL SERVER 2005

SQL Server 2005 es un sistema gestor de datos que analiza y administra datos haciendo posible avances en tres ámbitos clave: gestión de información empresarial, productividad de los desarrolladores e inteligencia empresarial. Microsoft SQL Server 2005 es una plataforma de base de datos para transacciones en línea (OLTP) a gran escala, bodegas de datos (data warehousing), y aplicaciones de comercio electrónico; a su vez es un plataforma de inteligencia de negocios con integración de datos, análisis, y soluciones de reporteo.[23]

En la siguiente tabla se describen las características que posee SQL SERVER 2005.

TABLA XVI

Tabla de descripción de características de SQL SERVER 2005

SQL SERVER 2005	
CARACTERÍSTICAS	**DESCRIPCIÓN**
LICENCIAMIENTO	LICENCIADO. Software propietario con tres opciones de licenciamiento: • Licencia de servidor más una licencia de acceso de cliente (CAL) por dispositivo • Licencia de servidor más una licencia de acceso de cliente (CAL) por usuario • Licencia por procesador [23]
MULTIPLATAFORMA	NO. SQL SERVER se ejecutan sólo en la plataforma Microsoft, en todas las versiones.
SOPORTE	SI. Este soporte está disponible automáticamente al comprar la licencia de uso. Dicho servicio le da un respaldo al usuario, de cara a la solución de posibles problemas que pudieran presentarse en el uso del producto adquirido.
ACID	SI. Este cumplimiento asegura la integridad de datos. SQL Server 2005 soporta la funcionalidad de la transacción, cumple las reglas ACID. [23]
INTEGRIDAD REFERENCIAL	SI. SQL SERVER tiene integridad referencial que garantiza que una entidad (fila o registro) siempre se relacione con otras entidades válidas, es decir, que existen en la base de datos. Implica que en todo momento dichos datos sean correctos, sin repeticiones innecesarias, datos perdidos y relaciones mal resueltas.
CONSULTAS	SI. Maneja claves primarias y foráneas en las tablas.
	SI. Las tablas temporales en SQL SERVER se crean en tempdb, existen dos tipos de tablas temporales en cuanto al

TABLAS TEMPORALES	alcance de la tabla • **#locales** • **##globales** [11]
VISTAS	SI. SQL SERVER puede cargar las vistas tradicionales como se definieron en la norma ANSI de SQL, existen también vistas indexadas (materializadas), pero la versión Enterprise.
CURSORES	SI. SQL SERVER cuenta con cursores, permitiendo la navegación entre la información.
TRIGGER	SI. Los desencadenadores son procedimientos Transact-SQL que se ejecutan automáticamente cuando se envía una instrucción update, insert o delete a una tabla base o vista. Hay dos clases generales de desencadenadores que difieren en el tiempo con respecto a la instrucción de desencadenamiento, bajo la que se realiza la acción. • Los desencadenadores AFTER • Los desencadenadores INSTEAD
FUNCIONES	SI. SQL SERVER facilita la creación sentencias mediante funciones del usuario.
PROCEDIMIENTOS	SI. SQL SERVER mediante el manejo de procedimientos hace posible una administración más fácil.
INDICES	Árbol-B. La forma más común de indexación son los índices no agrupados y agrupados que ofrece SQL SERVER**.**
PARTICIONAMIENTO DE TABLAS	NO. No existe particionamiento de tablas en SQL SERVER ESTANDAR, En SQL SERVER 2005 Enterprise, las páginas de tablas e índices están incluidas en una o varias particiones, estas particiones son horizontales y por rangos.
REPLICACIÓN	SI. SQL Server 2005 proporciona una serie de funcionalidades de replicación extremo a extremo, incluyendo la posibilidad de publicar bases de datos Oracle. SQL Server 2005 incluirá nuevas mejoras a las herramientas y sobre la escalabilidad de la replicación también. • Instantáneas (Snapshot) • Transaccional (Transactional) • Mezcla (Merge) [23]

SEGURIDAD	Alta. SQL SERVER cuenta con mecanismos de seguridad por defecto perfectamente aceptables, siempre y cuando siga las instrucciones del fabricante y mantenerse al día con los parches de seguridad. SQL SERVER funciona por defecto con puertos IP y le permiten cambiar ese puerto en caso de algún problema con virus.
ESTABILIDAD	Alta. SQL Server, es mucho más resistente a la corrupción de datos. Datos de SQL Server pasan por varios puntos de control y SQL Server se acuerda del proceso, incluso si pasa a ser cerrado sin previo aviso.
HERRAMIENTAS GUI DE ADMINISTRACIÓN	SI. SQL SERVER contiene herramientas gráficas de interfaz de usuario para facilitar la administración del DBMS, permitiéndole al usuario una administración más cómoda.
ASESOR DE OPTIMIZACION DE BASE DE DATOS	SI. En SQL SERVER existen indicadores de Rendimiento Principales ("Key Performance Indicators" , KPI) que permiten definir métricas de negocio en formato gráfico, adaptables, para ayudar a generar y hacer el seguimiento de entornos de prueba corporativos y además consejos para mejorar el rendimiento del Data Warehouse. [11]
IMPORTACION / EXPORTACION	SI. SQL SERVER permite importar y exportar datos, estas son tareas que facilitan al usuario el uso de información dentro y fuera de la base y almacén de datos.
DATA WAREHOUSING	SI. SQL SERVER 2005 introduce una nueva arquitectura de Servicios de Transformación de Datos (DTS). La arquitectura consiste en dos motores: El Runtime de Transformación de Datos (DTR). Este motor ejecuta paquetes DTS, tareas DTS, hace seguimiento de la ejecución del paquete y proporciona servicios para las tareas. El motor DTP extrae datos de las fuentes, aplica las transformaciones contra las columnas de datos y carga los datos en los sistemas de almacenamiento.
SERIVCIOS PARA CARGAS ETLs:	SI. Los Servicios de Integración pueden usarse para extraer, transformar y cargar datos (ETL) desde fuentes muy diversas y llevarlas a un destino único o múltiples destinos. [11]
MINERIA DE DATOS:	SI. SQL Server 2005 introduce cuatro nuevos algoritmos de Data Mining, así como herramientas y asistentes mejorados, haciendo que el data mining sea más accesible a negocios de cualquier tamaño.
HERRAMIENTAS DE BUSINESS INTELLIGENCE	SI. SQL SERVER ofrece servicios de Reporting que permite a los negocios integrar de forma sencilla datos desde fuentes heterogéneas y data warehouses en informes ricos, interactivos y gestionables, que pueden localizarse y consultarse en intranets, extranets y en Internet.
SERIVCIOS PARA PROCESAMIENTO OLAP	SI. Los servicios de Análisis con SQL Server 2005 (Analysis Services) se mueven en el entorno del análisis en tiempo real. Desde mejoras en la escalabilidad hasta una integración profunda con Microsoft Office, SQL SERVER 2005 amplía el concepto de "business intelligence" a todos los niveles de su negocio.

Elaborado por: Autora

4.5.4.1.2. MYSQL 5.1 (MYISAM)

MYSQL es una base de datos robusta que puede ser comparada con una base de datos comercial, compite con sistemas RDBMS propietarios como Oracle, SQL Server y DB2, disponiendo de procesamiento de transacciones a través del motor de almacenamiento MYISAM.

MYSQL es lo suficientemente flexible para trabajar en entornos con gran demanda, tales como aplicaciones web; al mismo tiempo, puede impulsar aplicaciones empotradas, almacenes de datos, indexación de contenidos, sistemas de mensajería, sistemas redundantes de alta disponibilidad, procesamiento de transacciones en línea (OLTP), y mucho más.

En la siguiente tabla se describen las características que posee MYSQL 5.1 (MYISAM).

TABLA XVII

Tabla de descripción de características de MYSQL 5.1(MYISAM)

MYSQL 5.1 (MYISAM)	
CARACTERÍSTICAS	**DESCRIPCIÓN**
LICENCIAMIENTO	MYSQL es libre bajo licencia GPL, es decir, el usuario tiene derecho a usar el programa, modificarlo y distribuir las versiones modificadas pero no tiene permiso de realizar restricciones propias con respecto a la utilización de ese programa modificado. Las versiones completas son pagadas. [11]
MULTIPLATAFORMA	SI. MYSQL está disponible para las siguientes plataformas: • Linux • Unix • Microsoft Windows • Solaris • OS X • IBM
SOPORTE	SI. El soporte en Open Source es más costoso que la del software Propietario debido que en el pago de la licencia este rubro está incluido, en cambio en software libre es más difícil encontrar ayuda porque no siempre hay suficientes personas capacitadas a parte de su costo.
ACID	SI. MySQL maneja operaciones de seguridad (conforme a ACID) con motor de almacenamiento de confirmar, deshacer y recuperación de fallas. [23]
INTEGRIDAD	SI. MYSQL tiene integridad referencial y garantiza que una entidad (fila o registro) siempre se relacione con otras

REFERENCIAL	entidades válidas, es decir, que existen en la base de datos. Implica que en todo momento dichos datos sean correctos, sin repeticiones innecesarias, datos perdidos y relaciones mal resueltas.
CONSULTAS	MYSQL maneja sólo claves primarias.
TABLAS TEMPORALES	SI. En MYSQL existen tablas temporales TEMPORARY TABLES, pero es necesario tener el permiso CREATE para crear tablas temporales.
VISTAS	Las vistas (incluyendo vistas actualizables) fueron introducidas en la versión 5.0 del servidor de base de datos MYSQL. MYSQL no soporta vistas materializables sin embargo, pueden llegar a ser emuladas con software específico para esta tarea.
CURSORES	SI. En versiones anteriores de MYSQL no existían, pero se han ido incorporando en esta versión.
TRIGGER	Rudimentarias. Los desencadenantes en MYSQL son rudimentarios debido a que al ser implementados a partir de MYSQL 5.0.2 que se incorporó el soporte básico para disparadores (triggers). [23]
FUNCIONES	SI. MYSQL permite el desarrollo de funciones, facilitando al usuario la administración.
PROCEDIMIENTOS	SI. Los procedimientos almacenados y funciones son nuevas funcionalidades a partir de la versión de MySQL 5.0.
INDICES	Árbol R-/R+. MYSQL crea otra estructura de datos (un índice) que usa para almacenar información extra acerca de los valores en la columna indexada. Los valores indexados son llamados frecuentemente claves. MYISAM maneja sus índices en arboles R-/R+, además cada tabla puede agrupar hasta 32 índices. [23]
PARTICIONAMIENTO DE TABLAS	NO. No tiene particionamiento de tablas, claro que versiones Enterprise si soporta cinco tipos de datos estándar en la forma de partición: gama, ropa vieja, clave, lista, y compuesto particiones compuestos (sub-distrito). [11]
REPLICACIÓN	De un sólo sentido (One-way only). MYSQL soporta la replicación en un solo sentido. Un servidor actúa como maestro, mientras que uno o más servidores de los demás actúan como esclavos. El maestro escribe las actualizaciones de sus ficheros de log binario y los servidores esclavos conectan periódicamente para obtener actualizaciones.
SEGURIDAD	Alta. MYSQL cuenta con mecanismos de seguridad por defecto perfectamente aceptable, siempre y cuando siga las instrucciones del fabricante y mantenerse al día con los parches de seguridad.
ESTABILIDAD	Baja. La estabilidad es un área en MySQL, en su configuración MyISAM, se queda un poco corto. MyISAM supone un funcionamiento ininterrumpido. Si se cierra de forma inesperada, el resultado podría ser la corrupción de todos los datos**.**

HERRAMIENTAS GUI DE ADMINISTRACIÓN	SI. MYSQL cuenta con herramientas gráficas de usuario, permitiéndole al gestor facilitar la administración del DBMS.
ASESOR DE OPTIMIZACION DE BASE DE DATOS	NO. Hay que tomar en cuenta que solo la versión Enterprise tiene el MySQL Query Analyzer y son triales descargables.
IMPORTACION / EXPORTACION	SI. MYSQL puede importar y exportar datos entre las bases y almacenes de datos.
DATA WAREHOUSING	NO. No posee herramientas de Data Warehousing.
SERVICIOS PARA CARGAS ETLs:	NO. No posee servicios para cargas ETLs, generalmente se utilizan otros de software libre como Talend, Jaspersoft, Pentaho Kettle.[20]
MINERIA DE DATOS:	NO. No posee servicios para minería de datos.
HERRAMIENTAS DE BUSINESS INTELLIGENCE	NO. No contiene herramientas BI.
SERVICIOS PARA PROCESAMIENTO OLAP	NO. No tiene servicios para procesamiento OLAP.

Elaborado por: Auto

4.5.5. ANÁLISIS DE RESULTADOS DEL DBMS (servidor de BD-Software)

Para poder analizar los resultados obtenidos en el experimento es importante determinar los valores de la calificación de cada característica del DBMS. A continuación se van a definir los valores del experimento.

1. LICENCIAMIENTO

TABLA XVIII

Puntaje de Licenciamiento

PUNTAJE		
ALTO	**MEDIO**	**BAJO**
6	**3**	**1**
	MYSQL	SQL SERVER

Elaborado por: Autora

ALTO ≈ 6/6. Gratis

MEDIO ≈ 3/6. Licencia GPL

BAJO ≈ 1/6. Licenciado

2. MULTIPLATAFORMA

TABLA XIX

Puntaje de Multiplataforma

PUNTAJE	
ALTO	**BAJO**
2	**1**
MYSQL	SQL SERVER

Elaborado por: Autora

ALTO ≈ 2/2. Multiplataforma

BAJO ≈ 1/2. No multiplataforma

3. SOPORTE

TABLA XX

Puntaje de Soporte

PUNTAJE	
ALTO	**BAJO**
2	**1**
SQL SERVER	MYSQL

Elaborado por: Autora

ALTO ≈ 2/2. Soporte técnico inmediato gratuito

BAJO ≈ 1/2. Soporte técnico pagado

4. ACID

TABLA XXI

Puntaje de ACID

PUNTAJE	
ALTO	**BAJO**
2	**1**
SQL SERVER MYSQL	

Elaborado por: Autora

ALTO ≈ 2/2. Características ACID

BAJO ≈ 1/2. No posee características ACID

5. INTEGRIDAD REFERENCIAL

TABLA XXII

Puntaje de Integridad Referencial

PUNTAJE	
ALTO	**BAJO**
2	**1**
SQL SERVER MYSQL	

Elaborado por: Autora

ALTO ≈ 2/2. Tiene Integridad Referencial

BAJO ≈ 1/2. No tiene Integridad Referencial

6. CONSULTAS

TABLA XXIII

Puntaje de Consultas

PUNTAJE		
ALTO	**MEDIO**	**BAJO**
12	**6**	**1**
SQL SERVER	MYSQL	

Elaborado por: Autora

ALTO ≈ 12/12. Consultas que manejan claves primarias y foráneas

MEDIO ≈ 6/12. Consultas que manejan claves primarias

BAJO ≈ 1/12. Consultas que no manejan claves primarias y foráneas

7. TABLAS TEMPORALES

TABLA XXIV

Puntaje de Tablas Temporales

PUNTAJE	
ALTO	**BAJO**
3	**1**
SQL SERVER MYSQL	

Elaborado por: Autora

ALTO ≈ 3/3. Maneja tablas temporales

BAJO ≈ 1/3. No maneja tablas temporales

8. VISTAS

TABLA XXV

Puntaje de Vistas

PUNTAJE		
ALTO	**MEDIO**	**BAJO**
2	**1**	**0**
SQL SERVER	MYSQL	

Elaborado por: Autora

ALTO ≈ 2/2. Vistas normales, indexadas

MEDIO ≈ 1/2. Vistas actualizables

BAJO ≈ 0/2. No maneja Vistas

9. CURSORES

TABLA XXVI

Puntaje de Cursores

PUNTAJE	
ALTO	**BAJO**
2	**1**
SQL SERVER MYSQL	

Elaborado por: Autora

ALTO ≈ 2/2. Maneja cursores

BAJO ≈ 1/2. No maneja cursores

10. TRIGGER

TABLA XXVII

Puntaje de Triggers

PUNTAJE		
ALTO	MEDIO	BAJO
2	1	0
SQL SERVER	MYSQL	

Elaborado por: Autora

ALTO ≈ 2/2. Maneja triggers

MEDIO ≈ 1/2. Maneja triggers rudimentarios

BAJO ≈ 0/2. No maneja triggers

11. FUNCIONES

TABLA XXVIII

Puntaje de Funciones

PUNTAJE	
ALTO	**BAJO**
2	**1**
SQL SERVER MYSQL	

Elaborado por: Autora

ALTO ≈ 2/2. Maneja funciones

BAJO ≈ 1/2. No maneja funciones

12. PROCEDIMIENTOS

TABLA XXIX

Puntaje de Procedimientos

PUNTAJE	
ALTO	**BAJO**
2	**1**
SQL SERVER MYSQL	

Elaborado por: Autora

ALTO ≈ 2/2. Maneja procedimientos

BAJO ≈ 1/2. No maneja procedimientos

13. INDICES

TABLA XXX

Puntaje de Índices

PUNTAJE	
ALTO	**BAJO**
10	**1**
SQL SERVER MYSQL	

Elaborado por: Autora

ALTO ≈ 2/10. Maneja índices

BAJO ≈ 1/10. No maneja índices

14. PARTICIONAMIENTO DE TABLAS

TABLA XXXI

Puntaje de Particionamiento de Tablas

PUNTAJE	
ALTO	**BAJO**
8	**1**
SQL SERVER MYSQL	

Elaborado por: Autora

ALTO ≈ 2/8. Maneja particionamiento de tablas

BAJO ≈ 1/8. No maneja particionamiento de tablas

15. REPLICACIÓN

TABLA XXXII

Puntaje de Replicación

PUNTAJE		
ALTO	**MEDIO**	**BAJO**
2	**1**	**0**
SQL SERVER	MYSQL	

Elaborado por: Autora

ALTO ≈ 2/2. Maneja replicación instantánea, transaccional y mezcla

MEDIO ≈ 1/2. Maneja replicación en un solo sentido

BAJO ≈ 0/2. No maneja replicación

16. SEGURIDAD

TABLA XXXIII

Puntaje de Seguridad

PUNTAJE		
ALTO	**MEDIO**	**BAJO**
4	**2**	**1**
SQL SERVER MYSQL		

Elaborado por: Autora

ALTO ≈ 4/4. Seguridad de alto nivel

MEDIO ≈ 2/4. Seguridad de nivel medio

BAJO ≈ 1/4. Seguridad de bajo nivel

17. ESTABILIDAD

TABLA XXXIV

Puntaje de Estabilidad

PUNTAJE		
ALTO	**MEDIO**	**BAJO**
5	**2.5**	**1**
SQL SERVER		MYSQL

Elaborado por: Autora

ALTO ≈ 5/5. Estabilidad de alto nivel

MEDIO ≈ 2.5/5. Estabilidad de nivel medio

BAJO ≈ 1/5. Estabilidad de bajo nivel

18. HERRAMIENTAS GUI DE ADMINISTRACIÓN

TABLA XXXV

Puntaje de Herramientas GUI de administración

PUNTAJE	
ALTO	**BAJO**
8	**1**
SQL SERVER MYSQL	

Elaborado por: Autora

ALTO ≈ 3/3. Maneja herramientas GUI de administración

BAJO ≈ 1/3. No maneja herramientas GUI de administración

19. ASESOR DE OPTIMIZACION DE BASE DE DATOS

TABLA XXXVI

Puntaje de Optimización de Base de Datos

PUNTAJE	
ALTO	**BAJO**
4	**1**
SQL SERVER	MYSQL

Elaborado por: Autora

ALTO ≈ 4/4. Maneja asesor de optimización de base de datos

BAJO ≈ 1/4. No maneja asesor de optimización de base de datos

20. IMPORTACION / EXPORTACION

TABLA XXXVII

Puntaje de Importación/Exportación

PUNTAJE	
ALTO	**BAJO**
2	**1**
SQL SERVER MYSQL	

Elaborado por: Autora

ALTO ≈ 2/2. Maneja importación / exportación de datos

BAJO ≈ 1/2. No maneja importación / exportación de datos

21. DATA WAREHOUSING

TABLA XXXVIII

Puntaje de Data Warehousing

PUNTAJE	
ALTO	**BAJO**
4	**1**
SQL SERVER	MYSQL

Elaborado por: Autora

ALTO ≈ 4/4. Maneja data warehousing

BAJO ≈ 1/4. No maneja data warehousing

22. SERVICIOS PARA CARGAS ETLs:

TABLA XXXIX

Puntaje de Servicios para cargas ETLs

PUNTAJE	
ALTO	**BAJO**
6	**1**
SQL SERVER	MYSQL

Elaborado por: Autora

ALTO ≈ 6/6. Maneja servicios para cargas ETLs

BAJO ≈ 1/6. No maneja servicios para cargas ETLs

23. MINERIA DE DATOS:

TABLA XL

Puntaje de Minería de Datos

PUNTAJE	
ALTO	**BAJO**
4	**1**
SQL SERVER	MYSQL

Elaborado por: Autora

ALTO ≈ 4/4. Maneja minería de datos

BAJO ≈ 1/4. No maneja minería de datos

24. HERRAMIENTAS DE BUSINESS INTELLIGENCE

TABLA XLI

Puntaje de Business Intelligence

PUNTAJE	
ALTO	**BAJO**
4	**1**
SQL SERVER	MYSQL

Elaborado por: Autora

ALTO ≈ 4/4. Maneja herramientas Business Intelligence

BAJO ≈ 1/4. No maneja herramientas Business Intelligence

25. SERVICIOS PARA PROCESAMIENTO OLAP

TABLA XLII

Puntaje de Servicios para Procesamiento OLAP

PUNTAJE	
ALTO	**BAJO**
5	**1**
SQL SERVER	MYSQL

Elaborado por: Autora

ALTO ≈ 5/5. Maneja servicios para procesamiento OLAP

BAJO ≈ 1/5. No maneja servicios para procesamiento OLAP

4.5.5.1.1. TABLA FINAL DE RESULTADOS

TABLA XLIII

Tabla de resultados de la comparación de DBMS

CARACTERÍSTICAS	SQL SERVER 2005	MYSQL 5.1(MYISAM)
LICENCIAMIENTO	1	3
MULTIPLATAFORMA	1	2
SOPORTE	2	1
ACID	2	2
INTEGRIDAD REFERENCIAL	2	2
CONSULTAS	12	6
TABLAS TEMPORALES	3	3
VISTAS	2	1

CURSORES	2	2
TRIGGER	2	1
FUNCIONES	2	2
PROCEDIMIENTOS	2	2
INDICES	10	10
PARTICIONAMIENTO DE TABLAS	1	1
REPLICACIÓN	2	1
SEGURIDAD	4	4
ESTABILIDAD	5	1
HERRAMIENTAS GUI DE ADMINISTRACIÓN	3	3
ASESOR DE OPTIMIZACION DE BASE DE DATOS	4	1
IMPORTACION / EXPORTACION	2	2
DATA WAREHOUSING	4	1
SERIVCIOS PARA CARGAS ETLs:	6	1
MINERIA DE DATOS:	4	1
HERRAMIENTAS DE BUSINESS INTELLIGENCE	4	1
SERIVCIOS PARA PROCESAMIENTO OLAP	5	1
TOTAL	**87**	**55**

Elaborado por: Autora

4.5.6. CONCLUSIÓN

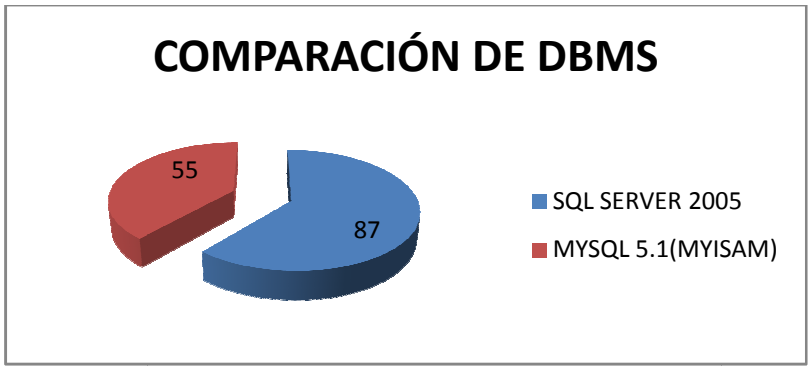


FIGURA IV. 26: Gráfico de resultado de la comparación de DBMS

Después de analizar los resultados es importante determinar que **SQL SERVER 2005** (DBMS elegido para el proyecto SII-ESPOCH) es el más adecuado para trabajar sobre optimización y utilización de herramientas Business Intelligence, cuyo resultado es notable al obtener un total de 87/100 en la puntuación total, que es una cantidad alta al compararse con 59/100 de MYSQL 5.1.

4.6. COMPARACIÓN DE DISEÑO DEL DATA WAREHOUSE

En la comparación del diseño del Data Warehouse se analizarán y se realizarán pruebas sobre la desnormalización, técnica que es la base del diseño.

4.5.7. TÉCNICA DE DESNORMALIZACIÓN

La técnica de desnormalización es una de las más importantes a nivel del diseño del Data Warehouse, debido a que es la base de la que parte el Tuning.

4.5.7.1. AMBIENTE DE PRUEBA #2

La prueba de la técnica de desnormalización será planteada en un escenario transaccional (OLTP) en el que existirá normalización y se comparará con un escenario para operaciones de consulta (OLAP) que será desnormalizada, consiguiendo de esta forma probar la eficacia de la técnica mencionada.

4.5.7.2. OBJETIVO DEL AMBIENTE DE PRUEBA #2

Comprobar el mejoramiento del rendimiento al aplicar la Técnica de Desnormalización en un ambiente OLAP

4.5.7.3. ¿CÓMO SE VA A MEDIR?

Las mediciones se van a hacer en base a:

- TIEMPO DE EJECUCIÓN DE LA CONSULTA
- LECTURA LÓGICA DE PÁGINAS
- COSTO DE LA CONSULTA

Dependiendo de la CANTIDAD DE DATOS

4.5.7.4. EXPERIMENTO

Para poder entender las pruebas que se realizarán, es importante conocer el diseño de los distintos escenarios:

DISEÑO DE LAS TABLAS NORMALIZADAS (OLTP)

El diseño de las tablas explica el ambiente de normalización que existe, siendo las tablas DWH_DIM_ESTUDIANTE_N, DWH_DIM_NACIONALIDAD Y DWH_DIM_SEXO el escenario de prueba OLTP. En el siguiente diagrama se puede ver la relación.

FIGURA IV. 27: Diseño OLTP

DISEÑO DE LAS TABLAS DESNORMALIZADAS (OLAP)

Al tratarse de un ambiente OLAP, es preciso notar la desnormalización que existe, es decir, las tablas DWH_DIM_SEXO y DWH_DIM_NACIONALIDAD están incluidas en la tabla DWH_DIM_ESTUDIANTE, en resumen tenemos la misma información, almacenada de una manera desnormalizada, como se indica en la siguiente figura:

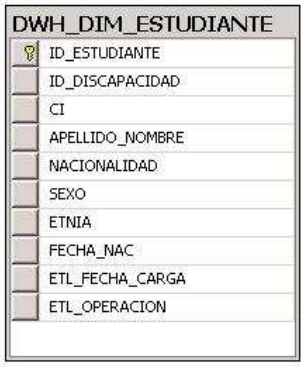


FIGURA IV. 28: Diseño OLAP

La prueba se basará en una sentencia SQL que mida el rendimiento en ambos escenarios, esta sentencia devolverá los mismos valores, pero al tratarse de escenarios diferentes, se acoplará al diseño de los mismos, siendo las sentencias utilizadas las siguientes:

Sentencia SQL utilizada en la prueba para el escenario OLTP:

```
SELECT [ID_ESTUDIANTE]
    ,[ID_DISCAPACIDAD]
    ,[CI]
    ,[APELLIDO_NOMBRE]
    ,B.[DESCRIPCION]
    ,C.[DESCRIPCION]
    ,[ETNIA]
    ,[FECHA_NAC]
    ,[ETL_FECHA_CARGA]
    ,[ETL_OPERACION]
  FROM [DWH_SIIESPOCH_PRUEBAS].[dbo].[DWH_DIM_ESTUDIANTE_N] A
  INNER JOIN [DWH_SIIESPOCH_PRUEBAS].dbo.DWH_DIM_NACIONALIDAD B
  ON A.ID_NACIONALIDAD=B.ID_NACIONALIDAD
  INNER JOIN [DWH_SIIESPOCH_PRUEBAS].dbo.DWH_DIM_SEXO C
  ON A.ID_SEXO=C.ID_SEXO
```

Esta sentencia utilizada en el ambiente OLTP, demuestra el uso de INNER JOINs que se requieren para obtener el mismo resultado que la sentencia utilizada en el ambiente OLAP.

Sentencia SQL utilizada en la prueba para el escenario OLAP:

```
SELECT [ID_ESTUDIANTE]
      ,[ID_DISCAPACIDAD]
      ,[CI]
      ,[APELLIDO_NOMBRE]
      ,[NACIONALIDAD]
      ,[SEXO]
      ,[ETNIA]
      ,[FECHA_NAC]
      ,[ETL_FECHA_CARGA]
      ,[ETL_OPERACION]
  FROM [DWH_SIIESPOCH_PRUEBAS].[dbo].[DWH_DIM_ESTUDIANTE]
```

Es una sentencia simple, ya que toda la información requerida se encuentra en la misma tabla.

4.5.7.4.1. TABLAS DE DATOS DE RESULTADO DE LA PRUEBA DE LA TÉCNICA DE DESNORMALIZACIÓN

En la siguiente tabla se demuestran los datos obtenidos durante la prueba.

TABLA XLIV

Tabla de resultados de la prueba de la técnica Desnormalización

CANTIDAD DE DATOS	ESCENARIO OLTP (NORMALIZADO)			ESCENARIO OLAP (DESNORMALIZADO)		
	TIEMPO DE EJECUCIÓN DE LA CONSULTA	LECTURA LÓGICA DE PÁGINAS	COSTO DE LA CONSULTA	TIEMPO DE EJECUCIÓN DE LA CONSULTA	LECTURA LÓGICA DE PÁGINAS	COSTO DE LA CONSULTA
1000	1 seg	17	0,0714	0 seg	15	0,0147524
100.000	2 seg	1296	2,63845	1 seg	1487	0,0147524
1'000.000	21 seg	12997	22,4434	17 seg	14870	0,0147524

Elaborado por: Autora

4.5.7.5. ANÁLISIS DE RESULTADOS DE DISEÑO DEL DATA WAREHOUSE – TÉCNICA DE DESNORMALIZACIÓN

TIEMPO DE EJECUCIÓN

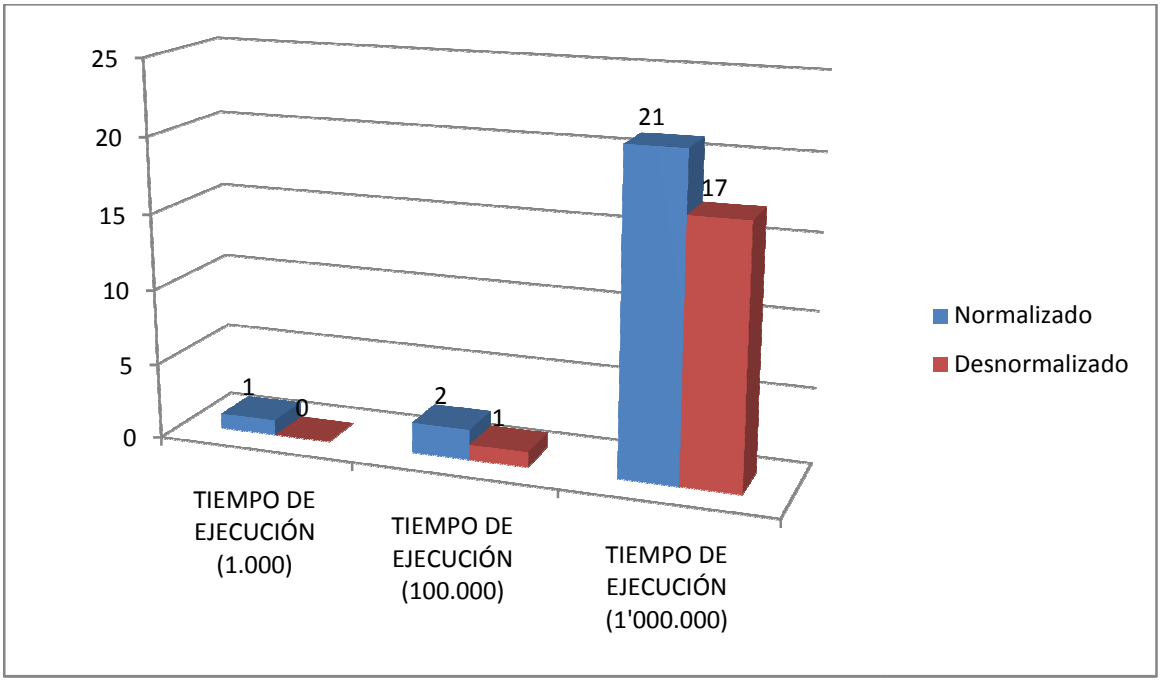


FIGURA IV. 29: Gráfico de resultado de tiempo de ejecución

El resultado de el ambiente desnormalizado en el tiempo de ejecución de la consulta demuestra que la técnica de desnormalización en realidad mejora el rendimiento, notándose más con mayor cantidad de datos, como es el caso de 17 segundos con una cantidad de 1’000.000 contra 21 segundos del ambiente normalizado.

LECTURA LÓGICA DE PÁGINAS

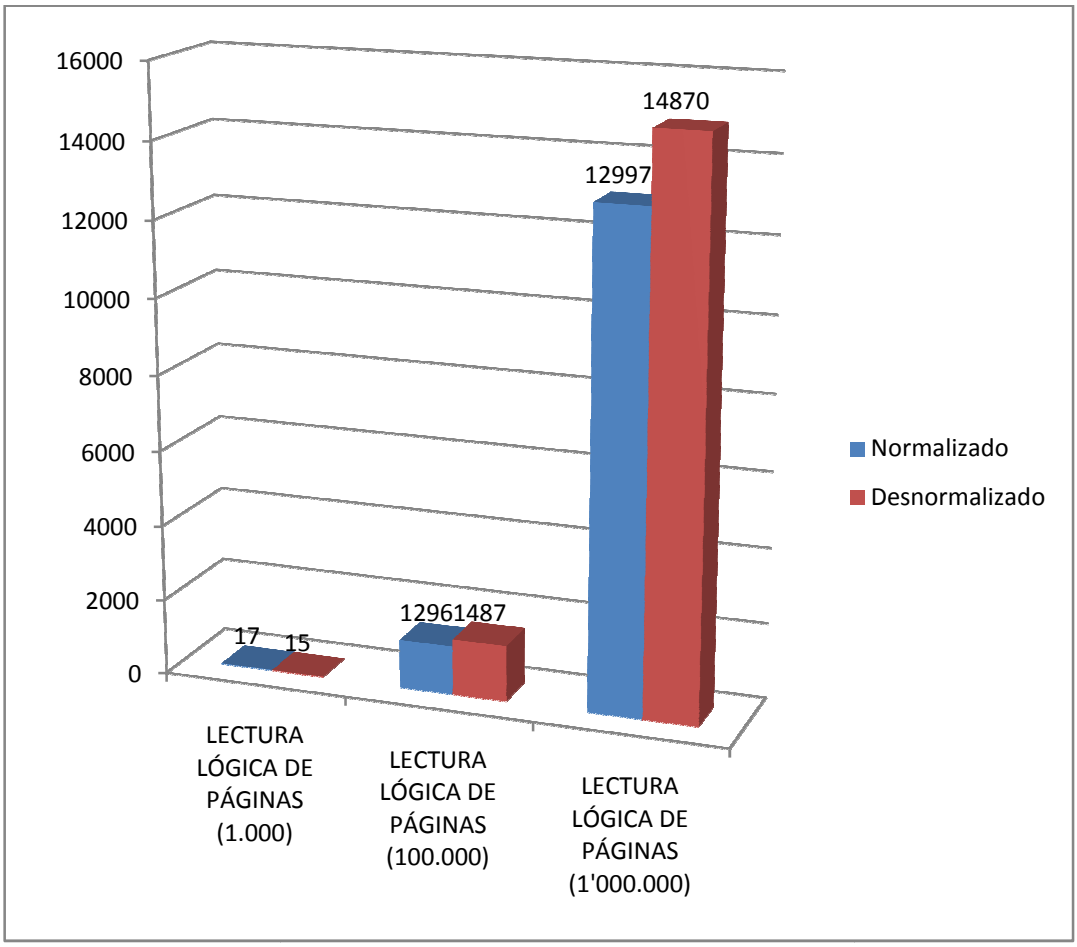


FIGURA IV. 30: Gráfico de resultado de la lecturas lógicas

En el caso de las lecturas lógicas es notable que en un ambiente desnormalizado estas son mayores debido a que al duplicar datos al evitar los JOINs de las tablas, esto hace que existan mayor cantidad de datos existentes y por lo tanto más lecturas lógicas que hacer.

COSTO DE LA CONSULTA

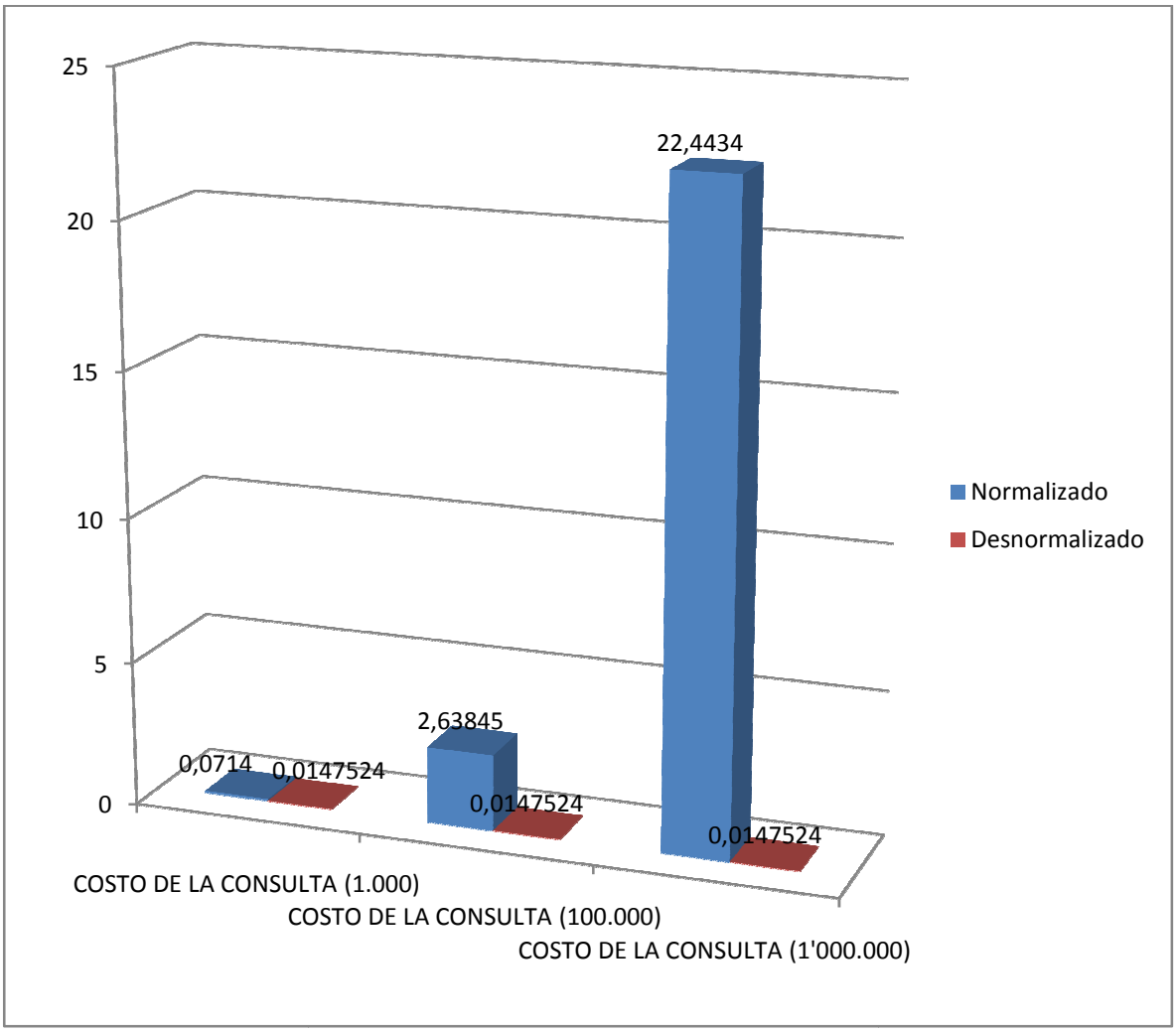


FIGURA IV. 31: Gráfico de resultado de la costo de la consulta

El costo de la consulta es uno de los valores en los que la diferencia al aplicar la técnica es más notable, debido a que al CPU le cuesta más las sentencias en un ambiente OLTP (normalizado), por tal motivo es mejor aplicar desnormalización en el Data Warehouse.

4.5.7.6. CONCLUSIÓN

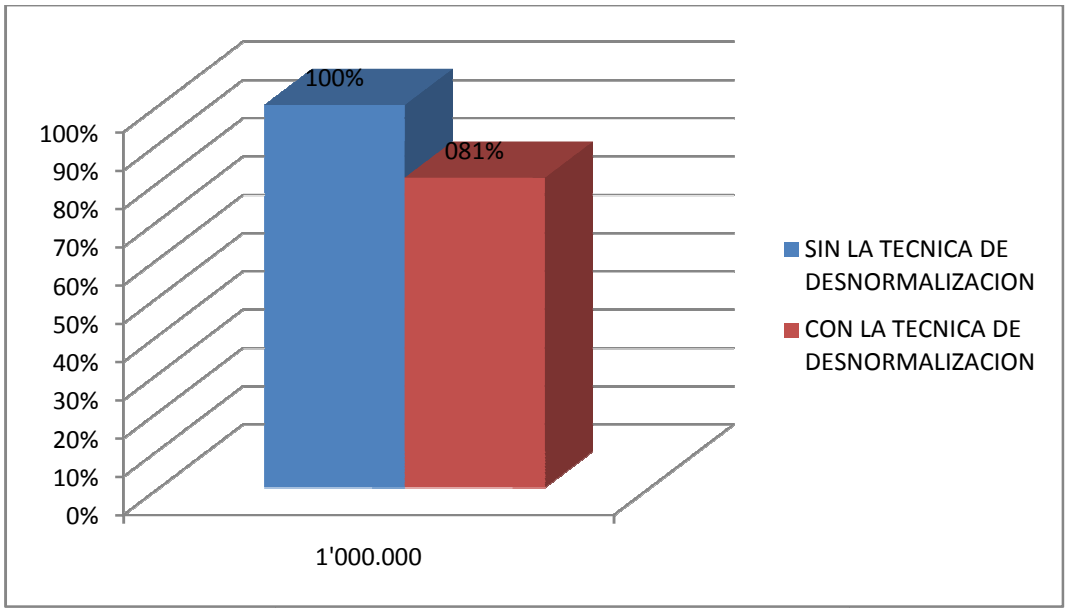


FIGURA IV. 32: Gráfico de resultado de la utilización de la Técnica de Desnormalización

Al analizar los resultados obtenidos sobre la prueba es importante notar la diferencia que existe en el tiempo, costo de la consulta y cantidad de lecturas lógicas, notándose el contraste de un ambiente OLTP normalizado y un ambiente OLAP desnormalizado, por tal motivo es preciso decir que, la técnica de desnormalización optimiza el rendimiento, notándose más claramente en los tiempos de ejecución de las sentencias SQL al manejar mayor cantidad de datos (1'000.000), siendo un **19.05%** la mejora existente en el rendimiento en esta prueba.

4.7. COMPARACIÓN DE CONSULTAS

Durante el análisis de la comparación de consultas, se analizarán las técnicas de utilización de índices, particionamiento de la tabla y la calidad del script.

4.7.1. TÉCNICA DE UTILIZACIÓN DE ÍNDICES

La utilización de índices es una de las técnicas más importantes en la optimización de rendimiento, que se aplica durante el tuneo del Data Warehouse

4.7.1.1. AMBIENTE DE PRUEBA #3

La prueba de la técnica de utilización de índices será planteada en un escenario para operaciones de consulta (OLAP) para esta técnica se analizarán los índices que serán implementados para mejorar el rendimiento del escenario OLAP, es por eso que estas pruebas se realizarán primero en el escenario OLAP sin estas técnicas y después el mismo escenario OLAP con las técnicas de utilización de índices implementadas, consiguiendo de esta forma probar la eficacia de la técnica mencionada.

Es importante mencionar que al trabajar sobre un escenario OLAP lo que realmente es primordial es mejorar el tiempo de lectura de la información, es decir mejorar el rendimiento en las sentencias SELECT, pero también es importante analizar la escritura de la información y la intervención de los índices en estas acciones, es por eso que sobre el mismo escenario se realizará una prueba UPDATE en el escenario OLAP sin técnicas de utilización de índices y después en el mismo escenario OLAP con técnicas de utilización de índices, consiguiendo de esta manera que el rendimiento de escritura sea aceptable .

Un data warehouse tiene operaciones de acción y selección, para la operación de selección se utilizará la sentencia SELECT como ya se hizo hincapié y para la acción de operación se utilizará la sentencia UPDATE, ya que es la sentencia más critica debido a que para lograr realizar el UPDATE, primero debe realizar un DELETE sobre el dato y posteriormente realizar un INSERT en el mismo lugar.

4.7.1.2. OBJETIVO DEL AMBIENTE DE PRUEBA #3

Comprobar el mejoramiento del rendimiento al aplicar la Técnica de Utilización de Índices en un ambiente OLAP con las sentencias select y update

4.7.1.3. ¿CÓMO SE VA A MEDIR?

Las mediciones se van a hacer en base a:

- TIEMPO DE EJECUCIÓN DE LA CONSULTA
- LECTURA LÓGICA DE PÁGINAS
- COSTO DE LA CONSULTA

Dependiendo de la CANTIDAD DE DATOS

4.7.1.4. EXPERIMENTO

Para poder entender las pruebas que se realizarán, es importante conocer el diseño del escenario de prueba:

DISEÑO DE LAS TABLAS UTILIZADAS EN EL ESCENARIO (OLAP)

El diseño de las tablas explica el ambiente de prueba que existe, siendo diez tablas las utilizadas, sobre estas se aplicará el uso de índices agrupados y no agrupados. Las tablas utilizadas son:

- DSA.Oasis_Estudiantes
- DSA.Oasis_Inscripciones
- DSA.Oasis_Grados
- DWH.DIM_UBICACION_ACADEMICA
- DWH.DIM_UBICACION_GEOGRAFICA
- DWH.DIM_PERIODO_ACADEMICO
- DWH.DIM_INSTITUCION
- DWH.DIM_TITULO_ESPOCH
- DWH.DIM_TITULO_INSTITUCION
- DWH.DIM_ESTUDIANTE

En la siguiente figura se muestra la relación de las tablas manipuladas en la prueba de utilización de índices, estas tablas podrán permitir posteriormente llenar la tabla DWH.FAC_INSCRIPCION_GRADO, es por eso que esta tabla también aparece en el diagrama, para poder entender la relación.

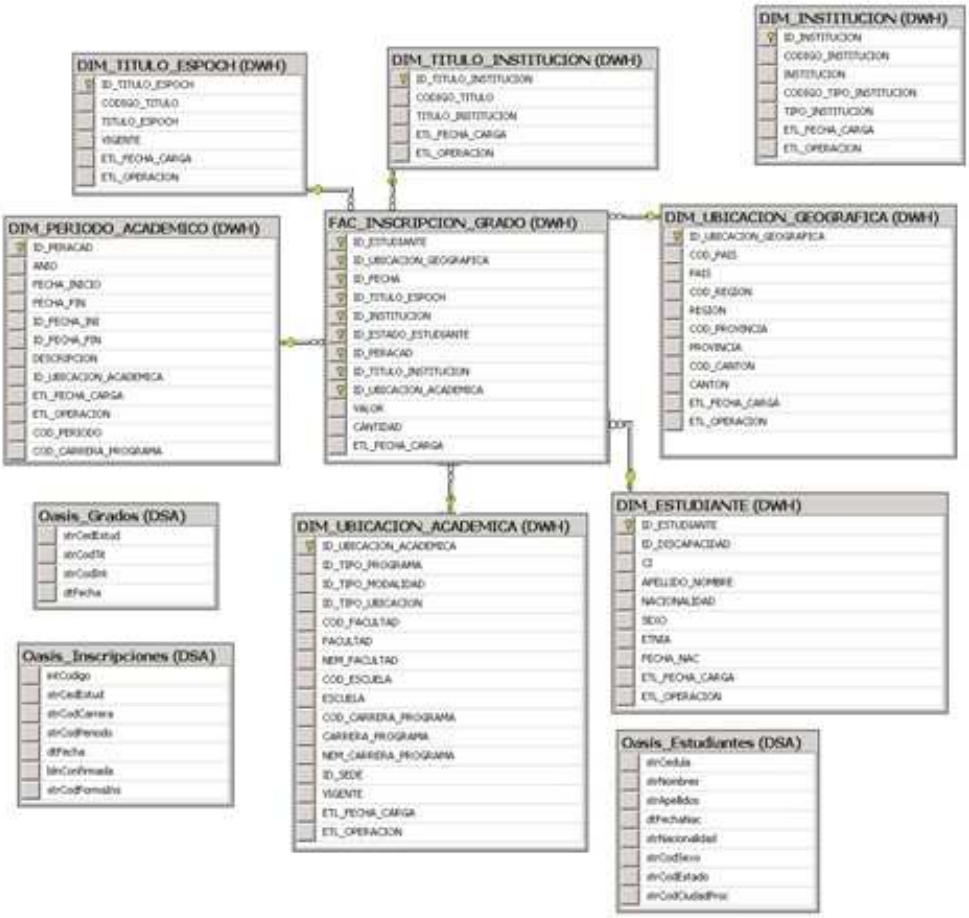


FIGURA IV. 33: Diseño OLAP

Elaborado por: Autora

4.7.1.4.1. CASO 1

Esta primera prueba de operación de selección (SELECT) devolverá los valores utilizados en la tabla fact DWH.FAC_INSCRIPCION_GRADO siendo la sentencia SELECT la siguiente:

Sentencia SQL SELECT utilizada en el escenario OLAP:

```
SELECT   DISTINCT  B.ID_ESTUDIANTE

    ,D.ID_UBICACION_GEOGRAFICA

    ,cast(COALESCE(A.dtFecha,'01/01/1900') as int) AS ID_FECHA

    ,J.ID_TITULO_ESPOCH AS ID_TITULO_ESPOCH

    ,CASE WHEN E.strCodInt IS NOT NULL THEN F.ID_INSTITUCION

        ELSE 0 END AS ID_INSTITUCION

    ,CASE WHEN A.blnConfirmada=1 THEN 1

        ELSE 2 END AS ID_ESTADO_ESTUDIANTE

    ,G.ID_PERACAD

   ,CASE WHEN (E.strCodTit IS NOT NULL AND H.ID_TITULO_INSTITUCION IS NOT NULL)  THEN
H.ID_TITULO_INSTITUCION

        WHEN (E.strCodTit IS NOT NULL AND H.ID_TITULO_INSTITUCION IS NULL) THEN
J.ID_TITULO_ESPOCH

        ELSE 0 END AS ID_TITULO_INSTITUCION

    ,G.ID_UBICACION_ACADEMICA  AS ID_UBICACION_ACADEMICA --,I.ID_UBICACION_ACADEMICA

    ,0 AS VALOR

    ,1 AS CANTIDAD

    ,GETDATE() AS ETL_FECHA_CARGA

from   DSA.Oasis_Inscripciones A INNER

 JOIN  DWH.DIM_ESTUDIANTE B    ON A.strCedEstud=B.CI

 JOIN DWH.DIM_UBICACION_ACADEMICA I ON I.COD_CARRERA_PROGRAMA=A.strCodCarrera

 JOIN DSA.Oasis_Estudiantes C ON A.strCedEstud=C.strCedula  AND C.strCedula=B.CI

 JOIN DWH.DIM_UBICACION_GEOGRAFICA D ON
LTRIM(RTRIM(C.strCodCiudadProc))=D.COD_CANTON

 JOIN DWH.DIM_PERIODO_ACADEMICO G ON G.COD_PERIODO=A.strCodPeriodo AND
G.COD_CARRERA_PROGRAMA=A.strCodCarrera

 LEFT JOIN DSA.Oasis_Grados E ON A.strCedEstud=E.strCedEstud

 LEFT JOIN DWH.DIM_INSTITUCION F ON E.strCodInt=F.CODIGO_INSTITUCION

 LEFT JOIN DWH.DIM_TITULO_INSTITUCION H ON H.CODIGO_TITULO=E.strCodTit

 LEFT JOIN DWH.DIM  TITULO  ESPOCH J ON J.CODIGO  TITULO=E.strCodTit
```

4.7.1.4.2. CASO 2

Para la segunda prueba de la operación de acción (UPDATE) se utilizará la siguiente sentencia:

Sentencia SQL UPDATE utilizada en el escenario OLAP:

```
UPDATE [DWH].[DIM_ESTUDIANTE]
SET [CI] = 'XXXXXXXXX-0'
```

Es importante mencionar que en la vida real las cédulas de identidad (CI) de los estudiantes no cambiarán en el tiempo, pero para poder demostrar el peso de los índices en un campo durante operaciones de acción, es primordial actualizar el campo que forma parte de la clave del índice.

Para la aplicación de la técnica utilización de índices hay que tomar en cuenta que estos son "estructuras" alternativas a la organización de los datos en una tabla. El propósito de los índices es acelerar el acceso a los datos mediante operaciones físicas más rápidas y efectivas. Siendo su objetivo acceder más rápido a los datos.

4.7.1.4.3. ÍNDICES UTILIZADOS PARA AMBOS CASOS

Estos índices utilizados serán de dos tipos:

- Índices Agrupados
- Índices no Agrupados

Además el éxito de la técnica de utilización de índices, se verá reflejada por la combinación de los índices SQL SERVER.

Índices creados que se utilizarán sobre el escenario OLAP (Definidos previamente según el análisis de la sentencia select SQL utilizada):

```
CREATE NONCLUSTERED INDEX INDEX_ESTUDIANTES ON
DSA.Oasis_Estudiantes (strCedula)
INCLUDE (strCodCiudadProc)
ON [INDEX]

CREATE NONCLUSTERED INDEX INDEX_ESTUDIANTES ON
DWH.DIM_ESTUDIANTE(CI)
INCLUDE (ID_ESTUDIANTE)
ON [INDEX]

CREATE NONCLUSTERED INDEX INDEX_GRADOS ON
DSA.Oasis_Grados(strCedEstud)
INCLUDE (strCodTit,strCodInt)
ON [INDEX]
```

4.7.1.4.4. TABLAS DE DATOS DE RESULTADO DE LA PRUEBA DE LA TÉCNICA DE UTILIZACION DE ÍNDICES

En esta primera tabla se muestran los resultados de las pruebas con la utilización de la sentencia SELECT.

CASO 1:

TABLA XLV

Tabla de resultados de la prueba de la técnica utilización de índices (select)

SENTECIA SQL SELECT						
CANTIDAD DE DATOS	**ESCENARIO OLAP (SIN TÉCNICAS)**			**ESCENARIO OLAP (CON TÉCNICAS)**		
	TIEMPO DE EJECUCIÓN DE LA CONSULTA	**LECTURA LÓGICA DE PÁGINAS**	**COSTO DE LA CONSULTA**	**TIEMPO DE EJECUCIÓN DE LA CONSULTA**	**LECTURA LÓGICA DE PÁGINAS**	**COSTO DE LA CONSULTA**
1000	0 seg	733	0,233807	0 seg	2700	0,15434
100.000	7265 seg (2:01:05)	198’962.614	3,53823	251 seg (00:04:11)	4’784.482	1,05446
1’000.000	54388 seg (15:06:38)	1.537’569.853	16,362	2581 seg (00:43:01)	44’366.046	1,05446

Elaborado por: Autora

En la segunda prueba se utilizó la sentencia UPDATE, siendo la siguiente tabla el resultado de esta.

CASO 2:

TABLA XLVI

Tabla de resultados de la prueba de la técnica utilización de índices (update)

SENTECIA SQL UPDATE						
CANTIDAD DE DATOS	**ESCENARIO OLAP (SIN TÉCNICAS)**			**ESCENARIO OLAP (CON TÉCNICAS)**		
	TIEMPO DE EJECUCIÓN DE LA CONSULTA	**LECTURA LÓGICA DE PÁGINAS**	**COSTO DE LA CONSULTA**	**TIEMPO DE EJECUCIÓN DE LA CONSULTA**	**LECTURA LÓGICA DE PÁGINAS**	**COSTO DE LA CONSULTA**
1000	0 seg	1004	0,117674	0 seg	4557	0,166171
100.000	0 seg	100.188	10,6573	3 seg	914.473	13,7485
1’000.000	8 seg	1’001.865	107,133	70 seg	10’237.293	360,222

Elaborado por: Autora

4.7.1.5. ÁNALISIS DE RESULTADOS DE CONSULTAS – TÉCNICA DE UTILIZACION DE ÍNDICES

4.7.1.5.1. CASO 1

TIEMPO DE EJECUCIÓN

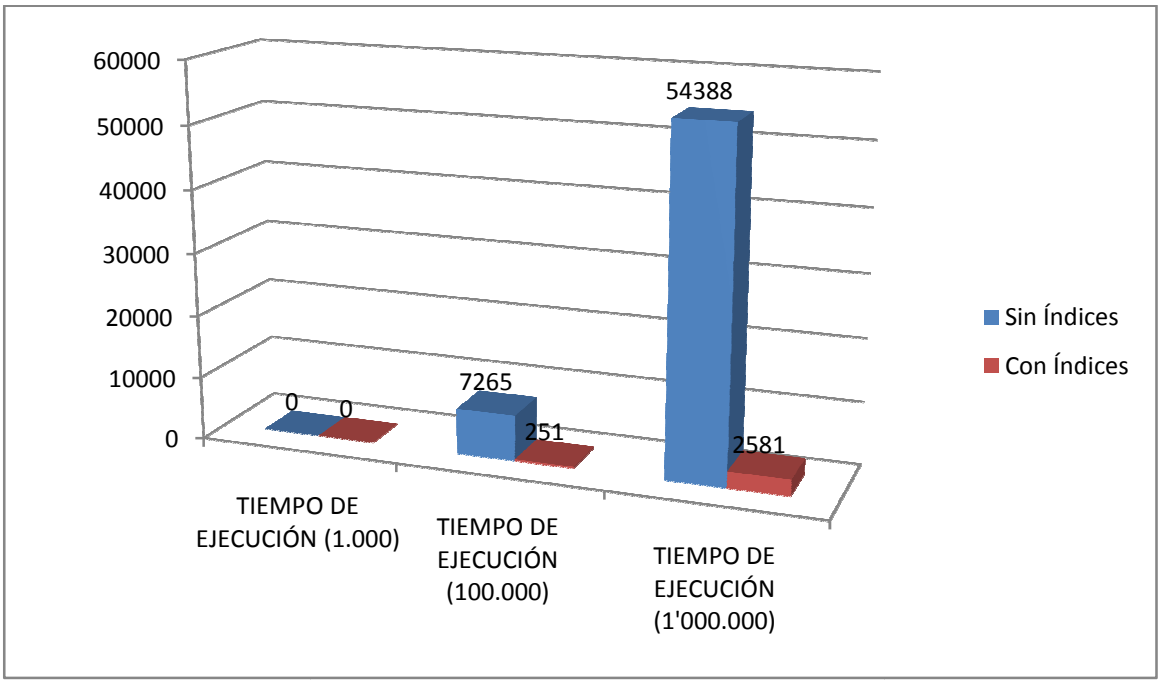


FIGURA IV. 34: Gráfico de resultado de tiempo de ejecución

El resultado de la utilización de índices en el tiempo de ejecución de la consulta demuestra que la técnica de utilización de índices en realidad mejora el rendimiento, notándose más con mayor cantidad de datos, como es el caso de 2581 segundos con una cantidad de 1’000.000 contra 54388 segundos del ambiente sin índices.

LECTURA LÓGICA DE PÁGINAS

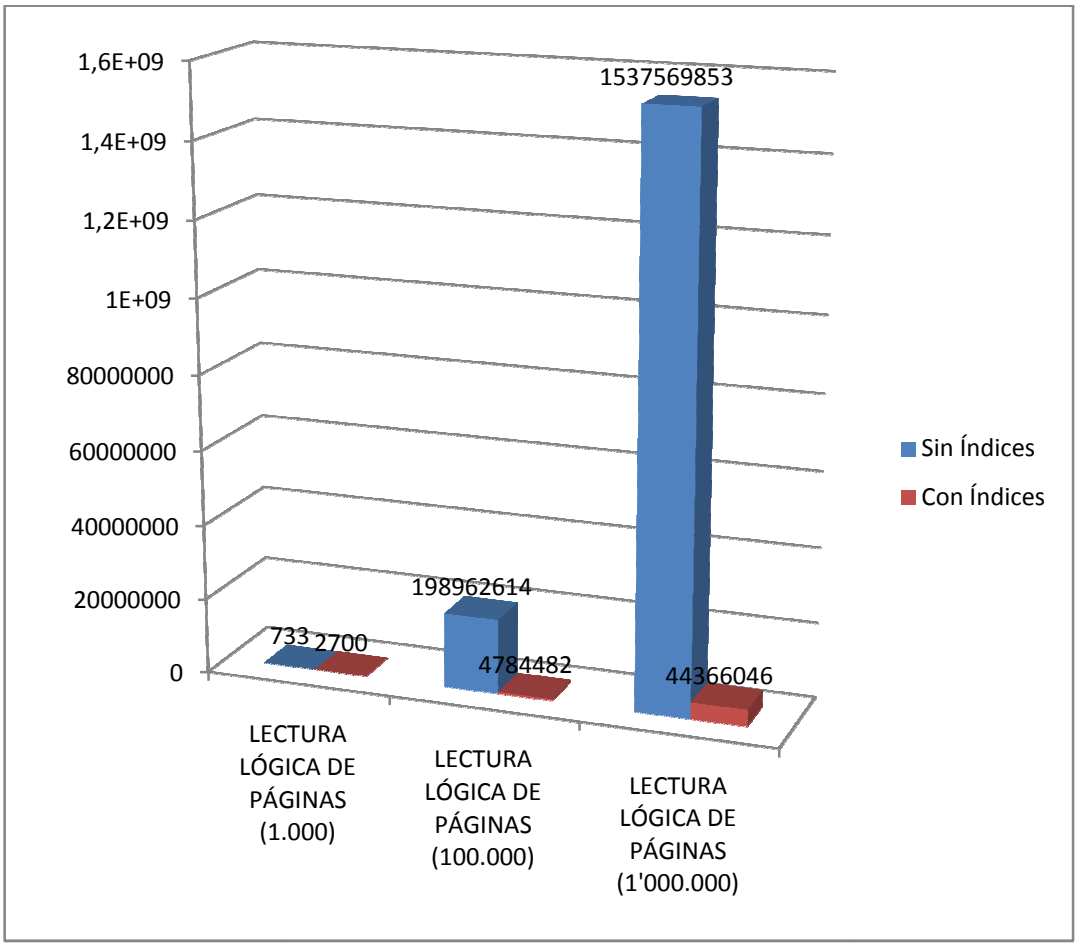


FIGURA IV. 35: Gráfico de resultado de la lecturas lógicas

En el caso de las lecturas lógicas es de igual manera notable que la técnica de indexación disminuye estas lecturas en grandes proporciones.

COSTO DE LA CONSULTA

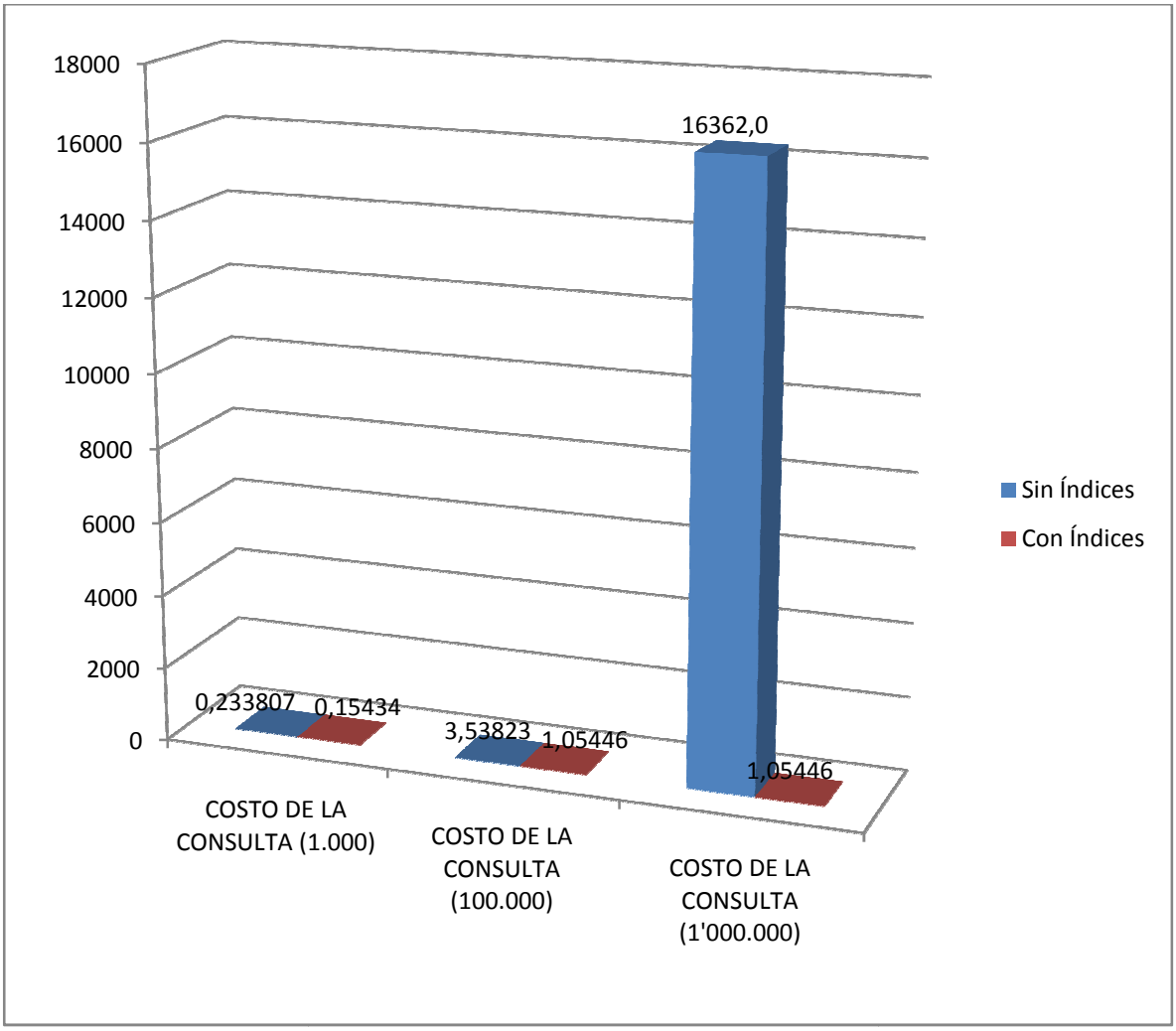


FIGURA IV. 36: Gráfico de resultado de la costo de la consulta

El costo de la consulta es uno de los valores en los que la diferencia al aplicar la técnica es más notable, debido a que al CPU le cuesta más las sentencias en un ambiente sin índices, por tal motivo es mejor aplicar la técnica de utilización de índices en el Data Warehouse.

4.7.1.5.2. CASO 2

TIEMPO DE EJECUCIÓN

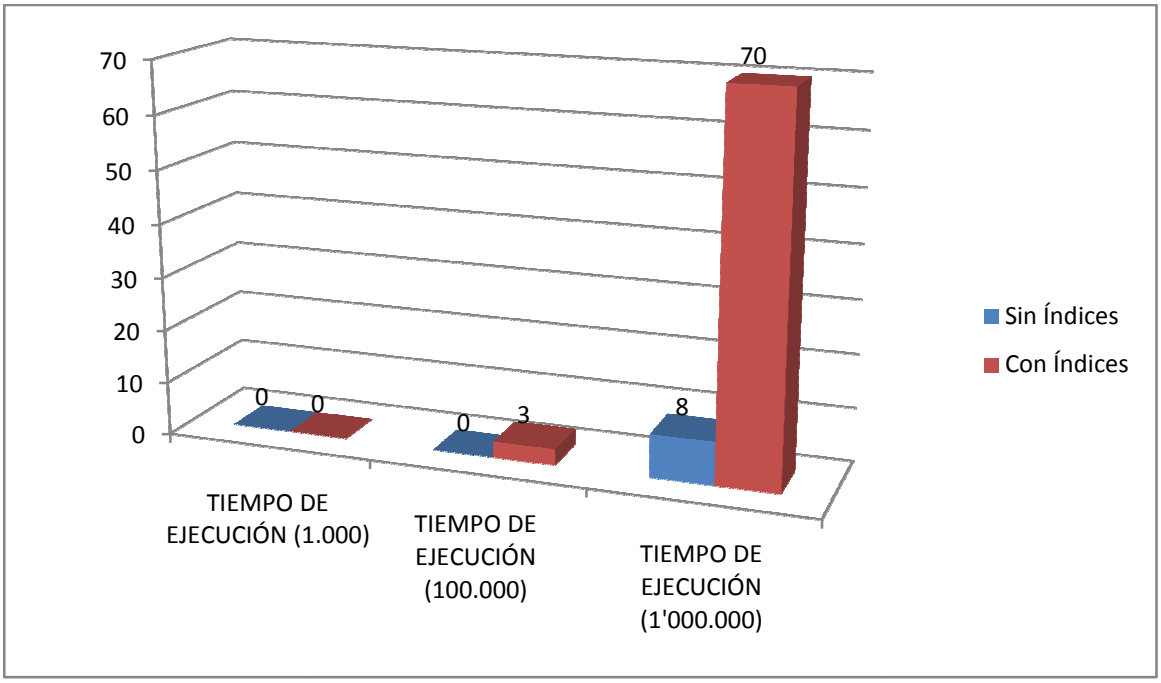


FIGURA IV. 37: Gráfico de resultado de tiempo de ejecución

El resultado de la utilización de índices en el tiempo de ejecución de la consulta demuestra que la técnica de utilización de índices afecta a las sentencias de acción en el data warehouse, como se puede ver en toda la cantidad de datos, pero con mayor diferencia cuando se maneja 1'000.000 que con índices se demora 70 segundos contra 8 segundos sin índices.

LECTURA LÓGICA DE PÁGINAS

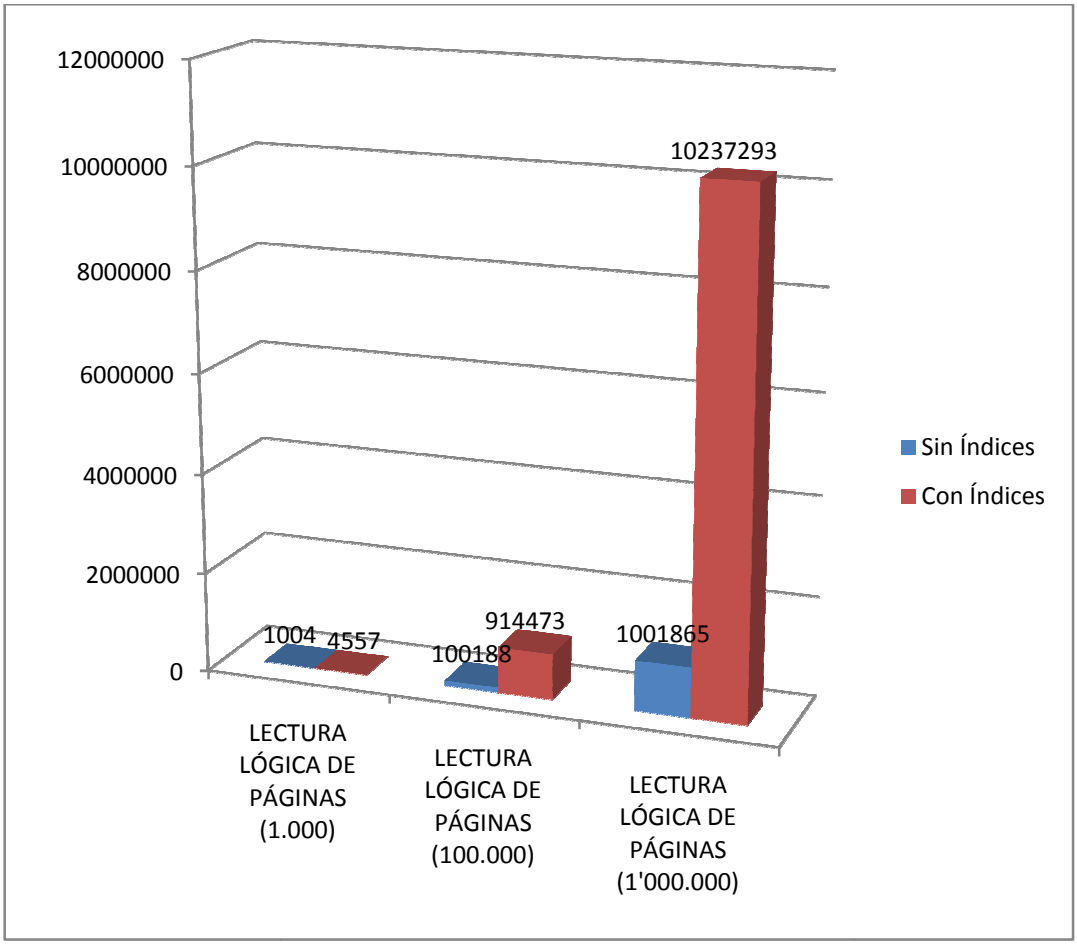


FIGURA IV. 38: Gráfico de resultado de la lecturas lógicas

De igual manera en el caso de las lecturas lógicas es notable que la técnica de indexación afecta a las sentencias update, aumentando las lecturas lógicas.

COSTO DE LA CONSULTA

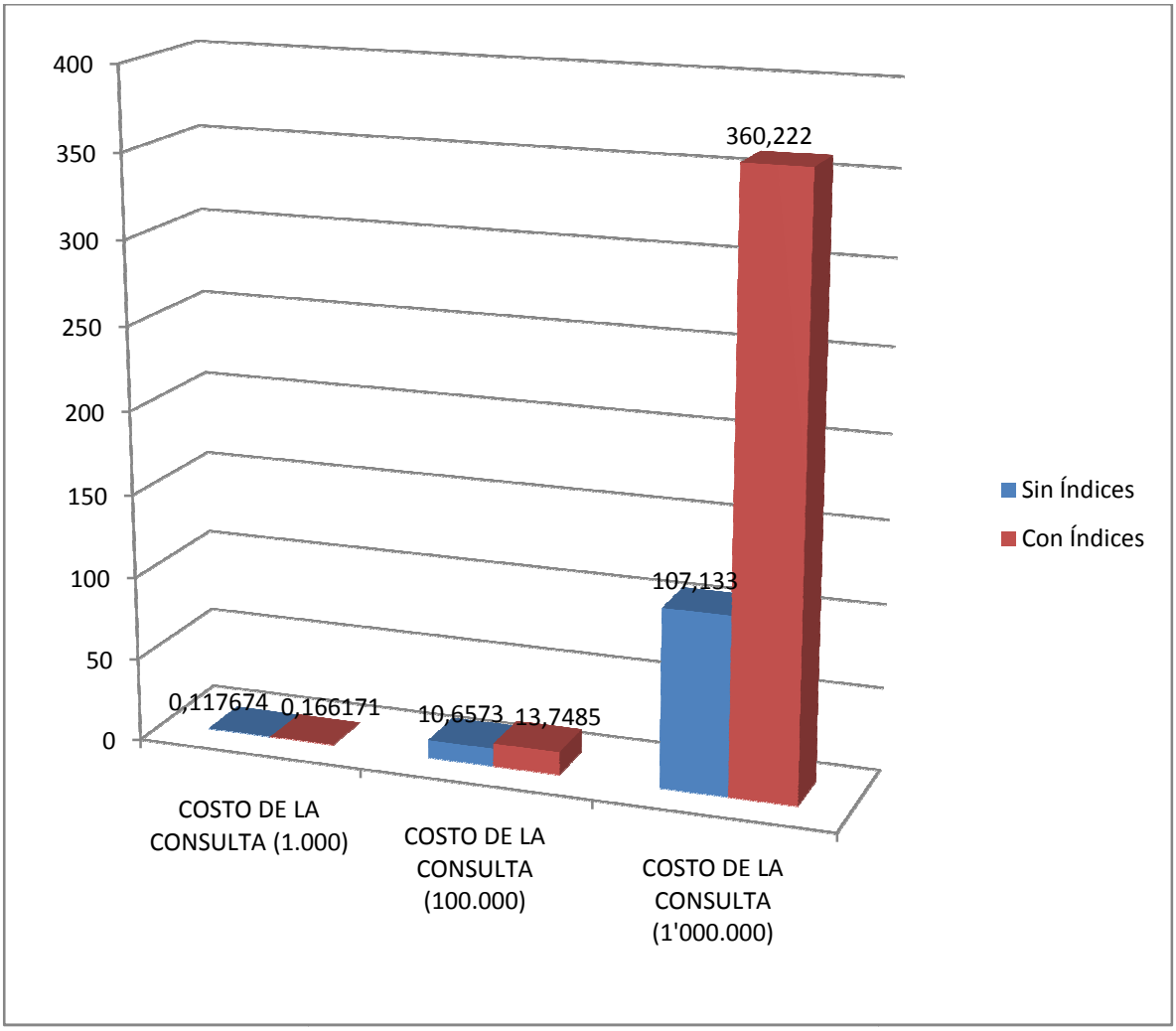


FIGURA IV. 39: Gráfico de resultado de la costo de la consulta

El costo de la consulta es uno de los valores en los que la diferencia al aplicar la técnica es más notable, debido a que al CPU le cuesta más las sentencias en un ambiente con índices.

4.7.1.6. CONCLUSIÓN

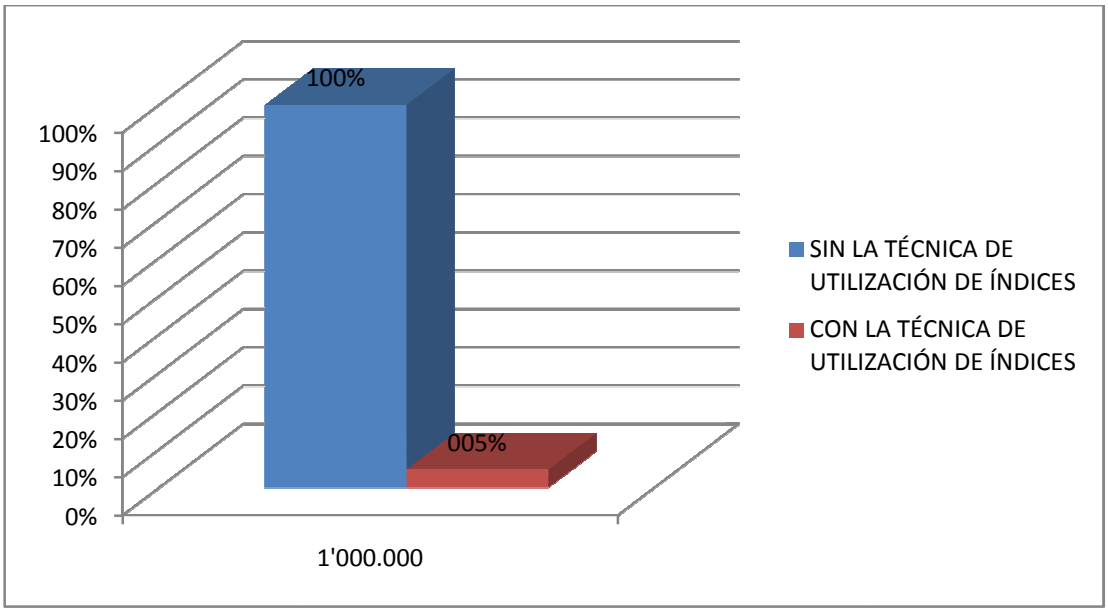


FIGURA IV. 40: Gráfico de resultado del uso de la Técnica de Utilización de Índices

Es importante notar que para este experimento se realizaron dos casos de prueba, en el caso 1 se analizó la sentencia SELECT, que en un ambiente OLAP, como sobre el que se está trabajando y en el que importa la optimización de estas sentencias, la técnica de utilización de índices en realidad mejora el rendimiento en un **95.15%**, pero en sentencias de acción que son más utilizadas en un ambiente OLTP, la utilización de índices no crea optimización de rendimiento, más bien es al contrario, pero como el tema de análisis es en un ambiente OLAP, es posible decir que la técnica de utilización de índices en el tiempo de la consulta, optimiza en gran magnitud el rendimiento en el data warehouse.

4.7.2. TÉCNICA DE PARTICIONAMIENTO DE TABLAS

La utilización del particionamiento de tablas es una de las técnicas que se realiza según la cantidad de datos existentes en las FACT, generalmente para aplicar esta

técnica hay que tener una fact con millones de datos, también es importante analizar la versión del SQL SERVER que se utiliza porque de esta depende si se puede aplicar o no.

4.7.2.1. AMBIENTE DE PRUEBA #4

La prueba de la técnica de particionamiento de tablas será planteada en un escenario para operaciones de consulta (OLAP) para esta técnica se analizará el particionamiento de la tabla, es por eso que estas pruebas se realizarán primero en el escenario OLAP sin esta técnica y después el mismo escenario OLAP con la técnica de particionamiento de tablas, demostrando de esta manera la eficacia de la técnica aplicada.

Es importante mencionar que la versión con la que se está trabajando es SQL SERVER 2005 STANDART y esta versión no soporta el particionamiento de tablas, por tal motivo, para demostrar esta técnica, se instaló en el mismo escenario la versión SQL 2005 DEVELOPER para poder demostrar el particionamiento de tablas.

4.7.2.2. OBJETIVO DEL AMBIENTE DE PRUEBA #4

Comprobar el mejoramiento del rendimiento al aplicar la Técnica de Particionamiento de Tablas en un ambiente OLAP

4.7.2.3. ¿CÓMO SE VA A MEDIR?

Las mediciones se van a hacer en base a:

- TIEMPO DE EJECUCIÓN DE LA CONSULTA

- LECTURA LÓGICA DE PÁGINAS
- COSTO DE LA CONSULTA

Dependiendo de la CANTIDAD DE DATOS

4.7.2.4. EXPERIMENTO

Para poder entender las pruebas que se realizarán, es importante conocer el diseño del escenario de prueba:

DISEÑO DE LAS TABLAS UTILIZADAS EN EL ESCENARIO (OLAP)

El diseño de las tablas explica el ambiente de prueba que existe, siendo la tabla de hechos DWH.FAC_INSCRIPCION_GRADO, la tabla a particionarse.

FIGURA IV. 41: Diseño OLAP

Elaborado por: Autora

Sentencia SQL SELECT utilizada en el escenario OLAP:

```
SELECT [ID_ESTUDIANTE]
      ,[ID_UBICACION_GEOGRAFICA]
      ,[ID_FECHA]
      ,[ID_TITULO_ESPOCH]
      ,[ID_INSTITUCION]
      ,[ID_ESTADO_ESTUDIANTE]
      ,[ID_PERACAD]
      ,[ID_TITULO_INSTITUCION]
      ,[ID_UBICACION_ACADEMICA]
      ,[VALOR]
      ,[CANTIDAD]
      ,[ETL_FECHA_CARGA]
  FROM [SII_ESPOCH_DEF].[DWH].[FAC_INSCRIPCION_GRADO]
 WHERE ID_FECHA>=40000
 ORDER BY ID_FECHA
```

4.7.2.4.1. SCRIPT UTILIZADO PARA EL PARTICIONAMIENTO DE LA TABLA

Para poder particionar la tabla es necesario crear la función y el esquema de partición, siendo el script utilizado el siguiente:

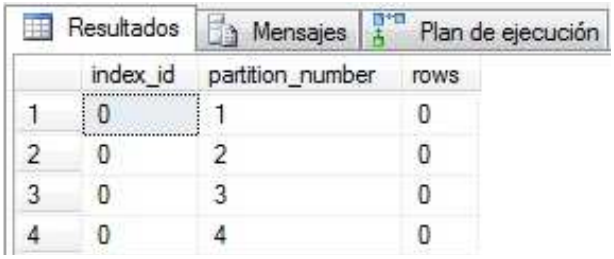
```
CREATE PARTITION FUNCTION PF(INT)
AS RANGE LEFT
FOR VALUES(1000,100000,1000000);

CREATE PARTITION SCHEME PS
AS PARTITION PF ALL TO([PRIMARY]);
```

Es importante tomar en cuenta que las tablas de particionamiento para poder aprovechar al máximo la mejora de rendimiento es preciso crear estas tablas en el esquema de partición creado e identificando la columna con la que se tomará el criterio de partición, que es la fecha, es por eso que la tabla a particionarse se crea de la siguiente manera:

```
CREATE TABLE [DWH].[FAC_INSCRIPCION_GRADO](
        [ID_ESTUDIANTE] [int] NOT NULL,
        [ID_UBICACION_GEOGRAFICA] [int] NOT NULL,
        [ID_FECHA] [int] NOT NULL,
        [ID_TITULO_ESPOCH] [int] NOT NULL,
        [ID_INSTITUCION] [int] NOT NULL,
        [ID_ESTADO_ESTUDIANTE] [int] NOT NULL,
        [ID_PERACAD] [int] NOT NULL,
        [ID_TITULO_INSTITUCION] [int] NOT NULL,
        [ID_UBICACION_ACADEMICA] [int] NOT NULL,
        [VALOR] [float] NOT NULL,
        [CANTIDAD] [int] NULL,
        [ETL_FECHA_CARGA] [datetime] NULL,
) ON PS([ID_FECHA])
```

Las particiones creadas son:


Resultados | Mensajes | Plan de ejecución

	index_id	partition_number	rows
1	0	1	0
2	0	2	0
3	0	3	0
4	0	4	0

FIGURA IV. 42: Particionamiento de la tabla

4.7.2.4.2. TABLAS DE DATOS DE RESULTADO DE LA PRUEBA DE LA TÉCNICA DE PARTICIONAMIENTO DE TABLAS

En esta primera tabla se muestran los resultados de las pruebas con la utilización de la sentencia SELECT.

TABLA XLVII

Tabla de resultados de la prueba de la técnica de particionamiento

CANTIDAD DE DATOS	ESCENARIO OLAP (SIN TÉCNICAS)			ESCENARIO OLAP (CON TÉCNICAS)		
	TIEMPO DE EJECUCIÓN DE LA CONSULTA	LECTURA LÓGICA DE PÁGINAS	COSTO DE LA CONSULTA	TIEMPO DE EJECUCIÓN DE LA CONSULTA	LECTURA LÓGICA DE PÁGINAS	COSTO DE LA CONSULTA
1000	0 seg	11	0,0251388	0 seg	9	0,0281596
100.000	0 seg	866	3,26881	0 seg	820	3,0565
1'000.000	7 seg	9043	27,1564	5 seg	8197	24,796

Elaborado por: Autora

4.7.2.5. ÁNALISIS DE RESULTADOS DE CONSULTAS – TÉCNICA DE PARTICIONAMIENTO DE TABLAS

TIEMPO DE EJECUCIÓN

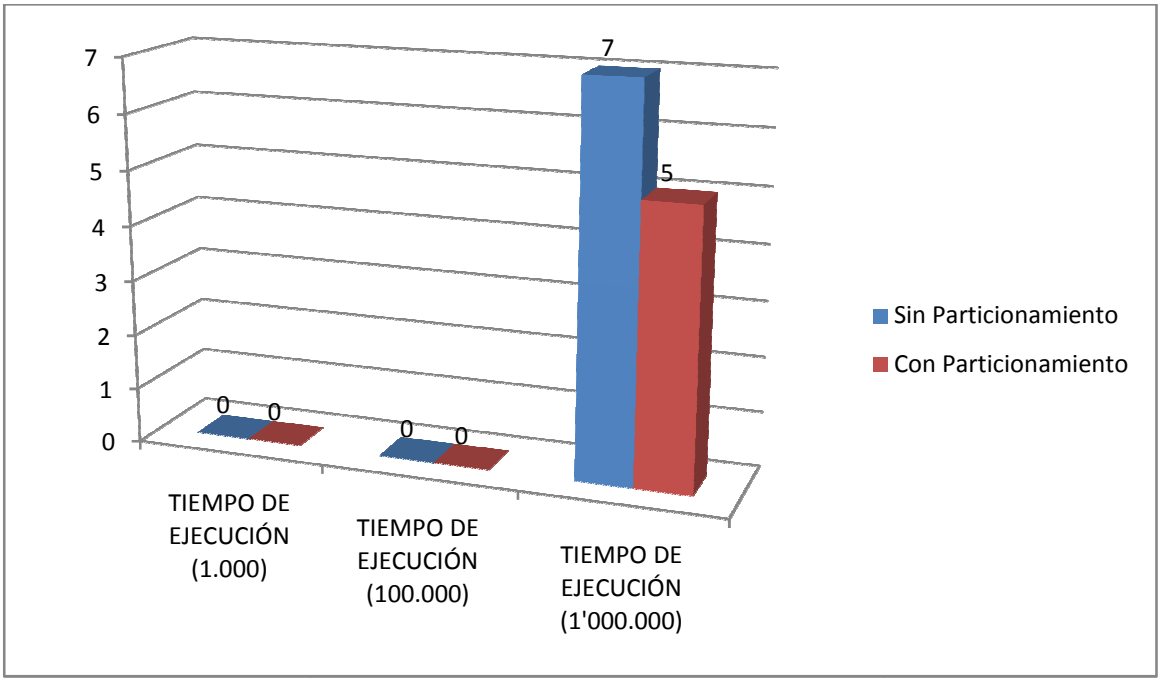


FIGURA IV. 43: Gráfico de resultado de tiempo de ejecución

El resultado de la utilización de particionamiento en la tabla, refleja en el tiempo de ejecución de la consulta una mejora en el tiempo de respuesta, siendo este resultado no tan variante, pero en mayor cantidad de datos este reflejará mayor mejora, tomando en cuenta que una tabla se debe particionar cuando existen mayor cantidad de datos.

LECTURA LÓGICA DE PÁGINAS

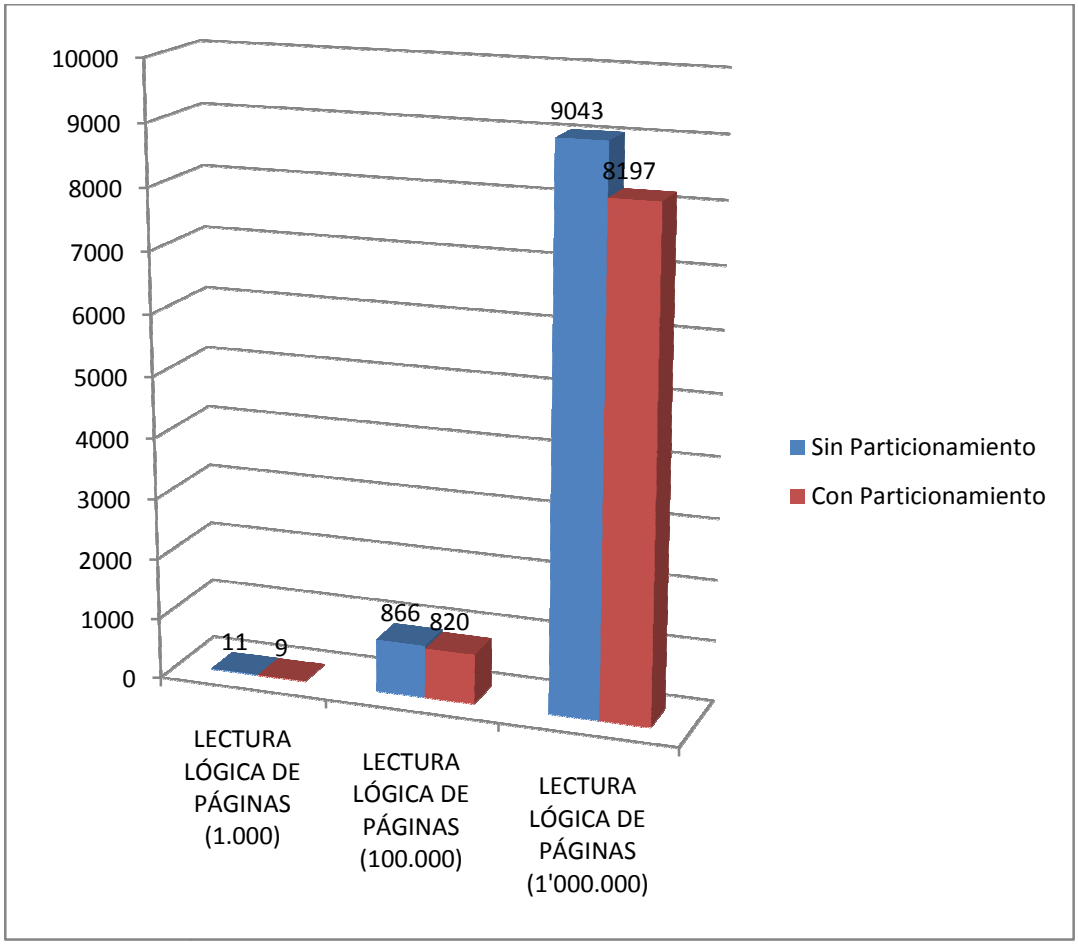


FIGURA IV. 44: Gráfico de resultado de la lecturas lógicas

En el caso de las lecturas lógicas es de igual manera notable su mejora, disminuyendo estas lecturas.

COSTO DE LA CONSULTA

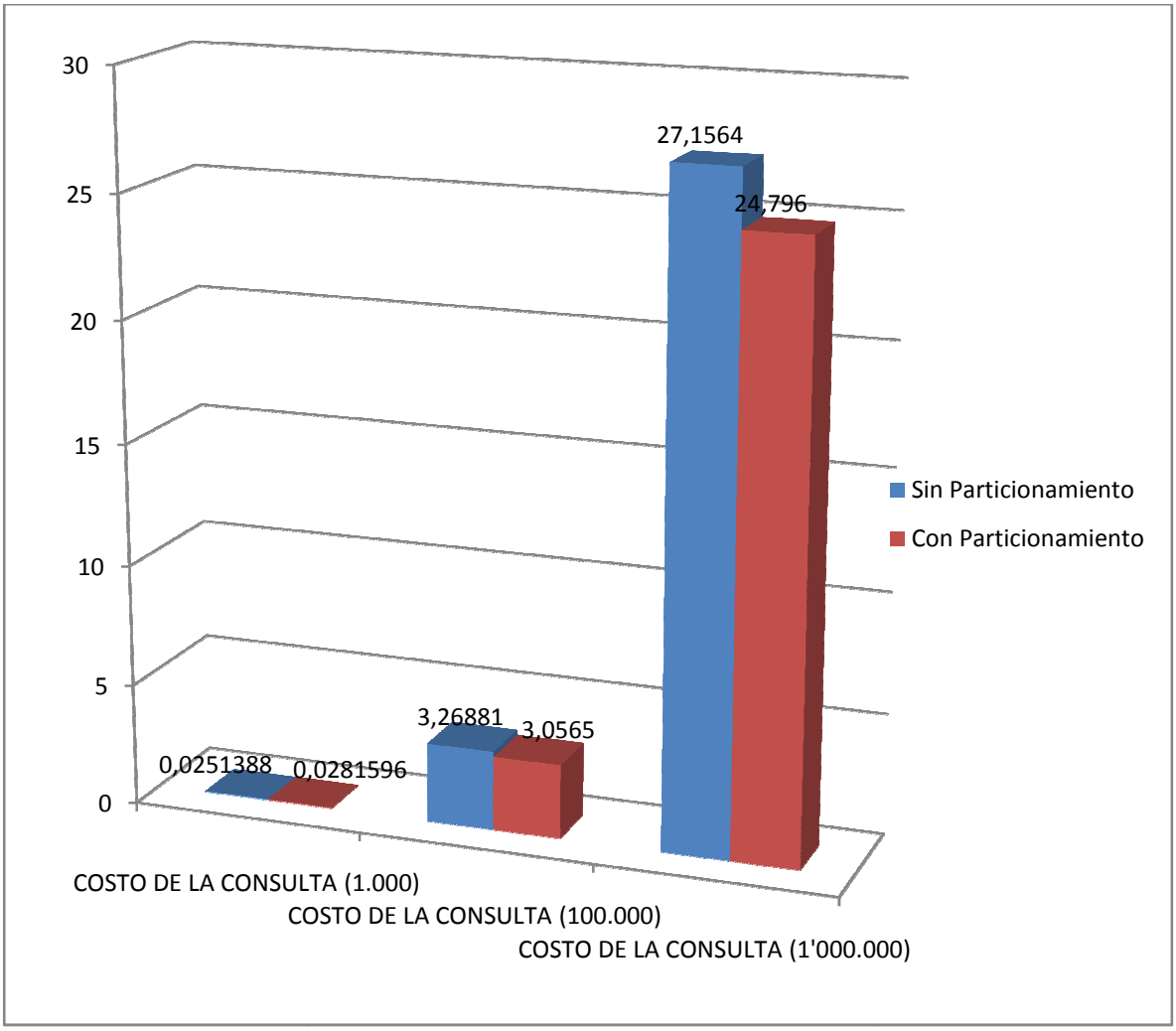


FIGURA IV. 45: Gráfico de resultado de la costo de la consulta

El costo de la consulta es uno de los valores en los que también se refleja la diferencia al aplicar la técnica, debido a que al CPU le cuesta más las sentencias en un ambiente sin particionamiento, aunque esta diferencia sea en pequeñas proporciones.

4.7.2.6. CONCLUSIÓN

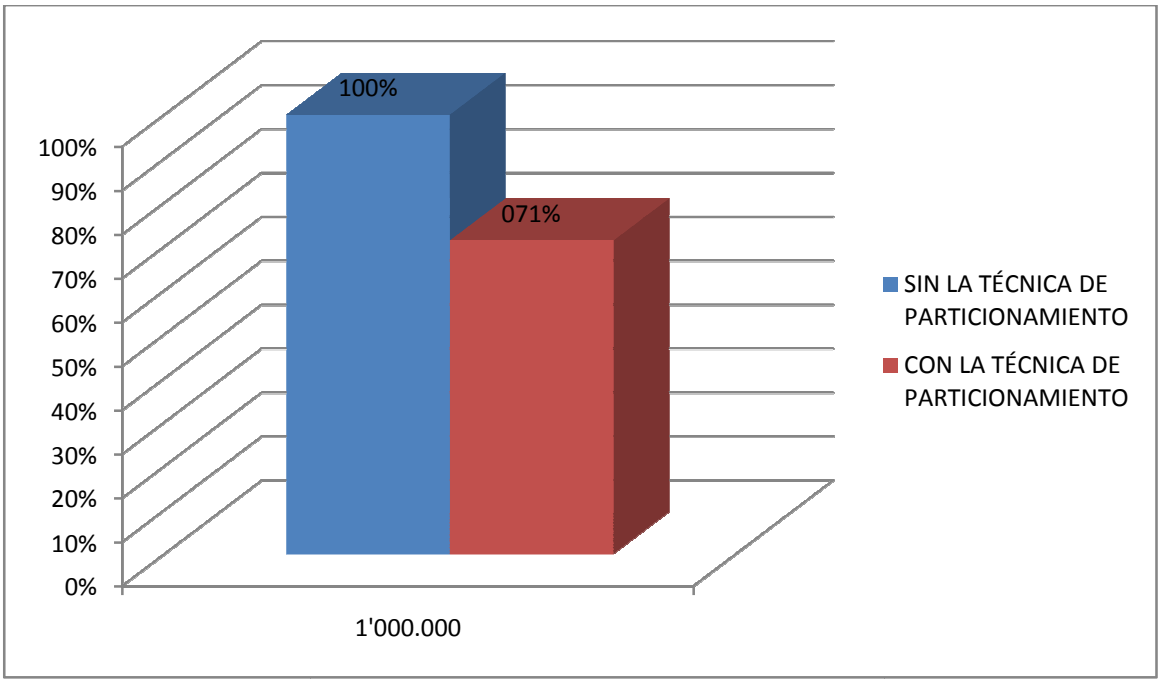


FIGURA IV. 46: Gráfico de resultado del uso de la Técnica de Particionamiento

Al analizar los resultados obtenidos sobre la prueba es importante notar la diferencia que existe en el tiempo de respuesta de la consulta, por tal motivo es preciso decir que, la técnica de particionamiento optimiza el rendimiento en un **28.57%** con 1’000.000 de datos.

4.7.3. TÉCNICA DE CALIDAD DE SCRIPT SQL

La utilización de script SQL con calidad es una de las técnicas en la que se involucra directamente los tiempos de respuesta y la lógica de lo que se quiere obtener al realizar una consulta, es decir se deben analizar correctamente la información que se quiere obtener y manejar ciertos criterios para hacer que la consulta devuelva la misma cantidad de datos en menor tiempo.

4.7.3.1. AMBIENTE DE PRUEBA #5

La prueba de la técnica de calidad de script SQL será planteada en un escenario para operaciones de consulta (OLAP) para esta técnica se analizará el script utilizado para demostrar el rendimiento obtenido en un escenario OLAP, es por eso que estas pruebas se realizarán primero en el escenario OLAP sin esta técnica y después el mismo escenario OLAP con la técnicas de de calidad de script SQL.

4.7.3.2. OBJETIVO DEL AMBIENTE DE PRUEBA #5

Comprobar el mejoramiento del rendimiento al aplicar la Técnica de Calidad de Script SQL en un ambiente OLAP

4.7.3.3. ¿CÓMO SE VA A MEDIR?

Las mediciones se van a hacer en base a:

- TIEMPO DE EJECUCIÓN DE LA CONSULTA
- LECTURA LÓGICA DE PÁGINAS
- COSTO DE LA CONSULTA

Dependiendo de la CANTIDAD DE DATOS

4.7.3.4. EXPERIMENTO

Para poder entender las pruebas que se realizarán, es importante conocer el diseño del escenario de prueba:

DISEÑO DE LAS TABLAS UTILIZADAS EN EL ESCENARIO (OLAP)

El diseño de las tablas explica el ambiente de prueba que existe, siendo las tablas utilizadas las que se muestran a continuación:

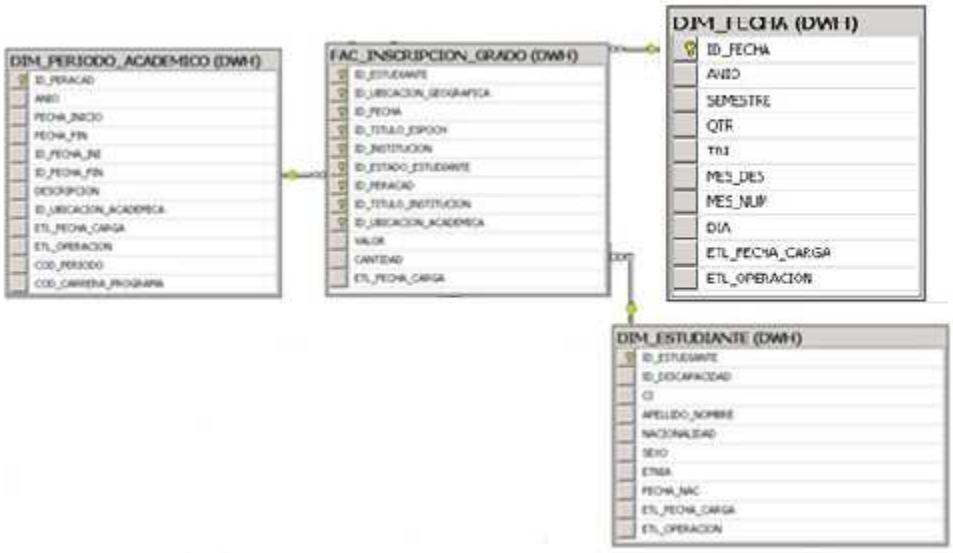


FIGURA IV. 47: Diseño OLAP

Elaborado por: Autora

4.7.3.4.1. SENTENCIA SQL SELECT UTILIZADA EN EL ESCENARIO OLAP CON LA QUE SE REALIZAN LAS PRUEBAS DE LA TECNICA DE CALIDAD DE SCRIPT SQL

Las sentencia SQL que se utilizará será el primer script utilizado y para demostrar la técnica se perfeccionará el script, demostrando en las pruebas el antes y el después de la técnica implementada.

El primer script es el siguiente:

```
SELECT A.[ID_ESTUDIANTE]
    ,B.APELLIDO_NOMBRE
    ,C.DESCRIPCION
    ,D.DIA
  FROM [SII_ESPOCH_DEF].[DWH].[FAC_INSCRIPCION_GRADO] A
  INNER JOIN DWH.DIM_ESTUDIANTE B ON A.ID_ESTUDIANTE=B.ID_ESTUDIANTE
  INNER JOIN DWH.DIM_PERIODO_ACADEMICO C ON C.ID_PERACAD=C.ID_PERACAD
  INNER JOIN DWH.DIM_FECHA D ON A.ID_FECHA=D.ID_FECHA
  WHERE A.ID_FECHA>=33602 AND A.ID_FECHA<=40236
```

El mismo script pero aplicando la técnica, es decir, tomando en cuenta las funciones agregadas de SQL SERVER y ordenando los JOINs de acuerdo a la cantidad de datos, siendo así de mayor a menor, en el siguiente script se puede observar la aplicación de la técnica:

```
SELECT A.[ID_ESTUDIANTE]
    ,B.APELLIDO_NOMBRE
    ,C.DESCRIPCION
    ,D.DIA
  FROM [SII_ESPOCH_DEF].[DWH].[FAC_INSCRIPCION_GRADO] A
  INNER JOIN DWH.DIM_ESTUDIANTE B ON A.ID_ESTUDIANTE=B.ID_ESTUDIANTE
  INNER JOIN DWH.DIM_FECHA D ON A.ID_FECHA=D.ID_FECHA
  INNER JOIN DWH.DIM_PERIODO_ACADEMICO C ON C.ID_PERACAD=C.ID_PERACAD
  AND A.ID_FECHA BETWEEN 33602 AND 40236
```

4.7.3.4.2. TABLAS DE DATOS DE RESULTADO DE LA PRUEBA DE LA TECNICA DE CALIDAD DE SCRIPT SQL

En la siguiente tabla se muestran los resultados de las pruebas con la utilización de la sentencia SELECT.

TABLA XLVIII

Tabla de resultados de la prueba de la técnica de calidad de script SQL

SENTECIA SQL SELECT						
CANTIDAD DE DATOS	**ESCENARIO OLAP (SIN TÉCNICAS)**			**ESCENARIO OLAP (CON TÉCNICAS)**		
	TIEMPO DE EJECUCIÓN DE LA CONSULTA	**LECTURA LÓGICA DE PÁGINAS**	**COSTO DE LA CONSULTA**	**TIEMPO DE EJECUCIÓN DE LA CONSULTA**	**LECTURA LÓGICA DE PÁGINAS**	**COSTO DE LA CONSULTA**
1000	0 seg	4	0,0397057	0 seg	4	0,0397057
100.000	14 seg	13.837	9,91565	13 seg	13.837	9,91565
1’000.000	126 seg (00:02:06)	78.848	51,9102	117 seg (00:01:57)	78.848	51,9102

Elaborado por: Autora

4.7.3.5. ÁNALISIS DE RESULTADOS DE CONSULTAS – TÉCNICA DE CALIDAD DE SCRIPT SQL

TIEMPO DE EJECUCIÓN

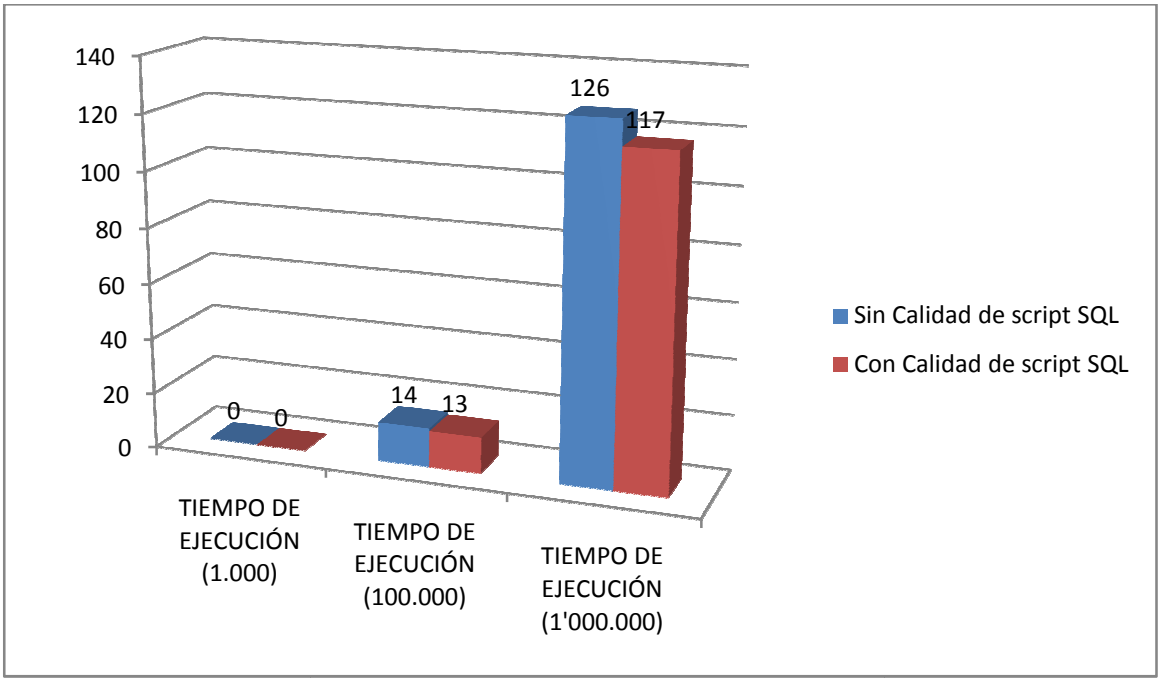


FIGURA IV. 48: Gráfico de resultado de tiempo de ejecución

El resultado de la Calidad de script SQL en el tiempo de ejecución de la consulta demuestra que esta técnica mejora el rendimiento, notándose más con mayor cantidad de datos, como es el caso de 117 segundos con una cantidad de 1’000.000 contra 126 segundos del ambiente sin la técnica.

Es importante notar que la técnica de Calidad de script SQL optimiza el tiempo de respuesta, siendo en si las demás mediciones iguales.

LECTURA LÓGICA DE PÁGINAS

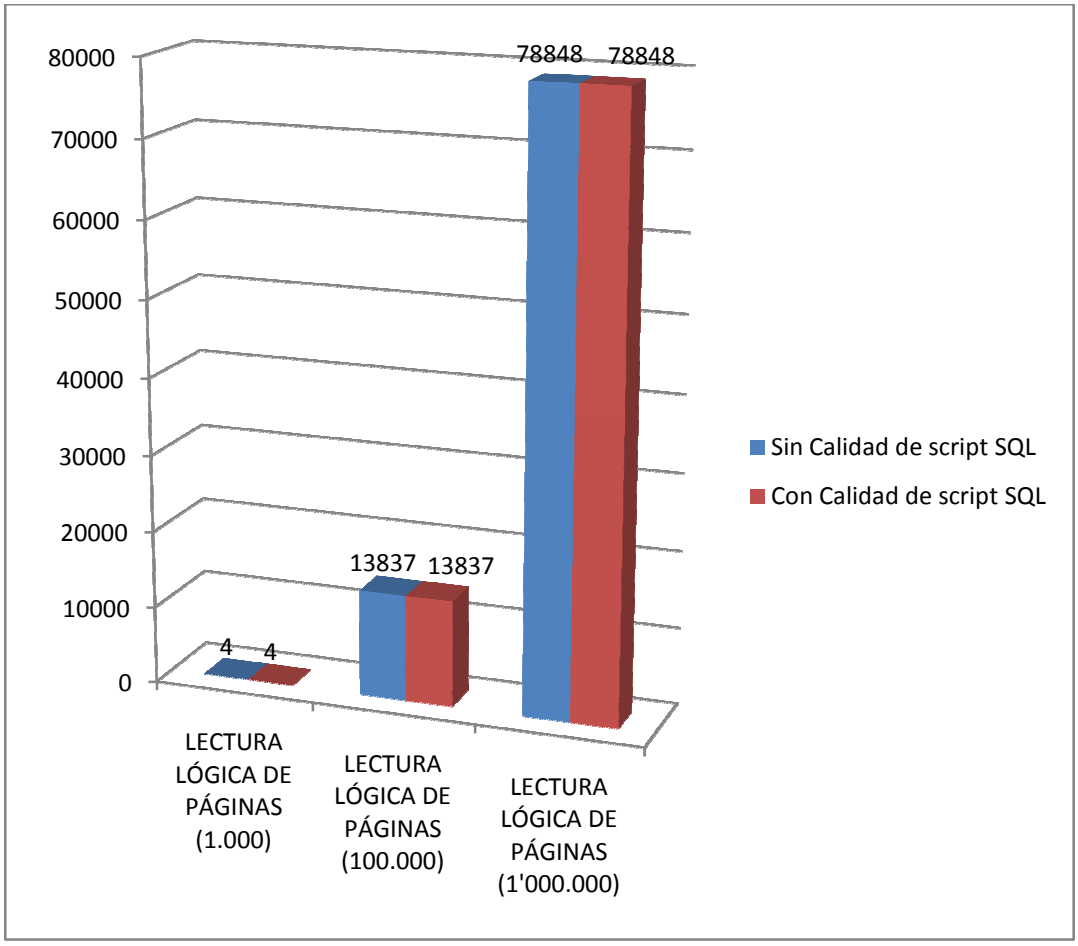


FIGURA IV. 49: Gráfico de resultado de la lecturas lógicas

En el caso de las lecturas lógicas el resultado es el mismo debido a que trabaja sobre los mismos datos sin variación de lecturas debido a que esta técnica optimiza el tiempo de respuesta sin cambiar la base de información.

COSTO DE LA CONSULTA

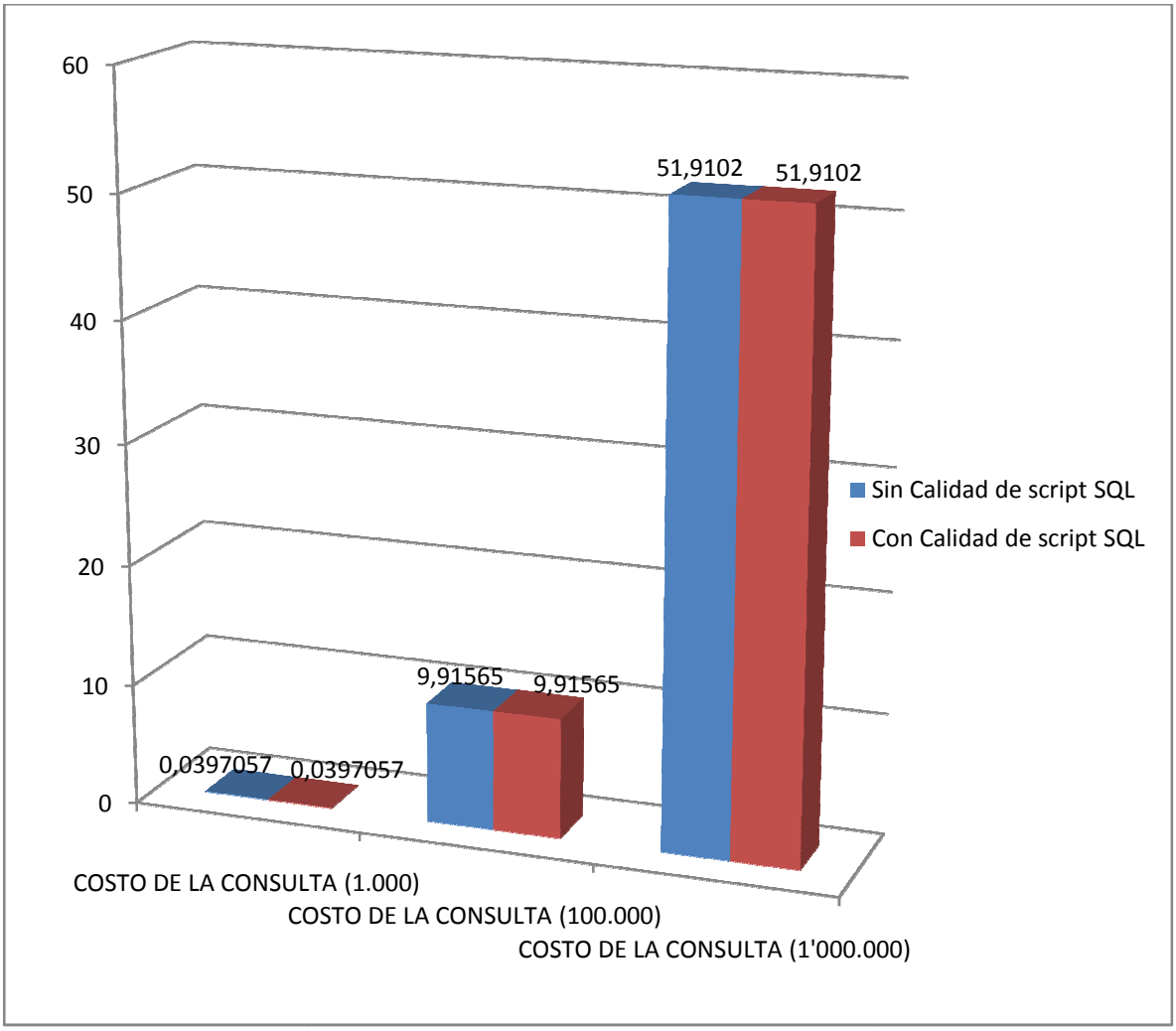


FIGURA IV. 50: Gráfico de resultado de la costo de la consulta

El costo de la consulta al igual que las lecturas lógicas conserva valores iguales debido a que el trabajo del CPU es el mismo en ambos casos.

4.7.3.6. CONCLUSIÓN

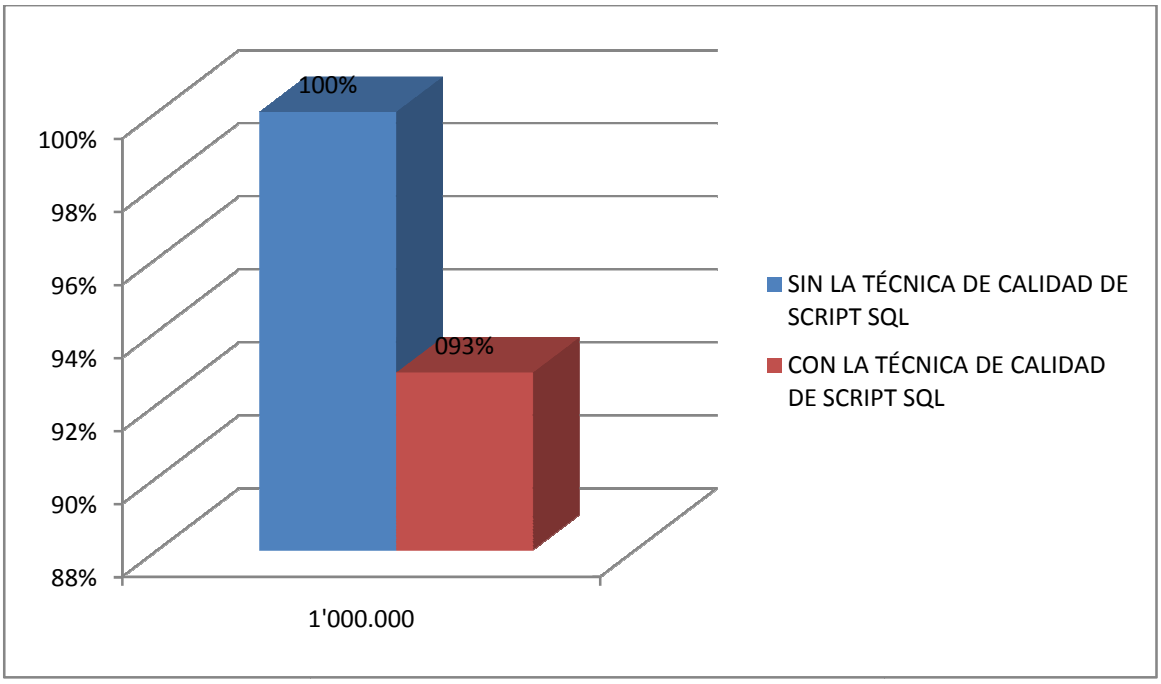


FIGURA IV. 51: Gráfico de resultado del uso de la Técnica de Utilización de Índices

Es importante notar que la técnica de Calidad de script SQL en realidad mejora el rendimiento en un **7.14%** en un ambiente OLAP, donde el tiempo de respuesta es el que demuestra la optimización, es por eso que es posible decir que la técnica de Calidad de script SQL optimiza el rendimiento en el data warehouse.

4.8. RESULTADOS DE OPTIMIZACIÓN DE LAS TECNICAS ANALIZADAS

Es importante analizar la optimización de cada una de las técnicas, para notar con mayor detalle la mejora de optimización que se produjo en el ambiente de consultas OLAP, es por eso que en el siguiente gráfico se muestran las técnicas con sus respectivos porcentajes de optimización.

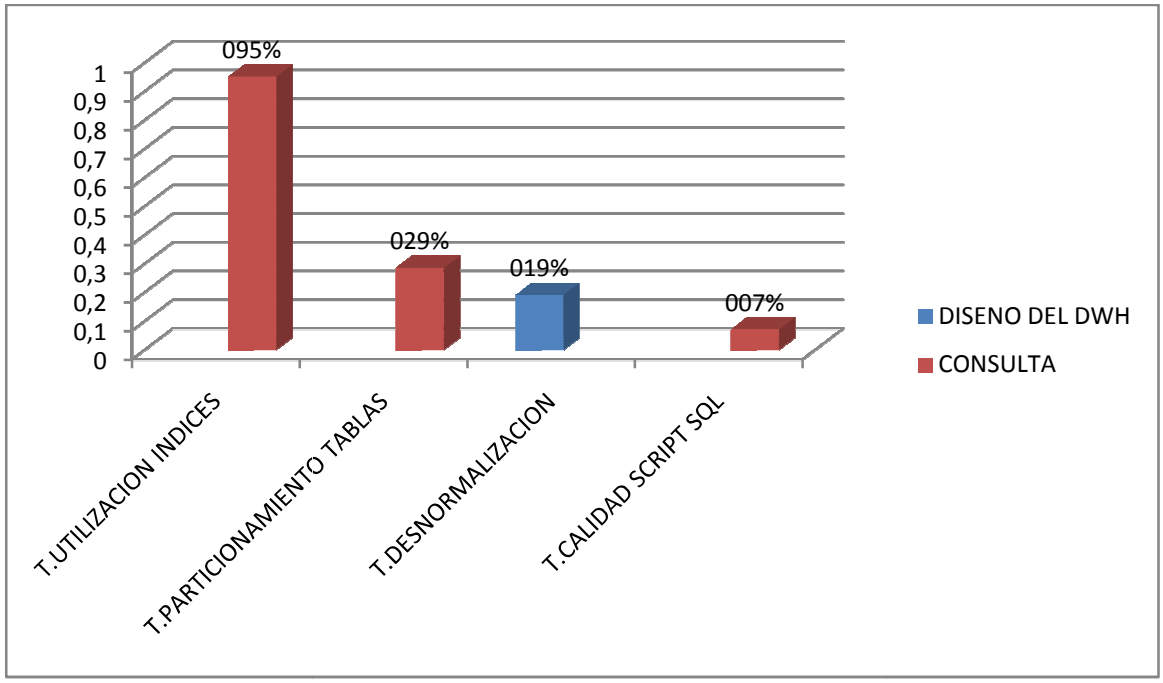


FIGURA IV. 52: Gráfico de resultado final de optimización del uso de Técnicas Tuning

Como la figura lo demuestra, la Técnica de Utilización de Índices es la que presenta mayor optimización, siendo este valor **95.15%**, es decir al crear índices correctos la optimización en un escenario OLAP es alto debido al nivel de optimización que crea esta técnica, además es importante mencionar que la técnica de particionamiento es la segunda técnica que más optimiza con un **28.57%**, seguida de la técnica de desnormalización a nivel del diseño del DWH, optimizando un **19.05%** y la técnica de calidad de script SQL optimiza el **7.14%** siendo esta técnica la de menor optimización.

CAPÍTULO V: GUÍA DE APLICACIÓN DE TÉCNICAS TUNING

5.1. INTRODUCCIÓN

Después de comprobar la mejora de rendimiento durante las pruebas realizadas bajo diferentes técnicas tuning, es importante crear una guía de aplicación de estas, creando un esquema de optimización en el Data Warehouse SII-ESPOCH.

Las técnicas implementadas que se aplicarán están bajo los siguientes criterios:

- DISEÑO DEL DATA WAREHOUSE
  - ✓ Técnica de Desnormalización
  - ✓ Técnica de Creación de FileGroups
- CONSULTAS
  - ✓ Técnica de Utilización de Índices

- ✓ Técnica de Calidad de Script SQL

5.2. ANTECEDENTES DEL PROYECTO SII-ESPOCH

Es evidente que la institución maneja una cantidad muy considerable de información pero el no poder acceder a ella adecuadamente es un problema eminente que dificulta la adecuada toma de decisiones

Propuesta presentada por el DESITEL denominó “Integración de los sistemas Financiero, RRHH, SAI de la ESPOCH mediante un data Warehouse para obtener información ágil y veraz que permitirá una acertada toma de decisiones y una mejora en la gestión de procesos internos de la ESPOCH”.

La UTP retoma esta propuesta y la amplia para lograr una solución integral frente a la notoria necesidad de contar con un Sistema de Información Estadística, que permita gestionar información institucional de relevancia, de forma ágil, confiable, oportuna, que sirva de soporte para una adecuada toma de decisiones dentro de la ESPOCH.

5.3. JUSTIFICACIÓN DEL PROYECTO SII-ESPOCH

La creación del Proyecto SII-ESPOCH realizará las siguientes actividades:

- Gestionar los recursos de información de manera más rápida, estratégica y efectiva, ya que es indispensable
- (SII-ESPOCH) se constituirá en una herramienta valiosa de apoyo a los directivos institucionales para la toma de decisiones
- Mejoras dentro de los procesos institucionales
- Tiempos rápidos de respuesta
- Optimización de recursos
- Fortalecimiento de las capacidades analíticas y de planificación

- Eficiente manejo de información que permita la toma oportuna de decisiones.

5.4. INDICADORES DEL SISTEMA ACADÉMICO QUE SE OBTENDRÁN DEL SISTEMA DE TOMA DE DECISIONES

1. Número de estudiantes inscritos por facultad, escuela, carrera, asignatura y centros, institucional, por periodo académico y por años
2. Número de estudiantes admitidos por sexo, procedencia (provincia, cuidad, colegio) por facultad, escuela, carrera, asignatura y centros, institucional, por periodo académico y por años.
3. Número de estudiantes por facultad, escuela, carrera, asignatura y centros, institucional, por periodo académico y por años
4. Número de estudiantes por facultad, escuela, carrera, asignatura y centros, institucional, por periodo académico y por años
5. Número de estudiantes por sexo por facultad, escuela, carrera, asignatura y centros, a nivel institucional, por periodo académico y por años
6. Número de estudiantes por país, provincia, cantones, por facultad, escuela, carrera, y centros, asignatura, institucional, por periodo académico y por años
7. Número de estudiantes procedentes de instituciones fiscales, particulares, fiscomisionales por facultad, escuela, carrera, y centros, asignatura, nivel institucional, por periodo académico y por años.
8. Número de estudiantes que repiten (por nota y por asistencia) por asignatura, institucional, por periodo académico y por años
9. Número de estudiantes que desertan por asignatura, institucional, por periodo académico y por años.
10. Número de estudiantes aprobados por, asignatura por periodo académico y por años
11. Índice de repitencia por asignaturas por periodo académico y por años
12. Índice de deserción por asignaturas, por periodo académico y por años
13. Número de estudiantes que han finalizado el pensum por facultad, escuela, carrera y centros , nivel institucional, por periodo académico y por años
14. Número de estudiantes graduados por facultad, escuela, carrera y centros, nivel institucional, por periodo académico y por años

15. Número de estudiantes matriculados en sistemas de graduación: tesis, seminarios, otras alternativas
16. Tasa de titulación por facultad, escuela, carrera y centros, institucional, por periodo académico y por años.
17. Número de docentes por facultades, escuelas, centros, institucional por periodos académicos y por años
18. Número de docentes por país, provincia, cantón distribuidas por facultades
19. Número de estudiantes por docente por periodo académico y por años.
    Nota: se debe considerar discapacidades, etnias y grupos etarios

5.5. DESARROLLO DE LA GUÍA DE APLICACIÓN DE TÉCNICAS TUNING

En esta guía de aplicación de técnicas tuning se describirán todas las técnicas implementadas que mejoran el rendimiento en el Data Warehouse. A continuación se detallarán los procesos realizados, creando un esquema de optimización en el Data Warehouse SII-ESPOCH.

DISEÑO DEL DATA WAREHOUSE

5.3.1. TÉCNICA DE DESNORMALIZACIÓN

Como se analizó en el Capítulo III la desnormalización consta de un conjunto de técnicas que se aplican al diseño para mejorar el rendimiento de las consultas. Estas técnicas de desnormalización crean redundancia de datos. Además es importante mencionar que la desnormalización es un punto claro de optimización en un data warehouse, porque en resumen lo que hace es disminuir uniones de tablas (JOINs) al momento de unir información que se encuentra relacionada, como se demostró en las pruebas realizadas.

Las técnicas de desnormalización aplicadas en el diseño del Data Warehouse de la ESPOCH son:

- Datos duplicados
- Claves subrogadas

5.3.1.1. DATOS DUPLICADOS

El uso de datos duplicados se utiliza cuando los datos de atributos individuales se introducen de forma redundante para reducir el número de tuplas que deben ser revisadas en una consulta. Es decir las tablas contienen información que al aplicar normalización, serían otras tablas.

Durante el desarrollo del diseño del data warehouse se desnormalizaron algunas tablas como se muestra en el diseño del data warehouse SII-ESPOCH en la siguiente figura:

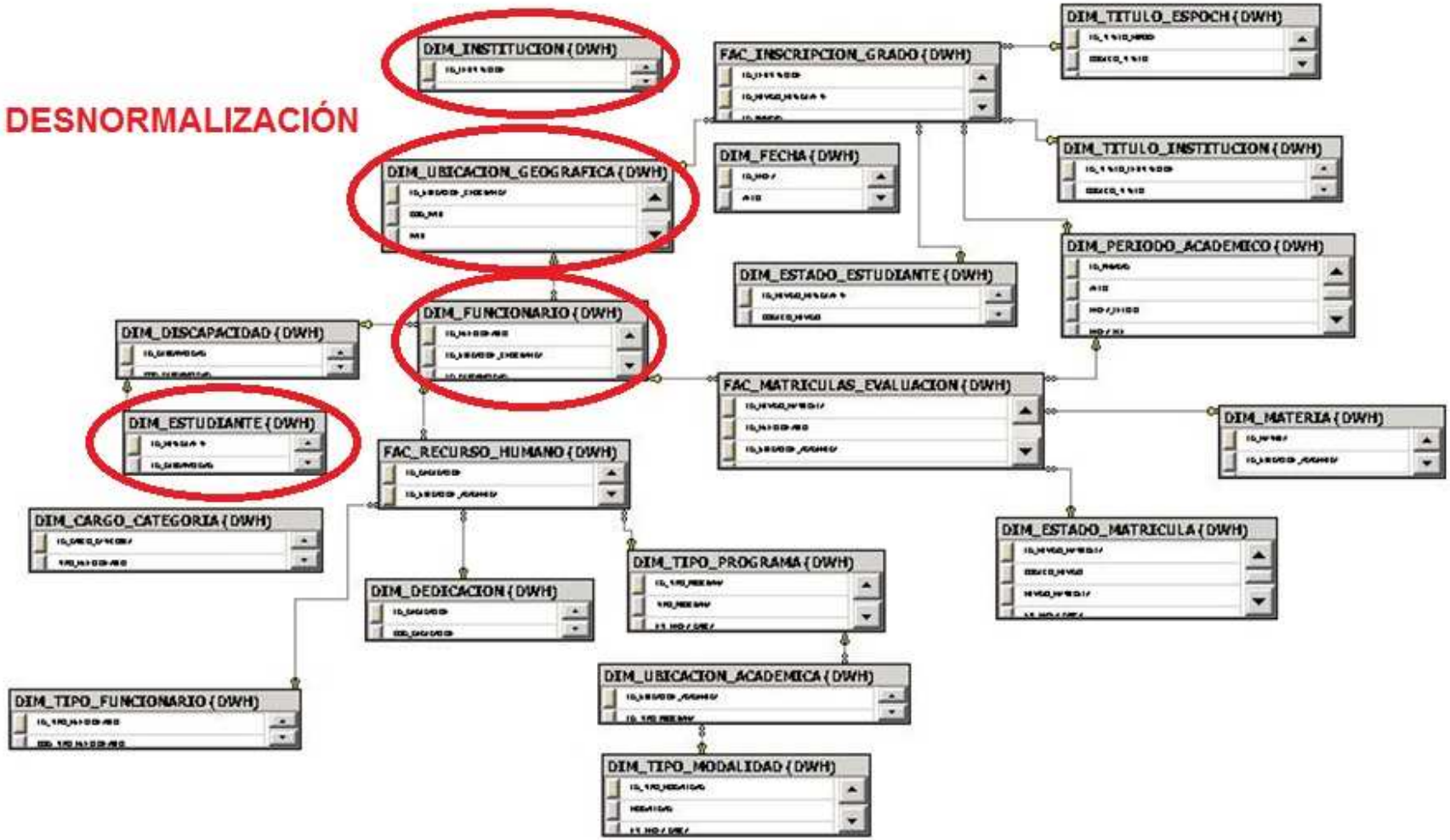


FIGURA V. 53: Diagrama del Data Warehouse SII-ESPOCH

5.3.1.2. TABLAS DESNORMALIZADAS

Las tablas desnormalizadas contienen la información completa, que en una base de datos OLTP serían otras tablas, es por eso que en las siguientes tablas se puede observar los campos que contienen información duplicada en todas las filas de las tablas, con la finalidad de obtener la información directamente de las tablas, evitando JOINs en las sentencias SQL.

Las tablas desnormalizadas con los campos que presentan la desnormalización son las siguientes:

FIGURA V. 54: Dimensión Estudiante Desnormalizada

FIGURA V. 55: Dimensión Funcionario Desnormalizada

FIGURA V. 56: Dimensión Ubicación Geográfica Desnormalizada

FIGURA V. 57: Dimensión Institución Desnormalizada

El uso campos dentro de las tablas desnormalizadas, como se muestran en las figuras (campos subrayados), permiten que el data warehouse funcione de mejor manera, siendo el tiempo de respuesta el que demuestre la eficacia de la técnica de desnormalización.

5.3.1.3. CLAVES SUBROGADAS

El uso de claves subrogadas es la técnica que más se utiliza en el diseño del DWH, ya que una clave subrogada es un identificador único que se asigna a cada registro de

una tabla de dimensión (DIM). Estas claves no tiene ningún sentido específico en el negocio, son de tipo numérico (INT) e autoincremental.

En todas las dimensiones del data warehouse se usaron claves subrogadas, identificándose con ID_ seguido del nombre de la tabla, como se muestra en la siguiente figura:

FIGURA V. 58: Dimensión Fecha con clave subrogada (ID_FECHA)

Además de facilitar la navegación entre tablas a la hora de crear instrucciones JOINs, es también importante destacar que el tamaño de una clave subrogada de tipo INT, permite ahorrar espacio y al crear la Técnica de utilización de índices también permite evitar que los arboles B sean muy profundos.

5.3.2. TÉCNICA DE CREACIÓN DE FILEGROUPS

Los grupos de archivos (FILEGROUPS) se denominan colecciones de archivos y se utilizan para simplificar la colocación de datos y tareas administrativas, tales como copia de seguridad y restauración.

El propósito de crear un nuevo FileGroup es poder acceder a datos e índices de forma paralela, esto optimiza las consultas lo cual mejora el tiempo de respuesta.

NOTA: Para obtener un rendimiento óptimo mediante creación de diferentes filegroups se debe por lo menos conseguir que el espacio asignado a DATOS y a INDICES se encuentre en discos físicos distintos

5.3.2.1. CREACIÓN DE FILEGROUPS

Lo primero que se tiene que hacer es crear los FileGroups (Grupos de Archivos) para luego poder asociar estos grupos de archivos a los archivos de la base de datos. La creación se realiza de la siguiente manera por medio de interfaz gráfica:

1. Se da click derecho sobre la base de datos que se está trabajando
2. Damos click sobre Propiedades (Properties)
3. Aparece la ventana y se selecciona FileGroups (ubicado al lado izquierdo superior de la ventana)

FIGURA V. 59: Selección de FileGroups

4. Sobre esta opción se ingresa el grupo de archivos INDEX como se muestra en la siguiente figura:

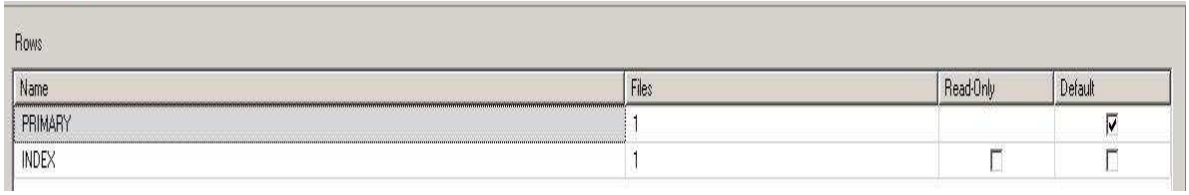


FIGURA V. 60: FileGroups (Grupos de Archivos)

5. Nuevamente sobre las opciones del lado izquierdo de la ventana, se elige la opción Files

FIGURA V. 61: Selección de Files

6. Sobre esta opción se crean los archivos y se les asigna un grupo de archivos, es decir los datos con su extensión .mdf y grupo de archivos PRIMARY, los índices con su extensión .ndf y grupo de archivos INDEX, además del log con su extensión .ldf, además es importante indicar la ubicación de almacenamiento de cada uno de estos como se muestra en la figura.

Database files:

Logical Name	File Type	Filegroup	Initial Size (MB)	Autogrowth	Path	File Name
DWH_SII-ESPOCH	Rows Data	PRIMARY	58	By 1 MB, unrestricted growth	E:\DWH_DB\DATA	DWH_SIIESPOCH.mdf
INDEX	Rows Data	INDEX	3	By 10 percent, unrestricted growth	E:\DWH_DB\INDEX	INDEX.ndf
DWH_SII-ESPOCH_log	Log	Not Applicable	895	By 10 percent, restricted growth to 209...	E:\DWH_DB\LOG	DWH_SIIESPOCH_log.ldf

FIGURA V. 62: Files (Archivos)

7. Finalmente aceptamos las modificaciones y la creación de FileGroups y files está concluida y lista para contribuir a la optimización.

5.3.3. TÉCNICA DE UTILIZACIÓN DE ÍNDICES

La creación de índices útiles es la acción más importante que se puede realizar para mejorar el rendimiento. El tipo y número de índices, así como las columnas que se van a indizar se deben seleccionar cuidadosamente gracias a una profunda comprensión de las necesidades del usuario y de los propios datos. Los índices resultan útiles cuando se va a consultar la tabla o realizar modificaciones en los datos.

La optimación de índices es una faceta importante en cualquier base de datos o almacén de datos. Su función principal es reducir el tiempo de respuesta al disminuir la actividad de entrada y salida, tanto a disco como a memoria.

No solamente la falta de índices, sino los índices mal diseñados constituyen las principales fuentes de atascos en aplicaciones de un almacén de datos. El diseño eficaz de los índices tiene gran importancia para conseguir un buen rendimiento de un almacén de datos, tanto en búsquedas de datos como en escritura.

Por tal motivo los índices creados en el almacén de datos SII-ESPOCH debido al correcto análisis de las consultas que se obtendrán, maximizan el rendimiento reflejado en el tiempo de respuesta. Los índices SQL SERVER utilizados son los índices agrupados y no agrupados.

Sólo se puede crear un índice agrupado en una tabla ya que este índice ordena físicamente los datos y este orden sólo se puede realizar una vez, los índices no agrupados ordenan los datos lógicamente y es aconsejable no tener demasiados índices ya que podrían empeorar el rendimiento, es por eso que un número promedio es 4 índices no agrupados por tabla.

5.3.3.1. INDICES CREADOS EN EL DWH SII-ESPOCH

Los índices creados en el Data Warehouse son los siguientes:

TABLA XLIX

Índices creados en el DWH SII-ESPOCH

NOMBRE DE LA TABLA	NOMBRE DEL ÍNDICE	DESCRIPCIÓN DEL ÍNDICE	LLAVE DEL ÍNDICE
DWH.FAC_INSCRIPCION_GRADO	PK_FAC_INSCRIPCION_GRADO	clustered, unique, primary key located on PRIMARY	ID_ESTUDIANTE, ID_UBICACION_GEOGRAFICA, ID_FECHA, ID_TITULO_ESPOCH, ID_INSTITUCION, ID_ESTADO_ESTUDIANTE, ID_PERACAD, ID_TITULO_INSTITUCION, ID_UBICACION_ACADEMICA
DWH.FAC_MATRICULAS_EVALUACION	PK_FAC_MATRICULAS_EVALUACION	clustered, unique, primary key located on PRIMARY	ID_ESTADO_MATRICULA, ID_FUNCIONARIO, ID_UBICACION_ACADEMICA, ID_FECHA, ID_PERACAD, ID_ESTUDIANTE, ID_MATERIA, ID_MATRICULA, ID_PARALELO
DIM_ESTUDIANTE	INDEX_ESTUDIANTES	nonclustered located on INDEX	CI
DSA.Oasis_Estudiantes	INDEX_ESTUDIANTES	nonclustered located on INDEX	strCedula
DSA.Oasis_Grados	INDEX_GRADOS	nonclustered located on INDEX	strCedEstud
DSA.Oasis_Matriculas	INDEX_MATRICULAS	nonclustered located on INDEX	sintCodigo
DSA.Oasis_Materias_Asignadas	INDEX_MATERIAS_ASIGNADAS	nonclustered located on INDEX	sintCodMatricula
DWH.DIM_MATERIA	INDEX_MATERIA	nonclustered located on INDEX	COD_MATERIA

Fuente: Ejecutado en SQL SERVER 2005 mediante el comando (sp_helpindex 'nombre_de_la_tabla')

Elaborado por: Autora

5.3.3.2. CRITERIOS DE OPTIMIZACIÓN PARA LA CREACIÓN DE ÍNDICES

Es importante analizar el porqué de cada índice creado, es por eso que a continuación se detallan los principales criterios tomados en cuenta:

- Crear un índice agrupado (Cluster) en todas las fact, cuyas columnas llaves del índice sea la PK de la tabla, además estas columnas deben ser de tipo INT porque reducen el espacio de almacenamiento de datos y de los índices al momento de crearlos.
- Las columnas más utilizadas en las consultas deben ser parte de los índices, de índices no agrupados que permitan al momento de crear una consulta en el que este campo esté en el WHERE, el tiempo de respuesta sea más rápido gracias al índice creado.
- Analizar si el uso excesivo de índices perjudican el rendimiento de escritura.
- Es recomendable crear índices no compuestos, es decir índices que contienen una sola columna de la tabla, esto hará que la consulta que utiliza este campo ocupe el índice creado por la misma.
- Analizar el uso de índices cubiertos que faciliten las consultas al estar cubiertas pero que no perjudiquen el rendimiento de escritura.
- Considerar los tables Scan presentes en los planes de ejecución, ya que debido a la presencia de estos, es importante tomar la decisión correcta de creación de índices.
- Eliminar índices creados y no utilizados para mejorar el espacio de almacenamiento.

5.3.3.3. SCRIPTS DE CREACIÓN DE ÍNDICES

Para la creación de los índices del Data Warehouse del SII-ESPOCH, se utilizó los siguientes scripts:

ÍNDICES UTILIZADOS PARA LA CARGA DE LA FACT DWH.FAC_INSCRIPCION_GRADO

```
CREATE NONCLUSTERED INDEX INDEX_ESTUDIANTES ON
DSA.Oasis_Estudiantes (strCedula)
INCLUDE (strCodCiudadProc)
ON [INDEX]

CREATE NONCLUSTERED INDEX INDEX_ESTUDIANTES ON
DWH.DIM_ESTUDIANTE(CI)
INCLUDE (ID_ESTUDIANTE)
ON [INDEX]

CREATE NONCLUSTERED INDEX INDEX_GRADOS ON
DSA.Oasis_Grados(strCedEstud)
INCLUDE (strCodTit,strCodInt)
ON [INDEX]
```

ÍNDICES UTILIZADOS PARA LA CARGA DE LA FACT DHW.FAC_MATRICULAS_EVALUACION

```
CREATE NONCLUSTERED INDEX INDEX_MATRICULAS ON
DSA.Oasis_Matriculas(sintCodigo)
INCLUDE (COD_CARRERA_PROGRAMA,strCodPeriodo,strCodEstud)
ON [INDEX]

CREATE NONCLUSTERED INDEX INDEX_MATERIAS_ASIGNADAS ON
DSA.Oasis_Materias_Asignadas(sintCodMatricula)
INCLUDE (strCodPeriodo,strCodPensum)
ON [INDEX]

CREATE NONCLUSTERED INDEX INDEX_MATERIA ON
DWH.DIM_MATERIA(COD_MATERIA)
INCLUDE (COD_PENSUM,ID_UBICACION_ACADEMICA)
ON [INDEX]
```

Es importante tomar en cuenta que los índices se crearon en el grupo de archivos INDEX, para contribuir a la optimización del rendimiento.

5.3.3.4. CRITERIOS DE MANTENIMIENTO DE ÍNDICES

Además de la adecuada creación de índices se debe tomar en cuenta su mantenimiento, que debe ser periódico, ayudando a que su optimización este en los puntos más altos. Por tal motivo es importante tomar en cuenta los siguientes criterios:

- Es importante tomar en cuenta que los índices creados tienen un equilibrio de rendimiento de escritura y lectura, pero en algunas ocasiones en el que los

costos de escritura sean muy altos, es aconsejable eliminar los índices (DROP INDEX nombre_tabla.nombre_índice), ejecutar las sentencias de acción y volver a crearlos, siendo los scripts muy útiles para facilitar esta tarea.

- Es importante reconstruír los índices periódicamente (DBCC DBREINDEX [nombre_tabla.nombre_índice])
- Manejar planes de mantenimiento (estadísticas de SQL Server)
- La reorganización de índices también actualiza las estadísticas sobre la tabla reorganizada.
- Consecuentemente el día de semana que reorganizamos todos los índices, no es necesario actualizar las estadísticas.
- Reorganización de Índices completa, una vez por semana en días no laborables y de aquellas tablas más importantes una vez en la semana.

5.3.3.4.1. Reorganización de Índices

Las tablas que contienen índices al ser actualizadas o por inserción de nuevos datos, generan fragmentación de estos índices. Estas fragmentaciones conllevan a la pérdida de performance al acceder a ellas.

La instrucción DBCC DBREINDEX reorganiza el índice de una tabla o todos los índices definidos para una tabla. La reorganización se realiza dinámicamente sin necesidad de conocer la estructura de la misma o las restricciones que ella tenga. Por lo tanto no es necesario conocer si una tabla tiene clave primaria o si esta clave es única y además pertenece a algún índice, ya que la reorganización no necesita eliminar y recrear éstas restricciones para realizar su trabajo.

La sintaxis de esta instrucción es:

```
DBCC DBREINDEX
   (    'basededatos.dueño.nombre_de_tabla'
          [ , índice
             [ , fillfactor ]
          ]
   )    [ WITH NO_INFOMSGS ]
```

Fillfactor es el porcentaje de espacio de página destinado a ser ocupado. El valor definido reemplaza al que fue generado en el momento de la creación del índice. Si se quiere mantener el valor original, entonces se utiliza el valor 0.

WITH NO_INFOMSGS se suprimen los mensajes generados en la ejecución.

Una de las formas de utilizarlo es, escribir un script con una sentencia DBCC DBREINDEX por cada tabla que se necesita reorganizar y agendarlas en forma periódica mediante un trabajo de mantenimiento dentro de algún horario disponible.

Por lo tanto, la recomendación será: elegir las tablas más accedidas y/o actualizadas, y reorganizarlas una vez entre semana. Para reorganizar todas las tablas que contengan índices se utiliza el mismo concepto, pero dentro de un procedimiento que recorra todas la tablas de la base y las reorganice, sin necesidad que escribamos todas la tablas que contiene la base de datos.

(En ANEXO 2 se adjunta el script de mantenimiento de actualización de índices)

Cabe indicar que esta reorganización se la puede hacer también a través de una interfaz gráfica, en la carpeta índices que posee cada tabla, es necesario dar click derecho sobre esta carpeta y escoger la opción Reorganizar todo, como se muestra en la siguiente figura:

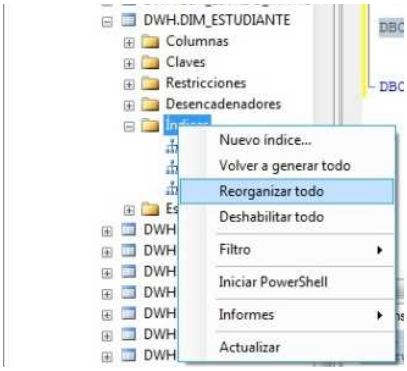


FIGURA V. 63: Reorganizar todo desde la interfaz gráfica

5.3.4. TÉCNICA DE CALIDAD DE SCRIPT SQL

Es importante analizar la calidad del SCRIPT SQL ya que el rendimiento proyectado en el tiempo de respuesta al ejecutar las consultas puede ser mejorado, tomando en cuenta criterios que son importantes y ayudan a obtener el mejor rendimiento.

5.3.4.1. CRITERIOS DE LA CALIDAD DE SCRIPT SQL

Es importante tomar en cuenta los siguientes criterios:

- El orden de los JOINs es importante en el momento de realizar las consultas en la base de datos, se debe encabezar el JOIN con la tabla que contenga la mayor cantidad de registros, seguida con las tablas que contengan menor cantidad de registros.
- En el momento de realizar un JOIN entre tablas se debe realizar la juntura primero por los campos de tipo número si las tablas lo poseen, luego seguir con los campos tipo string, y finalmente por los campos tipo fecha, esto mejorará el desempeño de la consulta.
- Utilizar las funciones agregadas de SQL SERVER

- Tratar de no utilizar cursores en la construcción de Procedimientos almacenados y Funciones, debido a la lentitud con que estos operan, en su defecto tratar de utilizar consultas de tipo SELF JOINS, es decir realizar consultas sobre una misma tabla cuando se trata de realizar un recorrido de registros.

5.3.5. BLOQUEOS DE CONSULTAS SQL

Los problemas de bloqueos en SQL SERVER se realizan cuando se ejecutan consultas, esto se puede solucionar mediante el nivel de aislamiento de transacciones con el que se configure el servidor SQL

Para controlar como afectan los bloqueos a las transacciones, se puede modificar el nivel de aislamiento a través de la instrucción SET TRANSACTION ISOLATION LEVEL <opción>. Con esta instrucción se controla como interpreta la transacción los bloqueos existentes y como genera nuevos bloqueos con sus operaciones de lectura/escritura.

Es por eso que en SQL Server 2005 se puede configurar el nivel de aislamiento SNAPSHOT que asegura la coherencia de los datos para toda la transacción, ya que activa el versionado de fila. Las instrucciones que se ejecuten en la transacción no verán las modificaciones de datos efectuadas por otras transacciones, en su lugar reciben una "copia coherente" de cómo estaban los datos al comienzo de la transacción.

El nivel de aislamiento SNAPSHOT no está disponible de forma predeterminada. La opción Snapshot usa el versionado de filas para proporcionar consistencia de lectura a nivel de transacciones. Las operaciones de lectura no piden bloqueos. Cuando leen filas modificadas por otra transacción, recuperan la versión de la fila que existía antes de que la transacción comenzara.

Se activa con la opción ALLOW_SNAPSHOT_ISOLATION, que por defecto está desactivada.

5.3.6. CACHÉ DEL SERVIDOR SQL SERVER

Es importante saber que las consultas que se ejecutan en el servidor se almacenan en la caché de procedimientos, desde el cual se puede tomar en cuenta dos consideraciones:

1. Al almacenar en caché las consultas realizadas, el tiempo de respuesta al presentar la cantidad de datos que devuelve la consulta, es menor, debido a que después de la primera ejecución, este fue almacenado. Contribuyendo a la mejora de rendimiento.

2. Al tener almacenado información en caché se va a llegar a un punto en el que se deberá liberar espacio, aunque es preciso recalcar que procesos muy antiguos y sin uso si se van descartando. Para poder borrar esta información se pueden ejecutar el siguiente comando:

```
DBCC FREESESSIONCACHE
```

Limpia la caché de consultas distribuidas.

Es importante recordar que el Servidor de SQL guardará una copia duplicada de cada procedimiento que se accede por más de un usuario). Por tal motivo es recomendable realizar un mantenimiento periódico, para que de esta manera se pueda prevenir problemas en el rendimiento del servidor.

5.3.7. ESTADÍSTICAS SQL SERVER

En el servidor de base de datos, la base de datos principal a veces no crea automáticamente las estadísticas a medida que las fuese necesitando. Esto claramente impacta el funcionamiento del servidor ya que SQL Server precisa de ellas para funcionar al máximo.

Cada vez que se crea o se actualizan los índices sobre alguna tabla de SQL Server, viene por defecto instruyéndole al servidor que actualice las estadísticas.

Existen dos formas tradicionales de ver las estadísticas. Una de ellas es consultando el catálogo interno de SQL Server o la otra es a través de la interfaz gráfica asociada al plan de ejecución de una consulta.

Para controlar las estadísticas de SQL se deben conocer los catálogos internos. El catálogo donde se almacena la información de los índices y estadísticas de una tabla se llama sysindexes.

Para consultarlo se debe entonces filtrar la información por el id de la tabla “nombre”, como se muestra en la siguiente consulta:

```
select * from sysindexes where id = object_id('nombre de la tabla')
```

Siendo el comando ejecutado el siguiente:

```
select * from sysindexes where id = object_id('DWH.DIM_ESTUDIANTE')
```

Como resultado de esta consulta, se puede obtiene:

TABLA L:

Estadística de la tabla DIM_ESTUDIANTE

id	status	first	indid	Root	Minlen	keycnt	groupid	Dpages	Used	rowcnt	Rowmodctr	xmaxlen	maxirow	OrigFillFactor	StatVersion	FirstIAM	impid	keys	name	statblob	maxlen	rows
150958041 6	0	0x630300000100	0	0x000000000000	31	0	1	538	835	35319	105957	295	0	0	0	0x640300000100	0	NULL	NULL	NULL	8000	35319
150958041 6	2050	0x040500000100	2	0x060500000100	13	2	1	66	68	35319	0	17	15	0	0	0x050500000100	0	NULL	PK_DIM_ESTUDIANTE	NULL	8000	35319
150958041 6	0	0xDD0600000100	3	0xDF0600000100	13	2	1	79	81	35319	0	17	22	0	0	0xDE0600000100	0	NULL	RELATIONSHIP_28_FK	NULL	8000	35319
150958041 6	8388672	NULL	4	NULL	0	1	0	0	0	0	0	0	NULL	0	0	NULL	0	NULL	_WA_Sys_00000003_59FA5E80	NULL	8000	0
150958041 6	0	0x200100000300	5	0x600100000300	13	2	2	145	147	35319	0	44	33	0	0	0x190000000300	0	NULL	INDEX_ESTUDIANTES	NULL	8000	35319

Fuente: Ejecutado en SQL SERVER 2005

El resultado indica que para la tabla nombre (id), hay creado índice no agrupado (indid) y que utiliza 145 páginas (dpages = 145).

Además se puede mencionar que las estadísticas pueden actualizarse o eliminarse manualmente a través de la interfaz gráfica o de consultas sql (drop statistics). Además, conviene saber que SQL Server se encarga de actualizarlas y eliminarlas cuando estima que es necesario, pero también puede agregarse una tarea de mantención que las actualice cada cierto tiempo.

Una columna que existe en sysindexes, es rowmodctr**.** Ésta lleva el registro de los cambios en la tabla, pero sólo para la o las columnas involucradas en la estadística, y cuando este contador llega a cierto límite, es momento de que sean actualizadas.

En sysindexes existe otra columna que es StatVersion, que refleja la cantidad de veces que la estadística ha sido actualizada. Después de cierta cantidad de actualizaciones (número no conocido), la estadística se elimina. En caso de necesitarse posteriormente, se vuelve a crear. Desde el punto de vista del rendimiento, crear un estadística de cero o actualizar una existente no es muy diferente.

5.3.7.1. PLANES DE MANTENIMIENTO: ACTUALIZACIÓN DE ESTADÍSTICAS

Dentro de las tareas habituales de Mantenimiento de las Bases de Datos se encuentran aquellas destinadas al control y respaldo de las mismas.

Pero también es necesario ejecutar trabajos de mantenimiento cuyos objetivos sean el de mantener la performance de las bases de datos y evitar su degradación.

Estos trabajos son independientes del estado de la base de datos. Puede ocurrir que a la base le falten estudios de optimización pero, al menos, se mantendrá la performance actual.

Si la base se encuentra optimizada, entonces más aún, son necesarios para evitar la degradación producto del uso continuo.

Cualquiera de estos trabajos deben realizarse fuera de línea por motivos de: alto consumo de recurso y bloqueo de las tablas en el momento de ejecución.

Por lo tanto se debe agendarlos en horarios, en donde las bases de datos se encuentren libres de cualquier actividad de producción.

La frecuencia de ejecución de estos trabajos depende básicamente de dos factores: tamaño de la base y tiempo libre para ejecutarlos.

Es aconsejable actualizar las estadísticas semanalmente.

5.3.7.1.1. Actualización de Estadísticas

Esta tarea permite actualizar la información sobre la distribución de los valores de las claves, para una o un grupo de estadísticas en una tabla o vista.

Esto podría realizarse en forma automática dado que, en las propiedades de la base de datos, existe la opción de Actualización de Estadísticas. Pero no es recomendable que esta opción se encuentre habilitada, ya que degrada la performance de la base de datos que se encuentra en producción y en uso diario. (La opción que debe estar habilitada es la de Crear Estadísticas Automáticamente, esta no consume recursos y es fundamental para mejorar la performance).

Es por ello que es conveniente realizar este trabajo fuera de línea, es decir en horarios libres de uso de la base.

Se puede utilizar para la actualización semanal (fin de semana), utilizando el valor de 100% para que se asegure que se recorran todos los datos que contienen todas las tablas.

Como esta tarea tomará muchas horas, dependiendo del tamaño de la base, no se puede utilizarla diariamente. Pero, de la misma forma que con la reorganización, podemos elegir las tablas más importantes y actualizar las estadísticas de las mismas en forma diaria y fuera de horario normal.

Para ello se utiliza la siguiente instrucción simplificada en su sintaxis:

```
UPDATE STATISTICS nombre_de_tabla WITH FULLSCAN
sp_updatestats
```

Con la opción FULLSCAN se asegura que todos los datos de la tabla sean recorridos, es equivalente al 100% definido en el párrafo anterior, con la diferencia que aquí se trata de una tabla o vista en lugar de todas.

Por lo tango se debe generar un script con una instrucción UPDATE STATISTICS por cada tabla que se necesita. Este script se debe ejecutar mediante un trabajo de mantenimiento y lo agendarlo con frecuencia diaria.

Como sugerencia general con el objetivo de mantener la performance se recomienda:

- Actualización de Estadísticas completa una vez por semana y de las tablas más importantes todos los días laborables.

En la base de datos DWH.SIIESPOCH, se hicieron manualmente las actualizaciones mediante el comando que actualiza todas las estadísticas de la base de datos:

```
sp_updatestats
```

(Ver en ANEXO 2 el resultado de la ejecución del comando)

(En ANEXO 2 se adjunta el script de mantenimiento de actualización de estadísticas)

5.3.8. TÉCNICA DE AGREGACIÓN (BO)

Las técnicas de agregación aplicadas al data warehouse son muy importantes a la hora de obtener tiempos de respuesta aceptables al momento de manejar reportes, siendo una solución selectiva y evitar manejar toda la información almacenada en una Fact.

5.3.8.1. TABLAS AGREGADAS

El data warehouse tiene, y debe tener, todo el detalle de información en su nivel atómico. Así, por ejemplo las ventas estarán detalladas por fecha, cliente, producto y punto de venta. Rápidamente se puede notar que se está trabajando con un volumen muy importante de información debido a que en algunos casos es habitual encontrarse con data warehouses con miles de millones de registros.

Sin embargo, la mayoría de consultas no necesitan acceder a tanto detalle. La persona que administra los productos puede estar interesada en los totales de venta de sus productos mes a mes, mientras que el administrador de área consulta habitualmente la evolución de ventas de sus zonas.

Incluso con el uso de índices, la compresión de las tablas, o con una inversión millonaria en hardware, estas consultas habituales deberían leer, agrupar y sumar decenas de millones de registros, lo que repercutiría directamente en el tiempo de respuesta y en el descontento de los usuarios.

La solución ante estas situaciones pasa siempre para la preparación de tablas agregadas. Las tablas agregadas sumarizan los indicadores de las tablas de detalle a un nivel superior. Por ejemplo, las ventas podrían precalcularse a nivel mensual, o por cliente, o por producto. De esta manera, las consultas típicas del administrador del producto o el administrador de área podrían ejecutarse en pocos segundos, sin necesidad de acceder a la tabla de ventas detalladas.

La existencia de estas tablas agregadas debe ser completamente transparente para el usuario de negocio. Es decir, tanto el administrador del área como el administrador del

producto trabajarán con el indicador "Ventas", y la herramienta Business Intelligence hará el resto.

Una de las cosas más complicadas es definir las tablas agregadas necesarias. De nada sirve crear muchos agregados si estos no se utilizan. Es necesario conocer las consultas habituales de los usuarios. Lo que no se debe hacer es mezclar hechos de diferente granularidad en una misma tabla de hechos.

Calcular los agregados en la misma tabla de ventas es un error muy grave e injustificable. Aunque puede parece adecuado en algunas situaciones, será sin duda una fuente de problemas e incoherencias futuras. Este tipo de construcciones erróneas suelen aparecer ante la falta de funcionalidad de las herramientas BI, lo que obliga al técnico a crear modelos extraños.

Se ha visto este tipo de tablas (que mezclan información a diferente nivel de detalle) en sistemas operacionales, donde resulta prohibitivo calcular los totales en el momento de generación del informe.

5.3.8.2. CRITERIOS CONSIDERADOS EN LA CREACION DE TABLAS AGREGADAS

Tomando en consideración las consultas que tienen un rendimiento poco óptimo al ejecutarlas en los reportes en SAP Business Objects se deben crear técnicas de agregación para las tablas de hechos con mayor número de registros para permitir mejorar el rendimiento.

Como criterios de optimización de Business Objects se tiene:

- Tratar de generar las medidas calculadas en la base de datos, para disminuir los cálculos en el reporte
- Crear tablas agregadas en las FACTs que tengan gran cantidad de registros

- Inclusión de técnicas de agregación en el Universo, en las tablas agregadas en el universo, produciendo técnicas de agregación que Business Objects Designer posee, para esto se utiliza la función Aggregate Aware propia de la herramienta.

5.6. COMPROBACIÓN DE LA HIPÓTESIS

Para poder comprobar la hipótesis “La comparación de Técnicas de Tuning permitirá implementar un esquema de optimización de rendimiento del Data Warehouse del SII-ESPOCH”, es importante demostrar la optimización obtenida en el Data Warehouse SII-ESPOCH, después de la aplicación de las técnicas tuning, este resultado se refleja en la siguiente tabla:

TABLA LI:

Resultados de la aplicación de Técnicas Tuning en el DWH SII-ESPOCH

TAREAS	TIEMPO DE EJECUCIÓN SIN TECNICAS TUNING	TIEMPO DE EJECUCIÓN CON TECNICAS TUNING
CARGA FACT DWH.FAC_INSCRIPCION_GRADO	1096 seg (00:18:16)	93 seg (00:01:33)
CARGA FACT DWH.FAC_MATRICULAS_EVALUACION	945 seg (00:15:45)	125 seg (00:02:05)

Elaborado por: Autora

En la figura que se muestra a continuación se puede observar el porcentaje de optimización de cada tarea en el Data Warehouse SII-ESPOCH.

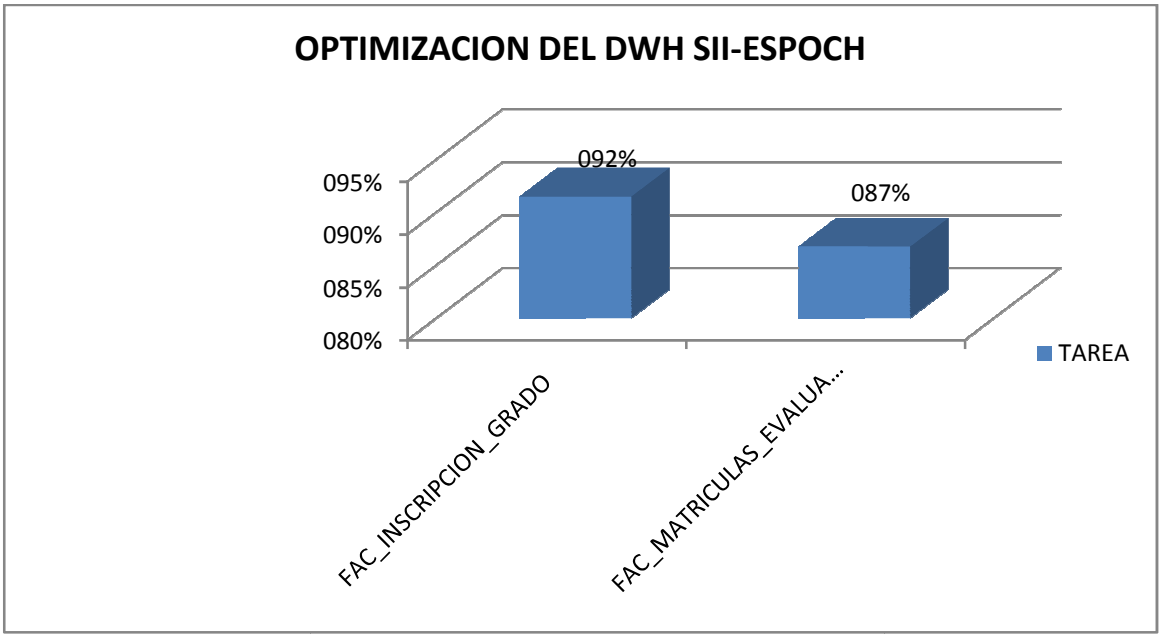


FIGURA V. 64: Gráfico final de resultado de optimización del uso de Técnicas Tuning en el DWH SII-ESPOCH

5.6.1. CONCLUSIÓN

Al mostrar en la tabla de resultados la optimización del Data Warehouse SII-ESPOCH, se puede demostrar que si fue posible realizar la comparación de Técnicas de Tuning que permitieron implementar un esquema de optimización de rendimiento del Data Warehouse del SII-ESPOCH, obteniendo un porcentaje promedio de optimización del **89,14%**.

CONCLUSIONES

- Al analizar las técnicas para realizar el Tuning en un Data Warehouse, se determinaron técnicas a nivel del Diseño del Data Warehouse y técnicas a nivel de las Consultas, siendo la Técnica de Desnormalización y Creación de FileGroups las técnicas del Diseño del DWH y las técnicas de Utilización de Índices, Particionamiento de Tablas y Calidad de Script SQL las técnicas a nivel de Consultas, siendo estas las que se aplicaron en Microsoft SQL Server, para mejorar el rendimiento de un sistema de toma de decisiones.

- Se determinó las capacidades que posee SQL SERVER 2005 (Propietario) tanto en prestaciones de Optimización y Herramientas BI, demostrando con un total de **87/100** puntos el cumplimiento de los requerimientos que comparados con el DBMS Open Source MYSQL 5.1 (MYISAM) que obtuvo el **55/100** puntos de cumplimiento, se mostró que SQL SERVER 2005 tiene las prestaciones correctas para el proyecto SII-ESPOCH, siendo una fortaleza para este.

- Se establecieron los parámetros de medición, siendo el tiempo de ejecución medido en segundos el más importante, además se agregaron lecturas lógicas y costo de la consulta para determinar el trabajo realizado por el DBMS, todas estas pruebas se realizaron sobre un escenario OLAP, para determinar la mejora del rendimiento en el DWH, según la aplicación de cada técnica.

- Al comparar el mejoramiento de rendimiento entre las técnicas tuning, reflejado en el tiempo de respuesta, se determinó que la Técnica que mejor optimización alcanzó, es la Técnica de Utilización de Índices con un **95,15%** de mejora, además que la Técnica de Particionamiento de Tablas optimizó un **28,57%**, la Técnica de Desnormalización mejoró el rendimiento un **19,05%** y por último la Técnica de Calidad de Script SQL optimizó un **7,14%** el Data Warehouse.

- Se desarrolló una Guía de las Técnicas Tuning en el Data Warehouse del proyecto SII-ESPOCH, en el que se implementaron las técnicas bajo diferentes criterios siendo la Técnica de Desnormalización y la Técnica de Creación de FileGroups dentro del Diseño del Data Warehouse y la Técnica de Utilización de Índices también la Técnica de Calidad de Script SQL dentro del criterio CONSULTAS, las técnicas aplicadas en el DWH, además se crearon planes de

mantenimiento tanto a nivel de índices y estadísticas de SQL SERVER, para el funcionamiento correcto de performance del servidor.

- La optimización de rendimiento alcanzado en el Data Warehouse SII-ESPOCH después de la aplicación de las Técnicas Tuning, se demuestra en el porcentaje de optimización que se obtuvo en las cargas de las FACTS y en la presentación del reporte, siendo para la carga de la FACT DWH.FAC_INSCRIPCION_GRADO el **91,51%** de mejoramiento y para la FACT el DWH.FAC_MATRICULAS_EVALUACION **86,77%** alcanzado en la optimización.

RECOMENDACIONES

- Del hardware del servidor también depende la rapidez en la obtención de la información, por tal motivo es importante analizar el procesador, la memoria RAM y el almacenamiento, considerando que un Data Warehouse se maneja históricos de información e índices, por tal motivo se debe considerar esta capacidad de almacenamiento.
- Durante la implementación de la Técnica de Utilización de Índices es importante analizar el rendimiento de escritura y lectura, por tal motivo se debe tomar un punto medio, pero también considerar que al trabajar sobre un Data Warehouse (Escenario OLAP) el tiempo de respuesta (rendimiento de lectura) es lo que realmente importa.
- Al utilizar índices en abuso no se mejora el rendimiento, más bien podría empeorarse.
- Se debe ejecutar un plan de mantenimiento semanal de índices y estadísticas SQL SERVER, de tal manera se asegura la integridad de la información y el performance del servidor SQL.
- Si se maneja gran cantidad de registros en las FACT del DWH, sería importante manejar la Técnica de Particionamiento de Tablas, por tal motivo se debe analizar la versión del DBMS SQL SERVER que existe, ya que no todas las versiones de SQL SERVER soportan este servicio, es recomendable utilizar la versión ENTERPRISE de MICROSOFT.

RESUMEN

Se realizó un análisis de las técnicas Tuning bajo diferentes criterios, como Diseño del Data Warehouse y Consultas, que permite determinar las técnicas óptimas para el Data Warehouse SII-ESPOCH, realizando optimización de rendimiento reflejado en el tiempo de respuesta al realizar una consulta en un escenario OLAP.

Para el desarrollo y estudio de las técnicas Tuning se utilizó MICROSOFT SQL SERVER 2005, siendo la Técnica de Utilización de Índices la más eficiente con un 95,15% de optimización, seguida de la Técnica de Particionamiento de Tablas con un 28,57%, Técnica de Desnormalización con 19,05% y por último la Técnica de Calidad de Script SQL con 7,14%, mejorando el rendimiento del sistema de toma de decisiones.

La comparación de Técnicas Tuning permitió crear un esquema de optimización de rendimiento en el Data Warehouse SII-ESPOCH, en el que se implementó las técnicas bajo diferentes criterios, además se crearon planes de mantenimiento tanto a nivel de índices y estadísticas de SQL SERVER, para el funcionamiento correcto de performance del servidor; para el uso adecuado de las técnicas es importante brindar un correcto mantenimiento.

SUMMARY

An analysis about the Tuning Techniques was made under different criteria such as Data Warehouse Design and Consultations, which permits to determine optimum techniques for Data Warehouse SII-Espoch; Doing optimization of performance which is reflected in the answer time upon doing a poll in an OLAP scenery.

MICROSOFT SQL SERVER was used for the developing and study of Tuning Techniques, being the Index Using Technique the most efficient with 95.15 percentage of optimization. Next was the Table Breaking Technique with 28.57 percentage. After comes the Dinormalization with 19.05%. Finally, Script SQL Quality Technique with 7.14 percentage. This leads to an improvement of the taken decision system.

The comparison of the Tuning Techniques permitted to create a scheme of performance optimization to the Data Warehouse SII-Espoch, in which the techniques were implemented under different criteria. Furthermore, plans were created for maintenance either for index and statistic levels of SQL SERVER or proper working of server performance. For an adequate usage of the techniques is important to offer a correct maintenance.

GLOSARIO DE TÉRMINOS

ÁRBOL-B: Los árboles-B ó B-árboles son estructuras de datos de árbol que se encuentran comúnmente en las implementaciones de bases de datos y sistemas de archivos. Los árboles B mantienen los datos ordenados y las inserciones y eliminaciones se realizan en tiempo logarítmico amortizado.

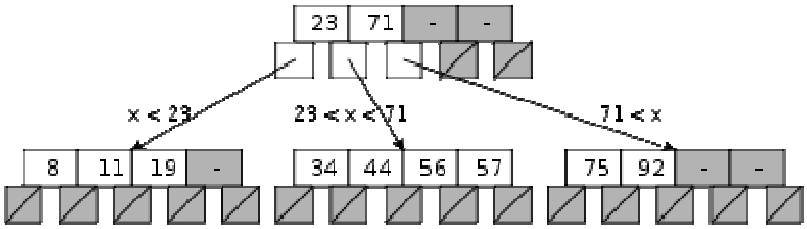


FIGURA V. 65: Ejemplo de árbol B

La idea tras los árboles-B es que los nodos internos deben tener un número variable de nodos hijo dentro de un rango predefinido. Cuando se inserta o se elimina un dato de la estructura, la cantidad de nodos hijo varía dentro de un nodo. Para que siga manteniéndose el número de nodos dentro del rango predefinido, los nodos internos se juntan o se parten. Dado que se permite un rango variable de nodos hijo, los árboles-B no necesitan rebalancearse tan frecuentemente como los árboles binarios de búsqueda auto-balanceables, pero por otro lado pueden desperdiciar memoria, porque los nodos no permanecen totalmente ocupados. Los límites superior e inferior en el número de nodos hijo son definidos para cada implementación en particular. Por ejemplo, en un árbol-B 2-3 (A menudo simplemente llamado árbol 2-3), cada nodo sólo puede tener 2 ó 3 nodos hijo.

Un árbol-B se mantiene balanceado porque requiere que todos los nodos hoja se encuentren a la misma altura.

Los árboles B tienen ventajas sustanciales sobre otras implementaciones cuando el tiempo de acceso a los nodos excede al tiempo de acceso entre nodos. Este caso se da usualmente cuando los nodos se encuentran en dispositivos de almacenamiento secundario como los discos rígidos. Al maximizar el número de nodos hijo de cada nodo interno, la altura del árbol decrece, las operaciones para balancearlo se reducen,

y aumenta la eficiencia. Usualmente este valor se coloca de forma tal que cada nodo ocupe un bloque de disco, o un tamaño análogo en el dispositivo.

Los creadores del árbol B, Rudolf Bayer y Ed McCreight, no han explicado el significado de la letra B de su nombre. Se cree que la B es de balanceado, dado que todos los nodos hoja se mantienen al mismo nivel en el árbol. La B también puede referirse a Bayer, o a Boeing, porque sus creadores trabajaban en el Boeing Scientific Research Labs en ese entonces.

CLAVES SUBROGADAS: Una clave subrogada es un identificador único que se asigna a cada registro de una tabla de dimensión. Esta clave, generalmente, no tiene ningún sentido específico de negocio. Son siempre de tipo numérico. Preferiblemente, un entero autoincremental.

Existen varios motivos por los que se deben crear claves subrogadas, siendo los principales los siguientes:

- **Fuentes heterogéneas.** El DWH suele alimentarse de diferentes fuentes, cada una de ellas con sus propias claves, por lo que es arriesgado asumir un código de alguna aplicación en particular. ¿Qué ocurriría si en el futuro se añade información de una aplicación que tiene su propio maestro de ciudades? Seguro que se generará un problema, aparecerán ciudades que no existían en el maestro. Es difícil gestionarlas, por esta razón lo mejor es crear propias claves subrogadas desde el inicio del proyecto.
- **Cambios en las aplicaciones origen.** Puede ocurrir que cambie la lógica operacional de alguna clave que se hubiera supuesto única, o que siempre debería estar informada. Lo mejor es crear propias claves subrogadas desde el inicio del proyecto, porque no se sabe si todos los datos requeridos se obtendrán.
- **Rendimiento.** En la base de datos, ocupa menos espacio un entero que una cadena. Identificar una ciudad con 5 bytes, o una persona con 9 bytes es un desperdicio considerable de espacio. De hecho, no debe preocupar el espacio que ocupa, sino el tiempo que se pierde en leerlo. Es por eso que las claves subrogadas se las lleva a las tablas de hechos, por lo que cada código es

susceptible de repetirse cientos de millones de veces. ConIVene optimizarlo al máximo. Lo mejor es crear claves subrogadas desde el inicio del proyecto.

Por supuesto, también se pueden cometer errores al generar una clave subrogada, pero se debe sustituir cualquier clave física por una clave entera numerada secuencialmente desde 1 hasta N. [10]

CLIENTE/SERVIDOR: Esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

DATA MINING: La minería de datos (DM, Data Mining) consiste en la extracción no trivial de información que reside de manera implícita en los datos. Dicha información era previamente desconocida y podrá resultar útil para algún proceso. Es decir, la minería de datos prepara, sondea y explora los datos para sacar la información oculta en ellos.

Bajo el nombre de minería de datos se engloba todo un conjunto de técnicas encaminadas a la extracción de conocimiento procesable, implícito en las bases de

datos. Está fuertemente ligado con la supervisión de procesos industriales ya que resulta muy útil para aprovechar los datos almacenados en las bases de datos.

Las bases de la minería de datos se encuentran en la inteligencia artificial y en el análisis estadístico. Mediante los modelos extraídos utilizando técnicas de minería de datos se aborda la solución a problemas de predicción, clasificación y segmentación.

Básicamente, el data mining surge para intentar ayudar a comprender el contenido de un repositorio de datos. Con este fin, hace uso de prácticas estadísticas y, en algunos casos, de algoritmos de búsqueda próximos a la Inteligencia Artificial y a las redes neuronales.

De forma general, los datos son la materia prima bruta. En el momento que el usuario les atribuye algún significado especial pasan a convertirse en información. Cuando los especialistas elaboran o encuentran un modelo, haciendo que la interpretación que surge entre la información y ese modelo represente un valor agregado, entonces se refiere al conocimiento.[9]

DATA WAREHOUSING: Se entiende por Data Warehousing el proceso de extraer y filtrar datos de las operaciones comunes de la organización, procedentes de los distintos sistemas de información operacionales y/o sistemas externos, para transformarlos, integrarlos y almacenarlos en un depósito o almacén de datos (Data Warehouse) con el fin de acceder a ellos para dar soporte en el proceso de toma de decisiones de una organización. Es decir, la finalidad es convertir los datos operacionales en información relacionada y estructurada, homogénea y de mayor calidad, identificada convenientemente y que se mantenga en el tiempo, es decir, los datos más recientes no sustituyen a los precedentes, pero tampoco se acumulan de cualquier manera, sino que se suelen mantener con un mayor nivel de detalle los datos actuales, y de manera más agregada los datos anteriores. Se pretende crear un círculo virtuoso para la información.[9]

(EDA) / SQL: Es una familia de productos cliente / servidor que permiten el acceso basado en SQL para datos relacionales y no relacionales a través de plataformas

múltiples.

EDA / SQL proporciona una visión uniforme, relacional de datos, independientemente de dónde y cómo se almacenan los datos. Basado en una arquitectura de red abierta, la familia de productos soporta la mayoría de las arquitecturas de red, incluida la SNA de IBM, y DECnet Digital.

GRANULARIDAD: La granularidad de los esquemas internos es el grado de detalle de éstos en función del esquema lógico. Cuanto más fina es la granularidad (a nivel de campo de registro es más fina que a nivel de registro completo)se consigue mayor grado de independencia de los datos.

HEAP: Un heap es una estructura de datos que almacena la posición física en la que se almacenó cada nueva fila dentro de las páginas asignadas a la tabla.

INSTRUCCIONES DDL: Son las instrucciones que se usan para crear/modificar las estructuras en las que se almacenan los datos, por ejemplo: create, drop,alter.

INSTRUCCIONES DML: Son las instrucciones que se usan para consultar la Base de Datos y para modificar los datos que esta contiene, por ejemplo: select, insert, delete, update.

META DATA: El componente final del data warehouse es el de la meta data. De muchas maneras la meta data se sitúa en una dimensión diferente al de otros datos del data warehouse, debido a que su contenido no es tomado directamente desde el ambiente operacional.

La meta data juega un rol especial y muy importante en el data warehouse y es usada como:

- Un directorio para ayudar al analista a ubicar los contenidos del data warehouse.

- Una guía para el mapping de datos de cómo se transforma, del ambiente operacional al de data warehouse.
- Una guía de los algoritmos usados para la esquematización entre el detalle de datos actual, con los datos ligeramente resumidos y éstos, con los datos completamente resumidos, etc.

La meta data juega un papel mucho más importante en un ambiente data warehousing que en un operacional clásico.

La meta data contiene (al menos):

- La estructura de los datos
- Los algoritmos usados para la esquematización
- El mapping desde el ambiente operacional al data warehouse. [4]

MAINFRAMES: Es una computadora central, equipo grande o principal que es grande, potente y costosa usada principalmente por una gran compañía para el procesamiento de una gran cantidad de datos; por ejemplo, para el procesamiento de transacciones bancarias. Además realizan millones de operaciones por segundo y soportan un gran número de estaciones de trabajo (terminales).

PARALELISMO: El paralelismo es la forma en la cual varios cálculos pueden realizarse simultáneamente, basado en el principio de dividir los problemas grandes para obtener varios problemas pequeños, que son posteriormente solucionados en paralelo. El paralelismo ha sido empleado durante muchos años, sobre todo para el alto rendimiento.

REGISTRO DE TRANSACCIONES: El registro de transacciones (.log) es un componente esencial de la base de datos y, si se produce un error del sistema, podría ser necesario para volver a poner la base de datos en un estado coherente. El registro de transacciones nunca se debe eliminar o mover, a menos que se conozcan totalmente las implicaciones de esas acciones.

STAGING AREA: Repositorio intermedio usado para carga temporal de datos en un proceso ETL. Suele contener datos solo durante el proceso de carga del data warehouse. El sistema que permanece entre las fuentes de datos y el data warehouse con el objetivo de:

- Facilitar la extracción de datos (los procesos ETL) desde las fuentes de origen de carácter múltiple realizando un pre tratado.
- Realizar lo que se conoce como data cleansing (limpieza de datos).
- Mejorar la calidad de datos.
- Ser usado como cache de datos operacionales con el que posteriormente se realiza el proceso de Data Warehousing.
- Uso de la misma para acceder en detalle a información no contenida en el Data warehouse.

¿Cuándo se estable conveniente usar una staging area en un proyecto de Data Warehousing? Depende de la estructura informacional de nuestra organización, así mismo de si es necesario algunos de los puntos anteriores.

BIBLIOGRAFÍA

BIBLIOGRAFÍA GENERAL

[1] HOTEK, Mike, THERNSTÖM, Tobias, WEBER, Ann, and GrandMasters. Microsoft SQL Server 2008-Database Development. Redmond – United State, GrandMasters and Mike Hotek, 2009. P.P. 193-254

[2] LAHDENMÄKI, Tapio y LEACH, Michael. Relational Database Index Design and the Optimizers: DB2, Oracle, SQL Server, et al. New Jersey – United State, A John Wiley & Sons, 2005. P.P. 11-294.

[3] SHAPIRO Adam. Ajuste y Optimización del Rendimiento de MS SQL SERVER para Programadores, Primera Parte: Introducción a los Aspectos de Rendimiento. New Jersey – United State, Microsoft Corporation, 2008. P.P. 1-50

DIRECCIONES DE INTERNET

[4] BUSINESS INTELLIGENCE, DATA WAREHOUSE, DSS - 27 de Octubre de 2009

http://www.businessintelligence.info/dss/toma-decisiones-business-intelligence.html

(17/11/2010)

[5] BUSINESS OBJECTS, BO - 2010

http://www.sap.com/spain/solutions/sapbusinessobjects/large/intelligenceplatform/index.epx

(9/12/2010)

[6] ESNIPS, DESNORMALIZACIÓN – 14 de Noviembre 2007

http://www.esnips.com/doc/47e14117-7c53-4c1e-8e27-0984113f0c31/desnormalizacion

(01/02/2011)

[7] EUI, DESNORMALIZACIÓN - 2009

http://bd.eui.upm.es/DYOBD/OPCONS_texto.pdf

(01/02/2011)

[8] EXFORSYS, DISEÑO DE DATA WAREHOUSE - 2010

http://www.exforsys.com/tutorials/msas/data-warehouse-design-kimball-vs-inmon.html

(15/11/2010)

[9] GRIMPI IT BLOG, DIFERENCIAS ENTRE CLUSTERED Y NON CLUSTERED INDEX EN SQL SERVER - 8 de Marzo de 2008

http://grimpidev.wordpress.com/2008/03/08/diferencias-entre-clustered-y-non-clustered-index-en-sql-server/

(2/01/2011)

[10] GUILLESQL, PARTICIONAMIENTO DE TABLAS E ÍNDICES EN SQL SERVER (PARTITIONING) - 27 de Octubre de 2010

http://www.guillesql.es/Articulos/Particionamiento_tablas_indices_SQLServer_Partitioning.aspx

(03/02/2011)

[11] IESSANVICENTE, SQL SERVER VS MYSQL - 2009

http://www.iessanvicente.com/colaboraciones/sqlserver.pdf

(01/02/2011)

[12] INTERTRON, PERFORMANCE EN BASE DE DATOS - 2010

http://www.mug-it.org.ar/performance-bases-de-datos-agarcia-v16-nGVYAAZrb4S1tdmqK0uFHQRtkVotmqY18PhQ5jK4P3s%3D-archive.aspx

(31/01/2011)

[13] LA VISIÓN DE UN INGENIERO DE CAMPO, ¿CÓMO REDUCIR EL COSTO DE UN LOOKUP DE SQL SERVER? - 9 de Septiembre de 2007

http://msmvps.com/blogs/pmackay/archive/2007/09/09/post19.aspx

(03/02/2011)

[14] LA VISIÓN DE UN INGENIERO DE CAMPO, ESTADÍSTICAS DE SQL SERVER - 3 de Febrero de 2008

http://msmvps.com/blogs/pmackay/archive/2006/02/03/82438.aspx

http://msmvps.com/blogs/pmackay/archive/2006/02/06/estadisticas-de-sql-server-parte-2.aspx

(02/03/2011)

[15] MONOGRAFIAS, DATA WAREHOUSE - 2007

http://www.monografias.com/trabajos17/data-warehouse/data-warehouse.shtml

(08/11/2010)

[16] MONOGRAFIAS, DBMS- 2010

http://www.monografias.com//trabajos29/comparacion-sistemas/comparacion-sistemas.shtml

(31/01/2011)

[17] MSDN MICROSOFT, FILEGROUPS - 2009

http://msdn.microsoft.com/en-us/library/ms187087.aspx

(01/02/2011)

[18] MSSQLCITY, COMPARACION DE SQL VS MYSQL DIAPOSITIVAS - 2009

http://www.google.com.ec/url?sa=t&source=web&cd=1&ved=0CB8QFjAA&url=http%3A%2F%2Fwww.uaem.mx%2Fposgrado%2Fmcruz%2Fcursos%2Fmiic%2Fsqlserver2.ppt&rct=j&q=SQL%20SERVER%202005%20STANDARD%20VS%20MYSQL&ei=FUJHTcGpEsb3gAfp35XcAQ&usg=AFQjCNHB5HZ66TZUGkUK3i7_uxU-T4vNfg&cad=rja

(31/01/2011)

[19] SINNEXUS, DATA WAREHOUSE, BD OLAP Y OLTP - 2010

http://www.sinnexus.com/business_intelligence/datawarehouse.aspx

http://www.sinnexus.com/business_intelligence/olap_vs_oltp.aspx

(05/11/2010)

[20] SINNEXUS, DSS, BI - 2010

http://www.sinnexus.com/business_intelligence/sistemas_soporte_decisiones.aspx

http://www.sinnexus.com/business_intelligence/arquitectura.aspx

(8/12/2010)

[21] SQLMAXCONNECTIONS, DATA WAREHOUSE - 2007

http://www.sqlmax.com/dataw1.asp

(08/11/2010)

[22] SQL SERVER PERFORMANCE, OPTIMIZING SQL SERVER PERFORMANCE USING FILES AND FILEGROUPS - 2009

http://www.sql-server-performance.com/articles/per/optimize_filegroup_performance_p1.aspx

(01/02/2011)

[23] SWIK, COMPARACION DE SQL VS MYSQL - 2009

http://swik.net/MySQL/MySQL+vs+MS+SQL+Server

(01/02/2011)

[24] WIKIPEDIA, ADMINISTRACIÓN BD - 13 Noviembre 2010

http://es.wikipedia.org/wiki/Administrador_de_base_de_datos

(10/12/2010)

[25] WIKIPEDIA, DATA WAREHOUSE - 1ro Octubre 2010

http://es.wikipedia.org/wiki/Almac%C3%A9n_de_datos

http://en.wikipedia.org/wiki/Data_warehouse

(08/11/2010)

[26] WORDPRESS, ANÁLISIS DE ÍNDICES DE SQL SERVER - 3 Noviembre 2009

http://elarquitecto.wordpress.com/2009/11/03/analisis-de-indices-de-sql-server/

(03/02/2011)

[27] WORDPRESS, DATA WAREHOUSING, DATA WAREHOUSE, DATA MARTS - 7 de Octubre 2007

http://informationmanagement.wordpress.com/2007/10/07/data-warehousing-data-warehouse-y-datamart/

(15/11/2010)

[28] YAHOO RESPUESTAS, TUNING, ANALISIS DE RENDIMIENTO - 2007

http://es.answers.yahoo.com/question/index?qid=20070212073424AA4G5fD

http://espanol.answers.yahoo.com/question/index?qid=20100410193034AAQIDZ9

(9/12/2010)

ANEXOS

ANEXO 1

1. PLANES DE EJECUCIÓN

Los planes de ejecución fueron las herramientas mediante las cuales se hicieron las mediciones de las técnicas en el desarrollo de estas. Cada técnica se aplicó en un escenario OLAP por tal motivo estos planes son los de un antes de la técnica y un después de esta. A continuación se adjuntan los planes de ejecución de las pruebas.

1.1. COMPARACIÓN DEL DISEÑO DEL DATA WAREHOUSE

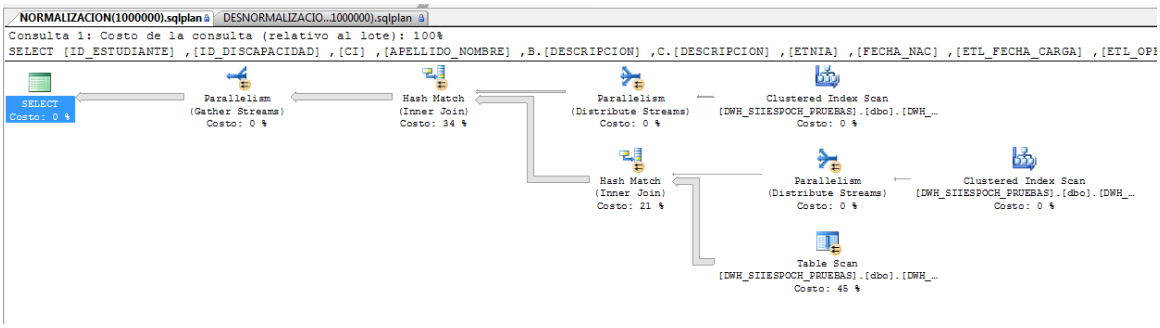
En el Diseño del Data Warehouse se aplicó la técnica de Desnormalización, cuyos planes de ejecución demostraron la optimización

1.1.1. TÉCNICA DE DESNORMALIZACIÓN

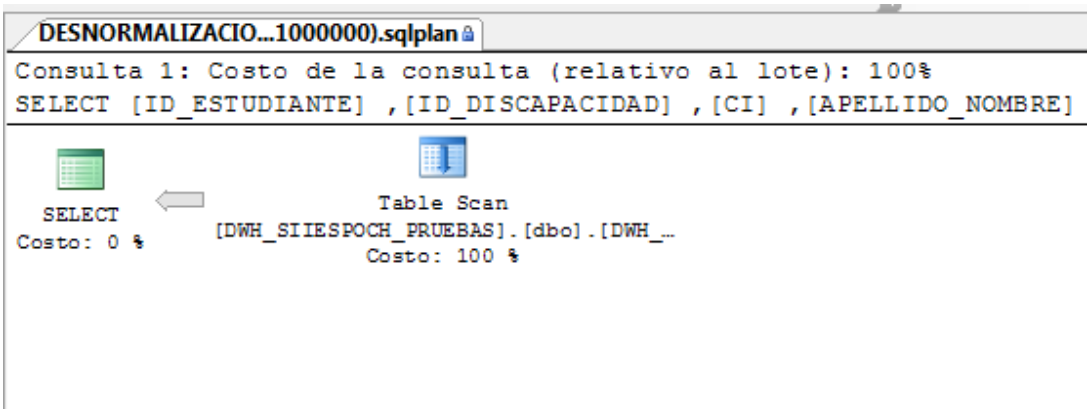
El ambiente de desarrollo de la técnica fue un ambiente OLTP (sin la técnica - antes) y un ambiente OLAP (con la técnica - después)

1.1.1.1. PLANES DE EJECUCIÓN DE LA TÉCNICA DE DESNORMALIZACIÓN

Plan de ejecución en el ambiente OLTP (ANTES) con 1'000.000 de datos



Plan de ejecución en el ambiente OLAP (DESPUÉS) con 1’000.000 de datos



1.2. COMPARACIÓN DE CONSULTAS

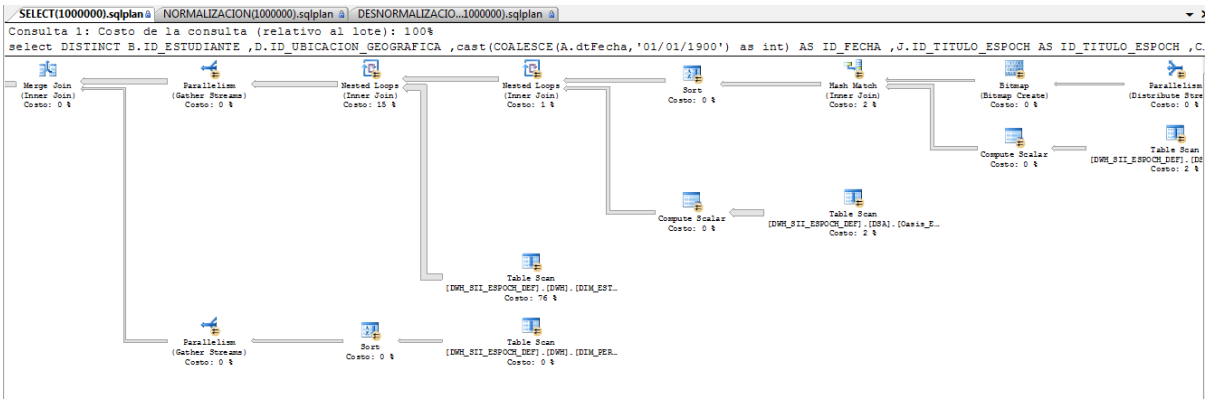
En las Consultas se aplicaron las técnicas de Utilización de Índices, técnica de Particionamiento de Tablas, Técnica de Calidad de Script SQL, cuyos planes de ejecución mejoraron el rendimiento

1.2.1. TÉCNICA DE UTILIZACIÓN DE ÍNDICES

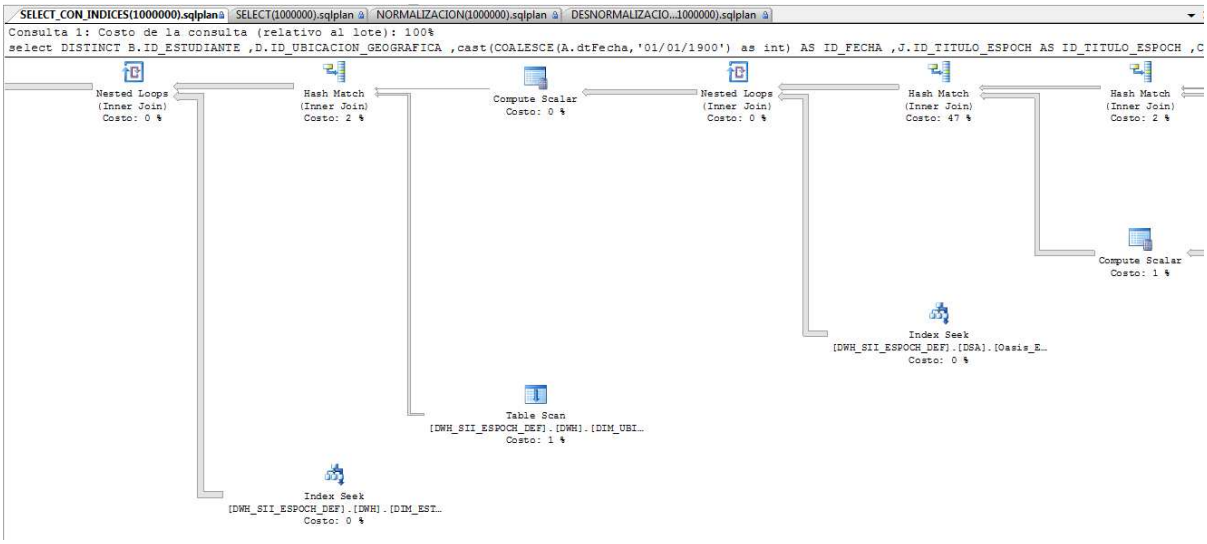
El ambiente de desarrollo de la técnica fue un ambiente OLAP (sin la técnica - antes) y un ambiente OLAP (con la técnica - después), hubieron dos casos, caso 1 SELECT, caso 2 UPDATE

1.2.2. PLANES DE EJECUCIÓN DE LA TÉCNICA DE UTILIZACIÓN DE ÍNDICES, CASO 1 SELECT

Plan de ejecución en el ambiente OLAP (ANTES) con 1’000.000 de datos

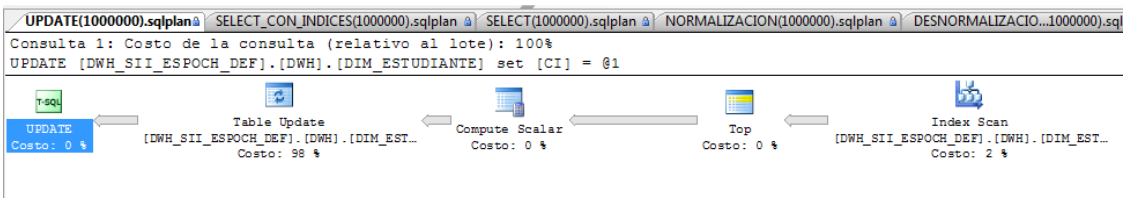


Plan de ejecución en el ambiente OLAP (DESPUÉS) con 1’000.000 de datos

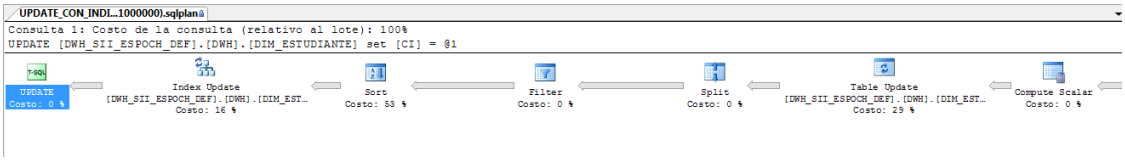


1.2.3. PLANES DE EJECUCIÓN DE LA TÉCNICA DE UTILIZACIÓN DE ÍNDICES, CASO 2 UPDATE

Plan de ejecución en el ambiente OLAP (ANTES) con 1’000.000 de datos



Plan de ejecución en el ambiente OLAP (DESPUÉS) con 1’000.000 de datos

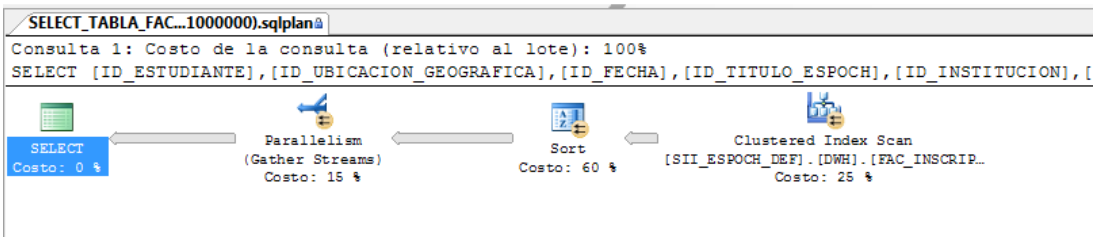


1.2.4. TÉCNICA DE PARTICIONAMIENTO DE TABLAS

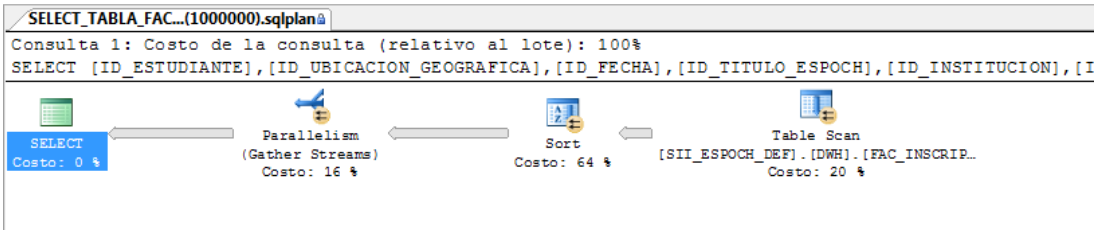
El ambiente de desarrollo de la técnica fue un ambiente OLAP (sin la técnica - antes) y un ambiente OLAP (con la técnica - después)

1.2.4.1. PLANES DE EJECUCIÓN DE LA TÉCNICA DE PARTICIONAMIENTO DE TABLAS

Plan de ejecución en el ambiente OLAP (ANTES) con 1’000.000 de datos



Plan de ejecución en el ambiente OLAP (DESPUÉS) con 1’000.000 de datos

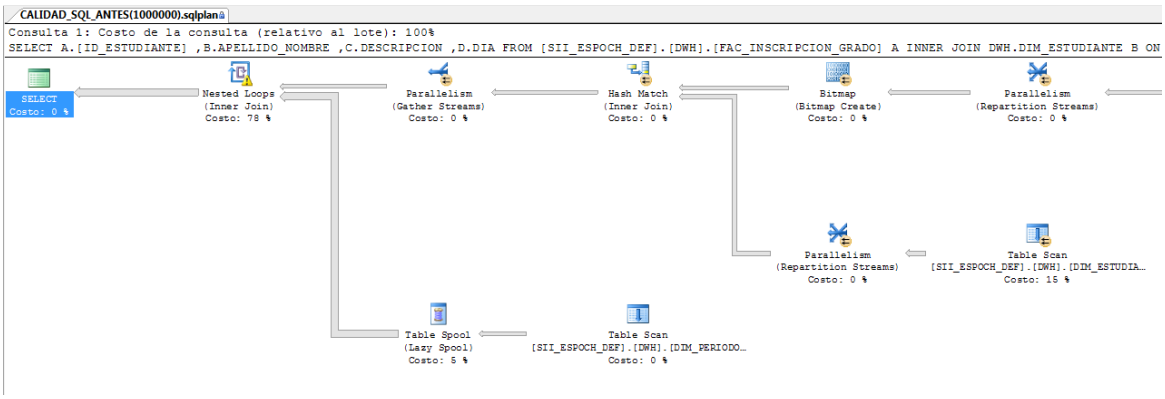


1.2.5. TÉCNICA DE CALIDAD DE SCRIPT SQL

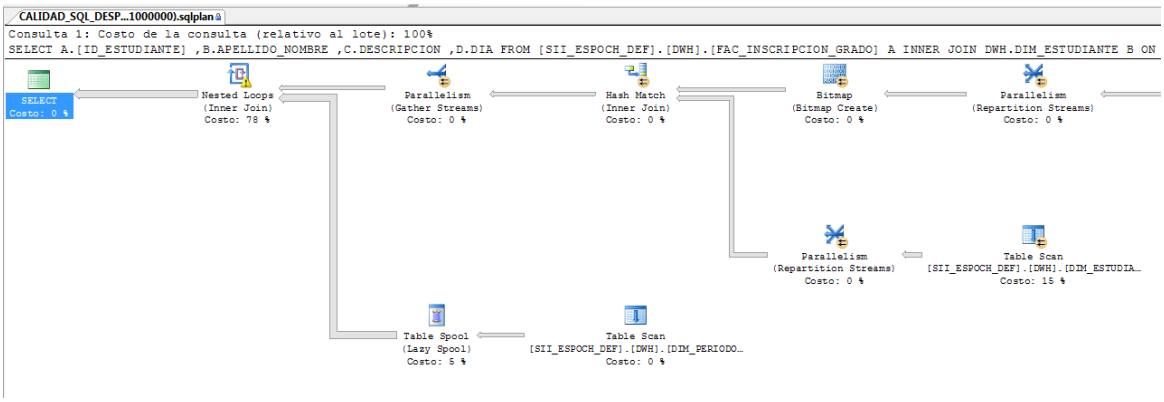
El ambiente de desarrollo de la técnica fue un ambiente OLAP (sin la técnica - antes) y un ambiente OLAP (con la técnica - después)

1.2.6. PLANES DE EJECUCIÓN DE LA TÉCNICA DE CALIDAD DE SCRIPT SQL

Plan de ejecución en el ambiente OLAP (ANTES) con 1’000.000 de datos

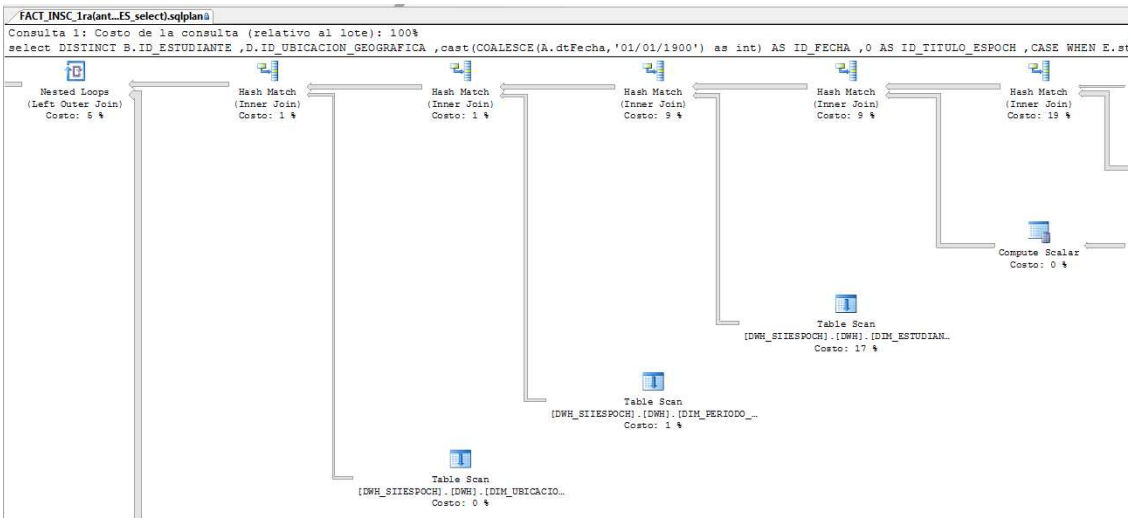


Plan de ejecución en el ambiente OLAP (DESPUÉS) con 1’000.000 de datos

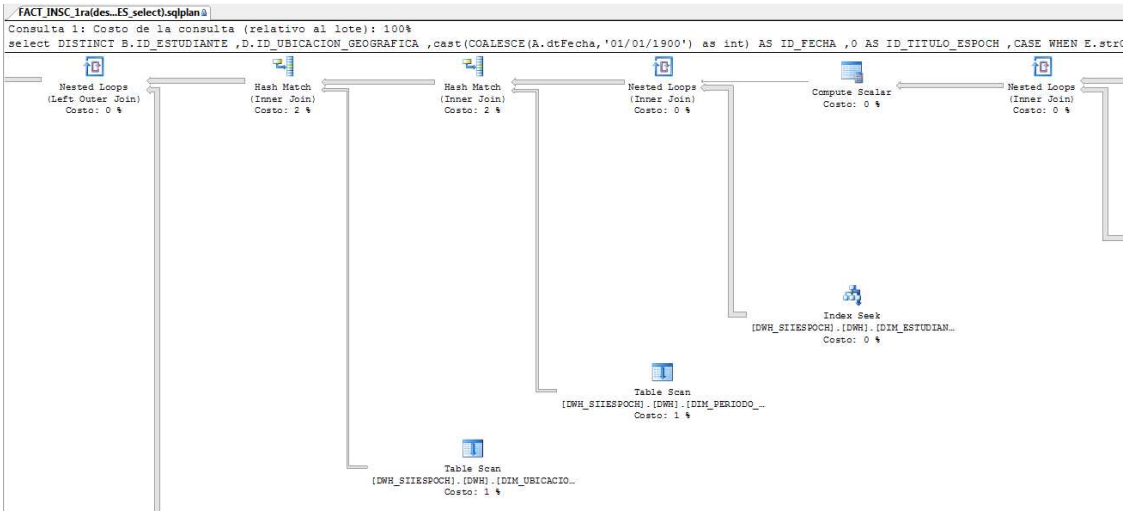


1.3. PLANES DE EJECUCIÓN DEL PROYECTO SII-ESPOCH

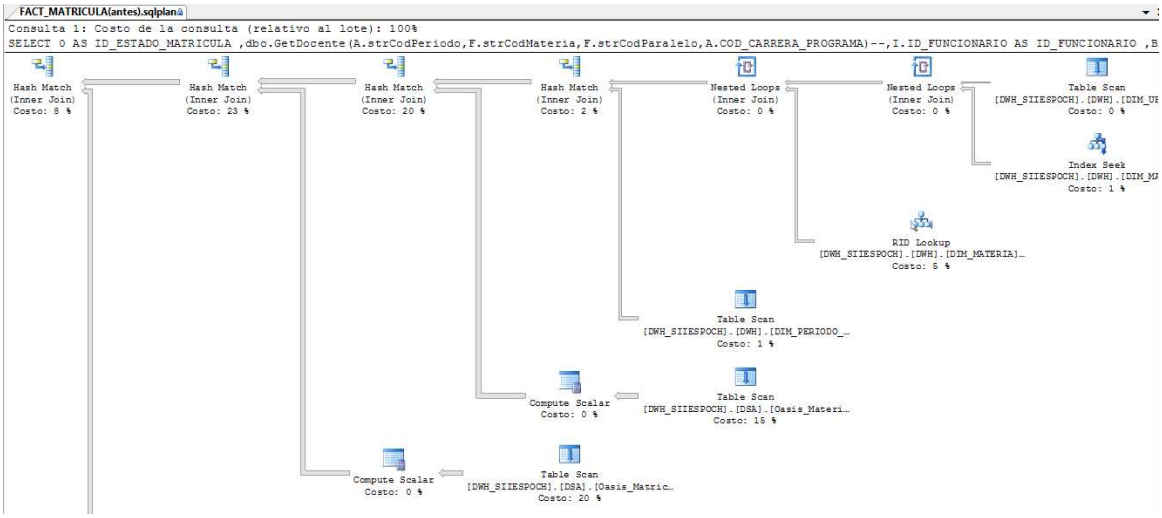
Plan de ejecución de la carga de la FACT DWH.FAC_INSCRIPCION_GRADO (SIN APLICAR TÉCNICAS TUNING)



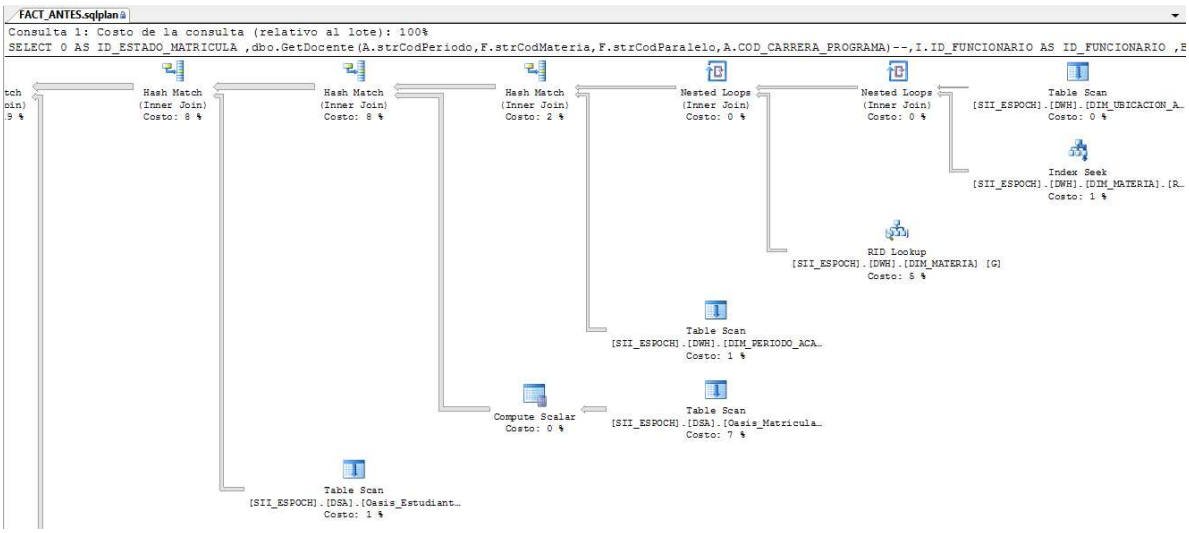
Plan de ejecución de la carga de la FACT DWH.FAC_INSCRIPCION_GRADO (APLICANDO TÉCNICAS TUNING)



Plan de ejecución de la carga de la FACT DWH.FAC_MATRICULAS_EVALUACION (SIN APLICAR TÉCNICAS TUNING)



Plan de ejecución de la carga de la FACT DWH.FAC_MATRICULAS_EVALUACION (APLICANDO TÉCNICAS TUNING)



ANEXO 2

2. SCRIPTS DE MANTENIMIENTO

Es necesario ejecutar trabajos de mantenimiento cuyos objetivos sean el de mantener la performance de las bases de datos y evitar su degradación.

Por lo tanto se debe agendarlos en horarios, en donde las bases de datos se encuentren libres de cualquier actividad de producción.

Es aconsejable actualizar las estadísticas semanalmente, es decir ejecutar los scripts que se indican a continuación.

2.1. SCRIPT DE MANTENIMIENTO DE ÍNDICES

El script de mantenimiento de índices es el siguiente:

```
-------------------------REORGANIZAR INDICES


DBCC DBREINDEX('DSA.Oasis_Estudiantes',INDEX_ESTUDIANTES,0)

DBCC DBREINDEX('DWH.DIM_ESTUDIANTE',INDEX_ESTUDIANTES,0)

DBCC DBREINDEX('DSA.Oasis_Grados',INDEX_GRADOS,0)

DBCC DBREINDEX(' DSA.Oasis_Matriculas', INDEX_MATRICULAS,0)

DBCC DBREINDEX(' DSA.Oasis_Materias_Asignadas', INDEX_MATERIAS_ASIGNADAS,0)

DBCC DBREINDEX(' DWH.DIM_MATERIA', INDEX_MATERIA,0)

DBCC
DBREINDEX('DWH.FAC_INSCRIPCION_GRADO',PK_FAC_INSCRIPCION_GRADO,0)

DBCC
DBREINDEX('DWH.FAC_MATRICULAS_EVALUACION',DWH.FAC_MATRICULAS_EVALU
ACION,0)
```

2.2. RESULTADO DE EJECUCIÓN DEL COMANDO SP_UPDATESTATS

Al ejecutar el comando sp_updatestats, se obtiene el siguiente mensaje de actualización:

```
Actualizando [dbo].[sysdiagrams]
    [PK__sysdiagrams__04E4BC85], no es necesaria su actualización...
    [UK_principal_name], no es necesaria su actualización...
    [_WA_Sys_00000001_03F0984C], no es necesaria su actualización...
    Se actualizaron 0 índices o estadísticas; no fue necesario actualizar 3.


Actualizando [DWH].[DIM_FECHA]
    [PK_DIM_FECHA], no es necesaria su actualización...
    Se actualizaron 0 índices o estadísticas; no fue necesario actualizar 1.
…….
Actualizando [DWH].[DIM_UBICACION_ACADEMICA]
    Se actualizó [PK_DIM_UBICACION_ACADEMICA]...
    Se actualizó [RELATIONSHIP_11_FK]...
    Se actualizó [RELATIONSHIP_12_FK]...
    Se actualizó [_WA_Sys_00000010_778AC167]...
    Se actualizó [_WA_Sys_00000006_778AC167]...
    Se actualizó [_WA_Sys_00000005_778AC167]...
    Se actualizó [_WA_Sys_00000004_778AC167]...
    Se actualizaron 16 índices o estadísticas; no fue necesario actualizar 0.


…….
Se han actualizado las estadísticas de todas las tablas.
```

2.3. SCRIPT DE MANTENIMIENTO DE LAS ESTADÍSTICAS DE SQL SERVER

El script de mantenimiento de actualización de estadísticas es el siguiente:

```
-------------------------ACTUALIZAR ESTADISTICAS


UPDATE STATISTICS DSA.Oasis_Docentes WITH FULLSCAN
UPDATE STATISTICS DSA.Oasis_Estudiantes WITH FULLSCAN
UPDATE STATISTICS DSA.Oasis_Grados WITH FULLSCAN
UPDATE STATISTICS DSA.Oasis_Inscripciones WITH FULLSCAN
UPDATE STATISTICS DSA.Oasis_Matriculas WITH FULLSCAN
UPDATE STATISTICS DSA.Oasis_Materias_Asignadas
WITH FULLSCAN
UPDATE STATISTICS DWH.DIM_CARGO_CATEGORIA WITH FULLSCAN
UPDATE STATISTICS DWH.DIM_DEDICACION WITH FULLSCAN
UPDATE STATISTICS DWH.DIM_DISCAPACIDAD WITH FULLSCAN
UPDATE STATISTICS DWH.DIM_ESTADO_ESTUDIANTE WITH FULLSCAN
UPDATE STATISTICS DWH.DIM_ESTADO_MATRICULA WITH FULLSCAN
UPDATE STATISTICS DWH.DIM_ESTUDIANTE WITH FULLSCAN
UPDATE STATISTICS DWH.DIM_FECHA WITH FULLSCAN
UPDATE STATISTICS DWH.DIM_FUNCIONARIO WITH FULLSCAN
UPDATE STATISTICS DWH.DIM_INSTITUCION WITH FULLSCAN
UPDATE STATISTICS DWH.DIM_MATERIA WITH FULLSCAN
UPDATE STATISTICS DWH.DIM_PERIODO_ACADEMICO WITH FULLSCAN
UPDATE STATISTICS DWH.DIM_SEDE WITH FULLSCAN
UPDATE STATISTICS DWH.DIM_TIPO_FUNCIONARIO WITH FULLSCAN
UPDATE STATISTICS DWH.DIM_TIPO_MODALIDAD WITH FULLSCAN
UPDATE STATISTICS DWH.DIM_TIPO_PROGRAMA WITH FULLSCAN
UPDATE STATISTICS DWH.DIM_TIPO_UBICACION WITH FULLSCAN
UPDATE STATISTICS DWH.DIM_TITULO_ESPOCH WITH FULLSCAN
UPDATE STATISTICS DWH.DIM_TITULO_INSTITUCION WITH FULLSCAN
UPDATE STATISTICS DWH.DIM_UBICACION_ACADEMICA WITH FULLSCAN
UPDATE STATISTICS DWH.DIM_UBICACION_GEOGRAFICA WITH FULLSCAN
UPDATE STATISTICS DWH.FAC_INSCRIPCION_GRADO WITH FULLSCAN
UPDATE STATISTICS DWH.FAC_MATRICULAS_EVALUACION WITH FULLSCAN
```