

# PROYEK AKHIR

# PROYEK AKHIR

Optimasi Desain Database Mata Garuda Dengan Menerapkan Partition Table

**Syandu Irdoni**

**NRP. 2110155024**

**Dosen Pembimbing:**

**Ferry Astika Saputra, ST, M.Sc.**

**NIP. 197708232001121002**

**Ahmad Syauqi Ahsan, S.Kom., M.T.**

**NIP. 197505302003121001**

**PROGRAM STUDI D4 TEKNIK INFORMATIKA**

**DEPARTEMEN TEKNIK INFORMATIKA DAN KOMPUTER**

**POLITEKNIK ELEKTRONIKA NEGERI SURABAYA**

**2017**

# PROYEK AKHIR

# PROYEK AKHIR



Optimasi Desain Database Mata Garuda Dengan Menerapkan Partition Table

**Syandu Irdoni**

**NRP. 2110155024**

**Dosen Pembimbing:**

**Ferry Astika Saputra, ST, M.Sc.**

**NIP. 197708232001121002**

**Ahmad Syauqi Ahsan, S.Kom., M.T.**

**NIP. 197505302003121001**

**PROGRAM STUDI D4 TEKNIK INFORMATIKA**

**DEPARTEMEN TEKNIK INFORMATIKA DAN KOMPUTER**

**POLITEKNIK ELEKTRONIKA NEGERI SURABAYA**

**2017**



Optimasi Desain Database Mata Garuda Dengan Menerapkan Partition Table

Nama Mahasiswa : Syandu Irdoni

NRP : 2110155024

Jurusan : Teknik Informatika PENS

ABSTRAK

IDS adalah alat yang digunakan untuk mendeteksi aksi-aksi kejahatan pada dunia maya. Mata Garuda sebagai interface manajemen event dan informasi keamanan jaringan ( SIEM ) mengalami kesulitan dalam mengelola data intrusi dalam jumlah yang besar. Pada versi terakhir database Mata Garuda yang menggunakan PostgreSQL 9.1 hanya mampu menangani data dalam jumlah 4 - 5 juta event pada beberapa query yang digunakan, selebih itu query akan berjalan sangat lambat. Pada sistem Mata Garuda, data disimpan menjadi beberapa tabel yang ukurannya berkembang sangat pesat, sedangkan padan beberapa referensi, join pada tabel yang berukuran besar sangat tidak direkomendasikan. Selain itu topologi masih menggunakan satu server database OLTP yang mendapat input dari kurang lebih 20 sensor, analisa trend dilakukan juga dilakukan pada database OLTP, sehingga beban ditangan oleh satu server. Oleh sebab itu pada proyek akhir ini akan dilakukan optimasi terhadap desain pengolahan data dari aplikasi Mata Garuda dengan menerapkan Partition Table, memisahkan database transaksi dan analisa, serta melakukan agregasi pada raw data sebelum dilakukan analisa menjadi report. Dari hasil percobaan Partition Table belum berhasil memberikan dampak yang signifikan dari permasalahan yang diangkat, pemisahan server database dengan menambahkan database tersendiri dari setiap sensor memberikan dampak dengan menurunnya beban server, karena beban dibagi di setiap sensor. Penerapan aggregasi pada data sebelum di query menjadi report memberikan efek positif, dengan menyusutnya jumlah row, penghitungan trend menjadi jauh lebih cepat. Secara keseluruhan desain sistem yang diajukan pada proyek akhir ini berhasil memberikan performa yang lebih baik dalam query report yang di butuhkan oleh aplikasi mata garuda.

**Kata kunci:**  IDS, Mata Garuda, Database, Partition Table, OLAP

Optimasi Desain Database Mata Garuda Dengan Menerapkan Partition Table

Nama Mahasiswa : Syandu Irdoni

NRP : 2110155024

Jurusan : Teknik Informatika PENS

**ABSTRACT**

The IDS is a tool used to detect cyber crime on network. Mata Garuda as security information and event management (SIEM) have difficulty managing large data intrusion. On latest database design in Mata Garuda using postgresql 9.1 is only able to handle the data up to 4-5 million event, greater than that some query will run very slow. At systems Mata Garuda, data saved in some tables and growing very rapidly, some reference says that join on big size tables highly not recommended. In addition topology using one database server OLTP that received input from more than 20 sensors, trend analysis is done in OLTP Database, which places a heavy load on the server. Therefore this Final Project will be carried Optimization design Against Data Processing Of Applications Mata Garuda with implementation of Partition Table, separating the Transaction Database and analysis, as well as perform aggregation of raw data before data ready for report. Experimental findings of the Partition Table not yet provide significant impact, the separation of the tansaction database server and analysis database provide impact declining workload Server, because workload now shared on sensor database. Implementation of data aggregation on the query report gives positive effects, with shrinking Period line, the trend calculation gain more performance. The overall design of the system succesfully provides better performance for report needed by applications Mata Garuda.

**Keywords** – IDS, Mata Garuda, Database, Partition Table, OLAP

KATA PENGANTAR

Segala puja dan puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa yang selalu memberi kesehatan jasmani dan rohani, juga kepandaian sehingga penulis bisa menyelesaikan laporan Proyek Akhir yang berjudul: **Optimasi Desain Database Mata Garuda Dengan Menerapkan Partition Table**.

Laporan Proyek Akhir ini disusun sebagai salah satu syarat lulus akademis dengan memperoleh gelar Sarjana Teknik Terapan (S.St.) di Politeknik Elektronika Negeri Surabaya.

Penulis mendapatkan banyak bantuan dari berbagai pihak dalam menyelesaikan laporan Proyek Akhir ini. Oleh karena itu, penulis ucapkan terimakasih sebanyak-banyaknya.

Meskipun Laporan Proyek Akhir ini berhasil disusun, penulis yakin bahwa masih banyak kekurangan, sehingga penulis mengharapkan saran yang membangun agar bisa memperbaiki kekurangan-kekurangan tersebut.

Semoga dengan ijin Tuhan Yang Maha Esa Laporan Proyek Akhir ini dapat memberikan manfaat bagi mahasiswa pada khususnya dan masyarakat luas pada umumnya.

Surabaya, Maret 2017

Syandu Irdoni

UCAPAN TERIMA KASIH

Segala puja dan puji syukur senantiasa dipanjatkan kepada ALLAH SWT yang telah memberikan nikmat, hidayah dan ‘inayah-Nya sehingga penulis bisa menyelesaikan Proyek Akhir ini. Kemudian tidak lupa penulis ucapkan terima kasih yang sebesar-besarnya kepada berbagai pihak yang turut membantu memperlancar penyusunan dan penyelesaian tugas akhir ini, diantaranya adalah:

1. Kedua Orang tua tercinta dan saudara-saudara yang telah memberikan do’a, semangat dan kasih sayang serta nasihat.
2. Bapak Ferry Astika Saputra dan Bapak Ahmad Syauqi Ahsan selaku dosen pembimbing yang membimbing penulis hingga proyek akhir ini selesai.
3. Training JICA dan PENS bersama Timor Leste, Brunei dan Laos.
4. Semua dosen penguji sidang TPPA dan sidang PA.
5. Semua dosen Teknik Informatika yang telah memberikan bimbingan dari mulai nol hingga sekarang.
6. Teman-teman B3IBS D3 IT B 2012.
7. Teman-teman D4 LJ IT 2015.
8. Semua penulis yang referensinya telah melengkapi, menambahkan, serta memperkaya proyek akhir ini.
9. Dan semua teman-teman atau kerabat yang belum disebutkan diatas.

DAFTAR ISI

[ABSTRAK iii](#_Toc474742086)

[KATA PENGANTAR v](#_Toc474742087)

[UCAPAN TERIMA KASIH vi](#_Toc474742088)

[DAFTAR ISI vii](#_Toc474742089)

[DAFTAR GAMBAR x](#_Toc474742090)

[DAFTAR TABEL xii](#_Toc474742091)

[BAB I 1](#_Toc474742092)

[PENDAHULUAN 1](#_Toc474742093)

[1.1 LATAR BELAKANG 1](#_Toc474742094)

[1.2 IDENTIFIKASI PERMASALAHAN 5](#_Toc474742095)

[1.3 TUJUAN 6](#_Toc474742096)

[1.4 KONTRIBUSI 6](#_Toc474742097)

[1.5 METODOLOGI 6](#_Toc474742098)

[1.5.1 Studi Literatur 6](#_Toc474742099)

[1.5.2 Perancangan Sistem Perangkat Lunak 7](#_Toc474742100)

[1.5.3 Implementasi Sistem 7](#_Toc474742101)

[1.5.4 Pengujian dan Evaluasi 7](#_Toc474742102)

[1.5.5 Pembuatan Laporan 7](#_Toc474742103)

[1.6 SISTEMATIKA PENULISAN 8](#_Toc474742104)

[BAB II 9](#_Toc474742105)

[TEORI PENUNJANG 9](#_Toc474742106)

[2.1 DASAR PERMASALAHAN 9](#_Toc474742107)

[2.2 PENELITIAN SEBELUMNYA 9](#_Toc474742108)

[2.3 DESAIN DATABASE MATA GARUDA 15](#_Toc474742109)

[2.4 DESAIN DATABASE APLIKASI SEJENIS 22](#_Toc474742110)

[2.5 KEUNIKAN DESAIN 23](#_Toc474742111)

[2.6 QUERY REPORT MATA GARUDA 23](#_Toc474742112)

[2.7 DASAR TEORI 25](#_Toc474742113)

[2.7.1 Partition Table 25](#_Toc474742114)

[2.7.2 Partitioning Pruning 28](#_Toc474742115)

[2.7.3 SQL ( Structured Query Language ) 29](#_Toc474742116)

[2.7.4 Join 30](#_Toc474742117)

[2.7.5 Indexing 30](#_Toc474742118)

[2.7.6 Query Optimizer 30](#_Toc474742119)

[2.7.7 Normalisasi 31](#_Toc474742120)

[2.7.8 Denormalisasi 34](#_Toc474742121)

[2.7.9 OLTP vs OLAP [9] 34](#_Toc474742122)

[2.7.10 Data Warehouse [10] 35](#_Toc474742123)

[2.7.11 ETL [9] 36](#_Toc474742124)

[2.7.12 Instalasi MySQL 5.7 38](#_Toc474742125)

[BAB III 41](#_Toc474742126)

[PERANCANGAN SISTEM 41](#_Toc474742127)

[3.1 DESKRIPSI UMUM SISTEM 41](#_Toc474742128)

[3.2 MIGRASI DATABASE ENGINE 42](#_Toc474742129)

[3.3 PERANCANGAN SISTEM 43](#_Toc474742130)

[3.3.1. Entity Relationship Diagram 43](#_Toc474742131)

[3.3.2. Flow ETL ( Extract, Transform, Load ) 57](#_Toc474742132)

[3.3.3. Agregasi Event 62](#_Toc474742133)

[BAB IV 67](#_Toc474742134)

[UJICOBA DAN ANALISA 67](#_Toc474742135)

[4.1 Uji Coba Sistem OLTP OLAP 67](#_Toc474742136)

[4.1.1. Instalasi warehouse 67](#_Toc474742137)

[4.1.2. Instalasi Sensor 69](#_Toc474742138)

[4.2 Uji Coba ETL 70](#_Toc474742139)

[4.3 Uji Coba Performa Tabel Partition vs Non-Partition 71](#_Toc474742140)

[4.4 Index Partition vs Non-Partition 72](#_Toc474742141)

[4.5 Uji Coba Performa Report 73](#_Toc474742142)

[4.4.1. Environtment Pengujian 73](#_Toc474742143)

[4.1.3. Data Pengujian 74](#_Toc474742144)

[4.1.4. Pengujian 1 ( 5 juta baris ) 74](#_Toc474742145)

[BAB V 80](#_Toc474742146)

[PENUTUP 80](#_Toc474742147)

[5.1 OPTIMASI DESAIN DATABASE 80](#_Toc474742148)

[5.2 KESIMPULAN 81](#_Toc474742149)

[5.3 SARAN 81](#_Toc474742150)

[DAFTAR PUSTAKA 82](#_Toc474742151)

DAFTAR GAMBAR

[Gambar 1.1. Topologi Mata Garuda 1 2](#_Toc474742152)

[Gambar 1.2. Topologi Mata Garuda 2 2](#_Toc474742153)

[Gambar 1.3. Schema default snort 4](#_Toc474742154)

[Gambar 1.4. Statistik Performa Desain ACID pada PostgreSQL 7.1.2 4](#_Toc474742155)

[Gambar 1.5. Metodologi Proyek Akhir 6](#_Toc474742156)

[Gambar 2.1. Hasil Perbandingan Row-store vs Column-store 10](#_Toc474742157)

[Gambar 2.2. Grafik Perbandingan Row-store vs Column-store 11](#_Toc474742158)

[Gambar 2.3. Query yang diujikan 12](#_Toc474742159)

[Gambar 2.4. Jumlah data yang diuji 12](#_Toc474742160)

[Gambar 2.5. Hasil query dengan 12 partisi 13](#_Toc474742161)

[Gambar 2.6. Hasil query 6 partisi 13](#_Toc474742162)

[Gambar 2.7. Hasil query 3 partisi 14](#_Toc474742163)

[Gambar 2.8. Hasil query tanpa partisi 14](#_Toc474742164)

[Gambar 2.9. Skema default snort 15](#_Toc474742165)

[Gambar 2.10. Vertical dan Horizontal Partitioning 26](#_Toc474742166)

[Gambar 2.11. Metode Partitioning Tabel 26](#_Toc474742167)

[Gambar 2.12. Contoh range partition 28](#_Toc474742168)

[Gambar 2.13. Case pengujian partisi 28](#_Toc474742169)

[Gambar 2.14. UNF ( UNNORMALIZED FORM ) 31](#_Toc474742170)

[Gambar 2.15. 1NF NORMAL FORM 1 32](#_Toc474742171)

[Gambar 2.16. 2NF NORMAL FORM 2 32](#_Toc474742172)

[Gambar 2.17. 3NF NORMAL FORM 3 33](#_Toc474742173)

[Gambar 2.18. BCNF BOYCE-CODD 33](#_Toc474742174)

[Gambar 2.19. OLTP vs OLAP 34](#_Toc474742175)

[Gambar 2.20. Perbandingan OLTP vs OLAP 35](#_Toc474742176)

[Gambar 2.21. Gambaran Data Warehouse 36](#_Toc474742177)

[Gambar 2.22. Download repo mysql 38](#_Toc474742178)

[Gambar 2.23. Pilih versi mysql 38](#_Toc474742179)

[Gambar 2.24. Update list repo 39](#_Toc474742180)

[Gambar 2.25. Instalasi Mysql 5.7 39](#_Toc474742181)

[Gambar 3.1. Deskripsi Umum Desain Database 41](#_Toc474742182)

[Gambar 3.2. Tabel-tabel hasil partitioning di PostgreSQL 43](#_Toc474742183)

[Gambar 3.3. Snort Default Schema 44](#_Toc474742184)

[Gambar 3.4. ETL ( Extract, Transform, Load ) 57](#_Toc474742185)

[Gambar 3.5. Flow ETL 58](#_Toc474742186)

[Gambar 3.6. Flowchart create\_scheaduler.sh 59](#_Toc474742187)

[Gambar 3.7. Flowchart push\_maindb.sh 60](#_Toc474742188)

[Gambar 3.8. Flowchart extract.sh 61](#_Toc474742189)

[Gambar 3.9. Flowchart extract\_process.sh 62](#_Toc474742190)

[Gambar 3.10. Flowchart daily\_aggregation 63](#_Toc474742191)

[Gambar 3.11. Flowchart monthly\_aggregation.sh 64](#_Toc474742192)

[Gambar 3.12. Flowchart annual aggregation 65](#_Toc474742193)

[Gambar 4.1. Stuktur Script Warehouse 68](#_Toc474742194)

[Gambar 4.2. Listener MySQL 68](#_Toc474742195)

[Gambar 4.3. Menambahakan Informasi sensor 68](#_Toc474742196)

[Gambar 4.4. SSH no password 68](#_Toc474742197)

[Gambar 4.5. File script warehouse 69](#_Toc474742198)

[Gambar 4.6. Crobjob warehouse 69](#_Toc474742199)

[Gambar 4.7. Struktur Script Sensor 69](#_Toc474742200)

[Gambar 4.8. Barnyard2.conf 70](#_Toc474742201)

[Gambar 4.9. Instalasi script pada sensor 70](#_Toc474742202)

[Gambar 4.10. Log ETL 70](#_Toc474742203)

[Gambar 4.11. Grafik Perbangingan Query Tabel Partisi vs Non-partisi 71](#_Toc474742204)

[Gambar 4.12. Grafik Perbandingan pembaccan index 72](#_Toc474742205)

[Gambar 4.13. Index Partition vs Non-partition 73](#_Toc474742206)

[Gambar 4.14. Grafik Perbandingan Query Report Harian 75](#_Toc474742207)

[Gambar 4.15. Grafik Perbandingan Query Report Bulanan 77](#_Toc474742208)

[Gambar 4.16. Grafik Perbandingan Query Report Tahunan 78](#_Toc474742209)

DAFTAR TABEL

[Tabel 2.1. Tabel Event 15](#_Toc474742210)

[Tabel 2.2. Tabel Sensor 16](#_Toc474742211)

[Tabel 2.3. Tabel Iphdr 17](#_Toc474742212)

[Tabel 2.4. Tabel ICMP Header 17](#_Toc474742213)

[Tabel 2.5. Tabel Tcphdr 18](#_Toc474742214)

[Tabel 2.6. Tabel Udphdr 18](#_Toc474742215)

[Tabel 2.7. Tabel Data 19](#_Toc474742216)

[Tabel 2.8. Tabel Signature 19](#_Toc474742217)

[Tabel 2.9. Tabel Sig\_class 20](#_Toc474742218)

[Tabel 2.10. Tabel sig\_reference 20](#_Toc474742219)

[Tabel 2.11. Tabel Reference\_system 21](#_Toc474742220)

[Tabel 2.12. Tabel ip\_location 21](#_Toc474742221)

[Tabel 2.13. List Tabel Snorby 22](#_Toc474742222)

[Tabel 2.14. List Query Report Mata Garuda 24](#_Toc474742223)

[Tabel 3.1. Tabel event 44](#_Toc474742224)

[Tabel 3.2. Tabel daily\_event\_aggregation 46](#_Toc474742225)

[Tabel 3.3. Tabel monthly\_event\_aggregation 48](#_Toc474742226)

[Tabel 3.4. Tabel annual\_event\_aggregation 49](#_Toc474742227)

[Tabel 3.5. Tabel sensor 50](#_Toc474742228)

[Tabel 3.6. Tabel sensor\_push\_log 51](#_Toc474742229)

[Tabel 3.7. Tabel ip\_location 52](#_Toc474742230)

[Tabel 3.8. Tabel ip\_map 53](#_Toc474742231)

[Tabel 3.9. Tabel services 55](#_Toc474742232)

[Tabel 3.10. Tabel protocols 55](#_Toc474742233)

[Tabel 3.11. Tabel signature\_reference 56](#_Toc474742234)

[Tabel 4.1. Pengujian query tabel partition vs non-partition 71](#_Toc474742235)

[Tabel 4.2. Tabel perbandingan index Partition vs Non-Partition 72](#_Toc474742236)

[Tabel 4.3. Spesifikasi Hardware Pengujian 73](#_Toc474742237)

[Tabel 4.4. Database Pengujian 74](#_Toc474742238)

[Tabel 4.5. Tabel perbandingan performa query harian 74](#_Toc474742239)

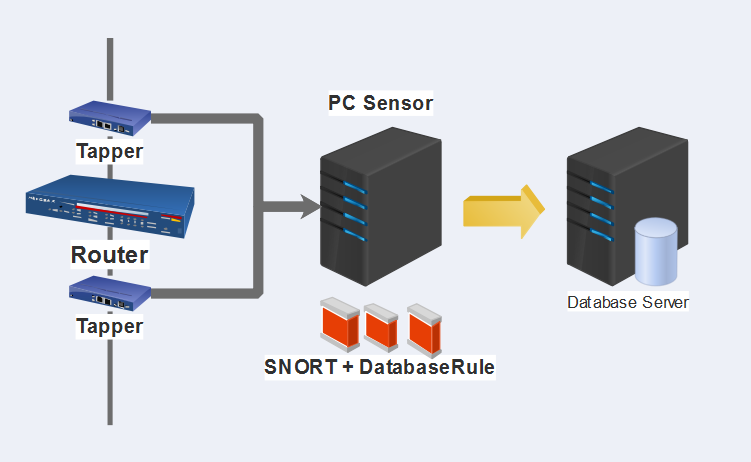
BAB I

PENDAHULUAN

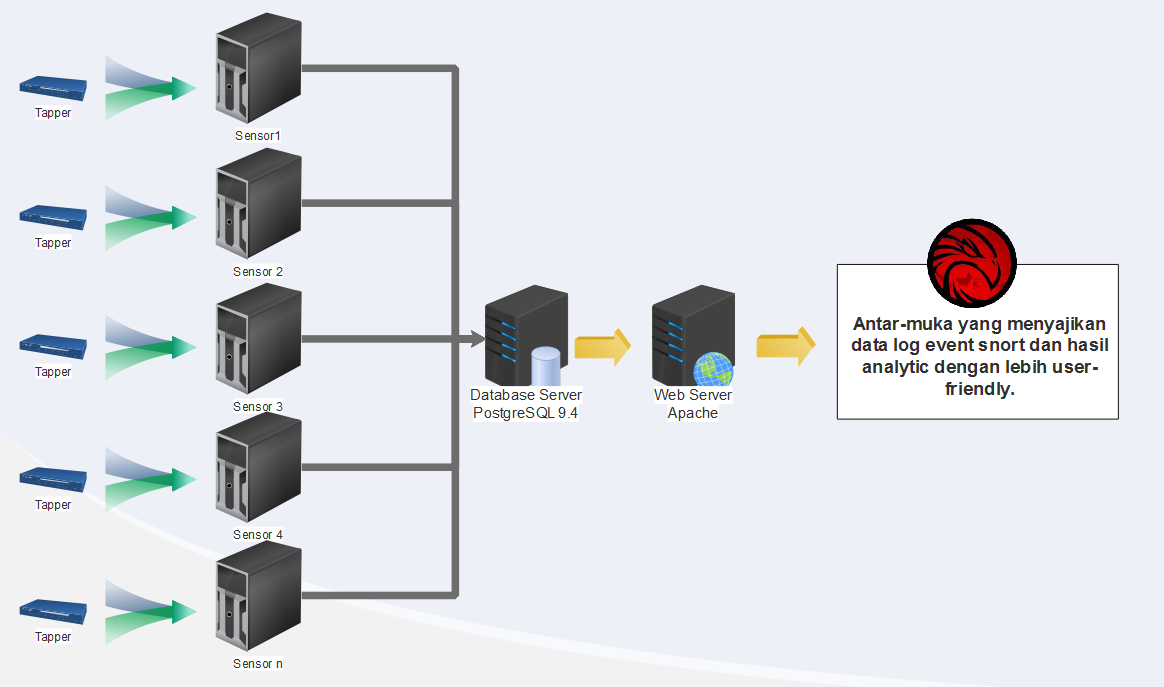
1. LATAR BE**L**AKANG

Internet merupakan salah satu media komunikasi yang paling banyak digunakan oleh hampir semua teknologi terutama perangkat komputer, jaringan internet menghubungkan banyak perangkat komputer untuk saling berkomunikasi satu dengan lainnya sehingga lalu lintas data yang ada pada jaringan internet sangat padat. Komunikasi data yang sangat banyak tersebut perlu dilakukan pemantauan atau monitoring untuk menghindari tindak kejahatan yang mungkin terjadi pada penggunaan teknologi internet. Salah satu tool atau sistem yang paling sering digunakan untuk melakukan monitoring keamanan jaringan internet adalah Intrution Detection System (IDS). IDS bertugas melakukan monitoring jaringan internet berdasarkan rule atau aturan-aturan yang telah ditetapkan administrator pada saat konfigurasi, begitu juga dengan alamat IP spesifik dari jaringan internet yang dipantau, hasil monitoring berupa log-log informasi mengenai paket data yang berhasil dipantau, log tersebut dapat disimpan pada folder sistem atau dapat juga disimpan pada database. Hasil log IDS tersebut yang menjadi bahan kajian bagi para administrator jaringan untuk melakukan analisa lebih lanjut. [1]

Pada tahun 2014 IDSIRTI bekerjasama dengan dosen dan mahasiswa dari kampus PENS ( Politeknik Elektronika Negeri Surabaya ) bersama membangun sebuah sistem antar muka yang memanage informasi dan event serangan keamanan jaringan dengan nama ”Mata Garuda”. Berikut adalah topologi sistem monitoring Mata Garuda yang diterapkan. [1]



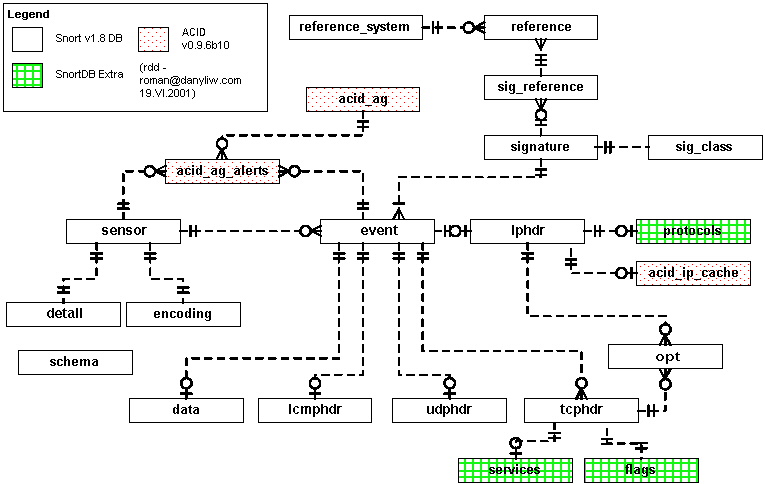
1. Topologi Mata Garuda 1



1. Topologi Mata Garuda 2

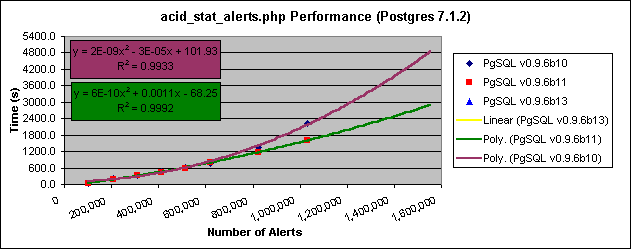
Alur monitoring dimulai dari tapper yang digunakan sebagai alat untuk memantau jaringan sehingga dapat dibaca packet-packet yang melewati jaringan yang sedang dimonitoring. Selanjutnya packet-packet tersebut akan diperiksa oleh PC Sensor yang menggunakan IDS SNORT dengan membandingkan struktur packet-packet yang lewat dengan rule-rule SNORT. Hasil monitoring jika ditemukan event akan dimasukkan kedalam database server terpusat yang menggunakan PostgreSQL Database. Setelah itu data siap diolah dan ditampilkan pada Network Administrator untuk dilakukan tindakan-tindakan berikutnya. [1]

Seiring dengan berjalannya waktu sistem monitoring Mata Garuda terus dikembangkan untuk dapat berjalan dengan optimal dalam kemampuan mengolah data dan menyajikannya. Saat ini Mata Garuda berjalan menggunakan framework CI ( CodeIgniter ) dengan database PostgreSQL 9.1. Pada versi Mata Garuda saat ini, terjadi permasalahan pada kemampuan database dalam mengolah dan menyajikan data. Kemampuan yang harus disiapkan untuk menjadi sistem manajemen dan informasi event keamanan jaringan yang handal, salah satunya adalah kecepatan sistem untuk mengolah dan menyajikan laporan monitoring, untuk mewujudkan hal itu ada banyak parameter yang harus dipenuhi, salah satunya adalah desain database yang sesuai. Pada desain pengembangan sistem Mata Garuda saat ini, database mengalami kesulitan dalam mengolah data ketika data yang tersimpan lebih dari 5 juta baris, beberapa query akan berjalan sangat lambat. Desain database pada sistem Mata Garuda saat ini menggunakan desain database ACID *( Analysis Console for Intrusion Databases )* bawaan dari snort.



1. **Schema default snort**

Data log event alert dari sensor snort disimpan pada beberapa tabel yang saling berelasi dengan tabel lainnya. Pada praktiknya, dengan server database yang memiliki spesifikasi 16 core dan 15 GB Memory dengan menggunakan engine database PostgreSQL desain tersebut hanya mampu mengolah data pada kisaran 4 juta sampai 5 juta baris data, itupun juga sangat lambat.



1. Statistik Performa Desain ACID pada PostgreSQL 7.1.2

**Pada referensi lain juga menunjukkan bahwa desain acid tersebut dengan menggunakan database engine PostgreSQL juga membutuhkan waktu yang kurang memuaskan untuk data yang hanya berkisar antara 200.000 baris hingga 1.800.000 baris.**

**Hipotesa awal kenapa hal ini bisa terjadi dikarenakan query yang digunakan untuk menyajikan data banyak menggunakan join, sedangkan jumlah data pada tiap tabel yang di-join-kan bertambah sangat pesat. Beberapa referensi mengatakan bahwa join terhadap tabel yang memiliki data yang besar sangat tidak direkomendasikan.**

***“Avoid joins to large tabels Joining of large data sets using nested loops is very expensive. Try to avoid it. Joins to smaller tabels is OK but you might want to preload them to memory before join . . . “*** [2]

**Pada proyek akhir ini, penulis mengusulkan penerapan Partition Table untuk memecah tabel yang besar menjadi partisi-partisi yang memiliki ukuran yang lebih kecil.** Secara definisi partisi tabel adalah memecah tabel menjadi beberapa bagian/segmen. Misalkan kita memilik data sebanyak 10 juta record, bila pada tabel konvensional maka data sebanyak itu akan di simpan dalam satu segmen, tetapi bila menggunakan partisi maka data sebanyak itu akan di pecah-pecah ke banyak partisi berdasarkan sebuah kondisi. **Sehingga join dapat dilakukan dengan kuantitas data pada setiap partisi lebih kecil dan diharapkan dapat memberikan solusi pada masalah yang telah diuraikan diatas.**

1. IDENTIFIKASI PERMASALAHAN

Berdasarkan dari uraian latar belakang di atas maka dapat dirumuskan suatu permasalahan yaitu aplikasi monitoring keamanan Mata Garuda membutuhkan solusi untuk dapat mengolah data dalam jumlah yang besar. Kemampuan mengolah data disini sangat berhubungan erat dengan database, dibutuhkan optimasi desain database baru untuk diterapkan pada system monitoring jaringan Mata Garuda.

Batasan masalah dalam tugas akhir ini adalah optimasi dilakukan pada system yang sudah ada, optimasi disini berkaitan dengan database, sehingga diharapkan performa system monitoring jaringan Mata Garuda dapat meningkat. Pengujian performa akan dilakukan dengan komputer lokal dengan mengeksekusi beberapa query yang layak yang digunakan oleh mata garuda.

1. TUJUAN

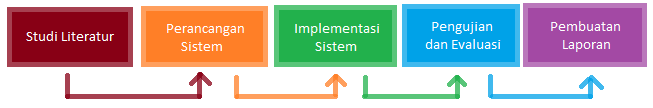
Tujuan dari proyek akhir ini adalah melakukan optimasi pengolahan data pada system monitoring jaringan Mata Garuda. Dengan menerapkan optimasi-optimasi yang mungkin dilakukan pada sistem yang sudah ada. Sehingga query, laporan serta analisis trend yang dilakukan system monitoring jaringan Mata Garuda dapat bekerja dengan optimal.

1. KONTRIBUSI

Hasil dari proyek akhir ini diharapkan dapat dapat mengoptimasi sistem mata garuda yang sudah ada. Dengan kecepatan dan cakupan data yang lebih banyak, analisa trend akan memiliki range yang lebih lebar, sehingga diharapkan juga respon terhadap intrusi akan lebih cepat.

1. METODOLOGI

Metodologi yang digunakan dalam proyek akhir ini meliputi beberapa tahapan seperti pada **Gambar 1.5**:



1. Metodologi Proyek Akhir
   * 1. Studi Literatur

Pada tahap ini yang dilakukan adalah studi mengenai system Mata Garuda yang sudah ada dan menganalisa hal yang mungkin menjadi akar permasalahan, mempelajari beberapa desain system pengolahan data pada aplikasi sejenis, bentuk-bentuk optimasi pengolahan data, optimasi menggunakan Partition Table, macam-macam Partition Table.

* + 1. Perancangan Sistem Perangkat Lunak

Perancangan sistem perangkat lunak yang dilakukan meliputi beberapa hal sebagai berikut :

1. Menganalisa perubahan apa saja yang diperlukan.

Disini akan dilakukan analisa terhadap system yang sudah ada, bagaimana melakukan partisi terhadap tabel-tabel yang sering di-join-kan.

1. Memutuskan desain partisi yang akan diterapkan

Ada beberapa bentuk partisi tabel, diantaranya adalah range partition, hash partition, dan list partition. Disini dilakukan analisa partisi dalam bentuk apa yang akan diterapkan.

1. Mempersiapkan desain optimasi partisi yang diajukan.

Mempersiapkan desain partisi serta komponen-komponen pendukungnya.

* + 1. Implementasi Sistem

Hasil perancangan optimasi sistem kemudian diimplementasikan untuk dicoba dan siap untuk di analisa performanya.

* + 1. Pengujian dan Evaluasi

Pengujian dan analisa dimaksudkan untuk mengetahui sejauh mana optimasi system yang diajukan pada proyek akhir ini dapat berfungsi lebih baik atau lebih buruk dari system yang sudah ada.

* + 1. Pembuatan Laporan

Membuat dokumentasi dari semua tahapan proses pengerjaan proyek akhir berupa laporan yang berisi tentang dasar teori, hasil proyek akhir, serta hasil analisa.

1. SISTEMATIKA PENULISAN

Sistematika penulisan dalam proyek akhir ini direncanakan sebagai berikut:

**BAB I PENDAHULUAN**

Bab ini berisi tentang pendahuluan yang terdiri dari latar belakang, perumusan masalah, batasan masalah, tujuan, manfaat, metodologi, serta sistematika penulisan dari Proyek Akhir ini.

**BAB II TEORI PENUNJANG**

Bab ini membahas mengenai teori – teori yang berkaitan dengan penyelesaian proyek akhir ini, yang didapatkan dari berbagai macam buku serta sumber-sumber terkait lainnya yang berhubungan dengan pembuatan Proyek Akhir ini.

**BAB III PERANCANGAN SISTEM**

Bab ini membahas mengenai perancangan system, meliputi perancangan diagram alur system, perancangan proses dan perancangan user interface.

**BAB IV UJI COBA DAN ANALISA**

Bab ini membahas seluruh hasil dan analisa dalam pembuatan proyek akhir ini dan bagaimana proses analisa tersebut hingga dapat ditampilkan kedalam perangkat mobile.

**BAB V PENUTUP**

Bab ini berisi kesimpulan dari hasil uji coba perangkat lunak dan saran untuk pengembangan, perbaikan serta penyempurnaan terhadap aplikasi yang telah dibuat.

**DAFTAR PUSTAKA**

Berisi tentang referensi-referensi yang telah digunakan sebagai landasan selama pembuatan Proyek Akhir ini.

BAB II

TEORI PENUNJANG

1. DASAR PERMASALAHAN

Berdasarkan dari uraian latar belakang di atas maka dapat dirumuskan suatu permasalahan yaitu aplikasi monitoring keamanan jaringan Mata Garuda membutuhkan solusi untuk dapat mengolah data dalam jumlah yang besar. Kemampuan mengolah data disini sangat berhubungan erat dengan database, dibutuhkan optimasi desain database baru untuk diterapkan pada system monitoring jaringan Mata Garuda.

Batasan masalah dalam tugas akhir ini adalah optimasi dilakukan pada sistem mata garuda yang sudah ada, optimasi disini berkaitan dengan database, sehingga diharapkan performa system monitoring keamaanan jaringan Mata Garuda dapat meningkat. Pengujian performa akan dilakukan dengan mengeksekusi beberapa query yang digunakan oeh aplikasi mata garuda.

1. PENELITIAN SEBELUMNYA

Berikut adalah beberapa penelitian sebelumnya yang memeiliki tema mirip dengan permasalahan pada proyek akhir ini.

Henk Ernst Blok, “DATABASE OPTIMIZATION ASPECTS FOR INFORMATION RETRIEVAL”. [3]

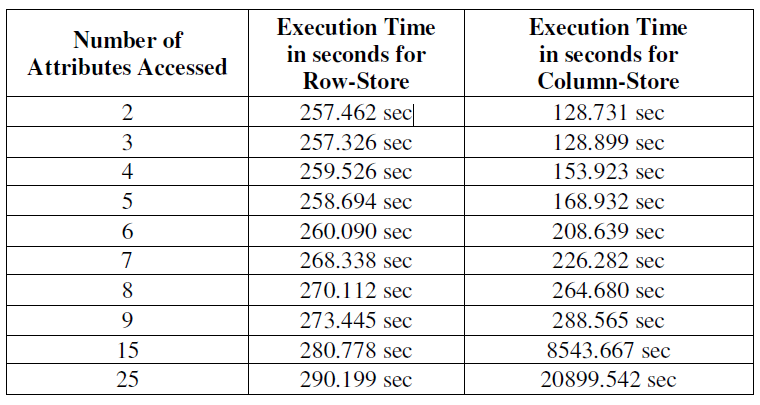
Penelitian ini berfokus pada pembahasan performa dari database yang dikombinasikan dengan Information Retrieval ( IR ). Beberapa topik yang menarik untuk digunakan referensi untuk mendukung proyek akhir ini antara lain adalah bagaimana performa retrieval data dapat di optimasi, bagaimanan database berinteraksi dengan dengan query yang diberikan user, serta komponen-komponen penting yang bertangung jawab pada performa query.

Haran Boral dan David J. DeWitt, “A METHODOLOGY FOR DATABASE SYSTEM PERFORMANCE EVALUATION”. [4]

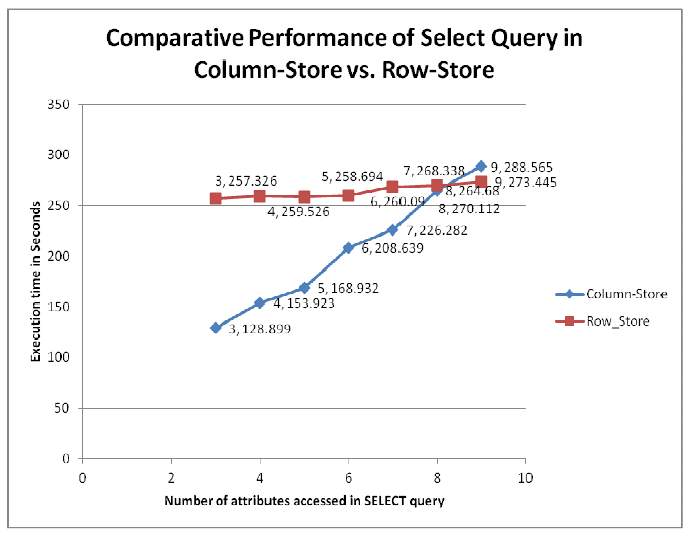
Penelitian ini membahas tentang evaluasi performa dari database management system dan database machine pada environtment multi user. Disini dijelaskan ada 3 faktor utama yang memiliki efek pada performa database dalam multiuser environtment. Tiga fak tersebut adalah multiprogramming level, query mix, dan degree of data sharing.

Aditi D. Andurkar, “IMPLEMENTATION OF COLUMN-ORIENTED DATABASE IN POSTGRESQL FOR OPTIMIZATION OF READ-ONLY QUERIES”. [5]

Pada penelitian ini disinggung mengenai permartisian tabel secara vertical yang telah diimplementasikan pada schema default dari snort. Pada penelitian ini juga dibuktikan bahwa model partisi vertical sangan baik diterapkan pada data yang berukuran sedikit, ketika data berkembang maka row-store terlihat lebih menjanjikan. Pada gambar 2.1 dan 2.2 adalah statistik dan grafik hasil percobaan simple select query antara row-store dan column-store.



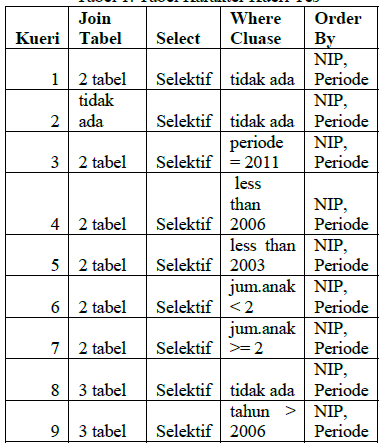
1. Hasil Perbandingan Row-store vs Column-store



1. Grafik Perbandingan Row-store vs Column-store

Y. D. Anto, J. Wibowo and T. Heru, “PENERAPAN BASIS DATA TERPARTISI DALAM RANCANG BANGUN SISTEM PENGGAJIANPENGUPAHAN DENGAN MODEL JARINGAN CLIENT-SERVER”. [6]

Dalam penelitian ini diterapkan pemartisian tabel secara horizontal. Disini dilakukan pengujian antara jumlah partisi yang diterpakan dengan kecepatan query. Berikut adalah hasil perbandingan yang dicuplik dari penelitian ini. Uji coba pemartisian dilakukan dengan 9 macam kueri yang memiliki karakteristik beda satu sama lainnya. Karakteristik query dapat dilihat pada gambar 2.3.



1. Query yang diujikan

Query-query yang diuji akan mengakses jumlah data yang berbeda, untuk jumlah data disetiap querinya dapat dilihat pada gambar 2.4 dibawah.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ku | Jum | Ku | Jum | Ku | Jum |
|  | Data |  | Data |  | Data |
| 1 | 477504 | 4 | 238752 | 7 | 320832 |
| 2 | 477504 | 5 | 119376 | 8 | 477504 |
| 3 | 39792 | 6 | 155672 | 9 | 253436 |

1. Jumlah data yang diuji

Pertama-tama yang data akan diuji dengan menggunakan tabel yang terpartisi menjadi 12, dengan mengeksekusi queri-queri yang sudah ditampilkan pada gambar 2.3.

|  |  |  |  |
| --- | --- | --- | --- |
| Kueri | Waktu Kueri  Partisi 12 ( Periode tahun ) | | |
| Uji 1 | Uji 2 | Uji 3 |
| 1 | 16.578 | 16.574 | 16.141 |
| 2 | 13.751 | 13.813 | 13.547 |
| 3 | 0.594 | 0.513 | 0.594 |
| 4 | 7.895 | 7.781 | 8.812 |
| 5 | 2.016 | 2.156 | 2.344 |
| 6 | 14.501 | 13.906 | 14.675 |
| 7 | 14.872 | 15.516 | 14.973 |
| 8 | 15.078 | 15.281 | 14.996 |
| 9 | 4.687 | 4.611 | 4.641 |

1. Hasil query dengan 12 partisi

Hasil yang didapatkan dari queri yang dilakukan pada tabel 12 partisi dibandingkan tabel yang memiliki 6 partisi terlihat bawha total waktu eksekusi banyak dimenangkan oleh tabel yang memiliki 6 partisi.

|  |  |  |  |
| --- | --- | --- | --- |
| Kueri | Waktu Kueri  Partisi 6 ( Periode tahun ) | | |
| Uji 1 | Uji 2 | Uji 3 |
| 1 | 17.031 | 17.563 | 16.819 |
| 2 | 14.578 | 14.312 | 14.473 |
| 3 | 0.547 | 0.547 | 0.531 |
| 4 | 7.751 | 8.781 | 8.782 |
| 5 | 2.157 | 2.251 | 2.21 |
| 6 | 14.334 | 14.453 | 14.657 |
| 7 | 14.719 | 15.344 | 15.172 |
| 8 | 16.607 | 16.047 | 16.703 |
| 9 | 6.594 | 6.609 | 6.187 |

1. Hasil query 6 partisi

Sedangkan hasil queri yang dilakukan pada tabel yang memiliki 3 partisi terlihat memiliki waktu eksekusi yang lebih lama lagi.

|  |  |  |  |
| --- | --- | --- | --- |
| Kueri | Waktu Kueri  Partisi 3 ( Periode tahun ) | | |
| Uji 1 | Uji 2 | Uji 3 |
| 1 | 17.091 | 16.781 | 17 |
| 2 | 14.5 | 14.344 | 14.156 |
| 3 | 0.51 | 0.547 | 0.55 |
| 4 | 8.789 | 8.719 | 8.781 |
| 5 | 2.359 | 2.641 | 2.625 |
| 6 | 2.5 | 2.406 | 2.24 |
| 7 | 15.078 | 15.28 | 14.891 |
| 8 | 17.547 | 17.487 | 17.642 |
| 9 | 8.021 | 6.987 | 6.907 |

1. Hasil query 3 partisi

Selain dicoba pada tabel berpartisi, pada penelitian ini juga dilakukan perbandingan terhadap tabel yang tidak dipartisi. Terlihat pada laporan pada gambar 2.8 tabel yang tidak memiliki partisi.

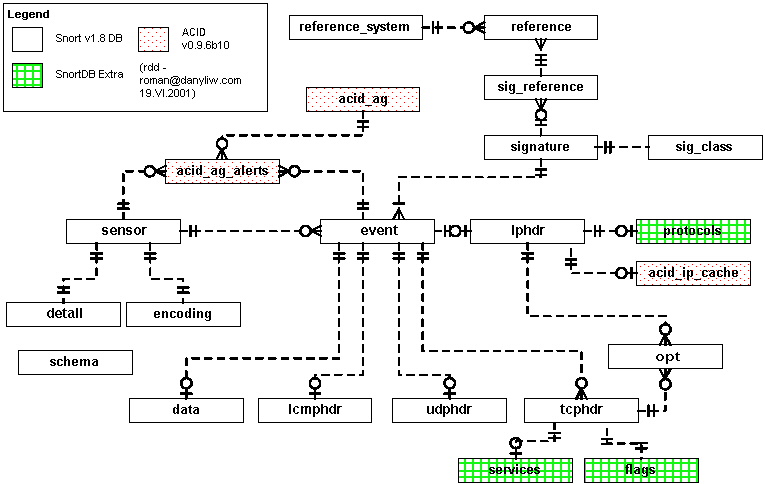
|  |  |  |  |
| --- | --- | --- | --- |
| Kueri | Waktu Kueri  Tanpa Partisi | | |
| Uji 1 | Uji 2 | Uji 3 |
| 1 | 51 | 60.422 | 57.156 |
| 2 | 36.219 | 33.672 | 33.873 |
| 3 | 0.53 | 0.51 | 1.43 |
| 4 | 24.469 | 24.579 | 24.89 |
| 5 | 4.469 | 4.89 | 4.906 |
| 6 | 53.672 | 53.657 | 49.156 |
| 7 | 55.61 | 54.984 | 56.61 |
| 8 | 39.78 | 37.985 | 39.812 |
| 9 | 9.46 | 10.12 | 10.547 |

1. Hasil query tanpa partisi

Terlihat pada gambar 2.8, tabel yang tidak dipartisi memiliki waktu eksekusi query yang jauh lebih tinggi. Kesimpulan yang didapat adalah kecepatan queri berbanding terbalik dengan jumlah partisi.

1. DESAIN DATABASE MATA GARUDA

Pada desain database Mata Garuda versi pertama, desain yang digunakan adalah desain default dari snort dengan beberapa tabel tambahan untuk melengkapi data master yang digunakan untuk menyajikan laporan. Berikut pada gambar 2.3 adalah ER diagram dari skema default snort, diikuti keterangan dari setiap tabelnya.



1. Skema default snort

Deskripsi Tabel event

1. Tabel Event

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Key | Desc |
| sid | int | Pri | Sensor ID |
| cid | bigint | Pri | Event ID |
| signature | int | Mul | Signature ID |
| timestamp | datetime | Mul | Time of events logged |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *sensor (sid)*  *iphdr (sid,cid)*  *icmphdr (sid,cid)*  *tcphdr (sid,cid)*  *udphdr (sid,cid)*  *data (sid,cid)*  *signature (signature)* |
| Deskripsi | : | Tabel *event* berisi semua *event* yang berhasil disimpan oleh IDS Snort dan memiliki waktu terjadinya *event* |

Deskripsi Tabel sensor

1. Tabel Sensor

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Key | Desc |
| sid | int | Pri | Sensor ID |
| hostname | text |  | Hostname Sensor |
| interface | text |  | Network interface |
| filter | text |  | BPF filter |
| detail | tinyint |  | Detail level logging |
| encoding | tinyint |  | Encoding format of the payload |
| last\_cid | bigint |  | Last event ID |
| ip\_address | inet |  | IP address sensor |
| location | varchar |  | Sensor location |
| company | varchar |  | Sensor owner |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *event (sid)*  *detail (detail)*  *encoding (encoding)* |
| Deskripsi | : | Tabel *sensor* berisi informasi mengenai *sensor* yang terhubung dengan *database* |

Deskripsi Tabel iphdr

1. Tabel Iphdr

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Key | Desc |
| sid | int | Pri | Sensor ID |
| cid | bigint | Pri | Event ID |
| ip\_src | int |  | Source IP Address |
| ip\_dst | int |  | Destination IP Address |
| ip\_ver | tinyint |  | Ip version |
| ip\_hlen | tinyint |  | Ip header length |
| ip\_tos | tinyint |  | Ip type of service |
| ip\_len | smallint |  | IP datagram length |
| ip\_id | smallint |  | IP ID |
| ip\_flags | tinyint |  | IP flags |
| ip\_off | smallint |  | IP fragment offset |
| ip\_ttl | Tinyint |  | IP time to live |
| ip\_proto | tinyint |  | IP protocol |
| ip\_csum | smallint |  | IP checksum |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *event (sid,cid)* |
| Deskripsi | : | Tabel *iphdr* berisi semua informasi dari *iphdr* sebuah *event* |

Deskripsi Tabel icmphdr

1. Tabel ICMP Header

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Key | Desc |
| sid | Int | Pri | Sensor ID |
| cid | Bigint | Pri | Event ID |
| icmp\_type | smallint |  | ICMP Type |
| icmp\_code | smallint |  | ICMP Code |
| icmp\_sum | int |  | ICMP Summary |
| icmp\_id | int |  | ICMP Id |
| icmp\_sq | int |  | ICMP Sequence |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *event (sid,cid)* |
| Deskripsi | : | Tabel *icmphdr* memuat informasi dari *protocol* ICMP *event* |

Deskripsi Tabel tcphdr

1. Tabel Tcphdr

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Key | Desc |
| sid | int | Pri | Sensor ID |
| cid | bigint | Pri | Event ID |
| tcp\_sport | smallint |  | TCP source port |
| tcp\_dport | smallint |  | TCP destination port |
| tcp\_seq | int |  | TCP sequence number |
| tcp\_ack | int |  | TCP ACK number |
| tcp\_off | tinyint |  | TCP offset |
| tcp\_res | tinyint |  | TCP reserved |
| tcp\_flags | tinyint |  | TCP flags |
| tcp\_win | smallint |  | TCP window |
| tcp\_csum | smallint |  | TCP checksum |
| tcp\_urp | smallint |  | TCP urgent pointer |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *event (sid,cid)* |
| Deskripsi | : | Tabel *tcphdr* memuat informasi dari *protocol* TCP *event* |

Deskripsi Tabel udphdr

1. Tabel Udphdr

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Key | Desc |
| sid | int | Pri | Sensor ID |
| cid | bigint | Pri | Event ID |
| udp\_sport | smallint |  | UDP source port |
| udp\_dport | smallint |  | UDP destination port |
| udp\_len | smallint |  | UDP length |
| udp\_csum | smallint |  | UDP checksum |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *event (sid,cid)* |
| Deskripsi | : | Tabel *udphdr* memuat informasi dari *protocol* UDP *event* |

Deskripsi Tabel data

1. Tabel Data

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Key | Desc |
| sid | int | Pri | Sensor ID |
| cid | bigint | Pri | Event ID |
| data\_payload | text |  | Packet payload |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *event (sid,cid)* |
| Deskripsi | : | Berisi tentang informasi data *payload* untuk setiap *event* |

Deskripsi Tabel signature

1. Tabel Signature

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Key | Desc |
| sig\_id | int | Pri | Signature ID |
| sig\_name | varchar |  | Signature name |
| sig\_class\_id | int |  | Classification ID |
| sig\_priority | int |  | Priority |
| sig\_rev | int |  | Revision number |
| sig\_sid | int |  | Internal signature id |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *event (sig\_id)*  *sig\_class (sig\_class\_id)*  *sig\_reference (sig\_id)* |
| Deskripsi | : | Tabel *signature* merupakan salah satu tabel utama pada *database* IDS Snort, yang berisi semua *signature known threats* |

Deskripsi Tabel sig\_class

1. Tabel Sig\_class

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Key | Desc |
| sig\_class\_id | int | Pri | Signature classification ID |
| sig\_class\_name | varchar |  | Classification name |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *signature (sig\_class\_id)* |
| Deskripsi | : | Berisi klasifikasi dari *signature* yang ada |

Deskripsi Tabel sig\_reference

1. Tabel sig\_reference

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Key | Desc |
| sig\_id | int | Pri | Signature ID |
| ref\_seq | int | Pri | Reference sequence number |
| ref\_id | int |  | Reference ID |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *signature (sig\_id)*  *reference (ref\_id)* |
| Deskripsi | : | Tabel informasi *signature reference* |

Deskripsi Tabel references

Tabel references

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Key | Desc |
| ref\_id | int | Pri | Reference ID |
| ref\_system\_id | int |  | Referemce system ID |
| ref\_tag | varchar |  | Reference tag |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *sig\_reference (ref\_id)*  *reference\_system (ref\_system\_id)* |
| Deskripsi | : | Tabel *reference* dari *vendor* |

Deskripsi Tabel reference\_system

1. Tabel Reference\_system

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Key | Desc |
| ref\_system\_id | int | Pri | Reference system ID |
| ref\_system\_name | varchar |  | Reference system name |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | *reference(ref\_system\_id)* |
| Deskripsi | : | Tabel sistem *reference* |

Deskripsi Tabel ip\_location

1. Tabel ip\_location

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Key | Desc |
| ip\_start | inet | Pri | Start IP address |
| ip\_end | inet | Pri | End IP address |
| country | text |  | Country of IP address |
| stateprov | text |  | State of IP address |
| city | text |  | City of IP address |
| latitude | double |  | Latitude |
| longitude | double |  | Longitude |
| timezone\_offset | varchar |  | Timezone offset |
| timezone\_name | text |  | Timezone name |
| isp\_name | text |  | Internet Service Provider name |
| connection\_type | text |  | Connection type |
| organization\_name | text |  | Organization owner |

|  |  |  |
| --- | --- | --- |
| Relasi tabel | : | - |
| Deskripsi | : | Tabel *ip\_location* memuat semua daftar *ip address* yang ada di dunia beserta informasinya. Tabel ini tidak berelasi dengan tabel manapun, namun penggunaanya akan dibutuhkan pada fungsi aplikasi Mata Garuda. |

Satu tabel yang digunakan untuk melengkapi report adalah tabel ip\_location, tabel ini berisi range ip beserta keterangan dari ip tersebut, seperti country, longitude, longitude, dsb. Query yang melibatkan tabel ini sangat menyita waktu, karena bentuk query belum dapat memanfaatkan index yang ada pada tabel. Sehinnga dibutuhkan satu tabel yang dinamakan ip\_map.

1. DESAIN DATABASE APLIKASI SEJENIS

* Snorby

Salah satu aplikasi sejenis yang berperan sebagai UI monitoring keamanan jaringan adalah snorby. Disini desain database yang digunakan snorby adalah juga modifikasi dari desain database bawaan snort.

1. List Tabel Snorby

|  |  |  |
| --- | --- | --- |
| agent\_asset\_names | favorites | search |
| aggregated\_events | icmphdr | sensor |
| asset\_names | iphdr | settings |
| caches | lookups | severities |
| classifications | notes | sig\_class |
| data | notifications | sig\_reference |
| delayed\_jobs | opt | signature |
| detail | reference | tcphdr |
| encoding | reference\_system | udphdr |
| event | schema | users |
| events\_with\_join |  |  |

|  |  |
| --- | --- |
|  | Tabel default ACID snort |
|  | Tabel tambahan dari snorby |
|  | View tambahan dari snorby |

Dapat dilihat dari tabel 2.14 diatas, snorby juga menggunakan tabel bawaan dari snort untuk memproses data event dari IDS. Disini snorby menambahkan beberapa tabel dan view untuk mendukung kinerjanya. Yang saya tangkap dari desain snorby ini adalah, agregasi event menggunakan view, sedangkan view merupakan sebuah query yang disimpan. Dengan kesimpulan yang saya ambil bahwa untuk performa tidak akan berubah secara signifikan.

1. KEUNIKAN DESAIN

Keunikan dalam desain yang diajukan disini ada pada penerapan Partition Table yang dirasa akan sangat membantu pengolahan data dalam jumlah besar. Penerapan OLTP dan OLAP, sehingga beban tidak berada pada satu database server. Sensor memiliki database tersendiri sebagai database OLTP yang berinteraksi langsung dengan event yang ditangkap oleh sensor. Database OLAP akan mengambil field-field yang dirasa saat ini perlu disimpan. Database OLAP disini bersifat archieve, sehingga data yang diterima bukan data agregasi. Selanjutnya data tersebut diaggregasi di dalam lingkup database OLAP. Selain itu optimasi-optimasi lain yang dimungkinkan akan diterapkan pada proyek akhir ini.

1. QUERY REPORT MATA GARUDA

Report pada aplikasi monitoring keamanan jaringan mata garuda mengacu pada beberapa aplikasi sejenis serta penelitian yang membahas tentang sistem monitoring keamanan jaringan. Sebelum masuk ke list report yang dibutuhkan, perlu diketahui pada report mata garuda dibagi menjadi 3 bagian , yaitu daily, monthly, dan annual, disetiap report akan memiliki opsi filter yang terdiri dari sensor pencatat, ip\_src, dan ip\_dst.

1. List Query Report Mata Garuda

|  |  |
| --- | --- |
| NAMA QUERY | DESKRIPSI |
| Display Event | Menampilkan event dengan format ( datetime, sid, cid, ip\_src, ip\_dst, signature\_name, signature\_priority, signature\_class, protocol, s\_port, d\_port ) |
| Top 10 by Signature | Menampilkan jenis sepuluh serangan tertinggi dengan pengelompokan berdasarkan jenis serangan. |
| Top 10 by Signature Class | Menampilkan jenis sepuluh serangan tertinggi dengan pengelompokan berdasarkan klasifikasi serangannya. |
| Event by Severity | Melaporkan jenis serangan berdasarkan prioritas ancaman yang terbagi menjadi HIGH, MEDIUM, LOW. |
| Top 10 by ip\_source | Menampilkan 10 ip asal serangan dengan kontribusi tertinggi. |
| Timeline Top 10 ip\_src | Menampilkan timeline dari ip asal serangan |
| Top 10 by ip\_destination | Menampilkan 10 ip target serangan dengan kontribusi tertinggi. |
| Timeline Top 10 ip\_dst | Menampilkan timeline dari ip target serangan |

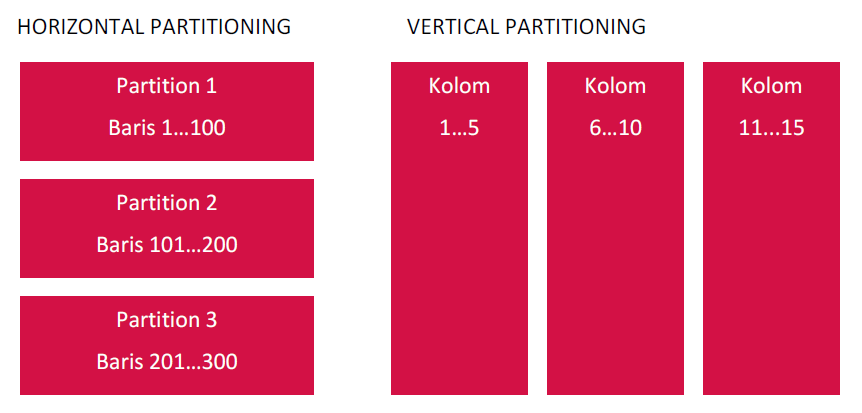
1. DASAR TEORI

Berikut adalah ebebrapa teori penunjang mengenai topik-topik yang berkaitan dengan proyek akhir ini:

1. Partition Table

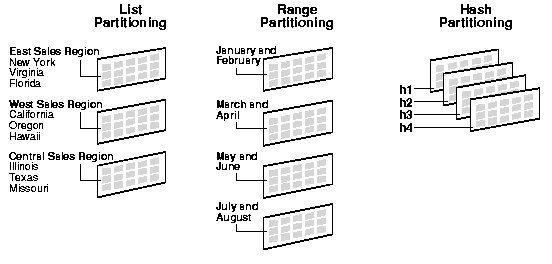
Secara definisi partisi tabel adalah memecah tabel menjadi beberapa bagian/segmen, tujuan dari menggunakan partisi adalah untuk mempercepat proses query. Misalkan kita memilik data sebanyak 10 juta record, bila pada tabel konvensional maka data sebanyak itu akan di simpan dalam satu segmen, tetapi bila menggunakan partisi maka data sebanyak itu akan di pecah-pecah ke banyak partisi berdasarkan sebuah kondisi, misalkan berdasarkan tanggal, maka ketika melakukan query hanya men-scan segmen dimana data itu berada, tidak 10 juta record itu di scan, sehingga proses query menjadi lebih cepat.

Dalam masalah pemartisian tabel dalam database, dikenal 2 jenis permartisian, yaitu berdasarkan row dan bedasarkan kolom. Berdasarkan row atau horizontal partition, kita membagi tabel menjadi tabel yang lebih kecil dengan aturan yang membagi row datanya. Sedangkan berdasarkan kolo atau vertical partition, kita memecah tabel dengan 10 kolom misalnya, menjadi 2 tabel dengan kolom 5 setiap tabelnya. Kedua jenis pemartisian tersebut memiki tujuan dan keuntungan sendiri-sendiri. Manfaat lain dari partitioning adalah tiap-tiap segment (partisi atau subpartisi) bisa ditaruh di tabelspace yang berbeda, sehingga kita mendapat manfaat dari spreading (menyebar) tabelspace, yaitu penyebaran I/O dan mengurangi resiko loss data karena tabelspace corrupt.



1. Vertical dan Horizontal Partitioning

Pada pemartisian horizontal dikenal 3 metode utama partisi, dan ada 2 macam composite (gabungan):



1. Metode Partitioning Tabel

* Range partitioning

Yaitu membagi tabel berdasarkan nilai range tertentu, salah satu contohnya adalah membagi tabel berdasarkan hari. Jadi jika kita ingin membagi tabel berdasarkan hari, kita dapat melakukan partitioning dengan partition function atau aturan partisi yang akan mengarahkan data yang masuk ke tabel-tabel partisi per-hari masing-masing. Jadi jika kita menerapkan ini untuk satu tahun, akan tercipta 365 partisi.

* List partitioning

Konsep pada list partition adalah data dikelompokkan berdasarkan nilai datanya. Cocok untuk kolom yang variasi nilainya tidak banyak. Misal data kota berdasarkan wilayah provinsi. List: Jember, Malang, Surabaya pada partisi Jawa Timur; Bandung, Cirebon pada partisi Jawa Barat. Jadi list partition ini berdasarkan list dari suatu segmen, sehingga data-datanya di list terlebih dahulu.

* Hash partitioning

Penggunaan hash partition ini jika tidak cocok dengan RANGE ataupun LIST Partition. Penentuan “nilai mana di taruh di partisi mana” itu diatur secara internal oleh database (berdasarkan hash value).

* Composite range-list partitioning

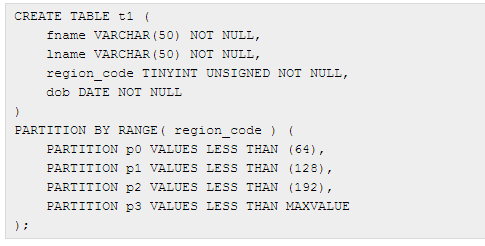
Konsep composite range-list partition ini menggabungkan partisi range dan list. Jika pada tabel mahasiswa, NIM dipartisi secara range, sedangkan untuk mendapatkan NIM sekian yang tinggal di wilayah provinsi Jawa Timur itu apa saja, maka menggunakan list. Jadi dipartisi kemudian di list berdasarkan kriteria tertentu.

* Composite range-hash partitioning

Composite range-hash partition merupakan konsep menggabungkan range partition dan hash partition. Sehingga partisi-partisi yang telah dibuat berdasarkan range/rentang akan dipartisi lagi ke dalam beberapa bagian berdasarkan hash.

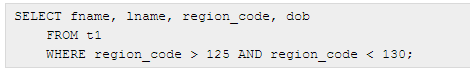
1. Partitioning Pruning

Partition pruning merupakan salah satu optimasi yang dapat dilakukan pada desain database dengan menerapkan permatisian pada suatu tabel. Konsep dari partision pruning sangatlah mudah, *“Do not scan partitions where there can be no matching values”.* Sebagai contoh dapat dilihat permatisian tabel t1 pada gambar 2.13 dibawah yang menerapkan pemartisian secara range.



1. Contoh range partition

Ketika kita melakukan query SELECT pada tabel tersebut dengan menggunakan sample query seperti pada gambar 2.14 dibawah, amaka yang terjadi adalah partisi yang diakses hanya partisi p1.



1. Case pengujian partisi

Dengan melakukan ini akan memungkinkan berkurangnya beban serta waktu eksekusi ketika mencari sebuah row data dibandingakan harus membaca pada tabel yang berukuran besar. Kemampuan mengacuhkan partisi yang tidak mungkin memiliki nilai ini disebut dengan partition pruning. Ketika suatu optimizer database dapat memanfaatkan partition pruning dalam menjalankan query, maka waktu eksekusi dapat berjalan lebih cepat. [7]

1. SQL ( Structured Query Language )

Adalah sebuah standard atau Bahasa yang digunakan untuk melakukan operasi yang berkaitan dengan databaseseperti proses melakuakan retrieve data, manipulasi data, dan menambahkan data pada database. SQL biasanya dibagi menjadi 2 kategori yaitu DML ( Data Manipulation Language ) dan DDL ( Data Definition Language ). Selain itu beberapa sumber juga menyebutkan beberapa kategori lain, diantaranya adalah DCL ( Data Control Language ) dan TCL ( Transactional Control ).

Berikut adalah penjelasan singkat mengenai macam-macam kategori SQL diatas :

1. DDL ( Data Definition Language )

Data Definition Language, adalah perintah SQL yang digunakan untuk memanipulasi database dan schema, daripada manipulasi pada data di dalamnya. Perintah DDL diantaranya adalah perintah CREATE, ALTER dan DROP., selain itu juga perintah TRUNCATE. Perintah truncate berurusan dengan data mirip dengan perintah delete, namun perintah truncate memiliki cara kerja yang berbeda. Perintah DDL akan secara otomatis tercommit, dan tidak dapat dilakukan rollback.

1. DML ( Data Manipulation Language )

Data Manipulation Language adalah perintah SQL yang melakukan operasi INSERT, UPDATE dan DELETE. Perintah SELECT dapat juga digolongkan pada kategori ini. Berbeda dengan perintah DDL, perintah DML dapat dilakukan commit dan rollback secara manual.

1. DCL ( Data Control Language )

Data Control Language merupakan kategori perintah SQL yang mengarah pada pemberian izin, seperti halnya perintah GRANT dan REVOKE.

1. TCL ( Transactional Control )

Kategori ini lebih mengarah bagaimana transaksi dijalankan, perintah-perintah yang bekerja untuk mengontrol transaksi. Diantara perintah tersebut adalah COMMIT, ROLLBACK, SAVEPOINT, dan TRANSACTION.

1. Join

Join merupakan bentuk query yang melibatkan lebih dari satu tabel. Ada beberapa jenis join, diantaranya adalah INNER JOIN, LEFT JOIN, RIGHT JOIN, dll.

1. Indexing

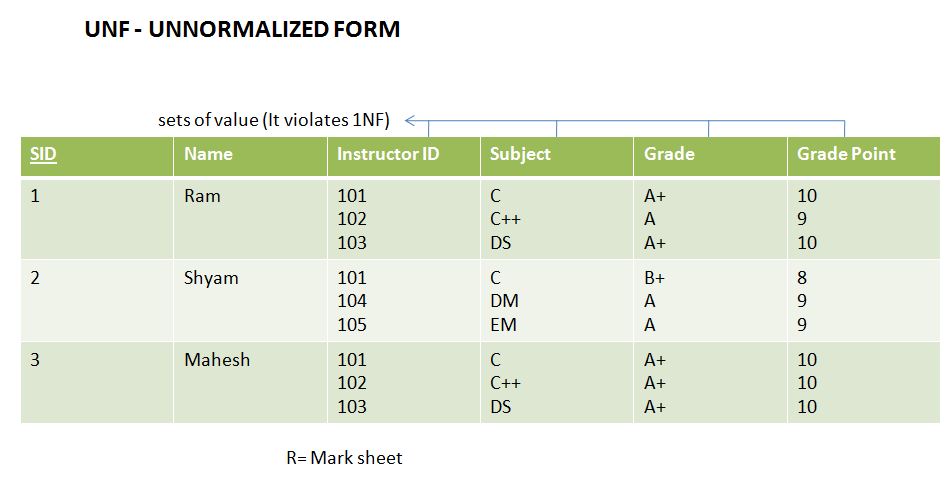
Tunjuan utama index adalah untuk menemukan row dengan kolom tertentu secara cepat. Tanpa index suatu database akan melakukan pencarian satu-persatu dari awal tabel hinnnga akhir untuk menemukan row yang sesuai. Dengan tanpa menggunakan index pencarian pada tabel yang besar akan memakan waktu yang lebih lama. Jika sebuah tabel memiliki index, dan query yang diberikan user dapat menggunakan index, maka database akan menggunakannya untuk pencarian lebih cepat dengan melihat data dari tengah tabel menggunakan metode B-TREE atau HASH tanpa harus membaca tabel secara keseluruhan. [7]

1. Query Optimizer

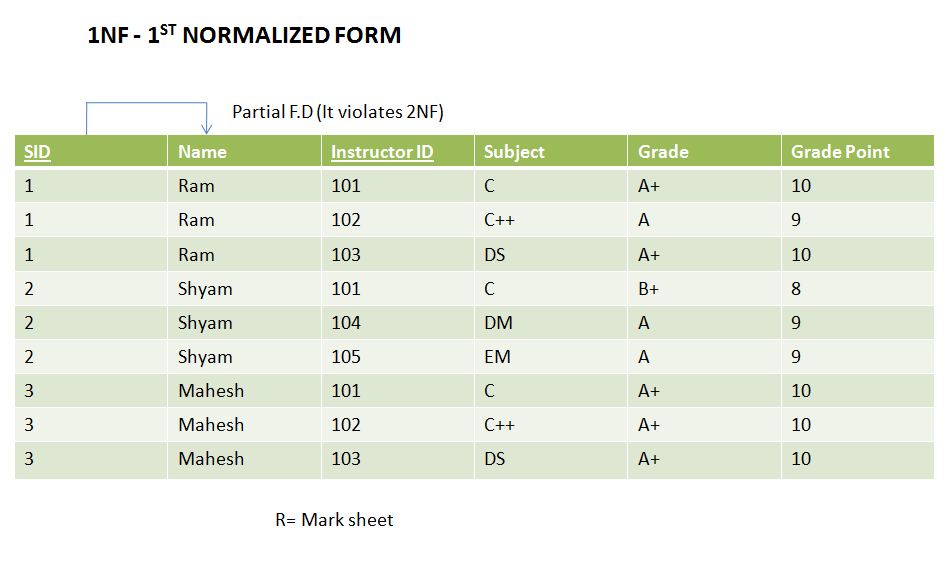
Adalah salah satu kompone pada DBMS yang bertugas untuk menetukan bagaimanan cara paling baik dalam menjalankan query yang diberikan oleh user. Query optimizer dapat menentukan cara mana yang akan diambil dalam menjalankan query dengan menggunakan perhitungan cost model, seperti cardinality dari suatu tabel, jumlah tuple dalam tabel dan estimasi dari effort yang akan diperlukan dalam menjalankan cara tersebut. Untuk dapat mengira-ngira secara baik kardinalitas dari sebuah hasil query, cost model perlu untuk mengkalkulasi berapa banyak tuples yang di pilih oleh query. Query optimizer bekerja pada semua layer database ( conceptual, logical, phisycal ). Pada layer konseptual optimizer bertugas untuk mentranslasi query ke bentuk yang standar. Pada layer logical optimizer menata langkah pencarian jawaban dari query yang diberikan user. Ketika ekspresi logical masuk ke layer phisycal optimizer akan menggunakan index struktur untuk mempercepat query, seperti peletakan tabelspace, disk, dsb. [3]

1. Normalisasi

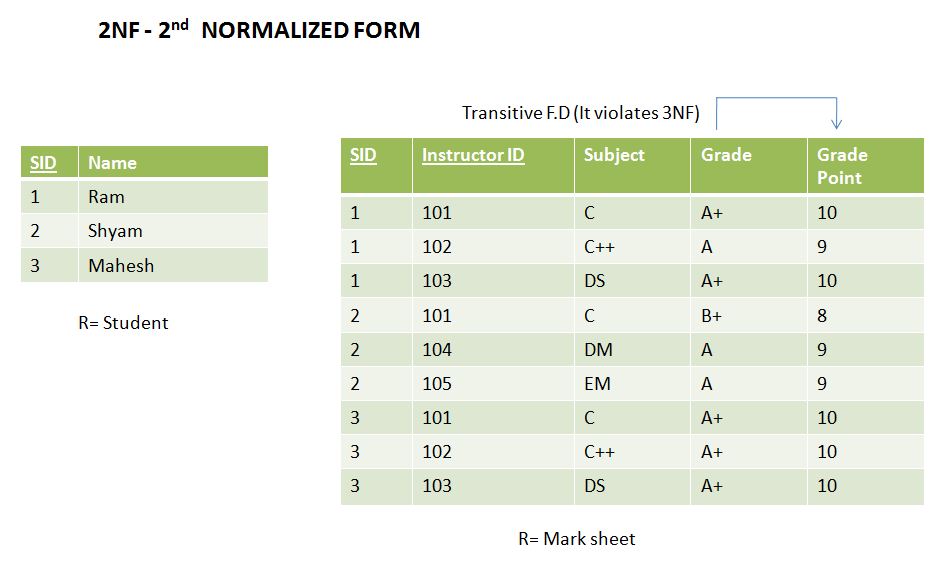
Normalisasi bertujuan untuk mengurangi redundancy data pada database. Normali sasi sangat baik untuk mengurangi ukuran database, namun normalisasi merupakan lawan dari performa pencarian dari query. Pada operasi yang melibatkan data yang besar, normalisasi dapat mengurangi perfoma pencarian. Normalisasi sering digunakan pada database OLTP. Pada Normalisasi dikenal 5 tahap penormalan data, 1NF, 2NF, 3NF, BCNF, 4NF dan 5NF. [8]



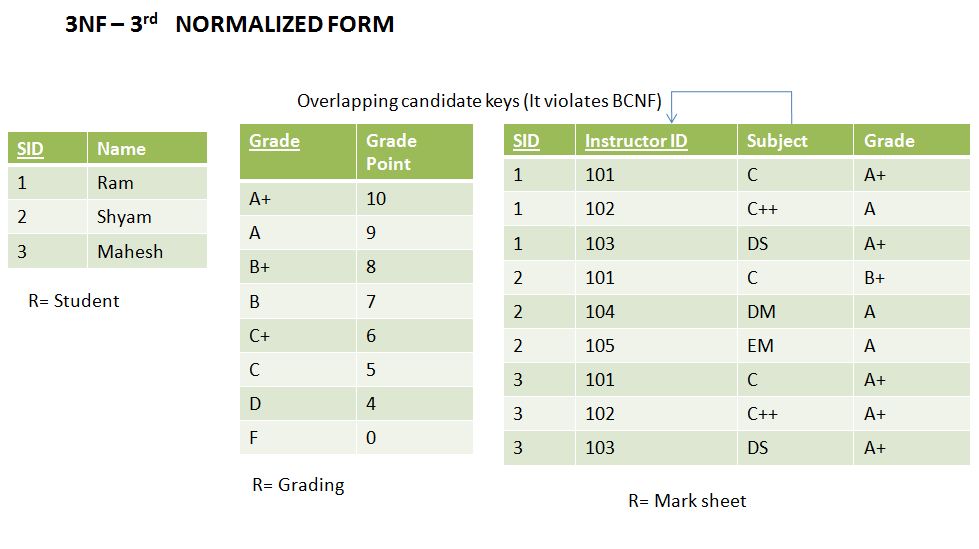
1. UNF ( UNNORMALIZED FORM )



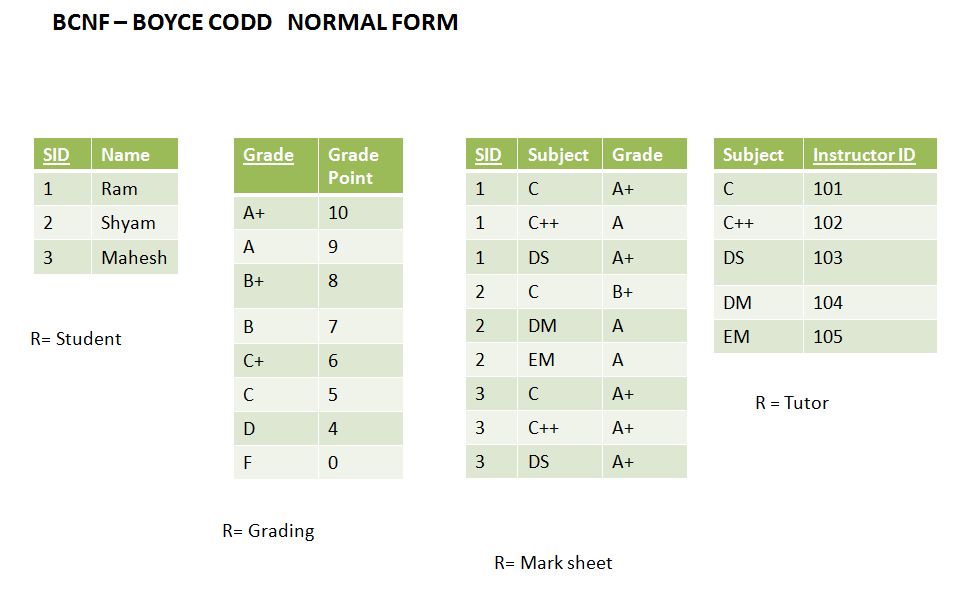
1. 1NF NORMAL FORM 1



1. 2NF NORMAL FORM 2



1. 3NF NORMAL FORM 3

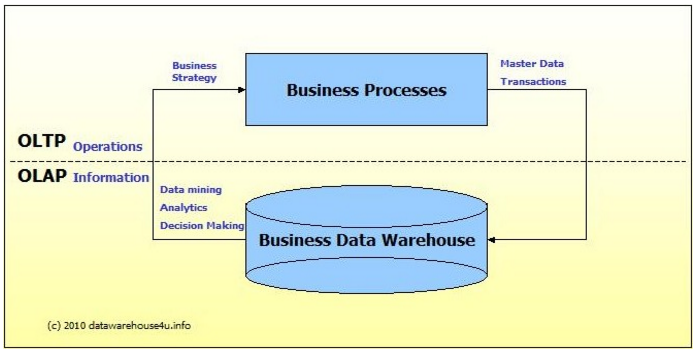


1. BCNF BOYCE-CODD
2. Denormalisasi

Berbeda dengan normalisasi, denormalisasi banyak diterapkan pada database OLAP. Denormalisasi mengorbankan bentuk data normal sehingga memungkinkan adanya data yang redundant, dengan bergitu performa pencarian data akan lebih cepat dikarenakan join antara tabel akan lebih sedikit. [8]

1. OLTP vs OLAP [9]

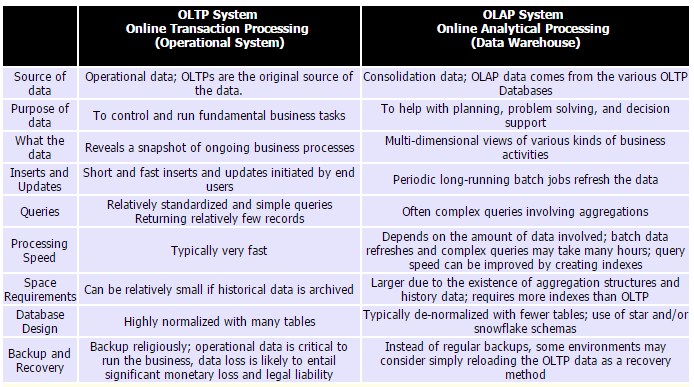
Kita dapat membagi sistem IT dua, yaitu sistem transaksi ( OLTP ) dan analisis ( OLAP ). Secara garis besar, OLTP adalah sebuah sistem yang menyediakan data untuk datawarehouse, sedangakan OLAP adalah sistem yang berfungsi untuk membantu menganalisa.



1. OLTP vs OLAP

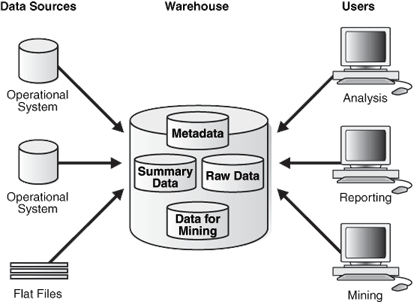
OLTP atau ( On-line Transaction Processing ) dapat dilihat melalui queri yang berjalan yang menggunakan (INSERT, UPDATE, DELETE). Pada sistem OLTP hal yang utama adalah query yang cepat, integritas data, maintenance data yang dapat diakses dari banyak tempat. Didalam OLTP data merupakan data sekarang dan bentuk schemanya merupakan bentuk normal.

OLAP atau (On-line Analytical Processing) dapat dilihat dari sedikitnya transaksi yang terjadi, Query yang ditujukan pada sitem OLAP biasanya sangat kompleks dan biasanya juga membutuhkan aggregasi. Dalam OLAP respon dari sebuah sistem adalah sangat penting. Dalam aplikasi OLAP biasanya dilakukan penggalian informasi dari data ( data mining ). Didalam OLAP database di aggregasi, berisi data historical, dan disimpan dalam bentuk multidimensional. Berikut pada gambar 2.21 adalah rangkuman perbandingan antara sistem OLTP dan sistem OLAP.



1. Perbandingan OLTP vs OLAP
2. Data Warehouse [10]

Data Warehouse (yang secara harafiah berarti gudang data) adalah kumpulan seluruh data yang dimiliki oleh perusahaan yang didesain untuk melakukan analisis dan pelaporan, dan bukan untuk pemrosesan transaksi. Jadi, dalam istilah yang lebih sederhana, data warehouse adalah basis data yang ditujukan untuk analisis, pelaporan, dan terkadang juga untuk penambangan pengetahuan (knowledge mining).



1. Gambaran Data Warehouse

Menurut Vidette Poe, data warehouse merupakan database yang bersifat analisis dan read only yang digunakan sebagai fondasi dari sistem penunjang keputusan.

Menurut Paul Lane, data warehouse merupakan database relasional yang didesain lebih kepada query dan analisa dari pada proses transaksi, biasanya mengandung history data dari proses transaksi dan bisa juga data dari sumber lainnya. Data warehouse memisahkan beban kerja analisis dari beban kerja transaksi dan memungkinkan organisasi menggabung/konsolidasi data dari berbagai macam sumber.

Dari definisi-definisi yang dijelaskan tadi, dapat disimpulkan data warehouse adalah database yang saling bereaksi yang dapat digunakan untuk query dan analisisis, bersifat orientasi subjek, terintegrasi, time-variant,tidak berubah yang digunakan untuk membantu para pengambil keputusan.

1. ETL [9]

ETL (Extract, Transform and Load) dalah process dalam datawarehouse yang berperan untuk mengambil data dari sebuah source data dan menempatkan serta menyesuaikannya pada sistem data warehouse. ETL dibagi menjadi beberapa sub-tugas sebsgai berikut :

* Extract

Pertama adalah melakuakn ekstaksi data dari sebuah source data ( SAP, ERP, OLTP, dll ), darisini data disiapkan agar menyesuaikan format yang diinginkan oleh datawarehousedan selanjutnya dapat dilakukan proses transform. Proses ini dapat menggunakan staging tabel atau melalui flat-file.

* Transform

Transformasi data terdiri beberapa step berikut, agar didapatkan data yang sesuai dengan yang dibutuhkan datawarehouse.

1. Jika diperlukan dapat dilakukan pengitungan tersendiri sebelum data masuk ke datawarehouse.
2. Cleaning, melakukan pembersihan dari data yang jelek, seperti data null, dan data yang tidak memiliki arti, yang akan menjadi noise pada saat data di mining.
3. Filtering, memilah-milah kolom apa saja yang mungkin dibutuhkan pada datawarehouse.
4. Menata kolom dengan penyesuaian sesuai dengan format data warehouse, seperti memecah kolaom atau sebaliknya.
5. Melakukan join dari source data yang lain.
6. Transpose data.
7. Melakuakn validasi data sebelum data dimasukkan ke sistem data warehouse.

* Load

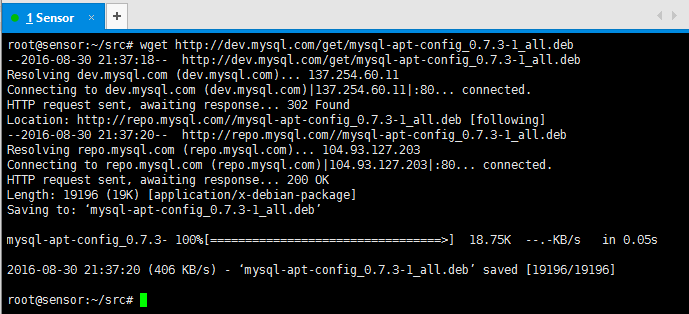
Mengunggah atau memasukkan data pada datawarehouse.

1. Instalasi MySQL 5.7

Instalasi MySQL akan menggunakan apt dengan repo mysql. Pertama perlu ditambahkan source repo mysql agar didapatkan versi mysql terbaru.

1. File .deb untuk repo mysql dapat di download di website resmi mysql.

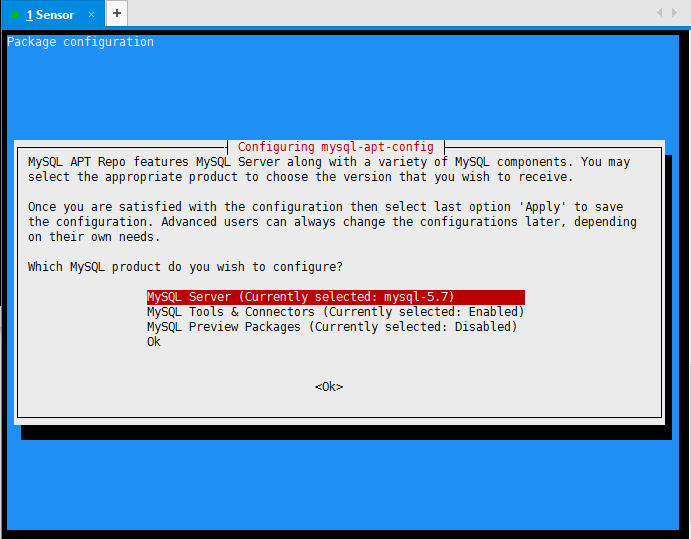
:~# wget http://dev.mysql.com/get/mysql-apt-config\_0.7.3-1\_all.deb



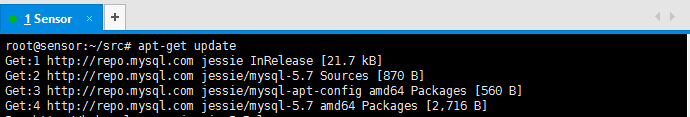
1. Download repo mysql
2. Selanjutnya kita perlu menampahkan source tersebut ke apt source OS kita.

:~# dpkg –i mysql-apt-config\_0.7.3-1\_all.deb

1. Pilih versi yang ingin ditambahkan ke apt source, pada proyek akhir ini saya menggunakan mysql versi 5.7.

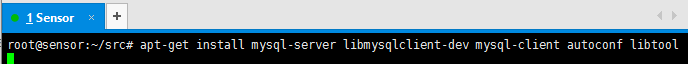


1. Pilih versi mysql
2. Lakukan update agar source yang baru ditambahkan siap digunakan.



1. Update list repo
2. Selanjutnya instalasi mysql 5.7 dapat dilakukan dengan perintah berikut. Selain mengintall mysql-server kita juga akan memerlukan beberapa library untuk melengkapi mysql-ser

:~# apt-get install mysql-server libmysqlclient-dev mysql-client



1. Instalasi Mysql 5.7

*HALAMAN INI SENGAJA DIKOSONGKAN*

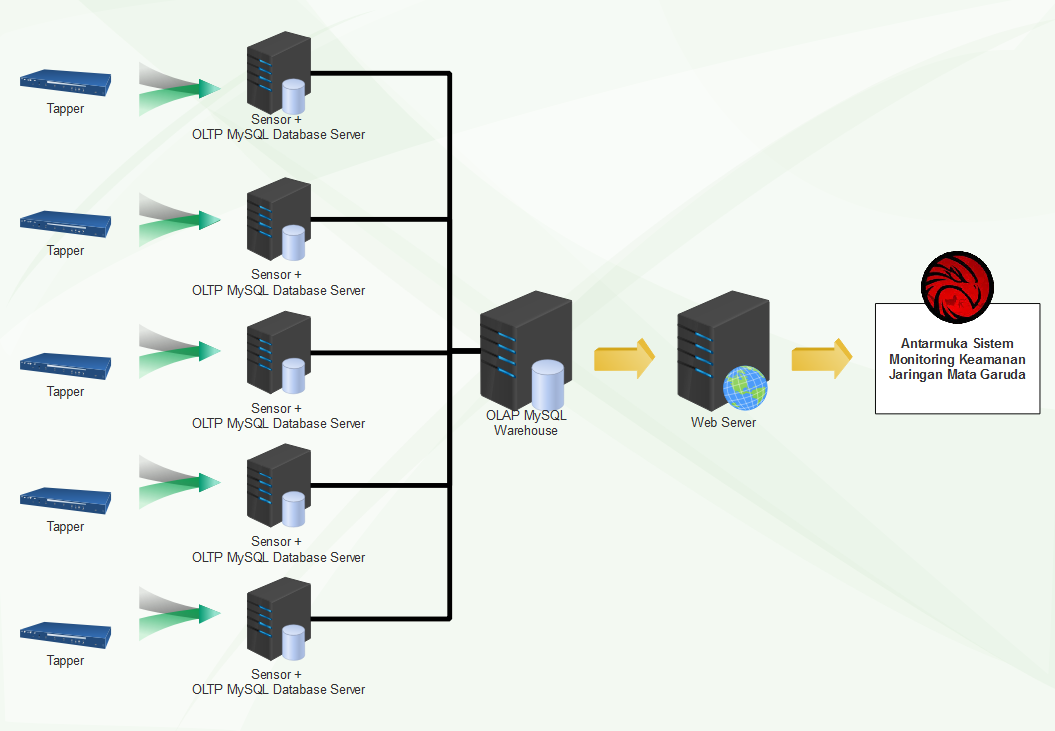
BAB III

PERANCANGAN SISTEM

Pada bab ini, akan dibahas metodologi dan desain sistem yang digunakan pada perancangan dan pembangunan desain yang dikerjakan pada proyek akhir ini. Pada proyek akhir ini dilakukan migrasi dari engine PostgreSQL ke MySQL dengan alasan-alasan yang akan diuraikan dibawah.

* 1. DESKRIPSI UMUM SISTEM

Berikut adalah gambaran umum dari desain database yang telah dikerjakan pada projek akhir ini. Beserta solusi-solusi yang berhasil diterapkan pada optimasi desain database yang saya kerjakan pada proyek akhir ini.



1. Deskripsi Umum Desain Database

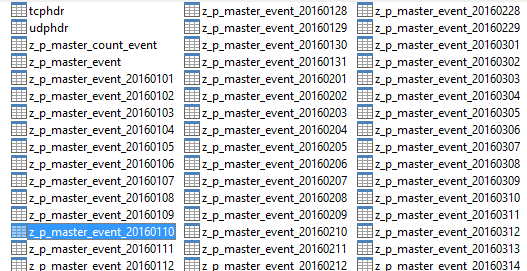
Pada proyek akhir ini perubahan desain berada pada alur aliran data dari sensor hingga data siap ditampilkan ke user interface. Pada desain sebelumnya data dari sensor langsung mengarah ke satu database utama, pengolahan juga menggunakan satu database server, sehingga sangat membebani, dengan data yang masuk cukup deras dari sensor. Pada desin database ini setiap sensor akan ditambahkan database tersendiri, sehingga input event dari snort tidak langsung masuk ke satu database saja, tetapi ditangani oleh database sensor masing-masing. Selanjutnya data dari sensor akan dikirim ke database utama atau warehouse melalui prtoses ETL, data yang dikirim disesuaikan dengan analisa trend yang dibutuhkan dari query Mata Garuda. Pengiriman data disini diatur periodic setiap 15 menit sekali. Data pada database sensor akan dihapus secara periodic, sehingga meninggalkan data 7 hari terakhir. Flow pengiriman data dapat dilihat pada sub-bab 3.3.1 dibawah.

* 1. MIGRASI DATABASE ENGINE

Pada proyek akhir ini dilakukan migrasi engine database dari engine database PostgreSQL ke MySQL dengan pertimbagnan sebagai berikut :

1. Metode Partisi PostgreSQL

Kendala yang pertama ketika dilakukan penerapan Partition Table di PostgreSQL adalah metode pemartisian masih dilakukan secara manual dengan memanfaatkan triger untuk membuat partisi dan mengarahkan insert event ke tabel partisi. Metode ini sesuai dengan arahan yang ada pada website resmi PostgreSQL, bahwa penerapan partisi tabel di PostgreSQL saat ini dapat dilakukan dengan menggunakan triger. [9] Dari metode tersebut tabel partisi terlihat layaknya tabel biasa, tidak tersembunyi, sehingga akan terlihat tabel yang cukup banyak dan tidak rapi, pada kasus ini lebih dari 300 tabel tercipta karena dilakukan pemartisian berdasarkan hari. Sedangakan di MySQL partisi dari sebuah tabel tidak ditampilkan, serta pembuatannya lebih terstruktur.



1. Tabel-tabel hasil partitioning di PostgreSQL
   1. PERANCANGAN SISTEM

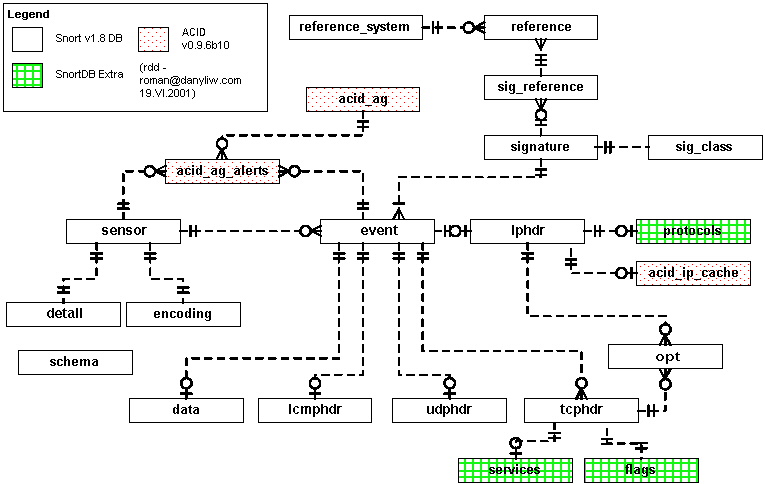
Pada tahap ini dilakukan proses perancangan alur pengiriman data dari sensor ke warehouse, desain database warehouse dan scenario pengolahan data hingga siap untuk ditampilkan.

1. Entity Relationship Diagram

Pada bagian ini akan dijelaskan ERD dari database sensor dan datawarehouse.

1. Database Sensor

Pada sisi sensor akan menggunakan schema database default dari snort. Dari hasil percobaan database MySQL dengan skema default ini mampu mengelola data selama satu minggu lebih. Dengan inputan hanya dari satu sensor beban yang diterima server ditak terlalu berat .



1. Snort Default Schema
2. Database Warehouse

Pada sisi warehouse, tabel didesain untuk mengurangi join dengan menerapkan denormalisasi dari tabel-tabel yang ada pada skema default snort. Berikut adalah tabel-tabel yang ada pada sis warehouse.

1. Tabel event

Tabel event disini merupakan tabel utama yang langsung menerima data dari hasil load ETL. Tabel event akan menampung raw data yang natinya akan di aggregasi sebelum di query menjadi report. Pada tabel ini dilakukan partisi secara horizontal menggunakan jenis partisi secara range berdasarkan hari, sehingga akan tercipta 356 partisi tiap tahunnya. Berikut adalah cuplikan kode pembuatan tabel event ber partisi.

|  |
| --- |
| CREATE TABEL `event` (  `long\_date` datetime NOT NULL,  `code` varchar(11) CHARACTER SET utf8 NOT NULL,  `sid` int(4) NOT NULL,  `cid` int(10) unsigned NOT NULL,  `ip\_src` int(10) unsigned NOT NULL,  `ip\_dst` int(10) unsigned NOT NULL,  `signature\_id` int(11) NOT NULL,  `signature\_name` varchar(255) NOT NULL,  `signature\_priority` tinyint(3) NOT NULL,  `sig\_class\_id` int(11) NOT NULL,  `sig\_class\_name` varchar(100) NOT NULL,  `protocol` int(11) NOT NULL,  `s\_port` int(11) NOT NULL,  `d\_port` int(11) NOT NULL,  `detail` int(1) NOT NULL DEFAULT '0'  ) ENGINE=InnoDB  PARTITION BY RANGE( TO\_DAYS(long\_date) ) (  PARTITION p20170101 VALUES LESS THAN ( TO\_DAYS('2017-01-02') ),  PARTITION p20170102 VALUES LESS THAN ( TO\_DAYS('2017-01-03') ),  PARTITION p20170103 VALUES LESS THAN ( TO\_DAYS('2017-01-04') ),  PARTITION p20170104 VALUES LESS THAN ( TO\_DAYS('2017-01-05') ),  PARTITION p20170105 VALUES LESS THAN ( TO\_DAYS('2017-01-06') ),  PARTITION p20170106 VALUES LESS THAN ( TO\_DAYS('2017-01-07') ),  PARTITION p20170107 VALUES LESS THAN ( TO\_DAYS('2017-01-08') ),  PARTITION p20170108 VALUES LESS THAN ( TO\_DAYS('2017-01-09') ),  . . . . . .  PARTITION p20171229 VALUES LESS THAN ( TO\_DAYS('2017-12-30') ),  PARTITION p20171230 VALUES LESS THAN ( TO\_DAYS('2017-12-31') ),  PARTITION p20171231 VALUES LESS THAN ( TO\_DAYS('2018-01-01') ),  PARTITION pMax VALUES LESS THAN (MAXVALUE)); |

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *long\_date* | *datetime* | *Pk* | *waktu event diterima sensor.* |
| *code* | *varchar* |  | *allias kolom long\_date* |
| *sid* | *int* | *Pk* | *sensor asal data serangan* |
| *cid* | *int* | *Pk* | *unique id dari event.* |
| *ip\_src* | *int* |  | *ip asal serangan* |
| *ip\_dst* | *int* |  | *ip target serangan* |
| *signature\_id* | *int* |  | *id jenis serangan dari event* |
| *signature\_name* | *varchar* |  | *nama serangan dari event* |
| *signature\_priority* | *int* |  | *tingkat ancaman dari event* |
| *sig\_class\_id* | *int* |  | *id klasifikasi serangan dari event* |
| *sig\_class\_name* | *varchar* |  | *nama klasifikasi dari serangan* |
| *protocol* | *int* |  | *protocol yang digunakan dalam serangan* |
| *s\_port* | *int* |  | *port asal serangan* |
| *d\_port* | *int* |  | *port target serangan* |
| *detail* | *int* |  | *informasi tambahan, bernilai ( 0 | 1 )* |

1. Tabel daily\_event\_aggregation

Tabel daily\_event\_aggregation adalah dable aggregasi yang akan diakses pada query report harian. Nilai waktu terkecil pada tabel ini adalah jam, disesuaikan dengan permintaan query dari report harian. Sehinnga jumlah row akan sangat menyusut, dan berdampak pada performa query. Aktivitas aggregasi harian dilakukan setiap suatu event yang masuk dari sensor sudah genap berisi 1 jam.

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *date* | *date* |  | *tanggal diterimanya event serangan* |
| *hour* | *tinyint* |  | *jam terjadinya serangan* |
| *sid* | *int* |  | *sensor asal data serangan* |
| *ip\_src* | *int* |  | *ip asal serangan* |
| *ip\_dst* | *int* |  | *ip target serangan* |
| *signature\_id* | *int* |  | *id jenis serangan dari event* |
| *signature\_name* | *varchar* |  | *nama serangan dari event* |
| *signature\_priority* | *tinyint* |  | *tingkat ancaman dari event* |
| *sig\_class\_id* | *int* |  | *id klasifikasi serangan dari event* |
| *sig\_class\_name* | *varchar* |  | *nama klasifikasi dari serangan* |
| *protocol* | *int* |  | *protocol yang digunakan dalam serangan* |
| *s\_port* | *int* |  | *port asal serangan* |
| *d\_port* | *int* |  | *port target serangan* |
| *counter* | *int* |  | *sum aggregation dari serangan* |

1. Tabel monthly\_event\_aggregation

Tabel aggregasi dengan nilai waktu terkecil adalah date, aktivitas aggregasi ini dilakukan setiap suatu event dari sensor sudah genap satu hari.

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *date* | *date* |  | *tanggal terjadi serangan* |
| *sid* | *int* |  | *sensor asal data seranngan* |
| *ip\_src* | *int* |  | *ip asal serangan* |
| *ip\_dst* | *int* |  | *ip target serangan* |
| *signature\_id* | *int* |  | *id jenis serangan dari event* |
| *signature\_name* | *varchar* |  | *nama serangan dari event* |
| *signature\_priority* | *tinyint* |  | *tingkat ancaman dari event* |
| *sig\_class\_id* | *int* |  | *id klasifikasi serangan dari event* |
| *sig\_class\_name* | *varchar* |  | *nama klasifikasi dari serangan* |
| *protocol* | *int* |  | *protocol yang digunakan dalam serangan* |
| *s\_port* | *int* |  | *port asal serangan* |
| *d\_port* | *int* |  | *port target serangan* |
| *counter* | *int* |  | *sum aggregation dari serangan* |

1. Tabel annual\_event\_aggregation

Tabel aggregasi tahunan dengan nilai terkecil adalah bulan, sehingga aktivitas aggregasi dapat dilakukan setelah event dari suatu sensor genap satu bulan.

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *month* | *int* |  | *bulan serangan terjadi* |
| *sid* | *int* |  | *sensor asal data serangan* |
| *ip\_src* | *int* |  | *ip asal serangan* |
| *ip\_dst* | *int* |  | *ip target serangan* |
| *signature\_id* | *int* |  | *id jenis serangan dari event* |
| *signature\_name* | *varchar* |  | *nama serangan dari event* |
| *signature\_priority* | *tinyint* |  | *tingkat ancaman dari event* |
| *sig\_class\_id* | *int* |  | *id klasifikasi serangan dari event* |
| *sig\_class\_name* | *varchar* |  | *nama klasifikasi dari serangan* |
| *protocol* | *int* |  | *protocol yang digunakan dalam serangan* |
| *s\_port* | *int* |  | *port asal serangan* |
| *d\_port* | *int* |  | *port target serangan* |
| *counter* | *int* |  | *sum aggregation dari serangan* |

1. Tabel sensor

Tabel sensor disini akan menyimpan informasi sensor yang terdaftar pada sistem, serta mencatat status service yang ada pada sensor. Tabel ini juga akan dibutuhkan dalam kegiatan ETL untuk mendapatkan id global dari suatu sensor.

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *sid* | *int* |  | *id dari sensor yang terhubung* |
| *hostname* | *varchar* |  | *nama sensor* |
| *interface* | *varchar* |  | *interface yang dimonitor pada sensor* |
| *ip\_address* | *varchar* |  | *ip dari sensor* |
| *location* | *varchar* |  | *lokasi sensor* |
| *company* | *varchar* |  | *perusahaan pemilik sensor* |
| *snort\_service* | *tinyint* |  | *keadaan service snort pada sensor* |
| *barnyard2\_service* | *tinyint* |  | *keadaan service barnyard pada sensor* |
| *mysql\_service* | *tinyint* |  | *keadaan service mysql pada sensor* |
| *push\_status* | *tinyint* |  | *keadaan push ETL dari sensor* |
| *push\_permision* | *tinyint* |  | *perizinan apakah sensor boleh mengirim data* |

1. Tabel sensor\_push\_log

Tabel sensor\_push\_log, berfungsi untuk mencatat event yang masuk dari sensor, proses-proses ETL dan aggregasi sangan bergantung pada tabel ini.

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *id* | *int* |  | *id auto increment* |
| *sid* | *int* |  | *id sensor asal data* |
| *job\_start* | *datetime* |  | *waktu awal dieksekusinya cron ETL* |
| *job\_end* | *datetime* |  | *waktu berakhirnya proses ETL* |
| *range\_start* | *datetime* |  | *batas awal data yang dikirim melalui ETL pada periode tersebut* |
| *range\_end* | *datetime* |  | *batas akhir data yang dikirim melalui ETL pada periode tersebut* |
| *first\_cid* | *int* |  | *unique id pertama dari serangan yang dikirim pada periode tersebut* |
| *last\_cid* | *int* |  | *unique id terakhir dari serangan yang dikirim pada periode tersebut* |
| *package\_name* | *varchar* |  | *nama dari paket ETL ( .zip )* |
| *extract\_status* | *int* |  | *status ekstraksi bernilai ( 0/1/2 )* |
| *aggregation\_status* | *int* |  | *status aggregasi dari event, bernilai ( 0/1/2/3/4/5/6 )* |
| *count\_event* | *int* |  | *jumlah event yang dikirim pada periode tersebut* |

1. Tabel ip\_location

Merupakan tabel yang memuat informasi ip dari seluruh dunia, tabel ini akan digunakan untuk mengisi tabel ip\_map.

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *ip\_start* | *bigint* |  | *batas awal range suatu kelompok ip* |
| *ip\_end* | *bigint* |  | *batas akhir range suatu kelompok ip* |
| *country* | *varchar* |  | *Negara yang terdaftar sebagai pemiliki ip.* |
| *stateprov* | *varchar* |  | *provinsi Negara yang terdaftar sebagai pemiliki ip.* |
| *city* | *varchar* |  | *kota Negara yang terdaftar sebagai pemiliki ip.* |
| *latitude* | *float* |  | *latitude dari suatu kelompok ip.* |
| *longitude* | *float* |  | *longitude dari suatu kelompok ip* |
| *timezone\_offset* | *varchar* |  | *angka timezone lokasi suatu kelompok ip.* |
| *timezone\_name* | *varchar* |  | *nama timezone dari suatu kelompok ip* |
| *isp\_name* | *varchar* |  | *nama isp penyedia suatu kelompok ip.* |
| *connection\_type* | *varchar* |  | *tipe koneksi yang digunakan dari suatu kelompok ip.* |
| *organization\_name* | *varchar* |  | *nama organisasi yang memiliki suatu kelompok ip.* |

1. Tabel ip\_map

Tabel ip\_map ini dibuat untuk memudahkan dalam melakukan query yang membutuhkan informasi dari sebuah ip. Pada schma yang sebelumnya pencarian informasi suatu ip dilakukan langsung ke tabel ip\_location. Pada praktiknya proses join atar tabel kesulitan dalam memanfaatkan index.

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *ip\_addr* | *int* |  | *alamat ip yang pernah menjadi penyerang / target* |
| *country* | *varchar* |  | *Negara yang terdaftar sebagai pemiliki ip.* |
| *stateprov* | *varchar* |  | *provinsi Negara yang terdaftar sebagai pemiliki ip.* |
| *city* | *varchar* |  | *kota Negara yang terdaftar sebagai pemiliki ip.* |
| *latitude* | *float* |  | *latitude dari suatu kelompok ip.* |
| *longitude* | *float* |  | *longitude dari suatu kelompok ip* |
| *timezone\_offset* | *varchar* |  | *angka timezone lokasi suatu kelompok ip.* |
| *timezone\_name* | *varchar* |  | *nama timezone dari suatu kelompok ip* |
| *isp\_name* | *varchar* |  | *nama isp penyedia suatu kelompok ip.* |
| *connection\_type* | *varchar* |  | *tipe koneksi yang digunakan dari suatu kelompok ip.* |
| *organization\_name* | *varchar* |  | *nama organisasi yang memiliki suatu kelompok ip.* |

1. Tabel services

Tabel service merupakan tabel tambahan dari snort yang berisi nama-nama service beserta protocol dan portnya.

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *port* | *int* |  | *port yang terdaftar dari suatu service* |
| *protocol* | *int* |  | *protocol suatu service* |
| *name* | *varchar* |  | *nama service* |
| *description* | *varchar* |  | *deskripsi dari suatu service* |

1. Tabel protocols

Tabel protocol berguna sebagai referance dari kolom protocol pada tabel event, dengan membrikan informasi nama protocol yang digunakan pada suatu event serangan.

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *protocol* | *int* |  | *id suatu protocol pada skema snort* |
| *name* | *varchar* |  | *nama protocol* |
| *description* | *varchar* |  | *deskripsi protocol* |

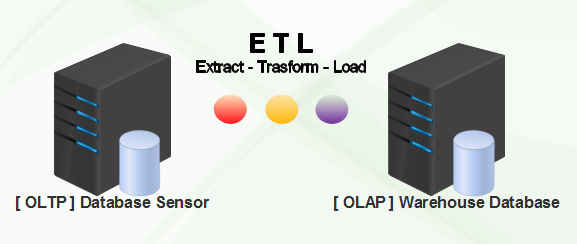
1. Tabel signature\_reference

Tabel signature\_reference berisi link yang menunjukan referensi-referensi yang berkaitan dengan suatu jenis serangan.

|  |  |  |  |
| --- | --- | --- | --- |
| *Field* | *Type* | *Key* | *Desc* |
| *sid* | *int* |  | *id asal data reference* |
| *signature\_id* | *int* |  | *id dari suatu serangan* |
| *signature\_name* | *varchar* |  | *nama serangan* |
| *ref\_seq* | *int* |  | *sequence dari reference, satu serangan bisa memiliki lebih dari satu reference* |
| *ref\_system\_name* | *varchar* |  | *nama reference yang diacu ( CVE, buttraq, dll )* |
| *ref\_tag* | *varchar* |  | *informasi tambahan dari reference, missal id reference pada CVE* |

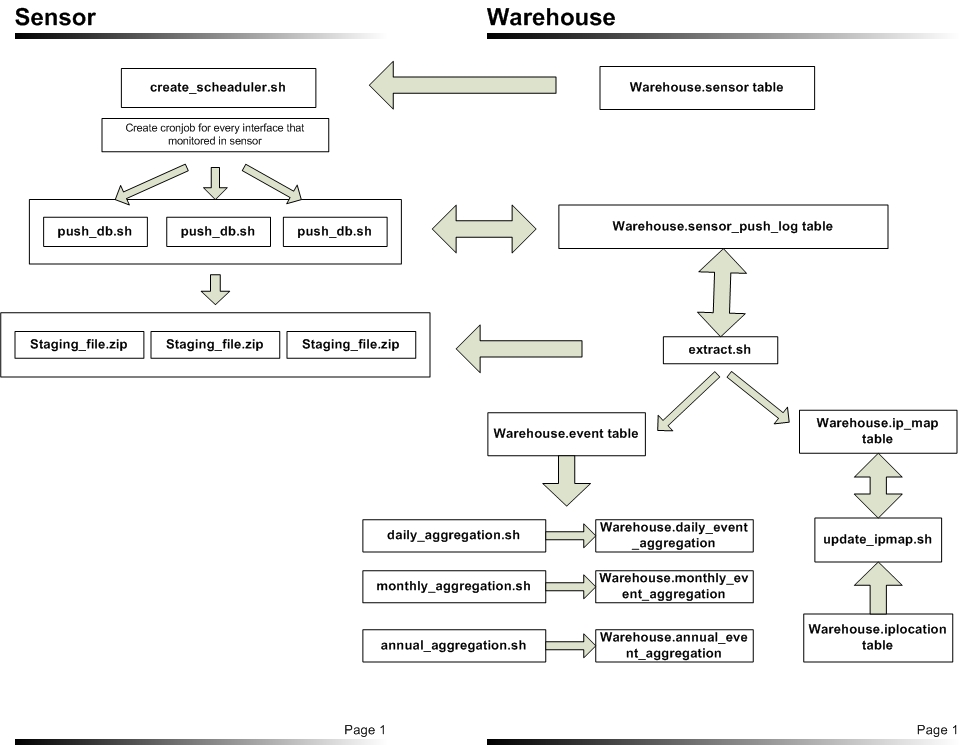
1. Flow ETL ( Extract, Transform, Load )

Pada sub-bab ini akan dijelaskan bagaimana data berinteraksi antara database OLTP yang ada di sensor dengan database warehouse.



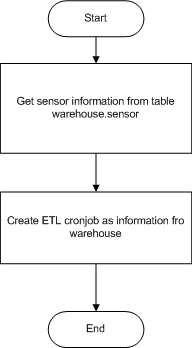
1. ETL ( Extract, Transform, Load )

Pada proyek akhir ini akan dipisahkan database yang lasung menerima input dari sensor dengan database yang akan memproses analisa data. Antara kedua database tersebut akan terjadi proses pertukaran data yang bisa disebut dengan proses ETL. Jadi pada sustu kurun waktu tertensu yang sudah di atur, akan terjadi pengiriman data dari database transaksi ke database warehouse. Dengan skema ini beban dari server database akan dibecah menjadi beberapa database server, sehinnga beban tidak berada pada satu database server. Untuk lebih jelasnya dapat dilihat diagram pada gambar 3.5.



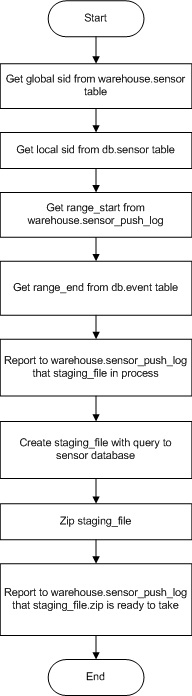
1. Flow ETL
2. Flowchart ETL sensor

Pada proyek akhir ini proses ETL dilakukan secara berkala dengan memanfaatkan cronjob. Sebelum suatu sensor dapat melakukan ETL, identitas sensor harus sudah ditambahkan pada tabel sensor pada sisi warehouse. Selanjutnya dengan script create\_scheaduler.sh akan dibuat job baru pada cron yang dieksekusi berkala sesuai dengan waktu yang ditentukan.



1. Flowchart create\_scheaduler.sh

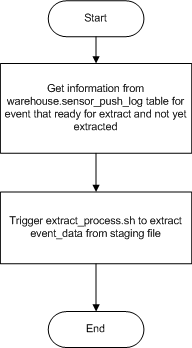
Selanjutnya cron kan menjalankan script push\_mainDB.sh, yang bertugas untuk membuat staging file, untuk lebih jelasnya dapat dilihat pada flowchart pada gambar 3.7



1. Flowchart push\_maindb.sh

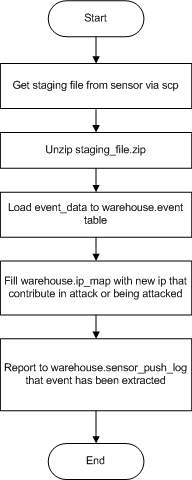
Tugas utama dari script push\_mainDB.sh disini adalah menyiapkan staging file untuk nantinya dikirim ke warehouse, dengan range waktu yang didapat dari tabel sensor\_push\_log dari server warehouse.

1. Flowchart ETL Warehouse



1. Flowchart extract.sh

File script yang ada pada sisi warehouse juga dieksekusi dengan menggunakan cron. Kode pertama yang melakukan load dari staging file ke tabel-tabel yang ada pada database warehouse adalah extract.sh. script extract.sh akan mengambil informasi dari tabel sensor\_push\_log untuk mendapatkan informasi staging file dari sensor mana yang siap untuk dilakukan ETL. Selanjunya akan dilakukan trigger pada script ectract\_process.sh untuk menjalan tugas.

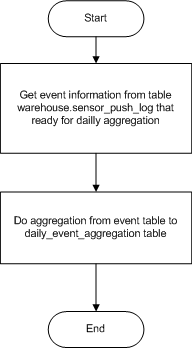


1. Flowchart extract\_process.sh

Pada kode extract\_process.sh akan dilakukan pull dari staging file yang ada pada sensor yang delanjunya akan dilakukan load ke tabe-tabel yang ada pada warehouse. Selain itu pada proses ini, kode juga akan meng-update status dari event pada tabel sensor\_push\_log dengan nilai 1 jika event sedang di load, dan 2 jika event telah selesai di load.

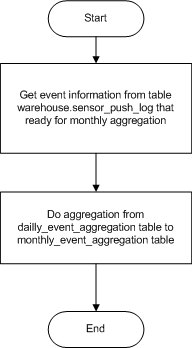
1. Agregasi Event

Pada skema proyek akhir ini, terdapat 3 tabel agregasi yang berfungsi menyimpan rangkuman dari raw data, sehingga pengaksesan report akan lebih cepat, karena tabel yang dibaca akan memiliki row yang lebih kecil.



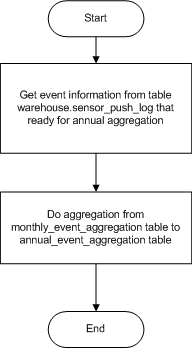
1. Flowchart daily\_aggregation

Pada aggregasi harian, nilai terkecil untuk diolah menjadi report adalah jam, sehingga proses aggregasi harian akan dilakukan ketika suatu event dari sensor sudah genap berdurasi satu jam. Untuk pengecekan durasi suatu event yang berhasil di load, kode daily aggregation akan mengakses tabel sensor\_push\_log, melakukan kalkulasi pada kolom range\_start dan range\_end yang dikonversi dengan nilai jam sebagai nilai terkecil, ketika ditemukan selisih kedua kolom tersebut bernilai lebih dari satu, maka event yang masuk atau berhasil di load sudah genap dalam waktu satu jam, event siap di aggregasi. Pengecekan ini dlakukan permenit, dengan menggunakan cron job. Pada aggregasi harian akan mengupdate status event pada tabel sensor push log dengan nilai 1 dan 2. Angka 1 memiliki arti jika aggregasi sedang di proses, sedangkan 2 menunjukkan proses aggregasi telah usai, event sudah berhasil diaggregasi dan masuk ke tabel daily\_event\_aggregation.



1. Flowchart monthly\_aggregation.sh

Pada aggregasi bulanan, nilai terkecil untuk diolah menjadi report adalah hari, sehingga proses aggregasi bulanan akan dilakukan ketika suatu event dari sensor sudah genap berdurasi satu hari. Untuk pengecekan durasi suatu event yang berhasil di load, kode monthly aggregation akan mengakses tabel sensor\_push\_log, melakukan kalkulasi pada kolom range\_start dan range\_end yang dikonversi dengan nilai hari sebagai nilai terkecil, ketika ditemukan selisih kedua kolom tersebut bernilai lebih dari satu, maka event yang masuk atau berhasil di load sudah genap dalam waktu satu hari, event siap di aggregasi. Pengecekan ini dlakukan permenit, dengan menggunakan cron job. Pada aggregasi harian akan mengupdate status event pada tabel sensor push log dengan nilai 3 dan 4. Angka 3 memiliki arti jika aggregasi sedang di proses, sedangkan 4 menunjukkan proses aggregasi telah usai, event sudah berhasil diaggregasi dan masuk ke tabel daily\_event\_aggregation.



1. Flowchart annual aggregation

Pada aggregasi tahunan, nilai terkecil untuk diolah menjadi report adalah bulan, sehingga proses aggregasi tahunan akan dilakukan ketika suatu event dari sensor sudah genap berdurasi satu bulan. Untuk pengecekan durasi suatu event yang berhasil di load, kode annual aggregation akan mengakses tabel sensor\_push\_log, melakukan kalkulasi pada kolom range\_start dan range\_end yang dikonversi dengan nilai bulan sebagai nilai terkecil, ketika ditemukan selisih kedua kolom tersebut bernilai lebih dari satu maka event yang masuk atau berhasil di load sudah genap dalam waktu satu bulan, event siap di aggregasi. Pengecekan ini dlakukan permenit, dengan menggunakan cron job. Pada aggregasi harian akan mengupdate status event pada tabel sensor push log dengan nilai 5 dan 6. Angka 5 memiliki arti jika aggregasi sedang di proses, sedangkan 6 menunjukkan proses aggregasi telah usai, event sudah berhasil diaggregasi dan masuk ke tabel daily\_event\_aggregation.

*HALAMAN INI SENGAJA DIKOSONGKAN*

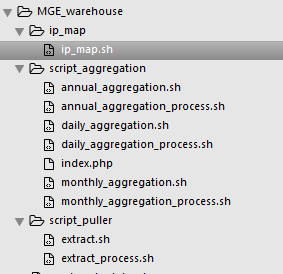
BAB IV

UJICOBA DAN ANALISA

Pada bab ini akan dibahas tentang pengujian dan analisa plugin untuk melihat apakah plugin yang dibuat telah sesuai dengan yang direncanakan sebelumnya. Dan juga untuk mengetahui apakah masih terdapat error pada aplikasi ini. Pengujian dilakukan dengan memberi beberapa tindakan yang berbeda-beda. Dari kegiatan ini diharapkan agar user tidak melakukan kesalahan selama menggunakan system/program ini. Dengan adanya pengujian dapat diketahui adanya kelemahan atau kekurangan yang terjadi sehingga dapat dilakukan perbaikan jika terjadi kesalahan.

1. Uji Coba Sistem OLTP OLAP
2. Instalasi warehouse

Pada sisi warehouse script dibagi menjadi 3 direktori dengan fungsi masing-masing. Folder ip\_map berisi script yang digunakan untuk mengupdate tabel ip\_map. Folder script\_aggregation berisi file script yang menangani agregasi dari harian hingga tahunan. Script puller memiliki tugas utama untuk mengambil dan meng-load data dari sensor ke tabel event da nip\_map.



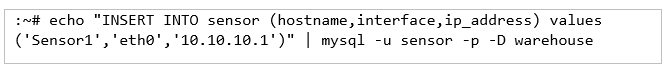
1. Stuktur Script Warehouse

Sebelum masuk kepada pengaturan pada warehouse, pastikan database server MySQL pada warehouse dapat diakses dari sensor. Selain membuat user untuk sensor, kita juga perlu mengubah setting pada server MySQL seperti pada gambar 4.2.



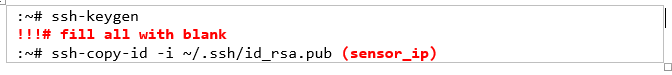
1. Listener MySQL

Tambahkan informasi sensor yang akan melakukan komunikasi dengan warehouse, dengan cara menambahkan record pada tabel sensor seperti dapat dilihat pada gambar 4.3.



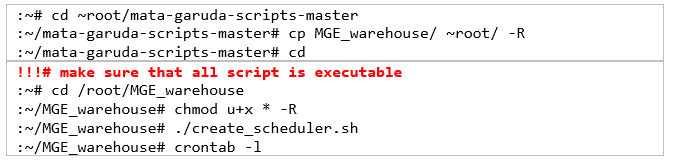
1. Menambahakan Informasi sensor

Setelah menambahkan informasi sensor, langkah selanjutnya adalah memastikan warehouse dapat melakukan konesi ssh ke sensor tanpa menggunakan password.



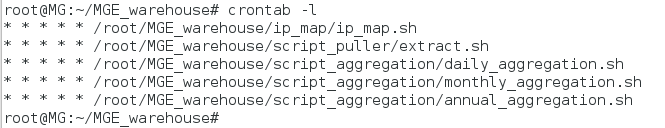
1. SSH no password

Tempatkan file script yang akan digunakan sesuai pengaturan dalam script, pada proyek akhir ini file diletakkan pada direktori /root.



1. File script warehouse

Selanjutnya dapat dieksekusi file create\_scheaduler.sh apda sisi warehouse, untuk membuat cronjob baru. Jika berhasi akan terlihat seperti paa gambar 4.6.

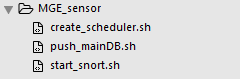


1. Crobjob warehouse

Pada fase ini instalasi pada warehouse sudah selesai, selanjunya kita perlu melakukan setting pada sensor masing-masing.

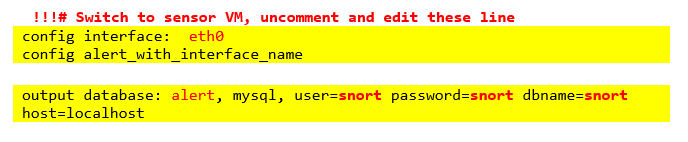
1. Instalasi Sensor

Pada sisi sensor ada 2 buah script, script create scheaduler digunakan untuk menginisiai pembuatan cronjob pada sensor. File push\_mainDB.sh merupadak file utama yang bertugas untuk membuat file staging dan melapor pada data warehouse.

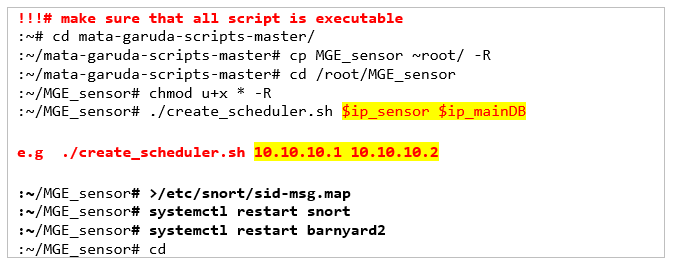


1. Struktur Script Sensor

Sebelum masuk lebih dalam lagi pada instalasi sensor, kita perlu memastikan pengaturan pada file config barnyard2 sudah sesuai.

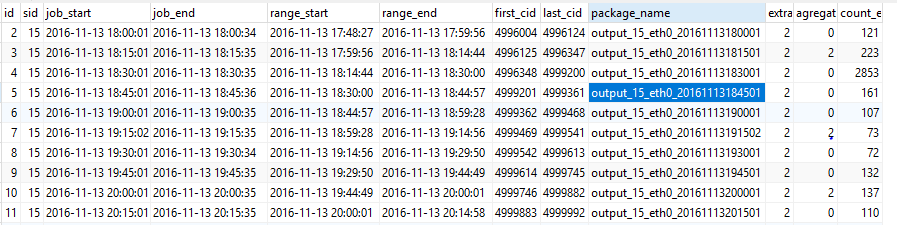


1. Barnyard2.conf



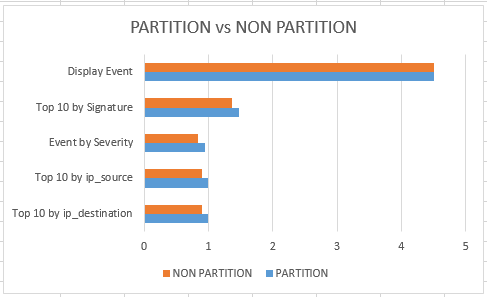
1. Instalasi script pada sensor
2. Uji Coba ETL

Berikut pada gambar 4.10 adalah tampilan log ETL yang terjadi antara sensor dan datawarehouse. Pada gambar 4.10 terlihat nilai ektract\_status adalah 2 yang berarti event berhasil di load, dan nilai aggregasi adalah 2 yang berarti event telah di aggregasi harian.



1. Log ETL
2. Uji Coba Performa Tabel Partition vs Non-Partition
3. Pengujian query tabel partition vs non-partition

|  |  |  |
| --- | --- | --- |
| NAMA QUERY | HASIL QUERY ( sec ) | |
| **PARTITION** | **NON-PARTITION** |
| Display Event | 4.5 sec | 4.5 sec |
| Top 10 by Signature | 1.476 sec | 1.359 sec |
| Event by Severity | 0.953 sec | 0.843 sec |
| Top 10 by ip\_source | 1 sec | 0.9 sec |
| Top 10 by ip\_destination | 1 sec | 0.9 sec |

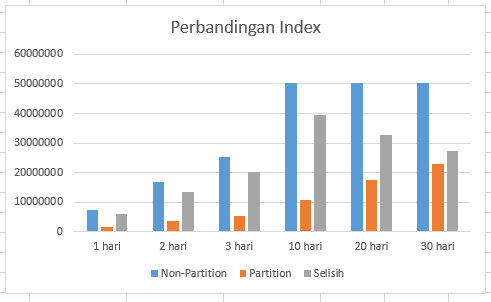


1. Grafik Perbangingan Query Tabel Partisi vs Non-partisi

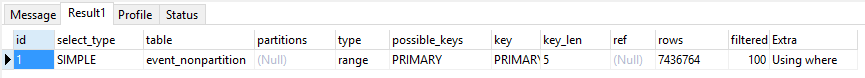
Pada percobaan yang perbandingan query pada tabel berpartisi dan tidak berpartisi terlihat bahwa belum ada perbedaan yang signifikan dari keduanya. Sehingga bisa ditarik kesimpulan, pemartisian horizontal pada proyek akhir ini belum berhasil memberikan efek positif.

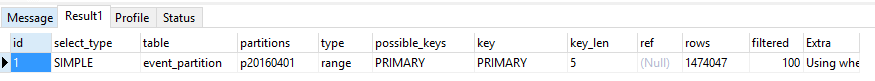
1. Index Partition vs Non-Partition
2. Tabel perbandingan index Partition vs Non-Partition

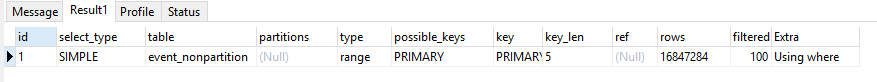
|  |  |  |  |
| --- | --- | --- | --- |
| Hari | Non-Partition | Partition | Selisih |
| 1 hari | 7436764 rows | 1474047 rows | 5962717 rows |
| 2 hari | 16847284 rows | 3501627 rows | 13345657 rows |
| 3 hari | 25270098 rows | 5242254 rows | 20027844 rows |
| 10 hari | 50268935 rows | 10774290 rows | 39494645 rows |
| 20 hari | 50268935 rows | 17519445 rows | 32749490 rows |
| 30 hari | 50268935 rows | 22868579 rows | 27400356 rows |

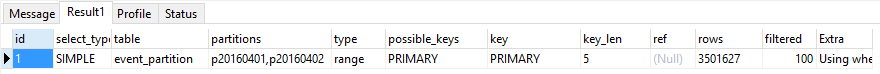


1. Grafik Perbandingan pembaccan index









1. Index Partition vs Non-partition

Dapat dilihat pada tabel 4.2 dan gambar 4.10, banyaknya row yang mungkin dibaca melalui index, pada tabel dengan partisi selalu berisi lebih sedikit row, ini karena pada tabel terpartisi index juga terpartisi. Dengan jumlah row yang dibaca lebih sedikit, seharusnya pencarian akan jauh menjadi lebih cepat.

1. Uji Coba Performa Report

Ujicoba sistem dilakukan dari tahap persiapan environtment, dilanjutkan dengan menjalankan beberapa query report mata garuda pada kedua desain database, serta membandingkan perfoma keduanya dalam bentuk tabel dan grafik.

1. Environtment Pengujian

Pengujian akan dilakukan pada server lokal dalam keadaan tidak ada transaksi yang masuk.

1. Spesifikasi Hardware Pengujian

|  |  |
| --- | --- |
| HARDWARE | |
| CPU | Intel(R) Xeon(R) CPU E5-2680 0 @ 2.70GHz |
| Core | 16 Core |
| Memory | 16 Gb |
| Storage | 8200 RPM |

1. Database Pengujian

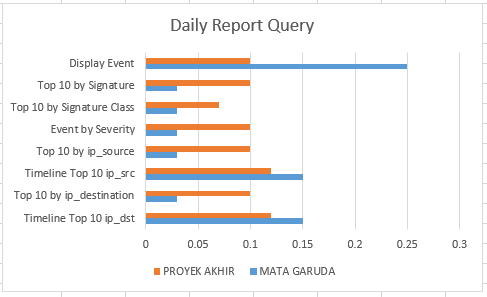
|  |  |
| --- | --- |
| DB Engine Version | |
| Mata Garuda | PostgreSQL 9.1 |
| Proyek Akhir | MySQL 5.7 |

1. Data Pengujian

Data pengujian akan menggunakan data dari data yang telah berhasil di buat saat simulasi, dengan penyesuaian agar didapatkan pengujian yang fair dari skema mata garuda dengan skema yang diajukan pada proyek akhir ini. Sesuai latar belakang yang telah disampaikan sebelumnya, data yang diujikan berjumlah 5.000.000 baris yang didapat langsung dari snort.

1. Pengujian 1 ( 5 juta baris )
2. Daily Report
3. Tabel perbandingan performa query harian

|  |  |  |
| --- | --- | --- |
| NAMA QUERY | HASIL QUERY ( sec ) | |
| **MATA GARUDA** | **PROYEK AKHIR** |
| Display Event | 0.25 sec | **0.1 sec [ ok ]** |
| Top 10 by Signature | **0.03 sec [ ok ]** | 0.1 sec |
| Top 10 by Signature Class | **0.03 sec [ ok ]** | 0.07 sec |
| Event by Severity | **0.03 sec [ ok ]** | 0.1 sec |
| Top 10 by ip\_source | **0.03 sec [ ok ]** | 0.1 sec |
| Timeline Top 10 ip\_src | 0.15 sec | **0.12 sec [ ok ]** |
| Top 10 by ip\_destination | **0.03 sec [ ok ]** | 0.1 sec |
| Timeline Top 10 ip\_dst | 0.15 sec | **0.12 sec [ ok ]** |

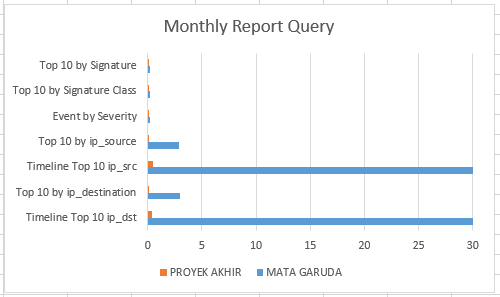


1. Grafik Perbandingan Query Report Harian

Dapat dilihat pada hasil percobaan tabel 4.5, terlihat query harian yang dilakukan pada data menunjukkan skema Mata Garuda lebih unggul. Padahal disini sudah dilakukan aggregasi perhari. Disini juga terlihat pada query display event ayng melibatkan lebih dari 2 tabel memiliki waktu query yang lebih lama, maka dapat ditarik kesimpulan join terlalu banyak dapat berimbas pada melambatnya kecepatan query.

1. Monthly Report

|  |  |  |
| --- | --- | --- |
| NAMA QUERY | HASIL QUERY ( sec ) | |
| **MATA GARUDA** | **PROYEK AKHIR** |
| Top 10 by Signature | 0.22 sec | **0.08 sec [ ok ]** |
| Top 10 by Signature Class | 0.2 sec | **0.08 sec [ ok ]** |
| Event by Severity | 0.2 sec | **0.09 sec [ ok ]** |
| Top 10 by ip\_source | 2.9 sec | **0.08 sec [ ok ]** |
| Timeline Top 10 ip\_src | 191 sec | **0.5 sec [ ok ]** |
| Top 10 by ip\_destination | 3 sec | **0.08 sec [ ok ]** |
| Timeline Top 10 ip\_dst | 120 sec | **0.4 sec [ ok ]** |

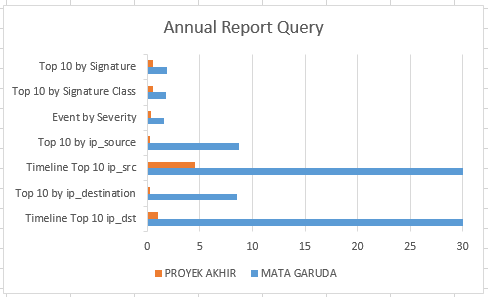


1. Grafik Perbandingan Query Report Bulanan

Pada percobaan query untuk jangka waktu bulanan, terlihat bahwa query yang telah dilakukan pada skema proyek akhir lebih cepat dibandingkan query yang dilakukan pada skema mata garuda. Analisa yang didapat adalah, dengan aggregasi bulanan jumlah row akan lebih menyusut lagi, sehingga berdampak baik pada prosek query data.

1. Annual Report

|  |  |  |
| --- | --- | --- |
| NAMA QUERY | HASIL QUERY ( sec ) | |
| **MATA GARUDA** | **PROYEK AKHIR** |
| Top 10 by Signature | 1.9 sec | **0.5 sec [ ok ]** |
| Top 10 by Signature Class | 1.8 sec | **0.5 sec [ ok ]** |
| Event by Severity | 1.6 sec | **0.4 sec [ ok ]** |
| Top 10 by ip\_source | 8.7 sec | **0.3 sec [ ok ]** |
| Timeline Top 10 ip\_src | 700+ sec | **4.5 sec [ ok ]** |
| Top 10 by ip\_destination | 8.5 sec | **0.3 sec [ ok ]** |
| Timeline Top 10 ip\_dst | 700+ sec | **1 sec [ ok ]** |



1. Grafik Perbandingan Query Report Tahunan

Pada query tahunan terlihat perbedaan performa yang signifikan, kecepatan query pada skema mata garuda terpaut jauh dari query yang dilakukan pada proyek akhir ini. Sehingga dapat diambil kesimpulan bahwa optimasi dengan menerapkan tabel aggregasi disini berhasil meningkatkan performa query.

BAB V

PENUTUP

Pada bab ini akan diuraikan rangkuman pekerjaan dan kesimpulan dari keseluruhan sistem database yang diajukan setelah dilakukan pengujian dan analisa mulai dari bab satu hingga bab empat. Pada bab ini terdapat juga saran-saran mendukung dalam upaya menyempurnakan penelitian lain yang menggunakan tugas akhir ini sebagai refrensi.

1. OPTIMASI DESAIN DATABASE

Pada sub bab ini akan di jelaskan rangkuman pengerjaan optimasi desain database yang telah dilakukan. Berikut dapat dirangkum optimasi-optimasi yang telah dilakukan serta pengaruhnya terhadap kemampuan penyajian data.

* Desain OLTP dan OLAP

Dengan memisahkan database yang berinteraksi langsung dengan sensor snort dan database yang digunakan untuk analisa, berimbas pada menurunnya beban server karena input dari banyak sensor tidak langsung ditangani oleh satu database.

* Horizontal Partition

Pemartisian horizontal yang diterapkan disini belum memberikan efek yang signifikan. Keuntungan yang didapat disini terlihat pada index, pada tabel yang dipartisi index juga terpartisi, sehingga index tidak akan terlalu besar untuk dibaca.

* Denormalization,

Tujuan denormalisasi adalah untuk mempercepat proses query select, dengan mengorbankan desain normal sehingga didapati redudansi data. Pada proyek akhir ini, data yang sebelumnya diquery melalui join dari beberapa tabel, dapat dilakukan melalui satu tabel yang disesuaikan dengan analisa dan report yang ada pada aplikasi Mata Garuda saat ini.

* Aggregation,

Aggregasi sangat membantu meningkatkan performa pembuatan report, terlihat dari perbandingan query yang dilakukan pada bab IV.

1. KESIMPULAN

Berdasarkan hasil analisa dari beberapa pengujian yang diterangkan pada bab sebelumnya, maka dapat diambil kesimpulan sebagai berikut:

* Partisi tabel yang diterapkan pada proyek akhir ini belum menunjukan kontribusi yang menonjol pada pengoptimalan kecepatan query. Keunggulan terlihat pada row yang dibaca melalui index yang lebih sedikit.
* Pemisahan database transaksi dan database analisa, memberikan efek positif pada beban server yang berkurang.
* Denormalisasi yang dilakukan, dan mengubah vertical partition dari tabel snort menjadi satu tabel yang berimas berkurangnya join, sangat mempermudah pembuatan query dan mempercepat performa query.
* Penerapan aggregasi harian, bulanan, dan tahunan berhasil meningkatkan performa query dikarenakan row yang menyusut.

1. SARAN

Guna menyempurnakan penelitian lain yang menggunakan tugas akhir ini sebagai refrensi, berikut ialah saran yang dapat penulis berikan:

* Untuk penelitian selanjunya, mungkin dapat dilakukan optimasi lagi pada struktur datawarehouse yang lebih rapi dengan menerapkan star schema atau snowflake.

DAFTAR PUSTAKA

|  |  |
| --- | --- |
| [1] | G. W. Italiano, F. A. Saputra dan I. Nadhori, Rancang Bangun Aplikasi Mata Garuda sebagai Contoh Implementasi Modern IDS. |
| [2] | P. Zaitsev, “Percona Database Performance Blog,” Percona, 9 June 2006. [Online]. Available: https://www.percona.com/blog/2006/06/09/why-mysql-could-be-slow-with-large-tabels/. [Diakses 1 12 2016]. |
| [3] | H. E. Blok, DATABASE OPTIMIZATION ASPECTS FOR INFORMATION RETRIEVAL, Enschede: Twente University Press, 2002. |
| [4] | H. Boral dan D. J. DeWitt, “A Methodology for Database System Performance Evaluation,” University of Wisconsin, Israel Institute of Technology. |
| [5] | A. D. Andurkar, “IMPLEMENTATION OF COLUMN-ORIENTED DATABASE IN POSTGRESQL FOR OPTIMIZATION OF READ-ONLY QUERIES,” *CS & IT-CSCP,* 2012. |
| [6] | Y. D. Anto, J. Wibowo dan T. Heru, “PENERAPAN BASIS DATA TERPARTISI DALAM RANCANG BANGUN SISTEM PENGGAJIANPENGUPAHAN DENGAN MODEL JARINGAN CLIENT-SERVER,” *SNASTI,* 2011. |
| [7] | Oracle Corporation and/or its affiliates, “MySQL Documentation,” 2017. [Online]. Available: http://dev.mysql.com/doc/refman/5.7. [Diakses 18 January 2017]. |
| [8] | M. Hollingsworth, “Data Normalization, Denormalization,and the Forces of Darkness”. |
| [9] | datawarehouse4u, “datawarehouse4u,” 2009. [Online]. Available: http://datawarehouse4u.info/OLTP-vs-OLAP.html. [Diakses 10 January 2017]. |
| [10] | F. Aldiansah, “Mari Berbagi Ilmu,” January 2013. [Online]. Available: http://fajriatusaldiansah.blogspot.co.id/2013/01/data-warehouse-data-mart-data-mining.html. [Diakses 11 January 2017]. |
| [11] | PostgreSQL Global Development Group, “PostgreSQL Documentation,” PostgreSQL Global Development Group, 2016. [Online]. Available: https://www.postgresql.org/docs/9.5/static/ddl-partitioning.html. [Diakses 11 1 2017]. |
| [12] | D. Dharmayanti, A. M. Bachtiar dan A. Heryandi, “PEMODELAN DATA WAREHOUSE PADA JURUSAN TEKNIK INFORMATIKA UNIKOM,” *Majalah Ilmiah Unikom,* vol. 12. |

**BIODATA PENULIS**



Nama : Syandu Irdoni

T.T.L : Kediri, 11 April 1994

Alamat : Jl. Slamet Riadi no 57 Kediri

No Hp : 083 846 244 160

Email : syanduirdoni@gmail.com

Riwayat Pendidikan :

* PENS ( D3 ) : Tahun 2012 - 201
* SMKN 1 Kediri : Tahun 2009 - 2012