Jurnal Sains Dan Teknologi Informasi
Vol 3, No 2, Maret 2024, Hal 68-78
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v3i2.4799
Website https://ejurnal.seminar-id.com/index.php/jussi

Penerapan Algoritma Additive Code Dalam Mengkompresi Record Database

Fathul Fauzan

Program Studi Teknik Informatika, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Budi Darma,
Jalan Sisingamanganraja No. 338, Medan, Sumatera Utara, Indonesia
Email: fathulfauzan420@gmail.com

Abstrak-Permasalahan kebutuhan ruang penyimpanan pada aplikasi kompresi*record database* merupakan salah satu masalah utama yang dapat mempengaruhi kinerja dari aplikasi kompresi*record database* sendiri. Hal ini disebabkan karena banyaknya *record* data yang tersimpan di dalam *database*.Salah satu solusi yang dapat dilakukan untuk mengatasi permasalahan tersebut adalah melakukan kompresi *file database*. Proses kompresi dalam penelitian ini dilakukan berdasarkan algoritma *Additive Code* dengan membangun aplikasi kompresi*record database* berbasis *dekstop*. Proses awal yang dilakukan untuk mengimplementasikan metode ini siapkan *file* SQL yang akan dikompresi, proses kompresi berlangsung pada saat *file* SQL selesai diupload kemudian menekan tombol kompresi padaaplikasi kompresi*record database* dan proses dekompresi secara berlangsung pada saat mengunduh *file* SQL yang pernah dikompresi dan menekan tombol dekompresi padaaplikasi kompresi*record database*. Berdasarkan hasil pengujian yang dilakukan diperoleh *Ratio of Compression* (RC) sebesar 1,33, *Compression Ratio* (Cr) berkurang sebanyak 75%, *Redudancy* (Rd)berkurang sebanyak 25% dan *Space Saving* (Ss) berkurang sebanyak 25%. Hasil ini menunjukkan bahwa kebutuhan ruang penyimpanan *database* aplikasi kompresi*record daatabase* lebih optimal daripada sebelum diterapkan proses kompresi.

Kata Kunci: Kompresi; File SQL; Aplikasi Kompresi Record Database; Algoritma Additive Code Abstract-The problem of storage

Abstract-The problem of storage space requirements in compressed record database applications is one of the main problems that can affect the performance of the compressed record database application itself. This is due to the large number of data records stored in the database. One solution that can be done to overcome this problem is to compress the database file. The compression process in this study was carried out based on the Additive Code algorithm by building a desktop-based database record compression application. The initial process carried out to implement this method is to prepare the SQL file to be compressed, the compression process takes place when the SQL file has finished uploading, then pressing the compression button on the database record compression application and the decompression process takes place when downloading the compressed SQL file and pressing the decompress button. on database record compression applications. Based on the results of the tests carried out, it was obtained that the Ratio of Compression (RC) was 1.33, the Compression Ratio (Cr) was reduced by 75%, Redundancy (Rd) was reduced by 25% and Space Saving (Ss) was reduced by 25%. These results indicate that the storage space requirements for database compression applications record databases more optimally than before the compression process was applied.

Keywords: Compression; SQL File; Database Record Compression Application; Additive Code Algorithm

1. PENDAHULUAN

Database merupakan kumpulan data atau informasi yang tersimpan secara sistematis. Dibutuhkan suatu tempat penyimpanan untuk mengumpulkannya agar data tersebut tidak hilang dan memudahkan saat mencarinya. Masalahnya database membutuhkan tempat penyimpanan yang besar apabila data tersebut terus bertambah seiring waktu dalam pengumpulan data tersebut. Apabila jumlah penyimpanan data semakin besar maka kinerja dalam pengolahan dan pengumpulan data menjadi terganggu dan lambat karena system juga membutuhkan ruang penyimpanan dalam pengoperasiannya. Dalam instansi sekolah khususnya SMK Negeri 1 Patumbak yang sekarang melakukan kegiatan ujian dan pendataan secara online juga pasti lama kelamaan juga akan membutuhkan penyimpanan yang besar dalam menyimpan database di server karena data yang terus bertambah .Sehingga disaat ruang penyimpanan semakin sedikit harus dilakukan penambahan ruang penyimpanan baru agar system tetap berjalan baik. Solusinya agar dapat meminimalisir jumlah penyimpanan yang besar dibutuhkan aplikasi kompresi record database guna mendapatkan ukuran data terbaik dalam mengelola kebutuhan penyimpanan database pada server di instansi sekolah tersebut.

Pada aplikasi kompresi record database yang digunakan file yang di upload yaitu berupa file dokumen yang berekstensi (*.sql). SQL merupakan singkatan dari Structured Query Language dan file dengan ekstensi ini dikembangkan oleh Oracle Corp. File SQL memiliki ukuran yang cukup besar, sehingga membutuhkan waktu lama dalam proses pengiriman dan menghabiskan banyak ruang. Melalui aplikasi tersebut yang dalam proses kompresi diharapkan data tetap dalam kondisi yang bagus setelah di kompresi. Perkiraan hasil maksimal dari kompresi data tersebut dapat mencapai di atas 30% setelah dilakukan proses kompresi.Kompresi data atau data compression merupakan sebuah teknik dalam ilmu computer untuk mengompresi data sehingga hanya membutuhkan lebih sedikit ruang penyimpanan dan meningkatkan efisiensi penyimpanan untuk mengurangi waktu pertukaran data, terdapat dua jenis kompresi data, yaitu kompresi data lossy adalah teknik kompresi di mana terdapat data yang hilang selama proses kompresi sehingga kualitas data yang dihasilkan jauh lebih rendah dari kualitas data aslinya,sedangan kompresi data lossless adalah kompresi yang tidak menghapus data selama proses kompresi dan tidak mengurangi kualitas kompresi data yang dihasilkannya[1]. Salah satu algoritma kompresi antara lain adalah algoritma additive code.Algoritma additive code adalah prosedur pemecahan yg meliputi beberapa algoritma kompresi yaitu algoritma golbach code, algoritma fibonacci code & algoritma elias gamma code yang dimana memungkinkan lebih efisien pada mengkompresi sebuah data, dalam waktu ini penelitian mengenai proses kompresi database memakai algoritma additive code masih

Jurnal Sains Dan Teknologi Informasi
Vol 3, No 2, Maret 2024, Hal 68-78
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v3i2.4799
Website https://ejurnal.seminar-id.com/index.php/jussi

sedikit padahal algoritma additive code adalah salah satu teknik kompresi yang berfungsi buat memperkecil suatu data menurut karakter dari objek yg akan dikompresi.

Berdasarkan penelitian yang dilaksanakan oleh Rizka Dwi Pratiwi, Surya Darma Nasution dan Fadlina pada tahun 2018, didapatkan kesimpulan bahwa pada dasarnya algoritma fixed length binary encoding (FLBE) menggantikan setiap karakter dengan kode biner alternatif yang panjangnya ditentukan berdasarkan berapa kali karakter tersebut ditampilkan. Karakter yang sering muncul untuk memiliki kode biner yang lebih sedikit dari pada karakter yang lebih jarang muncul. Algoritma fixed length binary encoding (FLBE) ini dapat menghentikan proses kompresi ketika panjang bit karakter mencapai jumlah karakter terakhir dan dapat menangani proses kompresi file database dengan baik[2].
Berdasarkan penelitian yang dilakukan oleh Syawaluddin Nainggolan pada tahun 2019 didapatkan kesimpulan bahwa algoritma goldbach codes bekerja untuk mengompresi file teks adalah dengan mengubah bilangan bulat positif n menjadi bilangan bulat positif genap 2(n+3) dan kemudian menulis sepasang bilangan prima secara terbalik. Algoritma dynamic markov compression merupakan teknik kompresi adaptif karena struktur finite-state engine berubah seiring dengan pemrosesan file. Dalam hal kompresi, algoritma goldbbach code lebih unggul dari pada algoritma dynamic markov compression[3].

Berdasarkan penelitian yang dilaksananakan oleh Mawar di tahun 2020, diperoleh kesimpulan bahwa setelah merancang dan mengimplementasikan perangkat lunak kompresi pada file PDF menggunakan algoritma punctured elias code maka hasil yang didapat pada perhitungan kompresi dan dekompresi file PDF yaitu dengan menginput file PDF. Menerapkan metode algoritma Punctured elias code pada kompresi file PDF dimulai dengan mencari nilai biner file PDF dari aplikasi Binary Viewer, setelah nilai biner didapatkan, biner dikompilasi untuk mencari frekuensi, setelah nilai frekuensi disusun, kemudian dilakukan kompresi[4]. Berdasarkan penelitian yang dilaksanakan oleh Bobby Ramadhana pada tahun 2021, didapatkan kesimpulan bahwa pada proses kompresi file PDF berhasil dilakukan dengan menerapkan dua algoritma yaitu fibonacci code dan levenstein code. File PDF terkompresi memiliki perubahan ukuran yang signifikan yaitu rata-rata penghematan ruang yang didapat lebih dari 50% dan waktu yang dibutuhkan untuk proses kompresi file PDF berbeda-beda karena dipengaruhi oleh panjang karakter file yang diinput[5].Berdasarkan penelitian yang dilakukan oleh Apijuddin pada tahun 2021, didapatkan kesimpulan bahwa prosedur kompresi file teks dimulai dari pemilihan file teks yang mau dikompresi, kemudian file tersebut dilakukan kompresi menggunakan algoritma stout codes sehingga menghasilkan file kompresi yang memiliki ukuran lebih kecil. Menerapkan algoritma stout code untuk kompresi file teks dapat dilakukan dengan cara membaca nilai heksadesimal dari sebuah file teks kemudian ubah nilai heksadesimal kenilai bit baru kedalam format biner, selanjutnya menyusun kembali nilai biner tersebut menjadi karakter yang baru[6].

Sesuai dengan latar belakang masalah yang diuraikan di atas maka dilakukan uraian tahapan dalam mengkompresi sebuah record database dan dituangkan pada sebuah penelitian menggunakan judul “Penerapan Algoritma Additive Code Dalam Mengkompresi Record Database“.

2. METODOLOGI PENELITIAN

2.1 Kompresi

Kompresi data atau *data compression* merupakan sebuah teknik dalam ilmu komputer untuk mengompresi data sehingga membutuhkan lebih sedikit ruang penyimpanan, membuat proses penyimpanan lebih efisien, dan mengurangi waktu pertukaran data. Kompresi data dimaksudkan untuk mengurangi jumlah bit atau mengurangi data dengan ukuran besar (*Redundancy* Data) menjadi ukuran data yang lebih kecil dan lebih ringan sehingga lebih mudah digunakan dalam proses penyimpanan dan mengirim sebuah informasi. Proses kompresi dapat dilakukan terhadap sebuah teks/biner, dokumen (docx, pdf, xlsx, pptx), gambar (JPEG, PNG, TIFF), audio (MP3, ACC, RMA, WMA) dan video (MPEG, H261, H263) [2]. Jenis data yang dapat dilakukan untuk kompresi data terdiri dari *Lossy Compression dan Lossless Compression* [3].

2.2 Dekompresi

Dekompresi data adalah data yang telah dilakukan kompresi dan tentu saja perlu dipulihkan lagi ke bentuk semula. Dekompresi data membantu menghilangkan kerumitan yang disebabkan oleh kompresi data. Seperti halnya kompresi, tidak jarang terjadi kesalahan transmisi saat mendekompresi data, sehingga dekompresi dapat berjalan lambat dan dapat memakan waktu. Aplikasi yang diperlukan untuk dekompresi tergantung pada metode kompresi data dan berbagai teknik dan algoritma yang tersedia untuk mendukung proses dekompresi. Dalam kebanyakan kasus, perangkat lunak atau aplikasi yang diperlukan untuk mendekompresi data juga dilengkapi dengan aplikasi atau perangkat lunak yang digunakan untuk mengompresi data[7].

2.2 Structured Query Language (SQL)

Structured Query Language (SQL) merupakan bahasa standar terdiri dari kumpulan instruksi, digunakan untuk berinteraksi dengan relational database. SQL memiliki tiga komponen utama yaitu Database Definition Language (DDL), Database Manipulation Language (DML), Data Control Language (DCL). SQL terdiri dari kumpulan pernyataan dimana semua program dan pengguna dapat mengakses data melalui basis data.

Jurnal Sains Dan Teknologi Informasi
Vol 3, No 2, Maret 2024, Hal 68-78
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v3i2.4799
Website https://ejurnal.seminar-id.com/index.php/jussi

2.3 Record Database

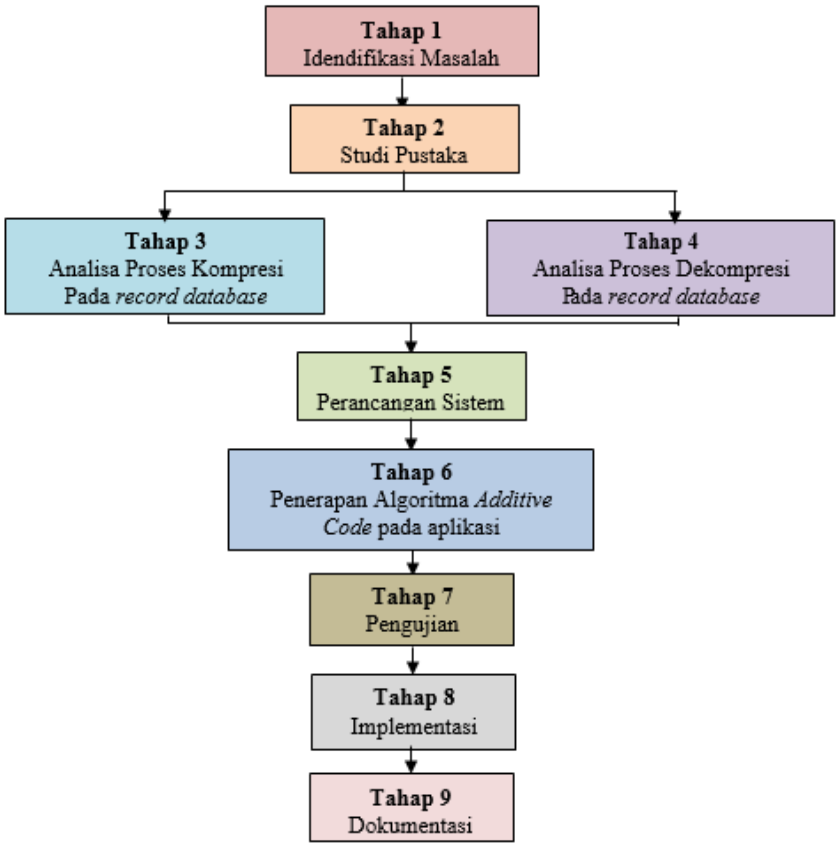
Record database adalah kumpulan *field*/karakter yang lengkap dihitung dalam satuan baris pada tabel di dalam sebuah basis data. *Record* dianalogikansebagai baris dan kolom membentuk kumpulan *field* yang saling terhubung. *Record* diibaratkan seperti *array*, dimana dapat membuat sebuah variabel yang berisi berbagai element. Perbedaannya ialah *record* mampu menampung berbagai jenis tipe data, sedangkan *array* hanya dapat menampung satu tipe data. *Database* atau dikenal dengan basis data merupakan sekumpulan data berupa *file*, tabel atau arsip yang saling berhubungan tersimpan secara bersamaan di dalam media penyimpanan elektronik. Data adalah sebuah informasi fakta berupa angka, huruf dan lainnya, sedangkan basis adalah kelompok[8].

2.4 Algoritma*AdditiveCode*

Algoritma *additive code* adalah algoritma yang memakai beberapa algoritma kompresi yaitu *golbach code*, *fibonacci code* dan *elias gamma code* yang dapat lebih efisien dalam mengkompresi sebuah *file*. Algoritma *additive code* adalah teknik kompresi yang digunakan untuk memperkecil suatu data berdasarkan karakter pada objek yang akan dilakukan proses kompresi dan mengarah pada kode yang sederhana dan efisien menunjukkan bahwa barisan lain dapat digunakan dengan cara yang sama[9].

2.5 Tahapan Penelitian

Pada Tahapan penelitian yang terdapat di dalam sebuah metodologi penelitian dibawah ini dijelaskan yang disebut dengan kerangka penelitian atau tahapan penelitian. Kerangka kerja penelitian terdiri dari beberapa tingkatan yang secara sistematis saling terkait. Kerangka kerja ini diperlukan untuk mempermudah pencarian. Berikut pada gambar 1 adalah kerangka kerja penelitian:



Gambar 1. Tahapan Penelitian

Dari diagram tahapan penelitian di atas, maka uraian masing-masing dari tahapan penelitian dapat dijelaskan sebagai berikut:

a. Tahapan Indektifikasi Masalah
   Pada tahapan ini dapat memprediksi, memperkirakan, dan menjelaskan penyebab masalah kapasitas memori penuh dan mentransfer file yang berekstensi (*.sql) secara perlahan hingga tidak dapat lagi tertampung di ruang memori.
b. Tahapan Studi Pustaka
   Pada tahapan ini memahami sebuah objek yang diteliti dengan membaca berbagai referensi seperti jurnal, buku, majalah, dan bahan bacaan lainnya.
c. Tahapan Analisa Proses Kompresi Pada File SQL

Jurnal Sains Dan Teknologi Informasi
Vol 3, No 2, Maret 2024, Hal 68-78
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v3i2.4799
Website https://ejurnal.seminar-id.com/index.php/jussi

Pada tahapan ini dijabarkan tentang algoritma additive code yaitu adalah teknik kompresi yang dapat memperkecil sebuah data file berdasarkan dengan karakter pada objek yang akan dikompresi sehingga menghasilkan kode yang sederhana dan efisien menunjukkan bahwa urutan lainnya dapat digunakan dengan cara yang sama.

d. Tahapan Analisa Proses Dekompresi Pada File SQL
Pada tahapan analisa proses pengembalian file SQL yang telah dikompresi menjadi bentuk file semula sebelum dikompresi menggunakan algoritma additive code.

e. Tahapan Perancangan Sistem
Pada tahapan ini memberikan gambaran tentang rancangan aplikasi yang diusulkan untuk kompresi file SQL. Tahap perencanaan ini menggunakan data yang dianalisis dalam format yang sederhana, mudah, dan mudah dipahami oleh pengguna.

f. Tahapan Penerapan Algoritma Additive Code pada program aplikasi kompresi
Pada Tahapan Penerapan ini yang dilakukan selama proses pengembangan / pengkodean perangkat lunak yang menentukan apakah sistem berjalan dengan baik dan memenuhi persyaratan, atau jika sistem masih perlu dilakukan perbaikan.

g. Tahapan Pengujian
Pada tahapan ini melakukan pengujian hasil kompresi file yang berekstensi (*.sql) dan pada proses ini bertujuan untuk mengetahui hasil akhir dari penelitian yang dilakukan oleh penulis.

h. Tahapan Implementasi
Pada Tahapan Implementasi ini yang dilakukan setelah pengujian perangkat lunak yang menentukan apakah sistem berjalan dengan baik dan memenuhi persyaratan untuk implementasi.

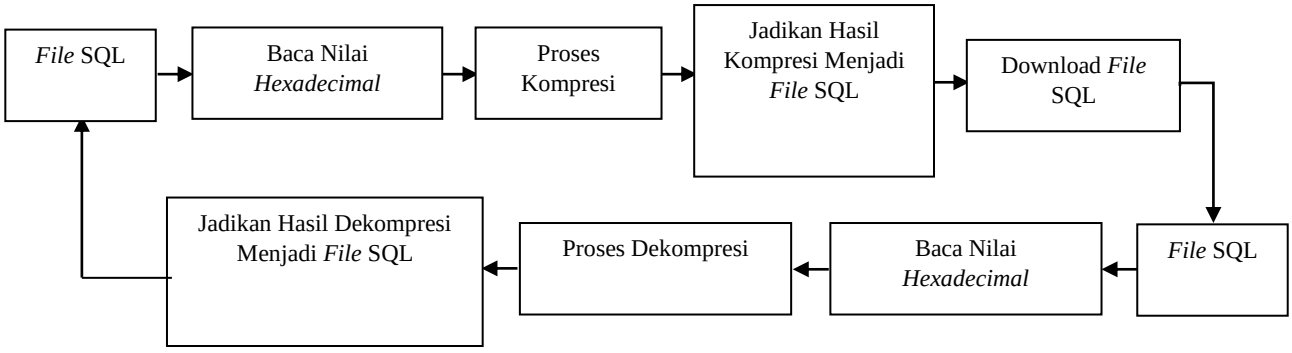
i. Tahapan Dokumentasi
Pada tahap dokumentasi ini adalah tahapan terakhir dari pelaksanaan penelitian yang dilakukan penulis dan dibuat dalam bentuk laporan. Tahap ini menjelaskan aplikasi yang dibuat untuk memudahkan pembaca yang ingin mengembangkan lebih banyak aplikasi.

3. HASIL DAN PEMBAHASAN

3.1 Analisa

Analisa penelitian ini merupakan perhitungan dan perancangan aplikasi kompresi *record database* yang di dalam aplikasi tersebut terdapat proses kompresi dan dekompresi terhadap*file* SQL menggunakan algoritma *Additive Code*. Algoritma *Additive Code* merupakan salah satu teknik kompresi *lossless* yang dapat memperkecil ukuran kapasitas data berdasarkan karakter pada objek yang dikompresi. Penelitian ini dibuat untuk mengetahui proses kompresi *file* SQL padaaplikasi kompresi *record database* yangmenerapkan Algoritma *Additive Code*, *file* SQL yang biasanya memiliki ukuran data yang relatif besar. Oleh karena itu, hasil penelitian ini diharapkan dapat membantu memperkecil ukuran *file* SQL.

Proses awal yang dilakukan untuk mengimplementasikan metode ini adalah siapkan *file* SQL yang dikompresi kemudian mengkonversi *file* SQL ke nilai *hexadecimal* menggunakan aplikasi *binary viewer*, setelah didapat nilai *hexadecimal file* SQL maka dilakukan proses kompresi. Hasil kompresi tersebut kembali menjadi *file* SQL, kemudian dapat didownload untuk disimpan. Dekompresi dilakukan setelah *file* SQL dikompresi dan tentu saja perlu dipulihkan lagi ke bentuk semula.Penelitian ini membahas 2 proses utama dalam pengkompresian *file* SQL pada aplikasi kompresi *record database* yaitu proses kompresi dan proses dekompresi.

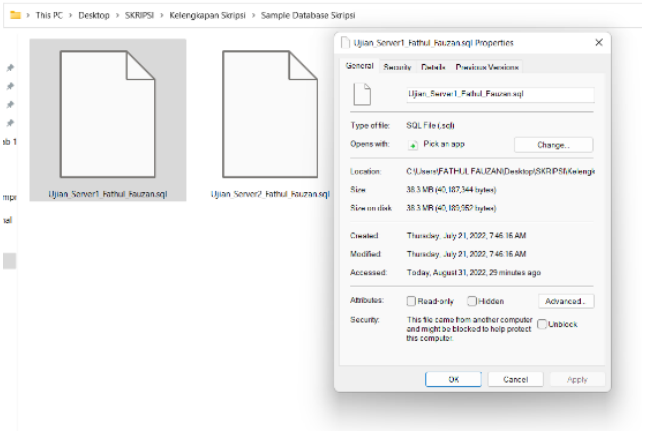


Gambar 2. Prosedur Kompresi dan Dekompresi File SQL

3.2 Proses Kompresi

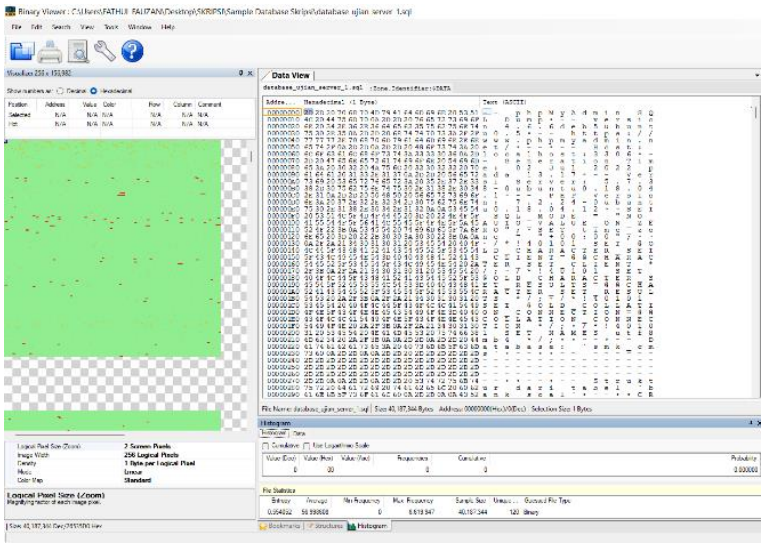
Sebelum dilakukan proses kompresi, terlebih dahulu siapkan *file* SQL yang akan dikompresi dengan menerapkan algoritma *Additive Code*. Berikut adalah contoh *file* SQL yang dilakukan proses kompresi dan dekompresi:

Jurnal Sains Dan Teknologi Informasi
Vol 3, No 2, Maret 2024, Hal 68-78
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v3i2.4799
Website https://ejurnal.seminar-id.com/index.php/jussi



Gambar 3. Sampel *file* SQL

Sebelum *file* yang dikompresi terlebih dahulu dilakukan pembacaanbiner yang terdapat pada *file* SQL untuk mendapatkan data berupa nilai *hexadecimal*.Berdasarkan gambar di bawah terdapat nilai *hexadecimal file* SQLyang diperoleh menggunakanaplikasi *bineryviewer*, hasil nilai *hexadecimal* ditampilkanpadagambarberikut:



Gambar 4. Nilai *Hexadecimal* Sampel *File* SQL

Berdasarkan gambar nilai *hexadecimal* di atas, mengambil 16 nilai *hexadecimal* untuk proses pengkompresian *file* SQL.

Tabel 1. Tampilan Nilai Bilangan *Hexadecimal*

25	50	44	46	2D	31	2E	35
0D	0A	25	B5	B5	B5	B5	0D

Dari tabel di atas, maka dihasilkan bilangan *hexadecimal file* SQL yang dihitung secara manual. Berikut nilai *hexadecimal* diurutkan berdasarkan frekuensinya, nilai frekuensi paling banyak akan berada diurutan pertama.

Tabel 2. Data Sebelum Dikompresi Menggunakan Algoritma *Additive Code*

n	Hexadecimal	Biner	Bit	Frekuensi	Bit x Frekuensi
1	B5	1011 0101	8	4	32
2	25	0010 0101	8	2	16
3	0D	0000 1101	8	2	16
4	50	0101 0000	8	1	8
5	44	0100 0100	8	1	8
6	46	0100 0110	8	1	8
7	2D	0010 1101	8	1	8
8	31	0011 0001	8	1	8
9	2E	0010 1110	8	1	8
10	35	0011 0101	8	1	8
11	0A	0000 1010	8	1	8

Jurnal Sains Dan Teknologi Informasi
Vol 3, No 2, Maret 2024, Hal 68-78
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v3i2.4799
Website https://ejurnal.seminar-id.com/index.php/jussi

Total	128 bit

Setelah bilangan *hexadecimal* diurutkan berasarkan frekuensi kemunculan dan mendapat nilai biner, maka selanjutnya menghitung bit dan pengurutan seluruh nilai biner dari bilangan *heksadesimal* diganti menjadi tabel kebenaran dengan menerapkan algoritma *Additive Code* serta memperoleh bit *file* terkompresi. Adapun proses kompresi *file* SQL dapat dilihat pada tabel 3 di bawah ini.

Tabel 3. Data Setelah Dikompresi Menggunakan Algoritma *Additive Code*

N	Hexadecimal	Frekuensi	Codeword	Bit	Bit x Frekuensi
1	B5	4	1010	4	16
2	25	2	0101	4	8
3	0D	2	1011	4	8
4	50	1	010010	6	6
5	44	1	100100	6	6
6	46	1	010011	6	6
7	2D	1	100101	6	6
8	31	1	01000100	8	8
9	2E	1	100110	6	6
10	35	1	011011	6	6
11	0A	1	100111	6	6
		Total			82 bit

Setelah kode berhasil *encode* dari perhitungan algoritma *AdditiveCode* diatas, langkah selanjutnya adalah menyusun kembali *string* bit yang telah dihasilkan dari proses kompresi sesuai dengan posisi karakter pada nilai *hexadecimal*.

Tabel 4. *String bit* Hasil Kompresi Menggunakan Algoritma *Additive Code*

25	**50**	**44**	**46**
0101	010010	100100	010011
2D	**31**	**2E**	**35**
100101	01000100	100110	011011
0D	**0A**	**25**	**B5**
1011	100111	0101	1010
B5	**B5**	**B5**	**0D**
1010	1010	1010	1011

Berdasarkan tabel 4 *string* bit yang dihasilkan oleh proses kompresi *file* SQL menggunakan algoritma *Additive Code* dapat ditulisan sebagai berikut:

Tabel 5. *String bit* Hasil Kompresi

0101	010010	100100	010011
100101	01000100	100110	011011
1011	100111	0101	1010
1010	1010	1010	1011

String bit dari hasil kompresi yaitu 82 bit "01010100101001000100111 00101010001001001100110111011100111010110101010101010101011"

Sebelum mendapatkan hasil keseluruhan akhir kompresi dilakukan penambahan *string* bititu sendiri yaitu *padding* bit dan *flagging* bit. Jika sisa bagipanjang *string* bit terhadap 8 adalah 0 maka tambahan 00000001, nyatakan dengan bit akhir.Sedangkan jika sisa bagi panjang *string* bit terhadap 8 adalahn(1, 2, 3, 4, 5, 6, 7) maka tambahkan 0 sebanyak 7 – *n* + "1" di akhir *string* bit. Lalu untuk *flagging* tambahkan bilangan biner dari 9 – n. Selanjutnya, *string* bit dibagi per 8 bit dan nyatakan dengan bit akhir. Karena jumlah string bit 82 tidak habis dibagi delapan da nsisanya2, bisa dengan menggunakan 82 mod 8 = 10 sisa 2. Dan dinyatakan dengan simbol n, n = 2.Berdasarkan perhitungan sisa bagi sebelumnya, didapat hasil n adalah 2. Maka selanjutnya menghitung 7 – n + "1" = 7 – 2 + "1" = **000001**(karena 7 – 2 = 5, maka bit "0" ada 5 yang ditambahkan + bit "1"). Lalu berikutnya 9 – n = 9 – 2 = 7, biner dari 7 = **00000111.**

Tabel 6. Penambahan *Padding* Dan *Flagging*

PADDING	***FLAGGING***
82 mod 8 = 10 sisa 2 7 – n + "1" = 7 – 2 + "1" = **000001**(karena 7 – 2 = 5, maka bit "0" ada 5 yang ditambahkan + bit "1").	9 – n 9 – 2 = 7 biner dari 7 = **00000111**

Jurnal Sains Dan Teknologi Informasi
Vol 3, No 2, Maret 2024, Hal 68-78
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v3i2.4799
Website https://ejurnal.seminar-id.com/index.php/jussi

String bit yang didapat ditambah dengan *padding* dan *flagging* yaitu: "01010100101001000100111001010100010010011001101110111001110101101010101010101011**00000100000111**" Panjang bit keseluruhan setelah ada penambahan *padding* dan *flagging* adalah 82+14=96 bit. Selanjutnya lakukan pemisahan bit manjadi beberapa kelompok. Setiap kelompok terdiri dari 8 bit seperti tabel dibawah ini:

Tabel 6. Pengelompokan Bit

01010100	10100100	01001110	01010100
01001001	10011011	10111001	11010110
10101010	10101010	11**000001**	**00000111**

Berdasarkan pembagian kelompok nilai biner, dihasilkan 12 kelompok nilai biner baru yang telah terkompresi bersama nilai biner penambahan bit. Setelah pembagian dilakukan, maka nilai yang telah dibagi dirubah ke dalam suatu karakter dengan mencari nilai desimal berdasarkan string bit tadi memakai kode ASCII untuk mengetahui nilai yang telah terkompresi bisa ditinjau dalam tabel di bawah ini:

Tabel 7. Hasil Karakter Terkompresi

No	Codeword setelah Kompresi	Desimal	Hexadecimal	Karakter
1	01010100	84	54	T
2	10100100	164	A4	¤
3	01001110	78	4E	N
4	01010100	84	54	T
5	01001001	73	49	I
6	10011011	155	9B	
7	10111001	185	B9	¹
8	11010110	214	D6	Ö
9	10101010	170	AA	ª
10	10101010	170	AA	ª
11	11**000001**	193	C1	Á
12	**00000111**	7	7	

Setelah nilai desimal didapatkan, maka mengubah nilai desimal ke dalamsuatu karakter. Karaktern dari proses kompresi yang dihasilkan disimpan dalam aplikasi kompresi *record database*.Selanjutnya mengukur hasil kompresi data diatas sesuai dengan parameter yang sudah ditentukan. Semakin kecil hasil kompresinya maka semakin berkualitas hasil kompresinya, begitu jugq sebaliknya. Berikut adalah parameter untuk mengukur kinerja algoritma Additive Code.

a. *Ratio of Compression* (RC)

$$RC = \frac{UkuranDataSebelumDikompresi}{UkuranDataSetelahDikompresi}$$

$$RC = \frac{128}{96} = 1{,}33$$

b. *Compression Ratio* (CR)

$$CR = \frac{UkuranDataSetelahDikompresi}{UkuranDataSebelumDikompresi} \times 100\%$$

$$CR = \frac{96}{128} \times 100\% = 75\%$$

c. *Redudancy* (Rd)

$$Rd = \frac{filesebelumdikompresi - filesetelahdikompresi}{Ukuranfilesebelumdikompresi} x\ 100\%$$

$$Rd = \frac{128 - 96}{128} x\ 100\% = 25\%$$

d. *Space Saving* (Ss)

$$S_s = \left(\left(1 - \frac{filesetelahdikompresi}{filesebelumdikompresi}\right)\right) x\ 100\%$$

$$S_s = \left(\left(1 - \frac{96}{128}\right)\right) x\ 100\% = (1 - 0.75)\ x\ 100\% = \ 25\%$$

Jurnal Sains Dan Teknologi Informasi
Vol 3, No 2, Maret 2024, Hal 68-78
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v3i2.4799
Website https://ejurnal.seminar-id.com/index.php/jussi

Berdasarkan hitungan di atas didapatkan hasil bahwa persentase *Compression Ratio* (CR) dengan menggunakan algoritma *additivecode* sebesar75%.

3.3 Proses Dekompresi

Proses dekompresi dilakukan pada saat *user* memilih *file* SQL dari aplikasi kompresi *record database*. Saat proses upload selesai dilakukan kemudian menekan tombol dekompresi maka aplikasi kompresi *record database* melakukan pembacaan nilai *hexadecimalfile* yang diupload kemudian dilakukan proses dekompresi.

Kemudian analisa karakter tersebut ke dalam bentuk biner dan desimal agar menghasilkan bithasil kompresi sebelumnya. Bit hasil kompresi sebelumnya dapat dilihat pada tabel di bawah:

Tabel 8. Nilai Desimal dan Biner Hasil Kompresi

No	Karakter	Biner	Desimal	Hexadecimal
1	T	01010100	84	54
2	¤	10100100	164	A4
3	N	01001110	78	4E
4	T	01010100	84	54
5	I	01001001	73	49
6		10011011	155	9B
7	¹	10111001	185	B9
8	Ö	11010110	214	D6
9	ª	10101010	170	AA
10	ª	10101010	170	AA
11	Á	11**000001**	193	C1
12		**00000111**	7	7

Mengembalikan binermenjadi *string* bit semula dengan menghilangkan biner *padding* dan *flagging* dapat dilakukan melalui langkah berikut ini. Lakukan pembacaan pada 8 bit terakhir, hasil pembacaan berupa bilangan desimal. Nyatakan hasil pembacaan dengan n, hilangkanbit pada bagian akhir sebanyak 7+n, setelah dilakukan perhitungan pembacaan bitakhir, nilai biner yang dihilangkan sebanyak 8 bit pada akhir n = 1, hilangkan 7 + natau 7 + 1 = 8. Penjelasan diatas menunjukkan bahwa bit akhir harus dihilangkan.Nyatakan hasil pembacaan dengan n, kemudian gunakan rumus $7 + n$ untuk mengembalikan *string* bit kebentuk semula.

"01010100101001000100111001010100010010011001101110111001110101101010101010101011**00000100000111**"

8 bit terakhir = **00000111** = 7 = n

7 + n = 7 + 7 = 14

Hilangkan dari *string* bit sebanyak 14 bit terakhir, sehingga menjadi:

"01010100101001000100111001010100010010011001101110111001110101101010101010101011"

Berdasarkan perhitungan dengan algoritma *Additive Code, string* bit diatas berjumlah 82 bit seperti diawal sehingga dilakukan pembacaan *string* bit awal. Pada pembacaan *string* bit dengan algoritma *Additive Code* dilakukan dengan menggunakan *Brute Force*. Dimana pembacaan *string* bit dilakukan dengan *index* terkecil yang kemudian dilanjut ke *index* setelahnya yang tidak mewakili karakter *Additive Code*. Di bawah ini tabel hasil pengecekan bit yang telah dikompresi sebagai berikut:

Tabel 9. Pengecekan Bit Dekompresi

Indeks	NilaiBiner	Keterangan	Karakter
1	0	TidakAda	
2	01	TidakAda	
3	010	TidakAda	
4	0101	Ada	25
5	1	TidakAda	
6	10	TidakAda	
7	101	TidakAda	
8	1011	Ada	0D
9	0	TidakAda	
10	01	Tidak Ada	
11	010	TidakAda	
12	0100	TidakAda	
13	01001	TidakAda	
14 dst.			

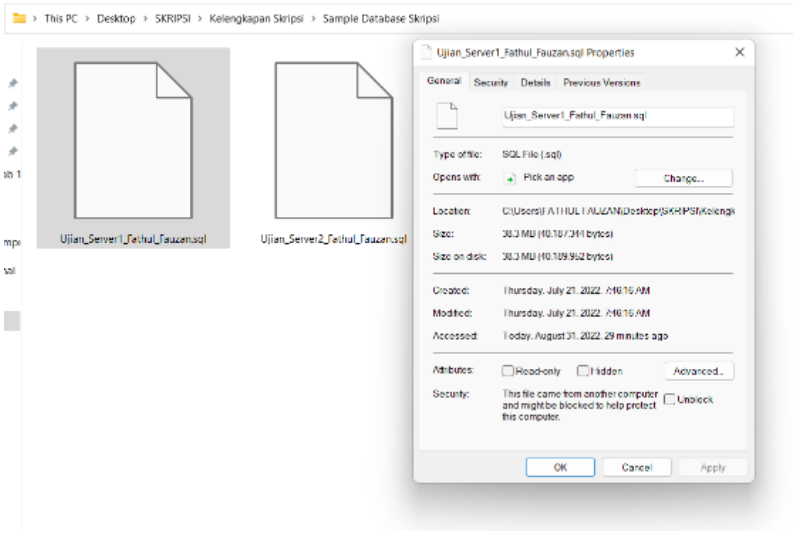
Hasil akhir dari proses dekompresi dapat dilihat pada tabel sebagai berikut:

Jurnal Sains Dan Teknologi Informasi
Vol 3, No 2, Maret 2024, Hal 68-78
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v3i2.4799
Website https://ejurnal.seminar-id.com/index.php/jussi

Tabel 10. Tampilan Nilai Awal Bilangan Hexadecimal

25	50	44	46	2D	31	2E	35
0D	0A	25	B5	B5	B5	B5	0D

Nilai *hexa decimal* dijadikan kembali menjadi *file* SQL. Berikut adalah hasil dekompresi berupa *file* SQL.



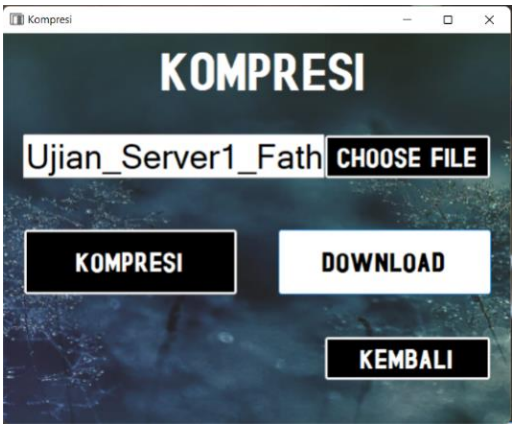
Gambar 5. Sampel *File* SQL Hasil Dekompresi

3.4 Hasil Pengujian

Pengujian dilakukan terhadap sistem yang telah dirancang dan akan diaplikasikan. Sistem yang akan diaplikasikan dirancang sesederhana mungkin agar pengguna sistem dapat mengaplikasikannya dengan mudah.

a. Halaman Kompresi

Halaman kompresi adalah halaman untuk melakukan proses kompresi file database. Dalam halaman ini pengguna harus memilih file yang melalui proses kompresi di *directory* pengguna. Setelah memilih file yang ingin dikompresi maka halaman kompresi tampil kembali dan serta dari file *database* yang dipilih muncul di teksbox pada halaman kompresi. Kemudian untuk melakukan proses kompresi file maka pengguna harus menekan tombol kompresi. Apabila proses kompresi selesai dilakukan maka tombol download pada aplikasi aktif dan file data yang telah dikompresi tadi dapat di download untuk disimpan pada penyimpanan komputer pengguna.

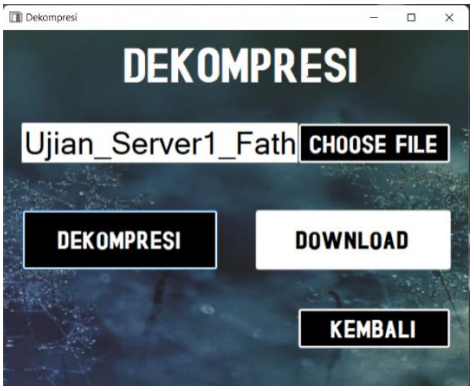


Gambar 6. Halaman Proses Kompresi

b. Halaman Dekompresi

Halaman dekompresi adalah halaman untuk melakukan proses dekompresi file database yang sebelumnya sudah melalui proses kompresi. Dalam halaman ini pengguna harus memilih file yang melalui proses dekompresi di *directory* pengguna. Setelah memilih file yang ingin didekompresi maka halaman dekompresi tampil kembali dan serta dari file *database* yang dipilih muncul di teksbox pada halaman kompresi. Kemudian untuk melakukan proses kompresi file maka pengguna harus menekan tombol dekompresi. Apabila proses dekompresi selesai dilakukan maka tombol download pada aplikasi aktif dan file data yang telah dikompresi tadi dapat di download untuk disimpan pada penyimpanan komputer pengguna.

Jurnal Sains Dan Teknologi Informasi
Vol 3, No 2, Maret 2024, Hal 68-78
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v3i2.4799
Website https://ejurnal.seminar-id.com/index.php/jussi



Gambar 7. Halaman Proses Dekompresi

Kualitas kompresi *file* SQL sebelum dikompresi dan setelah dikompresi dapat dilihat dari table hasil pengujian yang diukur berdasarkan parameter kinerja *Ratio of compression* (Rc), *Compression ratio* (Cr), *Redudancy* (Rd), dan *Space saving* (Ss).

Tabel 11. Hasil Pengujian Kualitas Kompresi

No	Sebelum kompresi		Sesudah kompresi			
	Nama *File* SQL	Ukuran	Nama *File* SQLTerkompresi	Ukuran	Kinerja	
1	database_ujian_server_1.sql	39,2 MB	database_ujian_server_1 Compression.sql	29,5 MB	Rc	1,33
					Cr	75%
					Rd	25%
					Ss	25%
2	database_ujian_server_2.sql	34,4 MB	database_ujian_server_2 Compression.sql	27,7 MB	RC	1,23
					CR	81,25%
					Rd	18,75%
					Ss	18,75%

Hasil pengujian diatas dapat disimpulkan bahwa *file* SQL terkompresi memiliki ukuran yang lebih kecil dibandingkan dengan *file* SQL yang belum dikompresi sehingga dapat menghemat ruang penyimpanan pada aplikasi kompresi *record database* berbasis *dekstop*. Sedangkan hasil pengujian dekompresi terhadap *file* SQLterkompresi dapat dilihat pada tabel berikut:

Tabel 12. Hasil Pengujian Dekompresi

No	Nama *File* SQL Terkompresi	Ukuran	Nama *File* SQL Sesudah Dekompresi	Ukuran
1	database_ujian_server_1.sql	39,2 MB	database_ujian_server_1 Compression.sql	29,5 MB
2	database_ujian_server_2.sql	34,4 MB	database_ujian_server_2 Compression.sql	27,7 MB

4. KESIMPULAN

Berdasarkan dari pembahasan penelitian yang telah dilakukan, maka didapatkan hasil akhir dari penelitian ini dimana proses kompresi berlangsung pada saat file SQL diupload kemudian ditekan tombol kompresi pada aplikasi kompresi record database dan apabila proses kompresi selesai dilakukan maka tombol download akan aktif kemudian kita dapat mendownload file SQL yang sudah dikompresi dan proses dekompresi berlangsung pada saat file SQL yang telah dikompresi diupload kemudian ditekan tombol dekompresi pada aplikasi kompresi record database dan apabila proses dekompresi selesai dilakukan maka tombol download akan aktif kemudian kita dapat mendownload file SQL yang sudah didekompresi. Setelah mengikuti prosedur kompresi dengan menggunakan algoritma additive code dihasilkan bahwa suatu file SQL yang memiliki ukuran yang cukup besar dapat dikompresi menjadi file SQL baru dengan ukuran yang lebih kecil sehingga meminimalisir ruang penyimpan dan Aplikasi kompresi record database berbasis dekstop berjalan dengan baik. Setelah menerapkan algoritma additive code untuk mengkompresi file SQL telah terbukti bahwa file SQL tersebut berhasil dikompresi dengan mengukur rasio kompresi, hasilnya adalah Ratio of Compression (RC) = 1,33, Compression Ratio (CR) = 75%, Redudancy (Rd) = 25%, dan Space saving (Ss) = 25%.

REFERENCES

[1] R. Y. TANJUNG, “Perancangan aplikasi kompresi file dokumen menggunakan algoritma additive code,” vol. 8, no. 4, hal. 108–113, 2020, doi: 10.30865/jurikom.v8i4.3593.


Jurnal Sains Dan Teknologi Informasi
Vol 3, No 2, Maret 2024, Hal 68-78
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v3i2.4799
Website https://ejurnal.seminar-id.com/index.php/jussi

[2] R. D. Pratiwi, S. D. Nasution, dan F. Fadlina, "Perancangan Aplikasi Kompresi File Teks Dengan Menerapkan Algortima Fixed Length Binary Encoding (Flbe)," J. Media Inform. Budidarma, vol. 2, no. 1, hal. 10–14, 2018, doi: 10.30865/mib.v2i1.813.
[3] S. Nainggolan, "Analisa Perbandingan Algoritma Goldbach Codes Dengan Algoritma Dynamic Markov Compression (DMC) Pada Kompresi File Teks Menggunakan Metode Eksponensial," Maj. Ilm. INTI, vol. 6, no. 3, hal. 395–399, 2019.
[4] M. Mawar, "Perancangan Aplikasi Kompresi File Pdf Dengan Menerapkan Algoritma Punctured Elias Codes," Inf. dan Teknol. Ilm., vol. 7, no. 3, hal. 217–223, 2020.
[5] R. A. Sandra, "Implementasi Kombinasi Algoritma Tunstall Code da Boldi-Vigna Untuk Kompresi File Pdf," vol. 8, no. 2, hal. 67–71, 2021.
[6] A. Apijuddin Kompresi dan A. S. Codes, "Perancangan Aplikasi Kompresi File Teks Menggunakan Algoritma Stout Codes," vol. 9, hal. 183–188, 2021.
[7] N. L. W. S. R. Ginantara et al., BASIS DATA : Teori dan Perancangan. 2020.
[8] E. Hariska et al., "Perancangan Aplikasi Kompresi File Gambar Menggunakan Algoritma Additive Code," vol. 5, hal. 193–202, 2021, doi: 10.30865/komik.v5i1.3671.
[9] D. Salomon dan G. Motta, Handbook of Data Compression 5th, vol. 53, no. 9. 2019.
[10] R. Wanti, "Rancang Bangun Sistem Informasi Akademik Pada SMK Citra Dharma Berbasis JAVA," J. Teknol. Inf., vol. 5, no. 2, hal. 86–92, 2019.
[11] A. N. A. Andrias, "Peranan Sistem Informasi Manajemen Pai Untuk Meningkatkan Kualitas Guru Di Smp Plus Al Munawar," Textura, vol. 1, no. 1, hal. 1–12, 2020.
[12] S. R. Saragih dan D. P. Utomo, "Penarapan Algoritma Prefix Code Dalam Kompresi Data Teks," KOMIK (Konferensi Nas. …, vol. 4, hal. 249–252, 2020, doi: 10.30865/komik.v4i1.2691.