PENGEMBANGAN APLIKASI PENGUMPULAN DATA MENGGUNAKAN *SPREADSHEET*

Laporan Tugas Akhir

Disusun sebagai syarat kelulusan tingkat sarjana

Oleh

Feryandi Nurdiantoro

NIM : 13513042

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

Juli 2017

PENGEMBANGAN APLIKASI PENGUMPULAN DATA MENGGUNAKAN *SPREADSHEET*

Laporan Tugas Akhir

Oleh

Feryandi Nurdiantoro

NIM : 13513042

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

Telah disetujui dan disahkan sebagai draf Laporan Tugas Akhir

di Bandung, 19 Juni 2017

Mengetahui,

Pembimbing I,	Pembimbing II,
Tricya Esterina Widagdo, ST., M.Sc.	Yudistira Dwi Wardhana Asnar, Ph.D
NIP 197109071997022001	NIP 198008272015041002

ABSTRAK

PENGEMBANGAN APLIKASI PENGUMPULAN DATA MENGGUNAKAN *SPREADSHEET*

Oleh

FERYANDI NURDIANTORO

NIM : 13513042

Penggunaan *spreadsheet* di dalam kehidupan sehari-hari tidak terlepas dari pengumpulan data. Hal ini disebabkan oleh mudahnya penggunaan *spreadsheet* sehingga banyak orang awam yang memilih menggunakan *spreadsheet* dibandingkan basis data. Pengumpulan data menggunakan aplikasi *spreadsheet* memiliki beberapa kelemahan seperti tidak adanya validasi, terisolasinya data yang dikumpulkan, serta terdapat kemungkinan sulitnya berkolaborasi didalam pengumpulan data. Sehingga masalah yang ingin diselesaikan disini adalah transformasi data menjadi bentuk basis data, melakukan verifikasi terhadap data, dan dapat dilakukan secara kolaboratif.

Pada laporan ini akan dibahas mengenai cara mentransformasikan data yang terdapat pada *spreadsheet* sehingga dapat diolah ke dalam bentuk basis data. Transformasi ini akan terdiri dari 4 tahap utama yakni, *clustering*, *row identification*, *cell identification*, dan *header-data assignment*. Algoritma yang akan digunakan adalah Hierarchical Clustering serta Conditional Random Field, data pembelajaran didapatkan dari Statistical Abstract of the United States (SAUS) 2010 serta pengumpulan manual. Validasi akan dilakukan terhadap 3 tipe validasi yakni, tipe data, domain data, dan relasi antar data. Algoritma tersebut akan dibangun diatas *spreadsheet* kolaboratif bernama Ethercalc sebagai solusi untuk dapat melakukan pengumpulan data secara kolaboratif.

Diakhir laporan akan dibahas mengenai hasil percobaan menggunakan perangkat lunak yang telah dibangun. Pengujian dilakukan dengan menggunakan beberapa data set yang dibuat dan diujikan kebenaran hasil algoritma serta kemampuannya untuk mengintrepertasikan data masukan menjadi data pada basis data sehingga didapatkan tingkat akurasi dari perangkat lunak yang dibangun.

Kata kunci: *spreadsheet*, pengumpulan data, *data quality*, *data management*.

KATA PENGANTAR

TBD

Daftar Isi

Daftar Gambar

Daftar Tabel

BAB I

PENDAHULUAN

Pada bab ini akan dibahas mengenai gambaran dasar dari pelaksanaan Tugas Akhir dalam bentuk penjelasan latar belakang yang mendasari pemilihan topik. Dari latar belakang tersebut, akan diurai kembali menjadi rumusan masalah, tujuan, batasan masalah, serta metodologi yang digunakan untuk keperluan Tugas Akhir ini.

I.1 Latar Belakang

Aplikasi *spreadsheet* merupakan aplikasi yang mudah ditemui dan wajar digunakan oleh banyak orang, baik secara personal maupun dalam sebuah organisasi komersial (Chan and Storey, 1996). Pada tahun 1979, aplikasi *spreadsheet* pertama dibuat dengan nama VisiCalc. Pengguna komputer pada saat itu dimanjakan dengan kapabilitas dan fleksibilitas aplikasi yang dapat melakukan operasi sederhana tanpa harus menggunakan komputer *mainframe*. Dengan semakin berkembangnya daya komputasi, saat ini telah banyak sekali muncul aplikasi *spreadsheet* baru dan penggunaannya juga semakin beragam.

Spreadsheet memiliki beberapa keunggulan dibandingkan dengan aplikasi pengolahan data jenis lain. Keunggulan yang paling terlihat adalah banyak orang yang mengetahui cara penggunaan aplikasi jenis *spreadsheet* dan terbiasa dalam menggunakannya. Selain itu, *spreadsheet* memiliki banyak fitur dan kemampuan yang jarang diketahui orang awam jika digunakan dengan benar. Dengan keunggulan ini, *spreadsheet* sering kali dijadikan pilihan utama dalam pengolahan dan pengumpulan data. Beberapa orang mungkin menganggap, penggunaan *spreadsheet* adalah personal sehingga tidak membutuhkan tim atau bantuan orang lain dalam pembuatan suatu *spreadsheet*. Hal ini tidak dapat dibenarkan, karena jika melihat kasus penggunaannya pada organisasi bisnis yang besar, *spreadsheet* yang dihasilkan sangatlah kompleks dan besar dengan pengembangan yang membutuhkan banyak orang (Panko, 1998).

Penggunaan *spreadsheet* yang dapat ditemui pada perusahaan atau instansi adalah sebagai

media pengumpulan data. Data yang dikumpulkan biasanya dalam bentuk *data frame* yakni *spreadsheet* yang mempunyai dua struktur utama yaitu bagian *value* atau data dan bagian *atribute* atau label (Chen and Cafarella, 2013). Data yang telah dikumpulkan biasanya akan disebarkan kepada pihak-pihak yang membutuhkan atau disimpan sebagai arsip data yang akan digunakan kembali pada saat dibutuhkan.

Sebagai media pengumpulan data, *spreadsheet* memiliki permasalahan penyimpanan data. Hal tersebut penting untuk diperhatikan jika data yang dikumpulkan mungkin tidak hanya akan ditampilkan dalam bentuk *spreadsheet* namun juga dapat digunakan oleh aplikasi lain. Permasalahan lain yang mungkin muncul adalah kompleksitas dan besarnya ukuran data pada *spreadsheet* yang membuat penggunaan *spreadsheet* pada sebuah bisnis sangatlah rentan akan kesalahan. Sebuah kesalahan kecil dapat berakibat fatal dan memberikan kerugian seperti kehilangan pendapatan, kesalahan pemberian harga, penipuan, dan kegagalan sistem akibat ketergantungan berlebih antar *spreadsheet* (EuSpRIG, 2010b). Telah banyak bukti dan penelitian yang menunjukan bahwa kesalahan pada *spreadsheet* sangat mudah ditemui. Contoh kesalahan yang dapat terjadi adalah kesalahan tipe data, kesalahan masukan, tidak divalidasinya data masukan, serta permasalahan *single version of truth* dimana bisa terdapat dua atau lebih versi dari data yang sama.

Untuk dapat mengurangi kesalahan-kesalahan yang sering terjadi pada *spreadsheet* secara lebih mendasar, dibutuhkan bantuan perangkat lunak untuk dapat melakukan kontrol terhadap masukan pengguna, melakukan validasi, serta dapat melakukan penyimpanan data yang telah dikumpulkan. Pada tugas akhir ini akan difokuskan pada pengembangan sebuah aplikasi pengumpulan data berbentuk *spreadsheet* yang dapat membantu pengguna mengurangi terjadinya masalah-masalah yang telah disebutkan sebelumnya.

I.2 Rumusan Masalah

Seperti yang telah dijelaskan pada latar belakang, salah satu penggunaan aplikasi *spreadsheet* adalah sebagai media pengumpulan data. Beberapa permasalahan yang muncul dalam pengumpulan data adalah kolaborasi, validasi, dan penyimpanan data. Pengumpulan data dapat dilakukan oleh banyak orang secara bersama-sama sehingga dapat menyebabkan konflik pada data masukan. Konflik ini bisa terjadi karena terdapat lebih

dari satu versi file yang diubah secara bersama-sama. Selain itu, data yang dimasukkan biasanya tidak melalui proses validasi sehingga dapat menyalahi domain data yang seharusnya. Setelah data dikumpulkan ke dalam bentuk *spreadsheet*, diperlukan media penyimpanan data sehingga data tidak terisolasi. Contoh dari media penyimpanan tersebut adalah menggunakan basis data.

Hal-hal tersebut merupakan beberapa masalah utama yang cukup penting untuk diselesaikan terutama dalam penggunaannya pada bisnis dan komersial sehingga akan dibentuk sebuah aplikasi yang membantu mengurangi tingkat kesalahan pada *spreadsheet*. Dalam rangka pembangunan aplikasi, terdapat beberapa permasalahan yang menjadi perhatian pada tugas akhir ini, yaitu

1. Bagaimana cara menangani konflik yang terjadi pada saat kolaborasi?
2. Bagaimana cara melakukan validasi terhadap data masukan?
3. Bagaimana cara penentuan bagian label dan data pada *spreadsheet* untuk dijadikan skema relasional?
4. Bagaimana cara melakukan penyimpanan data yang telah dibuat?
5. Bagaimana perubahan alur kerja yang terjadi akibat penggunaan aplikasi yang dibuat?

I.3 Tujuan

Tujuan yang ingin dicapai dalam Tugas Akhir ini adalah mengembangkan perangkat lunak pengumpulan data berbentuk *spreadsheet* yang dapat mengurangi tingkat kesalahan pada data masukan. Dengan adanya perangkat lunak ini diharapkan dapat memvalidasi data masukan dengan lebih baik dan diharapkan dapat meningkatkan efisiensi pengumpulan data dengan bantuan basis data.

I.4 Batasan Masalah

Dalam pengerjaan Tugas Akhir ini, terdapat beberapa batasan-batasan yang perlu diperhatikan. Batasan tersebut ditujukan untuk memperjelas dan memfokuskan objek penelitian

dan pengembangan tugas akhir. Batasan-batasan masalah pengerjaan tugas akhir adalah sebagai berikut,

1. Fitur kolaborasi pada *spreadsheet* bukan fokus utama, fitur akan diambil dari aplikasi *spreadsheet* yang telah ada.
2. Fokus data masukan pada Tugas Akhir ini berupa data mentah yang belum direkapitulasi.
3. Tingkat normalisasi struktur basis data yang terbentuk yang berasal dari data pada *spreadsheet* bukan merupakan fokus utama pada Tugas Akhir ini.
4. Basis data yang didukung untuk digunakan dalam penyimpanan data hanya MySQL

I.5 Metodologi

Metodologi yang digunakan dalam pengerjaan Tugas Akhir ini yakni:

1. Studi Literatur

   Pengerjaan tugas akhir diawali dengan mencari dan mempelajari referensi berupa jurnal ilmiah dan aplikasi-aplikasi yang telah ada sebelumnya yang dapat membantu pengembangan aplikasi yang dibuat pada tugas akhir ini. Literatur yang dicari dan dipelajari berkaitan dengan topik tugas akhir yaitu mengenai *spreadsheet*, penggunaannya pada bisnis, kesalahan yang sering dilakukan dalam pembuatan, metode *quality control* yang dapat dilakukan, serta hal-hal lain yang masih berkaitan dengan topik tugas akhir ini.

2. Analisis Masalah

   Pada tahap ini dilakukan analisis permasalahan yang berkaitan dengan topik yang diangkat pada tugas akhir ini. Selain itu, dilakukan penentuan spesifikasi dan fitur yang ada pada aplikasi tersebut sebagai bentuk solusi terhadap permasalahan yang dianalisis.

3. Perancangan Solusi

   Pada tahap ini dilakukan perancangan solusi yang dapat menyelesaikan masalah-masalah yang telah dijelaskan pada bagian analisis masalah. Bagian perancangan

ini juga menjelaskan arsitektur yang digunakan untuk membangun perangkat lunak berdasarkan spesifikasi dan metode yang digunakan.

4. Implementasi

   Pada tahap ini dilakukan pembangunan aplikasi sesuai dengan kebutuhan dan spesifikasi dari hasil analisis masalah serta rancangan solusi yang diajukan.

5. Pengujian dan Analisis Hasil

   Pada tahap ini dilakukan pengujian dengan menggunakan data set uji yang sesuai dengan batasan masalah ke dalam aplikasi yang diimplementasikan. Selanjutnya dilakukan analisis hasil pengujian dan penarikan kesimpulan.

I.6 Sistematika Pembahasan

Penulisan tugas akhir ini terdiri dari 5 bab, yaitu: BAB I Pendahuluan, BAB II Tinjauan Pustaka, BAB III Analisis dan Perancangan, BAB IV Rancangan, Implementasi, dan Pengujian, dan BAB V Penutup.

Bab satu membahas mengenai latar belakang permasalahan, rumusan masalah, tujuan, batasan masalah, metodologi serta sistematika pembahasan yang digunakan. Bab ini juga menjelaskan secara umum isi dari tugas besar serta gambaran dasar dari pelaksanaan tugas akhir.

Bab dua menjelaskan mengenai dasar teori yang digunakan didalam menyelesaikan permasalahan yang diangkat. Teori yang digunakan berasal dari literatur dan referensi yang berhubungan dengan permasalahan yang diangkat seperti hal-hal yang berkaitan dengan *spreadsheet*, penggunaannya pada bisnis, masalah yang sering terjadi dalam pembuatan, metode *quality control* yang dapat dilakukan untuk mencegah kesalahan, serta metode pendeteksian bagian label dan data yang pernah dibuat pada penelitian lain. Dasar teori ini menjadi dasar analisis dan rancangan solusi pada bab selanjutnya.

Bab tiga memaparkan analisa kebutuhan dan permasalahan yang dipilih yakni tingginya tingkat kesalahan yang terjadi pada *spreadsheet*. Dari hasil analisa yang dilakukan, di dapatkan bentuk solusi umum yang dapat digunakan untuk mengatasi permasalahan tersebut. Selanjutnya solusi umum tersebut dibuat rancangan dan arsitekturnya agar dapat

diimplementasikan.

Bab empat memperlihatkan rancangan perangkat lunak yang dibuat serta hasil implementasinya. Pada akhir bab akan ditunjukkan hasil pengujian yang dilakukan kepada aplikasi yang dibuat dan pembahasan dari pengujian tersebut. Pengujian dilakukan untuk mengetahui keberhasilan aplikasi yang dibuat untuk menyelesaikan permasalahan yang di definisikan pada rumusan masalah.

Bab lima berisikan kesimpulan terhadap hasil implementasi dan solusi yang dipaparkan untuk menyelesaikan permasalahan. Selain itu, terdapat bagian saran yang memaparkan saran pengembangan dan perbaikan yang dapat dilakukan untuk memperkaya fitur dan menyelesaikan permasalahan yang lebih luas.

BAB II

STUDI LITERATUR

Pada bab ini akan dideskripsikan kajian literatur yang terkait dengan persoalan Tugas Akhir. Studi literatur ini akan dijadikan dasar didalam melakukan penyelesaian persoalan yang telah didefinisikan.

II.1 *Spreadsheet*

Bagian ini akan membahas mengenai definisi umum *spreadsheet* serta teknologi yang sering digunakan didalam membuat *spreadsheet*. Dengan adanya definisi umum ini diharapkan dapat menyamakan persepsi mengenai *spreadsheet* yang dimaksud pada Tugas Akhir ini. Teknologi yang dijelaskan pada bagian ini merupakan teknologi yang memungkinkan untuk dijadikan dasar pengembangan perangkat lunak pada Tugas Akhir ini.

II.1.1 Definisi Umum

Secara harafiah, *spreadsheet* adalah suatu perangkat lunak yang dapat melakukan kalkulasi terhadap angka serta mengorganisir informasi yang ada di dalamnya berdasarkan kolom dan baris (Dictionary, 2016). Konsep dasar pada aplikasi *spreadsheet* modern adalah sebuah aplikasi yang berupa sekumpulan sel terdiri dari baris dan kolom yang disebut *sheet* yang dapat digambarkan sebagai matriks yang besar (Ronen et al., 1989).

Sel-sel pada *spreadsheet* dapat diisi data berupa data mentah maupun formula. Data mentah dapat berupa angka, teks, tanggal, dan nilai mata uang. Formula merupakan perintah yang dapat dimengerti komputer untuk menghitung dan memanipulasi data pada sel. Data hasil pengolahan dan masukan pada *spreadsheet* ditampilkan dalam bentuk sel yang namanya terdiri dari nama kolom dan nilai baris (Contoh: A1 untuk kolom pertama dan baris pertama). Selain itu, sel tersebut juga dapat memiliki *properties* berupa *value* yang diisikan, format sel, serta format data yang digunakan.

II.1.2 Teknologi *Spreadsheet*

Perkembangan teknologi *spreadsheet* digital modern dimulai pada tahun 1978, saat Bricklin mengembangkan *working prototype* dari konsep dasar *spreadsheet* menggunakan Integer BASIC. Pada tahun yang sama, Frankston dan Fylstra bergabung dan membentuk sebuah perangkat lunak bernama VisiCalc (Visible Calculator) yang merupakan sebuah perangkat lunak *spreadsheet* pertama yang bekerja dengan baik dan sukses dipasaran. Setelah keberhasilan VisiCalc, mulai muncul aplikasi serupa yang semakin baik salah satunya adalah Lotus. Dengan berkembangnya daya komputasi dan munculnya konsep *graphical user interface*, Microsoft mengembangkan Microsoft Excel yang merupakan *spreadsheet* pertama yang menggunakan antarmuka grafis dan menggunakan *mouse* sebagai alat kontrol (Power, 2004).

Saat ini, perangkat lunak berupa *spreadsheet* sangat banyak variasi dan tipenya. Perangkat lunak *spreadsheet* ini dapat dibagi berdasarkan konektivitasnya yakni *offline spreadsheet* dan *online spreadsheet*. Selain itu, perangkat lunak *spreadsheet* dapat juga dibagi berdasarkan keterbukaan dari *source code* yakni *open source* dan *closed source*. Bagian ini akan membahas masing-masing perangkat lunak tersebut secara umum.

II.1.2.1 Microsoft Excel

Microsoft Excel adalah perangkat lunak yang dikembangkan oleh Microsoft yang menyediakan fitur dasar dari *spreadsheet* serta dengan fitur-fitur lainnya yang selalu ditambahkan pada setiap iterasi pengembangan Excel. Microsoft Excel dapat dimiliki oleh pengguna melalui pembelian paket Microsoft Office yang berisikan produk esensial Microsoft lainnya (Microsoft, 2015b).

Sejak Microsoft Excel 2007, Microsoft menggunakan format Office Open XML (OOXML) sebagai format penyimpanan (Microsoft, 2015a). Office Open XML dikembangkan oleh Microsoft mulai dari tahun 2000 dengan diimplementasinya dukungan XML pada Microsoft Office 2000. Pada awal penggunaan aplikasi *office*, terdapat permasalahan *data interoperability* antar mesin dan sulitnya manipulasi data. Office Open XML diharapkan dapat menyelesaikan permasalahan ini dengan membentuk standar yang dapat diimplementasi berbagai aplikasi *office* (Microsoft, 2006).

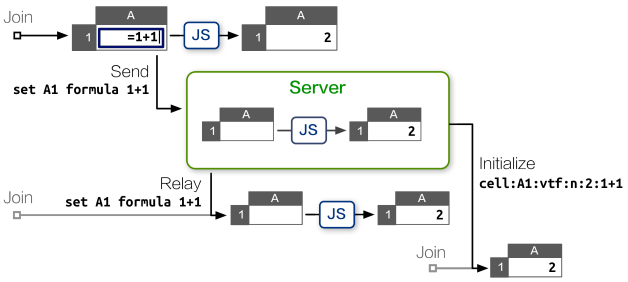
II.1.2.2 LibreOffice Calc

LibreOffice adalah perangkat lunak yang dikembangkan oleh komunitas dan proyek dari organisasi non-profit bernama The Document Foundation. LibreOffice adalah perangkat lunak yang gratis dan *open source* yang awalnya didasarkan pada perangkat lunak serupa yakni OpenOffice.org dan merupakan pengembangan lanjutan dari OpenOffice yang paling aktif. Hampir serupa dengan Microsoft Office, LibreOffice memberikan 6 perangkat lunak yang ada di dalamnya yakni: Writer (pemrosesan teks), Calc (*spreadsheet*), Impress (presentasi), Draw (grafik dan vektor), Base (basisdata), dan Math (editor formula) (Foundation, 2016).

LibreOffice Calc memiliki berbagai kemampuan yang dimiliki oleh kebanyakan *spreadsheet*. Didalam melakukan penyimpanan file, Calc menggunakan format OpenDocument. Format OpenDocument dikembangkan oleh Organization for the Advancement of Structured Information Standards (OASIS) yang bertujuan untuk membentuk *open standard* bagi *office document* (OASIS, 2016).

II.1.2.3 EtherCalc

EtherCalc merupakan perangkat lunak *spreadsheet online* yang *open-source* yang dikembangkan oleh Audrey Tang. EtherCalc merupakan pengembangan yang didasarkan dari perangkat lunak serupa yakni WikiCalc dan SocialCalc. WikiCalc merupakan aplikasi *spreadsheet* yang mengandalkan komputasi server untuk dapat berkolaborasi, sedangkan SocialCalc merupakan aplikasi *spreadsheet* yang menggunakan kemampuan javascript untuk melakukan komputasi pada *client-side*. EtherCalc dikembangkan diatas Node.js dan menggunakan javascript sebagai alat komputasi. Perangkat lunak hanya akan melakukan panggilan ke *server* saat melakukan suatu aksi dan *server* yang bertugas untuk menyimpan kumpulan aksi tersebut agar pengguna lain yang ikut berkolaborasi dapat melihat *spreadsheet* yang sama satu dengan yang lain (Tang, 2014). Gambar II.1 merupakan gambaran umum cara kerja EtherCalc dalam berkolaborasi.



Gambar II.1: Ilustrasi Cara Kerja Kolaborasi pada EtherCalc

II.1.2.4 Google Sheet

Google Sheet merupakan salah satu perangkat lunak pada *office suite* miliki Google. Google Sheet dapat berjalan diatas tiga *platform* yang berbeda yakni: sebagai *web application*, Chrome apps, dan *mobile apps*. Sheet memiliki kemampuan berkolaborasi secara *real-time* dan menyediakan fitur-fitur *spreadsheet* yang ada pada umumnya. Sheet memiliki fitur *revision history* sehingga setiap orang dapat melihat perubahan yang terjadi, *add-ons* yang dapat menambahkan fitur kepada Sheet melalui program yang dibuat oleh komunitas serta fitur *chat* dengan kolaborator. Google Sheet dikembangkan menggunakan bahasa *javascript* yang memberikan kemampuan penyimpanan dan kolaborasi secara *real-time* (Google, 2016).

II.1.2.5 OnlyOffice

Merupakan perangkat lunak sebagai *service* yang dikembangkan oleh Ascensio System SIA. OnlyOffice merupakan *office suite* yang berjalan diatas *web application* yang dipadukan dengan sistem *customer relationship management*. OnlyOffice tidak hanya terdiri dari *online office editor* namun juga terdapat fitur managemen dokumen, manajemen proyek, *mail service*, serta manajemen pelanggan seperti kontak, *invoice*, *opportonities*, dan *task*. OnlyOffice ditujukan kepada bisnis yang membutuhkan perangkat lunak yang dapat mengorganisir kebutuhan bisnis didalam satu aplikasi yang saling terintegrasi (SIA, 2016).

II.2 Penggunaan *Spreadsheet*

Spreadsheet dapat digunakan untuk melakukan kalkulasi terhadap suatu rumus atau formula yang sulit jika dikalkulasikan dengan cara manual. Selain itu, *spreadsheet* dapat juga digunakan untuk melakukan ramalan terhadap suatu perubahan variabel masukan. Pada perkembangannya, *spreadsheet* memiliki fitur-fitur tambahan seperti visualisasi data dan ekstraksi data penting dari kumpulan data yang ada.

Penelitian tentang penggunaan *spreadsheet* pada bisnis pernah dilakukan sebelumnya pada tahun 2014. Subjek yang diteliti adalah akuntan manajemen (Bradbard et al., 2014). Pada penelitian tersebut, didapatkan gambaran umum mengenai penggunaan *spreadsheet* secara umum. Menurut hasil penelitian tersebut beberapa fitur yang sering digunakan oleh pengguna *spreadsheet* secara terurut dari yang paling sering digunakan adalah sebagai berikut,

1. Menghitung fungsi matematika dasar (tambah, kurang, kali, bagi, dan lainnya)
2. Mengelola *worksheet* dan *workbook* (menambahkan, menghapus, merubah nama, dan lainnya)
3. Melakukan perubahan format dasar (menebalkan, memberi garis bawah, format angkat, dan lainnya)
4. Melakukan pengurutan data, penghitungan subtotal, serta meringkas data
5. Menggunakan fitur *cell addressing* baik absolut maupun relatif
6. Penggunaan fungsi kondisi (IF, COUNTIF), fungsi logika (AND, OR), fungsi pencarian (VLOOKUP, HLOOKUP), menautkan *workbook* lain, serta fungsi pembulatan (ROUND, CEILING, FLOOR)

Penggunaan *spreadsheet* sangat bergantung kepada domain bisnis atau organisasi yang menggunakan. Pada bisnis yang berorientasi komersial, *spreadsheet* dapat digunakan sebagai alat bantu perhitungan laba, pengeluaran, investasi, dan pajak. Pada organisasi-organisasi non komersial, *spreadsheet* dapat digunakan sebagai salah satu bentuk basis data yang menangani penyimpanan, pengelolaan, dan pengumpulan data yang mudah dan cepat.

II.3 Kesalahan dalam Penggunaan *Spreadsheet*

II.3.1 Kualitas Data

Kualitas Data (*Data Quality*) adalah tingkat kemampuan data untuk memenuhi kebutuhan penggunaannya (*usage requirement*) sehingga data dapat digunakan dengan baik (Khatri and Brown, 2010). Dimensi yang ada pada kualitas data dapat dibagi menjadi:

1. Akurasi, merujuk kepada tingkat kebenaran dari data.
2. Aktualitas, menunjukan bahwa data yang dicatat merupakan data terbaru.
3. Kelengkapan, menunjukkan bahwa nilai-nilai yang diperlukan tercatat (tidak hilang).
4. Kredibilitas, menunjukkan kepercayaan terhadap sumber serta isinya.

Tingkatan nilai untuk dimensi tersebut dapat berbeda pada setiap kasus yang ada. Contohnya, akurasi 85% untuk data nama dan alamat dokter merupakan nilai yang cukup baik bagi perusahaan asuransi yang menargetkan doktor sebagai konsumen potensial, namun tidak cukup baik untuk perusahaan obat yang ingin melakukan *recall* terhadap obat yang terdistribusi. Kualitas data yang buruk dapat menyebabkan akibat yang fatal dalam bisnis baik secara operasional maupun strategis.

II.3.2 Tingkat Kesalahan dalam Penggunaan *Spreadsheet*

Penelitian telah dilakukan oleh Panko (Panko, 1998) untuk mengetahui banyaknya kesalahan yang terjadi pada pengembangan *spreadsheet* terutama pada sektor bisnis. Dari penelitian ini, didapatkan bahwa 20% hingga 40% *spreadsheet* mengandung kesalahan. Pada kasus tertentu, bahkan ditemukan 90% *spreadsheet* yang diteliti memiliki kesalahan (Freeman, 1996).

Penelitian yang dilakukan oleh Panko juga menemukan 88% dari 113 *spreadsheet* yang diaudit melalui 7 lebih studi yang diteliti. Beberapa hasil yang telah di rangkum oleh penelitian tersebut mengunakan *spreadsheet* yang digunakan di dunia nyata dapat dilihat pada Tabel II.1.

Tabel II.1: Studi terhadap Kesalahan pada *Spreadsheet*

Pembuat	**Jumlah yang Diaudit**	**Rata-rata Sel**	**Persentase Error**	**Keterangan Kesalahan**
Butler (1992)	273	-	11%	Kesalahan perhitungan pada pajak
Dent (1994)	Tidak diketahui	-	30%	Menggunakan angka yang ditulis manuals yang mengakibatkan perhitungan berikutnya salah
Hicks (1995)	1	3856	100%	Kesalahan interpretasi pada data
Coopers & Lybrand (1997)	23	150+	91%	Kesalahan perhitungan yang meleset hingga 5%
Lukasic (1998)	2	2270 - 7027	100%	Kesalahan akibat melebih-lebihkan perhitungan hingga 16%
Clermont, Hanin, & Mittermeier (2002)	3	-	100%	Kesalahan akibat perhitungan sel kosong
Lawrence and Lee (2004)	30	2182	100%	Kesalahan perhitungan dan formula
Powell, Lawson, and Baker (2007)	25	-	64%	Kesalahan perhitungan dan formula
Powell, Baker & Lawson (2007)	50	-	86%	Kesalahan perhitungan dan formula

Dari kumpulan data diatas, dapat dilihat bahwa didalam pembentukan *spreadsheet* pada bidang bisnis, tidak mungkin terlepas dari kesalahan. Dengan tingginya tingkat kesalahan ini, bisnis dapat mengalami kerugian secara material maupun moral yang cukup besar (EuSpRIG, 2010a). Hal ini mengindikasikan bahwa tingginya tingkat kesalahan harus dapat diselesaikan agar tidak terjadi kerugian di dalam penggunaan *spreadsheet* terutama dalam bisnis.

II.3.3 Tipe Kesalahan dalam Penggunaan *Spreadsheet*

Tingkat fleksibilitas *spreadsheet* yang tinggi memberikan keleluasaan kepada penggunanya untuk melakukan banyak manipulasi dan pengelolaan data. Tingginya fleksibilitas ini dapat berakibat mudahnya *human error* terjadi pada saat penggunaan *spreadsheet* yang menyebabkan terjadinya kesalahan-kesalahan pada data. Tipe-tipe kesalahan pada *spreadsheet* dapat dibagi menjadi dua jenis tipe kesalahan yakni kesalahan kuantitatif, dan kesalahan kualitatif (Panko, 1998).

II.3.3.1 Kesalahan Kualitatif

Kesalahan kualitatif merupakan kesalahan yang berhubungan dengan kualitas *spreadsheet* tersebut lebih menitikberatkan pada kebiasaan dan prosedur yang salah didalam pembuatan *spreadsheet*. Beberapa kesalahan yang dapat diklasifikasikan sebagai kesalahan kualitatif adalah (Powell et al., 2009):

1. Melakukan *hard-code* pada suatu angka di dalam formula
2. Menggunakan formula yang panjang dalam perhitungan
3. Susunan data yang tidak direncanakan dengan baik
4. Tidak adanya dokumentasi mengenai *spreadsheet* yang dibuat

Kesalahan ini tidak langsung mengakibatkan nilai hasil keluaran yang salah namun menurunkan kualitas dari *spreadsheet* tersebut (Rajalingham et al., 2001). Selain itu, kesalahan kualitatif dapat menyebabkan kesalahan kuantitatif terutama pada saat penggunaan fungsi analisis *what-if* pada *spreadsheet* (Panko, 1998).

II.3.3.2 Kesalahan Kuantitatif

Kesalahan ini mengakibatkan *spreadsheet* mengeluarkan hasil dan nilai yang salah didalam operasi perhitungannya. Kesalahan kuantitatif dapat dibagi menjadi tiga tipe kesalahan yakni (Panko, 1998):

1. Kesalahan mekanikal (*mechanical error*) yang biasanya terjadi akibat kesalahan pengetikan angka atau rujukan sel yang salah pada suatu formula
2. Kesalahan logika (*logical error*) yang terjadi pada pembuatan formula yang salah atau penggunaan fungsi yang tidak tepat
3. Kesalahan akibat kelalaian pada interpretasi situasi atau spesifikasi yang diberikan sehingga *spreadsheet* yang dihasilkan tidak sesuai dengan domain permasalahan yang ada atau *ommision error* (Powell et al., 2009)

II.3.4 Penanganan Kesalahan pada *Spreadsheet*

Untuk mengatasi kesalahan yang dijelaskan pada subbab sebelumnya, menurut penelitian yang dilakukan oleh Panko (Panko, 1998), dijabarkan beberapa metode untuk menangani dan mengurangi kesalahan yang sering terjadi. Beberapa metode yang dapat digunakan yakni:

1. Membangun *preliminary design* sebelum pembuatan *spreadsheet* agar terdapat perencanaan yang baik di dalam pembangunan data di dalam *spreadsheet*
2. Melakukan proteksi terhadap sel yang tidak boleh diubah.
3. Melakukan pengecekan terhadap semua rumus dan formula yang dimasukan bahkan hingga rumus yang cukup sederhana dengan cara melakukan pengecekan manual.
4. Membuat dokumentasi untuk *spreadsheet* yang dibuat.
5. Tidak menekan pembuat *spreadsheet* terhadap kesalahan yang dibuat dengan memberikan hukuman. Kesalahan yang terjadi pada *spreadsheet* umumnya masih berada pada batas normal *human error* sehingga memberikan hukuman akan membuat rasa takut dalam melaporkan kesalahan.

6. Melakukan inspeksi terhadap formula, rumus, dan kode yang dibuat baik oleh individual maupun secara berkelompok.

II.4 Data pada *Spreadsheet*

II.4.1 Tipe Struktur Data pada *Spreadsheet*

Pada penelitian yang dilakukan oleh Chen dan Cafarella, *spreadsheet* dapat dibagi menjadi 2 jenis yakni; *data frame* dan *non-data frame*. *Data frame* merupakan tipe *spreadsheet* yang terdiri dari 2 komponen utama: area nilai dan area atribut atau metadata (biasanya berada di atas dan atau di kiri area nilai). *Non-data frame* adalah tipe *spreadsheet* selain tipe *data frame* yang telah didefinisikan sebelumnya. Tipe *non-data frame* dapat dibagi menjadi beberapa jenis yakni:

1. Relasi merupakan tipe *spreadsheet* yang dapat langsung diubah ke model relasional.
2. Formulir merupakan *spreadsheet* yang tidak ditujukan sebagai penyimpanan dan didesain untuk diisi oleh manusia.
3. Diagram yang digunakan sebagai visualisasi data, biasanya berisi banyak data tanpa skema informasi yang detil.
4. Daftar atau *List* merupakan catatan sejumlah nama atau hal (tentang kata-kata, nama orang, barang, dan sebagainya) yang disusun berderet dari atas ke bawah (dan Pengembangan Bahasa et al., 1991).
5. Jadwal merupakan *spreadsheet* digunakan sebagai pembuatan dan pengelolaan jadwal.
6. Silabus merupakan kerangka unsur kursus pendidikan, disajikan dalam aturan yang logis, atau dalam tingkat kesulitan yang makin meningkat (dan Pengembangan Bahasa et al., 1991).
7. *Scorecard* yakni suatu alat manajemen yang biasanya berguna untuk membantu manajer melacak aktivitas yang dilakukan oleh stafnya.

Penelitian ini menggunakan sampel 200 *spreadsheet* yang dilabeli oleh ahli dan didapatkan bahwa 50.5% *spreadsheet* merupakan tipe *data frame*, dimana 32.5% memiliki label atribut

dibagian atas atau bawah. Sedangkan 49.5% *spreadsheet* bertipe *non-data frame* terdiri dari 22.0% relasi, 10.5% formulir, 3.5% diagram, 3% berupa *list*, dan 10.5% lainnya (Chen and Cafarella, 2013).

II.4.2 Pengolahan Data pada *Spreadsheet*

Extract-Transform-Load (ETL) adalah proses yang digunakan sebagai metode integrasi data dari beberapa sumber dan aplikasi. ETL biasanya digunakan pada saat melakukan proses *data warehouse* dimana data dari sumber eksternal diambil, lalu ditransformasikan ke bentuk yang sesuai dengan kebutuhan (didalam prosesnya bisa terkadung pengecekan kualitas), dan memasukannya ke dalam basisdata yang telah ditentukan (Bansal, 2014). Terdapat tiga fase pada proses ETL yakni:

1. *Extract*, fase pertama ini adalah proses yang melakukan ekstraksi data dari sumber yang dipilih. Data biasanya tersedia didalam format *flat file* seperti csv, xls, dan txt atau melalui klien RESTful.
2. *Transform*, pada fase ini data dibersihkan agar sesuai dengan skema tujuan. Beberapa cara untuk mentransformasikan data adalah dengan menormalisasi data, menghapus duplikasi, melakukan pengecekan terhadap batasan-batasan, melakukan *filtering*, melakukan pengurutan dan pengelompokan, atau fungsi-fungsi lain yang didefinisikan.
3. *Load*, pada fase ini data yang telah ditransformasikan dimasukan ke dalam *data mart* atau *data warehouse* yang ditentukan.

II.4.2.1 Metode ETL pada *Spreadsheet*

Pada penelitian yang dilakukan oleh Chen, pengolahan data pada sebuah *spreadsheet* bertipe *data frame* dapat dibagi menjadi tiga proses yakni: *frame finder*, *hierarchy extractor*, dan *tuple builder*. Proses *frame finder* dilakukan dengan cara mengidentifikasi *data frame* serta mencari lokasi dari atribut dan nilai. Pada proses *hierarchy extractor*, atribut yang ada pada *data frame* yang ditemukan dicari hirarkinya setelah itu proses *tuple builder* membentuk *tuple* relasional untuk setiap nilai yang ada. Proses ini tidak membedakan *spreadsheet* tipe *data frame* atau bukan, sehingga diasumsikan jika *tuple* yang dihasilkan

memiliki kualitas yang baik, dapat dikatakan bahwa *spreadsheet* masukan bertipe *data frame* dan sebaliknya. (Chen and Cafarella, 2013)

II.4.2.1.1 *Frame Finder*

Tujuan dari proses *frame finder* adalah mengidentifikasi wilayah nilai dan wilayah atribut yang dapat berupa *left attribute* maupun *top attribute*. Untuk mensimplifikasi permasalahan, Chen menganggap bahwa *data frame* tidak akan berada sejajar secara horisontal, namun hanya secara vertikal. Sehingga proses ini dapat disimplifikasi menjadi labeling terhadap baris per baris. Label yang akan diberikan adalah *title*, *header*, *data*, dan *footnote*.

Pelabelan dapat dilakukan dengan algoritma *conditional random field* (CRF) karena terdapat keterkaitan antara satu baris terhadap baris yang lain didalam penggunaan baris. Contohnya, jika baris telah teridentifikasi sebagai *header*, maka besar kemungkinan bahwa baris selanjutnya adalah *data* atau *header*. CRF memiliki kemampuan untuk melakukan *machine learning* yang memperhitungkan label pada elemen sebelumnya.

II.4.2.1.2 *Hierarcy Extractor*

Proses ini bertujuan untuk mendapatkan hirarki dari atribut-atribut yang ada. Masukan dari proses ini adalah *data frame* dengan *top attribute* dan *left attribute* dan keluarannya berupa hirarki untuk masing-masing atribut atas dan kiri tersebut. Proses ini dapat dilakukan melalui dua algoritma: *classification* dan *enforced-tree classification*.

Classification dilakukan dengan cara berbeda untuk *left attribute* dan *top attribute*. Pada *left attribute*, klasifikasi dapat dilakukan dengan dua cara: pengecekan terhadap *formatting* pada sebuah sel dan kedekatan sel secara geometris. Semakin mirip *formatting* sebuah sel, maka semakin mungkin bahwa kedua sel bukan merupakan pasangan *parent* dan *child* dan semakin dekat sel secara geometris, kemungkinan kedua sel meruapakan pasangan *parent* dan *child* semakin besar. Sedangkan pada *top attribute* dapat dilakukan pengecekan posisi antar baris atribut bagian atas.

Kelemahan pada metode klasifikasi sebelumnya adalah terdapat kemungkinan tidak dapat dibentuk pohon dari hasil klasifikasi. *Enforced-tree classification* mencoba untuk

menyelesaikan permasalahan ini dengan dua langkah tambahan yakni: memastikan bahwa suatu atribut hanya dapat memiliki satu *parent* dimana yang terpilih menjadi *parent* adalah atribut dengan probabilitas tertinggi, dan memastikan bahwa tidak ada *cycle* yang terbentuk dengan cara menghapus keterhubungan dengan nilai probabilitas terkecil. Klasifikasi ini tetap menggunakan metode klasifikasi fitur yang dilakukan pada algoritma *classification*.

II.4.2.1.3 *Tuple Builder*

Proses ini dilakukan dengan cara mengiterasi setiap *value (v)* dan mencari atribut akar pada atribut bagian kiri dan atas dari *v*. Setelah dibentuk *relational tuple* untuk nilai *v* dengan atribut bagian kiri dan atas tersebut. Tingkat akurasi dari proses ini sangat bergantung dari dua proses sebelumnya.

II.5 Studi dan Penelitian Terkait

II.5.1 *Senbazuru: A Prototype Spreadsheet Database Management System*

Senbazuru (Chen et al., 2013) merupakan prototipe yang dikembangkan dengan tujuan untuk mempermudah pencarian, pengaksesan, pengubahan, dan melakukan *query* terhadap *spreadsheet*. Pengembangan ini ingin menyelesaikan permasalahan dimana data *spreadsheet* sangat tersebar diberbagai tempat sehingga untuk mendapatkan informasi yang diinginkan atau membandingkan antar informasi di dalam beberapa *spreadsheet* sangatlah sulit.

Didalam pengembangan prototipe ini, kesulitan teknikal yang harus dihadapi adalah proses ekstraksi dan perbaikan data. Didalam melakukan ekstraksi data, harus dilakukan beberapa proses berikut: mendeteksi mana atribut dan nilai, mengidentifikasi hirarki atribut, membentuk *relational tuple*, dan membentuk *tuple* tersebut menjadi tabel relasional. Dari hasil dari ekstrasi ini, masih sangat mungkin memiliki kesalahan sehingga proses perbaikan diperlukan didalam pembentukan tabel relasional ini. Proses perbaikan pada protitipe ini dilakukan manual dengan bantuan pengguna.

II.5.1.1 Arsitektur Sistem

Spreadsheet Database Management System (SSDBMS) yang dikembangkan pada prototipe ini memiliki tiga proses utama yakni: *search*, *extract*, dan *query*. Proses pencarian dilakukan terhadap repositori *spreadsheet* yang ada di internet, lalu data *spreadsheet* tersebut diekstraksi dan dijadikan tabel relasional, dan setelah itu pengguna dapat melakukan *query* terhadap tabel yang telah dibentuk.

II.5.1.1.1 *Search*

Komponen pencarian ini memudahkan pengguna untuk menemukan dataset yang tepat menggunakan bantuan internet. Saat prototipe ini dikembangkan, Senbazuru telah mengindeks 1800 *spreadsheet* yang didapatkan dari U.S. Census Bureau. Pengindeksan menggunakan bantuan *library* Python yakni xlrd untuk mengekstraksi teks dari sel lalu menggunakan Apache Lucene untuk melakukan indeks pada teks. Pencarian menggunakan metode *term frequency–inverse document frequency* (TF-IDF) untuk mendapatkan relevansi dokumen.

II.5.1.1.2 *Extract*

Proses ekstraksi data pada *spreadsheet* dilakukan melalui empat tahapan yakni:

1. *Frame Finder*

   Tahap ini dilakukan untuk mencari *frame* pada *spreadsheet* bertipe *data frame*. Dengan menggunakan algoritma *conditional random field* (CRF) untuk memberikan label pada setiap baris yang tidak kosong pada *spreadsheet*. Tahap ini akan menghasilkan *data frame* yang selanjutnya akan digunakan pada tahap selanjutnya, baris lain yang dianggap bukan *data frame* akan diabaikan.

2. *Hierarchy Extractor*

   Tahap selanjutnya adalah ekstraksi hirarki pada wilayah atribut dari *data frame* yang ditemukan. Pada setiap atribut, akan dicari atribut mana yang dideskripsikan oleh atribut lainnya dan seterusnya hingga terbentuk hirarki dari atribut-atribut yang ada. Kesalahan pada pembentukan hirarki sangat mungkin terjadi sehingga

pengguna akan diberikan kemampuan untuk memperbaiki hirarki yang salah pada bagian *repair interface*. Setelah perbaikan dilakukan oleh pengguna, Senbazuru akan menjalankan kembali CRF untuk melakukan pembelajaran terhadap label baru yang diberikan.

3. *Tuple Builder*

   Bagian ini melakukan pembentukan *tuple* antara wilayah nilai dan wilayah atribut yang sesuai.

4. *Relation Constructor*

   Tahap ini melakukan transalasi dari *tuple* yang terbentuk menjadi tabel relasional dengan cara membentuk kluster terhadap atribut yang satu jenis. Contohnya, terdapat atribut *Male*, *total*, dan *Female*, ketiga atribut tersebut memiliki jenis yang sama sehingga harus digabungkan menjadi satu kolom yakni *gender*. Pada Senbazuru, teknik pengklusteran ini menggunakan bantuan koleksi skema dari Freebase dan YAGO.

II.5.1.1.3 *Query*

Setelah proses sebelumnya selesai, maka pengguna dapat memasukan perintah relasional terhadap data *spreadsheet* yang telah diubah menjadi tabel relasional. Pada prototipe yang dikembangkan, perintah yang diimplementasikan adalah *join* dan *select*.

II.5.1.2 Kesimpulan Penelitian

Senbazuru merupakan prototipe untuk manajemen basis data berbasis *spreadsheet* yang dapat melakukan pencarian data pada internet melalui kata kunci yang diberikan. Prototipe ini berhasil melakukan ekstraksi data secara otomatis walaupun tidak terlepas dari kesalahan. Kesalahan yang terjadi masih harus seringkali dilakukan perbaikan secara manual. Namun dengan penggunaan algoritma CRF, protitipe dapat mengurangi kesalahan yang terjadi. Prototipe ini juga ditujukan sebagai demo kepada peserta konferensi VLDB dan diharapkan dapat menarik perhatian komunitas basis data.

II.5.2 *Spreadsheet As a Relational Database Engine*

Penelitian (Tyszkiewicz, 2010) pernah dilakukan terhadap pembuatan *spreadsheet* menjadi mesin basis data relasional. Penelitian ini dilatarbelakangi dengan tingginya penggunaan *spreadsheet* pada banyak bidang dan kurangnya kualitas data yang ada didalamnya yang dapat menyebabkan kesalahan-kesalahan terjadi pada perhitungan dan prediksi. Solusi yang dipaparkan pada penelitian ini adalah dengan menggabungkan *spreadsheet* dan *database engine* dengan menggunakan formula sebagai ganti dari *SQL query*.

II.5.2.1 Arsitektur Sistem

Pada implementasinya, sebuah *workbook* akan memiliki satu *worksheet* untuk setiap tabel data dan satu *worksheet* untuk setiap *view* pada basis data. Dengan menggunakan *external compiler* yang menerima masukan berupa SQL yang akan mengubahnya kedalam bentuk *spreadsheet*. Program tersebut akan mengubah SQL menjadi beberapa formula yang diterima oleh *spreadsheet* tersebut.

Terdapat dua bagian utama pada *spreadsheet* hasil implementasi yakni: tabel data dan bagian *view*. Bagian tabel data adalah tempat pengguna untuk memasukkan, mengubah, serta menghapus data. Secara teori, bagian tabel data tidak memiliki formula, namun didalam implementasinya mengikuti implementasi SQL dimana perlu dilakukan validasi data dan verifikasi terhadap *primary key*, *foreign key*, dan batasan lain yang ada diperintah *create table*.

Bagian *view worksheet* tidak dapat diubah oleh pengguna dan berisikan formula-formula yang independen terhadap data yang dimasukkan oleh pengguna. Selain itu, bagian *view* berisikan kolom-kolom yang berguna sebagai *intermediate result* yang selanjutnya akan digunakan oleh formula lain. Pada awalnya sel-sel akan berisikan "" yang merepresentasikan sel yang belum digunakan.

II.5.2.2 Kesimpulan Penelitian

Excel dan *spreadsheet* lain tidak didesain untuk menjadi *database engine* sehingga kebanyakan formula akan dilakukan menggunakan *linear scan* dan hal ini dapat mengurangi

performansi. Beberapa cara untuk meningkatkan performansi adalah mengeksploitasi *lazy evaluation* dari *if statement*, mengkomputasi hanya beberapa sel tetangga yang berkaitan, serta menggunakan file atau *workbook* lain untuk membagi *query* dan membangkitkannya ketika dibutuhkan.

Pada tes perfomansi yang dilakukan pada penelitian ini dapat disimpulkan bahwa untuk operasi dasar dan *query* sederhana penggunaan *spreadsheet* dapat digunakan dengan baik dengan waktu yang cukup cepat. Tingkat efektifitas penggunaan pada arsitektur ini cukup rendah, namun hasil tes tersebut menunjukan bahwa arsitektur ini memiliki potensial yang dapat dikembangkan.

BAB III

ANALISIS MASALAH PENGUMPULAN DATA PADA *SPREADSHEET*

Pada bab ini diuraikan analisis persoalan pengumpulan data pada *spreadsheet* yang telah diuraikan pada Bab I. Hasil dari bab ini digunakan untuk merancang aplikasi yang akan diimplementasikan seperti yang dijelaskan pada Bab IV.

III.1 Permasalahan Pengumpulan Data Pada *Spreadsheet*

Pada Subbab I.2 telah dijelaskan bahwa terdapat tiga permasalahan yang muncul didalam pengumpulan data menggunakan media *spreadsheet* yakni:

1. Kolaborasi

   Didalam pembuatan suatu *spreadsheet*, kadang dibutuhkan banyak pihak untuk dapat melengkapi seluruh data yang dibutuhkan. Hal ini membuat pengembangan dan pengumpulan data membutuhkan kontribusi banyak orang (Panko, 1998). Untuk itu dibutuhkan fitur kolaborasi pada aplikasi *spreadsheet* agar banyak pihak dapat melakukan pengumpulan data dan pengeditan secara bersamaan.

2. Validasi

   Data yang dikumpulkan pada suatu *spreadsheet*, sangat rentan akan kesalahan. Bahkan pada *spreadsheet* yang dibuat dengan sangat hati-hati, masih dapat ditemui sekitar 1 persen atau lebih kesalahan pada formula yang dibuat (Panko, 1998). Sehingga dibutuhkan mekanisme tambahan pada saat pengumpulan data untuk melakukan validasi terhadap masukan.

3. Penyimpanan Data

   Setelah data dikumpulkan, permasalahan selanjutnya yang muncul adalah penyimpanan data tersebut. Data biasanya akan tetap tersimpan dalam bentuk berkas *spreadsheet*, namun bentuk ini sulit untuk digunakan oleh aplikasi lain. Bentuk yang

biasanya digunakan oleh aplikasi lain untuk dapat mengambil dan menyimpan data adalah bentuk relasional. Sehingga dibutuhkan mekanisme penyimpanan data ke dalam bentuk relasional agar data dapat dengan mudah digunakan pada aplikasi lain.

Ketiga permasalahan tersebut yang dijadikan dasar pengembangan aplikasi pada Tugas Akhir ini dan akan dibahas pada subbab-subbab berikutnya.

III.2 Analisis Rancangan Aplikasi *Spreadsheet*

Dari permasalahan yang telah didefinisikan pada subbab III.1, dapat ditentukan beberapa hal yang harus dianalisis dalam merancang aplikasi *spreadsheet* ini, diantaranya adalah:

1. Penentuan aplikasi *spreadsheet* yang akan dikembangkan dan ditambahkan fiturnya. Aplikasi harus memiliki fitur kolaborasi sehingga banyak pengguna dapat mengakses suatu *spreadsheet* secara bersamaan.
2. Pemilihan metode masukan aplikasi *spreadsheet* dimana pengguna dapat membentuk struktur tempat pengguna lain memasukkan masukan. Contohnya dalam bentuk tabel atau formulir. Pengguna juga melengkapi domain dan batasan data yang dimasukkan.
3. Setelah pengguna selesai merancang struktur masukan, diperlukan penentuan bagian label dan data yang berkaitan dengan label tersebut. Bagian label dan data akan dijadikan menjadi bentuk relasional yang dapat disimpan dalam basis data.
4. Setelah label dan data ditentukan, tabel harus direpresentasikan dalam bentuk relasional sehingga dapat dimasukkan ke dalam basis data.
5. Pengguna lain yang hanya memiliki kapabilitas pengisian data, hanya dapat mengisi data sesuai struktur yang diberikan. Sehingga perlu melakukan validasi data dan dicek kesesuaiannya sesuai dengan batasan yang diberikan sehingga memenuhi domain yang ditentukan pada basis data.
6. Setelah proses validasi terpenuhi, diperlukan cara penyimpanan data sesuai dengan relasi *tuple* yang telah ditentukan. Pada saat pembukaan *file spreadsheet* tersebut,

akan dilakukan pemulihan data yang telah disimpan pada basis data untuk ditampilkan kembali.

Pada subbab-subbab selanjutnya, akan dibahas analisis untuk kelima poin diatas yang selanjutnya akan dijadikan dasar rancangan fitur yang akan dibuat pada aplikasi *spreadsheet*.

III.3 Analisis Perbandingan Aplikasi *Spreadsheet* yang Dikembangkan

Dari studi yang telah dilaksanakan pada subbab II.1.2, aplikasi *spreadsheet* dapat dibagi berdasarkan konektivitasnya yakni *offline spreadsheet* dan *online spreadsheet* dan dapat juga dibagi lagi berdasarkan keterbukaan dari *source code* yakni *open source* dan *closed source*. Pada Tabel III.1 dijabarkan parameter yang mendasari pemilihan aplikasi yang akan dikembangkan.

Tabel III.1: Perbandingan Aplikasi *Spreadsheet*

Parameter	**Offline Application**	**Web/Online Application**
Aksesibilitas	Dibutuhkan instalasi aplikasi / plugin yang bersangkutan.	Dapat menggunakan web browser yang tersedia.
Kolaborasi	Tidak tersedia secara online dan kolaboratif secara *default*.	Secara *default*, dapat diakses konkuren dan kolaboratif.
Fitur	Sudah banyak jenis formula yang didukung.	Tidak semua formula yang ada didukung.

Dari perbandingan diatas, diambil solusi dengan menggunakan aplikasi *spreadsheet* berbasis web yang dapat diakses secara konkuren dan memiliki *portability* yang lebih baik. Selain itu, untuk dapat mengembangkan fitur aplikasi dengan lebih baik, akan dipilih aplikasi yang berbentuk *open source*. Sehingga dipilih aplikasi yang memiliki tipe *online spreadsheet* dan *open source* yaitu EtherCalc. Fitur kolaborasi yang dibutuhkan akan ditangani oleh aplikasi EtherCalc ini dan bukan merupakan bagian dari fitur yang akan dikembangkan.

III.4 Analisis Metode Masukan Pengguna

Di dalam pembangunan perangkat lunak *spreadsheet* untuk mengurangi kesalahan, dapat diidentifikasikan dua metode masukan yang dapat diimplementasi. Model interaksi yang pertama adalah menggunakan formulir sebagai basis masukan data dan metode yang kedua adalah menggunakan aplikasi *spreadsheet* secara langsung sebagai media input data.

III.4.1 Berbasis Formulir

Metode masukan ini menggunakan *spreadsheet* sebagai tempat perancangan formulir. Pembuat formulir akan menentukan area label dan input secara manual serta diidentifikasikan berdasarkan warna atau properti lain yang unik pada sel tersebut. Formulir akan dibangkitkan oleh aplikasi agar menjadi bentuk HTML dan terhubung ke basis data. Pengisian data oleh pengguna dilakukan melalui formulir yang dibangkitkan dan dapat diakses melalui web. Beberapa cara dapat dilakukan untuk mengimplementasikan teknik ini antara lain, mengembangkan dari aplikasi *spreadsheet* yang telah ada menggunakan *plugin* atau membuat aplikasi baru yang dapat melakukan konversi otomatis menjadi formulir.

III.4.2 Berbasis *Spreadsheet*

Metode masukan berbasiskan *spreadsheet* menggunakan antarmuka yang telah disediakan oleh aplikasi. Penggunaannya dilakukan dengan membuat format pengisian seperti membuat tabel pada *spreadsheet* pada umumnya. Pada tabel harus terdapat label dan data sehingga metadata minimal yang dibutuhkan dapat dicapai. Fitur-fitur yang ada pada *spreadsheet* juga tetap dapat digunakan saat pembuatan tabel yang diinginkan. Dari pembuatan tabel tersebut dilakukan identifikasi label dari data tersebut untuk selanjutnya diproses melalui penyaringan masukan dan dimasukan ke tabel yang sesuai yang ada di basis data. Untuk mengimplementasikan teknik ini harus mengubah kode pada program *spreadsheet* atau mengekstensi fitur yang ada menggunakan *plugin*.

III.4.3 Perbandingan Metode Masukan

Kedua metode masukan pengguna tersebut memiliki beberapa perbedaan dan efek terhadap penggunaannya baik bagi sistem maupun pengguna. Pada Tabel III.2 dijabarkan perbandingan antara kedua metode masukan pengguna tersebut.

Tabel III.2: Perbandingan Metode Masukan Pengguna

Parameter	**Berbasis Formulir**	**Berbasis *Spreadsheet***
Fungsionalitas	Berhasil memisahkan bagian operasional dan data. Pengguna hanya memodifikasi dan melakukan input pada bagian data. Bagian operasional hanya dapat dimodifikasi oleh pembuat formulir tersebut.	Jika tidak menggunakan proteksi terhadap sel yang dapat ditulis, tidak terjadi pemisahan antara data dan operasional sehingga beberapa kesalahan yang terjadi pada saat input masih dapat terjadi.
Teknologi	Dibutuhkan adanya algoritma tambahan untuk menangani formulir dan masukannya, serta melakukan konversi dan *parsing* dari sel pada *spreadsheet* ke dalam bentuk formulir.	Dibutuhkan algoritma dan logika *parsing* yang lebih detil dan kompleks dalam menangani kasus-kasus table tertentu.
Antarmuka	Menggunakan antarmuka formulir yang kaku terurut dari atas ke bawah serta tidak dapat melihat hasil masukan secara langsung.	Struktur tabel atau masukan dapat disesuaikan dengan kebutuhan dan tidak perlu mempelajari hal lain jika sebelumnya telah menggunakan *spreadsheet* sebagai media untuk memasukan data.

Pada Tugas Akhir ini, akan dipilih metode masukan pengguna dengan berbasis *spreadsheet* sehingga pengguna dapat dengan lebih mudah didalam menggunakannya karena tidak memerlukan pembelajaran kembali di dalam menggunakan aplikasi. Selain itu, data yang

dimasukan lebih mudah untuk dilihat secara menyeluruh dibandingkan dengan berbasis formulir yang hanya dapat menerima satu masukan dalam suatu formulir.

III.5 Analisis Penentuan Bagian Data dan Label

Pada pembuatan *spreadsheet* pada umumnya, didapatkan bahwa kebanyakan *spreadsheet* pada umumnya berbentuk tabel yang terdiri dari dua unsur utama yakni data dan label atau disebut juga tipe *data frame* (Chen and Cafarella, 2013). Bagian data merupakan bagian yang biasanya dinamis dan merupakan masukan pengguna. Bagian label merupakan bagian yang memberikan keterangan dan konteks mengenai data yang dimaksud. Pada Gambar III.1 dapat dilihat bahwa area dengan nomor 1 disebut sebagai label yang menjelaskan data-data dibawahnya yakni pada area nomor 2.

Angkatan	IF	STI	
2008	111	31	1
2009	99	39	
2010	109	28	
2011	99	60	
2012	101	37	2
2013	99	46	
2014	111	55	
2015	148	47	

Gambar III.1: Contoh *Data Frame* Sederhana

III.5.1 Secara Manual

Penentuan label dan data dilakukan oleh pengguna secara langsung saat pembuatan tabel. Pengguna sendiri yang menentukan area mana yang merupakan label dan data mana yang dijelaskan oleh label tersebut. Dengan metode manual ini, pengguna dapat menyesuaikan bentuk tabel sesuai keinginan mereka. Metode ini menyerahkan sepenuhnya tanggungjawab keterhubungan sel label dan sel data kepada pengguna. Metode yang digunakan adalah menerima masukan dari pengguna berupa suatu *script* yang mendefinisikan letak tabel, *header*, dan data pada *spreadsheet*.

III.5.2 Secara Otomatis

Pencarian label dan data dapat menggunakan algoritma seperti yang telah dijelaskan pada Subbab II.4.2.1. Mekanisme untuk mengidentifikasi label dan data dapat dilakukan melalui

3 tahapan yakni, *frame finder*, *hierarchy extractor*, dan *tuple builder*. Pada tahap pertama yakni *frame finder* bertujuan untuk mengidentifikasi wilayah nilai (data) dan wilayah atribut (label) yang dapat berupa *left attribute* maupun *top attribute*. Tahap selanjutnya adalah *hierarchy extractor* bertujuan untuk mendapatkan hirarki dari atribut-atribut yang ada sehingga label yang tertulis dapat dicari keterhubungan dan konteksnya terhadap data yang ada. Tahap terakhir adalah *tuple builder* yang mentrasformasikan data dan label tersebut dalam bentuk *tuple* yang dapat diterima oleh basis data relasional.

III.5.3 Perbandingan Metode Penentuan

Untuk mengetahui perbandingan kedua metode, pada Tabel III.3 dijelaskan kelebihan dan kekurangan dua metode yang telah disebutkan sebelumnya.

Tabel III.3: Perbandingan Metode Penentuan Data dan Label

Keterangan	**Manual oleh Pengguna**	**Otomatis**
Kelebihan	Tingkat akurasi lebih tinggi karena data dan label yang ditentukan sesuai dengan keinginan pengguna.	Kenyamanan dalam penggunaan karena pengguna tidak perlu melakukan interferensi tambahan. Selain itu, sistem kemungkinan data dan label yang diambil dapat diubah ke dalam bentuk *tuple* relasional lebih tinggi.
Kekurangan	Interferensi pengguna yang banyak dan mungkinnya data dan label tidak dapat dijadikan bentuk *tuple* relasional.	Algoritma ini hanya optimal jika digunakan pada tabel yang terurut vertikal.

Dari perbandingan diatas, dapat dilihat terdapat kekurangan dan kelebihan didalam menggunakan salah satu metode tersebut. Berdasarkan hal tersebut, yang akan dipilih sebagai metode penentuan data dan label adalah gabungan dari kedua metode tersebut. Sistem awalnya akan melakukan *parsing* terhadap struktur yang telah dibuat oleh pengguna, kemudian pengguna dapat melihat hasil dari *parsing* tersebut sehingga pengguna dapat

memperbaiki jika terdapat hasil pelabelan yang salah. Dengan metode gabungan ini diharapkan dapat meningkatkan akurasi dan mengurangi kesalahan *parsing* yang terjadi namun tetap memberikan kenyamanan terhadap pengguna serta menjaga agar hasil pelabelan tetap dapat diubah ke dalam bentuk *tuple* relasional.

III.6 Analisis Representasi Tabel pada *Spreadsheet*

Didalam pembuatan *spreadsheet*, terdapat beberapa jenis tabel yang biasanya dibuat oleh pengguna dalam merepresentasikan data yang ingin disimpan. Pada jenis-jenis tabel tertentu, dibutuhkan transformasi bentuk tabel tersebut agar dapat dimasukkan ke dalam basis data bentuk relasional. Tipe-tipe tabel yang biasanya muncul di dalam pembuatan *spreadsheet* beserta teknik transformasi yang akan digunakan adalah sebagai berikut:

1. Tabel dengan *header* berada diatas data

NIM	Nilai 1	Nilai 2	UTS	UAS	Nilai Akhir	Nilai
13513001	90	88	76	80	82.18181818	A
13513002	50	60	82	67	66.72727273	BC
13513003	0	56	77	65	54.81818182	C
13513004	35	55	50	70	55.45454545	C
13513005	90	100	100	110	101.8181818	A

Gambar III.2: Tabel dengan *Header* Berada Diatas Data

   Representasi tabel jenis ini dapat langsung dibuat ke dalam bentuk relasional tanpa harus mengubah struktur tabel. Contoh tabel jenis ini dapat dilihat pada Gambar III.2. Bagian *header* merupakan baris pertama pada tabel, dan baris kedua hingga seterusnya merupakan bagian data. Bentuk ini tidak membutuhkan transformasi agar dapat dimasukkan ke dalam basis data bentuk relasional.

2. Tabel dengan *header* berada di kiri dan atas data

	O2	N2	CO2	Ar	H20
Kadar di permukaan laut	21%	78%	0.04%	0.93%	1%
Kadar di bawah laut	0.04%	0%	0%	0%	100%
Kadar di ketinggian 10 km	10%	80%	1%	1%	0.80%
Kadar di stratosphere	5%	90%	1%	1%	0.40%

Gambar III.3: Tabel dengan *Header* Berada di Kiri dan Atas Data

   Tabel dapat memiliki lebih dari satu *header* di lokasi yang berbeda. Contoh tabel

jenis ini dapat dilihat pada Gambar III.3. Representasi tabel dimana terdapat dua bagian *header* atau lebih akan dianggap tetap memiliki satu *header*. Bagian yang akan dianggap sebagai *header* adalah baris yang keseluruhannya berisi *header* dan berada diatas data. Pada contoh diatas, *header* adalah baris pertama. Sedangkan *header* di kiri tabel yang satu baris dengan data akan dianggap data dengan nama *header* yang dapat ditentukan oleh pengguna. Dengan demikian, representasi tabel tersebut akan menjadi mirip dengan tipe tabel sebelumnya. Representasi tabel setelah transformasi dapat dilihat pada Gambar III.4.

Letak pengukuran	O2	N2	CO2	Ar	H20
Kadar di permukaan laut	21%	78%	0.04%	0.93%	1%
Kadar di bawah laut	0.04%	0%	0%	0%	100%
Kadar di ketinggian 10 km	10%	80%	1%	1%	0.80%
Kadar di stratosphere	5%	90%	1%	1%	0.40%

Gambar III.4: Transformasi Tabel dengan *Header* Berada di Kiri dan Atas Data

3. Tabel dengan *header* yang berhirarki

Record Number	Client ID	Sales			
		2011	2012	2013	2014
1	ID000123	802938	123513	625430	234500
2	ID000125	234324	342000	342600	324200
3	ID000222	237899	234520	245100	235000
4	ID000245	345690	2347	34528	34526

Gambar III.5: Tabel dengan *Header* yang Berhirarki

Tabel dapat memiliki *header* yang berhubungan satu dengan yang lain dan membentuk hirarki seperti pada contoh Gambar III.5. Tabel dengan *header* yang memiliki hirarki akan tetap dianggap memiliki satu baris *header*. Transformasi yang dilakukan adalah dengan menggabungkan kata-kata pada tiap *header* dan dijadikan sebagai satu nama *header*. Hasil transformasi tabel tersebut dapat dilihat pada Gambar III.6.

4. Tabel dengan data yang digabung

   Tabel dapat juga memiliki penggabungan sel pada bagian data seperti yang dapat dilihat pada contoh Gambar III.7. Penggabungan sel ini biasanya digunakan sebagai

Record	Client ID	Sales 2011	Sales 2012	Sales 2013	Sales 2014
1	ID000123	802938	123513	625430	234500
2	ID000125	234324	342000	342600	324200
3	ID000222	237899	234520	245100	235000
4	ID000245	345690	2347	34528	34526

Gambar III.6: Transformasi Tabel dengan *Header* yang Berhirarki

Kelompok	NIM	Nilai			
		Laporan	Program	Demo	Bonus
CIMOL TEAM	13042	20%	80%	0%	5%
	13022	30%	20%	45%	5%
	13056	50%	0%	55%	5%
Suatu Tim	13099	33%	25%	50%	0%
	13600	33%	25%	50%	0%
	13400	33%	50%	0%	0%

Gambar III.7: Tabel dengan Data yang Digabung

penanda bahwa data pada setiap baris atau kolom yang digabung adalah sama. Sehingga, transformasi yang akan dilakukan adalah dengan memecah sel yang digabungkan tersebut dan mengisi setiap sel dengan konten yang sama dengan sel yang digabungkan. Hasil transformasi tabel dapat dilihat pada Gambar III.8.

Kelompok	NIM	Nilai Laporan	Nilai Program	Nilai Demo	Nilai Bonus
CIMOL TEAM	13042	20%	80%	0%	5%
CIMOL TEAM	13022	30%	20%	45%	5%
CIMOL TEAM	13056	50%	0%	55%	5%
Suatu Tim	13099	33%	25%	50%	0%
Suatu Tim	13600	33%	25%	50%	0%
Suatu Tim	13400	33%	50%	0%	0%

Gambar III.8: Transformasi Tabel dengan Data yang Digabung

Setelah ditransformasikan, tabel akan dapat dengan mudah dijadikan *tuple* relasional sehingga dapat langsung dimasukkan ke dalam basis data relasional yang dipilih.

III.7 Analisis Validasi Data

Setelah pengguna memasukan datanya kedalam struktur yang telah ditetapkan, pengecekan data dilakukan. Tujuan dari mekanisme ini adalah untuk menghindari kesalahan

masukan yang terjadi dan menyesuaikan tipe yang diterima oleh tabel pada basis data. Terdapat tiga hal utama yang divalidasi pada aplikasi ini, yakni:

1. Validasi tipe data. Contohnya, data masukan yang seharusnya berupa *integer* tidak dapat menerima masukan berupa *string*.
2. Validasi domain masukan. Pengguna dapat juga mengatur rentang *integer* yang boleh dijadikan masukan atau *string* apa saja yang dapat diterima oleh sistem.
3. Validasi relasi data. Data masukan dapat berupa nilai yang harus ada pada tabel lain, contohnya terdapat 2 tabel yakni nilai mahasiswa dan identitas mahasiswa. Tabel nilai mahasiswa memiliki kolom NIM yang nilainya harus ada pada tabel identitas mahasiswa. Pada *spreadsheet* biasanya hal ini diatasi dengan menggunakan perintah VLOOKUP.

Sistem akan melakukan validasi terhadap ketiga hal tersebut dan menolak data masukan sehingga pengguna dapat memperbaiki data.

III.8 Penyimpanan Data

Terdapat dua permasalahan pada penyimpanan data yang perlu dianalisis yakni, jenis penyimpanan data yang digunakan dan alur pengguna melakukan penyimpanan data tersebut. Kedua masalah ini akan dijelaskan pada dua subbab berikut.

III.8.1 Jenis Penyimpanan Data

Data yang telah dimasukan oleh pengguna baik data bentuk struktur *spreadsheet* maupun data masukan pada struktur disimpan ke dalam basis data yang presisten. Terdapat 2 jenis data yang harus disimpan ke dalam basis data di dalam membangun aplikasi *spreadsheet* ini, yakni:

1. Penyimpanan *File Spreadsheet*

   File spreadsheet yang dimaksud adalah data struktural seperti *value* pada suatu sel, *properties* suatu sel seperti warna, *border*, dan *alignment*, serta hal-hal lain yang berhubungan dengan bagaimana suatu *file spreadsheet* ditampilkan pada aplikasi. Tipe penyimpanan yang dapat digunakan untuk menampung data ini adalah NoSQL

karena tipe ini cocok untuk menangani data yang kurang terstruktur seperti *file spreadsheet* yang struktur penempatan datanya berbeda tiap pengguna. Cara penyimpanan yang cocok untuk menangani struktur ini adalah *document based* atau *key-value*. Sehingga proses penyimpanan *file spreadsheet* akan menggunakan mekanisme yang sudah disediakan oleh aplikasi EtherCalc dengan menggunakan redis.

2. Penyimpanan Data dan Label

   Data yang disimpan merupakan hasil dari pendeteksian label dan data yang akan diubah menjadi *tuple* relasional yang dapat diterima oleh basis data relasional. Contoh pengubahan yang terjadi dapat dilihat pada Gambar III.9.

Angkatan	IF	STI
2008	111	31
2009	99	39
2010	109	28
2011	99	60
2012	101	37
2013	99	46
2014	111	55
2015	148	47

Relational Tuple:	IF	Angkatan	2008	111

Gambar III.9: Contoh *Tuple* Relasional

Setelah data dan label dijadikan *tuple* relasional, *tuple* dimasukan ke dalam basis data dengan menggunakan operasi *insert* ke dalam tabel pada basis data. Tipe penyimpanan yang cocok digunakan adalah basis data relasional (SQL) karena data yang dibuat dalam bentuk tabel pada umumnya dapat dikonversikan menjadi *tuple* relasional.

III.8.2 Alur Penyimpanan Data

Dengan adanya dua jenis penyimpanan yang digunakan seperti telah dijelaskan pada Subbab III.8.1 terdapat dua kemungkinan alur penyimpanan data, yakni:

1. Penyimpanan ke basis data dilakukan setiap mendapatkan masukan pengguna
2. Penyimpanan ke basis data menunggu perintah dari pengguna

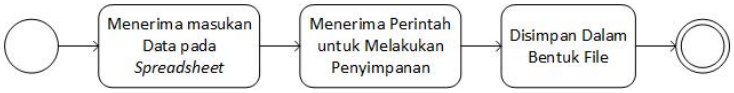
Jika menggunakan alur penyimpanan basis data yang dilakukan setiap mendapatkan masukan, terdapat beberapa kelebihan yakni sinkronisasi antara basis data yang menyimpan *file spreadsheet* yakni redis dan basis data yang menyimpan data yakni basis data relasional yang dipilih. Kedua basis data ini akan memiliki data yang sama setiap waktu. Namun kelemahan alur ini adalah performansi yang akan menurun diakibatkan sistem pendeteksian label dan data yang harus dilakukan setiap saat menerima masukan. Kelemahan lainnya adalah akan sulit untuk mengembangkan aplikasi menggunakan *user access control* karena sulit menentukan kapan dan oleh siapa data boleh dimasukkan ke dalam basis data.

Menggunakan alur penyimpanan yang menunggu perintah dari pengguna, memiliki kelebihan performansi yang lebih baik dibandingkan alur sebelumnya karena penyimpanan dilakukan *on demand*. Selain itu, jika akan mengembangkan aplikasi dengan *user access control* akan lebih mudah untuk menentukan siapa yang memiliki akses untuk melakukan penyimpanan. Kelemahan dari alur ini adalah tidak sinkronnya kedua basis data.

Dari pertimbangan tersebut, alur yang akan digunakan pada aplikasi ini adalah penyimpanan ke basis data yang menunggu perintah dari pengguna karena kelebihannya pada sisi performa dan *user access control*. Selain itu, tidak sinkronnya antara kedua basis data bukan merupakan masalah karena data yang ingin dikumpulkan merupakan data akhir dan data tiap perubahan yang dilakukan tidak begitu dipedulikan.

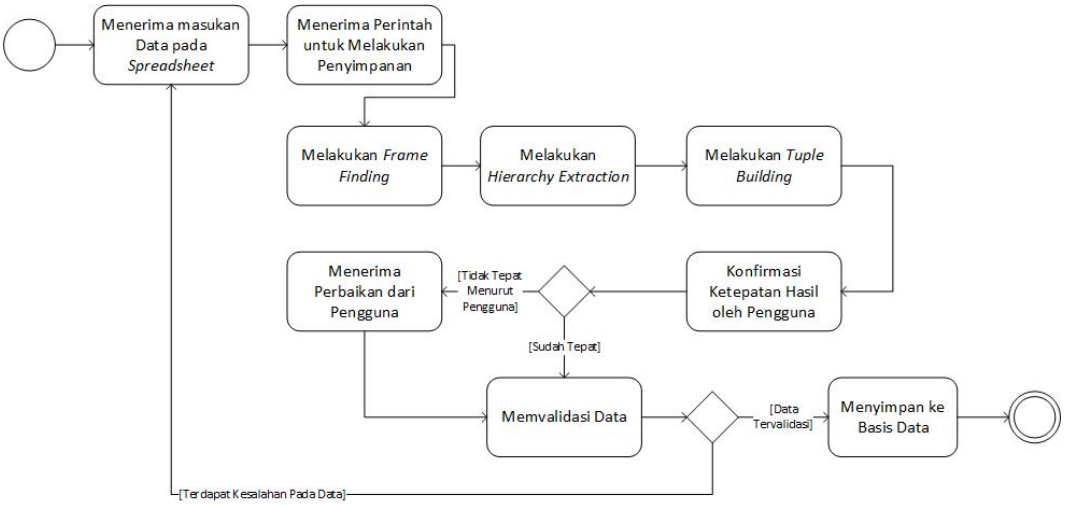
III.9 Analisis Alur Aplikasi

Aplikasi yang dibuat akan mengubah alur penggunaan *spreadsheet* yang umum dengan menambahkan mekanisme-mekanisme yang dibutuhkan. Alur aplikasi yang lama dapat dilihat pada Gambar III.10 dan aplikasi *spreadsheet* yang dibangun pada Tugas Akhir ini pada Gambar III.11.



Gambar III.10: Alur Kerja Aplikasi *Spreadsheet* Biasa

Dapat dilihat bahwa terdapat proses tambahan yang dilakukan pada aplikasi yang akan dibuat. Proses tersebut terdiri dari pencarian bagian label dan data serta proses validasi.



Gambar III.11: Alur Kerja Aplikasi *Spreadsheet* yang Akan Dibuat

Hasil dari identifikasi label dan data akan dijadikan *tuple* relasional yang disimpan dalam bentuk basis data. Alur kolaborasi bukan merupakan fokus pengembangan Tugas Akhir ini sehingga akan ditangani oleh aplikasi *spreadsheet* yang dipilih.

BAB IV

RANCANGAN, IMPLEMENTASI, DAN PENGUJIAN

IV.1 Perancangan Perangkat Lunak

Subbab perancangan perangkat lunak menjelaskan deskripsi aplikasi, analisis kebutuhan fungsional dan non-fungsional, desain perangkat lunak, serta interaksinya.

IV.1.1 Deskripsi Umum Aplikasi

Aplikasi pengumpulan data yang dibuat merupakan pengembangan terhadap aplikasi *spreadsheet* kolaboratif yang sudah ada sebelumnya yakni EtherCalc. Aplikasi yang ditambahkan ini yang akan melakukan pengaturan koneksi ke basis data. Saat pengguna menginginkan penyimpanan data ke dalam basis data yang dituju, aplikasi ini akan melakukan pencarian bagian label dan data dan melakukan validasi masukan sebelum memasukkan data ke dalam basis data.

IV.1.2 Spesifikasi Kebutuhan

Pada subbab ini akan dipaparkan *use case* aplikasi yang akan dibuat serta kebutuhan fungsional dan non-fungsional dari aplikasi. Kasus penggunaan oleh pengguna diberi ID dengan format UC-XX dengan UC menyatakan *use case* dan XX menyatakan nomor. Pengguna adalah pihak yang menggunakan aplikasi *spreadsheet* yang sudah ditambahkan fitur pengumpulan data. Kasus penggunaan oleh pengguna dijelaskan pada Tabel IV.1.

Tabel IV.1: Kasus Penggunaan oleh Pengguna

ID	Keterangan
UC-01	Pengguna dapat menentukan basis data tujuan dengan konfigurasi basis data yang diinginkan.
UC-02	Pengguna dapat menyimpan data yang dikumpulkan ke dalam basis data pada saat dibutuhkan.

ID	Keterangan
UC-03	Pengguna dapat menambah dan mengurangi tabel yang ingin disimpan secara manual maupun otomatis.
UC-04	Pengguna dapat memperbaiki atribut tabel yang ingin disimpan.
UC-05	Pengguna dapat memberikan batasan dan validasi pada suatu domain data.

Berdasarkan kasus penggunaan di atas, dirancang kebutuhan fungsional perangkat lunak yang diberi ID dengan format FR-XX dengan FR merupakan singkatan dari *functional requirement* dan XX menyatakan nomor kebutuhan. Kebutuhan fungsional dijelaskan pada Tabel IV.2.

Tabel IV.2: Kebutuhan Fungsional Aplikasi

ID	Keterangan	ID Use Case Terkait
FR-01	Aplikasi dapat melakukan koneksi kepada basis data yang ditentukan oleh pengguna melalui data masukan berupa *host*, *port*, *username*, *password*, dan *database* dari basis data yang dituju.	UC-01
FR-02	Aplikasi dapat melakukan perintah basis data kepada basis data yang dituju.	UC-01, UC-02
FR-03	Aplikasi menyediakan tombol untuk melakukan *commit* terhadap data yang akan disimpan.	UC-02
FR-04	Aplikasi menyediakan fitur menambah dan mengurangi tabel baik secara pendeteksian otomatis maupun manual.	UC-03
FR-05	Aplikasi dapat menampilkan hasil identifikasi label dan data dalam bentuk tabel.	UC-03, UC-04
FR-06	Aplikasi menyediakan fitur bagi pengguna agar dapat mengubah hasil identifikasi label dan data.	UC-04
FR-07	Aplikasi menyediakan fitur bagi pengguna agar dapat menambahkan batasan masukan pada suatu data.	UC-05

ID	Keterangan	ID Use Case Terkait
FR-08	Aplikasi dapat melakukan validasi data masukan sesuai dengan batasan yang diberikan oleh pengguna.	UC-05
FR-09	Aplikasi dapat memberitahukan kesalahan validasi yang terjadi.	UC-05

Selain kebutuhan fungsional, dijabarkan juga kebutuhan non-fungsional yang memiliki ID dengan format NF-XX dengan NF merupakan singkatan dari *non-functional requrirement* dan XX menyatakan nomor. Kebutuhan non-fungsional disajikan pada Tabel IV.3.

Tabel IV.3: Kebutuhan Non-fungsional Aplikasi

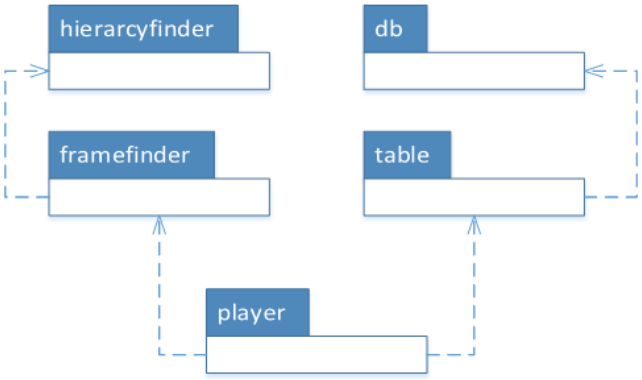
ID	Keterangan	ID Use Case Terkait
NF-01	Data masukan pengguna disimpan secara persisten.	UC-01, UC-02
NF-02	Aplikasi dapat berjalan diatas aplikasi *spreadsheet* EtherCalc.	-
NF-03	Laporan hasil analisis *response time* dan *storage* yang digunakan.	-

IV.1.3 Kebutuhan Modul

Pembangunan fitur ini diatas aplikasi EtherCalc terdiri dari lima buah modul, yaitu:

1. Modul `player`, bertugas sebagai jembatan antara *front-end* dan *back-end* dari fitur.
2. Modul `db`, bertugas untuk antarmuka baca tulis basis data.
3. Modul `framefinder`, bertugas untuk mendeteksi secara otomatis bagian label dan data pada tabel.
4. Modul `hierarchyfinder`, bertugas untuk mendeteksi secara otomatis tabel-tabel yang ada dalam suatu *sheet*.
5. Modul `table`, bertugas untuk mengelola tabel konfigurasi, melakukan validasi data, dan membuat representasi relasional.

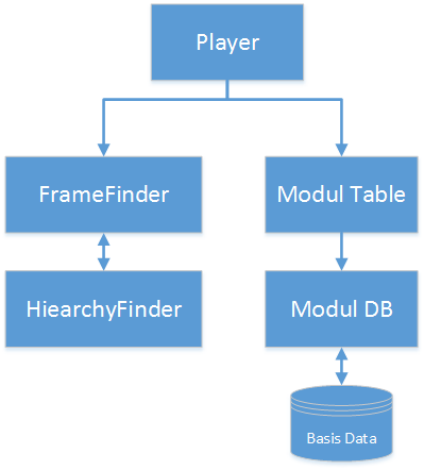
Ketergantungan antar modul dapat dilihat pada Gambar IV.1



Gambar IV.1: Ketergantungan Antar Modul

IV.1.4 Kolaborasi Antar Modul

Proses fitur ini akan dilakukan melalui modul `player` yang dapat menerima perintah pengguna melalui *front-end*. Selanjutnya modul `framefinder` akan melakukan pendeteksian label dan data secara otomatis pada masing-masing tabel yang terdapat pada *sheet*. Tabel-tabel tersebut didapatkan melalui modul `hierarchyfinder`. Selanjutnya, pada saat menerima perintah penyimpanan, modul `table` akan dipanggil oleh `player`. Jika data masukan sudah benar, maka modul `db` akan melakukan pennyimpanan ke dalam basis data. Kolaborasi antar modul disajikan pada Gambar IV.2.



Gambar IV.2: Kolaborasi Antar Modul

IV.1.5 Arsitektur

Aplikasi dibuat dengan menggunakan bantuan Docker sehingga diharapkan dapat dengan mudah dipasang pada berbagai *platform.* Aplikasi terdiri dari tiga *docker container* yakni untuk aplikasi utama, basis data redis, dan basis data MySQL. *File* `docker-compose.yml` yang dibuat untuk mendefinisikan *docker container* yang digunakan dapat dilihat pada Kode IV.1

```
ethercalc:
  build: .
  ports:
    - "80:8000"
  links:
    - redis:redis
    - mysql:mysql
  restart: always
redis:
  image: redis:latest
  volumes:
    - /var/lib/redis:/data
  command: redis-server --appendonly yes
  restart: always
mysql:
  image: mysql:5.7
  volumes:
    - dbdata:/var/lib/mysql
  restart: always
  environment:
    MYSQLROOTPASSWORD: root
    MYSQLDATABASE: TA
    MYSQLUSER: user
    MYSQLPASSWORD: user
```

Kode IV.1: Kode pada docker-compose.yml

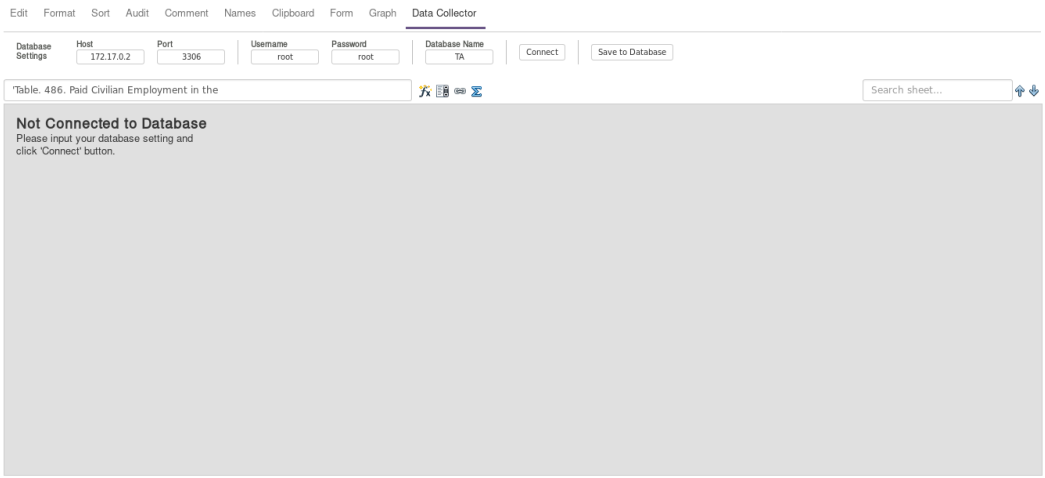
Modul-modul dan fitur dibuat diatas aplikasi EtherCalc, sehingga arsitektur aplikasi akan mengikuti arsitektur aplikasi EtherCalc yang telah dijelaskan pada Subbab II.1.2.

IV.2 Implementasi

Implementasi dilakukan dengan membangun modul yang telah dijabarkan pada Subbab IV.1.3 dengan menggunakan bahasa Javascript, menyesuaikan dengan modul lain yang telah ada pada aplikasi EtherCalc. Pada bagian ini akan dijelaskan antarmuka dan modul-modul yang diimplementasikan.

IV.2.1 Antarmuka

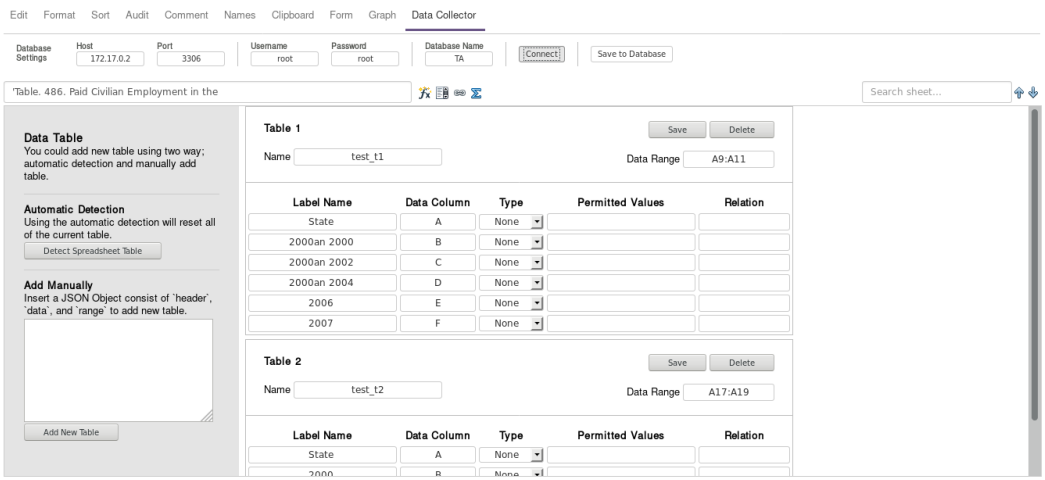
Antarmuka aplikasi pada Tugas Akhir ini mengikuti antarmuka yang telah ada sebelumnya pada aplikasi EtherCalc. Fitur baru yang ditambahkan akan menempati menu baru pada bagian atas antarmuka dengan nama 'Data Collector'. Antarmuka awal dapat dilihat pada Gambar IV.3.



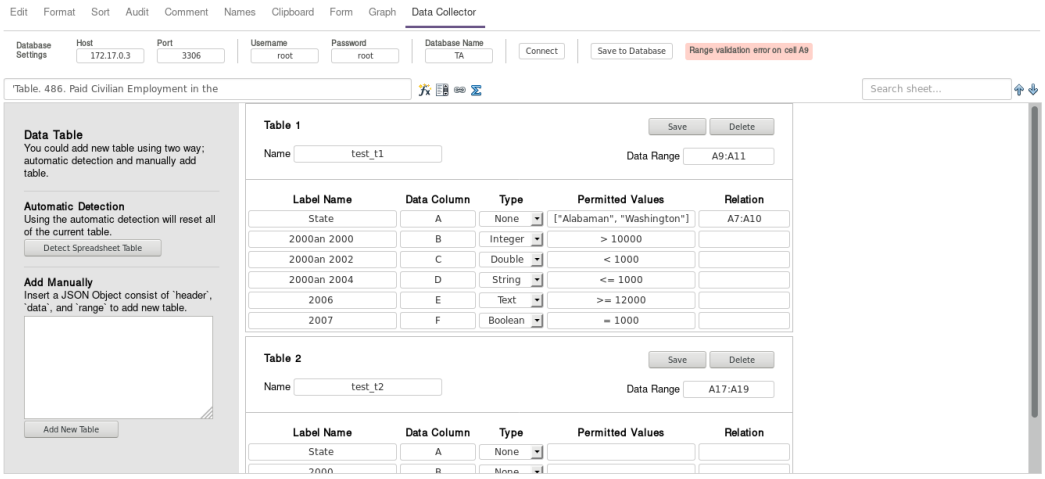
Gambar IV.3: Antarmuka Awal

Terdapat dua bagian utama pada antarmuka ini, yang pertama adalah area pengaturan basis data dimana pengguna bisa dapat mengatur koneksi ke basis data yang diinginkan. Yang kedua adalah area tempat tabel konfigurasi dapat ditambahkan dan dikurangi berdasarkan masukan pengguna. Pada bagian kiri area tabel konfigurasi, terdapat dua pilihan metode untuk membuat tabel konfigurasi yakni secara manual maupun otomatis. Kedua area ini dapat dilihat pada Gambar IV.4.

Pada bagian tabel konfigurasi, pengguna dapat menentukan nama label, tipe data, nilai-nilai yang diizinkan, dan relasinya terhadap kumpulan sel lain untuk suatu kolom data. Selain itu, pengguna dapat juga mengubah nama tabel tujuan ke basis data serta menghapus tabel konfigurasi. Jika terjadi kesalahan pada saat validasi, pengguna akan diberikan pesan kesalahan dan diharapkan untuk memperbaiki data pada sel yang dianggap memiliki kesalahan. Interaksi ini dapat dilihat pada Gambar IV.5.



Gambar IV.4: Antarmuka Tabel Konfigurasi



Gambar IV.5: Antarmuka Perubahan Tabel Konfigurasi

IV.2.2 Modul Main

Modul `Main` tidak dibuat dari awal karena mengembangkan dari modul yang telah ada dari aplikasi EtherCalc. Modul ini yang menyediakan API agar dapat digunakan oleh aplikasi. Seluruh API ini akan dipanggil melalui modul `Player`. API yang ditambahkan pada Tugas Akhir ini adalah:

1. `POST /_database/connect`

   Digunakan untuk melakukan pengecekan koneksi ke basis data. Parameter yang dibutuhkan adalah:

   - host: *host* basis data.
   - port: *port* basis data.
   - user: *user* basis dat yang digunakan.
   - password: *password* untuk *user* yang digunakan.
   - database: nama *database* yang digunakan.

2. `POST /_database/state`

   Digunakan untuk melakukan penyimpanan *state* tabel konfigurasi. Parameter yang dibutuhkan adalah:

   - tables: data tabel konfigurasi dalam bentuk JSON.
   - setting: pengaturan koneksi ke basis data yang dipilih berupa JSON Object.

3. `POST /_database/state/[spreadsheet_id]`

   Digunakan untuk mendapatkan data *state* tabel konfigurasi pada suatu *spreadsheet*. Parameter yang dibutuhkan adalah::

   - setting: pengaturan koneksi ke basis data yang dipilih berupa JSON Object.

4. `POST /_database/create`

   Digunakan untuk melakukan penyimpanan data yang telah dikumpulkan sesuai dengan aturan pada tabel konfigurasi. Parameter yang dibutuhkan adalah:

   - table: data dari tabel konfigurasi yang telah diubah menjadi bentuk relasional dalam bentuk JSON.

- setting: pengaturan koneksi ke basis data yang dipilih berupa JSON Object.

5. `GET /_framefinder/[spreadsheet_id]/[start_col]/[end_col]/[start_row]/[end_row]`

   Digunakan untuk mendapatkan hasil pencarian label dan data diantara kolom dan baris tertentu.

6. `GET /_hierarchical/[spreadsheet_id]`

   Digunakan untuk mendapatkan perkiraan kelompok sel yang merupakan suatu kesatuan tabel.

IV.2.3 Modul Player

Modul `player` merupakan modul yang menjembatani masukan pengguna dari yang berasal dari *front-end* sehingga dapat diterima oleh modul yang berada di *back-end*. Modul ini hanya terdiri dari satu kelas utama yakni kelas `player` yang berisi fungsi-fungsi yang dapat dipanggil oleh *front-end* yang dapat dilihat pada Tabel IV.4.

Tabel IV.4: Fungsi pada Kelas `Player`

Fungsi	Keterangan
refreshView	Melakukan pembaharuan tampilan sehingga menunjukkan tabel terbaru.
getDBSetting	Mengambil pengaturan basis data yang telah diberikan pengguna.
saveState	Menyimpan pengaturan yang telah dilakukan ke basis data.
loadState	Mengambil pengaturan yang pernah disimpan pada basis data.
saveConfig	Melakukan penyimpanan konfigurasi tabel yang dilakukan oleh pengguna.
deleteTable	Menghapus tabel konfigurasi yang dipilih.
addManual	Menambahkan tabel konfigurasi baru sesuai dengan masukan pengguna.
connect	Melakukan koneksi ke basis data yang dipilih.
save	Melakukan pemanggilan terhadap modul `table` dan melakukan penyimpanan ke basis data.
scan	Melakukan identifikasi tabel melalui pemanggilan modul `framefinder` yang selanjutnya akan menampilkan hasil identifikasi dan kolom perubahan konfigurasi yang dapat diisi pengguna.

IV.2.4 Modul DB

Modul basis data digunakan sebagai antarmuka modul lain untuk melakukan operasi I/O basis data. Pada Tugas Akhir ini, basis data yang digunakan adalah MySQL. Modul ini hanya terdiri dari satu kelas utama yakni kelas `db`. Kelas ini memiliki tugas sebagai penghubung aplikasi ke basis data MySQL yang dipilih. Fungsi-fungsi yang terdapat pada kelas ini dapat dilihat pada Tabel IV.5.

Tabel IV.5: Fungsi pada Kelas DB

Fungsi	Keterangan
createTable	Fungsi yang digunakan untuk membuat Table tempat pengisian data.
isTableExists	Melakukan pengecekan ada atau tidaknya tabel tersebut pada basis data.
getColumns	Mendapatkan kolom-kolom yang ada pada suatu tabel.
selectData	Mendapatkan data sesuai dengan syarat yang diminta.
insertData	Memasukkan data ke dalam tabel yang dipilih.
deleteData	Menghapus data dari tabel.
updateData	Melakukan pembaharuan data dari tabel.
dropTable	Menghapus tabel yang dipilih.

Tabel akan dibuat pada basis data yang ditentukan, setiap tabel merepresentasikan suatu tabel pada *spreadsheet* yang ditentukan oleh pengguna. *Header* yang didefinisikan oleh pengguna pada *spreadsheet* akan dijadikan *column* pada tabel basis data, tipe yang dibentuk mengikuti masukan pengguna. Tiap baris data yang ada dibawah *header* pada *spreadsheet* akan ditranslasikan menjadi bentuk relasional agar dapat dimasukan ke dalam tabel.

IV.2.5 Modul Hierarchyfinder

Modul `hierarchyfinder` menggunakan algoritma *hierarchical clustering* untuk dapat mengetahui mana yang merupakan suatu kesatuan tabel pada suatu *sheet*. Modul ini dapat menentukan tabel-tabel yang terdapat pada suatu *sheet* yang selanjutnya akan dilakukan identifikasi label oleh modul `framefinder`. Algoritma *hierarchical clustering* yang digunakan menganggap setiap sel pada *spreadsheet* merupakan suatu node. Sel-sel

yang bersebelahan dengan sel tersebut akan dianggap tetangga sehingga memiliki jarak sama dengan 0. Sel yang digabungkan dengan sel lain akan dihitung sebagai satu *node*. Aturan perhitungan jarak antar *node* dapat dilihat pada kode di Kode IV.2.

```
# Fungsi cellDistance(v1, v2)
# Parameter pada fungsi adalah v1 (node 1) dan v2 (node 2)
t = new Table null, null
colD = t.GetCellCol(v1[0]) - t.GetCellCol(v2[0])
rowD = t.GetCellRow(v1[0]) - t.GetCellRow(v2[0])

# Jika sel saling bertetangga tetapi bukan secara diagonal
# Jika bertetangga, jarak kedua sel adalah 0
if colD == 0
  if rowD == 1 or rowD == -1
    return 0
if rowD == 0
  if colD == 1 or colD == -1
    return 0

# Jika tidak, cek apakah sel bertetangga secara diagonal
# Jika bertetangga, jarak kedua sel adalah 0
leftTop = [[v1[1], v1[2]], [v2[1], v2[2]]]
leftBot = [[v1[1], (v1[2] + v1[4])], [v2[1], (v2[2] + v2[4])]]
righTop = [[(v1[1] + v1[3]), v1[2]], [(v2[1] + v2[3]), v2[2]]]
righBot = [[(v1[1] + v1[3]), (v1[2] + v1[4])], [(v2[1] + v2[3]), (v2[2] +
    v2[4])]]

if (leftTop[0][0] == righTop[1][0] and leftTop[0][1] == righTop[1][1])
  return 0
if (leftBot[0][0] == righBot[1][0] and leftBot[0][1] == righBot[1][1])
  return 0

if (leftTop[1][0] == righTop[0][0] and leftTop[1][1] == righTop[0][1])
  return 0
if (leftBot[1][0] == righBot[0][0] and leftBot[1][1] == righBot[0][1])
  return 0

if (leftTop[0][0] == leftBot[1][0] and leftTop[0][1] == leftBot[1][1])
  return 0
if (righTop[0][0] == righBot[1][0] and righTop[0][1] == righBot[1][1])
  return 0

if (leftTop[1][0] == leftBot[0][0] and leftTop[1][1] == leftBot[0][1])
  return 0
if (righTop[1][0] == righBot[0][0] and righTop[1][1] == righBot[0][1])
  return 0

# Jika tidak juga, hitung jarak menggunakan teknik Euclidian
dist = Math.sqrt(Math.pow((v2[1] + (v2[3]/2)) - (v1[1] + (v1[3]/2)), 2) +
    Math.pow((v2[2] + (v2[4]/2)) - (v1[2] + (v1[4]/2)), 2));
return dist
```

Kode IV.2: Perhitungan Jarak *Node*

Hasil dari pengelompokan ini berupa kelompok-kelompok tabel pada *spreadsheet*. Setiap tabel yang teridentifikasi selanjutnya akan dicari bagian *header* dan *label* menggunakan modul `framefinder`.

IV.2.6 Modul Framefinder

Modul `framefinder` melakukan pengidentifikasian terhadap tabel yang ada sehingga dapat diketahui baris yang merupakan *header* dan *data*. Implementasi modul ini dilakukan dengan mengikuti implementasi yang dilakukan pada penelitian yang dilakukan oleh Chen (Chen and Cafarella, 2013). Modul ini terdiri dari 5 kelas yang dapat dilihat pada Tabel IV.6.

Tabel IV.6: Kelas pada Modul `FrameFinder`

Nama Kelas	Keterangan
LoadSheet	Kelas ini berfungsi sebagai kelas yang melakukan pengambilan data dan konversi sel dan *sheet* pada *spreadsheet* ke dalam bentuk kelas-kelas yang ada pada modul ini.
MySheet	Merupakan kelas bentukan yang merepresentasikan *sheet* pada *spreadsheet* yang dipilih.
MyCell	Merupakan kelas untuk merepresentasikan *properties* yang ada pada sel pada *sheet* yang dipilih.
FeatureSheetRow	Melakukan ekstraksi fitur-fitur yang terdapat pada suatu *sheet* pada *spreadsheet*.
PredictSheetRows	Kelas ini digunakan untuk menghasilkan file dalam format yang dapat dibaca oleh algoritma Conditional Random Field (CRF) dari fitur-fitur yang telah diekstraksi pada *sheet*.

IV.2.6.1 Kelas LoadSheet

Kelas ini melakukan pengambilan data pada *spreadsheet* dengan cara membaca file *spreadsheet* dan membentuk representasi kelas yang dibutuhkan. Kelas ini memiliki satu atribut utama yakni cmysheet yang merupakan kelas MySheet. Fungsi-fungsi yang terdapat pada kelas ini dapat dilihat pada Tabel IV.13.

Tabel IV.7: Fungsi pada Kelas LoadSheet

Nama Fungsi	Keterangan
loadSheetDict	Fungsi ini merupakan fungsi utama yang bertugas untuk membuat representasi *spreadsheet* yang diterima ke dalam kelas MySheet.

Nama Fungsi	Keterangan
getValueType	Untuk mendapatkan tipe representasi data yang diberikan oleh *spreadsheet*. Contoh: tanggal, nominal uang, desimal, dan lain-lain.
getDataType	Untuk mendapatkan tipe data primitif pada suatu sel.
featureIndentation	Digunakan untuk mengecek keberadaan *property indentation* pada sel.
featureAlignStyle	Digunakan untuk mengecek keberadaan *property align* pada sel
featureFontBold	Digunakan untuk mengecek keberadaan *property bold* pada sel
featureFontHeight	Digunakan untuk mengecek keberadaan *property height* pada sel
featureFontUnderline	Digunakan untuk mengecek keberadaan *property underline* pada sel
featureFontItalic	Digunakan untuk mengecek keberadaan *property italic* pada sel
featureFontBgcolor	Digunakan untuk mengecek keberadaan *property background color* pada sel
featureBorderStyle	Digunakan untuk mengecek keberadaan *property border* pada sel.

IV.2.6.2 Kelas MySheet

Kelas MySheet merupakan kelas yang merepresentasikan *sheet* pada *spreadsheet* yang dipilih. Kelas ini memiliki 4 atribut yang dapat dilihat pada Tabel IV.8.

Tabel IV.8: Atribut pada Kelas MySheet

Nama Atribut	Keterangan
sheetdict	Merupakan representasi kumpulan sel-sel pada suatu *sheet*. Tiap sel direpresentasikan dalam bentuk kelas MyCell.
mergerowdict	Merupakan kumpulan sel-sel yang digabungkan.
maxcolnum	Nilai kolom terbesar pada sel.
maxrownum	Nilai baris terbesar pada sel.

Pada kelas ini terdapat 3 fungsi yang dapat dilihat pada Tabel IV.9.

Tabel IV.9: Fungsi pada Kelas MySheet

Nama Fungsi	Keterangan
getCellsArray	Digunakan untuk mendapatkan seluruh refresentasi sel pada kelas ini dalam bentuk *array*.
addMergeCell	Digunakan pada saat terdapat sel yang digabungkan. Sel tersebut akan dimasukkan ke dalam daftar *merged cells*.
insertCell	Menambahkan sel ke dalam kelas ini. Sel yang ditambahkan akan direpresentasikan dalam bentuk kelas MyCell.

IV.2.6.3 Kelas MyCell

Kelas MyCell merupakan kelas yang merepresentasikan sel pada suatu *sheet*. Kelas ini memiliki 17 atribut yang dapat dilihat pada Tabel IV.10.

Tabel IV.10: Atribut pada Kelas MyCell

Nama Atribut	Keterangan
x	Merupakan letak sel pada koordinat X.
y	Merupakan letak sel pada koordinat Y.
w	Nilai lebar sel.
h	Nilai tinggi sel.
cstr	Isi sel dalam bentuk *string*.
mtype	Tipe konten yang ada di dalam sel.
indents	Nilai indentasi jika terdapat indentasi pada konten.
centeralign	Bernilai *true* atau *false* bergantung pada *align* sel merupakan rata tengah atau tidak.
leftalign	Bernilai *true* atau *false* bergantung pada *align* sel merupakan rata kiri atau tidak.
rightalign	Bernilai *true* atau *false* bergantung pada *align* sel merupakan rata kanan atau tidak.
bottomborder	Bernilai *true* atau *false* bergantung pada *property bottom border* ada atau tidak.
upperborder	Bernilai *true* atau *false* bergantung pada *property upper border* ada atau tidak.
leftborder	Bernilai *true* atau *false* bergantung pada *property left border* ada atau tidak.

Nama Atribut	Keterangan
rightborder	Bernilai *true* atau *false* bergantung pada *property right border* ada atau tidak.
bold	Bernilai *true* atau *false* bergantung pada *property bold* ada atau tidak.
italic	Bernilai *true* atau *false* bergantung pada *property italic* ada atau tidak.
underline	Bernilai *true* atau *false* bergantung pada *property underline* ada atau tidak.

IV.2.6.4 Kelas FeatureSheetRow

Kelas ini bertugas untuk melakukan ekstraksi fitur pada tiap baris sel yang telah dikumpulkan dari *spreadsheet*. Fitur-fitur yang diambil untuk setiap barisnya dapat dilihat pada Tabel IV.11.

Tabel IV.11: Fitur yang Diambil dari Sel

Fitur
Baris memiliki sel yang digabung
Sel pada baris mencapai kolom paling kanan
Sel pada baris mencapai kolom paling kiri
Baris hanya memiliki 1 kolom
Baris memiliki sel rata tengah
Baris memiliki sel rata kiri
Baris memiliki sel yang ditebalkan (*bold*)
Baris memiliki sel berindentasi
Baris memiliki sel berisi kata ‘Table‘
Baris memiliki sel berisi kata berawalan tanda baca
Baris memiliki sel dengan presentase angka tinggi
Baris memiliki sel berisi huruf besar seluruhnya
Baris memiliki sel berisi kata dengan awal huruf besar
Baris memiliki sel berisi kata dengan awal huruf kecil
Baris memiliki persentase sel memiliki isi tinggi
Baris memiliki persentase sel memiliki isi berupa kata tinggi
Baris memiliki sel berisi karakter spesial

Fitur
Baris memiliki sel berisi karakter titik koma
Baris memiliki jumlah sel berisi tahun tinggi
Baris memiliki persentase sel berisi tahun tinggi
Baris memiliki jumlah sel berisi kata dengan huruf yang banyak tinggi

Fitur-fitur diatas mengikuti fitur yang didefinisikan pada penelitian yang dilakukan oleh Chen (Chen and Cafarella, 2013). Fitur-fitur ini akan digunakan dalam perhitungan algoritma CRF.

IV.2.6.5 Kelas PredictSheetRows

Kelas PredictSheetRows memiliki tugas untuk melakukan konversi fitur-fitur yang telah didapatkan pada kelas FeatureSheetRow menjadi bentuk file teks yang dapat dibaca oleh program CRFPP yang akan menjalankan algoritma CRF pada file tersebut. Contoh file yang dihasilkan oleh kelas ini dapat dilihat pada Kode IV.3.

```
DeadlineTA.xlsSheet11 1 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 Header
DeadlineTA.xlsSheet12 0 1 1 0 0 1 0 0 0 0 0 1 0 0 1 0 1 1 0 0 0 0 0 Data
DeadlineTA.xlsSheet13 0 1 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 Data
DeadlineTA.xlsSheet14 0 1 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 Data
DeadlineTA.xlsSheet15 0 1 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 Data
```

Kode IV.3: File Berisikan Fitur Tiap Baris

File tersebut ditulis dalam format `namafile_namasheet_bariske [fitur-fitur pada baris] label`. File ini yang akan diolah oleh algoritma dan digunakan untuk memprediksi label dari setiap baris tersebut. Algoritma yang digunakan adalah Conditional Random Field dan menggunakan aplikasi CRFPP yang berjalan eksternal diluar kelas ini untuk membaca file, membaca model, serta memprediski label.

IV.2.7 Modul Table

Modul `table` memiliki tugas untuk mengelola tampilan dan isi tabel konfigurasi, melakukan validasi data masukan, dan membuat representasi relasional yang dapat diterima oleh basis data. Modul `table` hanya memiliki satu kelas yakni kelas `table`. Kelas ini memiliki 8 atribut yang dapat dilihat pada Tabel IV.12.

Tabel IV.12: Atribut pada Kelas Table

Nama Atribut	Keterangan
sheet	Berisikan data untuk masing-masing sel pada suatu *sheet*.
rows	Representasi tabel yang berisikan; nama *header*, kolom data, dan aturan-aturan validasi.
range	*Range* sel data pada tabel.
title	Kumpulan baris yang diidentifikasi sebagai *title* oleh pengenalan otomatis.
footnote	Kumpulan baris yang diidentifikasi sebagai *footnote* oleh pengenalan otomatis.
header	Kumpulan baris yang diidentifikasi sebagai *header* oleh pengenalan otomatis.
data	Kumpulan baris yang diidentifikasi sebagai *data* oleh pengenalan otomatis.
name	Nama tabel yang akan dijadikan nama tabel pada basis data.

Pada kelas ini terdapat 6 fungsi yang dapat dilihat pada Tabel IV.9.

Tabel IV.13: Fungsi pada Kelas LoadSheet

Nama Fungsi	Keterangan
ParseData	Melakukan *parse* terhadap data dari *spreadsheet* menjadi bentuk objek tabel.
TupleSerializeWithChecker	Digunakan untuk mengubah struktur tabel menjadi tabel relasional dan melakukan validasi data masukan sesuai dengan aturan yang diminta pengguna.
Serialize	Mengubah objek tabel menjadi JSON.
Deserialize	Mengubah JSON menjadi objek tabel.
GetHTMLForm	Menghasilkan bentuk HTML dari objek tabel agar dapat ditampilkan pada antarmuka.

Modul ini berinteraksi langsung dengan modul `player` pada saat akan melakukan penyimpanan *state* tabel konfigurasi dan menampilkan tabel konfigurasi pada *front-end*. Selain itu, modul ini juga berhubungan dengan modul `db` pada saat melakukan penyimpanan.

IV.3 Pengujian

Pengujian dilakukan hanya kepada fitur yang di buat pada Tugas Akhir ini dan tidak kepada aplikasi EtherCalc secara keseluruhan. Pada subbab ini akan dibahas kasus-kasus yang diuji dan hasil dari pengujian tersebut.

IV.3.1 Tujuan Pengujian

Pada fitur yang dibangun pada Tugas Akhir ini, pengujian yang dilakukan mempunyai tujuan yaitu:

1. Memastikan bahwa fitur yang diimplementasi dapat berjalan dengan baik. Pengujian yang dilakukan akan dikaitkan pada kasus-kasus penggunaan yang dapat terjadi. Dari pengujian akan dilihat kesesuaian hasil sesungguhnya dengan hasil yang diinginkan.
2. Menganalisis dampak penambahan dan penggunaan fitur yang diimplementasi dengan mengukur *response time* fitur-fitur yang diimplementasi. Hal ini dilakukan untuk mengetahui pengaruh penambahan fitur terhadap keseluruhan aplikasi.
3. Menganalisis penggunaan tambahan penyimpanan yang dibutuhkan untuk menyimpan data pada dua basis data yang berbeda.

IV.3.2 Lingkungan Pengujian

Pengujian dilakukan pada komputer dengan spesifikasi pada Tabel IV.14.

Tabel IV.14: Spesifikasi Lingkungan Pengujian

Komponen	**Keterangan**
Prosesor	AMD A8-4500M APU with Radeon(tm) HD Graphics 1.90GHz
Memori Fisik	12 GB
Storage	50 GB
Sistem Operasi	Arch Linux 64-bit
Docker version	???.??
Web browser	Mozila Firefox ????

IV.3.3 Eksekusi Pengujian

Kasus pengujian disesuaikan dengan kebutuhan fungsional dan non-fungsional yang sudah dijelaskan pada Subbab IV.1. Setiap kasus uji diberi kode TC-XX dengan XX adalah nomor kasus uji. Daftar kasus uji dapat diperhatikan pada Tabel IV.15.

Tabel IV.15: Kasus Uji Fitur Aplikasi

ID	**Tujuan**	**ID kebutuhan terkait**
TC-01	Terkoneksi ke basis data MySQL yang dipilih pengguna dan dapat menggunakan basis data tersebut	FR-01, FR-02
TC-02	Mendeteksi label dan data secara otomatis pada *spreadsheet*	FR-04, FR-05
TC-03	Menambahkan tabel konfigurasi secara manual sesuai dengan masukan pengguna	FR-04
TC-04	Pengguna dapat mengubah atribut yang ada pada tabel konfigurasi	FR-06
TC-05	Data dapat dimasukkan ke basis data sesuai dengan tabel konfigurasi	FR-03, FR-09, NF-01
TC-06	Data yang dimasukkan ke basis data berhasil divalidasi sesuai aturan pengguna	FR-07, FR-08, FR-09, NF-01
TC-07	Pengujian *response time* yang terjadi akibat fitur yang ditambahkan	FN-03
TC-08	Melakukan perbandingan penggunaan *storage* untuk penyimpanan data	FN-03

Skenario dari masing-masing kasus uji dapat diperhatikan pada Lampiran A. Setiap skenario diberi kode dengan format TC-XX-YY dengan TC-XX adalah ID kasus uji dan YY adalah nomor skenario.

Daftar Pustaka

Bansal, S. K. (2014). Towards a Semantic Extract-Transform-Load (ETL) framework for big data integration. *Proceedings - 2014 IEEE International Congress on Big Data, BigData Congress 2014*, pages 522–529.

Bradbard, D. A., Alvis, C., and Morris, R. (2014). Spreadsheet usage by management accountants: An exploratory study. *Journal of Accounting Education*, 32(4):24–30.

Chan, Y. E. and Storey, V. C. (1996). The use of spreadsheets in organizations: Determinants and consequences. *Information and Management*, 31(3):119–134.

Chen, Z. and Cafarella, M. (2013). Automatic web spreadsheet data extraction. *Proceedings of the 3rd International Workshop on Semantic Search Over the Web - SS@ '13*, pages 1–8.

Chen, Z., Cafarella, M., Chen, J., Prevo, D., and Zhuang, J. (2013). Senbazuru: a prototype spreadsheet database management system. *Vldb*, 6(12):1202–1205.

dan Pengembangan Bahasa, P. P., dan Pengembangan Bahasa (Indonesia), P. P., dan Kebudayaan, I. D. P., and Balai Pustaka, P. (1991). *Kamus Besar Bahasa Indonesia*. BP (Series). Balai Pustaka.

Dictionary, M.-W. L. (2016). *OnlyOffice*. Dipetik November 7, 2016, dari http://www.merriam-webster.com/dictionary/spreadsheet.

EuSpRIG, E. S. R. I. G. (2010a). *EuSpRIG Horror Stories*. Dipetik Oktober 23, 2016, dari http://www.eusprig.org/horror-stories.htm.

EuSpRIG, E. S. R. I. G. (2010b). *EuSpRIG Why Are We Here?* Dipetik Oktober 23, 2016, dari http://www.eusprig.org/about.htm.

Foundation, T. D. (2016). *LibreOffice*. Dipetik November 26, 2016, dari https://www.libreoffice.org.

Freeman, D. (1996). How to make spreadsheets error-proof. *Journal of Accountancy*, 181(5):75–77.

Google (2016). *Google Sheet*. Dipetik November 27, 2016, dari https://developers.google.com/apps-script/overview.

Khatri, V. and Brown, C. V. (2010). Designing data governance. *Communications of the ACM*, 53(1):148.

Microsoft (2006). *Introducing the Office (2007) Open XML File Formats*. Dipetik November 17, 2016, dari https://msdn.microsoft.com/en-us/library/aa338205(v=office.12).aspx.

Microsoft (2015a). *Microsoft Office help and training - Office Support*. Dipetik November 17, 2016, dari https://support.office.com.

Microsoft (2015b). *Spreadsheet Software Program — Excel Free Trial*. Dipetik November 17, 2016, dari https://products.office.com/en/excel.

OASIS (2016). *OASIS Open Document Format for Office Applications (OpenDocument) TC*. Dipetik November 26, 2016, dari https://www.oasis-open.org/.

Panko, R. R. (1998). What We Know About Spreadsheet Errors.

Powell, S. G., Baker, K. R., and Lawson, B. (2009). Impact of errors in operational spreadsheets. *Decision Support Systems*, 47(2):126–132.

Power, D. J. (2004). A brief history of spreadsheets. *DSSResources. com*.

Rajalingham, K., Chadwick, D. R., and Knight, B. (2001). Classification of Spreadsheet Errors. *Proc. European Spreadsheet Risks Int. Grp. (EuSpRIG)*, page 9.

Ronen, B., Palley, M. a., and Lucas, H. C. (1989). Spreadsheet analysis and design. *Communications of the ACM*, 32(1):84–93.

SIA, A. S. (2016). *OnlyOffice*. Dipetik November 27, 2016, dari http://www.onlyoffice.com/.

Tang, A. (2014). *EtherCalc*. Dipetik November 27, 2016, dari https://ethercalc.net/.

Tyszkiewicz, J. (2010). Spreadsheet as a relational database engine. *Proceedings of the 2010 international conference on Management of data - SIGMOD '10*, page 195.