

**PENGEMBANGAN APLIKASI PENGUMPULAN DATA
MENGUNAKAN *SPREADSHEET***

Laporan Tugas Akhir 1

Disusun sebagai syarat kelulusan IF4091

Oleh

Feryandi Nurdiantoro

NIM 13513042



**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG**

Januari 2017

**PENGEMBANGAN APLIKASI PENGUMPULAN DATA
MENGUNAKAN *SPREADSHEET***

Laporan Tugas Akhir 1

Oleh

Feryandi Nurdiantoro

NIM 13513042

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

Bandung, 4 Januari 2017

Mengetahui,

Pembimbing I,

Pembimbing II,

Tricya Esterina Widagdo, ST., M.Sc.

NIP 197109071997022001

Yudistira Dwi Wardhana Asnar, Ph.D

NIP 198008272015041002

Daftar Isi

DAFTAR ISI	iv
DAFTAR GAMBAR	v
DAFTAR TABEL	vi
I Pendahuluan	1
I.1 Latar Belakang	1
I.2 Rumusan Masalah	3
I.3 Tujuan	3
I.4 Batasan Masalah	4
I.5 Metodologi	4
I.6 Jadwal Pelaksanaan	5
II Studi Literatur	7
II.1 <i>Spreadsheet</i>	7
II.1.1 Definisi Umum	7
II.1.2 Teknologi <i>Spreadsheet</i>	8
II.2 Penggunaan <i>Spreadsheet</i>	11
II.3 Kesalahan dalam Penggunaan <i>Spreadsheet</i>	12
II.3.1 Kualitas Data	12
II.3.2 Tingkat Kesalahan dalam Penggunaan <i>Spreadsheet</i>	12
II.3.3 Tipe Kesalahan dalam Penggunaan <i>Spreadsheet</i>	14
II.3.4 Penanganan Kesalahan pada <i>Spreadsheet</i>	15
II.4 Data pada <i>Spreadsheet</i>	16
II.4.1 Tipe Struktur Data pada <i>Spreadsheet</i>	16
II.4.2 Pengolahan Data pada <i>Spreadsheet</i>	17
II.5 Studi dan Penelitian Terkait	19
II.5.1 <i>Senbazuru: A Prototype Spreadsheet Database Management System</i> .	19
II.5.2 <i>Spreadsheet As a Relational Database Engine</i>	22

III Analisis Masalah Penanganan Kesalahan Pada <i>Spreadsheet</i>	24
III.1 Aspek Penanganan Kesalahan Pada <i>Spreadsheet</i>	24
III.2 Isu Penting pada Aplikasi <i>Spreadsheet</i>	27
III.3 Model Interaksi Pengguna	27
III.3.1 Berbasis Formulir	28
III.3.2 Berbasis <i>Spreadsheet</i>	28
III.3.3 Perbandingan Model Interaksi	28
III.4 Penentuan Bagian Data dan Label	30
III.4.1 Secara Manual	30
III.4.2 Secara Otomatis	30
III.4.3 Perbandingan Metode Penentuan	31
III.5 Validasi Data	32
III.6 Penyimpanan dan Pemulihan Data	32
III.7 Rencana Tindak Lanjut	34
DAFTAR PUSTAKA	35

Daftar Gambar

Gambar II.1	Ilustrasi Cara Kerja Kolaborasi pada EtherCalc	10
Gambar III.1	Alur Kerja Aplikasi <i>Spreadsheet</i> Biasa	26
Gambar III.2	Alur Kerja Aplikasi <i>Spreadsheet</i> yang Akan Dibuat	26
Gambar III.3	Contoh <i>Data Frame</i> Sederhana	30
Gambar III.4	Contoh <i>Tuple</i> Relasional	33

Daftar Tabel

Tabel I.1	Gantt Chart jadwal pelaksanaan tugas akhir	6
Tabel II.1	Studi terhadap Kesalahan pada <i>Spreadsheet</i>	13
Tabel III.1	Perbandingan Model Interaksi	29
Tabel III.2	Perbandingan Metode Penentuan Data dan Label	31

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

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*. Selain itu, *spreadsheet* jika digunakan dengan benar, memiliki banyak fitur dan kemampuan yang jarang diketahui orang awam. Dengan keunggulan ini, *spreadsheet* sering kali dijadikan pilihan utama dalam pengolahan data.

Bagi orang awam, *spreadsheet* sering terlihat sebagai aplikasi yang digunakan untuk melakukan perhitungan sederhana. Selain itu banyak orang menganggap, penggunaan *spreadsheet* adalah personal sehingga tidak membutuhkan tim atau bantuan orang lain dalam pembuatan sebuah *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 *spreadshett* yang dapat ditemui pada perusahaan dan instansi

adalah sebagai metode 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).

Sebagai media pengumpulan data, *spreadsheet* memiliki permasalahan penyimpanan data. Hal tersebut penting untuk diperhatikan karena 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 *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, 2010c). Telah banyak bukti dan penelitian yang menunjukkan bahwa kesalahan pada *spreadsheet* sangat mudah ditemui. Bahkan pada *spreadsheet* yang dibuat dengan sangat hati-hati, masih dapat ditemui sekitar 1 persen atau lebih kesalahan pada formula yang dibuat (Panko, 1998). Contoh kesalahan yang dapat terjadi adalah permasalahan *single version of truth* dimana bisa terdapat dua atau lebih versi dari data yang sama. Selain itu kesalahan dapat terjadi saat pengguna memasukkan data yang tidak tervalidasi.

Tingginya angka kesalahan yang dapat terjadi pada suatu *spreadsheet* merupakan hal yang sangat krusial terutama didalam bisnis. Metode pencegahan harus dapat dilakukan untuk dapat mengurangi angka kesalahan ini. Beberapa organisasi dapat menerapkan pencegahan dengan cara melakukan tahap-tahap metodologi yakni dengan pembuatan desain awal, melakukan metode *best practice* yang tersedia dan sesuai dengan situasi yang dihadapi, menerapkan *policy* khusus pada saat pembuatan *spreadsheet*, melakukan *testing*, serta pembuatan dokumentasi (EuSpRIG, 2010a). Namun, metodologi tersebut masih sangat rentan oleh kesalahan manusia karena perangkat lunak yang digunakan tetap tidak diubah didalam menjalankan metodologi tersebut.

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.

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

spreadsheet. Dengan adanya perangkat lunak ini diharapkan dapat memvalidasi data masukan dengan lebih baik. Selain itu, dengan adanya perangkat lunak ini 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. Fokus data masukan pada Tugas Akhir ini berupa data mentah yang belum direkapitulasi.
2. Tingkat normalisasi struktur basis data yang terbentuk yang berasal dari data pada *spreadsheet* bukan merupakan fokus utama pada Tugas Akhir ini.

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.

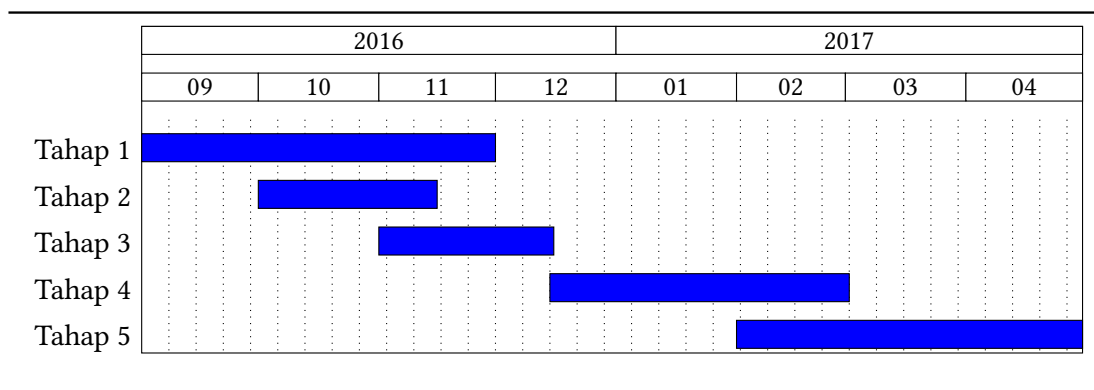
I.6 Jadwal Pelaksanaan

Pengerjaan tugas akhir ini direncanakan mulai pada September 2016 sampai April 2016. Pelaksanaan tugas akhir ini dibagi menjadi 5 tahap yang dapat dipetakan kepada metodologi pengerjaan sebagai berikut,

1. Tahap 1: Studi Literatur
2. Tahap 2: Analisis Masalah
3. Tahap 3: Perancangan Solusi
4. Tahap 4: Implementasi
5. Tahap 5: Pengujian dan Analisis Hasil

Jadwal pelaksanaan tugas akhir berdasarkan metodologi pengerjaan tugas akhir dapat dilihat pada Tabel I.1 dibawah ini.

Tabel I.1: Gantt Chart jadwal pelaksanaan tugas akhir



BAB II

STUDI LITERATUR

Pada bab ini akan mendeskripsikan kajian literatur yang terkait dengan persoalan tugas akhir. Studi literatur ini akan dijadikan dasar dalam melakukan penyelesaian persoalan.

II.1 *Spreadsheet*

Pada 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 dipasarkan. 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).

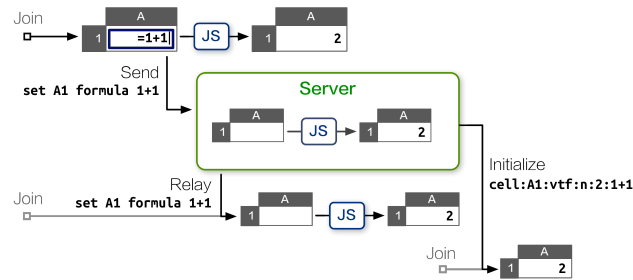
II.1.2.2 LibreOffice Calc

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 didalamnya 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 suit* milik 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 manajemen dokumen, manajemen proyek, *mail service*, serta manajemen pelanggan seperti kontak, *invoice*, *opportunities*, 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 (Khatiri and Brown, 2010). Dimensi yang ada pada kualitas data dapat dibagi menjadi:

1. Akurasi, merujuk kepada tingkat kebenaran dari data.
2. Aktualitas, menunjukkan 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 dokter 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 menggunakan *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

Pembuat	Jumlah yang Diaudit	Rata-rata Sel	Persentase Error	Keterangan Kesalahan
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, 2010b). 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 penggunaanya 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 dimasukkan 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 detail.
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 terkandung 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 dimasukkan 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 merupakan 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 teknis 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 ekstraksi ini, masih sangat mungkin memiliki kesalahan sehingga proses perbaikan diperlukan didalam pembentukan tabel relasional ini. Proses perbaikan pada prototipe 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 *xldr* 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 translasi 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, prototipe 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 performansi 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 menunjukkan bahwa arsitektur ini memiliki potensial yang dapat dikembangkan.

BAB III

ANALISIS MASALAH PENANGANAN KESALAHAN PADA *SPREADSHEET*

Pada bab ini akan diuraikan analisis persoalan penanganan kesalahan 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 Aspek Penanganan Kesalahan Pada *Spreadsheet*

Pada Subbab II.3.3 dijelaskan bahwa terdapat dua jenis kesalahan yang dapat terjadi pada penggunaan *spreadsheet* yakni kesalahan kualitatif dan kesalahan kuantitatif. Kesalahan kualitatif merupakan kesalahan penggunaan *spreadsheet* yang berhubungan dengan kualitas dan prosedur penggunaan. Contoh kesalahan kualitatif yang sering terjadi adalah penggunaan *spreadsheet* sebagai basis data. Kesalahan kuantitatif merupakan kesalahan yang menyebabkan keluaran menjadi salah. Contoh kesalahan kuantitatif yang sering terjadi adalah kesalahan masukan yang tidak divalidasi.

Pencegahan untuk kesalahan-kesalahan ini dapat dilihat pada Subbab II.3.4. Salah satu caranya adalah pembuatan *preliminary design* terhadap *spreadsheet* yang dibuat. Hal ini dapat dilakukan dengan cara pengembangan aplikasi pengumpulan data berbentuk *spreadsheet* yang menangani kolaborasi, validasi, dan penyimpanan. Dengan aplikasi ini diharapkan dapat menangani kesalahan kualitatif seperti yang dijelaskan pada contoh sebelumnya yakni penggunaan *spreadsheet* sebagai basis data dengan cara menghubungkan *spreadsheet* ke basis data yang sesungguhnya secara langsung. Serta menangani kesalahan kuantitatif dengan cara melakukan validasi masukan.

Aplikasi *spreadsheet* yang baik harus dapat diakses dan digunakan secara kolaboratif yang dapat dilakukan dengan membuat aplikasi *spreadsheet* tersebut berbasis *web*. Aspek-aspek aplikasi *web* yang perlu diperhatikan dalam membangun aplikasi *spreadsheet* adalah:

1. *Response time*

Membangun aplikasi berbasis *web* menggunakan HTTP sebagai protokol komunikasi dan tentunya didalam komunikasi tersebut membutuhkan waktu untuk mengirimkan *request* dari klien. Jika waktu yang dibutuhkan untuk melakukan respon lambat, akan sulit terjadi kolaborasi dan memperbesar kemungkinan terjadinya *race condition* pada masukan pengguna. Sehingga diperlukan waktu respon yang cepat untuk dapat menangani banyak *request* dalam satu waktu dan dalam waktu yang singkat.

2. Akses yang konkuren

Aplikasi berbasis *web* yang akan dibuat harus dapat diakses secara konkuren yakni diakses bersama-sama oleh banyak klien dalam satu waktu. Pengaksesan secara konkuren dapat berdampak pada terpenggilnya banyak *query* dalam satu waktu. Oleh karena itu aplikasi yang dibuat harus dapat menjalankan secara konkuren setiap *request* yang masuk.

3. Akses basis data

Akses terhadap basis data dibutuhkan sebagai media penyimpanan yang persisten dan konsisten. Oleh karena itu, akses terhadap basis data diperlukan untuk kemudahan penyimpanan dan pengambilan data serta mengatasi permasalahan ketidaksamaan *versioning* diantara suatu *file spreadsheet*.

Keempat aspek pada aplikasi berbasis *web* tersebut akan menjadi aspek utama yang dikembangkan didalam pembuatan aplikasi *spreadsheet* berbasis aplikasi *web*.

Berdasarkan pada Subbab II.3.3, terdapat dua jenis tipe kesalahan (Panko, 1998) yang bisa terjadi pada penggunaan *spreadsheet* yakni:

1. Kesalahan Kualitatif

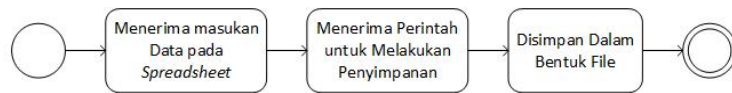
Kesalahan kualitatif merupakan kesalahan yang berhubungan dengan kualitas yang dipengaruhi oleh prosedur pembuatan dan rancangan pada *spreadsheet* tersebut. Contoh kesalahan kualitatif yang sering terjadi adalah penggunaan *spreadsheet* sebagai media penyimpanan atau basis data. Hal ini tidak sesuai dengan kegunaan utama dari *spreadsheet* sebagai media kalkulasi dan perhitungan statistik.

2. Kesalahan Kuantitatif

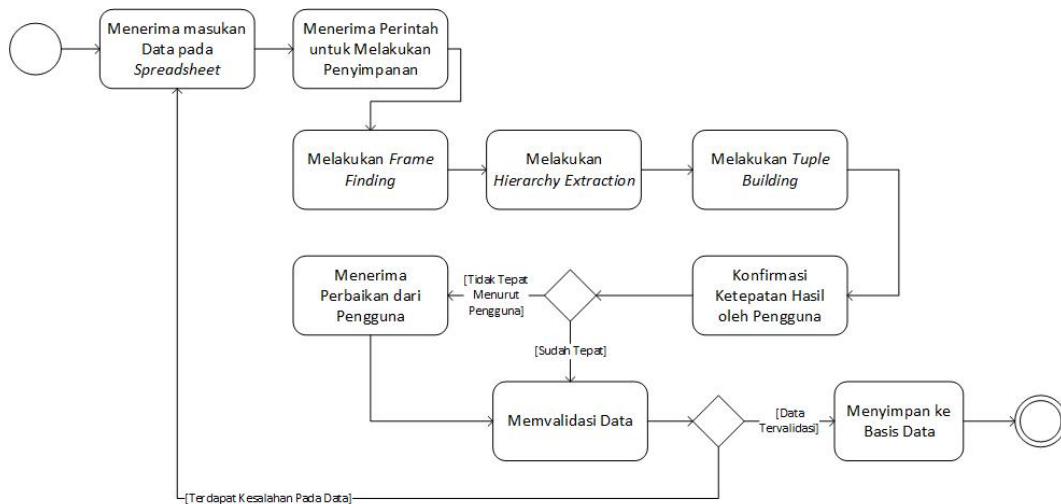
Kesalahan kuantitatif merupakan kesalahan yang menyebabkan hasil perhitungan menjadi salah atau tidak valid. Contoh kesalahan jenis ini yang sering terjadi adalah kesalahan mekanikal dimana pengguna salah memasukan data yang tidak sesuai dengan konstrain yang ada. Hal ini dapat diatasi dengan adanya validasi sebelum data diterima.

Dari kedua tipe tersebut, pada Tugas Akhir ini akan lebih ditekankan kepada kesalahan kuantitatif yang bertipe kesalahan mekanikal dengan memanfaatkan teknik validasi pada masukan.

Dengan menggunakan aplikasi ini, terdapat perubahan alur penggunaan *spreadsheet*. Gambar dibawah ini merupakan alur penggunaan aplikasi *spreadsheet* biasa pada Gambar III.1 dan aplikasi *spreadsheet* yang dibangun pada Tugas Akhir ini pada Gambar III.2.



Gambar III.1: Alur Kerja Aplikasi *Spreadsheet* Biasa



Gambar III.2: Alur Kerja Aplikasi *Spreadsheet* yang Akan Dibuat

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. Hasil dari identifikasi label dan data akan dijadikan *tuple* relasional yang disimpan dalam

bentuk basis data.

III.2 Isu Penting pada Aplikasi *Spreadsheet*

Di dalam membangun aplikasi *spreadsheet* tersebut, terdapat beberapa isu yang harus ditangani yakni:

1. Pemilihan model interaksi aplikasi *spreadsheet* dimana pengguna dapat membentuk struktur tempat pengguna lain memasukan input. Contohnya dalam bentuk tabel atau formulir. Pengguna juga melengkapi domain dan batasan data yang dimasukkan.
2. 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
3. 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.
4. 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.

III.3 Model Interaksi Pengguna

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

III.3.1 Berbasis Formulir

Model interaksi ini menggunakan *spreadsheet* sebagai tempat perancangan formulir. Pembuat formulir akan menentukan area label dan input secara manual serta diidentifikasi 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.3.2 Berbasis Spreadsheet

Model interaksi berbasis *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 dimasukkan 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.3.3 Perbandingan Model Interaksi

Kedua model interaksi tersebut memiliki beberapa perbedaan dan efek terhadap penggunaannya baik bagi sistem maupun pengguna. Pada Tabel III.1 dijabarkan perbandingan antara kedua model interaksi tersebut.

Tabel III.1: Perbandingan Model Interaksi

Parameter	Berbasis Formulir	Berbasis <i>Spreadsheet</i>
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 <i>parsing</i> dari sel pada <i>spreadsheet</i> ke dalam bentuk formulir.	Dibutuhkan algoritma dan logika <i>parsing</i> 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 <i>spreadsheet</i> sebagai media untuk memasukan data.

Pada Tugas Akhir ini, akan dipilih model interaksi 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.4 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.3 dapat dilihat bahwa area dengan nomor 1 disebut sebagai label yang menjelaskan data-data dibawahnya yakni pada area nomor 2.

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

Gambar III.3: Contoh *Data Frame* Sederhana

III.4.1 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.

III.4.2 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.4.3 Perbandingan Metode Penentuan

Untuk mengetahui perbandingan kedua metode, pada Tabel III.2 dijelaskan kelebihan dan kekurangan dua metode yang telah disebutkan sebelumnya.

Tabel III.2: 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 <i>tuple</i> relasional lebih tinggi.
Kekurangan	Interferensi pengguna yang banyak dan mungkin data dan label tidak dapat dijadikan bentuk <i>tuple</i> relasional.	Algoritma ini hanya optimal jika digunakan pada tabel yang terurut vertikal.

Dari perbandingan diatas, dapat dilihat terdapat kekurangan dan kelebihan didalam meng-

gunakan 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.5 Validasi Data

Setelah pengguna memasukkan 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* 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.6 Penyimpanan dan Pemulihan 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*.

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.4.

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.4: Contoh *Tuple* Relasional

Setelah data dan label dijadikan *tuple* relasional, *tuple* dimasukan ke dalam basis data dengan menggunakan operasi *insert* ke dalam tabel yang terhubung dengan data tersebut. Pada saat pemulihan data, data yang disimpan akan kembali ditampilkan pada sel yang bersesuaian sehingga pengguna dapat melanjutkan perubahan pada *spreadsheet* dan berkolaborasi. 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.7 Rencana Tindak Lanjut

Berdasarkan analisa yang telah dijelaskan pada bab-bab sebelumnya, pada Tugas Akhir ini akan dibangun aplikasi *spreadsheet* dengan menggunakan EtherCalc sebagai teknologi utama. Pada Subbab II.1.2 telah dijelaskan bahwa EtherCalc merupakan perangkat lunak *spreadsheet* yang *open source* dan memiliki kemampuan kolaborasi didalam penggunaannya. Sehingga dengan menggunakan EtherCalc, aspek-aspek yang telah dijelaskan pada Subbab III.1 yang harus ada didalam pembangunan *spreadsheet* ini sudah dapat dipenuhi. Pada Tugas Akhir ini akan dilakukan pengembangan dari EtherCalc yang berupa identifikasi label dan data, penanganan validasi data dan keterhubungannya dengan basis data relasional. Alur kerja dari aplikasi *spreadsheet* yang akan dibuat dapat dilihat pada Gambar III.2 pada Subbab III.1.

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 Best Practice*. Dipetik Oktober 23, 2016, dari <http://www.eusprig.org/best-practice.htm>.
- EuSpRIG, E. S. R. I. G. (2010b). *EuSpRIG Horror Stories*. Dipetik Oktober 23, 2016, dari <http://www.eusprig.org/horror-stories.htm>.
- EuSpRIG, E. S. R. I. G. (2010c). *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](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.