STUDI KASUS LINGUSQL: APLIKASI TRANSAKSI PERDAGANGAN SAHAM

Rikky Wenang Purbojati, Ade Azurat, Api Perdana, dan Heru Suhartanto

Fakultas Ilmu Komputer, Universitas Indonesia, Kampus Baru UI Depok, Jawa Barat, 16424, Indonesia

E-mail: rwenang@cs.ui.ac.id

Abstrak

Proses pengembangan perangkat lunak yang ideal selalu mensyaratkan pengujian yang menyeluruh untuk memperoleh hasil perangkat lunak yang memiliki tingkat kebenaran tertentu. Namun pada prakteknya pengujian secara menyeluruh sangat jarang dilakukan karena membutuhkan sumber daya waktu dan biaya yang banyak. LinguSQL adalah sebuah *tool* pengembangan eksperimen yang mengintegrasikan proses pengujian secara *whitebox* dan *blackbox* ke dalam aktifitas pembuatan kodenya. *Paper* ini memaparkan penerapan LinguSQL dalam pengembangan studi kasus sebuah aplikasi transaksi perdagangan saham. Penerapan LinguSQL pada studi kasus yang cukup kompleks diharapkan akan menampilkan keuntungan konsep pengujian secara menyeluruh serta, dalam konteks implementasi *tool*, menunjukkan bagian-bagian yang masih perlu dikembangkan lebih lanjut.

Kata Kunci: *automated testing, linguSQL, sotftware verification*

Abstract

The ideal process software development always requires thorough testing to obtain the software that has a certain degree of truth. However, in practice very rarely thorough testing done because it requires so much resources of time and cost. LinguSQL is an experimental tool that integrates the development process is whitebox and blackbox testing in manufacturing activity code. This paper describes the implementastion of LinguSQL in the development of a stock trading application case study. Implementation of LinguSQL on a complex case study will show the expected benefit of testing the concept a thorough and in the context of the implementation tool, showing the parts that still need to be developed further.

Keywords: *automated testing, linguSQL, sotftware verification*

1. Pendahuluan

Penerapan sistem informasi dengan menggunakan basis data saat ini sudah sangat menyatu dalam segala aspek kehidupan kita sebagai manusia modern. Kita menggunakan sistem informasi secara langsung ataupun tidak langsung mulai dari hal yang kecil seperti mengurus kartu tanda penduduk, sampai dengan yang besar seperti proses pemilihan umum. Dalam mengimplementasikan sistem informasi tersebut, kita harus dapat merancang algoritma-algoritma pemrograman yang benar dan sesuai dengan fungsionalitas yang diharapkan. Kesalahan dalam merancang dan mengimplementasikan algoritma dalam sistem informasi yang digunakan secara publik dapat mengakibatkan kerugian yang banyak baik finansial maupun non-finansial.

Untuk mencegah dan meminimalkan kesalahan-kesalahan yang dapat terjadi di dalam sebuah sistem informasi, seorang *programmer* dapat menerapkan teknik-teknik analisis pengujian. Kita dapat menggunakan teknik pengujian *blackbox*, yang menekankan kepada hasil akhir atau *output* dari sebuah algoritma, atau pengujian *whitebox*, yang menekankan kepada tingkat kebenaran alur algoritma tersebut diimplementasikan. Pada prakteknya karena penggunaan teknik *whitebox* memerlukan sumber daya waktu dan uang yang lebih banyak daripada teknik *blackbox*, kebanyakan pembuat perangkat lunak lebih memilih untuk mengintegrasikan sistem pengujian *blackbox* ke dalam proses pengembangan perangkat lunaknya.

Permasalahan yang muncul kemudian adalah teknik pengujian *blackbox* memiliki keefektifan mendeteksi kesalahan sebatas keberagaman *input* yang dimasukkan ke algoritma tersebut. Sebuah masukan yang diberikan kepada algoritma tersebut tidak dapat menjamin bahwa masukan yang lain akan memberikan hasil yang benar. Oleh sebab itu dalam kasus sistem informasi yang

bersifat kritis, sebuah proses verifikasi dan validasi algoritma harus dilakukan juga untuk menjamin algoritma memenuhi spesifikasi kebutuhan yang telah ditentukan sebelumnya.

LinguSQL adalah sebuah *tool* yang digunakan untuk melakukan verifikasi dan validasi pada sebuah algoritma di dalam skrip Lingu [1]. Bahasa Lingu sendiri adalah sebuah bahasa pemrograman abstrak ringan yang menekankan kepada operasi transformasi data pada basis data [2]. Lingu tidak memiliki seluruh fungsionalitas seperti layaknya bahasa pemrograman seperti C++ atau Java, namun bahasa ini memiliki sintaks dan tata bahasa yang memungkinkan kita untuk merancang sebuah algoritma manipulasi data pada sebuah basis data. Penerapan verifikasi dan validasi pada sebuah algoritma di dalam Lingu dapat menghasilkan ukuran kebenaran yang terdapat di dalam alur operasi algoritma tersebut.

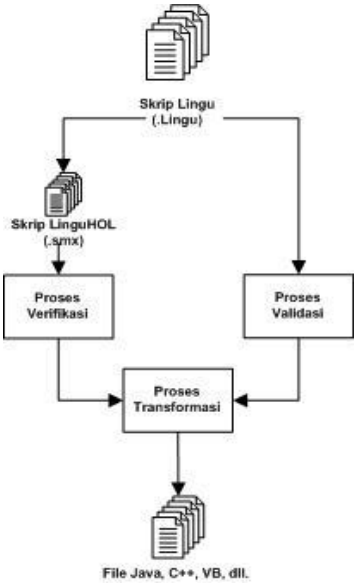
Penelitian ini akan membahas penggunaan LinguSQL dalam sebuah studi kasus mengenai aplikasi transaksi perdagangan saham dengan mengacu kepada praktek perdagangan di Indonesia. Aplikasi yang diimplementasikan merupakan versi sederhana dari aplikasi transaksi saham yang sesungguhnya dan mengikutsertakan aturan dan batasan yang ada di prakteknya. Sebagai bagian dari proses pengujian, LinguSQL akan menerapkan kasus verifikasi dan validasi menggunakan aturan dan batasan yang tersebut. Hasil yang diharapkan adalah bahwa LinguSQL dapat melakukan proses verifikasi dan validasi pada studi kasus yang cukup kompleks dan kritis seperti transaksi saham.

2. Metodologi

Paper ini membahas mengenai studi kasus transaksi perdagangan saham yang diimplementasikan sebagai skrip Lingu. Sifat abstrak Lingu menunjukkan bahwa skrip dalam bahasa ini lebih berupa susunan algoritma dan spesifikasi, karena itu tidak bisa di-*compile* ataupun dieksekusi. Transformasi bahasa Lingu menjadi bahasa pemrograman konkrit yang digunakan untuk pengembangan aplikasi nyata dilakukan oleh satu bagian dari LinguSQL.

LinguSQL adalah *tool* pengembangan yang digunakan untuk melakukan verifikasi, validasi, dan transformasi skrip Lingu. LinguSQL menggunakan bantuan *Higher-Order Logic* (HOL*) theorem prover* untuk melakukan pembuktian secara formal dari seluruh prosedur dan *method* yang dideklarasikan di skrip Lingu. Sedangkan untuk proses validasi, LinguSQL memiliki modul generator yang dapat menghasilkan sampel data acak untuk digunakan sebagai input dalam pengujian skenario (*blackbox)*. Lalu proses transformasi dilakukan oleh LinguSQL untuk bahasa pemrograman yang populer saat ini seperti Java. Secara garis besar proses pengembangan dan pengujian sebuah aplikasi dengan menggunakan LinguSQL terdiri dari empat tahapan.

Tahap Pendefinisian Data dan Fungsi: Tahapan ini dilakukan dengan cara mengidentifikasikan data-data yang terlibat dalam *domain* permasalahan aplikasi. Data tersebut harus didefinisikan dalam bentuk tabel di sebuah basis data. Selanjutnya adalah pendefinisian fungsi atau *method* apa saja yang akan diujikan dengan menggunakan LinguSQL. Pemilihan fungsi yang akan diujikan sepenuhnya tergantung dari tingkat kekritisan alur fungsi tersebut. Proses ini menghasilkan sebuah skrip Lingu dasar yang berisi struktur data dan method.



Gambar 1. Alur pengembangan LinguSQL.

Tahap verifikasi skrip LinguHOL: Tahap verifikasi dilakukan untuk memastikan kebenaran alur algoritma dari prosedur yang terdapat di skrip Lingu. Tahap ini memerlukan perubahan bentuk skrip Lingu menjadi LinguHOL. Perubahan ini termasuk menambahkan *verification condition* yang akan diujikan dalam pembuktian algoritma dalam skrip Lingu. Verifikasi dilakukan dengan memproses skrip LinguHOL ke dalam HOL *theorem prover*. LinguSQL akan menangkap hasil dari *theorem prover* dan menerjemahkan apakah skrip tersebut lolos dari proses verifikasi atau tidak.

Tahap validasi skrip Lingu: Tahap validasi digunakan untuk melakukan pengujian apakah

eksekusi prosedur yang telah didefinisikan di dalam definisi Lingu akan menghasilkan *output* yang sesuai harapan. Untuk melaksanakan pengujian ini, LinguSQL dilengkapi dengan sebuah *data generator* yang dapat menghasilkan sampel data terstruktur untuk digunakan sebagai input prosedur skrip Lingu. Proses validasi dianggap berhasil jika seluruh pengujian eksekusi kode validasi mengalami keberhasilan.

Tahap transformasi ke bahasa konkrit: Tahap terakhir dari pengembangan ini adalah pengubahan skrip Lingu menjadi bentuk bahasa pemrograman yang konkrit seperti Java. Sebuah modul transformasi dapat dikembangkan ke dalam LinguSQL untuk mengakomodasi kebutuhan bahasa pemrograman lainnya.

Hasil yang diharapkan dari pengembangan dengan menggunakan LinguSQL adalah sebuah komponen kode yang telah teruji secara menyeluruh. Proses pengujian dengan verifikasi atau *whitebox* dan validasi/*blackbox* dapat meminimalkan dan mencegah kesalahan algoritma. Hal ini mengurangi resiko kerugian pada organisasi yang menggunakan aplikasi tersebut.

Transaksi perdagangan saham merupakan kegiatan ekonomi yang melibatkan pertukaran uang dan saham yang dimiliki oleh perusahaan publik dan investor. Pertukaran jumlah uang yang sangat besar secara elektronis memerlukan sebuah sistem informasi yang handal dan teruji kebenarannya. Hal ini sangat kritis untuk menjamin bahwa seluruh transaksi yang dilakukan melalui cara elektronik ini dapat dipertanggungjawabkan dan sesuai dengan kenyataan yang terjadi di lantai bursa.

Satu contoh kasus yang cukup relevan sehubungan dengan perangkat lunak pasar saham adalah kasus kesalahan penghitungan indeks rata-rata Dow Jones yang terjadi pada tanggal 1 Maret 2007. Sebuah kesalahan terjadi pada proses penghitungan indeks rata-rata industri selama 70 menit yang menyebabkan kebingungan di lantai bursa saham. Akibat yang ditimbulkan adalah jatuhnya nilai Dow Jones sebesar 416.02 poin serta kerugian investor terkait dengan penghitungan indeks yang salah yang mencapai jutaan dolar [3].

Kasus tersebut menunjukkan bahwa pengembangan sistem informasi dengan pengujian secara menyeluruh sangat penting untuk menghindari kerugian finansial yang tidak perlu. Hal ini beserta sifat penyimpanan data transaksi secara elektronis di basis data menyebabkan perdagangan saham menjadi studi kasus yang sangat cocok untuk LinguSQL.

Secara historis praktik perdagangan saham telah ada di Indonesia sejak tahun 1912 pada zaman Hindia Belanda, namun baru pada tahun 1995 perdagangan saham secara elektronik dapat dilakukan dibawah kendali Bursa Efek Jakarta [4]. Prosedur untuk melakukan transaksi saham oleh seorang investor sesuai dengan aturan yang berlaku sampai Desember 2007, terdiri dari lima poin penting.

Pertama, seorang investor yang ingin melakukan transaksi jual-beli saham harus melalui broker atau sekuritas yang terdaftar di bursa saham. Kedua, perintah pembelian/penjualan yang diberikan kepada broker akan diteruskan ke lantai bursa. Ketiga, di lantai bursa, trader akan memasukkan penawaran pembelian/penjualan tersebut ke dalam *Jakarta Automatic Trading System* (JATS). Keempat, apabila terdapat penawaran pembelian/penjualan yang cocok (*matched*) dengan perintah investor, maka sistem JATS akan memfinalisasikan transaksi tersebut. Kelima, penyelesaian transaksi (kliring & pembayaran) akan diselesaikan tiga hari setelah kejadian transaksi sebenarnya.

Selain prosedur dan tata cara perdagangan yang harus diakomodasi sebagai alur logika perangkat lunak, sebuah sistem informasi perdagangan saham harus juga mengintegrasikan aturan dan batasan yang berlaku. Aturan dan batasan yang berlaku bersumber dari regulasi pemerintah melalui BAPEPAM (Badan Pengawas Pasar Modal), kondisi sosial budaya lokal seperti periode transaksi dan hari libur, serta batasan yang diperlukan untuk menjamin transaksi dapat diproses secara benar seperti batasan jumlah transaksi per detik dan sebagainya. Sebagai studi kasus, aturan-aturan tersebut dapat dijadikan kasus dimana proses verifikasi tersebut harus dilakukan untuk menjamin bahwa algoritma sistem informasi tidak pernah dan tidak akan melanggar aturan dan batasan itu. Terdapat beberapa contoh aturan dan batasan yang ada pada praktek perdagangan saham saat ini, diantaranya: 1) Transaksi saham hanya dapat dilakukan pada hari kerja, Senin-Kamis (9.30-12.00, 13.30-16.00) dan Jumat (9.30-11.30, 14.00-16.00); 2) Tiap broker/sekuritas memiliki batas jumlah transaksi (*trading limit*) sesuai jumlah jaminan untuk aktifitas transaksi selama satu hari; 3) Harga penawaran penjualan dan pembelian yang diajukan oleh broker tidak boleh lebih/kurang sebesar X% dari harga normal (X adalah variabel yang ditentukan oleh regulasi sesuai dengan nilai saham, antara 20-50%); 4) Sistem melakukan keputusan pemetaan penjualan dan pembelian berdasarkan prioritas harga dan waktu. Harga yang lebih tinggi akan lebih diprioritaskan daripada harga rendah, apabila sama akan diproses berdasarkan waktu pencatatan; 5) Sistem harus melakukan pemeriksaan data transaksi

untuk mencegah data *invalid* masuk ke dalam *database*, misal kesalahan entri data, data yang tidak sesuai skema baku, kesalahan sistem broker, dan lain-lain; 6) Sistem harus memberi peringatan apabila terjadi fluktuasi dalam harga saham atau volume perdagangan dalam satu hari. Anomali tersebut dapat menyebabkan sebuah jenis saham diberi status *suspend*, dan tidak diperdagangkan selama masa penyelidikan.

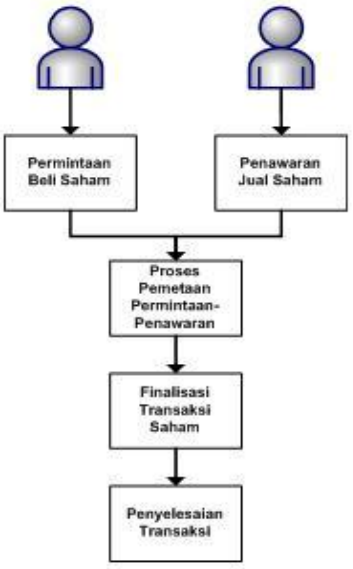
Aturan dan batasan ini harus dipatuhi oleh sistem yang digunakan untuk melakukan perdagangan saham. Kesalahan dalam perancangan algoritma yang menyebabkan aturan dan batasan ini dapat terlanggar merupakan hal yang harus dicegah pada tahap pengembangan. Sebuah proses verifikasi dan validasi yang menyeluruh harus diterapkan untuk meminimalkan dan mengidentifikasi kesalahan dalam algoritma sedini mungkin.

Studi kasus transaksi saham: Wilayah permasalahan transaksi perdagangan saham memiliki tingkat kompleksitas dan kekritisan yang sangat tinggi. Hal ini berkaitan dengan prosedur pencatatan transaksi yang atomik serta prosedur penyelesaian transaksi yang harus akurat dan handal. LinguSQL merupakan teknologi yang masih dalam tahapan awal pengembangan dan memiliki banyak keterbatasan fungsional. Dalam konteks tersebut studi kasus yang akan digunakan dalam penerapan LinguSQL merupakan hasil penyederhanaan dari proses bisnis transaksi saham sebenarnya. Walaupun demikian, seluruh sifat kekritisan dan aturan yang terdapat pada proses bisnis transaksi saham tetap dipertahankan sebagaimana prakteknya di dunia nyata.

Penyederhanaan dari sistem transaksi perdagangan saham yang akan digunakan meliputi pencatatan permintaan pembelian saham, pencatatan penawaran penjualan saham, serta proses pemetaan antara permintaan dan penawaran saham yang kemudian akan menghasilkan transaksi jual-beli saham sebenarnya. Skema dari sistem transaksi saham yang akan digunakan oleh LinguSQL ditunjukkan dalam gambar 2.

Alur sistem transaksi saham pada gambar 2 meliputi dua buah kegiatan awal investor, yaitu penawaran penjualan saham pada harga tertentu, serta permintaan pembelian saham pada harga tertentu. Sebuah mekanisme pemetaan akan dijalankan secara periodik yang berfungsi untuk mencocokkan harga antara permintaan dan penawaran saham yang sama. Apabila terdapat permintaan saham sebanyak 10 pada harga 5000, maka mekanisme ini akan berusaha mencarikan penawaran penjualan saham tersebut pada harga 5000.

Apabila ditemukan penawaran penjualan saham yang cocok, mekanisme ini akan memfinalisasikan transaksi saham dan meneruskan data transaksi ini ke sistem penyelesaian, yang tidak termasuk ke dalam studi kasus ini, untuk diselesaikan sampai dengan pembayaran ke investor.



Gambar 2. Diagram alur simulasi transaksi saham.

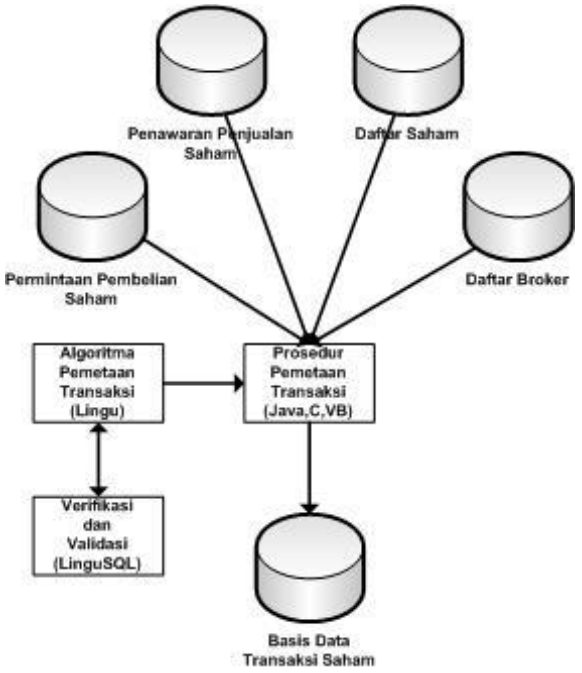
3. Hasil dan Pembahasan

Dengan mengacu kepada prosedur studi kasus yang telah disebutkan di bagian sebelumnya, implementasi studi kasus ini akan menggunakan basis data untuk mensimulasikan pencatatan data yang terlibat dalam prosedur tersebut. Gambar 3 adalah diagram basis data yang digunakan dalam implementasi basis data transaksi saham.

Tabel yang digunakan terdiri dari 4 (empat) tabel data masukan serta satu tabel keluaran yang akan berisi hasil pemrosesan prosedur pemetaan transaksi. Seperti yang akan dijelaskan oleh masing-masing paragraf di bawah ini.

Tabel permintaan pembelian saham: Tabel ini berisi data transaksi permintaan pembelian saham oleh investor. Data transaksi yang ada di tabel ini meliputi data saham yang hendak dibeli, identitas broker yang membeli, jumlah saham yang hendak dibeli, serta pada harga berapa broker tersebut bersedia membeli.

Id (Int)	**Timestamp** (String)	**Broker_Id** (Int)	**Stock_Id** (Int)	**Amount** (Int)	**Price** (Int)



Gambar 3. Diagram basis data transaksi saham.

Tabel penawaran penjualan saham: Tabel ini berisi data transaksi penawaran penjualan saham oleh investor. Data transaksi yang ada di tabel ini meliputi data saham yang hendak dijual, identitas broker yang menjual, jumlah saham yang dijual, serta pada harga berapa broker tersebut bersedia menjual.

Id (Int)	**Timestamp** (String)	**Broker_Id** (Int)	**Stock_Id** (Int)	**Amount** (Int)	**Price** (Int)

Tabel daftar saham: Tabel ini berisi data-data saham yang diperdagangkan beserta informasi detilnya. Data saham yang tercatat di tabel ini meliputi nama saham, volume saham yang diperdagangkan, harga yang tercatat saat ini.

Id (Int)	**Stock_Name** (String)	**Volume** (Int)	**Current_Price** (Int)	**Timestamp** (Int)

Tabel daftar broker: Tabel ini berisi data-data broker yang terdaftar di bursa saham beserta informasi detilnya. Data saham yang tercatat di tabel ini meliputi nama broker beserta jumlah batas transaksi yang dimilikinya.

Id (Int)	**Stock_Name** (String)	**Volume** (Int)

Tabel Transaksi Saham: Tabel ini berisi data-data transaksi saham yang dihasilkan oleh prosedur pemetaan transaksi. Data saham yang tercatat di tabel ini meliputi id saham yang diperdagangkan, nama broker pembeli dan penjual, volume saham yang diperjualbelikan, dan harga penjualan.

Buy (Int)	**Sell** (Int)	**BrkrBuy** (Int)	**BrkrSell** (Int)	**Amnt** (Int)	**Price** (Int)	**StockId** (Int)

Dalam implementasi studi kasus ini, seluruh basis data akan dipopulasikan dengan menggunakan sebuah generator data yang mensimulasikan input data entri. Seluruh prosedur dan proses verifikasi dan validasi akan dilakukan pada basis data ini dengan data tersebut.

Implementasi skrip Lingu: Sebuah skrip Lingu untuk studi kasus ini terdiri dari pendefinisian data yang digunakan, pendefinisian algoritma atau prosedur yang akan dijalankan pada data transaksi saham, serta pendefinisian skenario validasi untuk melakukan *blackbox* testing pada algoritma tersebut. Hasil akhir dari proses implementasi dengan skrip Lingu ini adalah sebuah komponen yang terverifikasikan secara formal (*whitebox*) serta tervalidasikan dengan skenario (*blackbox*).

Langkah pertama dari pembuatan skrip Lingu adalah pendefinisian data yang akan digunakan dalam aplikasi transaksi saham. Berdasarkan kebutuhan basis data yang telah didefinisikan sebelumnya, Lingu membutuhkan empat tabel jenis data, dimana tabel permintaan dan penawaran dapat digeneralisasikan menjadi jenis tabel yang sama. Bagian skrip Lingu yang mendefinisikan tabel-tabel tersebut dapat dilihat pada gambar 4.

Langkah selanjutnya dari pengembangan studi kasus transaksi saham adalah penyusunan algoritma pemetaan transaksi dan algoritma-algoritma pendukung lainnya dalam bahasa Lingu.

Algoritma-algoritma yang dirancang untuk studi kasus transaksi saham diambil dari titik kritis transaksi saham seperti proses pemetaan, dan juga dari pemeriksaan aturan dan batasan seperti pemeriksaan data transaksi yang *invalid*. Terdapat empat algoritma yang diimplementasikan untuk studi kasus ini.

Algoritma pemetaan transaksi saham: Algoritma ini mengambil data masukan dari tabel permintaan pembelian saham dan penawaran penjualan saham. Seluruh data kemudian diurutkan dengan prioritas nilai harga yang besar serta waktu pencatatan yang paling dahulu. Langkah selanjutnya adalah melakukan iterasi untuk seluruh data permintaan beli saham dan mencocokkannya dengan data di tabel penawaran saham. Jika terdapat data penawaran saham yang cocok, maka algoritma ini akan memindahkan dua data tersebut ke tabel transaksi saham yang telah final, lalu menghapus data-data tersebut dari tabel pembelian dan penawaran saham.

Algoritma pemeriksaan transaksi dengan broker *invalid*: Algoritma ini melakukan pembersihan tabel penawaran saham dan permintaan saham dari data transaksi dengan broker *invalid*. Definisi broker *invalid* diantaranya adalah broker yang tidak terdaftar di sistem, atau broker yang telah melewati batas transaksi untuk hari tersebut.

Algoritma pemeriksaan transaksi dengan saham *invalid*: Algoritma ini melakukan pembersihan tabel penawaran saham dan permintaan saham dari data transaksi dengan saham *invalid*. Definisi saham *invalid* diantaranya adalah saham yang tidak terdaftar di sistem, atau saham yang sedang dalam status *suspend* dan karenanya tidak boleh diperdagangkan.

Algoritma pemeriksaan transaksi dengan waktu pencatatan yang tidak sah. Algoritma ini melakukan pembersihan tabel penawaran saham dan permintaan saham dari data transaksi dengan waktu pencatatan di luar periode transaksi.

Berikut adalah salah satu contoh hasil skrip Lingu untuk algoritma pemeriksaan transaksi dengan data saham yang *invalid*.

Algoritma pada gambar 5 melibatkan tiga operator basis data yang berbeda yang disediakan oleh bahasa Lingu untuk melakukan pemeriksaan data saham yang *invalid*. Baris pertama dari badan *method* pada gambar 5 menginstruksikan Lingu untuk menyimpan seluruh nomor "Id" dari tabel permintaan pembelian saham pada tabel sementara "ids". Baris kedua kemudian mencari seluruh nomor "Id" di dalam tabel permintaan pembelian dimana di data tersebut terdapat nomor identitas saham yang ada di tabel daftar saham. Hasilnya adalah nomor identitas dari seluruh data permintaan pembelian yang memiliki data saham *valid*. Baris ketiga berfungsi menghilangkan nomor identitas data permintaan pembelian yang sah dari tabel sementara ids. Hasilnya adalah sebuah tabel sementara dengan nomor identitas data permintaan pembelian saham yang tidak memiliki data saham yang *valid*. Operasi pada baris terakhir kemudian memasukkan seluruh data permintaan pembelian dengan data saham yang tidak *valid* ke sebuah tabel untuk transaksi yang tidak sah.

```
type StockListTable = Record
{|      Id            :: Integer;
        Stock_Name     :: String;
        Stock_Volume   :: Integer;
        Current_Price  :: Integer;
        Update_Timestamp  :: String;
|}

type BrokerListTable = Record
{|      Id            :: Integer;
        Broker_Name    :: String;
        Trading_Limit  :: Integer;
|}

type TransactionTable = Record
{|      Id       :: Integer;
        Timestamp :: String;
        Broker_Id :: Integer;
        Stock_Id  :: Integer;
        Amount    :: Integer;
        Price     :: Integer;
|}

type FinalTransactionTable = Record
{|      Timestamp :: String;
        Buy_Id :: Integer ;
        Sell_Id :: Integer ;
        Buyer_Broker_Id :: String;
        Seller_Broker_Id :: String;
        Stock_Id  :: Integer;
      Amount    :: Integer;
        Price     :: Integer;
|}

type StockExchangedb = Dbase
{|
        StockTab  :: Table StockListTable;
        BuyTransactionTab  ::
      Table TransactionTable;
        SellTransactionTab ::
      Table TransactionTable;
        MatchedTransactionTab ::
      Table FinalTransactionTable;
        InvalidTransactionTab ::
      Table TransactionTable;
         BrokerTab::Table
         BrokerListTable;
|}
```

Gambar 4. Definisi tabel skrip Lingu.

```
method filterInvalidStockTransaction () ::
()

ok_ids,ids :: Table {| Id :: Integer; |};

do {

ids := findAll d<-database.BuyTransactionTab
where T found d.Id;

ok_ids:= findAll d<-
database.BuyTransactionTab,
b<-database.StockTab
where d.Stock_Id == b.Id found d.Id;

delete ids where ids.Id in ok_ids.Id;

insertAll d<-database.BuyTransactionTab,i<-
ids
where d.Id == ids.Id
to database.InvalidTransactionTab;
}
```

Gambar 5. Skrip Lingu pemeriksaan transaksi dengan saham *invalid*.

```
val filterInvalidStockTransaction_def =

Define `filterInvalidStockTransaction(

REF(StockTab: StockListTable set),
REF(BuyTransactionTab : TransactionTable
set),
REF(InvalidTransactionTab : TransactionTable
set)
)
=
pre  (empty InvalidTransactionTab
/\ ~(empty StockTab)
/\ ~(empty BuyTransactionTab) )

post ( ALLof InvalidTransactionTab (satisfy
r. ALLof StockTab
(satisfy q. ~(r.Stock_Id = q.Id))))

do /{
let
badids = select BuyTransactionTab I
(only r. ALLof StockTab(satisfy t.
~(t.Id = r.Stock_Id)))
in
insert badids InvalidTransactionTab I ALL
/}
return void`;

(*----- verification -----*)

reduce defs
filterInvalidStockTransaction_def ;
L0min_vcg.autoverify MY_TAC ;
L0min_vcg.VCs;
L0min_vcg.conclude();
```

Gambar 6. Skrip LinguHOL pemeriksaan transaksi dengan saham *invalid*.

Method-method pada skrip Lingu harus menjalani proses verifikasi untuk menjamin bahwa algoritma yang dirancang sudah memenuhi spesifikasi yang diberikan. Proses verifikasi ini melibatkan skrip Lingu yang telah diubah menjadi skrip LinguHOL. Skrip ini dipecah-pecah menjadi formula preposisi matematis yang kemudian dimasukkan ke dalam *verification condition* [5]. Sebuah *verification condition* dapat dikirim ke *Higher-Order Logic* (HOL) *theorem prover. Theorem prover* digunakan untuk membuktikan bahwa algoritma dan *verification condition* yang diberikan oleh *method-method* pada skrip Lingu telah memenuhi persyaratan dan kondisi yang diberikan. Namun terkadang *theorem prover* HOL menghadapi masalah-masalah yang cukup kompleks sehingga tidak bisa memberi verifikasi secara otomatis [1]. Untuk kondisi seperti itu, bantuan manusia dapat digunakan untuk mengarahkan HOL *theorem prover* menggunakan aturan yang tepat. Gambar 6 menunjukkan contoh dari skrip LinguHOL yang berasal dari *method* pemeriksaan transaksi dengan data saham yang *invalid*.

Skrip LinguHOL pada gambar 6 dihasilkan dari *method* yang berfungsi untuk membersihkan tabel transaksi pembelian saham dengan data saham yang tidak *valid*. Aspek yang ditambahkan pada skrip ini adalah pernyataan secara eksplisit mengenai prekondisi dan postkondisi yang diharapkan dari eksekusi *method* ini. Contoh prekondisi yang diterapkan di skrip ini adalah bahwa sebelum *method* dijalankan tabel data transaksi dengan saham *invalid* harus kosong, serta tabel daftar saham dan data transaksi pembelian saham tidak boleh kosong.

Proses eksekusi *method* didefinisikan setelah kata kunci "do". Pada skrip ini proses eksekusi hanya melibatkan satu pemanggilan *select query*, yang berfungsi untuk mengambil seluruh data pada tabel transaksi yang tidak memiliki data ID saham yang *valid*, serta satu pemanggilan *insert*, yang berfungsi untuk memasukkan seluruh data dengan identitas saham tidak dikenali ke dalam tabel transaksi *invalid*.

Kondisi yang diharapkan setelah proses eksekusi dijalankan adalah bahwa di dalam tabel transaksi *invalid* tidak ada satupun data transaksi yang memiliki data identitas saham yang *valid*. Dengan demikian proses menjamin bahwa tidak akan ada *false positive* dimana sebuah transaksi dengan data identitas saham yang *valid* akan dimasukkan ke tabel transaksi *invalid*. Jika kode skrip LinguHOL tersebut berhasil dibuktikan memenuhi persyaratan prekondisi dan postkondisi oleh *theorem prover*, maka dapat dikatakan bahwa algoritma tersebut telah terbukti kebenarannya sesuai spesifikasi.

```
validation checkInvalidStockFilter() :: Bool
{
 temp_trans :: TransactionTable;
 temp_stock :: StockListTable;
 isNotInInvalid, isInValid :: Bool;
 do
 {
 temp_trans.Id := 10;
temp_trans.Broker_id := 1;
        temp_trans.Stock_id := 5;

temp_trans.Amount := 10;
temp_trans.Price := 100;

temp_stock.Id := 5;
temp_stock.Stock_Volume   := 100;
temp_stock.Current_Price  := 1000;

insertAll x<-temp_trans
where T to
database.BuyTransactionTab;
insertAll y<-temp_stock
where T to
database.StockTab;

call filterInvalidStockTransaction();

isNotInInvalid :=
find a<-database.InvalidTransactionTab
where a.Id == temp_trans.Id
found F otherwise T;
isInValid :=
find a<-database.BuyTransactionTab
where a.Id == temp_trans.Id
found T otherwise F;
}
return isNotInInvalid /\ IsInValid;
pre    T;
post T;
}
```

Gambar 7. Skrip validasi pemeriksaan transaksi dengan saham *invalid.*

Langkah selanjutnya setelah melakukan verifikasi adalah melakukan validasi terhadap prosedur yang ada di skrip Lingu. Proses validasi berbeda dengan proses verifikasi, karena sebuah proses verifikasi bertujuan untuk membuktikan apakah sebuah algoritma berjalan dengan benar sesuai spesifikasi yang didefinisikan pengembang [6] sedangkan proses validasi bertujuan untuk membuktikan apakah seluruh algoritma berjalan sesuai dengan yang dibutuhkan oleh proses bisnis.

Dalam pembuatan skrip Lingu, sebuah bagian validasi dapat dimasukkan untuk tiap prosedur atau *method* yang ada. Bagian validasi memiliki prekondisi dan postkondisi seperti layaknya proses verifikasi, namun pada saat eksekusi proses ini LinguSQL akan men-*generate* sampel data ke dalam basis data. Sampel data ini digunakan untuk menjalankan skenario validasi dari skrip Lingu tersebut. Contoh kode validasi yang digunakan untuk menguji *method* pemeriksaan transaksi dengan saham yang *invalid* dapat dilihat pada gambar 7.

Proses validasi pada gambar 6 melakukan pengujian untuk *method filterInvalidStockTransac-tion*() yang berfungsi untuk melakukan pembersihan tabel transaksi pembelian saham yang memiliki data saham *invalid*. Kode tersebut menguji satu kasus untuk membuktikan bahwa pemanggilan *method* tersebut tidak akan menimbulkan *false positive*, yaitu kasus dimana sebuah catatan transaksi yang *valid* dikategorikan sebagai *invalid* karena kesalahan kategori.

Langkah terakhir yang dilakukan dalam studi kasus transaksi perdagangan saham adalah transformasi kode abstrak Lingu menjadi bahasa pemrograman kongkrit seperti Java, C++, atau VB. Proses transformasi akan mengubah seluruh kode abstrak Lingu dan menerjemahkannya ke dalam berkas-berkas bahasa pemrograman pilihan sesuai dengan sintaks dan tata bahasanya masing-masing. Dalam kasus ini kode abstrak Lingu ditransformasikan ke bahasa Java dan menghasilkan komponen program Java. Komponen program yang telah terverifikasi dan tervalidasi ini dapat di-*compile* lalu diintegrasikan ke dalam sistem informasi transaksi saham.

4. Kesimpulan

Paper ini membahas mengenai penerapan bahasa abstrak Lingu dan penggunaan *tool* verifikasi dan validasi LinguSQL dalam konteks studi kasus transaksi perdagangan saham. Studi kasus ini digunakan karena sifatnya yang *mission-critical*, membutuhkan pengujian yang menyeluruh untuk mencegah kerugian finansial, serta sangat bergantung kepada operasi basis data. Studi kasus diambil dengan mengacu kepada prosedur dan aturan yang ada pada praktek perdagangan saham Bursa Efek Indonesia. Penyusunan skrip Lingu serta pelaksanaan proses validasi dan verifikasi pada studi kasus ini dapat menghasilkan komponen konkrit dalam bentuk Java yang diintegrasikan ke dalam sistem yang digunakan untuk melakukan pencatatan transaksi perdagangan saham.

Beberapa kelemahan yang muncul dari implementasi studi kasus ini adalah keterbatasan ekspresi kondisional Lingu dalam operasi *update* dan *select,* belum sempurnanya aplikasi yang menjembatani transformasi Lingu ke LinguHOL, serta belum diimplementasikannya *return value* dari sebuah pemanggilan prosedur. Kekurangan-kekurangan tersebut harus dapat diatasi sebelum LinguSQL dapat digunakan sebagai perangkat pengembangan aplikasi di dunia nyata.

Referensi

[1] R. Wenang, I.S.W.B. Prasetya, S. Maizir, B. Wibowo, & A. Azurat, "LinguSQL: A Verification and Transformation Tool for Database Application" *In Proceedings of 6th National Seminar of Computer Science and Information Technology (SNIKTI)*, 2005.

[2] A. Azurat, I.S.W.B. Prasetya, T.E.J. Vos,H. Suhartanto, B. Widjaja, L.Y. Stefanus, R. Wenang, S. Aminah, & J. Bong, "Towards Automated Verification of Database Scripts" *In Proceedings of 18th International Conference on Theorem Proving in Higher Order Logics*, 2005.

[3] T. Young, "Dow Jones computer error could cost millions", Computing, http://www.computing.co.uk/computing/news/2184528/computer-glitch-causes-havoc, 2007, retrieved January 4, 2008.

[4] Bursa Efek Indonesia, "History of BEI", Bursa Efek Indonesia, 2008, http://www.bei.co.id/MainMenu/TentangBEI/History/tabid/61/lang/en-US/language/en-US/Default.aspx, retrieved December 28, 2008.

[5] H. Suhartanto, I.S.W.B. Prasetya, C. D. Puspa, & A. Azurat, Proses verifikasi piranti lunak basis data dengan Lingu dan theorem prover HOL, LP FEUI, 2007.

[6] D. Coughlin, An Integrated Approach to Verification, Validation, and Accreditation of Models and Simulations, 2000.