Remik: Riset dan E-Jurnal Manajemen Informatika Komputer
Volume 8, Nomor 3, Agustus 2024
http://doi.org/10.33395/remik.v8i3.13752
e-ISSN : 2541-1330
p-ISSN : 2541-1332

Penerapan Stored Procedure untuk Memudahkan Modifikasi dan Update Formula Logic pada ETL

[1]Muhammad Firdaus, [2] Shedriko
[1,2]Universitas Indraprasta PGRI
Jakarta, Indonesia

[1]dasurichi@gmail.com, [2]shedriko@gmail.com

*Penulis Korespondensi
Diajukan : 12/06/2024
Diterima : 05/08/2024
Dipublikasi : 05/08/2024

ABSTRAK
Berkembangnya teknologi basis data memungkinkan adaptasi perubahan struktur dan prilaku data menjadi lebih cepat, serta dapat mengakomodir perubahan kebutuhan pengguna terhadap data itu sendiri. Banyaknya permasalahan pada kenyataannya terjadi pada saat pengembangan aplikasi, dimana seorang *programmer* kesulitan menambahkan atau mengubah kondisi logik secara cepat dan tepat, apabila *server* serta sarana pendukungnya dibatasi sesuai dengan ketetapan prosedur IT (Informasi Teknologi) di perusahaan / organisasi tersebut. Penerapan *Stored Procedure* – SQL (*Structure Query Language*) dalam hal ini merupakan solusi di setiap masalah yang terjadi dengan setiap skenario kebutuhan data yang berbeda-beda. Namun dalam hal ini *Stored procedure* diperbantukan untuk mengatasi kesulitan transformasi data pada ETL (*Extract-Transform-Load)* dengan menjadikannya sebagai *second assist solution* akibat dari perubahan kebutuhan si pengguna. Hal ini membuat kami memutuskan mempergunakan R&D (*Research and Development*) dengan model SDLC (*System Development Life Cycle) Extreme Programming* yang telah disesuaikan dengan kondisi dilapangan sebagai metode yang tepat untuk penelitian kami. Semoga solusi ini dapat memberikan ide dan solusi terbaik kepada pada pengembang Sistem Basis Data atau pakar Analisa Data, agar dapat memberikan asupan vitamin yang tepat selama proses pengembangan Sistem Infomasi maupun yang masih mencari-cari solusi alternatif yang tepat pada skenario permasalahan yang sedang dihadapi.

Kata Kunci: Basis Data, pengembangan, *Stored Procedure*, SQL, *Syntax of Code*

I. PENDAHULUAN
Ketika kebutuhan pasar semakin pesat dan meningkatnya jumlah transaksi yang hadir di setiap nasabah, maka disitulah kebutuhan produk perbankan pun semakin meroket. Pada saat bank mencetak laba keuntungan, maka menjadi kewajiban pula bagi bank untuk melaporkan setiap kegiatan keuangan yang sedang berlangsung kepada Bank Indonesia dan OJK (Otoritas Jasa Keuangan) sebagai pihak Regulator. Semakin tingginya kebutuhan Bank terhadap TI (Teknologi Informasi dalam menampilkan laporan yang tepat dan akurat, menjadikan hal ini sebagai peluang terbuka untuk para pengembang aplikasi untuk memberikan Solusi terbaik bagi bank. Pengalaman kami di lapangan yang berkaitan dengan ETL (*Extract – Transform – Load*) data menjadikan subyek penelitian kali ini. ETL dalam hal ini masih menjadi primidona bagi Bank untuk menstimulus setiap data inputan yang di dapat, sehingga bisa dipilah-pilah dan di ringkas menjadi metadata final yang dibutuhkan dalam pelaporan. Setiap kali user secara manual edit perubahan, penambahan, serta penggabungan data dengan menggunakan aplikasi *spreadsheet* sebelum akhirnya di ekspor kedalam bentuk *text file*, dan konten dari *text file* tersebut disisipkan kedalam format *template* BI (Bank Indonesia) sebelum akhirnya di input kedalam aplikasi khusus

Remik: Riset dan E-Jurnal Manajemen Informatika Komputer
Volume 8, Nomor 3, Agustus 2024 e-ISSN : 2541-1330
http://doi.org/10.33395/remik.v8i3.13752 p-ISSN : 2541-1332

yang telah disediakan oleh BI (Firdaus & Shedriko, 2022). ETL sendiri dikenal sebagai sarana pendukung untuk memudahkan perpindahan, transformasi, serta pemuatan/*load* data ke tempat penyimpanan baru di area *staging*, sehingga dapat dengan mudah dirubah, di manipulasi, maupun ditambahkan sesuai dengan kebutuhan pengguna sebelum di proses berikutnya ke *data warehouse* ataupun *data mart* yang nantinya diteruskan ke aplikasi-aplikasi pihak ketiga (ASHRAF, 2020).

Penerapan *Stored Procedure* pada proses ETL bukanlah hal baru, namun berdasarkan pengalaman *tools* ini lebih efektif Ketika seringkali dihadapi dengan perubahan kebutuhan pengguna yang begitu cepat. Banyak sekali masalah yang muncul pada saat pengembangan, ketika hanya fokus pada *flow* (alur) data di perancangan ETL menggunakan perangkat lunak Microsoft Visual Studio. Pihak manajemen seringkali merubah proses bisnis ataupun kebijakan sesuai dengan tuntutan perubahan yang seringkali dicanangkan oleh pihak Regulator (Bank Indonesia maupun OJK). Mereka harus segera memutuskan dan meminta pihak pengembang untuk segera menyesuaikan dengan cepat agar tidak terkena denda dikemudian hari.

II. STUDI LITERATUR

Penggunaan *Stored Procedure* besar sekali manfaatnya bagi pengembang, beberapa diantaranya: dapat mengurangi duplikasi data, eksekusi *query* menjadi lebih cepat, serta lalu lintas jaringan minimum dan aman terhadap pengguna yang tidak terotorisasi. *Stored Procedure* ini juga dapat membagi beban *resource* yang terpakai pada saat program lainnya sedang berjalan. Hal ini juga dapat meringankan pengeksekusian *query* yang dilakukan secara rutin dan berkala (Warman, 2021). Namun pengembang jarang sekali memperhatikan aspek efektifitas maupun efesiensi *query*. Pada Basis Data dengan jumlah data sedikit mungkin belum terlihat pengaruhnya terhadap *performance*, tetapi ketika data semakin besar dan bahkan *growing* sehingga memenuhi kapasitas penyimpanan fisik data dan hal ini baru berdampak pada setiap komponen *query* yang telah dibuat dan dijalankan di *server* (Hastono, 2019). Pengalaman penulis pada *Stored Procedure* (SP) di Microsoft SQL Server sering kali dipakai untuk mempermudah mengelola perubahan logika pemograman serta juga dapat mengontrol perubahan kode itu sendiri. Dimana kondisi memungkinkan minor *remote* akses ke *server* Basis Data.

Berdasarkan hasil penelitian sebelumnya yang dilakukan oleh Agus Dudu dan rekan peneliti lainnya menunjukkan perbedaan yang jelas signifikan bahwa waktu eksekusi *Stored Procedure* untuk proses *query create* dan *read* malah cenderung meningkat sesuai dengan bertambahnya jumlah data, tetapi beda halnya dengan proses *query update* dan *delete*, dimana waktu eksekusi yang dilakukan bervariasi seiring dengan jumlah data yang telah diinputnya. Pengujian tersebut mempergunakan perangkat lunak MySQL *Server* dan SQLyog dengan konfigurasi mengikut waktu eksekusi pada *Stored Procedure* di perangkat lunak tersebut (Dudu et al., 2019). Studi kasus yang pernah dihadapi penulis selama implementasi ETL (*Extract-Transform-Load*) pada salah satu institusi keuangan mencatat bahwa masih adanya resiko yang muncul ketika menggunakan solusi data transformasi tradisional, dimana ketika dibutuhkan pengembangan atau peningkatan terhadap solusi sistem yang cepat namun tidak bisa menghindari proses pemetaan data yang membutuhkan waktu yang tidak sedikit akibat dari perubahan proses bisnis maupun Keputusan manajemen di institusi tersebut (Firdaus & Shedriko, 2024). Hal ini bukan berarti Solusi seperti tidak bisa menjadi andalan utama, tetapi apabila di kombinasikan dengan penggunaan *Stored Procedure* hal ini bisa meningkatkan kinerja pengembang serta mempercepat waktu pengembangan solusi sistem.

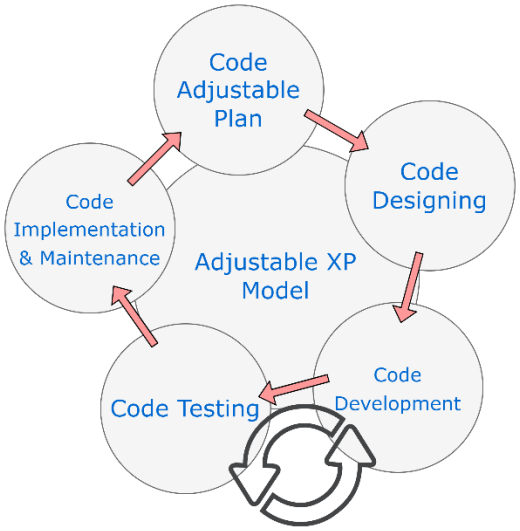
Pada dasarnya *Stored Procedure* merupakan bagian dari kalimat kode dari T-SQL (*Transaction Structure Query Language*) yang dikelompokan menjadi satu *batch* pemrosesan query. Suatu *batch* berupa urutan kalimat transaksi SQL yang dijalankan secara prosedural dan dikirim ke mesin pengelola *query* Basis Data untuk diproses secara bersamaan. Jumlah kalimat *query* yang diproses akan menyesuaikan dengan seberapa besar ukuran proses *batch* di kompilasi (Petkovic, 2020). Jadi *tool* ini dengan melihat manfaat dan penggunaannya dapat memberikan sumbangsih terhadap dunia Pendidikan maupun para praktisi di tanah air, seperti halnya pernyataan visi dan misi pada salah satu perguruan tinggi XYZ yaitu dapat membantu mengentaskan kemiskinan dengan memberikan kualitas Pendidikan yang lebih baik namun dengan biaya yang lebih terjangkau (Shedriko & Firdaus, 2022). Diharapkannya nanti dengan

Remik: Riset dan E-Jurnal Manajemen Informatika Komputer
Volume 8, Nomor 3, Agustus 2024
http://doi.org/10.33395/remik.v8i3.13752
e-ISSN : 2541-1330
p-ISSN : 2541-1332

menjabarkan penerapan Solusi ini secara jelas dapat memberikan pencerahan dan wawasan luas kepada para praktisi, sehingga hal-hal yang dapat memberikan kesalah-pahaman, serta komunikasi yang tidak baik dapat di hindari (Tabak & Dubovi, 2023), serta bisa bermanfaat dalam menghadapi kompetensi dan kesiapan dalam menghadapi tantangan yang cukup berat di lapangan dengan beragam Tingkat kesulitan data yang berbeda-beda di masa yang akan datang (McMaster & Rague, 2018).

III. METODE

Pada saat proses pelaksanaannya berlangsung, penelitian ini menggunakan metode penelitian *Research* and *Development*. Hal ini dapat membantu pengembangan serta validasi produk, yang seringkali diterapkan oleh pakar maupun praktisi dalam merancang model produk mereka (Yuliani et al., 2021). Metode ini merupakan penelitian dasar yang dapat mengobligasikan diri dalam menciptakan produk yang lebih efektif dan efisien, sehingga dapat menjabarkannya ke dalam bentuk prosedur maupun langkah-langkah yang dapat mencapai target tujuan akhir. Secara intensional, metode ini juga merupakan sistem dalam merencanakan pengembangan, peningkatan, maupun evaluasi terhadap produk yang sedang dikembangkan (Yuliani et al., 2021). Banyak model pendukung yang bisa digunakan untuk riset dan pengembangan sistem/solusi/produk, TI (Teknologi Informasi), namun menurut kami model pengembangan yang paling tepat menurut kami adalah menggunakan model SDLC (System Development Life Cycle) *Extreme Programming*, karena menurut kami dapat memberikan efek *domino* yang lebih cepat mengakomodir perubahan serta memitigasi resiko dengan baik (Firdaus & Shedriko, 2024). Namun dalam pengembangannya, kami sesuaikan model ini dengan kondisi dilapangan, sehingga kami bagi menjadi 4 tahap, yaitu: 1. *Code Adjustable Plan*, 2. *Code Designing*, 3. *Code Development*, 4. *Code Testing*, serta 5. *Code Implementation and Maintenance* (lihat gambar dibawah ini).



Gambar 1. Model *Adjustable Extreme Programming*
Sumber : Pribadi 2024

Pada model ini kami kembangkan untuk mempermudah modifikasi *syntax of code* sehingga dapat menyesuaikan perubahan saat proses pengembangan menjadi lebih cepat, namun efektif tanpa merubah keseluruhan *code* yang sebelumnya telah dibuat. Untuk lebih detail tahapan model sebagai berikut:

3.1 Code Adjustable Plan

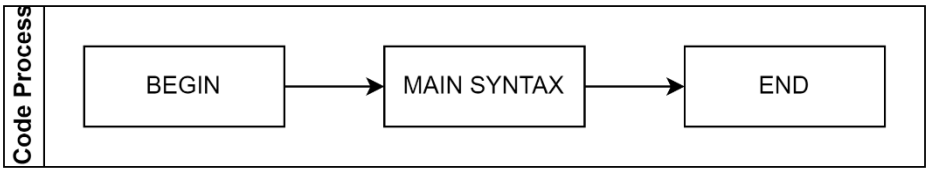
Pada tahapan ini dilakukan perencanaan penggunaan bahasa pemograman yang tepat dan menyesuaikan dengan kebutuhan dari aplikasi pihak ketiga ataupun *server* / *platform* OS (*Operating System*) / basis *data* yang di sediakan oleh pihak tim TI (Teknologi Informasi)

Remik: Riset dan E-Jurnal Manajemen Informatika Komputer
Volume 8, Nomor 3, Agustus 2024
http://doi.org/10.33395/remik.v8i3.13752
e-ISSN : 2541-1330
p-ISSN : 2541-1332

Pengguna.

3.2 Code Designing

Seperti halnya pada model SDLC lainnya, di tahapan ini perlu disiapkan rancangan *code* yang tepat dan menyesuaikan dengan *data mapping* yang telah diberikan oleh pihak pengguna ataupun pihak konsultan bisnis, sehingga rancangan *code* yang telah dibuat terhindar dari banyak *code* yang tidak terpakai ataupun perulangan yang dapat membingungkan tim pengembang lainnya ketika memperbaiki atau memodifikasi *code* sebelumnya. Adapun hal penting yang dilakukan oleh para pengembang perlu memperhatikan alur proses pengembangan *code* seperti pada gambar 2 dibawah ini.



Gambar 2. Proses Pengembangan *Code*
Sumber : Pribadi 2024

3.3 Code Development

Pada tahapan ini para pengembang memulai pembuatan *syntax of code* dengan mengikuti alur proses pada Gambar 2 dan menyesuaikannya dengan *data mapping* yang telah diberikan sebelumnya. Pentingnya *data mapping*, apabila hal ini tidak dilakukan sehingga proses pembuatan *syntax of code* menjadi tidak terarah dan *ad hoc*.

3.4 Code Testing

Sebelum hasil pengembangan diujikan oleh pengguna, di tahapan ini para pengembang perlu melakukan *cross check* dengan melihat apakah hasil maupun kompilasi program sesuai dengan harapan atau tidak (*trail and error*). Pengujian terhadap *syntax of code* yang telah dibuat, perlu memperhatikan juga dari segi estetika pemograman (*optional*), struktur pemograman, serta kesederhanaan. *Syntax of code* yang terlalu rumit dapat menyusahkan para pengembang itu sendiri dikemudian hari.

3.5 Code Implementation and Maintenance

Setelah selesai pengujian, di tahapan ini pengembang dapat menerapkan *syntax code* yang telah dibuat pada media kompilasi yang sudah disediakan oleh pihak TI pengguna, serta mengintegrasikannya dengan *syntax of code* yang sudah ada sebelumnya (*existing page*) atau *blank page*, apabila *syntax of code* yang dibuat memang baru. Perlu diingat pada tahapan ini memungkinkan langkah proses *backup* terlebih dahulu pada *code* yang mau di *update* / modifikasi / penambahan. Pada tahapan ini juga akan dilakukan pemeliharaan pada *syntax of code* yang telah dibuat sebelum baik secara berkala ataupun pada periode yang telah disepakati sebelumnya dengan pihak TI pengguna.

IV. HASIL DAN PEMBAHASAN

Adapun penerapan *syntax of code* yang kami lakukan adalah berikut ini: *Code Adjustable Plan*, dalam membuat barisan *syntax of* code yang efektif namun tepat dan presisi, tidak ada salahnya perlu melihat terlebih dahulu lingkungan pendukung TI yang telah di sediakan maupun yang akan disediakan oleh pihak pengguna. Setidaknya salah satu perspektif yang mudah dan bisa digunakan tanpa membebankan biaya TI yang tidak terduga / diluar dari komitmen bersama. Selama proses pelaksanaanya, kami menggunakan fitur T-SQL (*Transaction - Structure Query Language*) pada DBMS (*Database Management System*) Microsoft SQL Server. Aspek lainnya yang tidak kalah penting adalah kapan dan dimana kita bisa menyisipkan, menambah, merubah, menghapus baris *syntax of code* yang diperlukan, sesuai dengan kebutuhan bisnis si pengguna dan

Remik: Riset dan E-Jurnal Manajemen Informatika Komputer
Volume 8, Nomor 3, Agustus 2024
http://doi.org/10.33395/remik.v8i3.13752
e-ISSN : 2541-1330
p-ISSN : 2541-1332

tidak memberikan dampak ataupun resiko yang besar terhadap *existing code* sebelumnya, baik yang dikembangkan oleh pihak Vendor yang lain atau dari tim Vendor yang sama. Apabila kasus yang terjadi adalah menggantikan baris *syntax of code* yang lama, perlu difikirkan juga dengan alternatif solusi menggunakan teknik *Pilot Code*, yaitu menerapkan dan menguji 2 baris *syntax of code* (*old* dan *new*) sebelum pengguna memutuskan yang terbaik untuk mengeliminasi baris *syntax of code* yang lama (*old*). Terakhir, pastikan ketika join 2 SQL *table* dengan DB (*Database*) yang berbeda, memiliki atribut "*Collation Name*" yang sama, sehingga proses *query* bisa berjalan dengan baik.

Setelah selesai pada tahap sebelumnya, di tahap *Code Designing*, disini tim pengembang perlu merancang baris *syntax of code* dengan berpegangan pada 3 tahap proses utama, yaitu: *Begin of Code, Main Syntax,* dan *End of Code*, seperti contoh dibawah ini:

Begin of Code :

```
USE [DB Selected]
GO

CREATE PROCEDURE <SP name> <variable> <data type>
as

DECLARE <variable> (data type);
SET <variable> = (query select statement)

drop table if exists <TEMP_TABLE>
```

Main Syntax:

```
SELECT (query select statement)
INTO #<TEMP_TABLE>
FROM <TABLE_SELECTED>

WHILE (loop condition)
BEGIN
      (loop statement)
END
```

End of Code :

```
INSERT INTO <final table>
SELECT (final query select statement)
FROM #<TEMP_TABLE>
```

Pada bagian *Begin of Code*, penggunaan *DECLARE* mempermudah pengembang untuk dapat menambahkan variable data yang dibutuhkan, serta sebagai penentu proses eksekusi SP (*Store Procedure*) yang dijalankan melalui SQL *Agent*, baik terjadwal maupun secara *ad-hoc*. Apabila diperlukan penggunaan *temporary table* <TEMP_TABLE> dibagian *Main Syntax*, maka hal ini bisa mengurangi penulisan ke SQL *table* secara langsung tanpa harus menulis ke *storage* DB secara permanen. Penggunaan *WHILE* pada kondisi pengulangan bisa memberikan pilihan terbaik yang dibutuhkan, jika diperlukan penulisan data ke *table* dengan modifikasi secara teratur sesuai dengan kebutuhan. Di bagian *End of Code*, langkah terakhir yang dilakukan adalah dengan menulis data hasil final yang sudah selesai di validasi ke dalam *table* tujuan. Hal ini dapat mengurangi terlalu banyak *data flow* yang dibuat pada saat merancang integrasi data di SSIS dengan menggunakan aplikasi Microsoft Visual Studio.

Tahap *Code Development* inilah merupakan tahapan bagaimana *syntax of code* yang kita buat sesuai dengan Data mapping yang diberikan oleh pihak pengguna bisnis. Data mapping yang dibuat merupakan pemetaan setiap kolom data yang dibutuhkan sehingga membentuk metadata pelaporan.

Remik: Riset dan E-Jurnal Manajemen Informatika Komputer
Volume 8, Nomor 3, Agustus 2024
http://doi.org/10.33395/remik.v8i3.13752
e-ISSN : 2541-1330
p-ISSN : 2541-1332

Tabel 1. *Data Mapping* Sampel

Vendor Field	***Bank's Field***	***Mapping Comments***	***Type***	***Mandatory***	***Default Value***	***Description***
Col A	*Col X*	*Logic Statement*				
Col B	*Col Y*					
...	...					

Sumber : Pribadi 2024

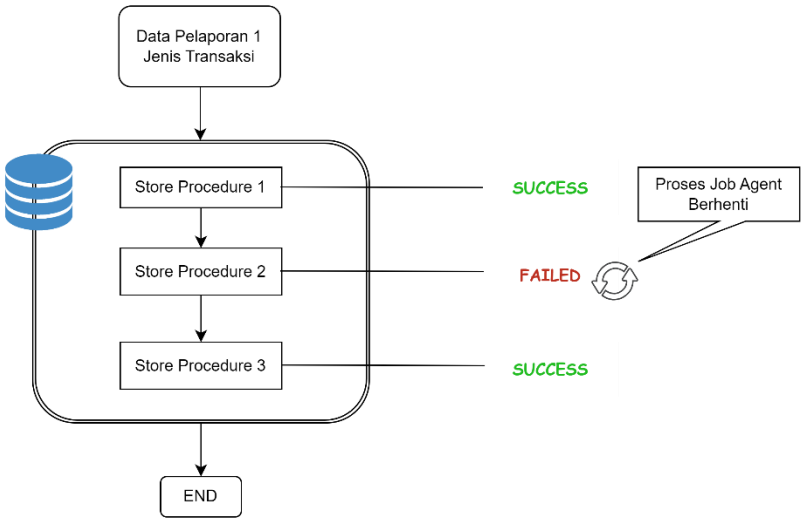
Pada *table* diatas, setiap kolom yang terbentuk memiliki atribut dengan informasi yang dibutuhkan untuk pelaporan. Pihak pengguna akan memetakan kolom tersebut dan memberikannya kembali ke pihak vendor untuk merealisasikannya ke dalam baris *syntax of code*.

```
SELECT (query select statement)
INTO #<TEMP_TABLE>
FROM <TABLE_SELECTED X> A → User Selected
LEFT JOIN <TABLE_SELECTED Y> B → User Selected
WHERE <Condition Statement> → Condition Explained Details

.....
```

Setiap logika kondisi dan bagaimana data diambil dan dibentuk, akan dijelaskan secara rinci di kolom *Mapping Comments*, serta tipe data, opsi pembentukan, nilai tetap dan penjelasan logika akan di jabarkan *how-to* dalam mengisi table oleh pihak vendor pengembang. Namun yang perlu diperhatikan selama pembuatan baris *syntax of code* tersebut, dibutuhkan informasi yang rinci dari pengguna, sehingga tidak ada informasi yang tertinggal dan mengakibatkan adanya *mis-information* di setiap pemetaan data.

Pada tahapan *Code Testing*, *query* tersebut di uji-coba kan dengan menjalankan langsung secara manual dan dengan menjalankan secara *end-to-end* dari fitur layanan SQL *Job Agent*. Hal ini guna untuk memastikan *Store Procedure* yang telah dibuat bisa berjalan dengan baik, serta ketika dijalankan melalui fitur layanan SQL *Job Agent*, setiap SP (*Store Procedure)* yang dijalankan bisa berhasil tanpa *error* yang berarti. Sebagai contoh, apabila pengembang membutuhkan lebih dari 1 SP untuk membentuk data pelaporan salah satu jenis transaksi, maka musti dipastikan *job agent* yang telah dijalankan menjalankan eksekusi setiap langkah proses mulai dari awal sampai akhir (tanpa adanya melewati setiap prosesnya). Walaupun beberapa pengembang mungkin tidak setuju dengan pernyataan ini, tapi nyatanya ketika terjadi *error* di salah satu langkah proses akan berdampak pada pembentuk data pelaporan final.



Gambar 3. Langkah Proses Pengujian *Store Procedure*
Sumber : Pribadi 2024

e-ISSN : 2541-1330
p-ISSN : 2541-1332

Seperti pada gambar 3 diatas, ketika ada salah satu proses *Job Agent* ada yang *error* selama eksekusi berjalan, maka proses berhenti, sampai ada langkah investigasi lebih lanjut ataupun analisa dari tim TI terkait. Nantinya *error* tersebut akan dilakukan Tindakan untuk perbaikan maupun kembali ke langkah sebelumnya.

Setelah selesai proses pengujian, hal perlu diperhatikan sebelum masuk ke dalam langkah *Code Implementation and Maintenance* adalah dengan memastikan setiap *script of code* yang telah dikembangkan dan diujikan merupakan versi final, sehingga terjamin adanya *human error* pada saat migrasi ke lingkungan Produksi. Pengembang perlu memastikan juga bahwa koordinasi dengan tim terkait, maupun internal tim terjalin komunikasi dengan baik. Ketika semua hal tersebut telah selesai, hingga waktunya dari pengguna untuk memastikan setiap data yang dibutuhkan dapat ditampilkan dengan baik, sesuai dengan kebutuhan pelaporan, serta terintegrasi dengan baik dengan data maupun platform sebelumnya. Proses tersebut bisa memakan waktu 1 / 2 hari ataupun seminggu dengan tingkat kerumitan *script of code* yang tinggi. Pengembang pun diberikan waktu kurang lebih selama sebulan lamanya untuk memonitor perkembangan maupun kejadian yang tidak terduga sampai sistem berjalan dengan normal.

V. KESIMPULAN

Dari hasil pembahasan yang disajikan diatas, maka bisa diambil kesimpulan bahwa dalam setiap tahapan yang telah dilalui perlu memastikan setiap prosesnya berjalan dengan baik, sehingga *script of code* yang dikembangkan memiliki kualitas yang baik. Kemudian, pengembang juga memiliki sudut pandang yang berbeda dengan menghadirkan pengalaman lain ketika sebelumnya kinerja proses dibebankan seluruhkan kepada *tools* SSIS (SQL *Server Integration Services*) dan sekarang bisa di pisahkan antara proses utama yang bisa dijalankan menggunakan eksekusi SP (*Store Procedure*) dan proses lainnya dijalankan menggunakan *flow process* pada *tools* SSIS. Ketika terjadi *error*, dan diperlukan perbaikan yang cepat, maka pengembang tidak perlu mengakses ke *tools* SSIS, melainkan langsung perbaiki di SP yang bermasalah.

Selain itu juga penulis menyarankan kepada sesama praktisi SQL di lapangan tidak ada salahnya memperhatikan *style* dalam pembuatan *script*, *tools* untuk memonitor proses eksekusi, *log table error*, serta pertimbangan untuk dibuatkannya UI (*User Interface*) agar bisa dijalankan oleh pengguna untuk memonitor setiap proses eksekusi pada SQL *Job Agent*, sehingga apabila terjadi *error*, maka bisa segera di eskalasi isu tersebut ke tim TI terkait.

VI. REFERENSI

ASHRAF, S. (2020). *Data Staging Area: How It Solves Data Quality Issues*. DataIntegrationInfo. https://dataintegrationinfo.com/data-integration-staging-area/

Dudu, A., Rohmana, A., Mubarok, H., & Gunawan, R. (2019). PENGUKURAN KINERJA STORED PROCEDURE PADA DATABASE RELASIONAL. *Jurnal Siliwangi*, *5*(2). https://doi.org/https://doi.org/10.37058/jssainstek.v5i2.1197

Firdaus, M., & Shedriko, S. (2022). Membangun Integrasi Data Staging Dan Data Mart Pada Perusahaan XYZ. *SEMNASRISTEK (Seminar Nasional Riset Dan Inovasi Teknologi)*, *6*(1), 876–882. https://doi.org/10.30998/semnasristek.v6i1.5822

Firdaus, M., & Shedriko, S. (2024). IMPLEMENTASI ETL DAN PENGOLAHAN DATA PELAPORAN BASEL-III OJK DI BANK UMUM. *SEMNASRISTEK (Seminar Nasional Riset Dan Inovasi Teknologi)*, *8*(1), 44–49. https://doi.org/https://doi.org/10.30998/semnasristek.v8i01.7131

Hastono, T. (2019). Optimasi Query Sistem Informasi Menggunakan Stored Procedure MySQL. *Jurnal Dinamika Informatika*, *8*(2).

e-ISSN : 2541-1330
p-ISSN : 2541-1332

McMaster, K., & Rague, B. (2018). A Comparison of Key Concepts in Data Analytics and Data Science. *Information Systems Education Journal (ISEDJ)*, *16*, 33–40. https://files.eric.ed.gov/fulltext/EJ1173725.pdf

Petkovic, D. (2020). *Microsoft SQL Server 2019 A Begineer's Guide (Seven Edition)* (T. Meister, Ed.; 7th ed.). McGraw-Hill Education.

Shedriko, S., & Firdaus, M. (2022). PENENTUAN KLASIFIKASI DENGAN CRISP-DM DALAM MEMPREDIKSI KELULUSAN MAHASISWA PADA SUATU MATA KULIAH. *SEMNASRISTEK (Seminar Nasional Riset Dan Inovasi Teknologi)*, *6*(1), 826–831. https://doi.org/https://doi.org/10.30998/semnasristek.v6i1.5814

Tabak, I., & Dubovi, I. (2023). What drives the public's use of data? The mediating role of trust in science and data literacy in functional scientific reasoning concerning COVID-19. *Science Education*, *107*(5), 1071–1100. https://doi.org/10.1002/sce.21789

Warman, I. (2021). ANALISA KINERJA QUERY STORED PROCEDURE PADA DATABASE MANAGEMENT SYSTEM (DBMS) MYSQL. *Jurnal Sains Dan Teknologi*, *21*(1). https://db-engines.com

Yuliani, W., Banjarnahor, N., kunci, K., Penelitian Pengembangan, M., & dan Konseling, B. (2021). *METODE PENELITIAN PENGEMBANGAN (RND) DALAM BIMBINGAN DAN KONSELING*. *5*(3). https://doi.org/10.22460/q.v2i1p21-30.642