Business Intelligence
Modul Praktikum

Intro Spoon IDE GUI (Kettle Pentaho Data Integration)

Topik
Pengenalan kettle pentaho data integration sebagai tool ETL, pengenalan lingkungan IDE GUI Spoon.

Tujuan
Setelah menyelesaikan praktikum ini, mahasiswa diharapkan mampu:

1. Memahami Fungsi Kettle sebagai tools ETL (Extract, Transform, Loading) Data.
2. Memahami Komponen dasar Kettle.
3. Memahami pembuatan *parameters* dan *variables* pada kettle.
4. Mampu menggunakan komponen-komponen dasar kettle (Step, Hop, Job).

Alat dan Bahan
- Komputer dengan **Pentaho Data Integration** terinstall
- Pentaho Data Integration dapat diperoleh di: http://community.pentaho.com/

Rangkuman/Teori
[menjelaskan konsep ETL, Model Konseptual Kettle, komponen kettle dan cara penggunaannya: Step, Hop, Job, parameters dan variables pada spoon]

ETL

ETL merupakan singkatan dari *Extract, Transform, Load*. ETL bisa diartikan sebagai sekumpulan proses untuk mengambil dan memproses data dari satu atau banyak sumber menjadi sumber baru.

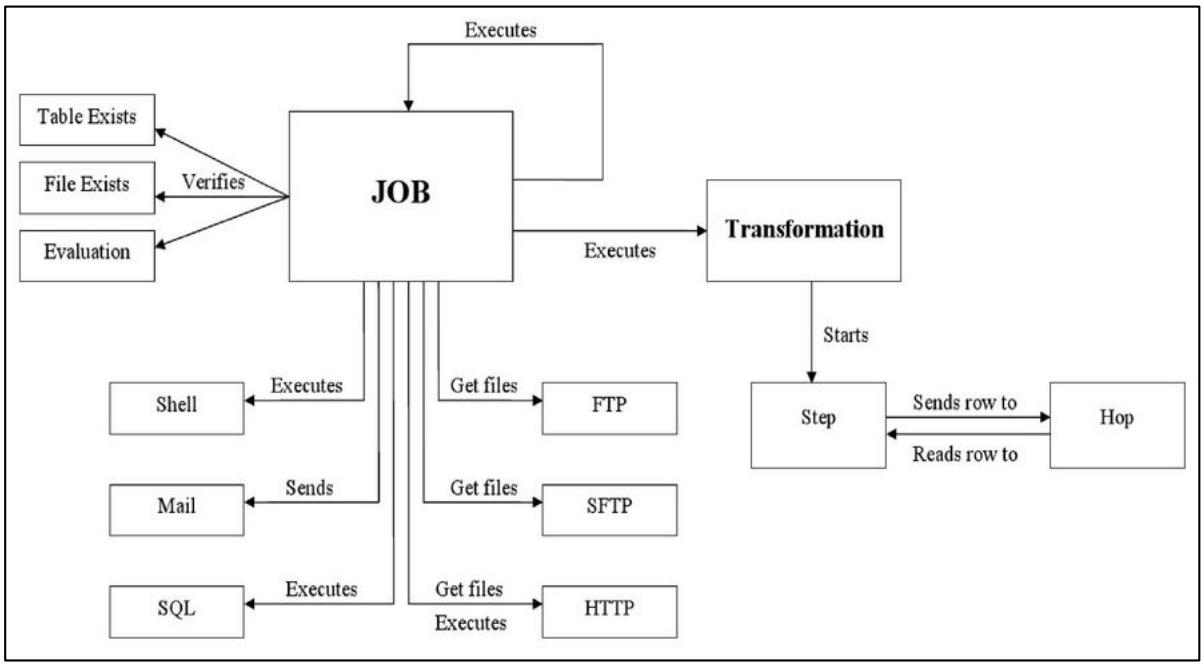
Extract pada ETL: Proses yang diperlukan untuk terhubung dengan beragam sumber data (file spreadsheet, data dari database, data dari web site, dsb) dan membuat data tersebut tersedia badi proses-proses selanjutnya.

Transform pada ETL: Proses atau fungsi-fungsi untuk mengubah data yang masuk menjadi data yang dikehendaki. Fungsi tersebut dapat berupa;

- Pemindahan data,
- Validasi data sesuai aturan yang ditetapkan,
- Modifikasi isi, tipe atau struktur data,
- Integrasi data dari sumber-sumber lain,
- Perhitungan,
- Dsb.

Load pada ETL: Proses yang diperlukan untuk mengisi data ke target. Contoh target; file spreadsheet, database OLAP.

Model Konsep Kettel



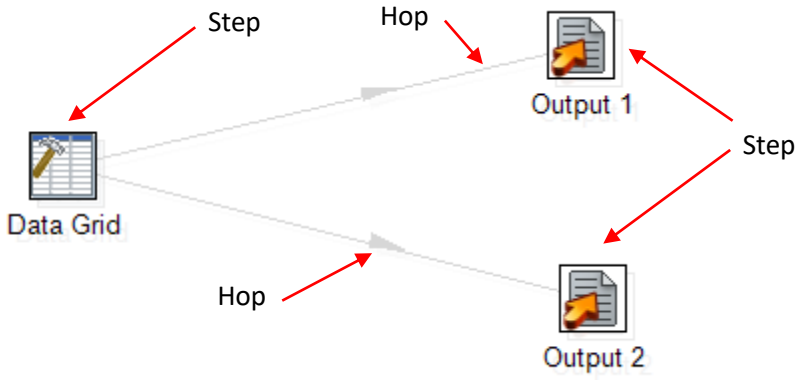
(sumber: http://docplayer.net/15178619-Cloud-computing-with-mysql-and-pentaho-data-integration-matt-casters-chief-data-integration-at-pentaho-kettle-project-founder.html)

Komponen Kettle

Step

Pada kettle terdapat ratusan step yang siap pakai, masing-masing step memiliki fungsi dan tugas tertentu (missal; step untuk membaca file teks, step untuk membaca database, step untuk melakukan perhitungan, step untuk uji kondisi, dsb).

Dalam spoon setiap step diwakili oleh icon-icon yang berbeda-beda. Step harus memiliki nama yang unik (tidak ada aturan baku untuk penamaan step). Step dihubungkan ke step lain menggunakan transformation hop.



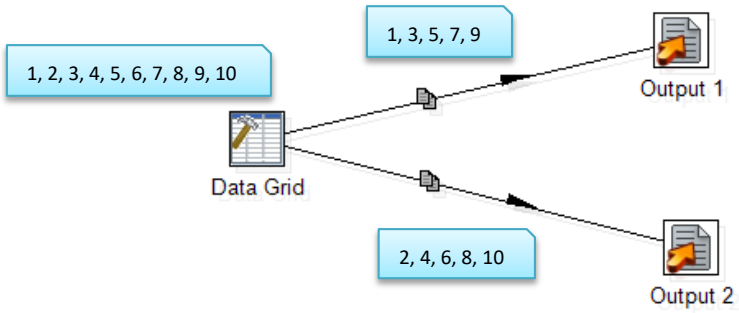
Ilustrasi Step dan Hop

Hop

Hop (*Transformation hop*) merupakan jalur bagi suatu *step* untuk menyalurkan data ke *step* lainnya yang dituju.

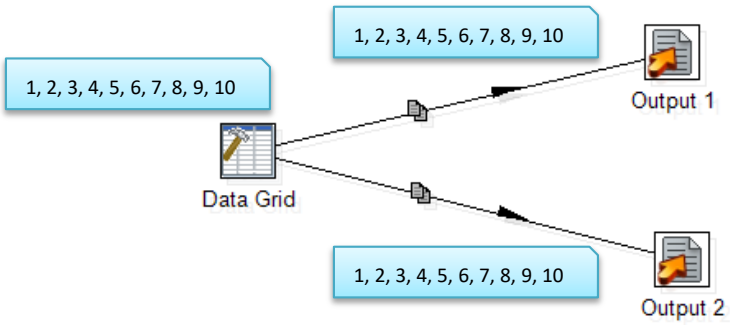
Jika suatu *step* memiliki lebih dari satu *transformation hop* (keluaran) maka baris data (***rows of data***) yang keluar dari step tersebut bisa di-*set* ke:

- *Distribute Data*
  Baris data akan dikirim terpecah ke tiap *transformation hop* (keluaran) secara bergantian.



Ilustrasi *distributed data* pada *transformation hop*

- *Copy data*
  Seluruh data akan diduplikasi ke semua *transformation hop* (keluaran).



Ilustrasi *distributed data* pada *transformation hop*

Hop menyalurkan data dari suatu *step* ke *step* lainnya, terdapat dua jenis data yang dikeluarkan step untuk dialirkan melalui hop, yaitu:

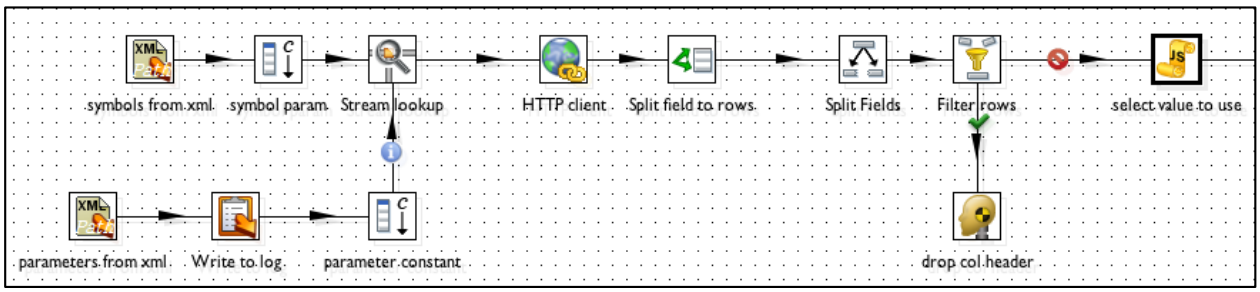
- ***Rows of Data***
  Rows of data adalah baris data yang dialirkan oleh *step* melalui *hop*, satu baris data merupakan kumpulan *field* yang mengandung data. Berikut adalah tipe-tipe data yang mungkin untuk field pada *rows of data*:
  - **String**: teks (tanpa batasan lebar/jumlah karakter)
  - **Number**: angka (double floating point)
  - **Integer**: angka bilangan bulat (signed long integer 64 bit)
  - **BigNumber**: Angka dengan tingkat presisi tidak terbatas
  - **Date**: Tanggal dan waktu (tingkat presisi sampai mili detik)
  - **Boolean**: *true* atau *false*
  - **Binary**: data dengan tipe biner (seperti gambar, suara, video, dsb.).
- ***Row Metadata***
  Row Metadata adalah baris informasi yang menjelaskan tentang baris data (*rows of data*) yang dikeluarkan oleh suatu step.
  Row metadata mengandung informasi sebagai berikut:
  - **Name**: nama field (harus unik)
  - **Data Type**: tipe data
  - **Length**: lebar data (khusus untuk field tipe String dan BigNumber)
  - **Precision**: lebar presisi decimal (khusus untuk field tipe BigNumber)
  - **Mask**: format tampilan data. (digunakan saat terjadi konversi data dari angka ke teks)
  - **Decimal**: simbol decimal yang digunakan (titik . atau koma ,)
  - **Group**: symbol pemisah ribuan (titik . atau koma ,)

Transformation

Transformation adalah komponen *kettle* yang menangani proses manipulasi aliran data, di dalam sebuah *transformation* terdapat satu atau lebih *step*. Semua proses **ETL** dilakukan dalam transformation.

Berikut adalah mekanisme pada *transformation*:

- Semua *step* di dalam *transformation* dijalankan secara paralel.
- *Transformation* akan berhenti setelah semua *step* berhenti bekerja.
- *Transformation* selalu memiliki *step* awal (*step* yang hanya memiliki *hop* keluaran) dan *step* akhir (*step* yang tidak memiliki *hop* keluaran) yang pasti.



(sumber: http://tigerseatgiants.blogspot.co.id/2011/03/when-learning-new-tool-it-is-always-fun.html)

Contoh isi *transformation* pada pentaho data integration

Job

Job adalah komponen *kettle* yang bertugas untuk melakukan pemeliharaan atau pengelolaan proses, seperti;

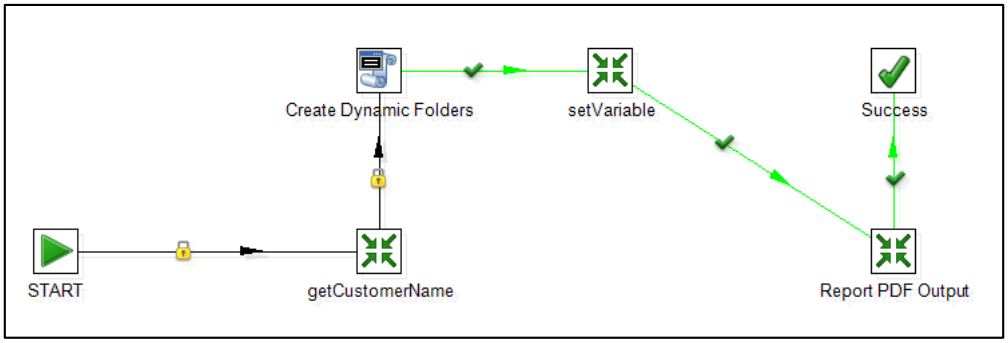
- menentukan apa yang perlu dilakukan jika sesuatu berjalan tidak semestinya,
- memeriksa apakah file yang digunakan sebagai masukan tersedia,
- memeriksa apakah database tujuan ada,
- menentukan tugas-tugas agar dilakukan dalam urutan tertentu.

Job terdiri dari satu atau lebih *job entry* yang dijalankan dalam urutan tertentu. Urutan eksekusi *job entry* ditentukan oleh *job hop*.

Berikut adalah mekanisme pada *job*:

- *Job entry* harus unik.
- dimungkinkan untuk membuat bayangan *job entry* (*shadow copies of a job entry*).
- *Job entry* akan menyalurkan sebuah hasil berupa *object* (*object result*) melalui *job hop*, *object result* bisa berisi baris data.

- *Job entry* akan mengirimkan baris data sekaligus ke *job entry* lainnya setelah *job entry* tersebut selesai memproses semuanya.
- *Job entry* (*by default*) dijalankan secara berurutan.
- Titik awal proses adalah sebuah *job entry* bernama *START*, dan hanya ada satu *job entry START*.



(sumber: http://ankurrajput-pentaho-bi.blogspot.co.id/2014/05/pentaho-report-output-using-pdi.html)

Contoh isi *job* pada pentaho data integration

Job Entry

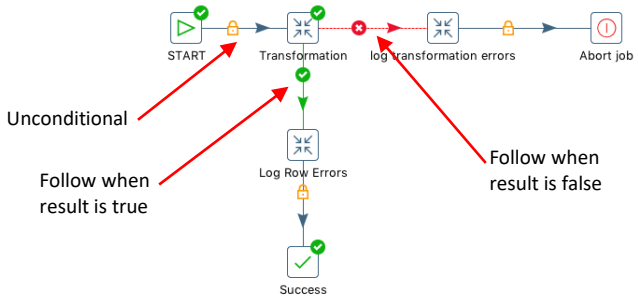
Job Entry adalah building block (pembentuk) inti dari suatu *job*. Setiap *job entry* diwakili oleh ikon-ikon unik dalam spoon.

Job Hop

Job hop digunakan untuk mengalirkan *object* keluaran (*object result*) dari suatu *job entry* ke *job entry* lainnya. *Job hop* menentukan alur eksekusi *job entry*, *job hop* akan melakukan evaluasi pada *object* keluaran (*object result*) dan menentukan arah eksekusi (ke *job entry* mana) selanjutnya.

Tipe evaluasi pada *job hop*:

- **Unconditional**: *job entry* berikutnya akan dieksekusi tanpa melihat apa yang terjadi pada *job entry* sebelumnya.
- **Follow when result is true**:
- **Follow when result is false**:

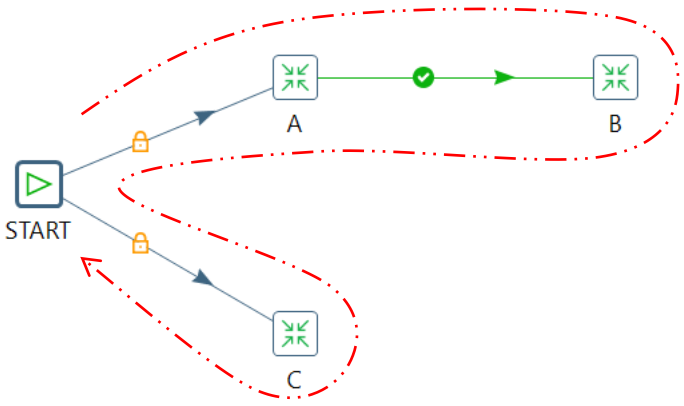


(sumber: http://stackoverflow.com/questions/36731260/pentaho-data-integration-error-handling)

Contoh *job hop* pada pentaho data integration

Eksekusi Backtracking pada *Job Multiple Path*

Pada *Job* dengan *multiple path* digunakan algorithma backtracking. Jalur eksekusi (*execution path*) ditentukan oleh evaluasi keberhasilan dari *job entry*. Status keberhasilan bisa berupa sukses (*success*) atau gagal (*failure*).



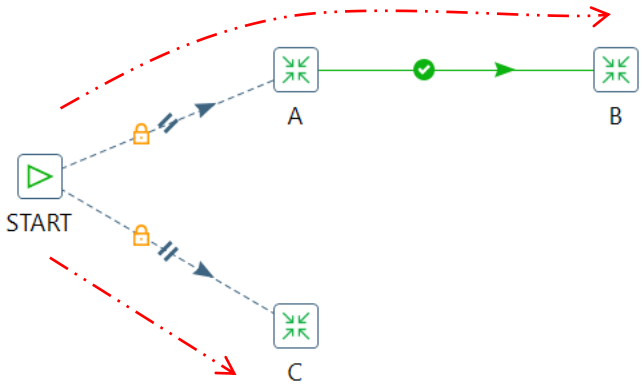
Contoh backtracking pada *job multiple path*

Perhatikan gambar di atas (alur dapat dilihat pada garis putus-putus warna merah), proses mulai dari:

- START, → *job entry* (*transformation*) A di eksekusi,
- setelah A selesai, → *job entry* (*transformation*) B di eksekusi,
- setelah B selesai, karena tidak ada *job entry* (*transformation*) lagi maka kembali ke → A,
- karena tidak ada cabang selain ke B maka kembali ke → START,
- kemudian → *job entry* (*transformation*) C di eksekusi,
- setelah C selesai, karena tidak ada *job entry* (*transformation*) lagi maka kembali ke → START,
- START tidak memiliki *job entry* (*transformation*) lagi maka *job* berakhir.

Eksekusi Paralel pada *Job Multiple Path*

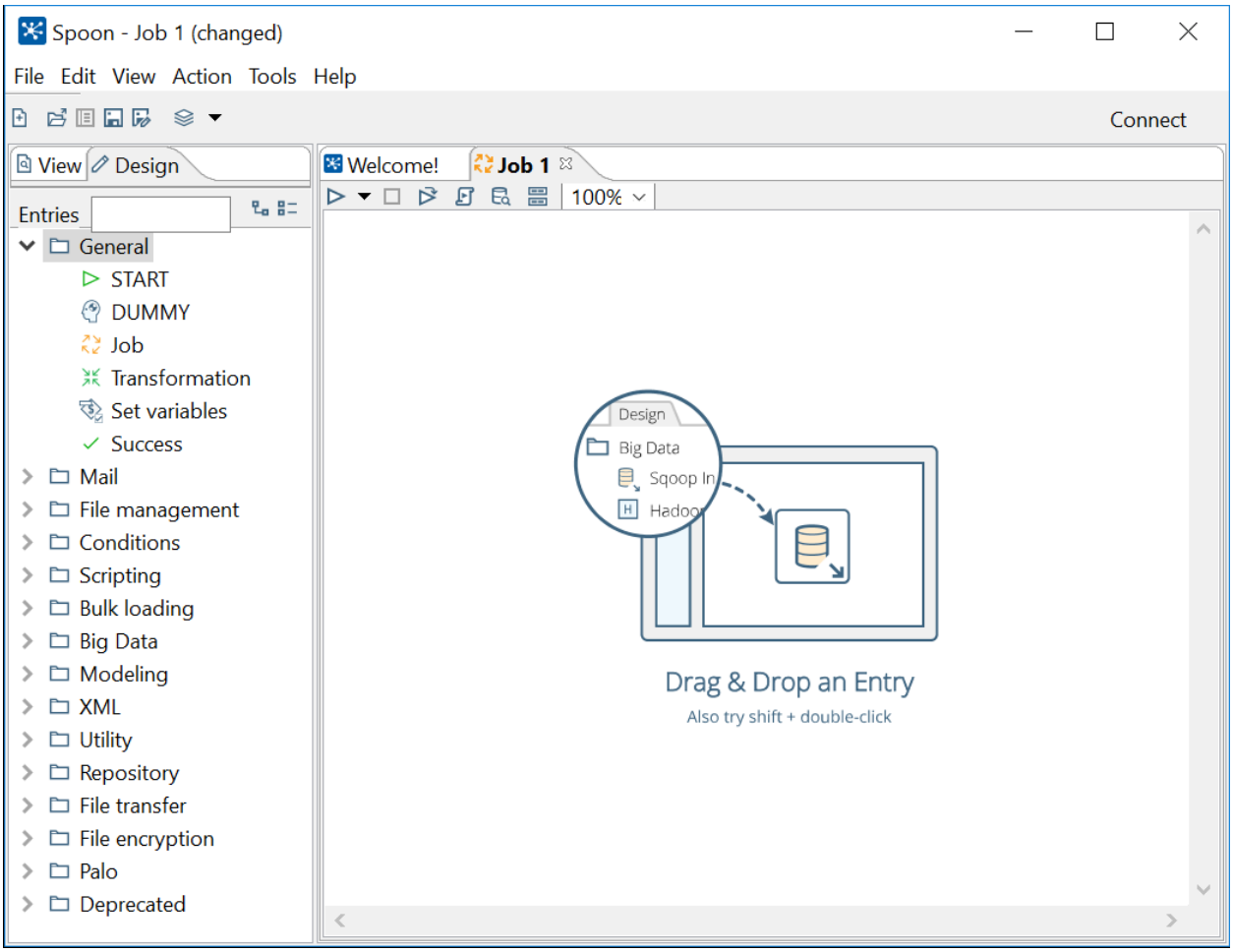
Selalin parallel pada *job multiple path* artinya semua percabangannya pada START akan di eksekusi secara parallel (serentak).



Contoh paralel pada *job multiple path*

Spoon

Spoon merupakan IDE GUI (visual) untuk membuat ETL dengan menggunakan kettle pentaho data integration. Untuk menjalankan spoon cukup dengan menjalankan **spoon.bat** yang terletak pada folder aplikasi pentaho data integration.



Gambar IDE GUI Spoon

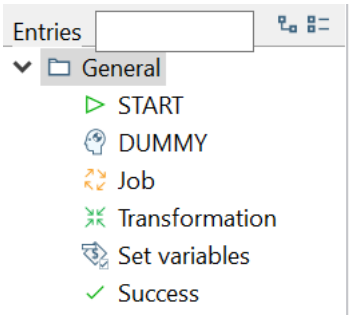
Praktek

Pada bagian ini, akan dibahas mengenai praktek yang akan dilakukan.

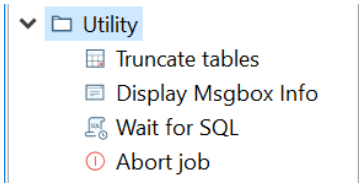
Praktek 1: Menampilkan message box

Ikuti langkah-langkah berikut ini untuk membuat job yang akan menampilkan message box!

Langkah 1: Jalankan spoon.bat
Langkah 2: Buat job baru, file → new → job.
Langkah 3: Click & Drag (atau *shift+double click*) START dari entries **general** ke ruang kerja

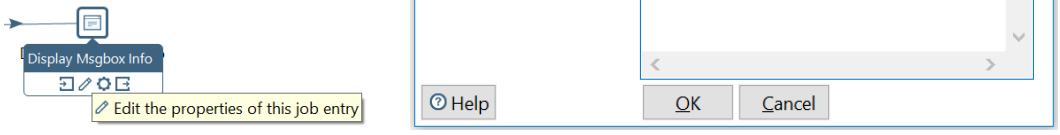


Langkah 4: *Click* & *Drag* (atau *shift+double click*) Display Msgbox Info dari entries **utility** ke ruang kerja.



Langkah 5: Buat *job hop* untuk menghubungkan *job entry* START dengan *job entry* Display Msgbox Info.

Langkah 6: Modifikasi (*edit*) *job entry* Display Msgbox Info, *right click* pada *job entry Display Msgbox Info* → edit. (atau *mouse over* pada job entry → click icon pensil.

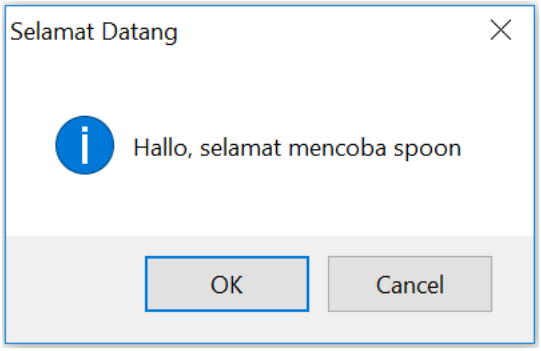


Ubah nama *job entry* menjadi **Sambutan**, ubah judul (*title*) message box yang akan tampil menjadi **Selamat datang**, ubah pesan yang akan ditampilkan menjadi **Hallo, selamat mencoba spoon**.

Langkah 7: Simpan job, file → save

Langkah 8: Jalankan job dengan click tombol play (run)

Message box yang ditampilkan:



Langkah 9: Selesai.

Praktek 2: Mengisi data ke tabel dengan perintah INSERT

Langkah 1: Pastikan anda sudah masuk dalam database yang anda buat pada praktek 1

Langkah 2: Masukan data untuk nim, nama_mhs, kd_prodi, thn_msk ke tabel mahasiswa, dengan menggunakan INSERT (sebanyak 3 row data)

Langkah 3: Tampilkan isi tabel mahasiswa untuk memastikan bahwa data yang anda isi masuk ke tabel mahasiswa

Praktek 3: Mengisi data ke tabel dengan perintah `LOAD DATA`

Langkah 1: Pastikan anda sudah masuk dalam database yang anda buat pada praktek 1

Langkah 2: Masukan data untuk nim, nama_mhs, kd_prodi, thn_msk ke tabel mahasiswa, dengan menggunakan LOAD DATA dari suatu file teks yang format teks-nya sesuai dengan format attribute/field pada tabel.

Langkah 3: Tampilkan isi tabel mahasiswa untuk memastikan bahwa data yang anda isi masuk ke tabel mahasiswa

Praktek 4: Mengisi data ke tabel dengan perintah `LOAD DATA`

1. Tampilkan nim dan nama mahasiswa dari tabel mahasiswa!
2. Tampilkan nim dan kd_prodi dari tabel mahasiswa!
3. Tampilkan seluruh mahasiswa yang tahun masuknya tahun 2013
4. Tampilkan nim dan nama mahasiswa yang kd_prodi-nya SIF.

Analisa Hasil Praktek

Berdasarkan praktek yang telah Anda lakukan, jawablah beberapa pertanyaan berikut ini:

1. Perintah apa yang digunakan untuk mengisi data ke dalam tabel?.

2. Perintah yang mana yang bisa digunakan untuk mengisi tabel jika kita tidak mengisi seluruh isi field tabel?
3. Jelaskan cara mengisi data ke suatu tabel melalui suatu file teks!
4. Bagaimana cara untuk menampilkan informasi hanya sebagian field (field tertentu) dari suatu tabel.
5. Bagaimana cara menampilkan informasi dengan kriteria tertentu dari suatu tabel?

Tugas

1. Buat tabel matakuliah (kd_mk: char(7), nm_mk: varchar(250), sks: int, kd_prodi: char(3))
2. Isi data untuk matakuliah tersebut dengan 10 baris data yang terdapat pada file matakuliah.xlsx dengan menggunakan perintah **INSERT INTO**
3. Untuk baris data berikutnya isikan ke dalam tabel dengan menggunakan perintah **LOAD DATA**