PERANGKAT ETL (*Extract, Transform, Load*)

Berikut adalah beberapa perangkat ETL yang umum digunakan:

1. Pentaho Data Integration (PDI)
2. Talend Open Studio for Data Integration
3. Microsoft SQL Server Integration Service (SSIS)
4. Oracle Data Integration (ODI)

PENTAHO DATA INTEGRATION (PDI)

Pada tahun 2001, Matt Casters seorang programmer dan konsultan *Business IntelligenceI* dari Belgia membuat perangkat ETL yang diberi nama Kettle merupakan singkatan dari KDE (K Desktop Environment) ETTL (Extraction, Transportation, Transformation, dan Loading) Environment yang ditujukan untuk dapat dijalankan di atas Linux. Pada tahun 2006 Kettle resmi goes open source dan resmi diambil alih oleh Pentaho dan Kettle diberi nama Pentaho Data Integration (PDI).

KOMPONEN PDI

PDI/Kettle terdiri dari beberapa komponen yang memiliki nama unik karena bernuansa tempat atau alat masak, bahkan arti Kettle (Bahasa Indonesia) adalah cerek atau teko.

1. Step
   Step adalah blok bangunan inti dari *transformation.* Masing-masing step memiliki fungsi dan tugas tertentu, misalkan ada step yang berfungsi untuk membaca file teks, ada step yang berfungsi untuk membaca database, ada step yang bisa melakukan beragam perhitungan, ada step untuk melakukan uji kondisi, ada step untuk menulis ke database, dan sebagainya. Step harus memiliki nama yang unik. Tidak ada aturan baku untuk penamaan step, jadi kita bebas memberikan nama apa saja. Step dihubungkan dengan step lainnya menggunakan *transformation hop* dalam Transformation.

2. Transformation Hop
   Hampir semua step bisa memiliki lebih dari satu transformation hop keluaran. Jika step memiliki lebih dari satu transformation hop keluaran, maka baris data (*row of data*) yang keluar dari step bisa dikonfigurasi menjadi *distribute data* dan *copy data.* Jika di-set ke distribute data, maka baris data akan ditulis terpecah ke transformation hop keluaran secara bergantian (*round robin*). Sedangkan jika keluaran step dikonfigurasi menjadi copy data, seluruh data akan diduplikasi ke semua transformation hop keluaran.

3. Row of Data
   Baris data (*row of data*) adalah data yang dilewatkan dari satu step ke step lain melalui transformation hop.
   Sebuah baris data berupa kumpulan nol atau lebih fields yang mengandung data. Data pada fields bisa bertipe sebagai berikut:

1. Sting: Sembarang tipe karakter tanpa batasan lebar.
2. Number: Double precision floating point.
3. Integer: Signed long integer (64 bit).
4. BigNumber: Bilangan dengan presisi tak terbatas.
5. Date: Nilai tanggal dan waktu dengan presisi sampai mili detik.
6. Boolean: Nilai Boolean, dapat berisi true atau false.
7. Binary: Sembarang tipe binary seperti gambar, suara, video, dan lain sebagainya.

4. Row Metadata

   Setiap step memiliki kemampuan untuk memberikan penjelasan mengenai baris yang dikeluarkannya. Penjelasan (description) dari baris ini dikenal dengan *row metadata.*

   Row metadata mengandung informasi sebagai berikut:

   1. Name: Nama field dan harus unik.
   2. Data type: Tipe data, misalkan String, Number, dan lain sebagainya.
   3. Length: Lebar data khusus untuk tipe data string dan BigNumber.
   4. Precision: Lebar presisi decimal khusus untuk tipe data BigNumber.
   5. Mask: Menentukan format tampulan data. Mask ini digunakan saat terjadi konversi data dari tipe data Number, Integer, BigNumber, dan Date menjadi tipe data String.
   6. Decimal: Menentukan symbol decimal, biasanya bisa berupa titik (.) atau koma (,).
   7. Group: Menentukan symbol pemisah ribuan, biasanya bisa berupa titik (.) atau koma (,).

5. Transformation

   Sesuai dengan namanya, transformation adalah komponen Kettle yang menangani proses manipulasi aliran data. Semua proses ETL dilakukan dalam transformation. Dalam sebuah transformation, terdapat satu atau lebih step.

   Satu hal yang perlu dicatat adalah transformation menjalankan semua step secara parallel. Ketika transformation dijalankan, maka serentak semua step bekerja pada *thread* masing-masing untuk terus memantau transformation hop masukan, memproses data yang masuk, dan mengeluarkannya melalui transformation hop keluaran sampai tidak ada lagi baris data yang masuk, dan saat itu step akan berhenti bekerja. Proses transformation akan berhenti setelah semua step berhenti bekerja. Walaupun dijalankan secara parallel, namun transformation selalu memiliki step awal dan step akhir yang pasti.

6. Job

   Dalam sebuah proyek ETL, sangat perlu adanya tugas pemeliharaan. Tugas pemeliharaan di sini misalkan menentukan apa yang perlu dilakukan jika sesuatu berjalan tidak semestinya, memeriksa apakah file yang digunakan sebagai masukan tersedia, apakah database tujuan ada, atau menentukan tugas-tugas agar dilakukan dalam urutan tertentu.

   Misalkan dalam sebuah proyek ETL ditentukan harus memproses file yang dihasilkan dari sebuah aplikasi, dan file tersebut disimpan dalam folder tertentu. Kettle tidak pernah tahu, kapan aplikasi tersebut mulai dan selesai membentuk file yang diperlukan, sehingga harus ada tugas khusus yang memeriksa apakah file yang dimaksud sudah tersedia atau belum dalam folder yang sudah ditentukan, jika belum tersedia maka proses ETLnbelum bisa dijalankan.

Bagaimana bisa dijalankan jika sumber masukannya tidak ada? Proses seperti ini tidak bisa dilakukan oleh transformation karena transformation berjalan secara parallel, dan disinilah tugas job berperan. Job terdiri dari satu atau lebih *job entry* yang dijalankan dalam urutan tertentu. Urutan eksekusi job entry ditentukan oleh *job hop.*

7. Job Entry

   Job entry adalah blok bangunan inti dari sebuah job. Seperti step pada transformation, setiap job entry juga diwakili oleh ikon-ikon unik dalam Spoon. Namun demikian, terdapat perbedaan Antara step dan job entry sebagai berikut:

   1. Walaupun nama dari job entry harus unik, namun bisa dimungkinkan untuk membuat bayangan job entry (*shadow copies of a job entry*). Dengan cara ini, dimungkinkan untuk menempatkan job entry yang sama pada beberapa lokasi dalam sebuah job. Karena berupa bayangan, maka jika kita menyunting salah satu job entry tersebut, maka semua job entry bayangan lainnya akan ikut tersunting.
   2. Sebuah job entry akan melewatkan sebuah obyek hasil (*result object*) ke job entry lainnya melalui job hop. Di dalam obyek hasil tersebut, bisa berisi baris data. Jika pada step baris data dikirim ke step lainnya dalam bentuk *stream*, maka pada job entry tidak demikian. Job entry akan mengirimkan baris data sekaligus ke job entry lainnya setelah job entry tersebut selesai memproses semuanya.
   3. Semua job entry dijalankan secara berurutan secara default. Namun pada kasus-kasus tertentu, bisa saja job entry dijalankan secara parallel. Karena job entry dijalankan secara berurutan, maka kita harus menentukan sebuah titik awal proses. Titik awal proses adalah sebuah job entry bernama **START**. Kita hanya bisa menempatkan satu buah job entry **START** dalam sebuah job.

8. Job Hop

   Job hop digunakan dalam job untuk menentukan urutan eksekusi job entry. Setiap job hop akan melakukan evaluasi pada hasil keluarannya dan ini menentukan arah eksekusi selanjutnya.

   Tipe evaluasi bisa berupa sebagai berikut:

   1. **Unconditional:** Artinya, job entry berikutnya akan dieksekusi tanpa melihat apa yang terjadi pada job entry sebelumnya. Tipe evaluasi ini ditandai dengan ikon kunci gembok berwarna kuning pada job hop berwarna hitam.
   2. **Follow when result is true:** Job hop ini akan dieksekusi jika hasil job entry sebelumnya memberikan nilai evaluasi true. Tipe evaluasi ini ditandai dengan ikon sukses berwarna hijau pada job hop berwarna hijau.
   3. **Follow when result is false:** Job hop ini akan dieksekusi jika hasil job entry sebelumnya memberikan nilai evaluasi false. Tipe evaluasi ini ditandai dengan ikon stop merah pada job hop berwarna merah.

9. Job Multiple Paths and Backtracking

   Job entries dijalankan menggunakan algoritma yang disebut *backtracking algorithm.* Jalur eksekusi (*execution path*) ditentukan oleh evaluasi keberhasilan dari job entry. Status keberhasilan bisa berupa sukses (*success*) atau gagal (*failure*).

Untuk memahami algoritma backtracking, Job entry A, B, dan C dijalankan secara berurutan sebagai berikut:

1. Job entry **START** mejadi titik awal proses. Job entry ini akan mencari semua job entry yang terhubung dengannya, dan menemukan A dan C.
2. A dijalankan.
3. Setelah A menyelesaikan tugasnya, A akan mencari semua job entry yang terhubung dengannya, dan A menemukan B.
4. B dijalankan.
5. Setelah B menyelesaikan tugasnya, B akan mencari semua job entry yang terhubung dengannya, namun tidak satu job entry pun yang ditemukan. Karena tidak ada lagi job entry pada jalur tersebut, maka di sini proses backtracking terjadi, jalur akan mundur kembali ke A, lalu ke job entry **START.**
6. Job entry **START** memiliki satu job entry lagi yang belum dijalankan, yaitu C.

10. Job Parallel Execution

    Adakalanya kita perlu menjalankan job entry secara parallel, dan ini bisa dilakukan dengan memberi tahu job entry bersangkutan agar job entry berikutnya yang terhubung dengannya secara parallel.

11. Parameters and Variables

    Contoh:

    Anda membuat proyek ETL dengan Windows, yang terdapat banyak job entry (banyak step) yang merujuk pada suatu direktori dan digunakan sebagai tempat menampung beragam input file.

    Namun karena sesuatu hal, Anda harus menjalankan ETL yang Anda buat dilingkungan Linux, dan lokasi tempat menampung input file ada ditempat yang berbeda dengan ETL Windows.

    Jika Anda harus menyunting satu demi satu job entry atau step untuk mengubah direktori Windows ke Linux, maka itu adalah pekerjaan buruk karena ETL menjadi sulit terpelihara. Bisa dibayangkan waktu yang diperlukan untuk menyunting file satu demi satu, belum lagi kalau ada file yang terlewatkan untuk disunting, ini bisa menjadi bug dalam ETL Anda.

    Oleh karena itu, perlu adanya mekanisme parameter dalam ETL, dan Kettle menyediakan fasilitas tersebut agar aspek-aspek tertentu dalam proyek ETL dapat diparameterkan. Dalam Kettle, Anda bisa melakukan hal tersebut dengan bantuan variable. Pada kasus di atas, lokasi input file bisa disimpan dalam variabel, sehingga jika lokasi input file berubah, cukup mengubah nilai variabelnya saja.

12. Spoon

    Merupakan *Integrated Development Environment* (IDE) yang berupa *Graphical User Interface* (GUI). Digunakan untuk merancang, menyunting, dan menjalankan Job dan Transformation.

13. Pan

    Pan adalah *command line tool* yang khusus digunakan untuk menjalankan Transformation. Biasanya digunakan jika ingin menjalankan Transformation melalui mekanisme penjadwalan (*scheduler*).

14. Kitchen

    Kitchen adalah *command line tool* yang khusus digunakan untuk menjalankan job. Biasanya digunakan jika ingin menjalankan job melalui mekanisme penjadwalan (*scheduler*).

15. Carte

    Carte adalah server (Jetty HTTP Server) yang berjalan di background untuk memantau ada tidaknya permintaan menjalankan sebuah job.

    Umumnya digunakan untuk meningkatkan kinerja Kettle dengan cara membagi beban kerja pada beberapa node Carte (*master* dan *slave*) dalam lingkungan cluster Kettle.