Nama : - Afina Putri Dayanti (825200049)

* Eric Anthony (825200050)

Jurusan : Sistem Informasi

Mata Kuliah : ETL and Data Warehouse

**Data Warehouse Architecture**

Sistem gudang data memiliki dua arsitektur utama: arsitektur aliran data dan arsitektur sistem. Arsitektur aliran data adalah tentang bagaimana penyimpanan data diatur dalam gudang data dan bagaimana data mengalir dari sistem sumber ke pengguna melalui penyimpanan data ini. Arsitektur sistem adalah tentang konfigurasi fisik server, jaringan, perangkat lunak, penyimpanan, dan klien. Dalam bab ini, saya akan membahas arsitektur aliran data terlebih dahulu dan kemudian arsitektur sistem. Secara khusus, saya akan memulai bab ini dengan membahas apa itu arsitektur aliran data, apa tujuannya, dan apa saja komponennya. Kemudian saya akan membahas empat arsitektur aliran data: single DDS, NDS + DDS, ODS + DDS, dan federated data warehouse. Tiga yang pertama menggunakan model dimensi sebagai penyimpanan data back-end mereka, tetapi mereka berbeda di penyimpanan data tingkat menengah. Arsitektur data warehouse federasi terdiri dari beberapa gudang data yang terintegrasi oleh lapisan pengambilan data. Saya kemudian akan membahas arsitektur sistem, yang perlu dirancang agar sesuai dengan arsitektur aliran data. Jadi, arsitektur aliran data ditentukan terlebih dahulu sebelum Anda mendesain arsitektur sistem. Arsitektur sistem mempengaruhi kinerja sebenarnya dari sistem gudang data yang dikirimkan ke pengguna akhir. Menjelang akhir bab ini, saya akan memperkenalkan studi kasus yang akan kita gunakan di seluruh buku ini. Dalam bab-bab selanjutnya, saya akan menunjukkan cara membangun gudang data untuk perusahaan dalam studi kasus, Amadeus Entertainment Group. Anda dapat menggunakan studi kasus ini untuk mempelajari semua aspek pergudangan data yang tercakup dalam buku ini: arsitektur, metodologi, persyaratan, pemodelan data, desain basis data fisik, ETL, kualitas data, metadata, pelaporan, basis data multidimensi, BI, CRM, pengujian, dan administrasi.

**Arsitektur Aliran Data**

Dalam data warehousing, arsitektur aliran data adalah konfigurasi penyimpanan data dalam sistem gudang data, bersama dengan pengaturan bagaimana data mengalir dari sistem sumber melalui penyimpanan data ini ke aplikasi yang digunakan oleh pengguna akhir. Ini termasuk menunjukkan aliran data dikendalikan, dicatat, dan dipantau, serta mekanisme untuk memastikan kualitas data di penyimpanan data. Saya membahas arsitektur aliran data secara singkat di Bab 1, tetapi dalam bab ini saya akan membahasnya secara lebih rinci, bersama dengan empat arsitektur aliran data: DDS tunggal, NDS + DDS, ODS + DDS, dan gudang data federasi. Arsitektur aliran data berbeda dari arsitektur data. Arsitektur data adalah tentang bagaimana data diatur di setiap penyimpanan data dan bagaimana penyimpanan data dirancang untuk mencerminkan proses bisnis. Kegiatan untuk menghasilkan arsitektur data dikenal sebagai pemodelan data. Saya akan tidak membahas arsitektur data dan pemodelan data dalam bab ini. Saya akan membahas topik tersebut di Bab 5. Penyimpanan data adalah komponen penting dari arsitektur aliran data. Saya akan memulai diskusi tentang arsitektur aliran data dengan menjelaskan apa itu penyimpanan data. Penyimpanan data adalah satu atau lebih database atau file yang berisi data gudang data, disusun dalam format tertentu dan terlibat dalam proses gudang data. Berdasarkan aksesibilitas pengguna, Anda dapat mengklasifikasikan penyimpanan data gudang data menjadi tiga jenis:

* Penyimpanan data yang menghadap pengguna adalah penyimpanan data yang tersedia untuk pengguna akhir dan ditanyakan oleh pengguna akhir dan aplikasi pengguna akhir.
* Penyimpanan data internal adalah penyimpanan data yang digunakan secara internal oleh komponen gudang data untuk tujuan mengintegrasikan, membersihkan, mencatat, dan menyiapkan data, dan tidak terbuka untuk kueri oleh pengguna akhir dan aplikasi pengguna akhir.
* Penyimpanan data hibrid digunakan untuk mekanisme gudang data internal dan untuk permintaan oleh pengguna akhir dan aplikasi pengguna akhir.

Penyimpanan data master adalah penyimpanan data yang menghadap pengguna atau hibrid yang berisi kumpulan data lengkap dalam gudang data, termasuk semua versi dan semua data historis. Berdasarkan format data, Anda dapat mengklasifikasikan penyimpanan data gudang data menjadi empat jenis:

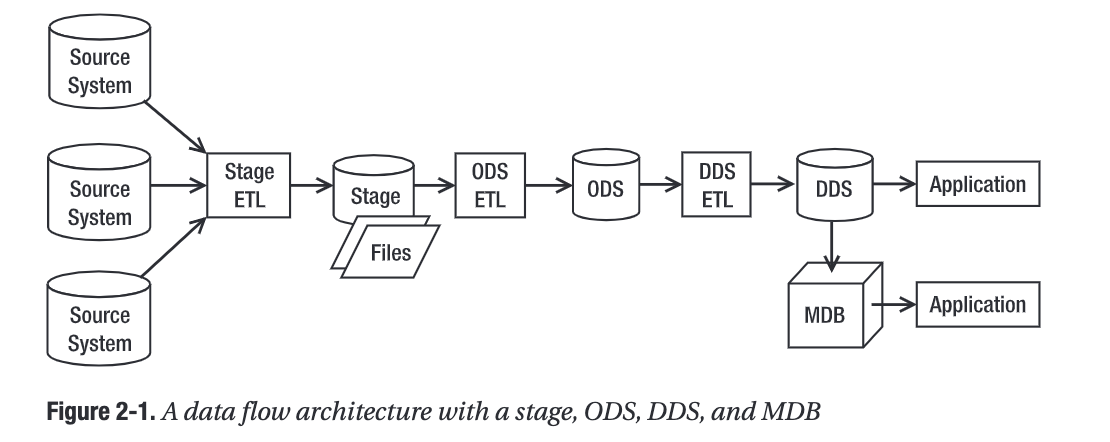
* Tahap adalah penyimpanan data internal yang digunakan untuk mengubah dan menyiapkan data yang diperoleh dari sistem sumber, sebelum data dimuat ke penyimpanan data lain di gudang data.
* Penyimpanan data ternormalisasi (NDS) adalah penyimpanan data master internal dalam bentuk satu atau lebih database relasional yang dinormalisasi dengan tujuan untuk mengintegrasikan data dari berbagai sistem sumber yang ditangkap dalam suatu tahapan, sebelum data dimuat ke penyimpanan data yang dihadapi pengguna.
* Operational Data Store (ODS) adalah penyimpanan data hybrid dalam bentuk satu atau lebih database relasional yang dinormalisasi, yang berisi data transaksi dan versi terbaru dari data master, untuk tujuan mendukung aplikasi operasional.
* Penyimpanan data dimensi (DDS) adalah penyimpanan data yang menghadap pengguna, dalam bentuk satu atau lebih database relasional, di mana data disusun dalam format dimensi untuk tujuan mendukung kueri analitik.

Saya membahas istilah relasional, normalisasi, denormalisasi, dan dimensi di Bab 1, tetapi saya akan mengulangi definisi di sini secara singkat:

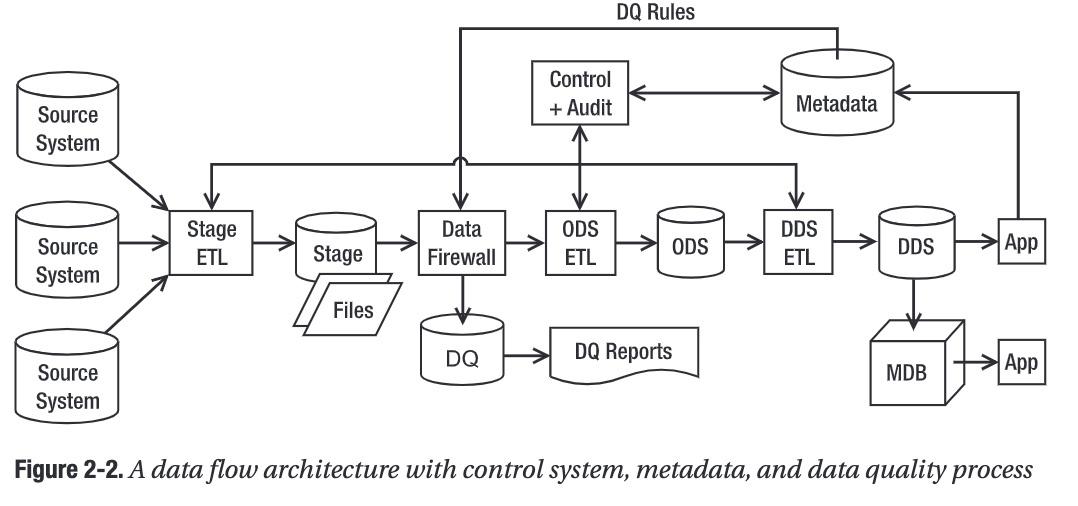
* Database relasional adalah database yang terdiri dari tabel entitas dengan hubungan induk-anak di antara mereka.
* Basis data yang dinormalisasi adalah basis data dengan sedikit atau tanpa redundansi data dalam bentuk normal ketiga atau lebih tinggi.
* Basis data yang didenormalisasi adalah basis data dengan beberapa redundansi data yang belum melalui proses normalisasi.
* Database dimensional adalah database denormalisasi yang terdiri dari tabel fakta dan tabel dimensi umum yang berisi pengukuran peristiwa bisnis, dikategorikan berdasarkan dimensinya.

Beberapa aplikasi memerlukan data dalam bentuk database multidimensi (MDB) daripada database relasional. MDB adalah bentuk database di mana data disimpan dalam sel dan posisi setiap sel ditentukan oleh sejumlah variabel yang disebut dimensi. Setiap sel mewakili peristiwa bisnis, dan nilai dimensi menunjukkan kapan dan di mana peristiwa ini terjadi. MDB diisi dari DDS.Extract, transform, and load (ETL) adalah sistem yang memiliki kemampuan untuk membaca data dari satu penyimpanan data, mengubah data, dan memuatnya ke penyimpanan data lain. Penyimpanan data tempat ETL membaca data disebut sumber, dan penyimpanan data tempat ETL memuat data disebut target.

Gambar 2-1 menunjukkan arsitektur aliran data dengan empat penyimpanan data: stage, ODS, DDS, dan MDB.



Panah pada Gambar 2-1 menunjukkan aliran data. Data mengalir dari sistem sumber ke suatu tahap, ke ODS, ke DDS, dan kemudian ke aplikasi yang digunakan oleh pengguna. Pada Gambar 2-1 ada tiga paket ETL antara empat penyimpanan data. Stage ETL mengambil data dari sistem sumber dan memuatnya ke dalam stage. ODS ETL mengambil data dari sebuah panggung dan memuatnya ke dalam ODS. DDS ETL mengambil data dari ODS dan memuatnya ke DDS. Paket ETL terdiri dari beberapa proses ETL. Proses ETL adalah program yang merupakan bagian dari paket ETL yang mengambil data dari satu atau beberapa sumber dan mengisi satu tabel target. Proses ETL terdiri dari beberapa langkah. Langkah adalah komponen dari proses ETL yang melakukan tugas tertentu. Contoh langkah adalah mengekstrak data tertentu dari sumber datastore atau melakukan transformasi data tertentu. Paket ETL di data warehouse dikelola oleh sistem kontrol, yaitu sistem yang mengatur waktu setiap paket ETL berjalan, mengatur urutan eksekusi proses dalam paket ETL, dan menyediakan kemampuan untuk me-restart paket ETL dari titik kegagalan. . Mekanisme untuk mencatat hasil dari setiap langkah proses ETL disebut audit ETL. Contoh hasil yang dicatat oleh audit ETL adalah berapa banyak catatan yang diubah atau dimuat dalam langkah itu, waktu langkah itu dimulai dan selesai, dan pengidentifikasi langkah sehingga Anda dapat melacaknya saat debugging atau untuk tujuan audit. Saya akan membahas lebih lanjut tentang ETL di Bab 7 dan 8. Deskripsi setiap proses ETL disimpan dalam metadata. Ini termasuk sumber tempat mengekstrak data, target memuat data ke dalamnya, transformasi yang diterapkan, proses induk, dan jadwal setiap proses ETL yang seharusnya dijalankan. Dalam data warehousing, meta-data adalah penyimpanan data yang berisi deskripsi struktur, data, dan proses di dalam data warehouse. Ini termasuk definisi dan pemetaan data, struktur data dari setiap penyimpanan data, struktur data dari sistem sumber, deskripsi setiap proses ETL, deskripsi aturan kualitas data, dan log dari semua proses dan aktivitas di gudang data. Saya akan membahas lebih lanjut tentang metadata di Bab 10. Proses kualitas data adalah kegiatan dan mekanisme untuk memastikan data dalam datawarehouse benar dan lengkap. Hal ini biasanya dilakukan dengan memeriksa data dalam perjalanannya ke dalam gudang data. Proses kualitas data juga mencakup mekanisme pelaporan data yang buruk dan perbaikannya. Firewall data adalah program yang memeriksa apakah data yang masuk sesuai dengan aturan kualitas data. Aturan kualitas data adalah kriteria yang memverifikasi data dari sistem sumber berada dalam kisaran yang diharapkan dan dalam format yang benar. Database kualitas data adalah database yang berisi data masuk yang melanggar aturan kualitas data. Laporan kualitas data membaca pelanggaran kualitas data dari database kualitas data (DQ) dan menampilkannya di atas kertas atau di layar. Saya akan membahas lebih lanjut tentang kualitas data di Bab 9. Gambar 2-2 menunjukkan arsitektur aliran data lengkap dengan sistem kontrol, metadata, dan komponen proses kualitas data.



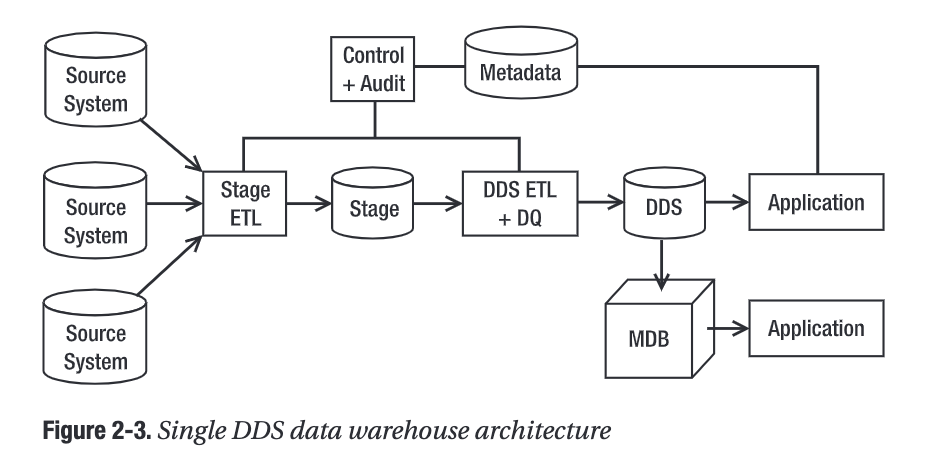
Arsitektur aliran data adalah salah satu hal pertama yang perlu Anda putuskan saat membangun sistem datawarehouse karena arsitektur aliran data menentukan komponen apa yang perlu dibangun dan oleh karena itu memengaruhi rencana dan biaya proyek. Arsitektur aliran data menunjukkan bagaimana data mengalir melalui penyimpanan data dalam gudang data. Arsitektur aliran data dirancang berdasarkan persyaratan data dari aplikasi, termasuk persyaratan kualitas data. Aplikasi data warehouse membutuhkan data dalam format yang berbeda. Format ini menentukan penyimpanan data yang perlu Anda miliki. Jika aplikasi memerlukan format dimensional, maka Anda perlu membangun DDS. Jika aplikasi memerlukan format yang dinormalisasi untuk tujuan operasional, maka Anda perlu membangun ODS. Jika aplikasi membutuhkan format multidimensi, maka Anda perlu membangun MDB. Setelah Anda menentukan penyimpanan data yang perlu Anda bangun, Anda dapat mendesain ETL untuk mengisi penyimpanan data tersebut. Kemudian Anda membangun mekanisme kualitas data untuk memastikan data di gudang data sudah benar dan lengkap. Dalam empat bagian berikut, saya akan membahas empat arsitektur aliran data dengan kelebihan dan kekurangannya:

* Arsitektur DDS tunggal memiliki tahap dan penyimpanan data DDS.
* Arsitektur NDS + DDS memiliki stage, NDS, dan penyimpanan data DDS.
* Arsitektur ODS + DDS memiliki stage, ODS, dan penyimpanan data DDS.
* Arsitektur gudang data federasi (FDW) terdiri dari beberapa gudang data yang terintegrasi oleh lapisan pengambilan data.

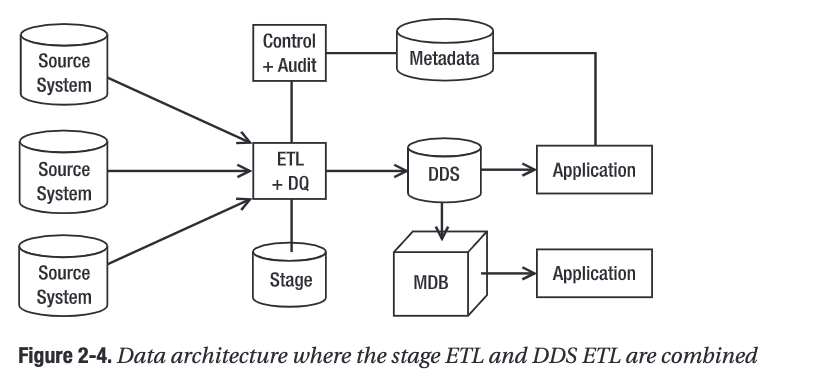
Keempat arsitektur aliran data ini hanyalah contoh. Saat membangun gudang data, Anda perlu mendesain arsitektur aliran data agar sesuai dengan persyaratan data dan persyaratan kualitas data proyek. Keempat arsitektur ini sama sekali tidak lengkap; Anda harus merancang arsitektur aliran data Anda sendiri. Misalnya, apakah mungkin untuk memiliki arsitektur aliran data tanpa panggung? Ya, itu mungkin, jika Anda dapat melakukan transformasi dengan cepat di memori ETLserver, terutama jika Anda memiliki volume data yang rendah. Bisakah Anda memiliki arsitektur aliran data tanpa DDS? Ya, Anda dapat memiliki penyimpanan data stage + NDS hanya jika Anda ingin memiliki gudang data yang dinormalisasi daripada yang berdimensi. Apakah mungkin untuk memiliki dua DDS? Ya, tentu saja Anda dapat memiliki beberapa DDS—Anda dapat memiliki sebanyak yang Anda butuhkan. Sekarang saya akan membahas keempat arsitektur aliran data ini satu per satu.

**Single DDS**

Di bagian ini, Anda akan belajar tentang arsitektur aliran data sederhana yang hanya terdiri dari dua penyimpanan data: stage dan DDS. Dalam arsitektur ini, penyimpanan gudang data inti dalam format dimensi. Dalam arsitektur DDS tunggal, Anda memiliki penyimpanan data satu dimensi. DDS terdiri dari satu atau beberapa data mart dimensi. Data mart dimensi adalah sekelompok tabel fakta terkait dan tabel dimensi terkait yang berisi pengukuran peristiwa bisnis, dikategorikan berdasarkan dimensinya. Paket ETL mengekstrak data dari sistem sumber berbeda dan menempatkannya di atas panggung. Astage adalah tempat Anda menyimpan data yang Anda ekstrak dari sistem penyimpanan untuk sementara, sebelum memprosesnya lebih lanjut. Tahap diperlukan ketika transformasi kompleks (dengan kata lain, tidak dapat dilakukan dengan cepat dalam satu langkah dalam memori), ketika volume data besar (dengan kata lain, tidak cukup untuk dimasukkan ke dalam memori), atau ketika data dari beberapa sistem sumber tiba pada waktu yang berbeda (dengan kata lain, tidak diekstraksi oleh ETL tunggal). Tahap juga diperlukan jika Anda perlu meminimalkan waktu untuk mengekstrak data dari sistem sumber. Dengan kata lain, proses ETL membuang data yang diekstraksi pada disk dan memutuskan koneksi dari sistem sumber sesegera mungkin, dan kemudian pada waktu mereka sendiri mereka dapat memproses data. Bentuk fisik dari suatu tahapan dapat berupa database atau file. ETL yang mengekstrak data dari sistem sumber memasukkan data ke dalam database atau menulisnya sebagai file. Paket ETL kedua mengambil data dari panggung, mengintegrasikan data dari sistem sumber yang berbeda, menerapkan beberapa aturan kualitas data, dan memasukkan data gabungan ke dalam DDS. Gambar 2-3 menjelaskan model umum arsitektur ini.



Pada Gambar 2-3, kotak “Kontrol + Audit” berisi sistem kontrol dan audit ETL, seperti yang saya bahas sebelumnya. Mereka mengelola proses ETL dan mencatat hasil eksekusi ETL. Database themetadata berisi deskripsi struktur, data, dan proses di dalam data warehouse. Aplikasi data warehouse, seperti laporan business intelligence (BI), membaca data di DDS dan membawa data ke pengguna. Data dalam DDS juga dapat diunggah ke database multidimensi, seperti Layanan Analisis SQL Server, dan kemudian diakses oleh pengguna melalui OLAP dan aplikasi penambangan data. Beberapa arsitek ETL lebih memilih untuk menggabungkan dua paket ETL yang mengelilingi panggung pada Gambar 2-3 menjadi satu paket, seperti yang digambarkan pada Gambar 2-4. Pada Gambar 2-4, tahapan ETL, DDSETL, dan proses kualitas data digabungkan menjadi satu ETL. Keuntungan menggabungkannya ke dalam satu paket adalah memiliki kendali lebih besar atas waktu ketika data ditulis dan diambil dari panggung. Secara khusus, Anda dapat memuat data langsung ke DDS tanpa memasukkannya ke disk terlebih dahulu. Kekurangannya adalah paket ETL menjadi lebih rumit.



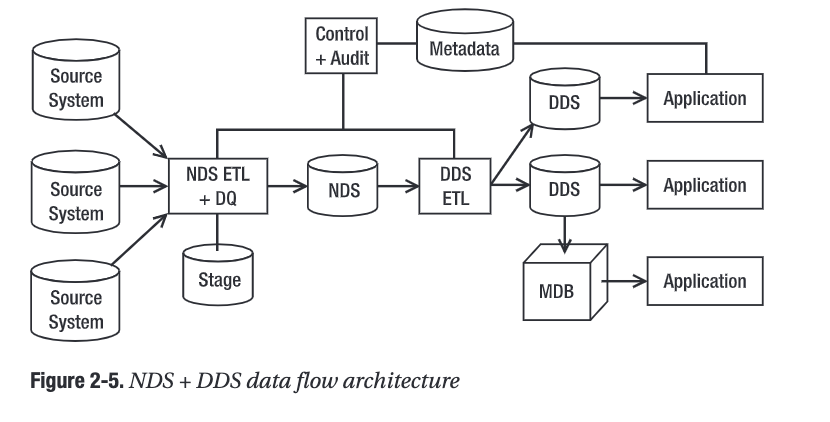
Keuntungan dari arsitektur DDS tunggal adalah lebih sederhana daripada tiga arsitektur berikutnya. Ini lebih sederhana karena data dari tahap dimuat langsung ke penyimpanan data dimensional, tanpa pergi ke penyimpanan normal apa pun terlebih dahulu. Kerugian utama adalah lebih sulit, dalam arsitektur ini, untuk membuat DDS kedua. DDS dalam arsitektur DDS tunggal adalah penyimpanan data master. Ini berisi satu set lengkap data dalam gudang data

termasuk semua versi dan semua data historis. Terkadang Anda perlu membuat DDS yang lebih kecil yang berisi subset data di DDS master untuk tujuan analisis spesifik di mana Anda ingin dapat mengubah data atau Anda ingin data menjadi statis. Untuk membuat DDS yang lebih kecil ini, Anda perlu membuat paket ETL baru yang mengambil data dari DDS master dan mengisi DDS yang lebih kecil. Anda perlu membangun paket ETL ini dari awal. Anda tidak dapat menggunakan kembali paket ETL yang ada karena paket ETL yang ada mengambil data dari tahap dan mengisi master DDS. Ini adalah aliran data yang sama sekali berbeda. Misalnya, pengguna mungkin memerlukan DDS statis yang lebih kecil yang hanya berisi data pesanan untuk tujuan menganalisis dampak kenaikan harga di berbagai akun pelanggan. Mereka menginginkan alat BI (Objek Bisnis, Cognos, atau Layanan Analisis, misalnya) untuk menjalankan DDS yang lebih kecil ini sehingga mereka dapat menganalisis kenaikan harga. Untuk membuat DDS yang lebih kecil ini, Anda perlu menulis paket ETL baru.

Anda akan menggunakan arsitektur DDS tunggal ketika Anda hanya membutuhkan penyimpanan satu dimensi dan Anda tidak memerlukan penyimpanan data yang dinormalisasi. Ini digunakan untuk solusi BI analitik yang sederhana, cepat, dan langsung di mana data hanya digunakan untuk memberi makan gudang data dimensional. Solusi DDS tunggal terutama berlaku bila Anda hanya memiliki satu sistem sumber karena Anda tidak memerlukan NDS atau ODS tambahan untuk mengintegrasikan data dari sistem sumber yang berbeda. Dibandingkan dengan arsitektur NDS + DDS atau ODS + DDS, arsitektur DDS tunggal adalah yang paling sederhana untuk build dan memiliki ETL run time tercepat karena data dimuat langsung ke DDS tanpa masuk ke penyimpanan data NDS atau ODS terlebih dahulu.

**NDS + DDS**

Dalam arsitektur aliran data NDS + DDS, ada tiga penyimpanan data: stage, NDS, dan DDS. Arsitektur ini mirip dengan arsitektur DDS tunggal, tetapi memiliki penyimpanan data yang dinormalisasi di depan DDS. NDS berada dalam bentuk relasional normal ketiga atau lebih tinggi. Tujuan memilikiNDS ada dua. Pertama, mengintegrasikan data dari beberapa sistem sumber. Kedua, mampu memuat data ke beberapa DDS. Berbeda dengan arsitektur DDS tunggal, dalam arsitektur NDS + DDS Anda dapat memiliki beberapa DDS. Gambar 2-5 menunjukkan arsitektur aliran data NDS + DDS.

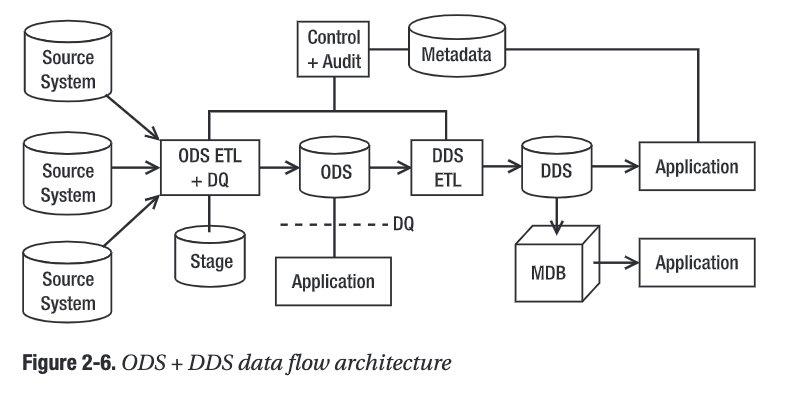
  
Dalam arsitektur NDS + DDS, NDS adalah master data store, artinya NDS berisi kumpulan data lengkap, termasuk semua data transaksi historis dan semua versi historis data master. Data transaksi historis adalah transaksi bisnis yang terjadi di masa lalu. Data dari setiap tahun disimpan di NDS. DDS, di sisi lain, bukanlah penyimpanan data utama. Ini mungkin tidak berisi semua data transaksi untuk setiap tahun. NDS berisi semua versi historis dari data master. Jika ada perubahan pada data master, atribut tidak akan ditimpa oleh nilai baru. Nilai baru dimasukkan sebagai catatan baru, dan versi lama (baris lama) disimpan di tabel yang sama. Mirip dengan sistem sumber OLTP, ada dua jenis tabel di NDS: tabel transaksi dan tabel master. Transactiontable adalah tabel yang berisi transaksi bisnis atau peristiwa bisnis. Mastertable adalah tabel yang berisi orang atau objek yang terlibat dalam acara bisnis. Tabel pesanan penjualan adalah contoh tabel transaksi. Tabel produk adalah contoh tabel master. Tabel transaksi NDS adalah sumber data untuk tabel fakta DDS. Dengan kata lain, tabel fakta di DDS diisi dari tabel transaksi di NDS. Tabel master NDS adalah sumber data untuk tabel dimensi DDS. Artinya, tabel dimensi di DDS diisi dari tabel master di NDS. Saya akan membahas lebih lanjut tentang NDS di Bab 5 ketika saya berbicara tentang pemodelan dimensi. NDS adalah penyimpanan data internal, artinya tidak dapat diakses oleh pengguna akhir atau aplikasi pengguna akhir. Data dari NDS dimuat ke dalam DDS dalam format dimensional, dan pengguna akhir mengakses DDS ini. Satu-satunya aplikasi yang dapat memperbarui NDS adalah NDS ETL. Tidak ada aplikasi lain yang dapat memperbarui NDS. Prinsip-prinsip tentang tahapan, kontrol, audit, dan metadata yang dibahas sebelumnya juga dapat diterapkan di sini. Beberapa entitas dapat terbang langsung ke NDS tanpa dipentaskan ke diskfirst. Dalam hal ini, integrasi/transformasi data dilakukan secara online dalam memori server ETL. ETL membaca data dari sistem sumber, mengubah atau mengintegrasikan data dalam memori, dan menulis data ke NDS secara langsung tanpa menulis apa pun ke stage. Transformasi data adalah mengubah, menghitung, atau memodifikasi data agar sesuai dengan basis data target. Integrasi data adalah menggabungkan record yang sama dari beberapa sistem sumber yang berbeda menjadi satu record atau menggabungkan atribut yang berbeda dari data master atau data transaksi. Dalam arsitektur NDS + DDS, DDS ETL yang memuat data ke dalam DDS lebih sederhana dibandingkan dengan arsitektur DDS tunggal. karena data di NDS sudah terintegrasi dan dibersihkan. Dalam beberapa kasus, DDS ETL perlu memasukkan data hanya secara bertahap ke DDS tanpa transformasi apa pun. Sebagian besar, jika tidak semua, perhitungan tabel fakta telah dilakukan di NDS. Data dalam NDS diunggah ke DDS. Fleksibilitas penggunaan NDS terpusat adalah Anda dapat membuat DDS yang Anda perlukan kapan saja dengan cakupan data sesuai kebutuhan. Kemampuan untuk membangun DDS baru setiap saat berguna untuk memenuhi persyaratan dari proyek yang melibatkan analisis data. Kemampuan untuk mengatur cakupan data saat membuat DDS baru berarti Anda dapat memilih tabel, kolom, dan baris mana yang ingin Anda transfer ke DDS baru. Misalnya, Anda dapat membuat DDS yang hanya berisi data mart profitabilitas pelanggan (satu tabel fakta dan semua dimensinya) yang hanya berisi data tiga bulan terakhir. Untuk mengisi DDS baru, Anda dapat menggunakan DDS ETL yang ada. Anda hanya perlu mengarahkan ETL ke DDS baru. Jika Anda membangun DDS ETL dengan benar, Anda dapat membangun kembali DDS apa pun dengan cepat, dengan kata lain, satu-satunya waktu yang kami butuhkan adalah waktu untuk menjalankan ETL. Anda tidak perlu menghabiskan waktu berhari-hari atau berminggu-minggu untuk membuat paket ETL baru untuk memuat DDS baru. Untuk mendapatkan fleksibilitas ini, DDS ETL perlu diparameterisasi; yaitu, rentang tanggal, tabel fakta, dan dimensi yang akan disalin semuanya ditetapkan sebagai parameter yang dapat dengan mudah diubah untuk menunjuk ke DDS baru. Detail koneksi database juga ditulis sebagai parameter. Ini memungkinkan Anda untuk mengarahkan ETL ke database lain.

Dalam arsitektur NDS + ODS, Anda dapat memiliki beberapa DDS. Tetapi ada satu DDS yang harus Anda bangun dan pertahankan: DDS yang berisi semua tabel fakta dan semua dimensi. Yang ini semacam wajib; semua DDS lainnya bersifat opsional—Anda membuatnya sesuai kebutuhan. Anda harus memiliki DDS wajib yang satu ini karena berisi satu set lengkap data gudang data dan digunakan oleh semua aplikasi BI yang memerlukan penyimpanan data dimensional. Tabel NDS menggunakan kunci pengganti dan kunci alami. Kunci pengganti adalah pengidentifikasi baris data master dalam gudang data. Dalam DDS, kunci pengganti digunakan sebagai kunci utama tabel dimensi. Kunci pengganti adalah bilangan bulat berurutan, dimulai dari 0. Jadi, ini adalah 0, 1, 2, 3, ..., dan seterusnya. Menggunakan kunci pengganti, Anda dapat mengidentifikasi catatan unik pada tabel dimensi. Kunci pengganti juga ada di tabel fakta untuk mengidentifikasi atribut dimensi untuk transaksi bisnis tertentu. Kunci pengganti digunakan untuk menghubungkan tabel fakta dan tabel dimensi. Misalnya, dengan menggunakan kunci pengganti, Anda dapat mengetahui detail pelanggan untuk pesanan tertentu. Dalam arsitektur NDS + DDS, kunci pengganti dipertahankan di NDS, bukan di DDS. Kunci alami adalah pengidentifikasi baris data master di sistem sumber. Saat memuat data dari tahap ke NDS, Anda perlu menerjemahkan kunci alami dari sistem sumber ke kunci pengganti gudang data. Anda dapat melakukan ini dengan mencari kunci pengganti di NDS untuk setiap nilai kunci alami dari sistem sumber. Jika kunci alami ada di NDS, itu berarti catatan sudah ada di NDS dan perlu diperbarui. Jika kunci alami tidak ada di NDS, itu berarti catatan tidak ada di NDS dan perlu dibuat.

Hanya aplikasi administratif internal yang mengakses NDS secara langsung. Ini biasanya aplikasi yang memverifikasi data yang dimuat ke NDS, seperti rutinitas kualitas data yang memeriksa NDSdata terhadap aturan firewall tertentu. Aplikasi pengguna akhir, seperti laporan BI, mengakses DDS (model dimensi) dan beberapa aplikasi, seperti aplikasi OLAP, mengakses database multidimensi yang dibangun dari DDS. Anda perlu memahami jenis penyimpanan data apa yang diperlukan oleh setiap aplikasi untuk dapat mendefinisikan arsitektur aliran data dengan benar. Keuntungan utama dari arsitektur ini adalah Anda dapat dengan mudah membangun kembali DDS utama; selain itu, Anda dapat dengan mudah membuat DDS baru yang lebih kecil. Ini karena NDS adalah master datastore, yang berisi kumpulan data lengkap, dan karena DDS ETL diparameterisasi. Ini memungkinkan Anda membuat penyimpanan data statis terpisah untuk tujuan analisis spesifik. Keuntungan kedua adalah lebih mudah untuk memelihara data master di penyimpanan yang dinormalisasi seperti NDS dan mempublikasikannya dari sana karena berisi sedikit atau tidak ada redundansi data sehingga Anda hanya perlu memperbarui satu tempat di dalam penyimpanan data. Kerugian utama adalah membutuhkan effort yang lebih dibandingkan dengan arsitektur single DDS karena data dari stage harus dimasukkan ke dalam NDS terlebih dahulu sebelum diupload ke DDS. Upaya untuk membangun ETL menjadi praktis dua kali lipat karena Anda perlu membangun dua set ETL, sedangkan di DDS tunggal hanya satu. Upaya untuk pemodelan data akan menjadi sekitar 50 persen lebih karena Anda perlu merancang tiga penyimpanan data, sedangkan dalam DDS tunggal Anda memiliki dua penyimpanan data. Arsitektur NDS + DDS menawarkan fleksibilitas yang baik untuk membuat dan memelihara penyimpanan data, terutama saat membuat DDS. NDS adalah kandidat yang baik untuk gudang data perusahaan. Ini berisi satu set data lengkap, termasuk semua versi data master, dan dinormalisasi, dengan hampir tidak ada redundansi data, sehingga pembaruan data lebih mudah dan cepat dibandingkan dengan penyimpanan data master dimensional. Ini juga berisi kunci alami kedua sistem sumber dan kunci pengganti gudang data, memungkinkan Anda untuk memetakan dan melacak data antara sistem sumber dan gudang data. Anda akan menggunakan arsitektur NDS + DDS ketika Anda perlu membuat beberapa DDS untuk tujuan berbeda yang berisi kumpulan data yang berbeda dan ketika Anda perlu mengintegrasikan data dalam bentuk yang dinormalisasi dan menggunakan data terintegrasi di luar gudang data dimensional.

**ODS + DDS**

Arsitektur ini mirip dengan arsitektur NDS + DDS, tetapi memiliki ODS sebagai pengganti NDS. Seperti NDS, ODS berada dalam bentuk normal ketiga atau lebih tinggi. Tidak seperti NDS, ODS hanya berisi versi data master saat ini; tidak memiliki data master historis. Struktur entitasnya seperti database OLTP. ODS tidak memiliki kunci pengganti. Kunci pengganti disimpan di DDS ETL. ODS mengintegrasikan data dari berbagai sistem sumber. Data dalam ODS dibersihkan dan terintegrasi. Data yang mengalir ke ODS telah melewati penyaringan DQ. Gambar 2-6 menunjukkan arsitektur aliran data ODS + DDS.



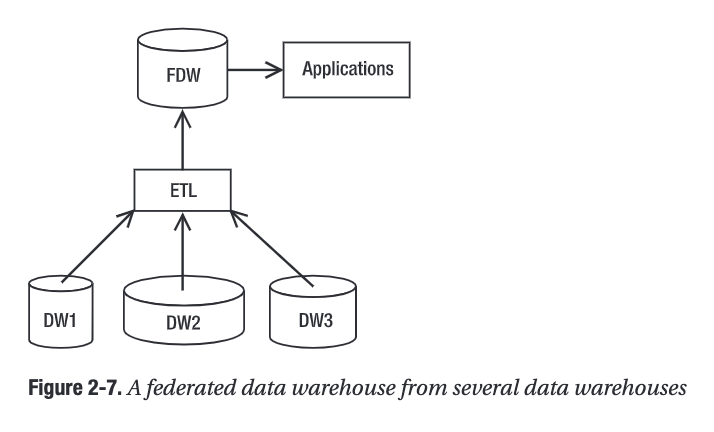
Seperti NDS, ODS berisi tabel transaksi dan tabel master. Tabel transaksi berisi peristiwa bisnis, dan tabel master berisi orang atau objek yang terlibat dalam peristiwa bisnis. Tabel fakta di DDS diisi dari tabel transaksi di ODS. Tabel dimensi di DDS diisi dari tabel master di ODS. Tidak seperti NDS, tabel master ODS hanya berisi versi data master saat ini. ODS tidak berisi versi historis dari data master. Tidak seperti NDS, yang merupakan penyimpanan data internal, ODS adalah penyimpanan data hybrid. Ini berarti ODSi dapat diakses oleh pengguna akhir dan aplikasi pengguna akhir. Dalam aplikasi NDS + DDS, NDS tidak dapat diakses oleh pengguna akhir dan aplikasi pengguna akhir. Tidak seperti NDS, ODS dapat diperbarui. Aplikasi pengguna akhir dapat mengambil data dari ODS, tetapi mereka juga dapat memperbarui ODS. Untuk memastikan kualitas data dalam ODS, aturan kualitas data juga diterapkan pada pembaruan ini. Aplikasi pengguna akhir tidak boleh memperbarui data yang berasal dari sistem sumber; itu hanya dapat memperbarui data yang dihasilkannya sendiri untuk melengkapi data sistem sumber. Jika OD digunakan untuk mendukung aplikasi dukungan pelanggan CRM, data seperti status dan komentar dapat ditulis di ODS secara langsung, tetapi semua data pelanggan masih dari sistem sumber. Dalam arsitektur ODS + DDS, DDS adalah penyimpanan data master . Tidak seperti arsitektur NDS + DDS, dalam arsitektur ODS + DDS Anda hanya memiliki satu DDS. DDS berisi satu set lengkap tabel fakta dan tabel dimensi. DDS berisi versi saat ini dan semua versi historis data master. Prinsip-prinsip tentang tahap, kontrol, audit, dan metadata yang dibahas sehubungan dengan arsitektur DDS tunggal adalah juga berlaku di sini. Beberapa entitas dapat terbang langsung ke ODS tanpa dipentaskan terlebih dahulu. Integrasi dan transformasi terjadi dalam memori server ETL. ETL DDS lebih sederhana daripada yang ada di arsitektur DDS tunggal karena data di ODS sudah terintegrasi dan dibersihkan. Dalam banyak kasus, ini benar-benar memberi makan DDS bertahap tanpa transformasi. Sebagian besar, jika tidak semua, perhitungan tabel fakta telah dilakukan di ODS. Dalam arsitektur ODS + DDS, aplikasi dapat mengakses gudang data di tiga tempat dalam tiga format berbeda: aplikasi yang membutuhkan data dalam bentuk normal dapat mengakses ODS, aplikasi yang membutuhkan data dalam format dimensi relasional dapat mengakses DDS, dan mereka yang membutuhkan data dalam format multidimensi dapat mengakses MDB. Arsitektur ini memiliki keunggulan sebagai berikut:

* Bentuk normal ketiga lebih ramping daripada NDS karena hanya berisi nilai saat ini. Hal ini membuat kinerja ODS ETL dan DDS ETL lebih baik daripada yang ada di arsitektur NDS + DDS.
* Seperti arsitektur NDS + DDS, dalam arsitektur ODS + DDS Anda memiliki tempat sentral untuk mengintegrasikan, memelihara, dan mempublikasikan data master.
* Penyimpanan relasional yang dinormalisasi dapat diperbarui oleh aplikasi pengguna akhir, sehingga mampu mendukung aplikasi operasional di tingkat transaksi.

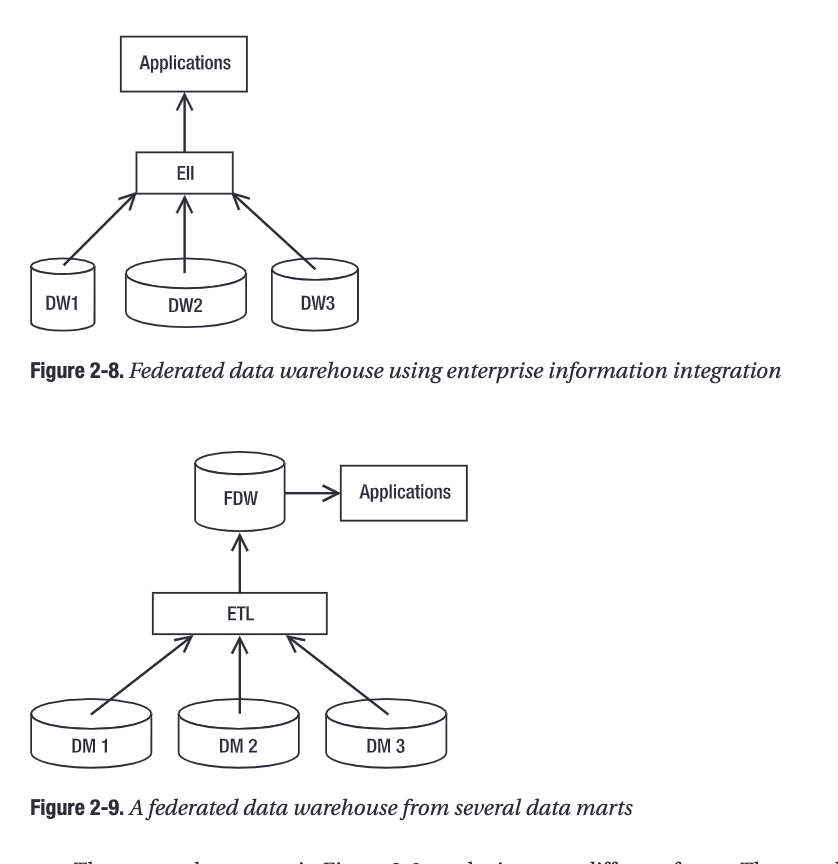
Kerugian dari arsitektur ini adalah untuk membangun DDS baru yang kecil (misalnya, data penjualan Q4 2007), Anda harus mendapatkannya dari DDS utama dan tidak dapat menggunakan DDS ETL yang ada untuk melakukannya. Anda perlu menulis kueri khusus ( dengan kata lain, buat tabel dari pilih), yang tidak disukai karena alasan standarisasi dan konsistensi, atau untuk membangun ETL baru, yang juga tidak disukai karena upaya, terutama jika itu adalah barang sekali pakai. gunakan arsitektur ODS + DDS ketika Anda hanya membutuhkan penyimpanan data satu dimensi dan Anda memerlukan penyimpanan data tersentralisasi dan ternormalisasi untuk digunakan untuk tujuan operasional seperti CRM. ODS berisi data terintegrasi yang terperinci, bernilai terkini, yang berguna untuk kueri transaksional.

**Federated Data Warehouse**

Sebuah gudang data federasi terdiri dari beberapa gudang data dengan lapisan pengambilan data di atasnya. Sebuah gudang data federasi mengambil data dari gudang data yang ada menggunakan ETL dan memuat data ke dalam penyimpanan data dimensi baru. Misalnya, karena aktivitas merger dan akuisisi, Anda dapat memiliki tiga gudang data. Mungkin yang pertama adalah gudang data adimensional, yang kedua adalah gudang data normal ketiga yang dinormalisasi, dan yang ketiga adalah gudang data relasional dengan beberapa tabel transaksi besar yang merujuk pada banyak tabel referensi. Gambar 2-7 menunjukkan gudang data federasi. Perincian data PLRT Asing sama dengan perincian tertinggi gudang data sumber. Jika granularity dari source data warehouse adalah G1, G2, dan G3, maka granularity dari data FDW adalah yang tertinggi dari (G1, G2, G3). Ini karena Anda dapat menggabungkan data hanya pada perincian yang sama dan karena Anda hanya dapat mengubah data dari perincian yang lebih rendah ke perincian yang lebih tinggi, bukan arah sebaliknya. Misalnya, jika G1 harian, G2 mingguan, dan G3 bulanan, maka perincian FDWis bulanan. Ini karena Anda dapat mengubah G1 dan G2 menjadi bulanan, tetapi Anda tidak dapat mengubah G3 menjadi harian atau mingguan. ETL yang mengekstrak data dari gudang data sumber perlu mengetahui waktu data. Data di gudang data sumber mungkin tidak sampai pada frekuensi yang sama. ETL FDW harus sesuai dengan frekuensi DW sumber. Jika DW sumber memiliki frekuensi pembaruan yang berbeda, ETL FDW perlu dijalankan beberapa kali agar sesuai dengan frekuensi setiap DW sumber.



FDW ETL perlu mengintegrasikan data dari DW sumber berdasarkan aturan bisnis. ETL perlu mengidentifikasi apakah catatan dari satu gudang data sumber merupakan duplikat catatan dari gudang data sumber lainnya. Catatan duplikat perlu digabungkan. Aturan bisnis perlu diterapkan untuk menentukan catatan mana yang harus bertahan dan atribut mana yang perlu diperbarui. Selain itu, ETL PLRT Asing perlu mengubah data dari DW sumber yang berbeda ke dalam struktur dan format umum yang sama. Anda juga perlu mengingat bahwa jumlah bidang subjek di PLRT Asing mungkin lebih sedikit daripada jumlah bidang subjek di DW sumber. . Hal ini karena PLRT Asing berada pada tingkat yang lebih tinggi daripada PRT sumber, dan pada tingkat ini, tidak semua bidang studi dapat diterapkan dan bermanfaat bagi pengguna bisnis. Misalnya, di gudang data global, Anda mungkin tertarik pada penjualan dan karyawan dan meninggalkan inventaris dan kampanye di gudang data regional. Ketika Anda memiliki beberapa gudang data seperti ini, dimungkinkan untuk menerapkan integrasi informasi perusahaan (EII) daripada ETL, seperti yang dijelaskan pada Gambar 2-8, terutama jika gudang data sumber serupa. EII adalah metode untuk mengintegrasikan data dengan mengakses sistem sumber yang berbeda secara online dan menggabungkan output dengan cepat sebelum memberikan hasil akhir kepada pengguna. Semuanya dilakukan saat pengguna meminta informasi. Tidak ada penyimpanan dalam bentuk apa pun yang menyimpan data teragregasi atau terintegrasi. Keuntungan utama menggunakan EII adalah kesegaran data; dengan kata lain, hasilnya benar-benar real time. Pada Gambar 2-8, bahkan jika gudang data DW3 diubah sedetik yang lalu, hasil akhirnya akan tetap menyertakan perubahan detik terakhir itu. Meskipun tidak ada gudang data terintegrasi, terkadang arsitektur ini juga disebut gudang data federasi. Misalnya, jika organisasi terdiri dari tiga wilayah geografis, seperti Asia Pasifik, Amerika, dan EMEA, maka setiap wilayah memiliki sistem ERP yang berbeda dan organisasi TI yang berbeda; karenanya, mereka memiliki tiga proyek pergudangan data. Tetapi mereka setuju untuk menstandarisasi pada struktur yang sama: pergudangan data dimensional dengan tabel fakta dan tabel dimensi. Alih-alih mengintegrasikan beberapa gudang data, gudang data federasi juga dapat diimplementasikan ketika ada beberapa data mart yang tidak terintegrasi dalam organisasi, seperti yang ditunjukkan pada Gambar 2-9.

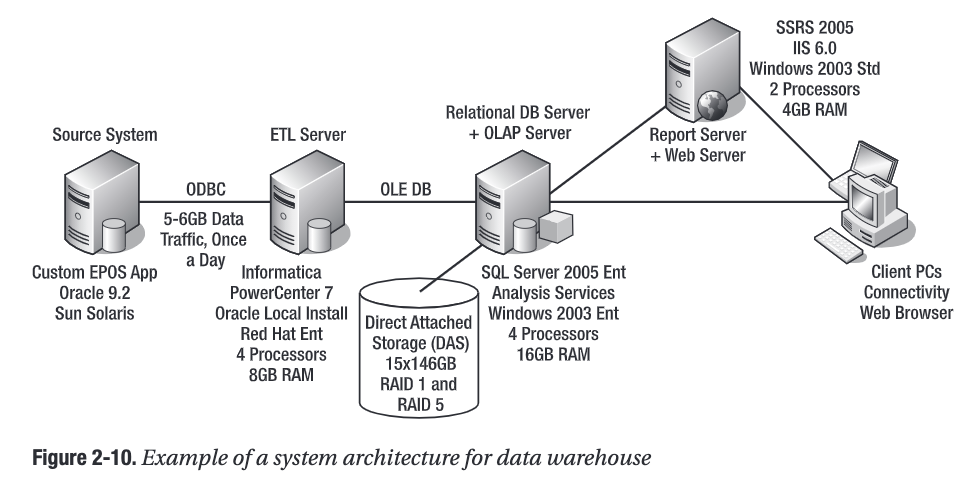


Data mart sumber pada Gambar 2-9 dapat dalam berbagai bentuk. Mereka bisa berdimensi, dinormalisasi, atau tidak keduanya. Mart ini dapat berupa dua tabel fakta yang sesuai yang dikelilingi oleh empat hingga delapan dimensi yang sesuai dalam format skema bintang, yang dirancang dengan baik menggunakan pendekatan pemodelan dimensi Kim-ball. Mereka bisa dalam format kepingan salju di mana dimensi dinormalisasi. Mereka bisa berada dalam bentuk normal ketiga (atau lebih tinggi). Dan dalam banyak kasus, mereka mungkin bukan data mart sama sekali; dengan kata lain, mereka tidak mengikuti pendekatan desain pergudangan data formal dan sebaliknya hanya kumpulan tabel dengan data yang baik di dalamnya, dan pengguna menyebut tabel ini data mart. Anda dapat menggunakan FDW untuk mengintegrasikan data mart ini. Prinsip yang sama dijelaskan sebelumnya ketika mengintegrasikan gudang data masih dapat diterapkan, khususnya, tingkat perincian, pencocokan/penggandaan, pemformatan ulang, penanganan masalah waktu, dan keragaman platform. Dalam beberapa kasus, Anda mungkin ingin mempertimbangkan untuk melewati beberapa mart dan mengakses sistem sumber secara langsung, terutama jika sistem sumber yang sama memasok data yang sama ke data mart yang berbeda. Anda juga masih dapat menggunakan pendekatan EII di sini saat mengintegrasikan data mart. Artinya, Anda tidak memiliki penyimpanan data permanen di tingkat federasi tetapi dapat mengakses sumber data dengan cepat saat pengguna mengeluarkan permintaan mereka.

Keuntungan utama dari arsitektur ini adalah Anda dapat mengakomodasi gudang data yang ada, dan oleh karena itu waktu pengembangan akan lebih singkat. Kerugian utama adalah bahwa, secara praktis, sulit untuk membangun gudang berkualitas baik dari beragam standar yang ditemukan di data mart sumber atau gudang data. Anda akan menggunakan arsitektur gudang data gabungan ketika Anda ingin memanfaatkan gudang data yang ada atau di mana Anda ingin mengintegrasikan data dari beberapa data mart.

**System Architecture**

Bagian sebelumnya membahas arsitektur aliran data. Mereka menunjukkan bagaimana data diatur dalam penyimpanan data dan bagaimana data mengalir dalam sistem gudang data. Setelah Anda memilih arsitektur aliran data tertentu, Anda kemudian perlu merancang arsitektur sistem, yang merupakan pengaturan fisik dan koneksi antara server, jaringan, perangkat lunak, sistem penyimpanan, dan klien. Merancang arsitektur sistem membutuhkan pengetahuan tentang perangkat keras (terutama server), jaringan (terutama yang berkaitan dengan keamanan dan kinerja, dan dalam beberapa tahun terakhir juga jaringan serat), dan penyimpanan (terutama jaringan area penyimpanan [SAN], array disk murah yang berlebihan [RAID], dan solusi backup tape otomatis). Gambar 2-10 menunjukkan contoh arsitektur sistem.



Dalam contoh ini, arsitektur sistem terdiri dari tiga server: satu server ETL, satu server database, dan satu server pelaporan. Sistem sumber adalah sistem point-of-sale elektronik yang mencatat penjualan eceran di toko-toko yang menggunakan Oracle 9.2 di Sun Solaris. ETLserver adalah Informatica PowerCenter 7 yang diinstal pada Red Hat Enterprise Linux. Penyimpanan data gudang data (tahap, DDS, dan MDB) di-host di mesin basis data relasional SQL Server dan Layanan Analisis, berjalan pada Windows Server 2003. Data disimpan secara fisik dalam DAS yang terdiri dari lima belas disk 146GB, membuat kapasitas mentah 2TB. Disk dikonfigurasi dalam RAID 1 dan RAID 5 untuk ketahanan data. (Saya akan membahas RAID di Bab 6 ketika saya membahas desain basis data fisik.) Laporan dibuat dan dikelola menggunakan SSRS yang diinstal pada Windows Server 2003.

Saya memilih contoh ini karena mewakili arsitektur khas untuk sistem menengah. Kami memiliki server ETL khusus, terpisah dari server database. Ini adalah data berukuran sedang; kapasitas mentah 2TB adalah sekitar 400GB hingga 500GB ruang basis data akhir yang dapat digunakan, dengan asumsi kami memiliki lingkungan pengembangan dan produksi. Platformnya sedikit campuran, seperti yang biasanya ditemukan di organisasi: sistem sumber dan ETL bukan Microsoft. Informatica mungkin sudah ada saat proyek gudang data dimulai, jadi mereka harus menggunakan apa yang sudah mereka miliki. Oleh karena itu, Anda dapat membuat arsitektur sistem dengan platform yang berbeda. Seorang arsitek data warehouse biasanya tidak merancang arsitektur sistem infrastruktur datawarehouse, tetapi menurut saya, akan sangat membantu jika mereka mengetahui subjek ini—mungkin tidak pada tingkat yang sangat rinci. , tetapi mereka perlu tahu pada tingkat tinggi. Misalnya, mereka tidak perlu memahami cara membangun klaster empat simpul menggunakan server Windows 2003, tetapi mereka perlu mengetahui ketersediaan tinggi seperti apa yang dapat dicapai dengan menggunakan teknologi pengelompokan. Untuk merancang arsitektur sistem gudang data, Anda terlebih dahulu membuat technologystack yang ingin Anda gunakan untuk ETL, database, dan BI, seperti Microsoft SQL Server (SSIS,SSAS, SSIS), Informatica + Oracle 9i + Cognos, dan sebagainya. Hal ini ditentukan berdasarkan kemampuan produk dan standar perusahaan. Setelah Anda menentukan technol-ogy stack, Anda melakukan desain tingkat tinggi pada server, konfigurasi jaringan, dan konfigurasi penyimpanan yang mendukung teknologi yang dipilih, termasuk desain ketersediaan tinggi. Kemudian Anda menentukan spesifikasi teknis rinci dari server, jaringan, dan usia penyimpanan. Hal ini dilakukan berdasarkan kapasitas dan persyaratan kinerja sistem. Anda kemudian memesan perangkat keras dan perangkat lunak dan membangun sistem di pusat data bersama dengan vendor perangkat keras dan jaringan. Anda kemudian menginstal dan mengkonfigurasi perangkat lunak. Merancang dan membangun lingkungan adalah hal mendasar dan penting untuk kinerja dan stabilitas sistem gudang data yang akan Anda bangun di atasnya.

Faktor lain yang sangat mempengaruhi arsitektur sistem adalah pilihan perangkat lunak dalam membangun gudang data, seperti versi tertentu dari SQL Server, Oracle, atau Teradata. Arsitektur sistem yang diperlukan untuk menjalankan perangkat lunak ini berbeda. Misalnya, Teradata berjalan pada perangkat keras pemrosesan paralel yang sangat besar. Itu tidak berbagi penyimpanan antara node. Di sisi lain, cluster SQL Server menggunakan penyimpanan pusat. Ini berbagi penyimpanan antara node. Dalam hal perangkat lunak, ada dua jenis perangkat lunak database: symmetric multi-processing (SMP) dan pemrosesan paralel besar-besaran (MPP). Sistem basis data SMP adalah sistem basis data yang berjalan pada satu atau lebih mesin dengan beberapa prosesor identik yang berbagi penyimpanan disk yang sama. Ketika sistem database SMP berjalan di lebih dari satu mesin, itu disebut konfigurasi cluster. Database secara fisik terletak di sistem penyimpanan disk tunggal. Contoh sistem database SMP adalah SQL Server, Oracle, DB/2, Informix, dan Sybase. Sistem database AnMPP adalah sistem database yang berjalan pada lebih dari satu mesin dimana setiap mesin memiliki penyimpanan disk sendiri. Database secara fisik terletak di beberapa sistem penyimpanan disk yang saling berhubungan satu sama lain. Sistem basis data MPP juga dikenal sebagai sistem basis data paralel. Contoh sistem database MPP adalah Teradata, Neoview, Netezza, dan DATAllegro. Mesin-mesin dalam sistem database SMP dan MPP disebut node. Sistem database MPP lebih cepat dan lebih terukur daripada sistem database SMP. Dalam sistem database MPP, atable secara fisik terletak di beberapa node, masing-masing dengan penyimpanannya sendiri. Saat Anda mengambil data dari tabel ini, semua node secara bersamaan membaca data dari penyimpanannya sendiri, sehingga proses membaca data dari disk lebih cepat. Demikian pula, ketika Anda memuat data ke dalam tabel ini, semua node secara bersamaan memuat sedikit data ke dalam disk mereka. Dalam sistem database SMP, ada kendala pada penyimpanan disk. Sistem database SMP, di sisi lain, lebih sederhana, lebih mudah dirawat, dan memiliki biaya lebih rendah. Dalam buku ini, saya akan menunjukkan cara menggunakan SQL Server 2005 untuk membangun gudang data, tetapi juga berjalan di SQL Server 2008. Saya akan menggunakan SSIS untuk alat ETL, saya akan membangun NDS dan DDS pada database SQLServer 2005, dan saya akan menggunakan SSRS untuk pelaporan dan SSAS untuk OLAP dan penambangan data.

**Kesimpulan**

Dalam bab ini, saya membahas beberapa arsitektur gudang data. Arsitektur DDS, NDS + DDS, dan ODS + DDS tunggal memiliki gudang data dimensional. DDS tunggal tidak memiliki toko yang dinormalisasi, sedangkan yang lain memiliki toko yang dinormalisasi. Arsitektur gudang data federasi terdiri dari beberapa gudang data yang terintegrasi oleh lapisan pengambilan data. Saya juga membahas arsitektur sistem khususnya untuk implementasi menggunakan SQL Server 2005. Anda akan menggunakan studi kasus yang diperkenalkan di bab ini untuk membantu Anda memahami subjek di bab selanjutnya. Dalam Bab 5, ketika saya membahas pemodelan dimensi dan pemodelan hubungan entitas, saya akan membahas desain gudang data secara lebih rinci. Dalam Bab 6, ketika membahas desain basis data fisik, saya akan membahas elemen arsitektur sistem secara lebih rinci.