**BUKU DIKTAT  
DATA WAREHOUSE**



**DOSEN PENGAMPU**

Prof. Dr. I Made Sukarsa, S.T., M.T.

**MATA KULIAH**

PANGKALAN DATA

**PROGRAM STUDI TEKNOLOGI INFORMASI**

**FAKULTAS TEKNIK**

**UNIVERSITAS UDAYANA**

**BALI**

**2025**

# KATA PENGANTAR

# DAFTAR ISI

# BAB I PENDAHULUAN

## Latar Belakang

Perkembangan teknologi informasi telah mendorong organisasi untuk memanfaatkan data dalam skala yang semakin besar. Hampir seluruh aktivitas operasional, baik di bidang bisnis, pendidikan, pemerintahan, maupun layanan publik, menghasilkan data secara terus-menerus. Data tersebut tersimpan dalam berbagai sistem operasional yang dirancang untuk mendukung transaksi harian, seperti sistem penjualan, sistem keuangan, sistem akademik, dan sistem sumber daya manusia.

Meskipun organisasi memiliki jumlah data yang sangat besar, kondisi ini tidak selalu diikuti dengan kemampuan untuk memperoleh informasi yang berguna bagi pengambilan keputusan. Data sering kali tersebar di banyak sistem, memiliki struktur yang berbeda, serta mengikuti aturan bisnis yang tidak seragam. Akibatnya, ketika dibutuhkan analisis menyeluruh, organisasi menghadapi kesulitan dalam mengintegrasikan data dan menghasilkan informasi yang konsisten. Kondisi ini dikenal sebagai information crisis, yaitu situasi di mana organisasi kaya akan data tetapi miskin informasi yang dapat dipercaya (Ponniah, 2010).

Sistem operasional atau Online Transaction Processing (OLTP) memang dirancang untuk mencatat transaksi secara cepat dan akurat, namun sistem ini tidak dioptimalkan untuk kebutuhan analisis jangka panjang. Pertanyaan-pertanyaan strategis seperti tren penjualan dari waktu ke waktu, perbandingan kinerja antar unit, atau pola perilaku pelanggan sulit dijawab jika hanya mengandalkan sistem operasional.

Untuk mengatasi permasalahan tersebut, diperlukan sebuah sistem yang mampu mengintegrasikan data dari berbagai sumber, menyimpan data historis secara terstruktur, serta menyediakan lingkungan yang mendukung analisis dan pengambilan keputusan. Data warehouse hadir sebagai solusi dengan menyediakan penyimpanan data terpusat yang dirancang khusus untuk kebutuhan analitis, bukan untuk transaksi harian (Inmon, 2005).

## Peran Data Warehouse dalam Sistem Informasi

Data warehouse memiliki peran penting dalam arsitektur sistem informasi modern. Berbeda dengan basis data operasional, data warehouse berfungsi sebagai repositori data historis yang telah melalui proses integrasi, pembersihan, dan penyeragaman. Data yang tersimpan di dalam data warehouse bersifat relatif stabil dan digunakan untuk keperluan analisis, pelaporan, serta business intelligence.

Dengan adanya data warehouse, sistem operasional dapat tetap fokus menjalankan transaksi harian tanpa terbebani oleh proses analisis yang kompleks. Sementara itu, data warehouse menyediakan lingkungan terpisah yang memungkinkan pengguna melakukan analisis multidimensi, melihat pola historis, serta mendukung pengambilan keputusan strategis secara lebih akurat (Inmon, 2005; Kimball & Ross, 2013).

Bagi mahasiswa Teknologi Informasi, pemahaman mengenai data warehouse menjadi penting karena konsep ini merupakan fondasi dari berbagai teknologi analitik modern, seperti data analytics, business intelligence, dan decision support system. Oleh karena itu, data warehouse tidak hanya dipahami sebagai teknologi penyimpanan data, tetapi sebagai bagian dari arsitektur sistem informasi yang mendukung pengelolaan data secara menyeluruh.

## Tujuan Penyusunan Buku Diktat

Buku diktat ini disusun sebagai bahan ajar untuk membantu mahasiswa memahami konsep dan implementasi data warehouse secara bertahap dan terstruktur. Penyajian materi dimulai dari pemahaman konseptual mengenai data warehouse, dilanjutkan dengan pembahasan perbedaan data warehouse dan sistem operasional, hingga tahapan perancangan dan implementasi data warehouse dalam konteks praktikum. Secara khusus, buku ini bertujuan untuk:

1. Memberikan pemahaman dasar mengenai konsep data warehouse dan perannya dalam sistem informasi.
2. Menjelaskan perbedaan antara data warehouse dan sistem OLTP agar mahasiswa memahami alasan perlunya data warehouse.
3. Menguraikan tahapan pembangunan data warehouse, mulai dari proses integrasi data hingga penyajian informasi.
4. Menyajikan contoh implementasi data warehouse menggunakan alat bantu seperti Pentaho dan bahasa pemrograman PHP.
5. Memperkenalkan konsep kualitas data (*data quality*) sebagai aspek penting dalam keberhasilan data warehouse.

Dengan pendekatan ini, mahasiswa diharapkan tidak hanya memahami teori data warehouse, tetapi juga mampu mengaitkan konsep tersebut dengan implementasi nyata dalam studi kasus.

## Sistematika Penulisan

Buku diktat ini disusun secara sistematis agar pembaca dapat memahami materi secara bertahap. Bab II membahas konsep dasar data warehouse, termasuk pengertian, tujuan, karakteristik, serta pendekatan pembangunan data warehouse. Bab III mengulas perbedaan antara data warehouse dan sistem OLTP. Bab IV menjelaskan tahapan dalam pembangunan data warehouse, mulai dari sumber data hingga penyajian informasi.

Bab V dan Bab VI membahas aspek perancangan data warehouse, termasuk desain skema dan denormalisasi. Bab VII dan Bab VIII menyajikan implementasi data warehouse melalui studi kasus menggunakan Pentaho dan alat bantu lainnya. Bab IX membahas konsep kualitas data (data quality) dan perannya dalam memastikan hasil analisis yang akurat dan dapat dipercaya.

Melalui sistematika ini, buku diktat diharapkan dapat menjadi panduan pembelajaran yang komprehensif bagi mahasiswa dalam memahami dan mengimplementasikan data warehouse.

# BAB II PENGERTIAN DWH

## Pengertian Data Warehouse

Data Warehouse (DWH) adalah sistem penyimpanan data yang dirancang khusus untuk **membantu analisis data dan pengambilan keputusan**. Data yang disimpan di dalam data warehouse bukan data transaksi harian, melainkan **data historis** yang telah dikumpulkan dan disatukan dari berbagai sistem operasional yang berbeda (Inmon, 2005; Ponniah, 2010).

Pada kegiatan sehari-hari, organisasi menggunakan sistem operasional untuk mencatat aktivitas rutin, seperti transaksi penjualan, pengisian data mahasiswa, atau pencatatan keuangan. Sistem tersebut dibuat agar transaksi dapat berjalan cepat dan akurat. Namun, sistem ini tidak dirancang untuk menjawab pertanyaan analitis seperti bagaimana perkembangan dari waktu ke waktu *atau* bagaimana perbandingan antar unit*.*

Di sinilah peran data warehouse dibutuhkan, yaitu sebagai tempat penyimpanan data yang telah dirapikan dan siap digunakan untuk analisis.

## Kebutuhan & Tujuan Data Warehouse

Dalam sebuah organisasi, data umumnya tersebar di banyak sistem operasional (OLTP). Setiap sistem dibuat untuk kebutuhan tertentu dan sering kali menggunakan teknologi, struktur basis data, serta aturan bisnis yang berbeda. Akibatnya, data tidak dapat langsung digabungkan ketika dibutuhkan untuk analisis menyeluruh.

Kondisi ini menimbulkan masalah yang dikenal sebagai information crisis, yaitu keadaan ketika organisasi memiliki banyak data tetapi kesulitan memperoleh informasi yang jelas, konsisten, dan dapat dipercaya untuk pengambilan keputusan (Ponniah, 2010).

Data warehouse dibangun untuk mengatasi permasalahan keterpisahan data pada sistem operasional dengan cara mengumpulkan data dari berbagai sistem yang berbeda, kemudian melakukan proses pembersihan dan penyeragaman agar data memiliki format dan makna yang konsisten. Selanjutnya, data tersebut disimpan secara terpusat dalam satu repositori khusus sehingga dapat diakses secara terpadu. Dengan mekanisme ini, data warehouse mampu menyediakan data yang siap digunakan untuk analisis dan pengambilan keputusan tanpa mengganggu kinerja sistem transaksi harian yang digunakan dalam operasional organisasi (Inmon, 2005).

## Data Sebelum Ada Data Warehouse

|  |
| --- |
|  |

Gambar 2. 1 Ilustrasi Data Sebelum Ada Data Warehouse (Sumber: Ilustrasi konseptual yang disusun oleh penulis (*AI-Generated*) berdasarkan teori Data Warehouse (Inmon, 2005; Ponniah, 2010)

Gambar ini menggambarkan kondisi data dalam organisasi sebelum adanya data warehouse. Setiap departemen memiliki sistem operasional dan basis data masing-masing yang digunakan untuk mendukung aktivitas harian. Sistem-sistem tersebut berjalan secara terpisah dan tidak saling terintegrasi.

Dalam kondisi ini, ketika manajemen membutuhkan informasi secara menyeluruh, data harus dikumpulkan dari masing-masing sistem secara terpisah. Proses ini berisiko menimbulkan ketidakkonsistenan data serta kesalahan analisis karena perbedaan struktur, format, dan aturan bisnis antar sistem (Ponniah, 2010). Kondisi inilah yang mendorong perlunya data warehouse sebagai solusi integrasi data tingkat *enterprise* (Inmon, 2005).

## Data Warehouse Sebagai Pusat Integrasi Data

|  |
| --- |
|  |

Gambar 2. 2 Data Warehouse sebagai Pusat Integrasi Data (Sumber: Ilustrasi konseptual yang disusun oleh penulis (*AI-Generated*) berdasarkan teori Data Warehouse (Inmon, 2005)

Gambar ini menggambarkan peran data warehouse sebagai pusat integrasi data dalam sebuah organisasi. Data berasal dari berbagai sistem operasional (OLTP) seperti sistem penjualan, sistem keuangan, dan sistem sumber daya manusia yang masing-masing memiliki basis data sendiri.

Data dari sistem-sistem tersebut dikumpulkan dan disimpan secara terpusat di dalam data warehouse. Dengan adanya penyimpanan terpusat ini, data dapat dianalisis secara terpadu tanpa harus mengakses sistem operasional satu per satu.

Pendekatan ini memungkinkan sistem operasional tetap fokus menjalankan transaksi harian, sementara data warehouse berfungsi menyediakan data historis yang stabil untuk keperluan analisis, pelaporan, dan pengambilan keputusan manajerial. Inmon (2005) menegaskan bahwa data warehouse dirancang sebagai lingkungan terintegrasi (*integrated environment*) yang mendukung pengambilan keputusan, bukan untuk memproses transaksi operasional.

## Data Warehouse Bukan Database Biasa

Penting untuk dipahami bahwa data warehouse bukan sekadar basis data biasa. Basis data operasional dirancang untuk menangani transaksi harian yang bersifat cepat dan sering mengalami perubahan. Sebaliknya, data warehouse dirancang untuk menyimpan data dalam jangka panjang, menyajikan data yang relatif stabil, serta mendukung kegiatan analisis dan pelaporan yang bersifat strategis. Perbedaan tujuan tersebut menyebabkan perbedaan dalam cara penyimpanan, struktur, dan pengelolaan data antara sistem operasional dan data warehouse (Ponniah, 2010).

## Pengertian Data Warehouse Menurut Para Ahli

### Bill Inmon

Bill Inmon mendefinisikan data warehouse sebagai: “*A data warehouse is a subject-oriented, integrated, time-variant, and non-volatile collection of data in support of management’s decision-making process*” (Inmon, 2005). Definisi ini menekankan bahwa data warehouse berfokus pada subjek tertentu, mengintegrasikan data dari berbagai sumber, menyimpan data historis, dan bersifat stabil.

### Ralph Kimball

Ralph Kimball mendefinisikan data warehouse sebagai: “*A data warehouse is a copy of transaction data specifically structured for query and analysis*” (Kimball & Ross, 2013). Kimball menekankan bahwa data warehouse merupakan salinan data transaksi yang disusun ulang agar mudah dianalisis, terutama melalui pendekatan dimensional modeling.

### Paulraj Ponniah

Ponniah mendefinisikan data warehouse sebagai: “*A data warehouse is a relational database designed for query and analysis rather than for transaction processing*” (Ponniah, 2010). Definisi ini menegaskan bahwa data warehouse dibuat untuk analisis, bukan untuk transaksi operasional.

## Sintesis Pengertian Data Warehouse

Berdasarkan berbagai definisi tersebut, data warehouse dapat dipahami sebagai sistem penyimpanan data terpusat yang berisi data historis dari berbagai sumber dan disusun secara khusus untuk mendukung analisis serta pengambilan keputusan (Inmon, 2005; Kimball & Ross, 2013; Ponniah, 2010).

## Data Warehouse Sebagai Arsitektur Sistem Informasi

Data warehouse bukanlah sebuah aplikasi tunggal, melainkan bagian dari **arsitektur sistem informasi**. Inmon menjelaskan bahwa data warehouse merupakan lingkungan terintegrasi yang mencakup sumber data, proses integrasi, penyimpanan data, serta penyajian informasi (Inmon, 2005).

Ponniah juga menegaskan bahwa data warehouse adalah sebuah environment, bukan sekadar produk atau perangkat lunak tertentu (Ponniah, 2010). Oleh karena itu, data warehouse harus dipahami sebagai sistem yang dirancang secara menyeluruh untuk mendukung kebutuhan analisis organisasi.

## Ilustrasi Alur Kerja Data Warehouse

|  |
| --- |
| Apa Itu Gudang Data Tradisional? Contoh & Tantangan | Estuary |

Gambar 2. 3 Alur Data dari Sistem Operasional hingga Analisis (Sumber: Diadaptasi dari Estuary (2023). *What Is a Traditional Data Warehouse? Examples & Challenges*)

Gambar ini menggambarkan alur data dalam arsitektur data warehouse tradisional, mulai dari sistem operasional hingga digunakan untuk analisis. Data berasal dari berbagai sumber operasional (OLTP), seperti basis data transaksi dan file, yang masing-masing digunakan untuk menjalankan aktivitas harian organisasi.

Sebelum data digunakan untuk analisis, data tersebut terlebih dahulu dikumpulkan dan diproses melalui area staging atau proses integrasi. Pada tahap ini, data disiapkan agar memiliki format dan struktur yang seragam sehingga dapat disimpan secara terpusat di dalam data warehouse.

Data warehouse berfungsi sebagai pusat penyimpanan data historis yang terintegrasi. Di dalam data warehouse, data tidak lagi digunakan untuk transaksi operasional, melainkan untuk mendukung kebutuhan analisis, pelaporan, dan pengambilan keputusan. Data yang telah tersimpan kemudian dimanfaatkan melalui data mart dan alat analitik seperti *reporting*, data mining, dan *analytics*.

Arsitektur ini menunjukkan pemisahan yang jelas antara sistem operasional dan sistem analitik. Dengan pemisahan tersebut, sistem operasional tetap fokus pada transaksi harian, sementara data warehouse menyediakan lingkungan terpisah yang stabil dan terstruktur untuk analisis. Konsep ini sejalan dengan pandangan Inmon (2005) yang menekankan bahwa data warehouse merupakan *integrated environment* yang dirancang khusus untuk mendukung pengambilan keputusan manajerial.

## Pendekatan Dalam Membangun Data Warehouse

Dalam praktiknya, data warehouse dapat dibangun dengan dua pendekatan utama, yaitu pendekatan top-down dan pendekatan bottom-up. Kedua pendekatan ini menjelaskan bagaimana memulai pembangunan data warehouse, bukan tentang alat yang digunakan.

### Pendekatan Top-Down (Bill Inmon)

|  |
| --- |
|  |

Gambar 2. 4 Pendekatan Top-Down dalam Pembangunan Data Warehouse (Sumber: Diadaptasi dari Inmon, W.H. (2005). *Building the Data Warehouse*

Gambar ini menggambarkan pendekatan top-down dalam pembangunan data warehouse sebagaimana dikemukakan oleh Bill Inmon. Pada pendekatan ini, organisasi terlebih dahulu membangun satu Enterprise Data Warehouse (EDW) yang berfungsi sebagai pusat integrasi data dari seluruh sistem operasional.

Data berasal dari berbagai sistem operasional (OLTP) dan sumber data lain seperti file, yang masing-masing digunakan untuk mendukung aktivitas harian organisasi. Sebelum dimasukkan ke dalam data warehouse, data tersebut melalui proses integrasi dan pembersihan agar memiliki struktur dan kualitas yang konsisten.

Enterprise Data Warehouse menjadi sumber data terpusat yang menyimpan data historis secara terintegrasi di tingkat *enterprise*. Setelah data tersimpan dan terstandarisasi di dalam EDW, data tersebut kemudian dimanfaatkan untuk kebutuhan analisis melalui data mart atau sistem pendukung keputusan seperti pelaporan dan *business intelligence*.

Pendekatan top-down ini menekankan konsistensi dan integrasi data secara menyeluruh di seluruh organisasi. Dengan membangun data warehouse sebagai fondasi utama terlebih dahulu, setiap analisis dan laporan yang dihasilkan menggunakan sumber data yang sama, sehingga mendukung pengambilan keputusan manajerial yang lebih akurat dan terpercaya (Inmon, 2005).

### Pendekatan Bottom-Up (Ralph Kimball)

|  |
| --- |
|  |

Gambar 2. 5 Pendekatan Bottom-Up dalam Pembangunan Data Warehouse (Sumber: Diadaptasi dari Kimball, R., & Ross, M. (2013). *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*)

Gambar ini menggambarkan pendekatan bottom-up dalam pembangunan data warehouse sebagaimana dikemukakan oleh Ralph Kimball. Pada pendekatan ini, pembangunan data warehouse dimulai dari pembuatan data mart yang berfokus pada satu proses bisnis tertentu, seperti penjualan, inventori, atau pembelian.

Setiap data mart dirancang menggunakan model dimensional yang terdiri dari tabel fakta dan tabel dimensi. Data mart tersebut dibangun untuk memenuhi kebutuhan analisis pada satu unit atau fungsi bisnis tertentu sehingga manfaat data warehouse dapat dirasakan lebih cepat.

Agar data mart yang berbeda dapat saling terhubung dan digunakan secara terpadu, Kimball memperkenalkan konsep bus architecture. Bus architecture berfungsi sebagai kerangka integrasi yang menggunakan dimensi bersama (*conformed dimensions*), seperti waktu, produk, atau pelanggan, sehingga data dari berbagai data mart tetap konsisten dan dapat dianalisis secara lintas fungsi.

Melalui integrasi bertahap antar data mart menggunakan bus architecture, kumpulan data mart tersebut secara keseluruhan membentuk sebuah sistem data warehouse yang lebih besar di tingkat enterprise. Pendekatan bottom-up ini bersifat lebih fleksibel dan memungkinkan organisasi membangun data warehouse secara bertahap sesuai kebutuhan bisnis dan ketersediaan sumber daya (Kimball & Ross, 2013).

### Perbandingan Pendekatan

Pendekatan top-down menekankan integrasi menyeluruh sejak awal, sedangkan pendekatan bottom-up menekankan pengembangan bertahap. Kedua pendekatan ini dapat dipilih sesuai kebutuhan organisasi (Inmon, 2005; Kimball & Ross, 2013).

## Karakteristik Data Warehouse

Data warehouse memiliki karakteristik khusus yang membedakannya dari basis data operasional. Bill Inmon menjelaskan bahwa data warehouse dibangun dengan empat karakteristik utama, yaitu *subject-oriented*, *integrated*, *time-variant*, dan *non-volatile* (Inmon, 2005). Keempat karakteristik ini menjadi fondasi konseptual dalam memahami data warehouse secara menyeluruh.

### Subject-Oriented

Data warehouse bersifat *subject-oriented*, artinya data disusun berdasarkan subjek utama yang menjadi fokus analisis, seperti penjualan, pelanggan, keuangan, atau persediaan. Berbeda dengan sistem operasional yang berorientasi pada proses bisnis harian, data warehouse berorientasi pada kebutuhan analisis suatu subjek tertentu. Dengan pendekatan ini, data warehouse memudahkan pengguna untuk melihat informasi secara terfokus dan komprehensif sesuai kebutuhan analisis manajerial (Inmon, 2005).

### Integrated

Karakteristik ***integrated*** berarti data dalam data warehouse berasal dari berbagai sumber yang berbeda, tetapi telah diseragamkan dalam satu format yang konsisten. Proses integrasi ini mencakup penyamaan nama atribut, struktur data, satuan pengukuran, serta aturan bisnis.

Integrasi data ini penting untuk memastikan bahwa data yang dianalisis memiliki makna yang sama dan tidak menimbulkan interpretasi yang berbeda akibat perbedaan sumber data (Ponniah, 2010).

|  |
| --- |
|  |

Gambar 2. 6 Alur Integrasi Data Menuju Data Warehouse (Sumber: Diadaptasi dari Estuary (2025), *Benefits of Data Integration*, dan disesuaikan dengan konsep Data Warehouse menurut Inmon (2005) dan Ponniah (2010))

Gambar ini memperlihatkan alur integrasi data dari berbagai sumber menuju satu data warehouse terpusat. Data yang berasal dari beberapa sistem operasional terlebih dahulu diproses melalui tahapan *extract*, *transform*, dan *load* (ETL) sebelum disimpan di dalam data warehouse.

Data warehouse berfungsi sebagai penyimpanan data terintegrasi yang terpisah dari sistem transaksi, sehingga data dapat dimanfaatkan secara optimal untuk keperluan pelaporan, visualisasi, dan *business intelligence*. Pendekatan ini sejalan dengan konsep data warehouse sebagai sistem pendukung pengambilan keputusan yang berorientasi analisis, bukan transaksi (Inmon, 2005; Ponniah, 2010).

### Time-Variant

Data warehouse bersifat ***time-variant*,** yaitu menyimpan data historis dalam rentang waktu tertentu. Setiap data dalam data warehouse memiliki dimensi waktu, seperti tanggal, bulan, atau tahun, sehingga memungkinkan analisis perubahan dan tren dari waktu ke waktu.

Karakteristik ini menjadikan data warehouse sangat berguna dalam analisis jangka panjang, perbandingan periode, dan evaluasi kinerja organisasi berdasarkan data historis (Inmon, 2005).

### Non-Volatile

Karakteristik ***non-volatile*** berarti data yang telah dimasukkan ke dalam data warehouse tidak diubah atau dihapus secara rutin. Data hanya ditambahkan (append) untuk memperkaya data historis yang sudah ada.

Sifat ini memastikan stabilitas data sehingga hasil analisis tidak berubah akibat modifikasi data, berbeda dengan sistem operasional yang terus mengalami pembaruan transaksi (Ponniah, 2010).

## Analogi Sederhana Data Warehouse

|  |
| --- |
| Mengenal tentang Data Warehouse: Definisi hingga Contoh Halaman 1 -  Kompasiana.com |

Gambar 2. 7 Analogi Data Warehouse sebagai Gudang Terpusat (Sumber: Ilustrasi Data Warehouse (Canva), dikutip dari Santoso, P. D. (2023), Kompasiana)

Gambar ini menggambarkan analogi data warehouse sebagai sebuah gudang besar yang berfungsi menyimpan data secara terpusat. Dalam analogi ini, beberapa basis data kecil merepresentasikan sistem operasional (OLTP) yang masing-masing menyimpan data transaksi secara terpisah.

Seperti halnya toko-toko kecil yang memiliki tempat penyimpanan sendiri, sistem OLTP berfokus pada pencatatan dan pemrosesan transaksi harian. Namun, ketika organisasi membutuhkan informasi secara menyeluruh, proses pencarian data akan menjadi tidak efisien jika harus dilakukan pada setiap sistem secara terpisah.

Data warehouse berperan sebagai gudang pusat yang mengumpulkan data dari berbagai sistem operasional tersebut dan menyimpannya dalam kondisi terstruktur dan terintegrasi. Dengan adanya penyimpanan terpusat, informasi dapat diperoleh dengan lebih mudah dan cepat tanpa harus memeriksa setiap sistem satu per satu.

Analogi ini membantu menjelaskan fungsi utama data warehouse sebagai pusat penyimpanan data historis yang mendukung analisis dan pengambilan keputusan, sebagaimana dijelaskan oleh Ponniah (2010) bahwa data warehouse berfungsi sebagai *central repository* untuk kebutuhan analisis.

# BAB III PERBEDAAN DWH DENGAN OLTP

## Pengertian Data Warehouse (DWH)

## Pengertian OLTP (Online Transaction Processing)

## Tujuan Penggunaan DWH dan OLTP

## Perbedaan Karakteristik DWH dan OLTP

## Contoh Perbedaan Penerapan

# BAB IV TAHAPAN DALAM DWH

# BAB V DESAIN DWH

# BAB VI DENORMALISASI

## Pendahuluan

Pada bab-bab sebelumnya, telah diuraikan perbedaan fundamental antara sistem pemrosesan transaksi (OLTP) dan sistem pendukung keputusan (Data Warehouse). Tahap krusial dalam pengembangan Data Warehouse terletak pada perancangan model data (*data modeling*).

Dalam mata kuliah Basis Data Relasional, pendekatan yang ditekankan adalah normalisasi hingga tingkat ketiga (3NF) dengan tujuan meminimalkan redundansi data. Namun, dalam konteks Data Warehouse, pendekatan tersebut tidak dapat diterapkan sepenuhnya. Penggunaan struktur data yang sangat ternormalisasi pada Data Warehouse berpotensi menurunkan kinerja *query* secara signifikan akibat kompleksitas operasi penggabungan (*join*) antar tabel.

Oleh karena itu, standar industri dalam perancangan Data Warehouse mengadopsi pendekatan Dimensional Modeling (Pemodelan Dimensi). Bab ini akan membahas secara komprehensif mengenai konsep pemodelan dimensi, perbandingan arsitektur skema, serta metodologi perancangan menurut Ralph Kimball.

# BAB VII IMPLEMENTASI DENGAN PENTAHO

## Deskripsi Studi Kasus

Dalam bab ini, kita akan mengangkat studi kasus sebuah perusahaan ritel yang sedang berkembang bernama "Toko Maju Jaya". Perusahaan ini memiliki kendala dalam pelaporan manajerial akibat arsitektur data yang terdesentralisasi.

### Latar Belakang Masalah

Perusahaan Toko Maju Jaya memiliki dua cabang toko yang beroperasi di kota berbeda, yaitu Bandung dan Jakarta. Setiap cabang menjalankan sistem operasional (*Point of Sales*) secara mandiri dengan basis data lokal masing-masing.

* Cabang Bandung mencatat transaksi di *database* db\_toko\_bandung.
* Cabang Jakarta mencatat transaksi di *database* db\_toko\_jakarta.

Kondisi ini menimbulkan masalah bagi manajemen pusat, antara lain:

1. Isolasi Data: Laporan penjualan hanya bisa dilihat per cabang, tidak bisa dilihat secara gabungan secara *real*-*time*.
2. Konflik Identitas (*Data Collision*): Karena sistem berjalan terpisah, penomoran ID barang dilakukan secara otomatis (*Auto Increment*) di masing-masing *database*. Hal ini menyebabkan ID Barang 1 di Bandung mungkin merujuk pada "Sabun", sedangkan ID Barang 1 di Jakarta merujuk pada "Susu". Jika data digabung mentah-mentah, akan terjadi kesalahan analisis produk.
3. Kinerja Pelaporan Lambat: *Query* langsung ke *database* operasional (OLTP) untuk laporan bulanan membebani kinerja kasir.

### Arsitektur Solusi Data Warehouse

Untuk mengatasi masalah tersebut, dirancang sebuah *Data Warehouse* terpusat dengan skema bintang (*Star Schema*). Solusi ini melibatkan proses ETL menggunakan *Pentaho Data Integration* (PDI) sebagai jembatan penghubung.

|  |
| --- |
|  |

Gambar 7.1 Arsitektur Integrasi *Data Warehouse*

Spesifikasi transformasi data yang akan dilakukan adalah:

* Penyeragaman Dimensi: Menggabungkan *data master* Barang dan Supplier dari kedua cabang menjadi satu tabel dim\_produk yang unik.
* Standarisasi Lokasi: Membuat tabel dim\_cabang untuk membedakan asal transaksi.
* Agregasi Fakta: Mengubah data transaksi harian yang detail menjadi data bulanan (Agregat) untuk mempercepat pembacaan laporan tren penjualan.

## Persiapan Lingkungan dan Koneksi Database

Tahap teknis pertama dalam implementasi adalah membangun konektivitas. *Pentaho Data Integration* (Spoon) dirancang dengan sifat *database-agnostic*, yang berarti perangkat lunak ini memerlukan *driver* koneksi (JDBC) untuk dapat berinteraksi dan melakukan pertukaran data dengan basis data MySQL.

### Spesifikasi Database

Lingkungan data terdiri dari *database* sumber yang bersifat transaksional (OLTP) dan *database* target yang bersifat analitikal (OLAP).

1. *Database* Sumber

Sumber data berasal dari dua *database* terpisah, yaitu db\_toko\_bandung dan db\_toko\_jakarta. Kedua *database* ini memiliki struktur tabel yang identik, yang terdiri dari:

1. tb\_barang (*Master* Data Barang)

Menyimpan daftar produk yang dijual.

Atribut Kunci: id\_barang (PK), nama\_barang, kategori, merek, harga\_beli, harga\_jual, stok, dan id\_supplier.

1. tb\_transaksi (Data Transaksi *Header*)

Menyimpan informasi umum transaksi penjualan.

Atribut Kunci: id\_transaksi (PK), tanggal\_transaksi, total\_harga, metode\_pembayaran, id\_pelanggan, id\_pegawai, dan id\_cabang.

1. tb\_detail\_transaksi (Data Transaksi Detail)

Berfungsi menyimpan rincian item barang yang dibeli dalam setiap transaksi.

Atribut Kunci: id\_detail (PK), id\_transaksi (FK), id\_barang (FK), jumlah, harga\_satuan, dan subtotal.

1. tb\_cabang (Master Data Lokasi)

Berfungsi menyimpan identitas profil toko cabang.

Atribut Kunci: id\_cabang (PK), nama\_cabang, kota, provinsi, manajer, dan tanggal\_dibuka.

1. Tabel pendukung:

* tb\_supplier: Menyimpan data pemasok (nama\_supplier, alamat, kota, kontak).
* tb\_pelanggan: Menyimpan data pembeli (nama\_pelanggan, no\_telp, kota).
* tb\_pegawai: Menyimpan data karyawan toko (nama\_pegawai, jabatan, gaji).

1. *Database* Target

Basis data tujuan diberi nama dwh\_toko\_bercabang. Basis data ini menggunakan arsitektur *Star Schema* yang terdiri dari tabel dimensi dan tabel fakta yang telah disiapkan untuk menampung hasil proses ETL:

1. dim\_produk (Tabel Dimensi Produk)

Tabel ini merupakan hasil denormalisasi (penggabungan) dari tb\_barang dan tb\_supplier.

Atribut Kunci: id\_produk (*Primary* *Key* baru), nama\_produk, kategori, merek, harga\_satuan, supplier.

Atribut Lineage**:** id\_produk\_jakarta dan id\_produk\_bandung untuk menyimpan ID produk asli dari sumber (Toko Jakarta dan Toko Bandung) untuk pelacakan data.

1. dim\_cabang (Tabel Dimensi Cabang)

Menyimpan data lokasi toko yang sudah distandarisasi.

Atribut: id\_cabang (*Primary Key* baru), nama\_cabang, kota, provinsi, manajer, tanggal\_dibuka.

1. dim\_waktu (Tabel Dimensi Waktu)

Menyimpan hierarki waktu untuk analisis periode, digenerate dari tanggal\_transaksi.

Atribut: id\_waktu (PK), tahun, bulan, nama\_bulan, kuartal.

1. fakta\_penjualan\_bulanan (Tabel Fakta Utama)

Menyimpan performa penjualan terperinci per produk, per cabang, per bulan.

*Foreign Keys*: id\_waktu, id\_cabang, id\_produk.

Measures (Metrik):

* total\_qty: Total jumlah barang terjual.
* total\_penjualan: Total omzet (*Gross Sales*).
* total\_laba: Total keuntungan bersih (Harga Jual - Harga Beli).

1. fakta\_total\_penjualan\_cabang (Tabel Fakta Agregat)

Menyimpan ringkasan total omzet per toko untuk pelaporan level eksekutif.

*Foreign Keys*: id\_waktu, id\_cabang.

Measures (Metrik):

* total\_penjualan: Akumulasi pendapatan cabang.
* total\_transaksi: Jumlah struk yang diterbitkan.
* rata\_rata\_transaksi: Nilai rata-rata belanja per pelanggan.

### Konfigurasi Koneksi Database

Setelah memahami spesifikasi data, langkah teknis selanjutnya adalah mendaftarkan ketiga basis data tersebut ke dalam *Pentaho Data Integration* (Spoon) agar dapat diakses. Berikut merupakan prosedur konfirgurasinya,

1. Membuka *Interface* Koneksi

|  |
| --- |
|  |

Gambar 7.2 *Menu Database Connection* di Spoon

* Buka aplikasi Spoon.
* Pada panel sebelah kiri (*Tab File*), arahkan pada *New*, klik pada folder *Database connections*.

1. Konfigurasi Koneksi Sumber

|  |
| --- |
|  |

Gambar 7.3 Koneksi Cabang Bandung

Buat koneksi baru untuk Cabang Bandung dengan parameter berikut:

* *Connection Name*: db\_toko\_bandung
* *Connection Type*: MySQL
* *Access*: Native (JDBC)
* *Host Name*: localhost
* *Database Name*: db\_toko\_bandung
* *Username*: root (atau sesuaikan dengan *user database* lokal Anda)
* *Port*: 3306

Setelah parameter terisi, tekan tombol *Test*. Jika sukses, simpan koneksi. Lakukan langkah yang sama untuk db\_toko\_jakarta.

1. Konfigurasi Koneksi Target

|  |
| --- |
|  |

Gambar 7.4 Koneksi *Database* Target

Buat koneksi ketiga untuk tujuan penyimpanan data:

* *Connection Name*: dwh\_toko\_bercabang
* *Database Name*: dwh\_toko\_bercabang

Pastikan ketiga koneksi tersebut muncul di daftar *Database connections* sebelum melanjutkan ke tahap transformasi.

## Implementasi ETL Tabel Dimensi

Pada sub-bab ini, kita akan membedah *file* transformasi dwh\_dim.ktr yang berfungsi mengisi *data master*.

Dalam arsitektur *Data Warehouse*, pengisian data harus mengikuti urutan yang ketat. Tabel dimensi wajib diisi terlebih dahulu sebelum tabel fakta. Hal ini dikarenakan tabel fakta mengandung *Foreign Key* yang merujuk pada tabel dimensi. Jika dimensi kosong, proses pengisian fakta akan gagal (*referential integrity violation*).

|  |
| --- |
|  |

Gambar 7.5 Transformasi Tabel Dimensi

Gambar di atas memperlihatkan desain transformasi dwh\_dim.ktr secara keseluruhan. Pada *canvas* tersebut, terlihat tiga alur data yang berjalan secara paralel dan independen untuk mengisi tabel dimensi. Alur bagian atas berfungsi membangkitkan dimensi waktu menggunakan *query* SQL, alur tengah menggabungkan data identitas cabang dari dua sumber berbeda, sedangkan alur bawah menyatukan data produk dan *supplier*. Ketiga alur tersebut bermuara pada komponen *Insert* / *Update*, yang bertugas memastikan data master tersimpan ke dalam *database* dwh\_toko\_bercabang tanpa menciptakan duplikasi data.

### Strategi Loading: Insert / Update

Fungsi *Insert* digunakan jika sistem menemukan data baru (misalnya produk baru ditambahkan di toko), maka baris baru akan dibuat di *Data Warehouse*. Sedangkan Fungsi *Update*, digunakan jika data sudah ada namun atributnya berubah (misalnya perubahan nama produk), maka sistem hanya akan memperbarui baris tersebut.

### Transformasi Dimensi Cabang (dim\_cabang)

Transformasi ini bertujuan untuk menyatukan identitas toko dari kedua basis data sumber ke dalam satu tabel dimensi lokasi (dim\_cabang). Data ini penting untuk memungkinkan analisis perbandingan performa antar-toko.

|  |
| --- |
|  |

Gambar 7.6 Transformasi Dimensi Cabang

Gambar di atas menampilkan detail alur transformasi untuk dimensi cabang. Terlihat dua komponen *Table Input* yang bekerja secara simultan mengambil data profil toko dari *database* Bandung dan Jakarta.

Langkah-langkah:

* 1. *Query* Ekstraksi Pada step *Table Input* untuk cabang, kita menggunakan *Query* SQL sederhana untuk mengambil seluruh atribut identitas toko yang diperlukan dari tabel sumber (tb\_cabang).

|  |
| --- |
| SELECT  nama\_cabang,  kota,  provinsi,  manajer,  tanggal\_dibuka  FROM tb\_cabang |

Kode Program 7.1 *Query* SQL Tabel *Input* Cabang

Di dalam setiap komponen *input*, *query* SQL tersebut dijalankan pada masing-masing koneksi *database* (db\_toko\_bandung dan db\_toko\_jakarta). Kedua aliran data ini kemudian disatukan menuju komponen akhir *Insert* / *Update*. Pada komponen tersebut, kolom nama\_cabang digunakan sebagai kunci pencarian (*lookup key*) untuk memastikan data master cabang dari berbagai lokasi fisik terkonsolidasi menjadi satu daftar referensi tunggal tanpa duplikasi.

* 1. Konfigurasi *Lookup* & *Update*

Pada bagian *Key(s) to look up the value(s)*, kolom nama\_cabang ditetapkan sebagai acuan utama. Sistem akan membandingkan nama cabang dari data sumber dengan data yang ada di tabel dimensi.

|  |
| --- |
|  |

Gambar 7.7 Konfigurasi *Insert* / *Update* Cabang

**Kunci Pencarian**: nama\_cabang = nama\_cabang

Di bagian *Update fields*, seluruh atribut deskriptif seperti kota, provinsi, manajer, dan tanggal\_dibuka diatur statusnya menjadi Y (*Update*). Artinya, jika di kemudian hari terjadi pergantian manajer atau koreksi data lokasi pada *database* sumber, *Data Warehouse* akan otomatis memperbarui informasi tersebut tanpa membuat baris data ganda. Hal ini sangat krusial untuk menjaga integritas data lokasi agar laporan perbandingan antar-cabang selalu valid.

### Transformasi Dimensi Produk (dim\_produk)

Dimensi produk menggabungkan data barang dan *supplier* menjadi satu tabel denormalisasi.

|  |
| --- |
|  |

Gambar 7.8 Transformasi Dimensi Produk

Gambar di atas menampilkan detail alur transformasi untuk dimensi produk. Terlihat dua komponen *Table Input* yang bekerja secara simultan mengambil *data master* barang dari *database* Bandung dan Jakarta.

Langkah-langkah:

1. *Query* Ekstraksi

Pada step *Table Input* untuk produk, kita menggunakan *Query* SQL yang melakukan *JOIN* antara tabel barang dan *supplier* untuk mendapatkan nama pemasok secara langsung.

|  |
| --- |
| SELECT  b.nama\_barang,  b.kategori,  b.merek,  b.harga\_jual AS harga\_satuan,  s.nama\_supplier AS supplier  FROM tb\_barang b  JOIN tb\_supplier s ON b.id\_supplier = s.id\_supplier |

Kode Program 7.2 *Query* SQL Tabel *Input* Produk

Di dalam setiap komponen *input*, *query* SQL telah dikonfigurasi untuk melakukan *JOIN* antara tabel barang dan *supplier*, sehingga data yang mengalir sudah memuat informasi pemasok. Kedua aliran data ini kemudian disatukan menuju komponen akhir *Insert* / *Update*, di mana terjadi proses unifikasi data ke dalam tabel dim\_produk.

1. Konfigurasi *Lookup* & *Update*

Pada bagian *Key(s) to look up the value(s)*, kolom nama\_produk ditetapkan sebagai acuan utama. Sistem akan membandingkan nama produk yang masuk dari data sumber dengan data yang sudah tersimpan di tabel dimensi.

|  |
| --- |
|  |

Gambar 7.9 Konfigurasi *Insert* / *Update* Produk

**Kunci Pencarian**: nama\_produk = nama\_produk

Di bagian *Update fields*, atribut deskriptif seperti kategori, merek, harga\_satuan, dan supplier diatur statusnya menjadi Y (*Update*). Pengaturan ini sangat penting untuk menangani perubahan data bisnis; misalnya, jika harga barang naik atau terjadi pergantian *supplier* di *database* toko, *Data Warehouse* akan secara otomatis memutakhirkan informasi tersebut tanpa perlu membuat baris produk baru yang duplikat.

Selain kolom-kolom yang terlihat pada tangkapan layar di atas, konfigurasi ini juga memetakan kolom id\_barang dari masing-masing sumber ke kolom *data lineage* (id\_produk\_bandung dan id\_produk\_jakarta) yang berada di urutan bawah tabel. Hal ini dilakukan untuk menjaga jejak asal data (*traceability*) dari sistem OLTP ke *Data Warehouse*.

### Transformasi Dimensi Waktu (dim\_waktu)

Dimensi waktu mengonversi data tanggal dari tabel transaksi menjadi satu tabel referensi hierarkis (tahun, bulan, kuartal) untuk keperluan analisis periodik.

|  |
| --- |
|  |

Gambar 7.10 Transformasi Dimensi Waktu

Gambar di atas menampilkan konfigurasi ekstraksi untuk dimensi waktu. Berbeda dengan dimensi lainnya, dimensi waktu dibangkitkan (*generated*) langsung dari data transaksi menggunakan fungsi manipulasi tanggal.

Langkah-langkah:

1. *Query* Generasi Waktu

Pada step *Table Input*, kita menggunakan fungsi bawaan SQL (*YEAR*, *MONTH*, *CASE*) untuk memecah tanggal transaksi menjadi atribut hierarki waktu yang lebih deskriptif (Tahun, Bulan, Kuartal).

|  |
| --- |
| SELECT DISTINCT  YEAR(tanggal\_transaksi) AS tahun,  MONTH(tanggal\_transaksi) AS bulan,  CASE MONTH(tanggal\_transaksi)  WHEN 1 THEN 'Januari'  WHEN 2 THEN 'Februari'  WHEN 3 THEN 'Maret'  WHEN 4 THEN 'April'  WHEN 5 THEN 'Mei'  WHEN 6 THEN 'Juni'  WHEN 7 THEN 'Juli'  WHEN 8 THEN 'Agustus'  WHEN 9 THEN 'September'  WHEN 10 THEN 'Oktober'  WHEN 11 THEN 'November'  WHEN 12 THEN 'Desember'  END AS nama\_bulan,  CASE  WHEN MONTH(tanggal\_transaksi) BETWEEN 1 AND 3 THEN 'Q1'  WHEN MONTH(tanggal\_transaksi) BETWEEN 4 AND 6 THEN 'Q2'  WHEN MONTH(tanggal\_transaksi) BETWEEN 7 AND 9 THEN 'Q3'  WHEN MONTH(tanggal\_transaksi) BETWEEN 10 AND 12 THEN 'Q4'  END AS kuartal  FROM tb\_transaksi  ORDER BY tahun, bulan |

Kode Program 7.3 *Query* SQL Generasi Dimensi Waktu

Hasil eksekusi *query* di atas akan menghasilkan daftar periode unik yang pernah terjadi transaksi. Data tersebut kemudian dialirkan ke komponen *Insert* / *Update* dengan menggunakan kombinasi kolom tahun dan bulan sebagai kunci utama. Hal ini menjamin bahwa satu periode bulan (misalnya: Januari 2025) hanya tercatat satu kali di dalam tabel dim\_waktu.

1. Konfigurasi *Lookup* & *Update*

Pada bagian *Key(s) to look up the value(s)*, kombinasi kolom tahun dan bulan ditetapkan sebagai kunci pencarian utama. Sistem akan memeriksa apakah pasangan Tahun dan Bulan tersebut sudah ada di *database*.

|  |
| --- |
|  |

Gambar 7.11 Konfigurasi *Insert* / *Update* Waktu

**Kunci Pencarian**: tahun = tahun DAN bulan = bulan

Penggunaan dua kunci ini mutlak diperlukan untuk memastikan keunikan data. Sebagai contoh, angka bulan "1" (Januari) pasti akan muncul berulang kali setiap tahunnya (Jan 2024, Jan 2025). Dengan mengunci kombinasi Tahun+Bulan, sistem menjamin tidak akan ada duplikasi periode yang sama. Di bagian *Update fields*, atribut deskriptif seperti nama\_bulan dan kuartal diatur statusnya menjadi Y (*Update*). Ini memastikan kelengkapan data referensi waktu tersimpan dengan benar di *Data Warehouse*.

## Implementasi ETL Tabel Fakta

# BAB VIII IMPLEMENTASI DENGAN PHP

# BAB IX TOOL DATA VISUALISASI

# BAB X DATA QUALITY

## Signifikansi Kualitas Data dalam Data Warehouse

## Dimensi Kualitas Data (Kriteria Data Standar)

## Sumber Polusi Data (Penyebab Kerusakan Data)

## Alat dan Proses Perbaikan Kualitas Data

## Peran Master Data Management (MDM)

# DAFTAR PUSTAKA