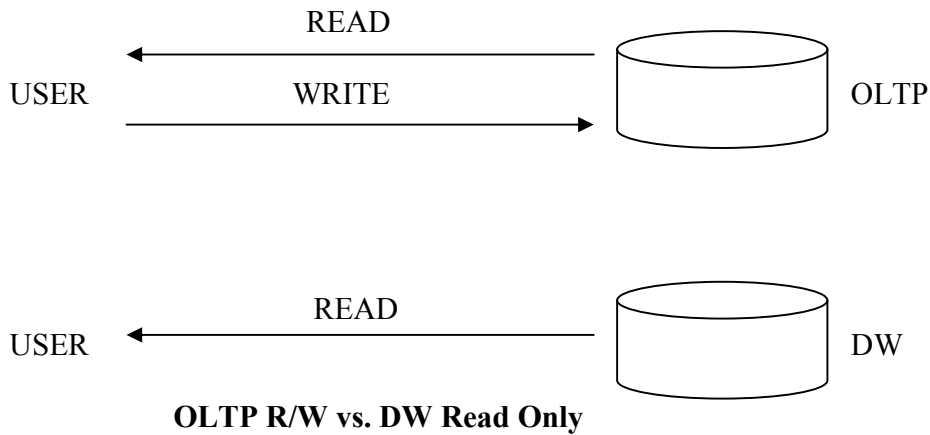


Bab 1 : PENDAHULUAN DATA WAREHOUSE

1.1. Perbedaan Aktifitas Data warehouse dan OLTP

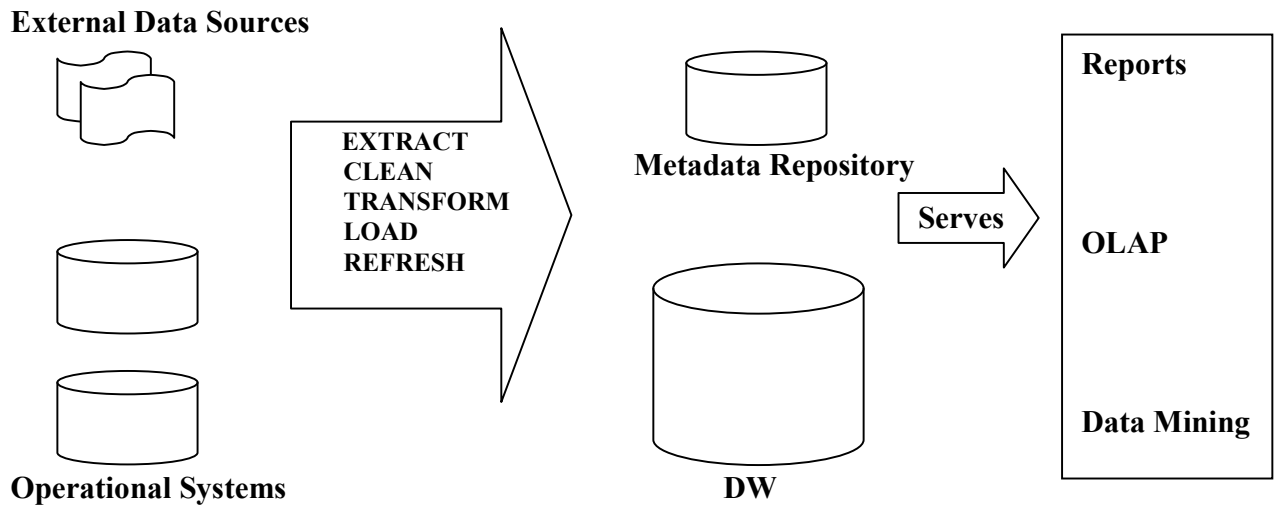


1.2. Operational Systems vs Data Warehousing Systems

Operational	Data Warehouse
Data saat ini	Menangani data historis
Data bersifat dinamis	Data bersifat statik
Akses Read/Write	Akses Read only
Untuk keperluan Transaksi sehari-hari	Untuk keperluan analisa
Berorientasi pada Aplikasi	Berorientasi pada Subyek
Used by clerical staff for day-to-day operations	Digunakan oleh top manager untuk proses analisa
Normalisasi data (ER model)	Denormalisasi data (model dimensional)
Harus dioptimasi untuk query dalam bentuk yang singkat	Query melibatkan data dalam jumlah besar (query lebih kompleks).

1.3. Arsitektur Data Warehouse

Berikut arsitektur Data Warehouse:

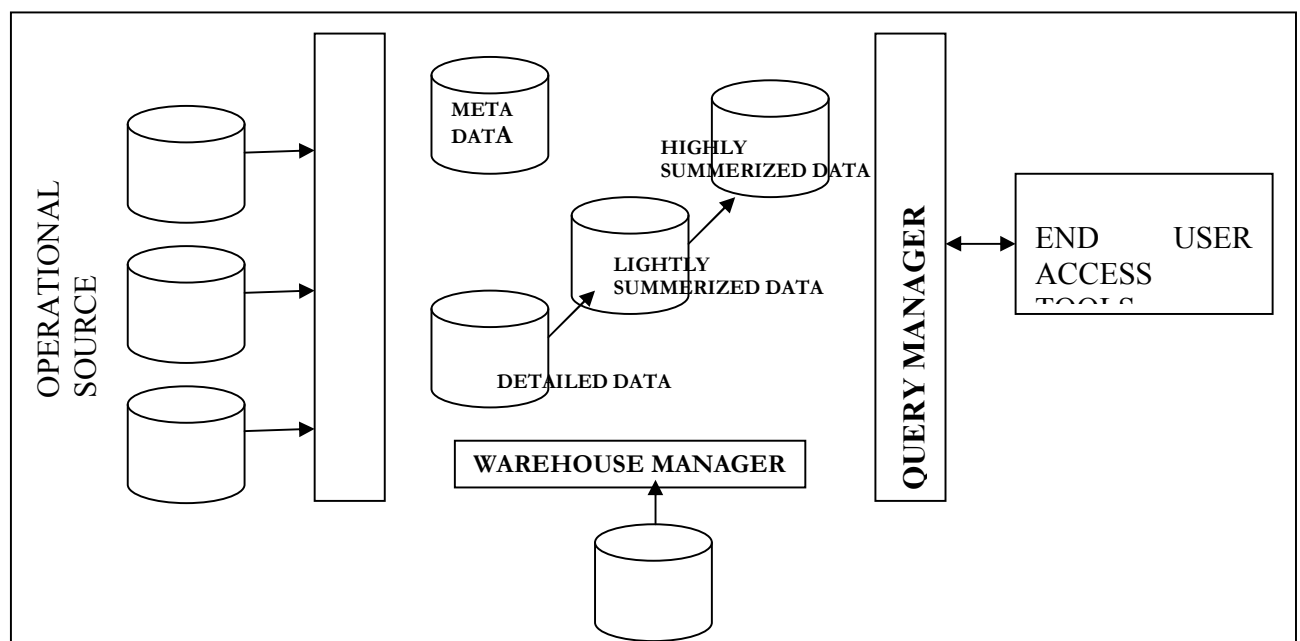


Bab 2 : Arsitektur dan Komponen Data Warehouse

Data yang pada data warehouse berasal dari sistem operasional dan juga dari sumber lain dari luar (external sources). Semua sumber data ini disebut dengan *source systems*. Data *diekstrak* dari source systems lalu disimpan dalam suatu tempat yang dinamakan *data staging area*, dimana data dibersihkan, ditransformasikan, dikombinasikan, diduplikasi untuk dapat siap di-load ke dalam data warehouse. Pada data staging area juga terdapat aktifitas seperti pengurutan. Bagian dari sistem yang menyediakan query dan layanan presentasi disebut dengan *presentation server*. Presentation server merupakan mesin target dimana data di-load dari data staging area kemudian disimpan untuk dilakukan query secara langsung oleh end users, penyiapan laporan dan aplikasi yang lain. Pada dasarnya Data Warehouse membutuhkan tiga sistem yang berbeda yaitu :

1. Source Systems
2. Data Staging Area
3. Presentation servers

Fungsi dari tiap sistem telah dijelaskan sebelumnya.



2.2. Komponen Data Warehouse

Komponen dari Data Warehouse terdiri dari :

1. DATA OPERASIONAL

Sumber data untuk data warehouse diambil dari :

- (i) Data dari sistem mainframe
- (ii) Data juga didapatkan dari DBMS relasional seperti Oracle, Informix.
- (iii) Juga sumber dari luar (external data) yang didapatkan dari database komersial

2. LOAD MANAGER

Load manager bertugas membentuk semua operasi yang berhubungan dengan proses ekstraksi dan loading data ke dalam data warehouse. Operasi yang terlibat termasuk transformasi sederhana, dan aktifitas lain sampai data siap di-load ke data warehouse.

3. WAREHOUSE MANAGER

Warehouse manager bertanggung jawab atas manajemen data dalam warehouse.

Tugasnya antara lain :

- (i) Menganalisa data dan memastikan terdapat konsistensi
- (ii) Transformasi dan merging (menggabungkan) sumber data dari penyimpanan sementara (temporary storage) ke dalam tabel yang ada di data warehouse
- (iii) Membuat indexes dan view pada base tabel (tabel master)
- (iv) Melakukan denormalisasi
- (v) Men-generate agregasi (operasi sum atau count)
- (vi) Membbackup dan mengarsip data

Pada situasi tertentu, warehouse manager juga bertanggung jawab untuk mengusulkan *query profiles* yang menentukan tipe indeks mana dan agregasi mana yang perlu dibentuk

4. QUERY MANAGER

Query manager bertanggung jawab terhadap semua operasi yang berhubungan dengan manajemen query user. Komponen ini dibangun dengan menggunakan tools-tools seperti OracleBI SpreadSheet Add-In, JDeveloper,dll.

5. DETAIL DATA

Merupakan area dalam warehouse yang menyimpan semua detail data dalam skema database. Pada kebanyakan kasus, detail data tidak tersedia secara online tapi diaggregasi ke level detail berikutnya. Detail data ditambahkan secara regular ke dalam warehouse sebagai tambahan pada data yang telah diaggregasi.

6. DATA YANG TER-SUMMARY

Area data warehouse menyimpan semua data yang teraggregasi yang digenerate oleh warehouse manager. Tujuan dari informasi yang di-summary ini adalah untuk mempercepat performansi query. Data yang tersummary di-update secara kontinyu setiap kali ada data baru yang perlu di-load ke dalam warehouse.

7. MEM-BACKUP DAN MENGARSIP DATA

Area dari data warehouse yang menyimpan data detail dan data yang telah tersummary digunakan untuk tujuan pengarsipan dan backup. Data ini ditransfer ke dalam tempat penyimpanan seperti magnetic tapes atau optical disks.

8. META DATA

Data warehouse juga menyimpan semua definisi Meta data (data yang menerangkan tentang data) yang digunakan oleh semua proses dalam warehouse.

Metadata digunakan untuk berbagai macam tujuan, meliputi :

- (i) Proses ekstraksi dan load – Meta data digunakan sebagai mapping dari sumber ke tampilan informasi yang ada di warehouse.
- (ii) Proses manajemen warehouse– Meta data digunakan menghasilkan tabel summary secara otomatis.
- (iii) Sebagai bagian dari proses Manajemen Query, Metadata digunakan untuk mengarahkan query ke sumber data yang benar.

Struktur dari Meta data bisa berbeda untuk setiap proses, dikarenakan tujuan yang berbeda pula.

9. END-USER ACCESS TOOLS

Prinsip dari tujuan data warehouse adalah untuk menyediakan informasi ke manajer bisnis untuk proses pengambilan keputusan secara strategis. User yang menggunakan Data warehouse berinteraksi dengan menggunakan end user access tools. Berikut contoh dari beberapa end user access tools:

- (i) Tool untuk Pelaporan dan Query
- (ii) Tool Pengembangan Aplikasi
- (iii) Tool Sistem Informasi Eksekutif
- (iv) Tool untuk OLAP (Online Analytical Processing)
- (v) Tool untuk Data Mining

2.3. Proses E T L (EXTRACT TRANSFORMATION LOAD)

2.3.1. EKSTRAK

Beberapa elemen dalam basis data operasional berperan penting dalam pengambilan keputusan, tapi beberapa yang lain tidak, karena itu perlu dilakukan ekstraksi data yang relevan yang berguna dalam proses pengambilan keputusan. Yang perlu diperhatikan adalah:

- (i) File dan tabel mana yang perlu diakses dari basis data sumber ?
- (ii) Field mana yang perlu diekstrak dari file dan tabel tersebut ?
- (iii) Bagaimana format outputnya ?
- (iv) Jadwal dari proses ekstraksi yang dijalankan.

2.3.2. TRANSFORM

Kadangkala terdapat ketidakkonsistenan pada sumber data yang ada, karena itu perlu dilakukan proses transformasi. Ketidak konsistenan bisa dalam penamaan field yang tidak sama ataupun pengkodean data yang tidak sama. Misal nama pegawai diberi nama EMP_NAME di satu database, dan diberi nama ENAME di database yang lain. Beberapa konversi yang perlu untuk dilakukan :

- (i) Karakter dikonversi dari ASCII ke EBCDIC atau sebaliknya.
- (ii) Teks yang campur huruf besar kecil diubah semua ke dalam huruf besar.
- (iii) Data numerik harus dikonversi ke dalam format yang standar
- (iv) Format data harus standart.
- (v) Satuan yang digunakan (misal Rp/ \$)
- (vi) Koding data (Male/ Female, M/F) harus dikonversi ke dalam format yang sama.

Salah satu tools yang bisa digunakan untuk keperluan ini adalah produk komersial **DataMAPPER** yang dibuat oleh Applied Database Technologies.

2.3.3. CLEANSING atau Pembersihan data

Kualitas informasi adalah pertimbangan penting dalam menentukan nilai dari suatu informasi. Developer Data warehouse harus memastikan data yang dimasukkan dalam data warehouse bebas dari error. Proses ini disebut dengan **Data Cleansing**.

Data Cleansing berhubungan dengan beberapa tipe kesalahan. Misal adanya data yang hilang, data yang tidak benar (nomer telpon yang salah ketik), data yang tidak konsisten, data yang saling konflik jika melibatkan dua atau lebih sumber data. Beberapa algoritma yang digunakan untuk membersihkan data dibahas pada mata kuliah Data Mining.

2.3.4. LOADING

Proses Loading menyatakan perpindahan data dari komputer yang menyimpan basis data sumber ke lokasi tempat disimpan data dalam data warehouse. Proses ini langsung dikerjakan setelah proses ekstraksi dan pembersihan data. Tool yang digunakan antara lain Oracle Warehouse Builder merupakan salah satu tool di Oracle yang menyediakan fitur untuk mebuat operasi ETL pada Oracle Data Warehouse.

Bab 3

Pemodelan Data Warehouse

Pemodelan Dimensional adalah suatu nama dari teknik desain logika yang digunakan untuk data warehouse. Berbeda dengan Entity Relational Modeling (ER Modelling) yang digunakan untuk menyimpan data transaksi pada system OLTP.

Perbedaan ER Modelling dengan Dimensional Modelling

ER Modelling digunakan untuk mendukung proses operasional dalam suatu database. Bentuk konvensional dari ER Model ini terdiri dari ;

- Menghilangkan redundansi pada data model.
- Memfasilitasi pemasukan suatu data dengan beberapa pengenalan(identifiers), dan
- Optimasi performansi OLTP

Pemodelan Dimensional didesain untuk mendukung pelaporan dan analisa yang dibutuhkan dalam system data warehouse.

Kenapa ER Modelling tidak cocok untuk Data Warehouse ?

- End user tidak dapat memahami atau mengingat suatu ER Model. End User tidak dapat menavigasikan ER Model. Tidak ada interface(GUI) yang dapat mengambil ER Model secara umum dan dapat dipergunakan oleh end user.
- ER Modelling tidak dioptimasi untuk queri yang koplek, ad hoc. Aka tetapi dioptimasi untuk queri yang berulang-ulang.
- Menggunakan teknik ER Modelling menghilangkan basic fungsi dari Data Warehouse.

Pengenalan Dimensional Modelling

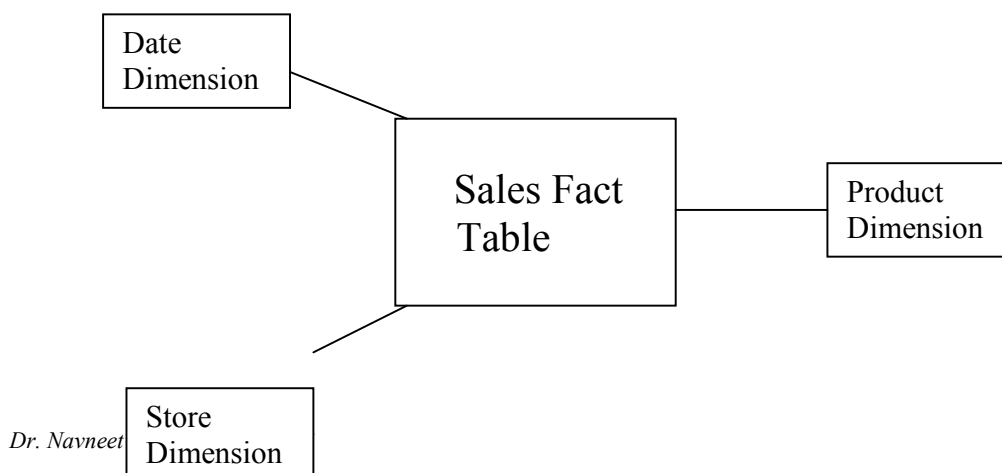
Tujuan dari Dimensional Modelling ini adalah untuk mempresentasikan suatu set pengukuran bisnis dalam kerangka standar sehingga dapat lebih mudah dipahami oleh end user. Dimesional Model berisi informasi yang mirip dengan ER Model akan tetapi data dalam format symmetric yang didesain untuk :

- User Understanability
- Query Performance
- Resilience to Change

Komponen utama dari Dimensional Modelling ini adalah tabel fakta dan tabel dimensi. Tabel fakta adalah primary tabel yang dalam beberapa dimensi model yang dimaksudkan untuk pengukuran suatu bisnis.

Suatu fakta bergantung pada beberapa factor, sebagai contoh **sale amount**, suatu fakta, yang bergantung pada produk, lokasi dan waktu. Factor factor ini adalah yang disebut dengan dimensi.

Skema star



Catatan poin-poin penting tentang Skema Star:

- Schema yang paling populer untuk mendesain Data Warehouse adalah *Skema star*
- Masing-masing dimensi disimpan pada satu *dimensi tabel* dan masing-masing masukan diberi identifikasi yang unik.
- Tabel dimensi berhubungan pada satu atau lebih *tabel fakta*.
- Tabel fakta mengandung satu kunci gabungan yang tersusun dari primary key dari tabel dimensi.
- Tabel fakta juga mengandung fakta sekitar kombinasi tertentu dari dimensi. Antara lain satu kombinasi dari *store_key*, *date_key* dan *product_key* memberikan sejumlah satu produk tertentu dijual pada satu hari tertentu pada penyimpanan tertentu.
- Tabel fakta punya foreign keys terhadap semua dimensi tabel dalam satu Skema Star. Dalam contoh di situ ada tiga foreign keys (date key, product key, and store key).
- Tabel fakta dibuat normal, sedangkan tabel dimensi tidak
- Tabel fakta sangat besar dibandingkan dengan tabel dimensi.

Mendesain Model Dimensi

Step 1 - Select the Business Process

The first step in the design is to decide what business process (es) to model by combining an understanding of the business requirements with an understanding of the available data

Step 2 - Declare the Grain

Once the business process has been identified, the data warehouse team faces a serious decision about the granularity.

The grain of a fact table represents the level of detail of information in a fact table. Declaring the grain means specifying exactly what an individual fact table record represents.

In the star schema discussed above, the most detailed data would be transaction line item detail in the sale receipt.

(date, time, product code, product name, price/unit, number of units, amount)

18-SEP-2002, 11.02, p1, dettol soap, 15, 2, 30

But in the above dimensional model we provide sales data rolled up by product (all records corresponding to the same product are combined) in a store on a day. A typical fact table record would look like this:

18-SEP-2002, Product1, Store1, 150, 600

This record tells us that on 18th Sept. 150 units of Product1 was sold for Rs. 600 from Store1.

Step 3 – Choose the Dimensions

Once the grain of the fact table has been chosen, the date, product, and store dimensions are readily identified. It is often possible to add more dimensions to the basic grain of the fact table, where these additional dimensions naturally take on only one value under each

combination of the primary dimensions. If the additional dimension violates the grain by causing additional fact rows to be generated, then the grain must be revised to accommodate this dimension.

Step 4 – Identify the Facts

The first step in identifying fact tables is where we examine the business, and identify the transaction that may be of interest. In our example the electronic point of sale (EPOS) transactions give us two facts, quantity sold and sale amount.

3.2. Kelebihan Pemodelan Multidimensi

The dimensional model has a number of important data warehouse advantages that the ER model lacks. Its strengths are:

1. The dimensional model is a predictable, standard framework. Report writers, query tools, and end user interfaces can all make strong assumptions to make the user interfaces more understandable and to make processing more efficient
2. Star schema can withstand changes in user behavior. The logical design can be done independent of the expected query patterns.
3. It is gracefully extensible to accommodate new data elements and new design decisions.
 - All existing tables can be changed by either adding new data rows or by alter table commands. Data should not have to be reloaded.
 - No query or reporting tool needs to be reprogrammed to accommodate the change
 - Old applications continue to run without yielding different results.
4. Standard approaches available for handling common modeling situations in the business world.
5. Support for aggregates. Aggregates are summary records that are logically redundant with base level data already in the data warehouse, but are used to enhance query performance.
6. A dimensional model can be implemented in a relational database, a multi-dimensional database or even an object-oriented database.

Bab 4

Multidimensional Databases and MOLAP

Proses bisnis dalam kaitannya dengan data multi dimensi adalah menanyakan hal tentang penjualan **product** pada **regions** atau **daerah** yang berbeda untuk **periode waktu / time** yang spesifik.

Dimension yang terlibat : Product, Region, Time period
Measure pada tabel Fakta : Sale (nilai penjualan)

Multi Dimensional Database dan selanjutnya disingkat **MDDDB** adalah sistem perangkat lunak yang didesain untuk memungkinkan proses penyimpanan yang nyaman dan efisien dan sistem pemanggilan dari data yang berukuran besar yang bersifat :+

1. Sangat berkaitan
 2. Dapat disimpan, ditampilkan dan dianalisa dalam **perspektif** atau tinjauan yang berda.
- Perspektif ini disebut dengan **Dimensi**.

Contoh kasus

Perusahaan automobil ingin meningkatkan volume penjualannya. Untuk itu perusahaan tersebut perlu melihat data histori penjualan dalam bentuk multi dimensi semisal :

- Nilai Penjualan berdasarkan model
- Nilai Penjualan berdasarkan warna
- Nilai Penjualan berdasarkan nama Dealer
- Nilai Penjualan berdasarkan waktu tertentu

Analisa data mencakup pertanyaan atau query semisal :

Apa trend penjualan untuk periode waktu tertentu sehubungan dengan spesifikasi model, warna yang dipilih untuk setiap nama dealer yang ada ?

Perusahaan tersebut memiliki data penjualan sebagai berikut :

Nilai penjualan untuk nama dealer GLEASON

MODEL	COLOR	SALES VOLUME
MINI VAN	BLUE	6
MINI VAN	RED	5
MINI VAN	WHITE	4
SPORTS COUPE	BLUE	3
SPORTS COUPE	RED	5
SPORTS COUPE	WHITE	5
SEDAN	BLUE	4
SEDAN	RED	3
SEDAN	WHITE	2

Sales Volumes

MODEL	Mini Van	6	5	4
	Coupe	3	5	5
	Sedan	4	3	2
		Blue	Red	White
		COLOR		

Matriks diatas berupa array 2 dimensi. Array merupakan komponen penting dari MDDB.

Dalam array, setiap axis disebut dengan dimensi (MODEL & COLOR)

Setiap elemen dalam dimensi menempati satu **posisi**.

Untuk model, ada 3 posisi, van, sedan, dan coupe.

Untuk color, ada 3 posisi, blue, white, dan red.

Konsekuensi dari Semakin kompleksnya Tabel Relasional

Jika kita tambahkan field baru, yaitu nama dealer ke dalam tabel relasional, maka tabel berikut tentu kelihatan aneh jika dipresentasikan ke user :

SALES VOLUMES FOR ALL DEALERSHIPS

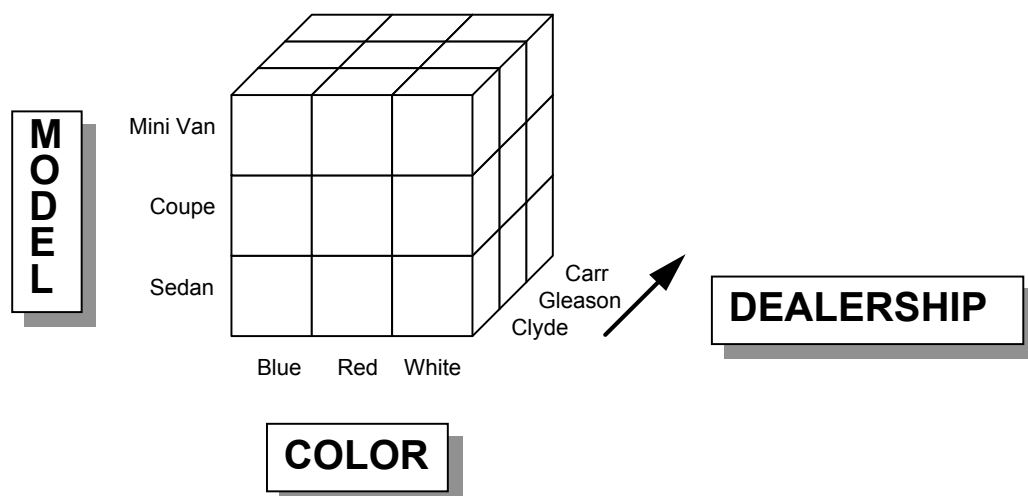
MODEL	COLOR	DEALERSHIP	VOLUME
MINI VAN	BLUE	CLYDE	6
MINI VAN	BLUE	GLEASON	6
MINI VAN	BLUE	CARR	2
MINI VAN	RED	CLYDE	3
MINI VAN	RED	GLEASON	5
MINI VAN	RED	CARR	5
MINI VAN	WHITE	CLYDE	2
MINI VAN	WHITE	GLEASON	4
MINI VAN	WHITE	CARR	3
SPORTS COUPE	BLUE	CLYDE	2
SPORTS COUPE	BLUE	GLEASON	3
SPORTS COUPE	BLUE	CARR	2
SPORTS COUPE	RED	CLYDE	7

SPORTS COUPE	RED	GLEASON	5
SPORTS COUPE	RED	CARR	2
SPORTS COUPE	WHITE	CLYDE	4
SPORTS COUPE	WHITE	GLEASON	5
SPORTS COUPE	WHITE	CARR	1
SEDAN	BLUE	CLYDE	6
SEDAN	BLUE	GLEASON	4
SEDAN	BLUE	CARR	2
SEDAN	RED	CLYDE	1
SEDAN	RED	GLEASON	3
SEDAN	RED	CARR	4
SEDAN	WHITE	CLYDE	2
SEDAN	WHITE	GLEASON	2
SEDAN	WHITE	CARR	3

Penyelesaian

Kita harus menambahkan dimensi baru yang dinamakan Dealer ke dalam database. Sekarang array yang ada berukuran 3 dimensi. Jika ada 3 dealer, maka array sekarang berukuran 3x3x3 dengan 27 sel). Sebelumnya berukuran 2-Dimensi dengan 3x3 = 9 sel).

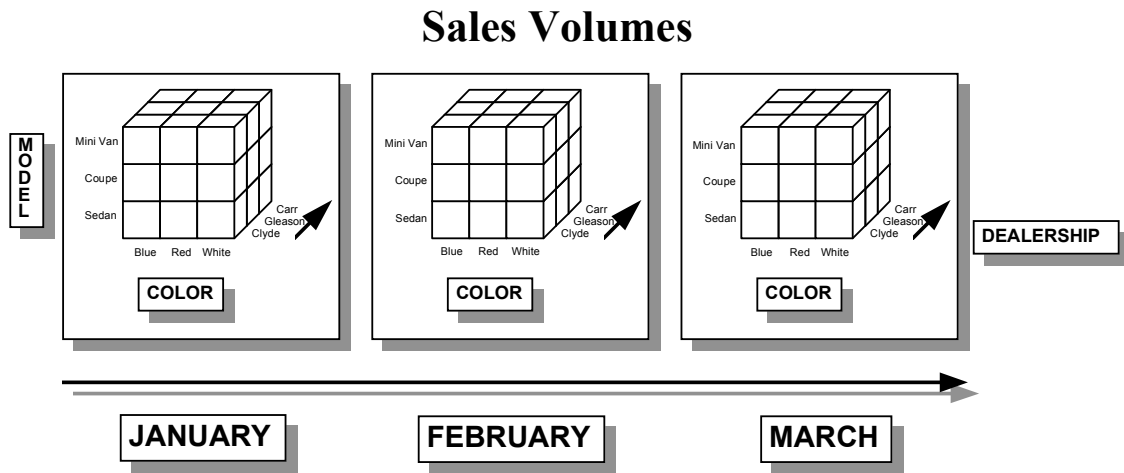
Sales Volumes

**Keuntungan Performansi**

Pertimbangkan array 10x10x10 (jika setiap dimensi terdapat 10 posisi). Jika seorang user ingin mencari nilai penjualan dari sedan berwarna biru yang dijual oleh dealer Gleason. Maka dalam sistem relasional kita harus mencari dari 1000 record untuk mendapatkan data yang dicari. Tapi dalam sistem multidimensi, sistem hanya perlu mencari 3 dimensi dari 10 posisi untuk mencari record yang sesuai. Jadi maksimum terdapat 30 pencarian dibanding dengan maksimum 1000 pencarian dari sistem relasional.

Menambahkan Dimensi

Model 3Dimensi dapat diperluas menjadi empat dimensi, dst. Misal kita tambahkan dimensi waktu untuk bulan penjualan, maka gambar dari data multi dimensi akan menjadi seperti ini :



Bab 5 :

Surrogate keys

Surrogate key adalah bilangan integer yang terurut yang diperlukan untuk mengumpulkan data pada tabel dimensi. Misal, produk pertama diberi nilai 1, produk berikutnya diberi nilai 2, dan seterusnya.

Untuk setiap record pada tabel master, tandai surrogate key (bilangan integer yang dimulai dari 1) secara berurutan. Proses sederhana ini akan membaca secara berurutan data yang berikutnya.

Production Key	Surrogate Key
Prod 1	1
Prod 2	2
Prod 3	3

Dan seterusnya..

Kunci untuk Tabel Fakta

Misal data penjualan sebagai berikut :

(location1, prod1, time3, units, amount)

Maka data ini dimasukkan sebagai satu baris data dalam tabel fakta sales sebagai berikut:

(1, 6, 3, units, amount)

Dimana 1 adalah surrogate key untuk location1, 6 surrogate key untuk prod1, dan 3 adalah surrogate key untuk time3. Sedangkan units dan amount karena merupakan measure tidak merubah nilainya.

Buffering Data Warehouse dari Perubahan Operasional

Organisasi pada umumnya, adanya suatu kode yang kadaluwarsa atau tidak aktif lagi bisa dipakai ulang atau di-reused. Sebagai contoh, bank mungkin mereused nomer rekening jika terdapat nomer yang tidak aktif untuk periode waktu yang lama atau nomer rekening tersebut telah ditutup oleh customer. Perubahan tersebut tidak akan berpengaruh pada

system operasional, karena system operasi tidak menyimpan data histori. Sedangkan data warehouse menyimpan data untuk waktu yang lama, sehingga dengan adanya reused akan memunculkan konflik. Penggunaan dari surrogate keys akan membantu kita untuk membuat record baru dalam tabel dimensi yang memiliki nilai berbeda dari kebanyakan atribut.

Penghematan Ruang Penyimpanan (Space)

Jika surrogate keys hanya memiliki panjang 4 bytes, maka akan sedikit membutuhkan ruang penyimpanan dari pada production keys yang dinyatakan dalam bilangan alfanumerik. Jika tipe data alfanumerik membutuhkan 8 bytes panjangnya. Sehingga dengan menggunakan surrogate keys kita akan bisa menghemat 4 bytes. Jika terdapat 1 billion records dalam tabel fakta, maka kita bisa menghemat $4 \times 1 \text{ billion bytes} = 3.73 \text{ GB}!!!!$

Bab 6

Perubahan Data pada Dimensi

Untuk merespon perubahan data pada dimensi, ada 3 pilihan:

Cara I: Mengganti data lama dengan data baru, dengan mengabaikan histori data.

Cara II: Menambahkan dimensi baru tambahan dengan menggunakan surrogate key yang baru.

Cara III: Membuat field tambahan yang menerangkan data lama dan data baru yang sekarang dipakai.

Cara I: Mengganti data lama dengan data baru

Cara I mudah untuk dijalankan, tetapi tidak mempertahankan history apapun dari nilai attribute yang terdahulu. Misal produk Intellikidz 1.0 yang tadinya menjadi bagian dari departemen education sekarang menjadi departemen strategy

Maka kita ganti data untuk record ini :

Surrogate key	Description	Department	Natural Key
12345	Intellikidz 1.0	Education	ABC922-Z

Menjadi seperti ini :

Surrogate key	Description	Department	Natural Key
12345	Intellikidz 1.0	Strategy	ABC922-Z

Cara II: Menambahkan dimensi baru tambahan

Tabel Fakta				Tabel Dimensi				
		1234567						
		1234567		1234567	Navneet	Goyal	Single	CUST11111
		1234567						
				1234600	Navneet	Goyal	Married	CUST11111
		1234600						
		1234600						

Cara III: Membuat field tambahan yang menerangkan data lama dan data baru yang sekarang dipakai.

Surrogate key	Description	Department	Prior Dept.	Natural Key
---------------	-------------	------------	-------------	-------------

12345	Intellikidz 1.0	Strategy	Education	ABC922-Z
-------	-----------------	----------	-----------	----------

Struktur Indeks

Index adalah sekumpulan struktur data yang mengambil property sebagai input dari record - pada umumnya nilai dari satu atau lebih field dan menemukan record dengan property secara cepat. Index memperbolehkan kita menemukan record tanpa harus mencari tidak lebih dari bagian semua record yang mungkin. Field pada index berdasarkan pemanggilan kunci yang dicari.

Terdapat banyak perbedaan struktur data yang menyediakan sebagai index :

1. Primary index pada file terurut
2. Secondary index pada file tak terurut
3. B-trees, pada umumnya digunakan index untuk beberapa file
4. Hash indexes

Index dapat diklasifikasikan sebagai berikut :

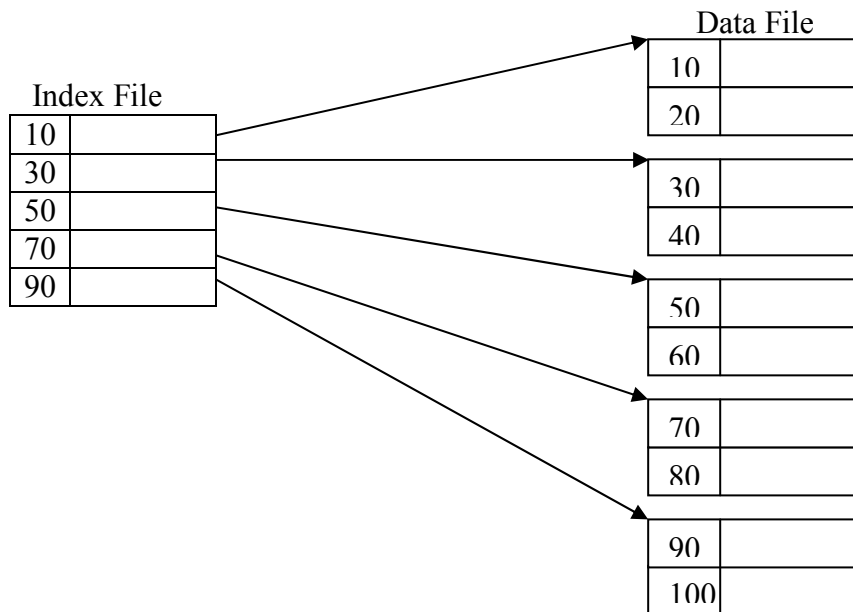
- Single-Level
Contoh : primary, secondary, clustering
- Multi-Level
Contoh : : ISAM (Indexed Sequential Access Method), B-tree, B⁺-tree

Index dapat juga diklasifikasikan sebagai :

- Dense
Jika file index berisi nomor record sama dengan data file
- Sparse
Jika file index berisi nomor record kurang dari data file

1. Primary Indexes

When the search key is a key of the relation, we call the index as *primary* index, and when the search key is not a key of the relation, the index is called *clustering* index.

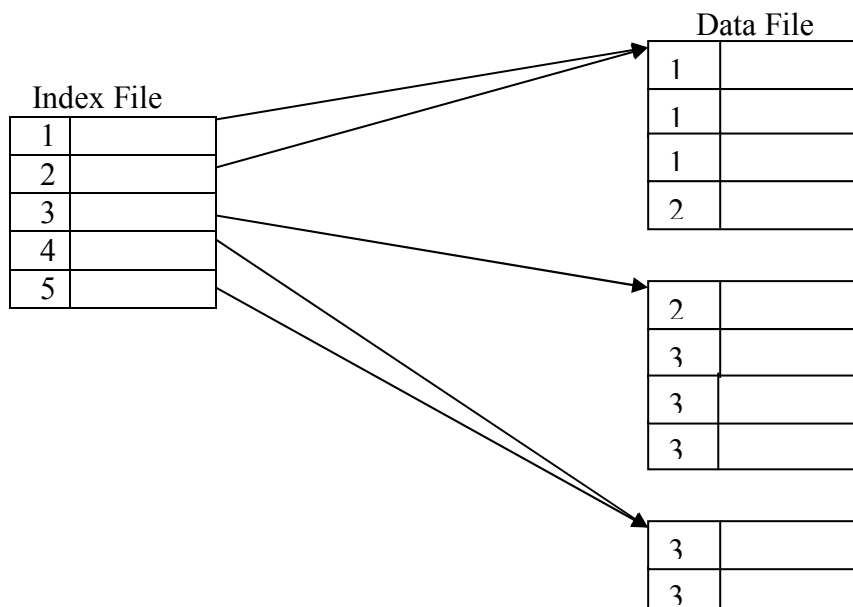


Points to note:

- Primary index requires that the ordering field of the data file have a distinct value for each record.
- Primary index is sparse
- Contains as many records as there are blocks* in the data file (there are 5 blocks in this example and each block can hold only 2 records).
- The first record in each block of the data file is called *anchor record* of the block, or simply *block anchor*.
- There can be only one primary index on a table

2. Clustering Indexes

The clustering index file, like the primary index file, has two fields. The first field contains distinct values of the clustering field, and the second field contains block pointers. The block pointer points to the first block in the data file that has a record with that value for its clustering field.



4	
5	

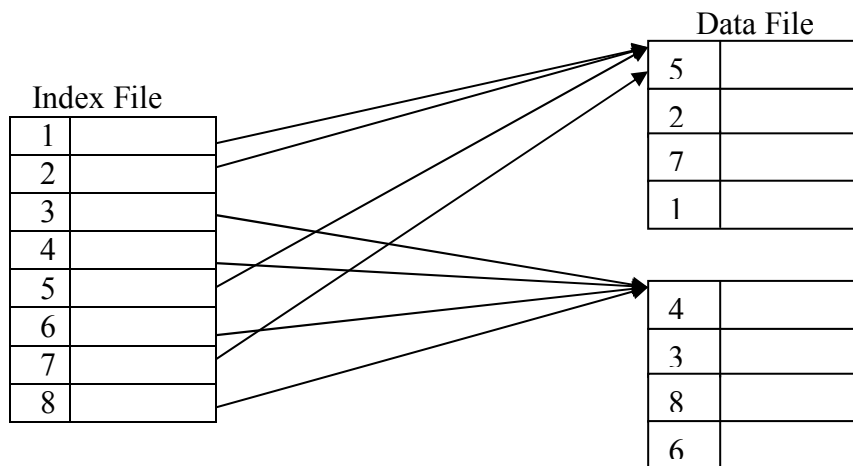
Clustering index is always non-dense.

3. Secondary Indexes

Secondary indexes do not require the data file to be sorted on the indexed field. They can be created on both key and nonkey fields.

Secondary Index on Key Field

The key field on which a secondary index is created is often called a secondary key. There is one index entry for each record in the data file., which contains the value of the secondary key and a pointer to either the block in which the record is stored or to the record itself. Such an index is dense.



Secondary index on a key field

Secondary Index on Nonkey Field

We can also create a secondary index on a nonkey field of a file. There are two options for implementing such an option:

- Option 1 is to include several index entries with the same index field value- one for each record. This would be a dense index

Data File

<i>Emp#</i>	SSN	Name	Dept #	DOB	SALARY
			3		
			5		
			1		
			3		
			2		
			3		
			4		

			5		
			3		

Index File would look like this:

1	
2	
3	
3	
3	
3	
4	
5	
5	

- Option 2 is to have variable length records for the index entries, with a repeating field for the pointer-one pointer to each block that contains a record with matching indexing field value. This would be a non-dense index.

The index File would look like this:

1	B ₁ (1)
2	B ₂ (1)
3	B ₃ (1), B ₃ (2), B ₃ (3), B ₃ (4)
4	B ₄ (1)
5	B ₅ (1)

Where B_i (n) (i represents the indexing field value and n takes value from 1 to number of matching records for that indexing field value)

Single-Level Indexes

Summary of Single-Level Indexes

Types of Indexes

	Ordering Field	Nonordering Field
Key Field	Primary Index	Secondary Index (key)
Nonkey Field	Clustering Index	Secondary Index (nonkey)

Properties of Index Types

Type of Index	Number of Index Entries	Dense or Sparse	Block Anchoring
Primary	No. of blocks in data file	Sparse	Yes
Clustering	No. of distinct index field values	Sparse	Yes/no*
Secondary (key)	Number of records in data file	Dense	No
Secondary (nonkey)	No. of records** No. of distinct index field values***	Dense Sparse	No

* Yes if every distinct value of the ordering field starts from a new block; no otherwise

** Untuk Option 1

*** Untuk Option 2

Bab 9 Bitmap Indexes

Data Warehouse biasanya menyimpan jumlah besar dari data. Data ini kalau sering terpakai untuk OLAP. Waktu respon pendek sangat dibutuhkan untuk mendukung keputusan online. Ada banyak cara untuk menambahkan kinerja dari satu data warehouse. Bitmap Index adalah salah satunya.

Dua keuntungan penting Bitmaps :

1. Bitmap Index mengijinkan operasi bit efisien untuk menjawab queries
2. Bitmap Index dapat jauh lebih ringkas dibandingkan tree tradisional index b + dan dan lebih baik dalam menggunakan teknik kompresi

Cara pengkodean bitmaps sebagai berikut:

Example 1: Consider a relation R with 12 records (*i.e.* cardinality=12) having an attribute A.

$\Pi_A(R)$	B^0	B^1	B^2	B^3	B^4	B^5	B^6	B^7	B^8
3	0	0	0	1	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	1
2	0	0	1	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	1	0
5	0	0	0	0	0	1	0	0	0
6	0	0	0	0	0	0	1	0	0
4	0	0	0	0	1	0	0	0	0

This is a value-list index. We can write the above bitmap alternatively as:

0	000000010000
1	001000000000
2	010101100000
3	100000000000
4	000000001001
5	000000000100
6	000000000010
7	000000001000
8	000010000000

We can see that there are 9 bitmaps (each corresponding to one distinct value), and in each bitmap the position of “1” gives the record number. For example, consider the bitmap corresponding to value 2. It indicates that record numbers 2, 4, 6, and 7 have value 2 in the field A

Example 2: Who buys gold jewelry? Consider a database of customers that contains information about their name, address, age and salary. Assuming that age and salary are the only relevant factors. Suppose our database has the following 12 jewelry customers:

Age	Salary
25	60
45	60
50	75
50	100
50	120
70	110
85	140
30	260
25	400
45	350
50	275
60	260

Creating bitmaps on Age and Salary

25	1 0 0 0 0 0 0 0 1 0 0 0
30	0 0 0 0 0 0 0 0 1 0 0 0
45	0 1 0 0 0 0 0 0 0 1 0 0
50	0 0 1 1 1 0 0 0 0 0 1 0
60	0 0 0 0 0 0 0 0 0 0 0 1
70	0 0 0 0 0 1 0 0 0 0 0 0
85	0 0 0 0 0 0 1 0 0 0 0 0

60	1 1 0 0 0 0 0 0 0 0 0 0
75	0 0 1 0 0 0 0 0 0 0 0 0
100	0 0 0 1 0 0 0 0 0 0 0 0
110	0 0 0 0 0 1 0 0 0 0 0 0
120	0 0 0 0 1 0 0 0 0 0 0 0
140	0 0 0 0 0 0 1 0 0 0 0 0

260	0 0 0 0 0 0 0 1 0 0 0 0
275	0 0 0 0 0 0 0 0 0 0 1 0
350	0 0 0 0 1 0 0 0 0 1 0 0
400	0 0 0 0 0 0 0 0 1 0 0 0

Suppose we want to find the jewelry buyers having age 50 and salary 120. To find this, take the bitmap corresponding to age 50 and salary 120

Age 50: 0 0 1 1 1 0 0 0 0 0 1 0

Sal 120: 0 0 0 0 1 0 0 0 0 0 0 0

AND : 0 0 0 0 1 0 0 0 0 0 0 0

This tells us that only the 5th record in the table satisfies the above condition. The above query is an example on *equality* query. Bitmaps are also useful for *range* queries. (Selection queries include equality and range queries)

Consider the following query:

Find the jewelry buyers with an age in the range 45-55 and a salary in the range 100-200.

Age 45: 0 1 0 0 0 0 0 0 0 1 0 0

Age 50: 0 0 1 1 1 0 0 0 0 0 1 0

OR : 0 1 1 1 1 0 0 0 0 1 1 0 → bit-vector representing age between 45-55

Sal 100: 0 0 0 1 0 0 0 0 0 0 0 0

Sal 110: 0 0 0 0 0 1 0 0 0 0 0 0

Sal 120: 0 0 0 0 1 0 0 0 0 0 0 0

Sal 140: 0 0 0 0 0 0 1 0 0 0 0 0

OR : 0 0 0 1 1 1 1 0 0 0 0 0 → bit-vector representing salary between 100-200

0 1 1 1 1 0 0 0 0 1 1 0
0 0 0 1 1 1 1 0 0 0 0 0

AND 0 0 0 1 1 0 0 0 0 0 0 0

This tells us that only the 4th and the 5th records, which are (50, 100) and (50, 120), are in the desired range.