BINUS UNIVERSITY GRADUATE PROGRAM

People
Innovation
Excellence

LECTURE NOTES

7023T Advanced Database System

Session 07
Designing the Physical Database and Planning for Performance

People
Innovation
Excellence

LEARNING OUTCOMES

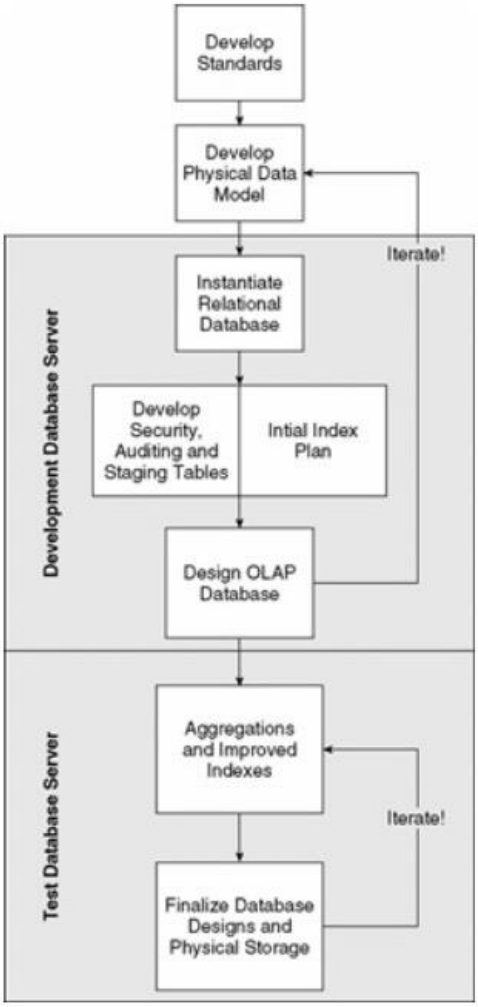
- Peserta diharapkan mampu memahami proses desain fisik database.
- Peserta diharapkan dapat menjelaskan bagaimana melakukan standarisasi dalam perancangan database.
- Peserta diharapkan mampu mengidentifikasi berbagai beberapa teknik indeks dan penerapannya pada database.
- Peserta diharapkan mampu memahami beberapa teknik agregasi dan bagaimana optimalisasi agregasi.

OUTLINE MATERI (Sub-Topic):

1. *High level physical design process*
2. *Developing Standards*
3. Indeks
4. Agregasi
5. *Data Modelling Tools*

High level physical design process

Pada pembahasan sebelumnya telah dijelaskan proses desain database logis (*logical database design*). *Physical design process* adalah proses yang dilakukan untuk mengubah desain logis menjadi database fisik. Pada prakteknya detail implementasinya akan berbeda untuk setiap platform maupun proyek yang berbeda, hal ini diakibatkan perkembangan dalam teknologi perangkat keras dan perangkat lunak sangat pesat. Gambar 1 memperlihatkan *high level model* dari *physical design process*, dimulai dengan mendefinisikan standar penamaan dan database, kemudian membangun model fisik pada server pengembangan (*development server*). Setelah menentukan konfigurasi *security*, *auditing*, indeks, dan tabel *staging*, selanjutnya dapat dimulai tahap desain database OLAP (Online Analytical Processing). Proses desain fisik harus diakhiri dengan pengujian pada server khusus (*test server*) yang memiliki konfigurasi yang sama dengan server yang akan digunakan pada saat operasional nanti.



Gambar 1. *High level physical design process*

Developing Standards

Proses desain fisik (*physical design*) dimulai dengan mendefinisikan standar dalam hal konvensi penamaan tabel dan kolom (nama yang digunakan harus sama dengan yang didefinisikan pada model logis – *logical model*, nilai null (harus dihindari pada tabel dimensi), struktur direktori dan nama file, serta *primary* dan *foreign key*. Seperti yang sudah dijelaskan pada bagian sebelumnya, tabel dimensi disarankan untuk menggunakan *surrogate key* yang bertipe bilangan bulat sebagai *primary key*.

Setelah mendefinisikan standar, selanjutnya dilakukan pengembangan model fisik. Sangat direkomendasikan untuk menggunakan model logis (*logical model*) sebagai langkah awal dari model fisik. Dengan pendekatan ini diharapkan model fisik benar-benar merepresentasikan model logis. Satu hal yang pertama kali perlu dilakukan adalah memperluas *dimensional modelling worksheet* dengan menambahkan informasi model fisik, seperti diilustrasikan pada gambar 2.

Table Name:	DimOrderInfo
Table Type	Dimension
View Name	OrderInfo
Description	OrderInfo is the "junk" dimension that includes miscellaneous information about the Order transaction
Used in schemas	Orders
Generate script?	Y

	Target									
Column Name	Description	Datatype	Size	Key?	FK To	NULL?	Default Value	Unknown Member	Example Values	SCD Type
OrderInfoKey	Surrogate primary key	smallint		PK ID		N		-1	1, 2, 3, 4...	
BKSalesReasonID	Sales reason ID from source system	smallint				N		-1		
Channel	Sales channel	char	8					Unknown	Reseller, Internet, Field Sales	1
SalesReason	Reason for the sale, as reported by the customer	varchar	30					Unknown		1
SalesReasonType	Type of sales reason	char	10					Unknown	Marketing, Promotion, Other	1
AuditKey	What process loaded this row?	int		FK	Audit Dim	N		-1		1

Gambar 2a. *Detailed dimensioal design (target table)*

Source						
Source System	Source Schema	Source Table	Source Field Name	Source Datatype	ETL Rules	Comments
ETL Process					Standard surrogate key	
OEI	Sales	SalesReason	SalesReasonID	int	Convert to char, left-pad with zero, R for reseller row.	We need to insert a single row for Reseller
OEI	Sales	SalesReason	Derived		"Internet" for real sales reasons, "Reseller" for reseller row.	
OEI	Sales	SalesReason	Name	nvarchar(50)	Convert to varchar; "Reseller" for reseller row.	
OEI	Sales	SalesReason	ReasonType	nvarchar(50)	Convert to varchar; "Reseller" for reseller row.	
Derived					Populated by ETL system using standard technique	

Gambar 2b. *Detailed dimensioal design (source table)*

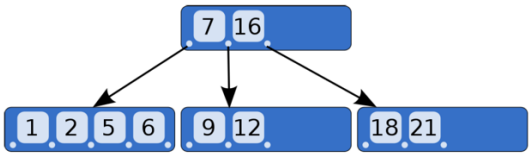
Kegiatan berikutnya adalah memperkirakan ukuran database yang akan disimpan pada *harddisk.* Hal yang penting untuk diingat disini adalah ukuran dari tabel fakta adalah sekitar 90% dari ukuran data warehouse. Kalkulasi perlu dimulai dengan memperkirakan ukuran byte dari setiap baris pada tabel fakta, kemudian mengalikannya dengan perkiraan jumlah barisnya. Estimasi tersebut sebaiknya juga memperhitungkan ruang penyimpanan yang diperlukan untuk penyimpanan tabel *staging*, tabel audit, tabel *acces monitoring*, dan tabel *security*.

Indeks

Setelah memperkirakan kebutuhan ruang penyimpanan, selanjutnya dilakukan perencanaan strategi indeks. Pada tahap ini ditentukan kolom apa saja yang perlu diindeks pada setiap tabel dan jenis indeks seperti apa yang akan digunakan pada masing-masing tabel. Tujuan dari tahapan ini adalah untuk menerapkan indeks terhadap kolom-kolom yang paling sering digunakan dalam BI *query* maupun *report*. Jenis indeks yang paling populer dalam RDBMS adalah B-Tree, indeks *clustered*, indeks *non-clustered* dan indeks *bitmapped.*

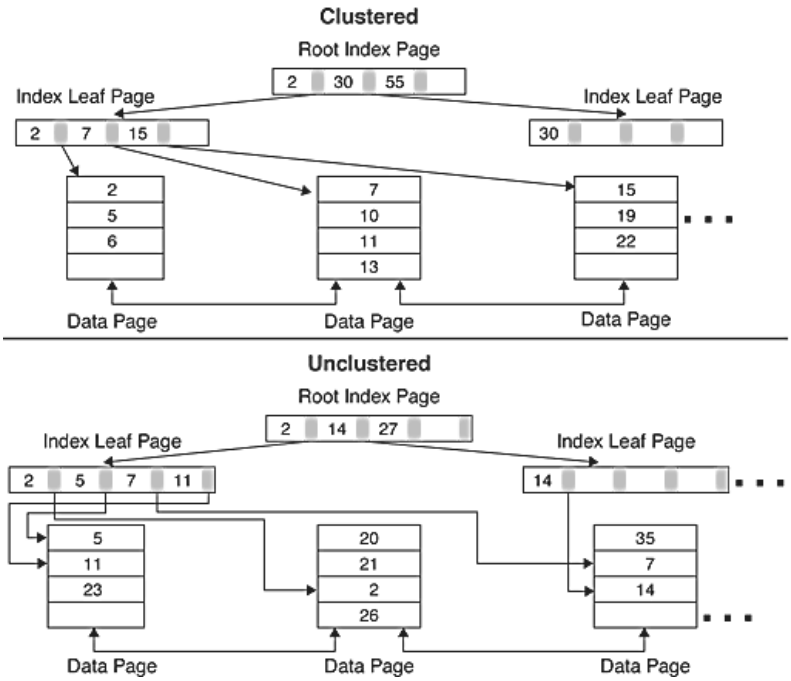
B-Tree adalah struktur data *tree* yang menyimpan data dengan cara terurut sehingga memungkinkan proses pencarian, akses sekuensial, penyisipan, dan penghapusan data dapat dilakukan dengan kompleksitas logaritmik. Gambar 3 mengilustrasikan bagian dari sebuah B-tree, nilai 7 dan 16 disimpan secara terurut pada node induk. Diantara dua nilai tersebut diletakkan sebuah anak panah menuju ke node anak yang mengandung data dengan nilai

yang lebih besar dari 7 dan lebih kecil dari 16. Selain itu, di sebelah kiri dari nilai 7 ada anak panah panah menuju node yang berisi data dengan nilai kurang dari 7. Demikian juga di sebelah kanan nilai 16, ada anak panah menuju node yang berisi data dengan nilai lebih besar dari 16. Semua node anak dibawahnya mengikuti aturan tersebut. Implementasi B-tree untuk indeks database dimana setiap data merepresentasikan sekumpulan baris yang memiliki nilai yang diindeks oleh B-tree. Indeks B-tree bekerja secara optimal pada kolom dari sebuah tabel yang memiliki banyak nilai yang berbeda.



Gambar 3. Contoh indeks B-tree

Indeks *clustered* adalah salah satu jenis metode indeks yang mengurutkan data pada tabel secara fisik pada media penyimpanan. Oleh karena itu sebuah tabel hanya boleh memiliki satu indeks *clustered*. Node-node *leaf* dari sebuah indeks *clustered* mengandung *data pages.* Pada indeks *non-clustered*, urutan logis dari index tidak sama dengan urutan fisik dari setiap baris data pada media penyimpanan. Node-node *leaf* dari indeks *non-clustered* tidak mengandung *data pages*, melainkan indeks dari tiap baris data. Gambar 4 memperlihatkan perbedaan antara indeks *clustered* dan *non-clustered.*



Gambar 4. Contoh indeks *clustered* versus *non-clustered*

Indeks *bitmap* adalah jenis indeks yang memanfaatkan bitmap atau seringkali disebut sebagai bit *array*. Pemrosesan *query* pada tabel dengan indeks *bitmap* dilakukan dengan cara melakukan operasi logika terhadap bit *array*. Indeks ini bekerja optimal pada kolom-kolom yang meemiliki kardinalitas rendah, atau tidak mengandung banyak perbedaan nilai diantara baris data yang satu dengan lainnya. Indeks *bitmap* tidak membutuhkan ruang yang besar dan menunjukkan kinerja yang lebih baik dibanding jenis indeks lainnya pada tabel dengan kardinalitas rendah. Namun indeks ini kurang efisien dibandingkan indeks B-tree maupun indeks *clustered* khusunya pada tabel dengan frekuensi *update* tinggi, oleh karena itu indeks *bitmap* secara umum lebih tepat digunakan untuk *query* yang cepat terhadap data *read-only* seperti data warehouse atau OLAP, dan kurang cocok untuk database OLTP. Gambar 5 memperlihatkan contoh penggunaan indeks *bitmap* untuk kolom gender dan income_level. Jumlah bit yang dibutuhkan untuk *bitmap* adalah sejumlah data, dalam contoh tersebut 5.

record number	*name*	*gender*	*address*	*income_level*
0	John	m	Perryridge	L1
1	Diana	f	Brooklyn	L2
2	Mary	f	Jonestown	L1
3	Peter	m	Brooklyn	L4
4	Kathy	f	Perryridge	L3

Bitmaps for *gender*

m	10010
f	01101

Bitmaps for *income_level*

L1	10100
L2	01000
L3	00001
L4	00010
L5	00000

Gambar 5. Contoh penggunaan indeks *bitmap*

Agregasi

Faktor yang memiliki kontribusi terbesar terhadap kinerja dari data warehouse berukuran besar adalah operasi agregat. Pada umumnya sistem DW/BI memiliki lebih dari satu operasi agregat, masing-masing untuk mengelompokkan data berdasarkan setiap dimensinya. Operasi agregat dapat dilakukan dengan beberapa pendekatan, diantaranya:

- Menyimpan hasil agregat pada tabel terpisah, tabel tersebut merupakan hasil eksekusi *query* dengan pengelompokkan berdasarkan dimensi tertentu (*GROUP BY*).
- Menyimpan hasil agregat pada *materialized view* (hanya dapat dilakukan pada sistem database Oracle). *Materialized view* adalah *view* yang secara fisik disimpan pada database, dengan demikian memungkinkan proses indeks, partisi, maupun pembatasan akses.

- Memanfaatkan pengelolaan agregat dari OLAP engine. Pendekatan ini sangat optimal namun kurang fleksibel dibanding dua pendekatan sebelumnya dalam hal indeks maupun penyimpanan secara fisik.
- Memanfaatkan fungsi navigasi terhadap hasil agregasi yang disediakan oleh tool BI. Pendekatan ini sangat efisien, namun hanya dapat digunakan pada lingkungan yang mendukung.

Hal pertama yang penting untuk dilakukan adalah memilih operasi agregasi yang tepat. Kita tidak mungkin menghitung semua kemungkinan operasi agregat, karena hal ini akan menyita ruang penyimpanan yang sangat besar. Di lain pihak, terlalu sedikit operasi agregat akan meningkatkan respon dari *query* namun beberapa operasi agregat harus dieksekusi saat dibutuhkan. Menentukan operasi agregat yang paling optimal biasanya membutuhkan analisa terhadap pola *query* yang sering diminta oleh pengguna. Beberapa database OLAP telah menyediakan tools untuk mengatasi permasalahan ini.

Setelah dipilih operasi agregat yang paling tepat, selanjutnya perlu ditentukan strategi *update* pada saat terjadi penambahan baris data baru pada tabel fakta. Pengguna tidak diperbolehkan mengakses agregat sebelum proses *update* terhadap agregat selesai, jika tidak maka akan terjadi inkonsistensi.

Data Modelling Tools

Data modelling tools dapat sangat membantu dalam *dimensional modelling* dan desain model fisik. Perangkat tersebut dilengkapi dengan kemampuan untuk menghasilkan *script* untuk pembuatan obyek-obyek database berdasarkan informasi dari model data. Setelah terkoneksi dengan database, obyek pada lingkiungan database dapat dibuat secara otomatis. Selain itu, *data modelling tools* juga dilengkapi dengan kemampuan untuk melakukan *reverse engineering* terhadap obyek database menjadi *script* yang dilengkapi dengan opsi menyimpan hasil proses sebagai metadata. Beberapa contoh *data modelling tools* diantaranya Embarcadero-ERStudio, ErWin, dan OracleDesigner.

SIMPULAN

- Proses *high level physical design* sebaiknya dilakukan secara iteratif agar semua obyek dapat dibuat dengan baik.
- Setiap obyek database yang akan dibuat harus mengikuti standar yang sudah ditetapkan pada desain logis, hal ini dimaksudkan agar adanya konsistensi antara desain logis dengan desain fisik.
- Indeks B-tree efektif untuk digunakan pada tabel yang memiliki banyak perbedaan nilai pada kolom yang diindeks, sebaliknya indeks bitmap lebih cocok digunakan pada tabel dengan sedikit perbedaan nilai.

BINUS UNIVERSITY GRADUATE PROGRAM

People
Innovation
Excellence

DAFTAR PUSTAKA

1. Kimball, R. (2008). *The Data Warehouse Lifecycle Toolkit*. John Wiley & Sons.
2. Kimball, R., & Ross, M. (2011). *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. John Wiley & Sons.
3. Inmon, W. H. (2005). *Building the Data Warehouse*. John wiley & sons.