BRIN
BADAN RISET
DAN INOVASI NASIONAL

Modul
BASIS DATA
Pelatihan Jabatan Fungsional
Analis Data Ilmiah

Fungsi Layanan Pengembangan Kompetensi Kedinasan
Direktorat Pengembangan Kompetensi
Deputi Sumber Daya Manusia Ilmu Pengetahuan dan Teknologi
2022

Penanggung Jawab:

1. Edy Giri Racman Putra, Ph.D.
2. Nining Setyowati Dwi Andayani, S.E., M.M.
3. Raden Arthur Ario Lelono, Ph.D.
4. Alpha Fadila Juliana Rahman, S.Pd., M.Pd.

Tim Penyusun Modul:

1. Probo Herawani S.Kom., M.T.I.
2. Andre Sihombing S.Kom., M.Sc.
3. Canggih Pramono Gultom S.Kom. MCS.
4. Farham Harvianto M. Kom.
5. Endang Febrian Khusnul Hidayati S.Si., M.Si.
6. Naily Kamaliah, M.Si.

Diterbitkan oleh:

Direktorat Pengembangan Kompetensi - BRIN

Gedung B.J. Habibie, Jalan M.H. Thamrin Nomor 8,

Jakarta Pusat 10340

Diterbitkan pertama kali tahun 2022

KATA PENGANTAR

Pengembangan kompetensi Aparatur Sipil Negara (ASN) khususnya Pegawai Negeri Sipil (PNS) dalam mengembangkan karier jabatan fungsionalnya menjadi suatu tuntutan sehingga mampu menjalankan tugas dan fungsinya dengan sebaik – baiknya sehingga mampu memberikan kontribusi yang nyata terhadap bangsa dan Negara Kesatuan Republik Indonesia (NKRI).

Badan Riset dan Inovasi Nasional (BRIN) sebagai Instansi pembina 11 (sebelas) jabatan fungsional yang meliputi peneliti, perekayasa, pengembangan teknologi nuklir, analis perkebunrayaan, analis pemanfaatan iptek, analis data ilmiah, kurator koleksi hayati, penata penerbitan ilmiah, teknisi perkebunrayaan, teknisi penelitian dan perekayasaan, dan pranata nuklir. BRIN melalui kedeputian Sumber Daya Manusia Ilmu Pengetauan dan Teknologi (SDMI) bertanggung jawab dalam penyelenggaraan pengembangan kompetensi 11 (sebelas) jabatan fungsional tersebut.

Kedeputian SDMI – BRIN melalui Direktorat Pengembangan Kompetensi sebagai penyelenggara pengembangan komptensi jabatan fungsional bertanggung jawab dalam menyiapkan kebutuhan tersebut baik berupa pengelolaan pembelajaran, fasilitator, modul, bahan ajar dan sebagainya, yang merujuk kepada regulasi peraturan BRIN nomor 28 tahun 2022 tentang pedoman pelatihan pembentukan jabatan fungsional peneliti, peraturan BRIN nomor 29 tahun 2022 tentang pedoman pelatihan jabatan fungsional kurator koleksi hayati, peraturan BRIN nomor 30 tahun 2022 tentang pedoman pelatihan jabatan fungsional analis pemanfaatan iptek, peraturan BRIN nomor 31 tahun 2022 tentang pedoman pelatihan jabatan fungsional analis data ilmiah, peraturan BRIN nomor 32 tahun 2022 tentang pedoman pelatihan jabatan fungsional analis perkebunrayaan, peraturan BRIN nomor 33 tahun 2022 tentang pedoman pelatihan jabatan fungsional teknisi perkebunrayaan, dan peraturan BRIN nomor 34 tahun 2022 tentang pedoman pelatihan jabatan fungsional penata penerbitan ilmiah.

Kami megucapkan syukur ke hadirat Tuhan Yang Maha Esa, atas berkat rahmat-Nya, modul pelatihan jabatan fungsional analis data ilmiah yang berjudul

“BASIS DATA” dapat diselesaikan tepat waktu. Modul ini digunakan dalam pelatihan jabatan fungsional yang dibina oleh BRIN yang diselenggarakan oleh Kedeputian SDMI - BRIN melalui Direktorat Pengembangan Kompetensi. Kami berharap modul ini dapat memberikan manfaat dan kontribusi dalam meningkatkan dan mengembangkan kompetensi jabatan fungsional yang dibina BRIN.

Jakarta, Desember 2022

Plt. Deputi Sumber Daya Manusia
Ilmu Pengetauan dan Teknologi
Badan Riset dan Inovasi Nasional

(*Tanda tangan*)

Edy Giri Rachman Putra, Ph.D.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

PENDAHULUAN

A. Deskripsi Singkat

Mata pelatihan ini menjelaskan tentang pengantar basis data, struktur data, standardisasi data, normalisasi data, serta data *warehouse*.

B. Alokasi Waktu

Jabatan Fungsional Analis Data Ilmiah dapat diselenggarakan dengan tiga metode. Setiap metode memiliki alokasi waktu yang berbeda. Berikut adalah alokasi waktu pembelajaran untuk mata pelatihan Basis Data.

Metode	*On* Kampus	Asinkronus	Sinkronus	Total
Klasikal	8 JP	-	-	8 JP
Bauran	5 JP	3 JP	3 JP	11 JP
Jarak Jauh	-	3 JP	6 JP	9 JP

C. Tujuan Pembelajaran

1. Hasil Belajar

Hasil belajar yang ingin dicapai melalui modul pelatihan ini adalah peserta pelatihan mampu melakukan pra-pemrosesan data ilmiah, sesuai dengan kaidah ilmiah.

2. Indikator Hasil Belajar

Ketercapaian hasil belajar dapat diukur melalui indikator hasil belajar, yaitu peserta pelatihan mampu:

a. menjelaskan konsep basis data: *design database*, *relation database*, SQL vs NoSQL, *standard query*, *transaction handling*, *permission and security*, *backup and restore database* dengan benar;

b. membedakan struktur data: 1 dimensi, 2 dimensi, 3 dimensi, dan lebih dari 3 dimensi dengan benar;

c. melakukan standardisasi data sesuai dengan kaidah atau standar yang berlaku;
d. melakukan normalisasi *database* dengan benar; dan
e. menjelaskan konsep data warehouse: pengertian data *warehouse*, lingkungan data *warehouse*, desain data dan desain arsitektur data *warehouse*, struktur data *warehouse*, kaitan data *warehouse* dengan *data mining* dengan benar.

D. Materi Pokok

Mata pelatihan ini terdiri dari 5 (lima) Materi Pokok, yaitu:

1. Pengantar basis data;
   a. Konsep Basis Data, *Relational Database*, *SQL* dan *NoSQL*
   b. *Query Standar* dan *transaction handling*
   c. Pengaturan *Permission* dan *Security*
   d. *Backup* dan *Restore Database*
2. Struktur data;
   a. Desain Database
   b. Pengertian Struktur Data
   c. Jenis Struktur Data
   d. Struktur Data pada Basis Data
3. Standardisasi data;
   a. Konsep Standardisasi Data
   b. Standar Data
   c. Teknik Standardisasi Data pada Basis Data
4. Normalisasi database;
   a. Konsep Normalisasi
   b. Normalisasi Bentuk Normal pertama (1NF)
   c. Konsep mengenai kunci-kuncipada tabel *database* – Super, Primary, Candidate, Foreign, Key, dll
   d. Normalisasi Bentuk Normal kedua (2NF)
   e. Normalisasi Bentuk Normal ketiga (3NF)
   f. Normalisasi Boyce-Codd Normal Form (BCNF)
   g. Normalisasi Bentuk Normal keempat (4NF)

5. *Data warehouse*.
    a. Definisi Gudang Data
    b. Komponen Gudang Data
    c. Perancangan Gudang Data
    d. Dimensional Modelling
    e. Operasi OLAP

MATERI POKOK 1:

PENGANTAR BASIS DATA

Indikator Hasil Belajar:
Peserta mampu:

1. Menjelaskan konsep basis data, *relational database*, SQL vs NoSQL, dengan benar;
2. Melakukan *query* standar, dan *transaction handling* pada *database* dengan benar;
3. Memahami *permission* dan *security* pada *database* dengan benar.
4. Melakukan *backup* dan *restore database*

A. Konsep Basis Data/*Database*, *Relational Database*, *Database* SQL vs NoSQL

Pada era Big Data saat ini, data dapat dikategorikan sebagai kekayaan baru bahkan lebih berharga bila dibandingkan dengan minyak. Data yang valid bisa menjadi salah satu kunci pembangunan. Data yang begitu besar harus disimpan dan dikelola dengan memperhatikan relasi keterkaitan dengan data yang lain, pengelolaan yang baik untuk keberlanjutan pemanfaatan, serta keamanan yang terjamin.

Data yang ada di dunia nyata sangat beragam bentuk atau strukturnya. Data yang berupa teks seperti dari buku, artikel atau yang diambil dari media sosial berisi tentang opini maupun interaksi dengan pengguna lainnya adalah contoh dari data tidak terstruktur. Sementara itu data seperti nama, tanggal, provinsi, berat badan, tinggi badan memiliki tipe data yang sama pada umumnya merupakan contoh dari data terstruktur. Data-data tersebut harus disimpan dan dikelola dengan baik serta disesuaikan karakteristik dari strukturnya, misalnya seperti relasi dengan data lain, tipe data (teks, numerik, atau kategorik), dan frekuensi perubahan data.

1. Basis Data/*Database*

Basis Data atau *database* adalah kumpulan dari data yang memiliki

relasi atau hubungan tanpa adanya pengulangan. Misalnya pada data profil pegawai, terdiri dari nama, NIP, unit kerja, nomor telepon dan data-data yang berhubungan dengan pegawai tersebut disimpan pada suatu aplikasi seperti Excel. Hal tersebut merupakan bentuk kecil dari sebuah *database*.

Database terdiri dari *relational database* dan *non relational database*. Beberapa *database* yang sering digunakan adalah MySQL, PostgreSQL, Microsoft SQL Server, Oracle Database, Apache Cassandra, MongoDB, CouchDB, dan CouchBase.

2. *Relational Database*

Relational Database adalah kumpulan item data yang memiliki relasi atau hubungan yang telah ditentukan sebelumnya. Berbagai item ini disusun menjadi satu set **tabel** dengan **kolom** dan **baris**. Tabel digunakan untuk menyimpan informasi tentang objek yang akan direpresentasikan dalam *database*. Tiap kolom pada tabel memuat jenis data tertentu. Baris pada tabel merepresentasikan kumpulan nilai terkait dari satu objek atau entitas. Tiap baris pada tabel dapat ditandai dengan pengidentifikasi unik yang disebut *primary key*, dan baris di antara beberapa tabel dapat dibuat saling terkait menggunakan *foreign key*. Beberapa contoh *relational database* diantaranya MySQL, PostgreSQL, Microsoft SQL Server, dan Oracle Database.

3. *Database SQL* dan *NoSQL*

SQL (*Structured Query Language*) biasanya berbentuk *Relational Database* Management System (RDBMS). RDBMS adalah program yang melayani sistem basis data yang entitas utamanya terdiri dari tabel-tabel yang mempunyai relasi dari satu tabel ke tabel yang lain. Sementara itu, NoSQL biasanya berbentuk *non relational database* atau *non distributed database*. SQL memiliki schema yang telah ditetapkan di awal dan bersifat statis, sedangkan NoSQL memiliki schema yang dinamis yang dapat berubah-ubah sesuai dengan kondisi data. Beberapa contoh database NoSQL diantaranya Apache Cassandra,

MongoDB, CouchDB, dan CouchBase.

B. Query standar dan *transaction handling*

Query adalah sekumpulan baris perintah yang diproses untuk mendapatkan informasi yang berasal dari database. Istilah lain yang sering digunakan untuk query adalah query database. Query database ini digunakan untuk memudahkan pengelolaan data yang ada di database. Dengan menggunakan query yang tepat, data dan informasi yang diperlukan akan tampil dan bisa digunakan. Orang yang bertanggung jawab untuk menangani database disebut dengan Database Administrator. Query adalah istilah yang juga dikenal dengan query language atau bahasa query. Bahasa query yang paling populer di kalangan Database Administrator adalah SQL. Bahasa query ini menjadi standar manajemen database.

Sebelum pembahasan terkait query standar yang biasa digunakan pada proses pengelolaan database, perlu diketahui juga tipe-tipe data yang biasanya digunakan untuk mendefinisikan nilai suatu atribut, antara lain :

a. numeric

Tipe data numerik termasuk juga bilangan bulat dengan berbagai ukuran (INT atau SMALLINT) dan bilangan desimal (FLOAT).

b. character-string

Tipe data string atau huruf ditandai dengan tanda petik ketika melakukan insert data dan memiliki aturan case sensitive atau besar kecilnya huruf dapat mempengaruhi nilai suatu atribut. Jenis untuk tipe data character-string diantaranya adalah CHAR(n) dan VARCHAR(n), dimana n adalah panjang maksimum dari jumlah total karakter.

c. boolean

Tipe data boolean memiliki nilai TRUE atau FALSE. Pada SQL, oleh karena dimungkinkan adanya data bernilai NULL, sehingga kemungkinan nilai logika memiliki nilai lainnya yaitu UNKNOWN.

d. date

Tipe data date digunakan untuk data-data tanggal yang terdiri dari tanggal (Date), bulan (Month), dan tahun (Year) dengan format YYYY-MM-DD. selain itu juga berisi waktu yaitu jam (Hours), menit

(Minuts), dan detik (Seconds) dengan format HH:MM:SS. Nilai dari keseluruhan bagian tanggal sampai dengan detik sudah ditentukan di SQL, misalnya untuk bulan hanya bernilai 1-12, tanggal bernilai 1-31, serta detik 1-60.

Setelah mengetahui tipe-tipe dari data, berikutnya harus diketahui beberapa query standar yang digunakan dalam pengelolaan database antara lain :

1. DDL (Data Definition Language)

DDL (Data Definition Language) yaitu sebuah bahasa pemrograman komputer yang digunakan untuk membuat dan memodifikasi struktur sebuah objek database di database terutama dalam bentuk skema. DDL tidak bisa lepas dari Structure Query Language (SQL) untuk menampilkan tabel, kolom, data, dan batasan. Perintah yang termasuk dalam DDL antara lain :

a. CREATE

create table digunakan untuk membuat tabel beserta atributnya. Perintah create tidak terbatas untuk membuat tabel saja, akan tetapi di SQL versi tertentu, dapat juga digunakan untuk membuat view, index, database, function, procedure, atau trigger.

b. ALTER

alter table digunakan untuk menambah kolom baru, menghapus kolom, atau mengubah kolom yang sudah ada pada sebuah tabel.

c. DROP

drop table digunakan untuk menghapus tabel yang sudah ada. Seperti perintah create drop juga dapat digunakan untuk menghapus

Misalnya kita akan membuat tiga tabel customer, item, dan order. Customer adalah tabel yang berisi karakteristik dari pelanggan, item adalah tabel yang berisi karakteristik dari sebuah produk, dan order adalah karakteristik dari transaksi pelanggan membeli atau memesan sebuah produk.

```
create table customer
(cust_num numeric,
cust_name char(20),
address varchar(256),
credit_level numeric,
check (credit_level >= 1000),
primary key (cust_num));
```

Query di atas untuk membuat tabel customer dengan atribut cust_num sebagai primary key atau atribut kunci. Primary key disini harus bernilai unik sehingga menjadi pembeda dalam tabel customer. Pendefinisian primary key pada cust_num juga dapat dilakukan pada jenis data menjadi ‘numeric not null unique’.

```
create table item
(item_num numeric,
item_name char(20),
price numeric,
weight numeric,
primary key (item_num));


create table order
(ord_num char(15),
cust_num numeric not null,
item_num numeric not null,
quantity numeric,
total_cost numeric,
primary key (ord_num),
foreign key (cust_num) references
customer
on delete no action on update
cascade,
foreign key (item_num) references
item
on delete no action on update
cascade);
```

Query pendukung yang sering digunakan dalam pendefinisian atribut menggunakan perintah create table adalah :

1) *not null* : untuk mendefinisikan nilai pada atribut tersebut tidak boleh kosong
2) *unique* : untuk mendefinisikan atribut tersebut adalah kunci dalam suatu tabel dimana nilainya harus unik dari setiap baris atau record
3) *primary key* : satu atau beberapa atribut yang digunakan untuk mengidentifikasi keunikan suatu tabel atau atribut.
4) *foreign key* : untuk menghubungkan atau merelasikan suatu tabel dengan tabel lainnya. atribut yang menjadi primary key bisa sekaligus menjadi foreign key.

Setelah memiliki ketiga tabel di atas, dapat juga dilakukan

modifikasi atau perubahan dari atribut yang telah dibuat menggunakan alter table, serta dapat menghapus menggunakan drop table.

```
alter table customer
modify (cust_name varchar(256)); --(1)
alter table customer
add column cust_credit_limit numeric; --(2)
alter table customer
drop column credit_level; --(3)
drop table customer; --(4)
```

Query di atas (1) untuk mengubah karakter atribut cust_name dari char(20) menjadi varchar(256). Selanjutnya (2) menambah atribut baru yaitu cust_credit_limit yang bertipe numerik. Query berikutnya (3) untuk menghapus atribut credit_level, dan yang terakhir (4) adalah untuk menghapus tabel *customer*. Pada perintah di atas juga dapat ditambahkan kriteria tambahan.

2. DML (Data Manipulation Language)

DML adalah kelompok perintah yang berfungsi untuk memanipulasi data dalam basis data, misalnya untuk pengambilan, penyisipan, pengubahan dan penghapusan data. Perintah yang termasuk dalah kategori DML adalah diantaranya sebagai berikut :

a. SELECT

select merupakan perintah dasar yang sering digunakan dalam SQL. Perintah ini digunakan untuk mengekstrak data atau menampilkan data dari tabel dengan kriteria tertentu. Format penggunaan select adalah :

```
select <daftar atribut>
from <daftar tabel>
where <kondisi>
```

dimana <daftar atribut> adalah atribut-atribut yang ingin ditampilkan, <daftar tabel> adalah tabel-tabel yang dibutuhkan untuk menampilkan atribut, dan <kondisi> adalah ekspresi kondisi yang dibutuhkan dalam menampilkan data.

Berikut adalah contoh penggunaan perintah *select* :

```
select *
from customer; --(1)
select cust_name, cust_num,
credit_level
from customer
where address = 'Enterprise'
```

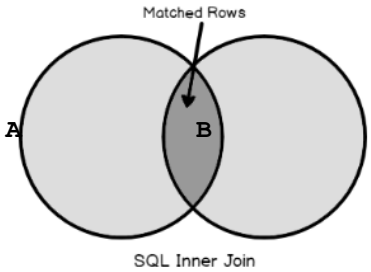
Tanda * pada query di atas (1) menunjukkan pemilihan semua atribut yang ada dalam suatu tabel. Sementara itu query (2) adalah perintah untuk menampilkan kolom atau atribut cust_name, cust_num, credit_level dari tabel customer dengan kriteria address bernilai Enterprise, credit_level bernilai lebih dari 7, dan data yang ditampilkan diurutkan secara ascending berdasarkan cust_name.

Pada penggunaan query ini, juga bisa dilakukan agregasi seperti penjumlahan, rata-rata, menghitung baris, menampilkan nilai maksimum, nilai minimum.

Perintah yang tujuan penggunaannya untuk untuk penggabungan tabel melalui kolom atau key tertentu dimana memiliki nilai terkait untuk mendapatkan satu set data dengan informasi lengkap adalah *subquery* dan *join*. Akan tetapi yang paling sering digunakan adalah perintah *join*. Pada perintah join, ada beberapa macam penggunaan join seperti :

1) *inner join*

inner join digunakan untuk membandingkan baris di setiap tabel untuk dicek apakah nilai sama atau tidak. Jika nilai kedua tabel sama, maka akan terbentuk tabel baru yang hanya menampilkan baris yang sama dari kedua tabel tersebut. Berikut adalah ilustrasi proses inner join:

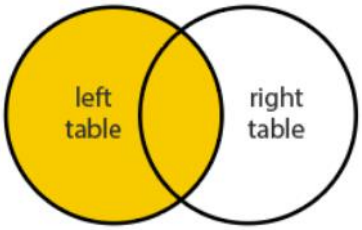


Data yang akan diambil pada proses inner join adalah irisan dari tabel A dan tabel B. Contoh penggunaannya adalah sebagai berikut :

```
select *
from A
inner join B
on A.nip = B.nip
```

2) *left join*

left join menghasilkan nilai berdasarkan tabel kiri (*left table*) dan nilai yang sama di tabel kanan (*right table*). Jika tabel kanan tidak sama nilainya dengan tabel kiri, maka akan diisi nilai NULL pada tabel kanan.

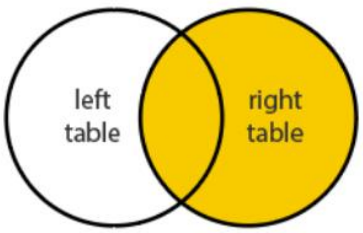


Contoh penggunaannya adalah sebagai berikut :

```
select *
from A
left join B
on A.nip = B.nip
```

3) *right join*

Konsep *right join* hampir sama seperti *left join* hanya yang menjadi master adalah tabel kanan (*right table*). Jika tabel kiri tidak sama nilainya dengan tabel kanan, maka akan diisi nilai NULL pada tabel kiri.



Contoh penggunaannya adalah sebagai berikut :

```
select *
from A
right join B
on A.nip = B.nip
```

b. UPDATE

Perintah *update* data merupakan salah satu perintah SQL yang digunakan untuk mengubah data sesuai dengan kondisi yang diinginkan. Berikut adalah contoh penggunaan perintah *update* :

```
update customer
set credit_level = 7
where credit_level = 6;
```

Query di atas digunakan untuk mengubah tabel customer yang memiliki credit_levelnya bernilai 6 menjadi bernilai 7.

c. INSERT

Perintah *insert* digunakan untuk menambahkan satu atau lebih nilai ke tabel tunggal mana pun dalam database relasional. Berikut adalah contoh penggunaan perintah *insert* :

```
insert into customer
values
(010,'klingon','rogueShip',4);
```

Query di atas digunakan untuk menambahkan nilai 010, ‘klingon’, ‘rogueShip’, dan 4 pada atribut-atribut yang ada di tabel customer.

d. DELETE

Perintah *delete* digunakan untuk menghapus satu atau beberapa baris dari tabel. Subset atau kumpulan baris dapat didefinisikan untuk dihapus menggunakan suatu kondisi, jika tidak, semua record akan dihapus. Berikut adalah contoh penggunaan perintah *delete* :

```
delete from customer
where credit_level < 2;
```

Query tersebut digunakan untuk menghapus data-data yang memiliki nilai credit_level kurang dari 2 pada tabel customer.

3. DCL (Data Control Language)

Data Control Language (DCL) adalah salah satu dari kelompok perintah SQL yang digunakan untuk melakukan kontrol terhadap *privilege* atau hak akses khusus untuk berinteraksi dengan database. Hak akses khusus ini diperlukan sebagai prasyarat bagi setiap user database untuk melakukan berbagai aksi di database, seperti: membuat object, menghapus object, mengubah object, menampilkan hasil query, dan seterusnya.

Sederhananya, setiap pengguna database hanya dapat melakukan aksi-aksi yang sudah diberikan oleh user dengan kontrol tertinggi di dalam database tersebut. Misalnya, pengguna A hanya diberikan akses untuk membuat dan menampilkan sesuatu di database. Maka, pengguna tersebut hanya dapat melakukan perintah tersebut saja, tidak bisa melakukan ubah data, hapus data, dan seterusnya. Hal ini menjadi penting untuk diketahui karena di dalam dunia kerja dengan banyak pengguna database, manajemen pembagian hak akses seperti ini sangat krusial dan berpengaruh terhadap integritas dan keamanan data.

Perintah dari DCL diantaranya adalah GRANT dan REVOKE. Selain itu dalam terminologi database relasi, transaksi akan menghasilkan COMMIT atau ROLLBACK. Tiap transaksi diperlakukan secara koheren dan dapat diandalkan terlepas dari transaksi lainnya. Berikut merupakan penjelasan dari beberapa perintah DCL :

a. GRANT

Perintah *grant* biasanya digunakan ketika admin database ingin memberikan hak akses ke user lainnya. Seperti yang dijelaskan sebelumnya, pemberian hak akses ini dapat dibatasi atau

diatur. Dalam hal ini admin pun dapat memberikan akses mengenai perintah dalam DML di pembahasan sebelumnya.

b. REVOKE

Perintah *revoke* sering digunakan untuk mencabut maupun menghapus hak akses seorang pengguna yang awalnya diberikan akses oleh admin database melalui perintah *grant* sebelumnya.

c. COMMIT

commit dalam SQL mengakhiri transaksi dalam sistem manajemen basis data relasional dan membuat semua perubahan terlihat oleh pengguna lain.

d. ROLLBACK

rollback merupakan perintah untuk mengembalikan database ke keadaan sebelumnya. Rollback penting untuk integritas database, karena ini berarti database dapat dikembalikan ke salinan yang bersih bahkan setelah operasi yang salah dilakukan.

C. Pengaturan *Permission* Dan *Security*

Salah satu cara yang paling umum dalam penggunaan *Database Security* adalah dengan akses kontrol. Akses kontrol mendasar pada pemberian atau pengambilan *privilege* (akses) ke dalam suatu database system. *Privilege* diberikan kepada *user* tanpa harus memberi akses penuh ke dalam suatu *database*. Akses tertentu hanya boleh diberikan jika suatu pekerja tidak mampu melaksanakan pekerjaan atau tugasnya tanpa akses tersebut. Pemberian *privilege* (akses) secara berlebihan dapat menyebabkan terjadinya kompromi data. User yang menciptakan objek dari database biasanya mendapatkan akses penuh ke seluruh fungsionalitas database. DBMS dapat mencatat setiap level akses yang tersedia ke user-user yang ada.

Database Security merujuk pada kegiatan atau upaya untuk melindungi database dari ancaman luar. *Database Security* kebanyakan

menggunakan *security control* untuk mengontrol akses ke dalam suatu database. Pengelolaan database seharusnya mematuhi ACID atau *Atomic*, *Consistent*, *Isolated*, dan *Durable*.

Atomisitas/*Atomic* mensyaratkan bahwa transaksi secara keseluruhan berhasil terlaksana atau jika sebagian transaksi gagal, maka seluruh transaksi menjadi tidak valid. Konsistensi/*Consistent* mengamanatkan data yang ditulis ke database sebagai bagian dari transaksi harus mematuhi semua aturan yang ditetapkan, dan pembatasan termasuk batasan, jenjang, dan pemicu. Isolasi/*Isolated* sangat vital untuk mencapai kontrol konkurensi dan memastikan tiap transaksi independen terhadap dirinya sendiri. Durabilitas/*Durable* mensyaratkan bahwa semua perubahan yang dilakukan terhadap database bersifat permanen setelah transaksi berhasil diselesaikan.

Pengaturan *permission* dan pengelolaan user untuk menjaga *security* atau keamanan umumnya dilakukan menggunakan DCL oleh DBA (*Database Administrator*). Selain itu, untuk menjaga keamanan dan integritas data, juga diperlukan adanya proses *backup database* secara rutin, sehingga ketika terjadi *crash* karena suatu hal, data yang hilang dapat dilakukan *restore database*.

D*. Backup* Dan *Restore Database*

Backup database adalah menyalin data dalam database ke file eksternal (isi data dalam bentuk query sql). Restore adalah menyalin data dari file eksternal (dengan

mengeksekusi query sql) ke dalam database. Hal ini dilakukan untuk mengantisipasi terjadinya crash pada storage ataupun pada database. Proses backup dan restore dari setiap jenis database memiliki query serta tahapan yang berbeda-beda. Pada backup, yang dapat dilakukan adalah backup keseluruhan database, atau hanya beberapa tabel tertentu saja.

E. Rangkuman

Data merupakan kekayaan baru yang lebih berharga dibandingkan dengan minyak. Data yang valid dapat menjadi kunci pembangunan. Oleh karena itu, dibutuhkan pengelolaan data tersebut untuk dapat menjaga keamanan dan integritas data yang ada. Basis data/*Database* merupakan kumpulan dari data yang memiliki relasi atau hubungan tanpa adanya pengulangan. Database terdiri dari database relational dan database non relational. Database relational biasanya digunakan untuk data-data terstruktur, sedangkan database non relational biasanya digunakan untuk data-data yang tidak terstruktur.

Database relational adalah kumpulan item data yang memiliki relasi atau hubungan yang telah ditentukan sebelumnya. Berbagai item ini disusun menjadi satu set tabel dengan kolom dan baris. Perbedaan signifikan antara database relational atau biasa dengan SQL sebagai bentuk RDBMS dengan NoSQL adalah pada pembuatan schema. Pendefinisian schema yang ada di SQL sudah ditentukan di awal pembuatan desain database, sementara schema untuk NoSQL dapat berubah-ubah atau dinamis.

Pegelolaan database dengan mudah dapat dilakukan dengan query yaitu sekumpulan baris perintah yang diproses untuk mendapatkan informasi yang berasal dari database. Query sederhana yang perlu diketahui ada tiga jenis yaitu DDL (Data Definition Language), DML (Data Manipulation Language), dan DCL (Data Control Language). Masing-masing jenis query tersebut memiliki fungsi dan perintah yang berbeda-beda. Namun, perlu diketahui juga tipe-tipe data untuk mendefinisikan atribut-atribut dalam suatu tabel.

Pengelolaan database berfokus untuk menjaga keamanan serta integritas dari data, sehingga perlu adanya pembatasan user dalam melakukan DDL maupun DML dalam database dimana harus mematuhi ACID atau *Atomic*, *Consistent*, *Isolated*, dan *Durable.* Selain itu, untuk menjaga keamanan dan integritas data, juga diperlukan adanya proses *backup database* secara rutin, sehingga ketika terjadi *crash* karena suatu hal, data yang hilang dapat dilakukan *restore database*.

F. Evaluasi

Untuk menguji pemahaman Anda terkait materi Pengantar Basis Data, silahkan kerjakan latihan di bawah ini.

1. Buatlah Query DDL, DML, dan DCL ! (dengan data yang dapat diunduh pada link berikut https://github.com/efkhusnulh/introbasdat atau data yang anda punya)

MATERI POKOK 2:

STRUKTUR DATA

Indikator Hasil Belajar:
Peserta mampu:
membedakan struktur data: 1 dimensi, 2 dimensi, 3 dimensi, dan lebih dari 3 dimensi, dengan benar.

A. Desain Database

Desain database merupakan tahapan penting yang harus dilakukan dalam pengembangan *database* dengan memperhatikan kebutuhan *user* dan kebutuhan *hardware* (storage, jaringan, dll). Tahapan melakukan desain database yang perlu diperhatikan adalah :

1. Analisis Kebutuhan

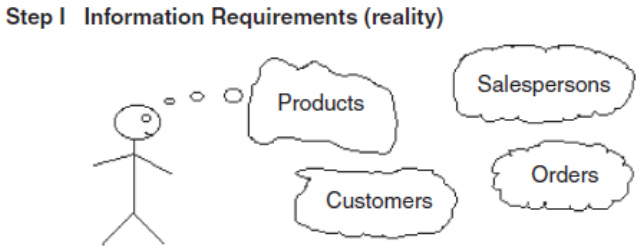
Kebutuhan database diperoleh dari berdiskusi atau mewawancari produsen data dengan *user*/pengguna data serta menggunakan informasi untuk menghasilkan spesifikasi kebutuhan resmi. Spesifikasi yang dimaksud adalah data yang diperlukan untuk diproses, relasi antar data, dan platform perangkat lunak untuk database.

Tujuan dari tahap analisis kebutuhan ini adalah :

a. untuk menjabarkan kebutuhan informasi dan data dasar yang menjadi penyusunnya,
b. untuk mendapatkan gambaran hubungan atau keterkaitan antar data yang dibutuhkan,
c. untuk menentukan jenis transaksi yang akan dilakukan pada database dan interaksi antara transaksi dengan data,
d. untuk mengidentifikasi kinerja, integritas, dan keamanan serta batasan-batasan adiministrasi yang harus diterapkan,
e. untuk menentukan spesifikasi teknologi, hardware, software, bahasa pemrograman, kebijakan, SOP, dan antarmuka,
f. untuk mendokumentasikan seluruh proses kebutuhan user. Data

dasar biasanya didokumentasikan berupa kamus data.

Ilustrasi kebutuhan data dari user dapat dilihat pada Gambar 1. contoh yang digunakan adalah user merupakan seorang manajer ingin mengetahui kinerja bagian pemasaran, produk yang paling banyak dijual, dan kapan transaksi yang paling banyak.

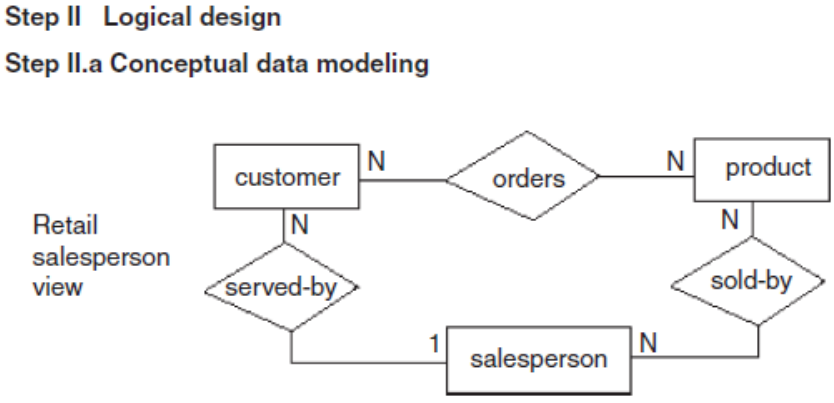


Gambar 1. Ilustrasi Kebutuhan *User*

2. Logical Design

a. Pemodelan Konsep Data/Conceptual Data Modelling

Pada tahapan ini, desain yang dibuat masih berbentuk konsep secara keseluruhan dan umum. Kebutuhan dari user dituangkan dalam sebuah diagram untuk mempermudah identifikasi. Tahap ini fokus pada model data yang akan digunakan tanpa memikirkan logika-logika penyimpanan *database* dan pertimbangan fisik *database* tersebut. Output dari tahapan ini biasanya ERD (Entity Relationship Diagram). ERD (Entity Relationship Diagram) digunakan untuk memodelkan struktur dan hubungan antar data baik hubungan yang sederhana sampai yang relatif kompleks.



Gambar 2. Ilustrasi Pemodelan Konsep Data menggunakan ERD

Komponen umum penyusun ERD antara lain **entitas**, **atribut**, dan **relasi**. Pada Gambar 2 dapat dilihat bahwa **entitas** merupakan aktor atau objek yang berinteraksi dalam transaksi data seperti *customer*, *product*, dan *salesperson*. Komponen selanjutnya adalah relasi. **Relasi** adalah hubungan antar entitas untuk menunjukkan adanya koneksi di antara sejumlah entitas yang berasal dari himpunan entitas berbeda. Misalnya pada Gambar 2, seorang customers melakukan orders satu atau beberapa product. Oleh karena itu orders merupakan relasi dari seorang customer dan sebuah product.

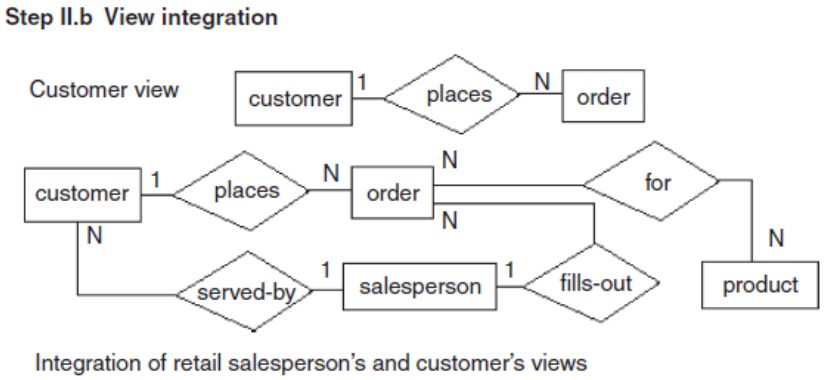
Pada relasi, perlu diidentifikasi juga hubungan antar entitas tersebut apakah *one to one (1-1), one to many (1-N), many to one (N-1), atau many to many(N-N)*. Gambar 2 terlihat hubungan antara customer dengan product adalah many to many (N-N). Jika diinterpretasikan, satu customer dapat membeli beberapa produk dan satu produk dapat dibeli beberapa customer.

Sementara itu, setiap entitas memiliki **atribut** untuk mendeskripsikan karakteristik dari suatu entitas misalnya nama customer, nomor telepon customer, alamat customer, dll. Salah satu atribut yang penting adalah atribut kunci atau attribute key. Atribut kunci terdiri dari primary key dan foreign key. Primary key merupakan atribut unik yang dapat menjelaskan entitas, misalnya NIK, NIP. Foreign key merupakan atribut unik yang digunakan sebagai penghubung antara atribut satu dengan lainnya.

Output lain yang dapat digunakan dalam pemodelan konsep data adalah UML (Unified Modelling Language). UML adalah suatu metode dalam pemodelan secara visual yang digunakan sebagai sarana perancangan sistem berorientasi objek. Beberapa bentuk UML yang sering digunakan antara lain : Use Case Diagram, Activity Diagram, Sequence Diagram, dan Class Diagram.

b. Pengintegrasian Perspektif /*View Integration*

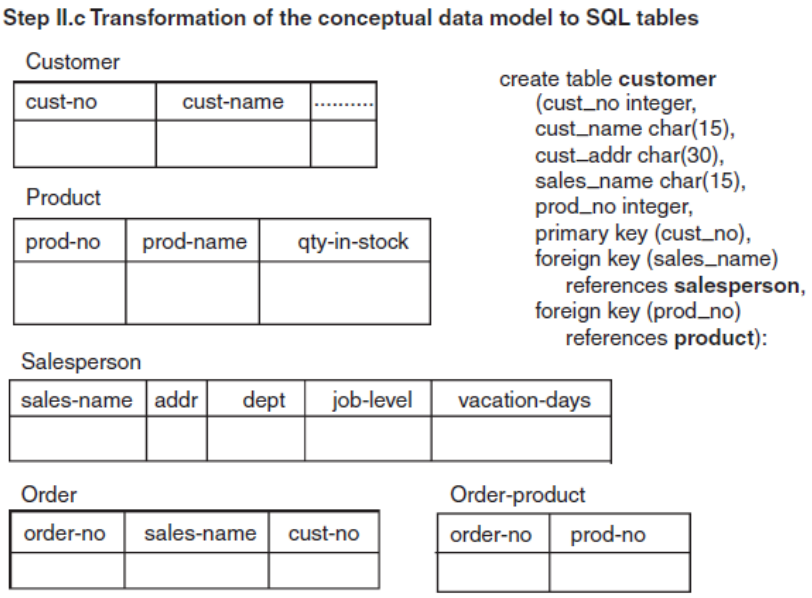
Biasanya, bila desainnya besar dan lebih dari satu orang terlibat dalam analisis kebutuhan, beberapa tampilan data dan hubungan terjadi, mengakibatkan inkonsistensi karena terjadi keragaman dalam taksonomi, konteks, atau persepsi. Untuk menghilangkan redundansi dan inkonsistensi dari model, pandangan ini harus menjadi "rasionalisasi" dan dikonsolidasikan menjadi satu perspektif.



Gambar 3. Ilustrasi Perspektif Integrasi

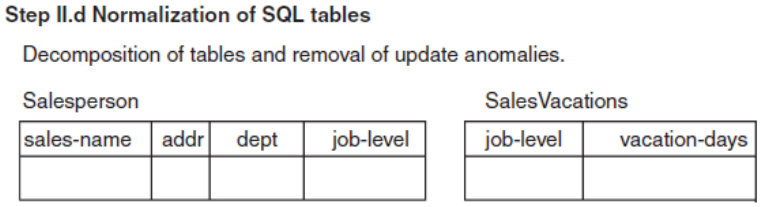
Pada Gambar 3, ada dua kemungkinan tampilan database product/customer digabungkan menjadi satu tampilan global berdasarkan data umum untuk customer dan order. Integrasi tampilan juga penting ketika aplikasi harus diintegrasikan, dan masing-masing mungkin ditulis dengan tampilan databasenya sendiri.

c. Transformasi model konsep data ke tabel SQL/*Transformation of the conceptual data model to SQL tables*



Gambar 4. Transformasi model konsep data menjadi tabel SQL

d. Normalisasi tabel/*Normalization of tables*



Gambar 5. Ilustrasi Normalisasi table

Pada tahapan ini, dilakukan validasi model data yang telah dibuat, apakah sesuai dengan logika-logika database dan sudah sesuai secara struktural. Pada tahapan ini juga bisa dilakukan normaliasi untuk melihat kesesuaian model data yang dalam mendukung transaksi yang ada.

3. Phisycal Design

Tahap ini lebih fokus pada struktur *database*. Penyimpanan data serta hubungan atau keterkaitan data pada *database* akan dibuat pada tahap ini. Selain itu, tahap ini juga memperhatikan integritas data yang akan disimpan di database tersebut. Integritas data pada transaksi *database* dapat dipastikan dengan mematuhi ACID atau *Atomic, Consistent, Isolated, dan Durable*.

Atomisitas/*Atomic* mensyaratkan bahwa transaksi secara keseluruhan berhasil terlaksana atau jika sebagian transaksi gagal, maka seluruh transaksi menjadi tidak valid. Konsistensi/*Consistent* mengamanatkan data yang ditulis ke database sebagai bagian dari transaksi harus mematuhi semua aturan yang ditetapkan, dan pembatasan termasuk batasan, jenjang, dan pemicu. Isolasi/*Isolated* sangat vital untuk mencapai kontrol konkurensi dan memastikan tiap transaksi independen terhadap dirinya sendiri. Durabilitas/*Durable* mensyaratkan bahwa semua perubahan yang dilakukan terhadap database bersifat permanen setelah transaksi berhasil diselesaikan.

4. Database implementation, monitoring, and modification

Tahap ini lebih fokus pada struktur *database*. Penyimpanan data serta hubungan atau keterkaitan data pada *database* akan dibuat pada tahap ini. Selain itu, tahap ini juga memperhatikan integritas data yang akan disimpan di database tersebut. Integritas data pada transaksi *database* dapat dipastikan dengan mematuhi ACID atau *Atomic, Consistent, Isolated, dan Durable*.

B. Pengertian Struktur Data

Pada umumnya, struktur data digunakan untuk mengimplementasikan bentuk fisik dari sebuah tipe data. Struktur data adalah bagian yang paling krusial dalam mendesain sebuah *software* yang efisien. Struktur data juga menjadi dasar dari desain algoritma yang nanti akan digunakan dalam jalannya program komputer.

Bahasa pemrograman awal seperti *Fortran, C, C*++ mengizinkan pengembang untuk mendefinisikan struktur data sesuai dengan keinginan. Saat ini, banyak bahasa pemrograman menyertakan kumpulan ekstensi struktur data bawaan untuk mengatur kode dan informasi. Contohnya seperti pada bahasa pemrograman Python yaitu *List* dan *Dictionary*, Javascript yaitu *Array* dan *Object* yang umum digunakan untuk menyimpan informasi dan mengaksesnya.

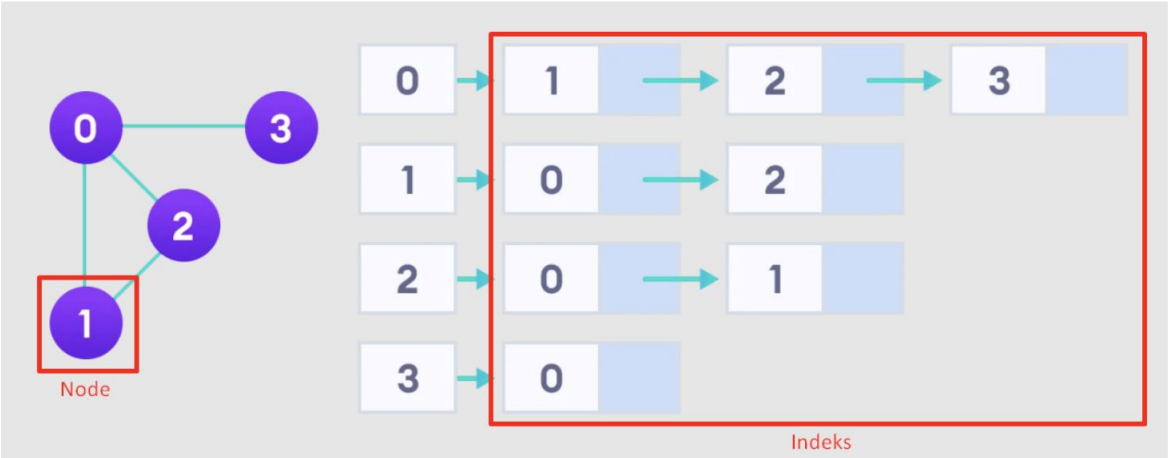
Software Engineer menggunakan algoritma yang berhubungan erat

dengan struktur data seperti *List, Queue,* dan *Mapping* dari satu value ke value yang lainnya. Pendekatan ini dapat diterapkan ke banyak aplikasi, termasuk mengatur *record* dalam *Relational Database* dan membuat *index* dari *record* yang digunakan sebagai struktur data yang disebut *binary tree.*

Berikut adalah beberapa contoh bagaimana struktur data digunakan :

- Penyimpanan Data (*Storing Data*)
- Mengatur Sumber dan Pelayanan (*Managing Resource and Service*)
- Pertukaran Data (*Data Exchange*)
- Pengurutan (*Ordering and Sorting*)
- Pengindeksan (*Indexing)*
- Pencarian (*Searching*)
- Skalabilitas (*Scalability*)

Struktur data itu sendiri adalah cara menyimpan dan mengatur data secara terstruktur pada sistem komputer atau database sehingga lebih mudah diakses. Secara teknis, data dalam bentuk angka, huruf, simbol, dan lainnya ini diletakkan dalam kolom-kolom dan susunan tertentu.

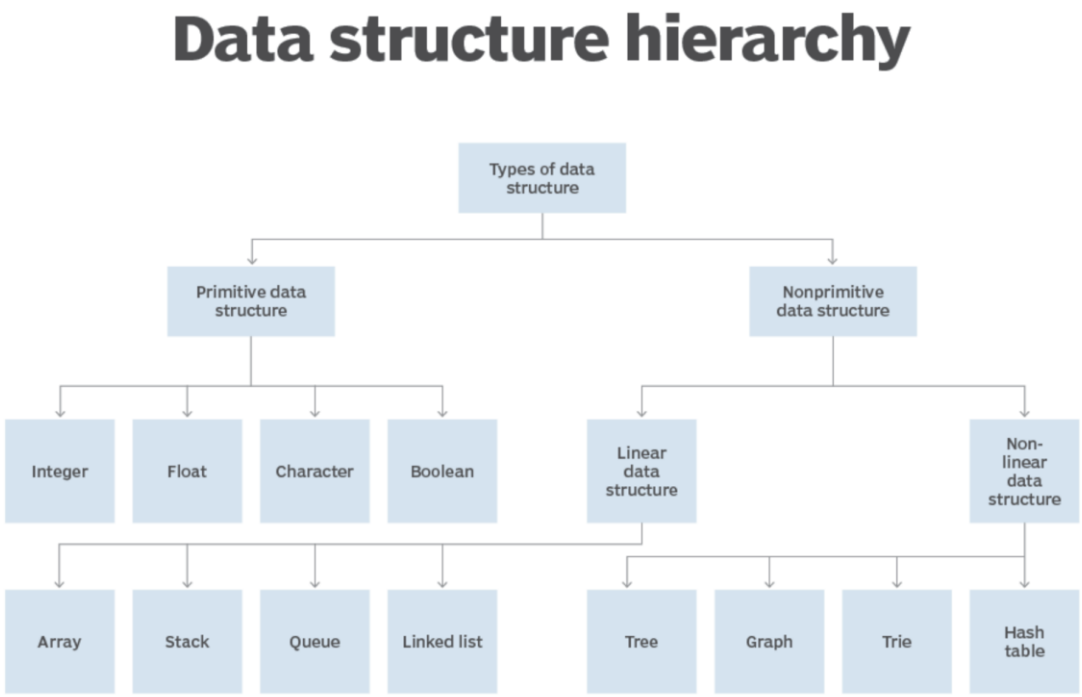


Gambar 6. Ilustrasi dari Struktur Data

Dalam menyusun data, terdapat beberapa istilah yang perlu dipahami, yaitu *node* dan indeks. Berikut adalah penjelasan tentang kedua istilah tersebut.

- *Node*, yaitu elemen yang terdapat dalam struktur data. Setiap *node* berisi pointer ke *node* selanjutnya.
- Indeks, yaitu objek dalam sistem database yang bisa mempercepat proses pencarian data.

Untuk mengenal lebih jauh tentang struktur data, berikut adalah Hirarki dari stuktur data.



Gambar 7. Hirarki Struktur data menunjukkan bagaimana Tipe Data dan Struktur Data saling berhubungan

1. Karakteristik Data

Struktur data sering diklasifikasikan berdasarkan karakteristiknya. Berikut adalah tiga karakteristiknya :

a. Linear dan Non-Linear

Karakteristik ini menggambarkan tentang item data yang disusun berdasarkan urutan, seperti larik atau dalam urutan yang tidak berurutan seperti grafik.

b. Homogen dan Heterogen

Karakteristik ini menggambarkan apakah semua item data dalam repositori tertentu memiliki tipe yang sama. Salah satu

contohnya adalah kumpulan elemen dalam larik atau berbagai tipe seperti abstrak yang didefinisikan sebagai struktur pada bahasa pemrograman C atau spesifikasi *class* dalam bahasa pemrograman Java.

c. Statis dan Dinamis

Karakteristik ini menjelaskan bagaimana struktur data dikompilasi. Struktur Data statis memiliki ukuran, struktur dan lokasi memori yang tetap pada waktu kompilasi. Struktur Data dinamis memiliki ukuran, struktur, dan lokasi memori yang dapat menyusut atau meluas, tergantu penggunaan.

2. Tipe Data

Jika struktur data adalah blok penyusun algoritma dan program komputer, tipe data primitif atau tipe data dasar adalah blok penyusun struktur data. Tipe data dasar atau biasa disebut tipe data primitif meliputi berikut ini :

- **Boolean**

  Menyimpan *Logical value* yaitu *true* dan *false.*

- **Integer**

  Menyimpan rentang bilangan bulat matematika atau menghitung angka. Bilangan bulat berukuran berbeda memiliki rentang nilai yang berbeda. Misalnya, bilangan bulat 8-bit memiliki nilai dari 128 hingga 127 dan bilangan bulat 32-bit menyimpan dari 0 sampai 4.294.967.295.

- **Floating-point numbers**

  Menyimpan bilangan pecahan / desimal.

- **Character**

  Tipe data yang dipakai untuk mewakili simbol dari sebuah karakter terdiri atas tipe data char

- **Pointers**

  Merupakan nilai referensi yang merujuk ke nilai lain.

- **String**

  Merupakan larik karakter yang diikuti dengan kode berhenti biasanya kalo dalam nilai “0” atau dikelola menggunakan panjang *filed* yang

merupakan nilai bilangan bulat.

C. Jenis Struktur Data

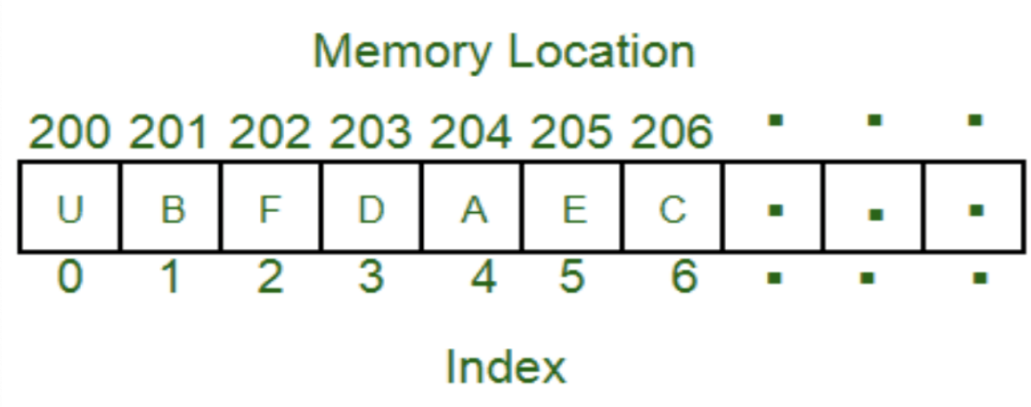
Jenis struktur data yang digunakan dalam situasi tertentu ditentukan oleh jenis operasi yang akan dibutuhkan atau jenis algoritma yang akan diterapkan. Berbagai jenis struktur data meliputi sebagai berikut :

1. *Array*

Array atau dalam bahasa indonesia disebut larik, yaitu menyimpan kumpulan item di lokasi memori yang berdampingan. Item yang bertipe sama disimpan bersama sehingga posisi setiap elemen dapat dihitung atau diambil dengan mudah oleh sebuah indeks. Panjang larik biasa tetap atau fleksibel.

Kapasitas elemen yang dapat dialokasikan pada tipe *array* bersifat statis. Jika ingin menyisipkan elemen baru ke *array*, maka harus membuat *array* baru dengan ukuran yang lebih besar. Sebaliknya, jika ingin menghapus elemen tertentu, harus membuat *array* dengan ukuran kecil.

Selain itu, *array* juga memungkinkan menyimpan beberapa data dengan jenis yang sama dalam satu nama. Tipe *array* biasa digunakan untuk membangun struktur data, seperti vektor dan matriks.



Gambar 8. Ilustrasi penyimpanan data pada Array di Memory

Keunggulan tipe *array* :

- Bisa digunakan sebagai implementasi tipe lainnya, seperti *queue*

dan *stack.*

- Proses pencarian data biasa dilakukan lebih cepat.

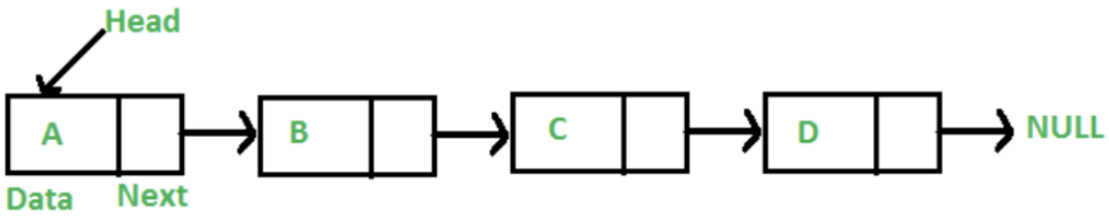
Kekurangan tipe *array* :

- Penambahan dan pengurangan data membutuhkan waktu yang elbih lama karena tipe *array* menampung data secara berurutan.

2. *Linked List*

Linked list adalah struktur data yang terdiri dari urutan data linier yang dihubungkan satu sama lainnya. Saat menggunakan tipe *linked list,* harus mengakses data secara manual. Hal ini karena tidak bisa mencari data dengan sistem acak.

Tipe *linked list* berbagi menjadi tiga jenis, yaitu *singly linked list, double linked list , circular linked list.* Ketiganya dapat dibedakan dari proses traversal atau proses kunjungan ke stiap *node* dalam satu wakti.



Gambar 9. Ilustrasi Linked List

Keunggulan tipe *linked list* :

- Ukuran lebih dinamis
- Alokasi penggunaan memori bisa disesuaikan dengan kebutuhan.
- Penambahan atau pengurangan data lebih cepat.

Kekurangan tipe *linked list* :

- Menguras memori yang lebih besar.
- Tidak bisa kembali ke *node* sebelumnya, kecuali pada jenis *doubly linked list.*
- Proses *traversal* lebih panjang karena tidak langsung mengakses data dengan indeks.

3. *Queue*

Queue adalah tipe struktur data linier yang mengikuti urutan tertentu, yaitu FIFO (*First In First Out*). Jadi, data yang masuk pertama kali adalah data yang pertama kali diambil. Analogi sederhana yang menggambarkan tipe ini adalah orang yang sedang mengantri, siapa yang data pertama, itulah yang dilayani terlebih dahulu.

Gambar 10. Ilustrasi Queue

Keunggulan tipe Queue :

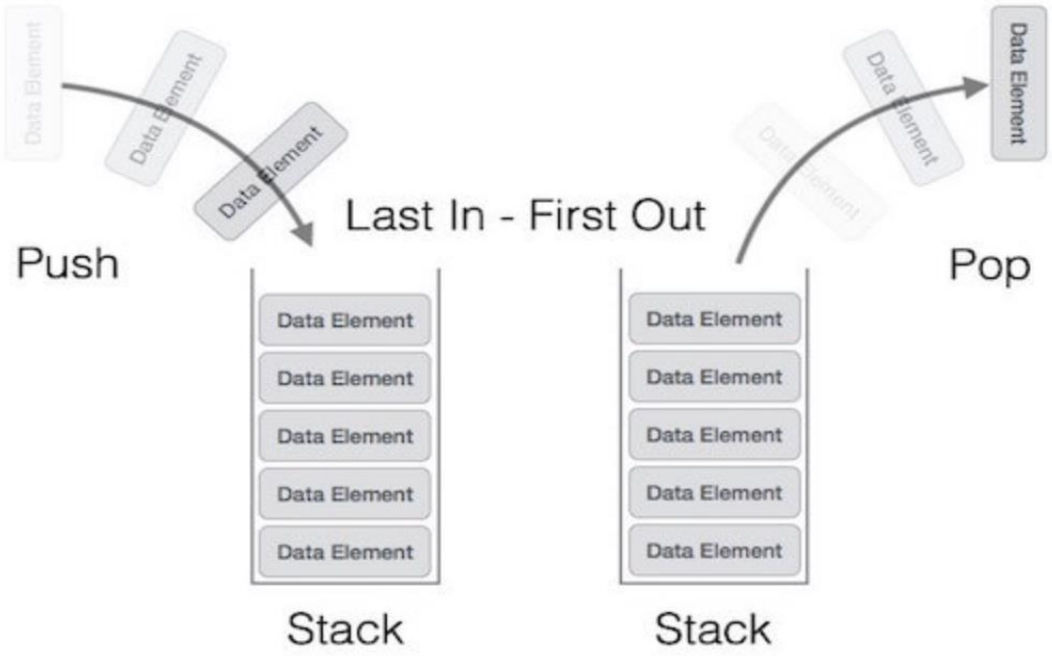
- Data yang masuk akan dilayani sesuai urutannya.
- Proses antrian data lebih cepat dan optimal.
- Menangani beberapa tipe data sekaligus.

Kekurangan tipe Queue :

- Jika waktu pelayanan habis, maka data yang terakhir masuk tidak bisa dilayani.
- Proses yang rumit saat harus menambah atau menghapus elemen dari tengah.
- Butuh waktu lama untuk mencari antrian.

4. *Stack*

Stack adalah tipe struktur data yang linier dan mengikuti urutan tertentu. Adapun urutan yang digunakan adalah LIFO (*Last In First Out*) atau FILO (*First in Last Out*). Kedua istilah tersebut artinya sama, yaitu data yang terakhir masuk akan menjadi data yang keluar pertama kali. Sebaliknya, data yang pertama masuk akan menjadi data yang keluar terakhir.



Gambar 11. Ilustrasi Stack

Keunggulan tipe *Stack* :

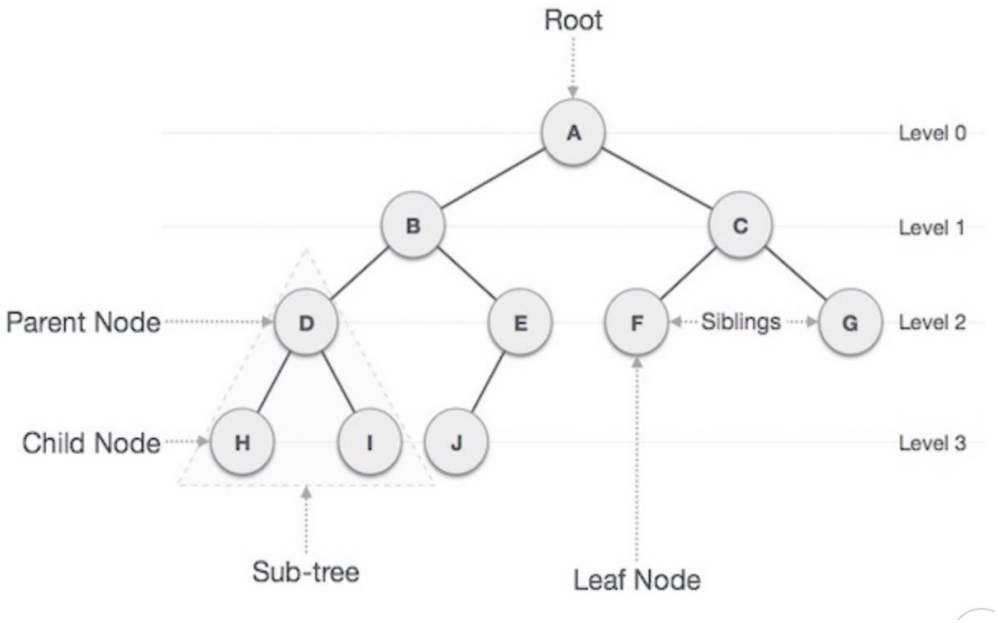
- Dapat mengelola data secara efisien.
- Bisa membersihkan object secara otomatis.
- Dapat mengontrol memori dengan mandiri.

Kekurangan tipe *Stack* :

- Kapasitas memori yang sangat terbatas.
- Kemungkinan terjadi *overflow* ketika jumlah object terlalu banyak.
- Tidak dapat mengakses data secara acak.

5. *Tree*

Tree adalah tipe struktur data yang memiliki bentuk seperti pohon. Tipe *tree* efisien untuk menyimpan data secara hierarkis karena disusun dalam berbagai level. Jadi, tipe ini sering dianggap sebagai kumpulan *node* yang saling dihubungkan.



Gambar 12. Ilustrasi *Tree*

Setiap *node* bisa berisi beberapa data atau link dari *node* lainnya. Beberapa istilah yang ada pada tipe *tree* antara lain :

- *Root* : *node* yang berada di paling atas.
- *Child node :* turunan dari setiap *node.*
- *Parent node* : *node* yang berisi *sub-node.*
- *Siblings : node* yang berasal dari *parent node* yang sama.
- *Leaf node : node* yang tidak memiliki turunan lagi.

Keunggulan tipe *tree* :

- Proses pencarian data bisa dilakukan dengan cepat.

Kekurangan tipe *tree* :

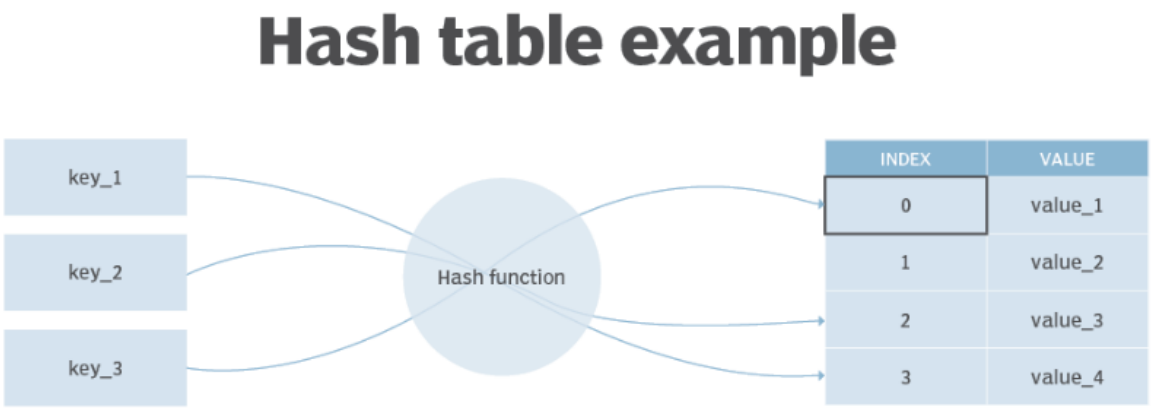
- Membutuhkan waktu yang lebih lama untuk memasukkan data karena harus menyesuaikan dengan urutan hasilnya.

6. *Hash*

Hash table adalah tipe yang digunakan untuk menyimpan data secara asosiatif. Tipe ini akan menyimpan data dalam format *array.* Hal ini memungkinkan untuk mengakses data dengan cepat karena cukup menggunakan indeksnya saja.

Operasi utama yang digunakan dalam *hash table* adalah *search* (

untuk mencari elemen), *insert* (untuk menyisipkan elemen), dan *delete* (untuk menghapus elemn). Contoh penggunaan tipe *hash table* adalah mencari data nama dan nomor telepon.



Gambar 13. Ilustrasi Hash Table

Hash Table juga diketahui sebagai *hash map* yang menyimpan kumpulan item dalam *array* asosiatif yang memplot kunci ke nilai. *Hash table* menggunakan fungsi hash untuk mengkonversi indeks menjadi *array buckets* yang berisi item data yang diinginkan.

Keunggulan tipe *hash table* :

- Daripada tipe lainnya, kadang penggunaan *hash table* lebih efisien untuk mencari data sehingga sering dipakai di *software* untuk pengindeksan basis data.
- Mudah melakukan sinkronisasi.

Kekurangan tipe *hash table* :

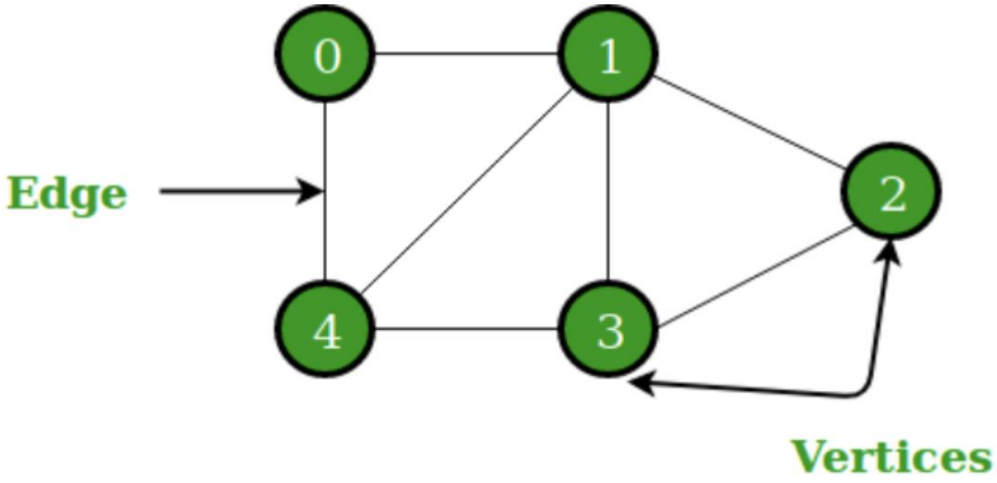
- Kemungkinan bentrokan data sangat besar sehingga menjadi tidak efisien.

7. *Graph*

Graph adalah tipe yang berisi beberapa *node* yang saling terhubung. *Node* pada tipe *graph* disebut sebagai simpul. Jadi, setiap garis akan saling menghubungkan dua simpul. Biasanya tipe ini digunakan untuk menunjukkan jaringan tertentu. Contohnya seperti jaringan telepon.

Tipe *graph* terbagi menjadi dua jenis jenis, yaitu *directed graph* dan

undirected graph. *Directed graph* artinya setiap garis akan terhubung ke semua simpul. Sedangkan *undirected graph* artinya tidak semua simpul akan terhubung dengan garis. Jika sebuah simpul tidak terhubung dengan simpul lainnya, maka disebut dengan *isolated vertex*.



Gambar 14. Ilustrasi Graph

Keunggulan tipe *graph* :

- Data membantu memeriksa hubungan antar *node* dengan cepat.
- Cocok digunakan untuk grafik yang tidak mengandung banyak *node.*

Kekurangan tipe *graph* :

- Membutuhkan waktu lama untuk memodifikasi data.

D. Struktur Data pada Basis Data

Basis Data adalah sekumpulan dari data yang disimpan secara terorganisir dalam tabel yang berisi baris dan kolom menggunakan perangkat lunak yang dikenal sebagai *Database Management System* (DBMS).

DBMS digunakan untuk memodifikasi, mendefinisikan, memanipulasi dan mengolah data. Beberapa contoh DBMS adalah MySql, Oracle, PostgreSql dll.

Struktur Data adalah cara mengatur dan menyimpan data sehingga memudahkan pengaksesan dan modifikasi data. Ada berbagai jenis struktur data yang hadir untuk menyimpan data dan struktur data yang berbeda

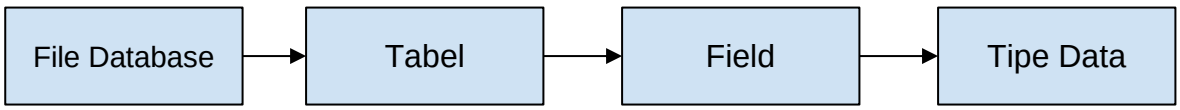
memiliki fitur yang berbeda dan unik. Beberapa struktur data yang umum digunakan adalah *Array, Linked List, stack Queue, dll.*

Berikut adalah perbedaan antara Basis Data dan Struktur data

Tabel 1. Perbedaan mendasar antara Basis Data dan Struktur Data

Basis Data	Struktur Data
Data yang terorganisir	Menggunakan format khusus untuk menyimpan data untuk tujuan yang khusus.
Digunakan untuk mengakses data yang mengelolanya dengan mudah.	Digunakan untuk efisiensi dan untuk mengurangi kerumitan Program.
Structured Query Language (SQL) digunakan untuk melakukan operasi pada data dalam database. DBMS digunakan untuk mengelola database.	Bahasa pemrograman seperti C, C++, Java, Python digunakan untuk melakukan operasi menggunakan Struktur data.
Basis data relasional, basis data NOSQL, basis data terpusat, basis data hierarkis adalah beberapa jenis basis data.	Struktur data linier dan struktur data non-linier adalah jenis struktur data.
Menyimpan Data dalam Memori permanen	Menyimpan Data dalam Memori sementara
Non Volatile Memory	*Volatile Memory*

Model data relasional pada suatu basis data merupakan suatu kumpulan table relasional. Suatu tabel relasional adalah suatu file flat yang terdiri atas sekumpulan kolom dan sejumlah baris.



Gambar 15. Skema data Relasional

- File Database

  File utama meliputi keseluruhan basis data dan disimpan ke media penyimpanan.

- Tabel

  Sebuah tabel adalah sekumpulan data spesifik, terdapat berbagai macam field dan rekord
- Field

  Field adalah kategori (kolom) yang ada dalam sebuah tabel yang memiliki tipe data yang berbeda-beda.
- Tipe Data

  Properti dari setiap field yang terdiri dari alfanumerik, numerik, date, suara dan gambar.

Adapun macam-macam dari tipe data relasional adalah sebagai berikut :

Tabel 2. Tipe data relasional pada database

Tipe Data	Keterangan
CHAR	Tipe data karakter / string dengan panjang tetap.
VARCHAR	Tipe data karakter dengan panjang tidak tetap
NUMERIC	Tipe data numerik real
DEC	Tipe numerik yang mengandung nilai desimal
INTEGER	Tipe data bilangan bulat
FLOAT	Tipe data bilangan real
BLOB	Tipe data biner untuk menyimpan data gambar atau suara
DATE	Tipe data tanggal
DATETIME	Tipe data tanggal dan waktu
AUTO INCREMENT	Tipe data numerik yang akan dinaikkan sebesar satu secara otomatis
BOOLEAN	Tipe data logic (true / false)

Berikut ini adalah contoh data relasional dan penerapan pada tabel-nya :

Tabel 3. Contoh Tabel Pegawai

Nama Field	Tipe Data	Panjang
nip	Varchar	15
nama	Varchar	255
tgl_lahir	Date	
pegawai_aktif	Boolean	

Tabel 4. Contoh Isi Tabel Pegawai

nip	nama	tgl_lahir	pegawai_aktif
1990101212010001	Budi Raharjo	1990-10-10	true
1991101212010002	Adam Malik	1991-10-12	true
1992101212010001	Intan Permatasari	1992-12-13	true

Ada dua macam tabel yaitu :

1. Tabel Master

   Berisi tentang data-data master yang digunakan untuk referensi pada tabel transaksi, seperti tabel master pegawai, tabel master jabatan, tabel master satuan kerja dsb.

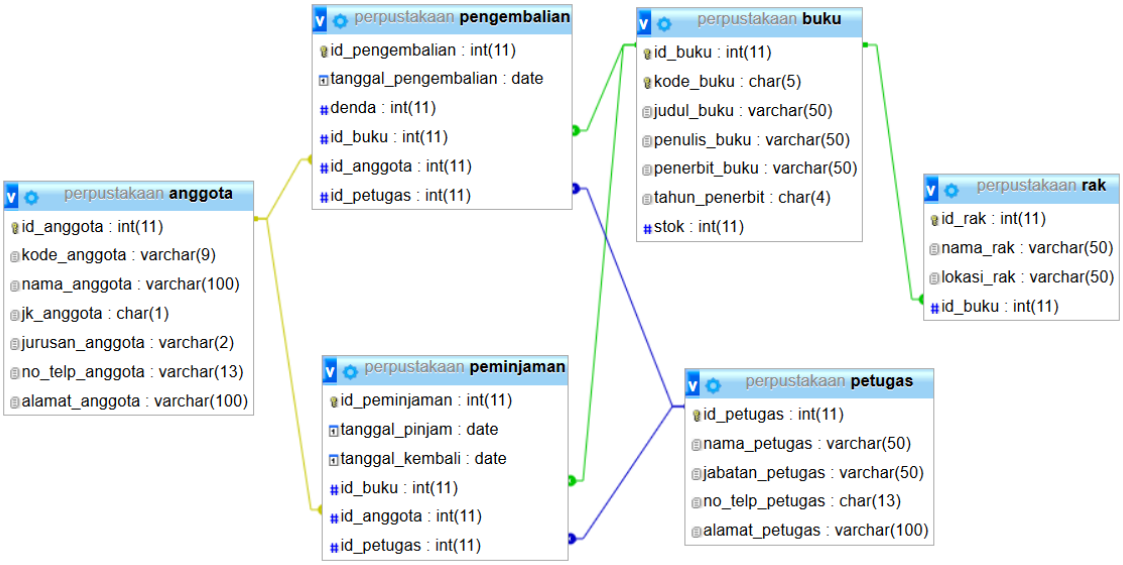
2. Tabel Transaksi

   Berisi tentang data-data transaksi yang terdiri paling sedikit 1 relasi pada tabel master data, seperti tabel transaksi peminjaman buku, tabel transaksi pengembalian buku, tabel transaksi keuangan dsb.

Dalam menghubungkan antara tabel master dan transaksi terdapat kunci yang terdiri dari dua jenis, yaitu kunci utama dan kunci tamu. Kunci utama (primary key) adalah suatu kolom dimana nilai unik digunakan untuk mengidentifikasi setiap basis data di dalam tabel. Ini digunakan untuk mencegah data menjadi duplikat. Kunci utama harus memiliki karakteristik : Mandatory (tidak boleh null), Unique (tidak sama dengan yang lainnya), Stable (tidak berubah), Short (tidak terlalu panjang). Sedangkan kunci tamu (Foreign Key) adalah nilai kolom pada suatu tabel yang diperlukan untuk

memenuhi nilai kolom kunci utama pada tabel yang lainnya.

Berikut adalah contoh skema basis data peminjaman buku :



Gambar 16. *Skema Database Perpustakaan*

E. Rangkuman

Desain Database adalah satu hal yang penting yang harus dilakukan dalam pengembangan database dengan memperhatikan kebutuhan user dan kebutuhan hardware. Tahapan yang perlu dilakukan adalah analisis kebutuhan, logical design, physical design, database implementation, monitoring, dan modification.

Struktur data adalah bagian yang paling krusial dalam mendesain sebuah software yang efisien.

Struktur data juga menjadi dasar dari desain algoritma yang nanti akan digunakan dalam jalannya program komputer. Secara hirarki struktur data dibagi menjadi dua, yaitu Struktur Data Primitif dan Struktur data Non-Primitif. Struktur data juga sering diklasifikasikan berdasarkan karakteristiknya, yaitu Linear non Linear, Homogen Heterogen, dan Statis Dinamis. Jenis struktur data yang digunakan dalam situasi tertentu ditentukan oleh jenis operasi yang akan dibutuhkan atau jenis algoritma yang akan diterapkan, seperti Array, Linked list, Queue, Stack, Tree, Hash, Graph.

Struktur Data pada basis data adalah sekumpulan dari data yang disimpan secara terorganisir dalam tabel yang berisi baris dan kolom menggunakan perangkat lunak yang dikenal sebagai *Database Management System* (DBMS).

F. Evaluasi

Untuk menguji pemahaman Anda terkait materi Struktur Data, silahkan kerjakan latihan di bawah ini.

1. Jelaskan menggunakan bahasa anda mengenai struktur atau susunan penyimpanan data pada basis data relasional!
2. Jelaskan pengertian dari *Structure Query Language* (SQL) dan perintah - perintah yang terdapat didalamnya!
3. Sebutkan algoritma yang berhubungan erat dengan struktur data
4. Jelaskan tentang node dan Indeks menurut anda
5. Sebutkan apa saja yang termasuk struktur data linear dan non linear

6. Jelaskan Perbedaan Basis data dan Struktur data
7. Silahkan melakukan percobaaan mendesain database relasional *Point Of Sales*

MATERI POKOK 3:

STANDARDISASI DATA

Indikator Hasil Belajar:
Peserta mampu melakukan standardisasi data sesuai dengan kaidah atau standar yang berlaku.

A. Konsep Standardisasi Data

Menurut Peraturan Badan Pusat Statistik (BPS) No. 4 Tahun 2020 tentang Petunjuk Teknis Standar Data Statistik, standardisasi data adalah proses untuk membawa data ke dalam format umum yang memungkinkan untuk perbandingan data, analisis lintas sektor yang bersifat kolaboratif, dan berbagi pakai data. Dalam konteks yang lebih teknis, standardisasi data mengacu pada pengkondisian input data untuk memastikan bahwa data memenuhi aturan konten dan format data (DAMA International, 2015). Dengan demikian, dalam standarisasi data perlu ditetapkan standar data sebagai acuan dan diterapkan dalam pengelolaan basis data serta pertukaran data. Penetapan standar data dilakukan dengan konsensus antar stakeholder yang menghasilkan data, sehingga dicapai kesepakatan standar data yang harus dipatuhi bersama.

Standarisasi data ini sangat penting baik untuk data didalam basis data itu sendiri, maupun data yang berlaku lintas basis data, lintas unit kerja, lintas instansi, dan bahkan lintas negara. Seringkali masing-masing pengelola basis data mendefinisikan sendiri konten dan format data dengan konteks bisnis masing-masing tanpa memperhatikan konteks bisnis yang lebih luas. Bahkan terkadang elemen/ atribut data yang sama dalam satu tabel juga mempunyai format isian yang berbeda. Hal ini mengakibatkan data yang dihasilkan berbeda-beda baik dari sisi konten maupun format, padahal data tersebut mempunyai konteks bisnis yang sama. Data seperti ini sulit diintegrasikan dan dibagipakaikan, perlu biaya dan waktu yang tinggi, karena perlu proses standarisasi yang tidak mudah pada saat konsolidasi data. Berikut contoh berbagai kondisi data yang tidak

terstandardisasi.

Tabel 5. Contoh Isian Elemen Data Tidak Standar Dalam Satu Tabel

kode_kegiatan	judul_kegiatan	bidang_penelitian	jenis_kegiatan
001	Pengembangan Model Identifikasi Daerah Rawan Banjir	Penginderaan Jauh	Penelitian Terapan
002	Pengembangan Vaksin Vius Influenza xxx	Kesehatan dan obat	Terapan
003	Identifikasi jamur jenis baru	Biology	Penelitian Dasar
004	Pengembangan Model Penghitungan Luas Daerah Terbakar	Remote sensing	Litbang Terapan

Pada Tabel 5 disajikan contoh data yang tidak seragam dalam satu elemen data dalam satu tabel, yaitu pada kolom bidang_penelitian dan jenis_kegiatan. Data tersebut seharusnya dapat distandardisasi seperti yang disajikan pada Tabel 6.

Tabel 6. Standarisasi Data Tabel 2

kode_kegiatan	judul_kegiatan	bidang_penelitian	jenis_kegiatan
001	Pengembangan Model Identifikasi Daerah Rawan Banjir	Penginderaan Jauh	Penelitian Terapan
002	Pengembangan Vaksin Vius Influenza xxx	Kesehatan dan obat	Penelitian Terapan
003	Identifikasi Jamur Kuping Jenis Baru	Biologi	Penelitian Dasar
004	Pengembangan Model Penghitungan Luas Daerah Terbakar	Penginderaan Jauh	Penelitian Terapan

Standarisasi data ini sangat penting untuk menyeragamkan konten dan format data sehingga:

1. memudahkan pengumpulan data karena tersedia standar data yang dapat digunakan sebagai acuan dalam pengumpulan data;

2. memberikan pemahaman yang jelas dan seragam terkait data yang dihasilkan;
3. memudahkan pertukaran data karena antar basis data sudah menggunakan standar data yang sama;
4. memudahkan integrasi data karena konsistensi data yang dihasilkan antar basis data terjaga;
5. memudahkan pengolahan dan analisis data karena data yang dihasilkan sudah standar sehingga mengurangi proses pembersihan dan standardisasi data pada saat preprosesing data;
6. memberikan jaminan kualitas data karena data yang berkualitas adalah yang sesuai dengan standar data yang ditetapkan.

B. Standar Data

Penggunaan standar data mampu menurunkan ambiguitas data yang dihasilkan beragam produsen data. Standar data dapat digunakan sebagai garansi kualitas data itu sendiri. Karena data dikatakan berkualitas jika memenuhi standar data itu sendiri. Standar data ini diturunkan dari definisi bisnis data tersebut, sehingga dalam penyusunan standar data harus memahami aspek bisnis terlebih dahulu terhadap data yang akan distandarkan. Menurut perpres no 39 tahun 2019 tentang Satu Data Indonesia, standar data adalah standar yang mendasari data tertentu. Standar data yang ditetapkan minimal mencakup konsep, definisi, klasifikasi, ukuran, dan satuan. Menurut Loshin (2010), standar data adalah kesepakatan antara pihak-pihak tentang definisi istilah bisnis umum dan cara istilah tersebut diberi nama dan direpresentasikan dalam data. Standar menggabungkan seperangkat aturan yang menjelaskan bagaimana objek data disimpan, dipertukarkan, diformat, atau disajikan dan mencakup aturan-aturan yang dengannya informasi dibagikan. Ini termasuk:

- Identifikasi dan definisi istilah bisnis umum,
- Penentuan objek data mana yang akan dibagikan,
- Daftar elemen data yang menyusun objek data tersebut, dan

• Aturan penamaan elemen data, format/struktur, dan penyajian.

Dengan demikian, dua komponen utama yang harus ada dalam standar data meliputi:

1. Definisi objek data dan juga elemen-elemen data yang menyusun objek tersebut sesuai istilah bisnis umum. Definisi objek data ini, meliputi definisi, klasifikasi, ukuran, satuan, dan dasar rujukan definisi.
    a. Konsep merupakan ide yang mendasari Data dan tujuan Data tersebut diproduksi.
    b. Definisi merupakan penjelasan tentang Data yang memberi batas atau membedakan secara jelas arti dan cakupan Data tertentu dengan Data yang lain.
    c. Klasifikasi merupakan penggolongan Data secara sistematis ke dalam kelompok atau kategori berdasarkan kriteria yang ditetapkan oleh Pembina Data atau dibakukan secara luas.
    d. Ukuran merupakan unit yang digunakan dalam pengukuran jumlah, kadar, atau cakupan.
    e. Satuan merupakan besaran tertentu dalam Data yang digunakan sebagai standar untuk mengukur atau menakar sebagai sebuah keseluruhan.
    f. Dasar rujukan merupakan sumber referensi yang menjadi rujukan dalam penentuan definisi objek data.
2. Format data yang meliputi penamaan elemen data, format/ struktur, dan penyajian. Format data dapat mencakup tipe data, panjang data, format penulisan.
    a. Tipe data merupakan jenis tipe data yang biasa dikenal dalam bahasa pemrograman dan komputer yang digunakan sebagai bentuk klasifikasi data untuk mempermudah kategori dalam bahasa pemrograman (integer, float, char, string, dsb).
    b. Panjang data adalah panjang minimal atau maksimal data yang akan disimpan.

c. Format penulisan adalah format penulisan yang merepresentasikan data.

Berikut contoh matriks standar data yang dapat digunakan sebagai acuan dalam standardisasi data.

Tabel 7 .Contoh matriks standar data

Nama Data	Kegiatan riset
Definisi	Kegiatan riset adalah kegiatan kreatif yang dilakukan dengan sistematis untuk menambah pengetahuan (*stock of knowledge*), dan pemanfaatan pengetahuan untuk merancang penerapan baru (*to devise new applications*)
Klasifikasi	Riset dasar, riset terapan, dan pengembangan eksperimental
Ukuran	-
Satuan	-
Dasar rujukan	Frascatti Manual 2015

Rincian standar data pada masing-masing elemen data

Elemen Data	**Penulisan**	**Definisi**	**Klasifikasi**	**Ukuran**	**Satuan**	**Tipe data**	**Panjang**	**Format Penulisan**
Judul Kegiatan	judul_keg	Judul kegiatan riset	-	-	-	String	255	Capitalize Each Word
Jenis Kegiatan	jenis_keg	Kategori kegiatan riset	Riset dasar, riset terapan, pengembangan eksperimental	-	-	String	30	Capitalize Each Word

Elemen Data	Penulisan	Definisi	Klasifikasi	Ukuran	Satuan	Tipe data	Panjang	Format Penulisan
Bidang Riset	bidang_riset	Kategori bidang riset	Klasifikasi bidang riset yang merujuk pada Frascatti Manual	-	-	String	50	Capitalize Each Word
Lokasi	lokasi	Lokasi level provinsi dimana kegiatan riset dilaksanakan	Klasifikasi wilayah provinsi yang merujuk pada Kemendagri	-	-	String	50	Capitalize Each Word
Tahun pelaksanaan	tahun	Tahun dilaksanakannya kegiatan riset dan inovasi	-	-	-	Date (year)	4	Angka tahun minimal > 2020
Ketua Peneliti	ketua	Nama ketua kegiatan riset	-	-	-	String	50	huruf
Anggaran	anggaran	Jumlah dana yang dialokasikan untuk kegiatan riset	-	Jumlah	Rupiah	Numeric	6-15	angka

C. Teknik Standardisasi Data pada Basis Data

Standardisasi data perlu dilakukan, baik pada proses perekaman data maupun pada data yang sudah tersimpan dalam basis data. Teknik standardisasi data pada proses perekaman data, yaitu cara untuk mengkondisikan agar data yang akan disimpan sesuai dengan standar. Sedangkan teknik standardisasi data pada data yang sudah tersimpan,

yaitu cara memperbaiki data ke dalam standar yang sudah ditetapkan. Untuk melakukan standardisasi perlu ditetapkan terlebih dahulu standar data yang disesuaikan dengan kebutuhan bisnis seperti yang sudah dijelaskan pada subbab Standar Data. Standar data tersebut yang menjadi acuan dalam standardisasi data.

Berikut beberapa cara yang dapat dilakukan untuk standardisasi data pada proses perekaman data dalam basis data:

1. Membuat batasan/ validasi pada elemen data di basis data

   Yaitu dengan mengatur tipe data yang sesuai, panjang data yang boleh disimpan, dan menerapkan *constraint* pada basis data dengan mengacu pada standar data yang sudah ditetapkan. Constraint/ batasan dalam basis data adalah pembatasan nilai-nilai yang diperbolehkan untuk diisikan pada sebuah kolom yang terdapat pada basis data. Sebagai contoh dengan mengatur data yang boleh/ tidak boleh null, rentang nilai yang boleh disimpan, dan relasi dengan tabel lain yang menjadi *foreign key*. Cara ini akan menstandarkan data pada saat penyimpanan data dalam basis data.

   Berikut beberapa jenis *constraint* yang sering digunakan:

   a. Not Null

      Constraint Not Null berfungsi untuk membatasi setiap data yang dimasukkan pada kolom pada tabel basis data harus memiliki nilai.

      Contoh:
      CREATE TABLE periset (kode Integer NOT NULL, nama Varchar (30) NOT NULL, alamat Varchar(30));

   b. Unique

      Constraint Unique berfungsi untuk membatasi setiap data yang dimasukkan pada elemen data pada tabel basis data, data yang dimasukkan harus berbeda dengan data yang ada sebelumnya sehingga tidak ada data yang sama dalam satu kolom. Constraint ini bisa dipasangkan dalam beberapa kolom dalam satu tabel.

Contoh:

CREATE TABLE periset (kode Integer UNIQUE, nama Varchar (30), alamat Varchar(30));

c. Primary Key

Constraint Primary Key berfungsi sebagai acuan dalam suatu tabel data. Berbeda dengan Unique Constraint yang dapat dipasangkan ke dalam beberapa kolom dalam satu tabel, Constraint primary key ini hanya bisa dipasangkan pada salah satu kolom saja dalam satu tabel. Jadi tidak mungkin ada 2 primary key yang dipasang dalam 2 kolom di suatu tabel. Pasti hanya ada satu kolom primary key dalam satu tabel.

Contoh:

CREATE TABLE periset (kode integer, nama varchar(30), alamat varchar(30), PRIMARY KEY (kode));

d. Foreign Key

Constraint Foreign Key berfungsi sebagai rujukan ketika memasukkan data dalam suatu kolom. Jika ada data yang akan dimasukkan ke kolom yang memiliki constraint Foreign Key, maka data tersebut sebelumnya sudah harus ada di tabel lain yang menjadi rujukan.

Contoh:

CREATE TABLE kegiatan_riset (kode_kegiatan integer, tgl_kegiatan date, nama_kegiatan varchar(50), Primary Key (kode_kegiatan), FOREIGN KEY (kode_periset) REFERENCES periset (kode));

e. Check

Constraint check berfungsi untuk melakukan pengecekan data sebelum disimpan di dalam kolom tabel pada basis data. Apabila data yang akan disimpan di kolom tidak sesuai persyaratan yang dibuat, maka data tersebut tidak dapat disimpan di kolom tersebut, sehingga basis data akan menampilkan pesan error bahwa data tersebut tidak lolos uji cek.

Contoh:

CREATE TABLE periset (kode integer, nama varchar (30), CHECK (nama <> '-'), alamat varchar(30));

f. Default

Constraint Default menentukan nilai default untuk kolom pada tabel basis data. Default kolom adalah sebuah nilai yang akan dimasukkan dalam kolom secara otomatis ketika transaksi Insert tidak memasukkan nilai tertentu pada kolom tersebut.

Contoh:

CREATE TABLE kegiatan_riset (kode char(4) NOT NULL, judul varchar (30) NOT NULL, anggaran Numeric DEFAULT 0);

Selain penggunaan *constraint*, dapat juga digunakan tipe data yang berisi kumpulan konstanta seperti penggunaan tipe data ENUM pada basis data. Sebagai contoh:

CREATE TABLE periset (kode Integer NOT NULL, nama Varchar (30) NOT NULL, jenis_kelamin ENUM ("L","P"), alamat Varchar(30));

Validasi pada basis data ini harus sinkron dengan validasi pada sisi aplikasi yang digunakan untuk pengisian data. Apalagi untuk validasi yang tidak dapat diakomodir pada basis data, harus dapat diakomidir pada sisi aplikasi.

2. Menggunakan data induk dan referensi yang menjadi rujukan bersama

Data induk adalah data yang merepresantasikan objek atau entitas dalam proses bisnis yang memberikan konteks pada transaksi bisnis dan analisis data, sebagai contoh data pegawai, data infrastruktur, data unit atau instansi, dsb. Sedangkan data referensi adalah komponen yang mendiskripsikan substansi data yang berupa spesifikasi dan kategorisasi dan ketentuan mengenai data, sebagai contoh data bidang kepakaran, data klasifikasi baku lapangan usaha Indonesia (KBLI), dsb.

Penggunaan data induk dan referensi pada basis data akan menyeragamkan data yang digunakan bersama. Data induk dan referensi yang digunakan sebaiknya menggunakan data induk dan

referensi yang menjadi rujukan bersama, baik lingkup instansi, nasional, ataupun global. Contoh data induk yang dapat menjadi acuan nasional seperti data induk kependudukan dan data wilayah administrasi pemerintahan yang dikelola oleh Kementerian Dalam Negeri, data induk perguruan tinggi yang dikelola oleh Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi, dsb. Contoh data referensi yang dapat menjadi rujukan nasional seperti Klasifikasi Baku Lapangan Usaha Indonesia (KBLI) yang dikelola oleh Badan Pusat Statistik, data program studi pendidikan tinggi yang dikelola oleh Kemendikbud, dsb.

3. Standardisasi data pada data yang sudah tersimpan dalam basis data

   Meskipun standardisasi data sudah dilakukan pada saat proses perekaman data, terkadang data yang tersimpan masih ada data yang tidak sesuai standar. Untuk data yang sudah tersimpan dalam basis data, berikut langkah yang perlu dilakukan untuk standardisasi data.

   a. Melakukan asesmen terhadap kondisi data dengan mengacu pada standar data.

   b. Melakukan penanganan terhadap data yang tidak sesuai standar.

      - Format penulisan. Data yang tidak sesuai format akan menjadi masalah pada saat pengolahan dan analisis data. Misalnya format tanggal ada yang “dd/mm/yy” dan “mm/dd/yy”, ini perlu diseragamkan sesuai standar yang sudah ditentukan.

      - Penyeragaman penggunaan klasifikasi. Pada proses ini ditentukan terlebih dahulu klasifikai yang sesuai dengan standar, kemudian seragamkan data lain yang belum menggunakan klasifikasi tersebut. Misalnya jenis_kelamin sesuai standar adalah “L”, dan “P”, maka data yang tidak sesuai klasifikasi tersebut disesuaikan.

      - Penyeragaman penggunaan data referensi. Pada proses ini ditentukan terlebih dahulu data referensi yang sesuai dengan standar, kemudian sesuaikan data yang belum menggunakan referensi tersebut.

D. Rangkuman

Standardisasi data adalah proses untuk membawa data ke dalam format umum yang memungkinkan untuk perbandingan data, analisis lintas sektor yang bersifat kolaboratif, dan berbagi pakai data. Dalam konteks yang lebih teknis, standardisasi data mengacu pada pengkondisian input data untuk memastikan bahwa data memenuhi aturan konten dan format data. Standarisasi data ini sangat penting baik untuk data didalam basis data itu sendiri, maupun data yang berlaku lintas basis data, lintas unit kerja, lintas instansi, dan bahkan lintas negara.

Standar data adalah kesepakatan antara pihak-pihak tentang definisi istilah bisnis umum dan cara istilah tersebut diberi nama dan direpresentasikan dalam data. Penggunaan standar data mampu menurunkan ambiguitas data yang dihasilkan beragam produsen data. Standar data dapat digunakan sebagai garansi kualitas data itu sendiri. Standar data menggabungkan seperangkat aturan yang menjelaskan bagaimana objek data disimpan, dipertukarkan, diformat, atau disajikan dan mencakup aturan-aturan yang dengannya informasi dibagikan.

Standardisasi data perlu dilakukan, baik pada proses perekaman data maupun pada data yang sudah tersimpan dalam basis data. Teknik standardisasi data pada proses perekaman data, yaitu cara untuk mengkondisikan agar data yang akan disimpan sesuai dengan standar. Sedangkan teknik standardisasi data pada data yang sudah tersimpan, yaitu cara memperbaiki data ke dalam standar yang sudah ditetapkan. Untuk melakukan standardisasi perlu ditetapkan terlebih dahulu standar data yang disesuaikan dengan kebutuhan bisnis seperti yang sudah dijelaskan pada subbab Standar Data. Standar data tersebut yang menjadi acuan dalam standardisasi data.

E. Evaluasi

Untuk menguji pemahaman Anda terkait materi Standardisasi Data, silahkan kerjakan latihan di bawah ini.

1. Identifikasi salah satu basis data yang sudah dikelola pada instansi Anda, pilih salah satu dataset dan buatlah standar datanya!
2. Lakukan asesmen terhadap dataset tersebut sesuai standar data yang sudah dibuat!
3. Lakukan standardisasi data jika ada data yang tidak sesuai standar pada dataset tersebut!
4. Dari hasil asesmen kondisi data, berikan usulan teknik standardisasi data yang perlu dilakukan dari sisi proses perekaman data pada basis data!

MATERI POKOK 4:

NORMALISASI *DATABASE*

Indikator Hasil Belajar:
Peserta mampu:

1. Memahami konsep normalisasi pada database.
2. Memahami konsep kunci-kunci pada database.
3. Melakukan normalisasi tabel pada database.

A. Konsep Normalisasi

Pada saat mendesain tabel-tabel pada basis data relasional, ada beberapa permasalahan yang harus diperhatikan yaitu pada sisi performa, konsistensi dan pemeliharaan. Sebagai contoh pada Tabel 8 apabila sebuah database hanya didesain memiliki satu buah tabel yang terdiri dari banyak atribut, dapat dipastikan akan terdapat banyak data yang berulang, proses temu kembali yang kompleks, serta inkonsistensi pada data ketika user memberikan perintah INSERT, DELETE dan UPDATE terhadap baris-baris data pada tabel tersebut.

Tabel 8. Contoh tabel mahasiswa

sid	nama	alamat	tlp	jurusan	kampus	tlp_kantor	kaprodi
S0001	Witan	Depok	3428719	Sistem Informasi	B	5552222	Dr. Jason
S0002	Leli	Bogor	8472250	Sistem Informasi	B	5552222	Dr. Jason
S0003	Deni	Ciledug	4892849	Sistem Informasi	B	5552222	Dr. Jason
S0004	Yuni	Kota	4820258	Sistem Informasi	B	5552222	Dr. Jason
S0005	Anton	Slipi	8440282	Sistem Informasi	B	5552222	Dr. Jason

Pada Tabel 8 dapat terlihat bahwa informasi mengenai data jurusan yang sama tampil berkali kali (data redundant), yang tentunya akan

menggunakan lebih banyak ruang penyimpanan. Selain menyia-nyiakan ruang penyimpanan, data redundant juga akan menimbulkan masalah lain seperti, diantaranya.

1. INSERT anomaly.

   Pada tabel mahasiswa saat ini sudah terdapat lima mahasiswa dengan jurusan sistem informasi, bayangkan bila terdapat penambahan seratus mahasiswa baru dengan jurusan yang sama. Maka pada tabel tersebut harus dilakukan penambahan informasi nama kampus, nomor telepon dan nama kaprodi yang sama secara berulang sebanyak seratus kali yang berujung pada insert anomaly. Pertimbangkan juga kemungkinan adanya kesalahan input pada nama kaprodi dan nomor telepon atau penambahan jurusan yang tidak dapat dilakukan karena belum adanya mahasiswa yang mendaftar pada jurusan tersebut. Secara singkat INSERT anomaly adalah situasi dimana tidak dapat melakukan penambahan beberapa jenis data secara langsung pada database.

2. DELETE anomaly.

   Penyebab dari terjadinya penambahan informasi secara berulang pada kasus sebelumnya adalah karena pada tabel tersebut terdapat dua informasi yang seharusnya terpisah yaitu data mahasiswa dan data jurusan. Apabila dilakukan penghapusan pada data mahasiswa secara keseluruhan, maka secara tidak langsung data jurusan juga akan terhapus. Kasus ini dinamakan DELETE Anomaly yaitu penghapusan data yang tidak seharusnya dimana data yang seharusnya tidak terhapus tetapi ikut terhapus.

3. UPDATE anomaly

   Bila terdapat pergantian kaprodi pada jurusan tersebut, maka seluruh baris pada tabel yang berisi informasi mengenai data kaprodi tersebut harus ikut diperbaharui. Selain ini merupakan pekerjaan yang membutuhkan banyak waktu, pada prosesnya dimungkinkan ada satu

atau dua baris data pada tabel yang terlewat untuk diperbaharui. Hal ini akan menyebabkan terjadinya inkonsistensi data, karena dimungkinkan jurusan yang seharusnya dipimpin oleh kaprodi x tetapi tercatat pada database dipimpin oleh kaprodi y. Inkonsistensi data pada data yang berulang akan menyebabkan terjadinya UPDATE anomaly.

Beberapa permasalahan diatas dapat diselesaikan dengan teknik normalisasi. Sebagai contoh pada Tabel 8 sebelumnya dapat dipecah menjadi dua tabel yaitu tabel mahasiswa dan tabel jurusan seperti contoh dibawah.

Tabel 9. Contoh tabel mahasiswa setelah perbaikan

sid	nama	alamat	tlp	jurusan
S0001	Witan	Depok	3428719	Sistem Informasi
S0002	Leli	Bogor	8472250	Sistem Informasi
S0003	Deni	Ciledug	4892849	Sistem Informasi
S0004	Yuni	Kota	4820258	Sistem Informasi
S0005	Anton	Slipi	8440282	Sistem Informasi

Tabel 10. Contoh tabel jurusan setelah perbaikan

jurusan	kampus	tlp_kantor	kaprodi
Sistem Informasi	B	5552222	Dr. Jason

Dapat kita lihat setelah tabel kita pecah maka data jurusan pada tabel jurusan cukup disimpan satu kali yang artinya mengurangi pemakaian ruang penyimpan. Pada tabel mahasiswa masih terdapat data redundant pada kolom jurusan, hal ini tetap dipertahankan untuk mempertahankan relasi hubungan antara tabel mahasiswa dan tabel jurusan. Dapat disimpulkan bahwa normalisasi tidak menghilangkan redundansi pada data tapi meminimalisir redundansi.

Lebih lanjut bila terdapat pergantian data pada tabel jurusan, maka cukup dilakukan satu kali perubahan pada tabel jurusan yang secara otomatis akan mempengaruhi tabel mahasiswa secara keseluruhan karena tabel mahasiswa dan tabel jurusan saling terkait. Demikian pula dengan permasalah INSERT anomali dan DELETE anomaly pada kasus diatas secara otomatis terselesaikan dengan normalisasi.

Secara umum normalisasi dapat didefinisikan sebagai proses identifikasi dan pengelompokan atribut dari suatu relasi sehingga menghasilkan struktur relasi yang baik dengan redundansi seminimal mungkin serta menghilangkan sebagian besar ambiguitas pada data (Puspitasari et al, 2016). Beberapa keuntungan dari normalisasi yang juga dapat disimpulkan dari contoh kasus diatas diantaranya adalah:

1. Memaksimalkan penggunaan ruang penyimpanan dengan meminimalisasi redundansi pada data.
2. Meminimalisasi kemungkinan terjadinya inkonsistensi pada data.
3. Meminimalisasi resiko terjadinya data anomali pada saat memodifikasi data

Beberapa tipe dalam normalisasi diantaranya yaitu:

1. Bentuk normal pertama (1NF)
2. Bentuk normal kedua (2NF)
3. Bentuk normal ketiga (3NF)
4. Bentuk normal Boyce-Codd (BCNF)
5. Bentuk normal keempat (4NF)
6. Bentuk normal kelima (5NF)

B. Normalisasi Bentuk Normal pertama (1NF)

Seluruh tabel dalam database harus setidaknya memenuhi normalisasi bentuk normal pertama, jika sebuah tabel tidak memenuhi bentuk normal pertama maka dapat dianggap memiliki desain database yang buruk. Memastikan tabel sudah dalam bentuk normal pertama penting untuk memastikan desain database yang dibuat dapat dengan mudah dikembangkan dan proses temu kembali data dapat dilakukan.

Sebuah tabel dikatakan memenuhi bentuk normal pertama jika dan hanya jika seluruh kolom pada tabel hanya memiliki nilai tunggal (atomic). Yaitu, setiap kolom hanya diperbolehkan memiliki satu nilai pada setiap baris di tabel tersebut. Peraturan lainnya dalam normalisasi bentuk pertama yaitu:

1. Setiap kolom hanya diperbolehkan berisi nilai dengan tipe yang sama.
2. Setiap kolom harus memiliki nama yang unik karena kesamaan pada nama kolom akan mempersulit proses temu kembali data.
3. Urutan penyimpanan pada data tidak berpengaruh pada tabel.

Contoh:

Tabel 11. Contoh tabel mata kuliah

sid	nama	alamat	mata-kuliah	nama
S0002	Witan	Depok	Database, Pemrograman	Jason, Jim
S0003	Leli	12345	Database, Algoritma	Jason, Jeremy
S0001	Deni	Ciledug	Algoritma	Jeremy

Dapat dilihat pada contoh tabel diatas terdapat beberapa aturan yang dilanggar yaitu:

1. Pada nilai kolom mata-kuliah terdapat dua nilai dalam satu baris data.
2. Pada nilai kolom alamat terdapat perbedaan pada tipe data yaitu text dan numerik
3. Pada nama kolom “nama mahasiswa” dan “nama dosen” memiliki nama yang sama.

Untuk memenuhi syarat agar tabel memenuhi persyaratan normalisasi bentuk pertama maka dapat dilakukan perbaikan sebagai berikut:

1. Memisahkan seluruh mata kuliah dan nama dosen pada tabel menjadi dua baris data.
2. Menyesuaikan isian pada data alamat, dan
3. Menambahkan pembeda pada nama kolom menjadi “nama_mahasiswa” dan “nama_dosen”.

Setelah dilakukan perbaikan maka tabel akan menjadi seperti berikut:

Tabel 12. Contoh tabel mata kuliah setelah perbaikan

sid	nama_mahasiswa	alamat	mata_kuliah	nama_dosen
S0002	Witan	Depok	Database	Jason
S0002	Witan	Depok	Pemrograman	Jim
S0003	Leli	Bogor	Database	Jason
S0003	Leli	Bogor	Algoritma	Jeremy
S0001	Deni	Ciledug	Algoritma	Jeremy

C. Konsep mengenai kunci-kunci pada tabel database - Super, Primary, Candidate, Foreign Key, dll

Sebuah tabel pada database memiliki kolom/atribut kunci yaitu sebuah kolom atau bisa terdiri dari gabungan kolom yang berfungsi untuk mengidentifikasi secara unik baris data pada tabel. Selain berguna untuk mengidentifikasi baris data pada tabel, kunci pada tabel database juga berfungsi untuk memastikan integritas data pada sebuah tabel terjaga karena kolom yang menjadi kunci bersifat wajib terisi sehingga bila ada data baru yang dimasukan kedalam tabel dapat ditemukan kembali menggunakan kolom kunci tersebut. Fungsi terakhir kunci pada tabel database adalah sebagai atribut penghubung antar tabel pada database. Berikut adalah penjelasan mengenai beberapa kunci yang ada pada database.

1. Super Key

   Super key adalah seluruh atribut yang dapat mengidentifikasi baris data pada tabel. Super key dapat terdiri dari sebuah kolom maupun gabungan kolom.

2. Candidate Keys

   Candidate keys merupakan subset minimal dari super key. Bila seluruh kolom atau gabungan kolom dapat menjadi super key maka

pada candidate keys hanya dipilih yang paling minimum. Sebagai contoh apabila pada suatu tabel terdapat dua super key yaitu sid, email dan kombinasi sid dan email maka kombinasi sid dan email tidak dapat menjadi candidate key karena subset nya berupa sid dan email juga menjadi super key.

3. Primary Key

   Primary key merupakan kunci yang dipilih untuk mengidentifikasi baris data pada tabel dari beberapa candidate key yang tersedia. Primary key sendiri harus bersifat unik dan wajib terisi.

4. Alternate key

   Alternate key merupakan candidate key yang tidak dipilih untuk menjadi Primary Key.

5. Foreign key

   Foreign Key merupakan kunci asing pada sebuah tabel yang berfungsi sebagai penghubung atau rujukan terhadap tabel utama dari Foreign key tersebut.

6. Composite Key

   Composite key merupakan kunci yang terdiri dari gabungan dua kolom atau lebih pada tabel yang digunakan untuk mengidentifikasikan baris data.

7. Surrogate Key

   Surrogate key merupakan kolom pada tabel yang dibuat khusus dengan tujuan agar dapat digunakan sebagai pengidentifikasi baris data dikarenakan kolom-kolom yang terdapat pada tabel tersebut tidak dapat dijadikan kunci. Sebagai contoh apabila kita menambahkan kolom “Student ID” (sid) pada tabel hanya untuk keperluan menjadi PK dan sid tidak memiliki arti apapun pada data tersebut maka sid disebut sebagai Surrogate Key.

D. Normalisasi Bentuk Normal kedua (2NF)

Sebelum masuk dalam penjelasan mengenai peraturan pada normalisasi bentuk kedua, akan dijelaskan terlebih dahulu konsep functional dependency (FD). Functional dependency (FD) adalah setiap kolom atau atribut bukan kunci yang bergantung pada primary key (PK). Silahkan melihat contoh tabel berikut untuk kemudahan dalam memahami konsep PK dan FD.

Tabel 13. Contoh tabel dengan PK dan FD

sid (PK)	nama_mahasiswa	alamat	mata_kuliah	nama_dosen
S0001	Deni	Ciledug	Database	Jason
S0002	Witan	Bogor	Pemrograman	Jim
S0003	Deni	Bogor	Database	Jason

Pada tabel diatas kolom sid dapat ditetapkan sebagai PK karena dengan menggunakan sid kita dapat mengidentifikasi / memanggil setiap baris data pada tabel secara unik. Dapat kita lihat pada kolom lain terdapat data yang sama sehingga tidak memungkinkan untuk dijadikan sebagai PK. Sebagai contoh bila kita ingin mengetahui alamat dari mahasiswa yang bernama Deni maka ada dua baris data yang terpanggil. Sedangkan bila kita memanggil data menggunakan sid maka data yang ditampilkan hanya baris data yang bergantung pada sid tersebut.

Dapat kita katakan bahwa seluruh kolom seperti nama_mahasiswa, alamat, mata_kuliah dan nama_dosen yang bukan PK bergantung pada sid sebagai PK. Hal ini memberikan contoh secara langsung apa yang dimaksud dengan FD sesuai definisi pada penjelasan sebelumnya.

Setelah mengetahui konsep PK dan FD maka selanjutnya apa sajakah peraturan agar sebuah tabel dapat dikatakan sudah dalam bentuk normalisasi bentuk kedua. Sebuah tabel dikatakan memenuhi normalisasi bentuk kedua jika dan hanya jika tabel tersebut sudah dalam bentuk 1NF dan seluruh kolom yang bukan merupakan kunci utama bergantung secara

functional pada semua PK dan bukan hanya sebagian dari PK (partial dependency).

Contoh:
Diketahui tabel X memiliki lima kolom yaitu A,B,C,D,E; dimana A dan B merupakan Primary key.
Kolom C, D, dan E bergantung penuh pada kolom A dan B sebagai Primary key maka tabel X dapat dikatakan sudah dalam bentuk 2NF.
Dalam notasi dapat kita tuliskan sebagai berikut:
X=(A,B,C,D,E); PK= (A,B).
FD: A,B → C,D,E
A, B → C,D,E artinya:
A,B → C
A,B → D
A,B → E

Sebagai contoh lain tabel X memiliki lima kolom yaitu A,B,C,D,E; dimana A dan B merupakan Primary key.
Kolom C, dan D bergantung pada kolom A dan B sebagai Primary key sedangkan kolom E hanya bergantung kolom B maka tabel X belum memenuhi aturan bentuk 2NF.
X=(A,B,C,D,E); PK= (A,B).
FD: A,B → C,D, dan B → E
A, B → C,D:
A,B → C
A,B → D
B → E

Untuk dapat memenuhi aturan dalam bentuk 2NF maka tabel dapat dipecah menjadi dua tabel yaitu: X1 = (A,B,C,D) dan X2 = (B,E). Dengan bentuk yang baru maka dapat dikatakan bahwa tabel X1 dan X 2 sudah memenuhi aturan normalisasi bentuk kedua.

Contoh kasus:

Tabel 14. Nilai

sid (PK)	Mata_kuliah (PK)	Nilai	nama_dosen
S0001	Database	90	Jason
S0002	Pemrograman	85	Jim
S0003	Database	100	Jason

Pada tabel nilai diatas sudah memenuhi aturan 1NF hanya saja belum memenuhi aturan 2 NF karena hanya kolom nilai yang bergantung pada secara penuh pada PK (sid dan mata_kuliah) sedangkan kolom dosen tidak bergantung pada sid tetapi hanya bergantung pada kolom mata_kuliah saja. Untuk memenuhi aturan 2NF maka dapat dilakukan dekomposisi tabel nilai menjadi dua tabel yaitu tabel nilai dan tabel matakuliah.

Sid (PK)	Mata_kuliah (PK)	Nilai
S0001	Database	90
S0002	Pemrograman	85
S0003	Database	100

MKID (PK)	Mata_kuliah	nama_dosen
MK001	Database	Jason
MK002	Pemrograman	Jim
MK003	Algoritma	Jeremy

Saat ini tabel nilai dan tabel matakuliah sudah dalam bentuk 2NF.

E. Normalisasi Bentuk Normal ketiga (3NF)

Sebuah tabel dikatakan memenuhi normalisasi bentuk ketiga jika dan hanya jika tabel tersebut sudah dalam bentuk 2NF dan setiap kolom yang bukan merupakan kunci tidak bergantung secara fungsional pada kolom

yang juga bukan merupakan kunci dalam tabel tersebut (transitive dependency).

Contoh:

Diketahui tabel X memiliki lima kolom yaitu A,B,C,D,E; dimana A dan B merupakan Primary key.

Kolom C, D, dan E bergantung pada kolom A dan B sebagai Primary key dan kolom D dan E juga bergantung pada kolom C maka tabel X belum dalam bentuk 3NF karena kolom D dan E yang bukan PK bergantung pada kolom C yang juga bukan PK.

Dalam notasi dapat kita tuliskan sebagai berikut:

X=(A,B,C,D,E); PK= (A,B).

FD: A, B → C,D,E dan C → DE

A, B → C,D,E dan C → DE artinya:

A,B → C

A,B → D

A,B → E

C → D

C → E

Untuk dapat memenuhi aturan dalam bentuk 3NF maka tabel dapat dipecah menjadi dua tabel yaitu: X1 = (A,B,C) dan X2 = (C,D,E). Dengan bentuk yang baru maka dapat dikatakan bahwa tabel X1 dan X 2 sudah memenuhi aturan normalisasi bentuk ketiga.

Contoh kasus:

Tabel 15. Penilaian

sid (PK)	Mata_kuliah (PK)	Nilai	tipe_ujian	presentase_nilai
S0001	Database	90	UTS	40%
S0001	Database	85	UAS	60%
S0002	Pemrograman	85	UTS	40%
S0002	Pemrograman	80	UAS	60%

S0003	Database	100	UTS	40%
S0003	Database	100	UAS	60%

Pada tabel diatas kolom presentase_nilai tidak bergantung pada PK tabel yaitu sid dan mata_kuliah tetapi bergantung pada kolom tipe_ujian sehingga tabel tersebut belum dalam bentuk 3NF. Untuk memenuhi aturan 3NF maka dapat dilakukan dekomposisi tabel penilaian menjadi dua tabel yaitu tabel nilai dan tabel tipe_ujian.

sid (PK)	Mata_kuliah (PK)	Nilai	tipe_ujian
S0001	Database	90	UTS
S0001	Database	85	UAS
S0002	Pemrograman	85	UTS
S0002	Pemrograman	80	UAS
S0003	Database	100	UTS
S0003	Database	100	UAS

Tipe_ujian (PK)	presentase_nilai
UTS	40%
UAS	60%

Saat ini tabel penilaian dan tabel tipe_ujian sudah dalam bentuk 3NF

F. Normalisasi Boyce-Codd Normal Form (BCNF)

Sebuah tabel dikatakan memenuhi BCNF jika dan hanya jika tabel tersebut sudah dalam bentuk 3NF dan untuk setiap FD A → B maka A harus merupakan super key. Dalam definisi yang lain jika terdapat kolom primary key yang bergantung pada kolom yang bukan merupakan primary key maka tabel tersebut belum dalam bentuk BCNF.

Contoh:

Diketahui tabel X memiliki tiga kolom yaitu A,B,C, dengan FD: A → B dan B → C. Tabel ini bukan BCNF karena B bukan merupakan super key.

Dalam notasi dapat kita tuliskan sebagai berikut:

X=(A,B,C);.

FD: A → B dan B → C

Apakah A merupakan superkey?

A → B (diketahui)

A → B dan B → C maka A → C (transitif)

A → A (refleksif)

Karena A → (A,B,C) maka A adalah superkey.

Apakah B merupakan superkey ?

B → C (diketahui)

B → B (refleksif)

B → A (x) karena B → (B,C) maka B bukan superkey.

Untuk dapat memenuhi aturan dalam bentuk BCNF maka tabel dapat dipecah menjadi dua tabel yaitu: X1 = (A,B) dengan FD: A → B dan X2 = (B,C) dengan FD: B → C. Dengan bentuk yang baru maka dapat dikatakan bahwa tabel X1 dan X 2 sudah memenuhi aturan BCNF karena kolom A dan kolom B masing masing adalah superkey pada tabel nya.

Contoh kasus:

Tabel 16. Pendaftaran kelas

Sid (PK)	Mata_kuliah (PK)	nama_dosen
S0001	Database	Jason
S0001	Pemrograman	Jim
S0002	Algoritma	Jeremy
S0003	Pemrograman	Jim
S0004	Algoritma	Jackson

- Setiap mahasiswa dapat mendaftar pada lebih dari satu mata kuliah
- Untuk setiap mata kuliah dapat diajar oleh lebih dari satu dosen yang berbeda
- Setiap dosen hanya mengajar satu mata kuliah

Tabel pendaftaran kelas sudah memenuhi aturan 1NF karena tidak ada nilai ganda pada kolom, nama kolom sudah unik dan masing-masing kolom sudah terisi nilai dengan tipe data yang sama. Lebih lanjut tabel pendaftaran kelas juga sudah memenuhi aturan 2NF dan 3NF karena tidak terdapat partial dependencies dan transitive dependencies, yaitu kolom nama_dosen bergantung penuh pada composite key sid dan mata_kuliah.

Permasalahan pada tabel yang tidak memenuhi aturan BCNF adalah kolom mata kuliah dapat bergantung pada kolom dosen dimana kolom dosen bukan merupakan PK sedangkan kolom mata kuliah dan sid bersama merupakan composite key.

Untuk mengatasi masalah ini dapat dilakukan pemecahan tabel menjadi dua tabel yaitu tabel mahasiswa dan tabel dosen. Sehingga aturan BCNF dapat terpenuhi.

Sid (PK)	Nid
S0001	N0001

Nid (PK)	nama_dosen	mata_kuliah
N0001	Jason	Database

G. Normalisasi Bentuk Normal keempat (4NF)

Sebuah tabel dikatakan memenuhi normalisasi bentuk keempat jika dan hanya jika tabel tersebut sudah dalam bentuk BCNF dan tidak boleh memiliki multi-valued dependency (MVD) di dalamnya. Tabel dikatakan memiliki MVD ketika:

1. Diketahui A —> B dan untuk satu nilai pada kolom A terdapat dua atau lebih nilai pada kolom B.

2. Tabel setidaknya memiliki tiga kolom.
3. Untuk kondisi tabel dengan tiga kolom (A,B,C) maka kolom B dan C tidak boleh saing ada ketergantungan.

Contoh:

Nid	mata_kuliah	Hobi
N0001	Database	Olahraga
N0001	Algoritma	Melukis

Pada contoh tabel diatas dapat dilihat bahwa satu mahasiswa memiliki dua mata kuliah dan dua hobi yang berbeda. Kondisi ini tentunya akan menimbulkan dua baris baru dalam tabel agar semua informasi dapat terwakili.

Nid	mata_kuliah	Hobi
N0001	Database	Olahraga
N0001	Algoritma	Melukis
N0001	Database	Melukis
N0001	Algoritma	Olahraga

Solusi dari kondisi ini tidak lain adalah dengan memecah tabel menjadi dua tabel yaitu tabel mata kuliah dan tabel hobi.

Nid	mata_kuliah
N0001	Database

Nid	hobi
N0001	Olahraga

Dengan ini maka tabel sudah memenuhi aturan bentuk normal keempat.

H. Rangkuman

Normalisasi pada database perlu dilakukan untuk memastikan kecepatan pemrosesan data (performa), menjaga agar data dengan nilai yang sama tidak berbeda ketika muncul di tempat lain (konsistensi) dan memudahkan ketika diperlukan adanya perubahan pada data (pemeliharaan). Tujuan dari dilakukannya normalisasi diantaranya adalah menghilangkan ambiguitas, mengurangi redundancy pada data sehingga memaksimalkan ruang penyimpanan dan mengurangi kompleksitas pada data. Beberapa konsep yang harus dipahami dalam normalisasi diantaranya adalah *Functional Dependency, Partial Dependency, Transitive Dependency* dan konsep mengenai kunci-kunci pada database.

I. Evaluasi

Untuk menguji pemahaman Anda terkait materi Normalisasi *Database*, silahkan kerjakan latihan di bawah ini.

1. Jelaskan menggunakan bahasa anda mengapa diperlukan adanya normalisasi pada database!
2. Pada modul diatas sudah dijelaskan beberapa konsep mengenai dependency, jelaskan keterkaitan antara dependency menurut bahasa anda?
3. Jelaskan menurut bahasa anda perbedaan dari normalisasi 3NF dan BCNF!
4. Berdasarkan tabel berikut, lakukan proses normalisasi yang diperlukan sehingga menghasilkan tabel yang memenuhi kondisi normal!

sid	nama	alamat	kd_jur	nama_jur	kd_mk	nama_mk	sks	nilali
S001	Witan	Depok, Malang	SI	Sistem Informasi	SI01	SPK	3	B
S002	Leli	Bogor	SI	Sistem Informasi	SI01	SPK	3	B
S003	Deni	Depok, Surabaya	TI	Teknik Informatika	TI01	Database	4	A
S004	Yuni	Depok, Medan	TI	Teknik Informatika	TI02	Algoritma	4	C
S005	Anton	Slipi	SI	Teknik Informatika	TI03	Pemrograman	4	A

MATERI POKOK 5:

DATAWAREHOUSE

Indikator Hasil Belajar:

Peserta mampu Menjabarkan terminologi umum pada Datawarehouse atau akan sering kita sebut sebagai Gudang Data, Menjabarkan komponen, arsitektur, dan proses pengembangan Gudang Data, Menjabarkan komponen, arsitektur dan proses pengembangan *Extract, Transform* dan *Load.*

A. Defenisi Gudang Data

Gudang data merupakan pengumpulan data yang berorientasi subjek, terintegrasi, berdasarkan varian waktu serta tidak mudah berubah dalam mendukung proses pengambilan keputusan manajemen (inmon, 1993)

a) Subject oriented

Gudang Data di organisasi kedalam beberapa subyek seperti *customer, products* dan *sales* daripada aplikasi besar seperti *(customer invoicing, stock control* dan *product sales)*

b) Integrated

Gudang Data mengintegrasikan aplikasi yang berorientasi data dari berbagi aplikasi dan juga mengikut sertakan data yang tidak konsisten.

c) Time-variant

Data di dalam gudang data hanya akurat dan valid didalam waktu tertentu atau dalam interval waktu tertentu.

d) Non-volatile

Data di dalam gudang data tidak diupdate secara real time namun akan diperbaharui dari operational database secara berkala.

Data di dalam gudang data tidak diupdate secara *real time* namun akan diperbaharui dari database operasional secara berkala. Untuk membedakan antara database operasional biasa dengan Gudang data

dapat melihat gambar dibawah ini.

OLTP systems	Data warehousing systems
Holds current data	Holds historical data
Stores detailed data	Stores detailed, lightly, and highly summarized data
Data is dynamic	Data is largely static
Repetitive processing	*Ad hoc*, unstructured, and heuristic processing
High level of transaction throughput	Medium to low level of transaction throughput
Predictable pattern of usage	Unpredictable pattern of usage
Transaction-driven	Analysis driven
Application-oriented	Subject-oriented
Supports day-to-day decisions	Supports strategic decisions
Serves large number of clerical/operational users	Serves relatively low number of managerial users

Gambar 17. Berikut ini merupakan perbedaan antara database transasksi biasa dengan Gudangdata

B. Komponen Gudang Data

Dalam membangun gudang data terdapat beberapa komponen yang perlu diketahui, berdasarkan Corporate Information Factory (CIF) yang dikembangkan oleh (inmon, 1993). Komponen tersebut dapat kita bagi menjadi beberapa bagian diantaranya.

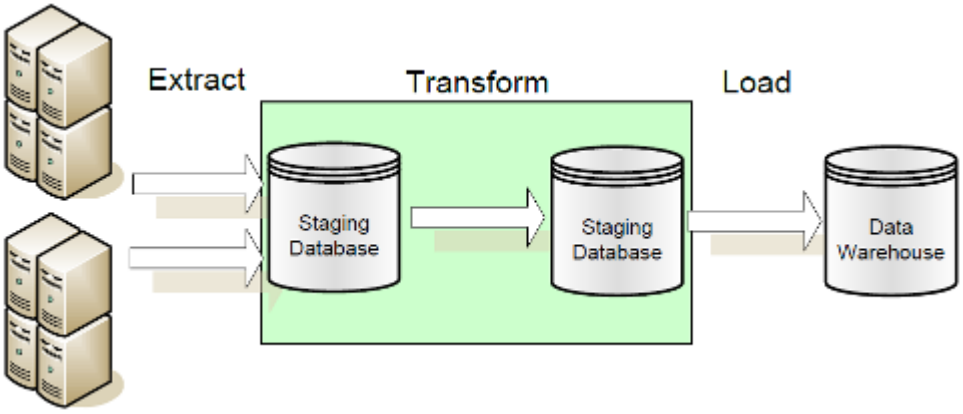
a) External world & Applications

merupakan orang atau sistem yang menghasilkan data operasi sehari - hari. Komponen ini merupakan inputan untuk pengembangan gudang data.

b) Integrasi dan Transformasi Layer

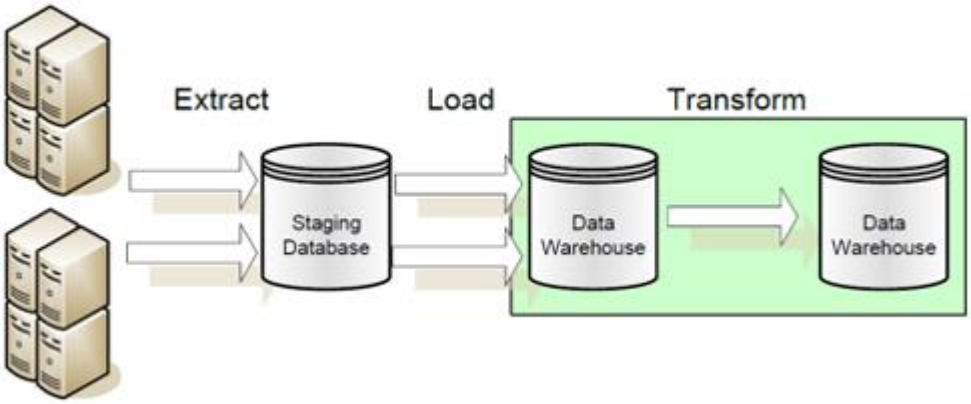
Merupakan lapisan yang mengintegrasikan data dari berbagai sumber dan mengkosolidasi data menjadi satu sumber berdasarkan subyeknya. Pada lapisan ini akan terdapat program komputer untuk melakukan transformasi data dari *external world* menjadi *corporate data.* Data yang diintegrasikan berasal dari berbagi jenis sumber data baik

struktur maupun un-struktur. Pendekatan terkenal saat ini yang dilakukan adalah ETL dan ELT



Gambar 18. proses ekstrak transform dan load

Proses Extract Transform Load (ETL)



Gambar 19. Proses ekstrak load transform

c) *Extraction, Transformation and Loading*

Gudang Data memiliki *back-end tools* dan *utilities* untuk mengumpulkan dan *refresh* data. Berikut ini merupakan *tools* dan *utilities* termasuk fungsi - fungsi yang ada didalamnya

1. *Data extraction* mengumpulkan data dari berbagai sumber, heterogen dan external resources.
2. *Data cleaning* dimana data dibersihkan dari error dan diperbaiki .
3. *Data transformation* dimana data diubah dari format sumber data menjadi data warehouse format.

4. *Load*, dimana data akan di sorting, *summarizes, consolidate,* melakukan pengecekan integritas dan membangun index maupun partisi
5. *Refresh trigger* untuk melakukan updatedari data sources ke warehouse

   Data cleaning dan data transformasi merupakan langkah paling penting dalam meningkatkan kualitas data yang merupakan bagian dari *data mining.*

d) Operational Data Store

Merupakan lapisan yang mengintegrasikan, mendetailkan dan menyimpan data terbaru dari *external world dan applications.* Mengkonsolidasikan data dari berbagi sumber yang terpisah, memiliki kesamaan dengan database transaksional. Struktur berbeda jauh dengan gudang data sehingga disimpan dalam database yang berbeda. Dapat kita sampaikan bahwa operasional data store sebagai konsolidasi operational database.

e) Enterprise Data Warehouse

Merupakan jantung utama dari arsitektur Data warehouse menyimpan data yang berdasarkan subyek, terintegrasi, diringkas dari sumber data external world and applications yang diproses dari lapisan integrasi dan disatukan dalam Operational Data Store. Struktur yang dimiliki oleh Enterprise Data Warehouse sangat berbeda dengan operational data biasanya dalam bentuk model dimensional.

f) Data Marts

Merupakan koleksi data yang disesuikan dengan kebutuhan informasi baik departemen maupun bisnis proses. Menerima masukan dari Enterprise Data Warehouse dan merupakan sumber data untuk *online analytical.*

Ciri-cirinya antara lain

- Berfokus hanya pada persyaratan satu departemen atau fungsi bisnis.
- Biasanya tidak berisi data operasional yang terperinci tidak seperti gudang data.

- Lebih mudah dipahami dan dinavigasi.
- Alasan Membuat Data Mart
- Untuk memberi pengguna akses ke data yang paling sering mereka butuhkan untuk dianalisis.
- Untuk meningkatkan waktu respons pengguna akhir karena pengurangan volume data yang akan diakses.
- Membangun data mart lebih sederhana dibandingkan dengan membangun gudang data perusahaan.
- Biaya penerapan data mart biasanya kurang dari yang diperlukan untuk membangun gudang data.

g) *Decision Support System*

Merupakan antarmuka untuk online analytical processing engines (OLAP) atau sebagai Business Intelligence.

h) *Cross media Storage Manager*

Merupakan bagian yang menyimpan data historis yang tidak selalu diakses setiap saat. Merupakan backup data bagi Enterprise Datawarehouse.

i) *Metadata Rerepository*

Metadata merupakan data tentang data. Metada digunakan didalam gudang data sebagai obyek gudang data. *Metadata* dibuat untuk nama dan defenisi yang diberikan ke dalam gudang data.

Metada repository seharusnya mengandung beberapa data berikut;

- Deskripsi tentang struktur gudang data termasuk kedalamnya skema warehouse, dimensi, hirarki dan defenisi data serta juga lokasi datamart dan konten.
- *Operational Metadata* yang mengandung data *lineage* (sejarah data migrasi dan alur transformasi terhadap data tersebut),status data aktif, *archived* dan *purged* dan data monitoring *(warehouse usage statistic, error reports dan audit trails)*
- Algoritma yang digunakan untuk peringkasan, yang meliputi ukuran dan dimensi definisi algoritma, data tentang perincian, partisi, area subjek, agregasi,ringkasan, serta kueri dan laporan yang telah

ditentukan sebelumnya.

- Pemetaan dari lingkungan operasional ke gudang data, yang meliputi database sumber dan isinya, deskripsi gateway, partisi data, data ekstraksi, pembersihan, aturan dan default transformasi, penyegaran dan pembersihan data aturan, dan keamanan (otorisasi pengguna dan kontrol akses).
- Data terkait kinerja sistem, yang mencakup indeks dan profil yang ditingkatkan akses data dan kinerja pengambilan, selain aturan untuk waktu dan penjadwalan siklus penyegaran, pembaruan, dan replikasi.
- Metadata bisnis, yang mencakup istilah dan definisi bisnis, kepemilikan data informasi, dan kebijakan pengisian.

Metadata sangat penting dalam gudang data dan memiliki fungsi yang berbeda dengan data yang lain. Contohnya metada membantu dalam membuat business intelligence analyst dalam menemukan konten dari data warehouse dan melakukan transformasi data dari operasional data ke dalam gudang data.

C. Perancangan Gudang Data

Gudang data dapat di implementasikan dalam berbagai pendekatan seperti *top down, bottom-up* dan *mixed.*

1. **Top Down :** top down methodology didasarkan pada desain dari data warehouse dan lebih sistematik. Namun pendekatan ini membutuhkan waktu pengembangan yang lebih lama dan sangat memungkinkan untuk tidak selesai sesuai dengan waktu yang diberikan.
2. **Bottom-up :** Bottom up merupakan pendekatan dengan menggunakan sebuah prototype dan prototype tersebut akan diperbaiki sesuai dengan perubahan dan masukan pada schema yang diberikan. Pendekatan ini lebih cepat dan hasilnya mudah diukur. Kekurangannya dalam visi bagaimana keseluruhan system akan di bangun.
3. **Bottom-up** . Metodologi campuran didasarkan pada desain keseluruhan gudang data, tetapi kemudian dilanjutkan dengan

pendekatan prototyping, secara berurutan menerapkan bagian yang berbeda dari keseluruhan sistem. Pendekatan ini sangat praktis dan biasanya lebih disukai, karena memungkinkan langkah-langkah kecil dan terkontrol diambil dengan mengingat keseluruhan sistem yang akan dibangun.

Langkah-langkah dalam pengembangan gudang data atau data mart dapat berupa diringkas sebagai berikut.

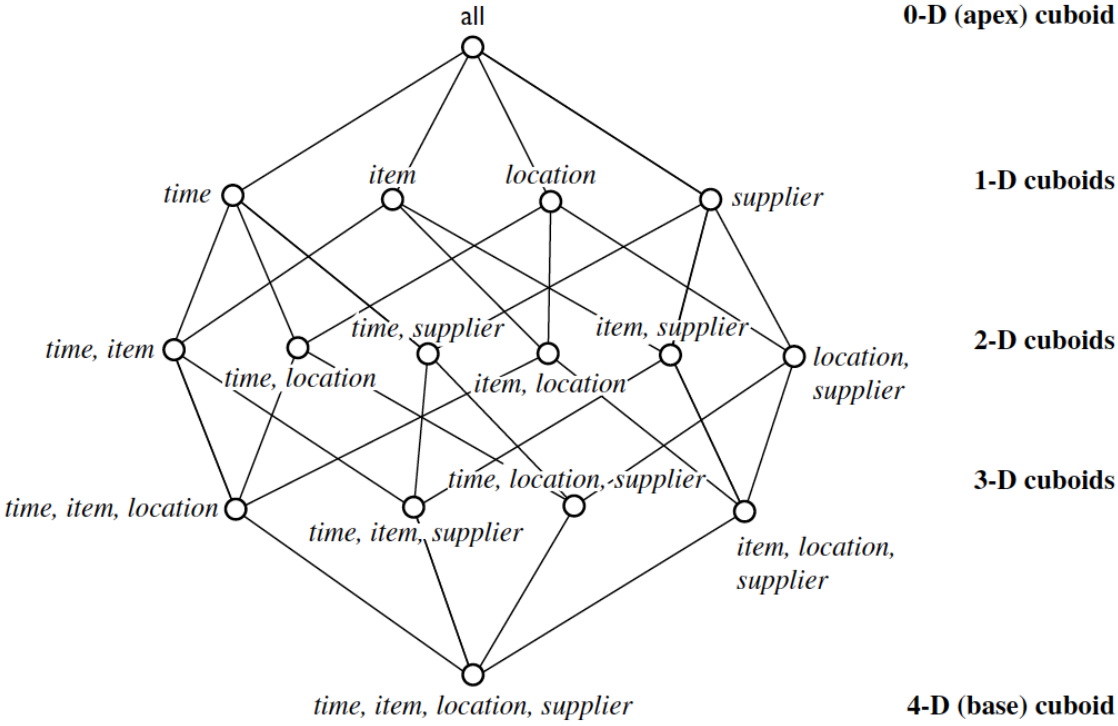
- Satu atau lebih proses dalam organisasi yang akan diwakili dalam gudang data diidentifikasi, seperti penjualan, logistik atau akuntansi.
- Perincian yang sesuai untuk mewakili proses yang dipilih diidentifikasi dan tingkat *atomic* data didefinisikan.
- Langkah-langkah yang relevan untuk diungkapkan dalam tabel fakta untuk multidimension analysis untuk menghasilkan multidimensioanl modelling kemudian dipilih.
- Terakhir, dimensi tabel fakta ditentukan.

D*. Data Cube :* Multidimensi Data Model

Data Cube merupakan salah satu keunggulan yang dimiliki oleh Datawarehouse dibandingkan dengan database operasional yang dimiliki sebelumnya. Data cube memungkinkan data untuk dimodelkan dan dilihat dalam beberapa dimensi dan didefinisikan dengan *dimension* dan *fact*.

Sebuah multidimensi data model biasanya di organisasi dalam satu thema seperti *sales* yang nantinya thema ini akan direpresentasikan dalam sebuah *fact* tabel. Sedangkan ketika berbicara thema *sales* maka akan dibutuhkan dimensi tabel yang terkait dengan waktu,item, cabang dan lokasi. Dimensi ini memungkinkan toko untuk melacak seperti penjualan bulanan *item product,* dan di cabang dan lokasi mana produk ini terjual. Kita akan membahas lebih lanjut tentang dimensi dan fakta tabel di topik berikutnya.

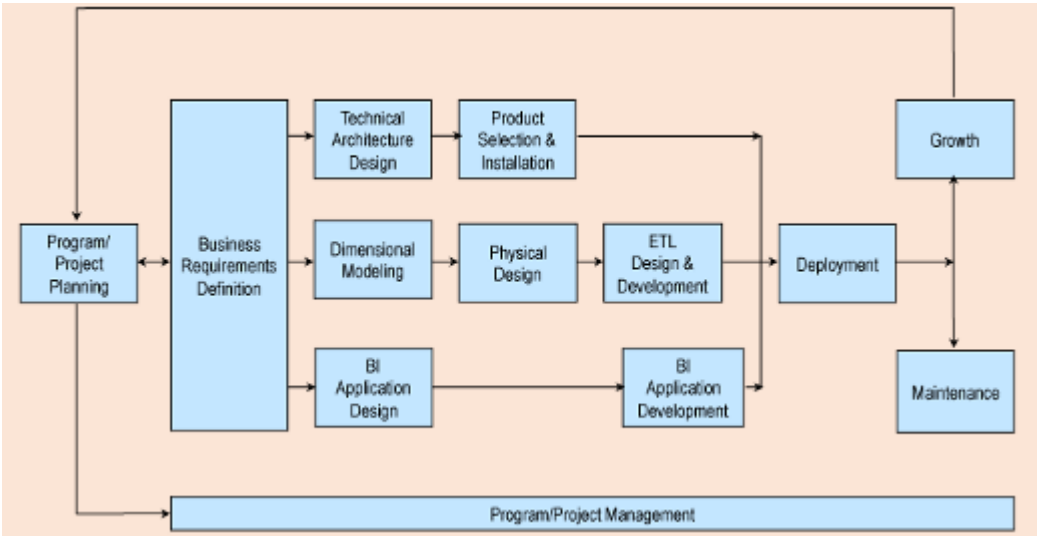
Multidimensi model seringkali direpresentasikan ke dalam 3-D geometric struktur, namun di dalam beberapa kasus dimensional lebih dari 3-D sehingga sering disebut dalam gudang data sebagai *cube* sebaga *n-dimensional.*



Gambar 20. 4-D cube untuk waktu,item,lokasi dan supplier

E. Dimensional Modeling

Pembentukan *Data Marts dan Enterprise Data Warehouse* memerlukan perancangan model yang digunakan nantinya sebagai sebuah penyimpanan dari data yang sudah diintegrasikan. Dalam merancang dimensi dari data yang dibutuhkan biasanya akan merujuk kepada metodologi *Kimball Life Cycle.*



Gambar 21. Kimball Lifecycle

Metodologi *Kimball Life Cycle* merupakan aktivitas pengembangan data warehouse yang dikembangkan oleh Kimball Group dan rekan - rekan mereka di *Metaphor Computer System. Metaphor computer system* adalah sebuah perusahaan pendukung keputusan. Sejak saat itu, metodologi ini telah berhasil digunakan oleh tim proyek *data warehouse dan business intelligence (DW/BI)* di hampir setiap industri, area aplikasi, aktivitas bisnis dan platform teknis.

Salah satu aktivitas perancangan yang terdapat di dalama aktivitas Kimball adalah *DImensional Modelling.*

Dimensional Modelling merupakan teknik yang digunakan untuk menyusun data dengan tujuan sebagai berikut ;

- Memudahkan pengguna yang memahami bisnis untuk mengerti dengan mudah strutur data maupun desain penyimpan tersebut.
- Untuk meningkatkan performansi dari query

Dimensional modelling dapat dibayangkan terdiri dari fact tabel dan dimensional tabel. Lebih lanjut *dimensional modelling* dapat didetail dalam beberapa komponen sebagai berikut ;

1. Fact table : merupakan tabel pada database yang menyimpan perhitungan dari peformansi suatu business prosess. Memiliki *foreign key* ke masing dimension
   Contoh : Waktu, Lokasi, Konsumen dan Product
2. Dimension table : merupakan tabel pada database yang menyimpan konteks dari dari fact table
   Contoh : Waktu, Lokasi, Konsumen dan Product
3. Atribut : Menghubungkan antara fact table dan dimension yang mendefenisikan proses bisnis.
   Contoh : Product: Nama, Kategor, Departement

Aturan dalam membentuk Fact Table

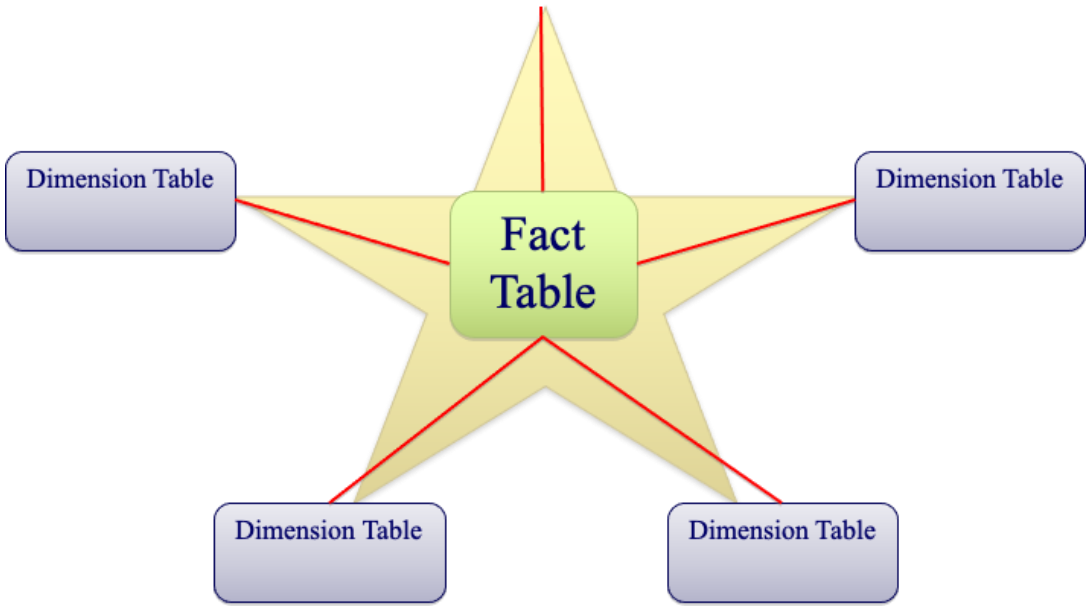
- *Primary key* dalam fact table menggunakan angka paling minimum dan tidak menggunakan *surrogate key.*
- *Referential key* harus. Setiap *foreign key* pada fact table harus memiliki

value

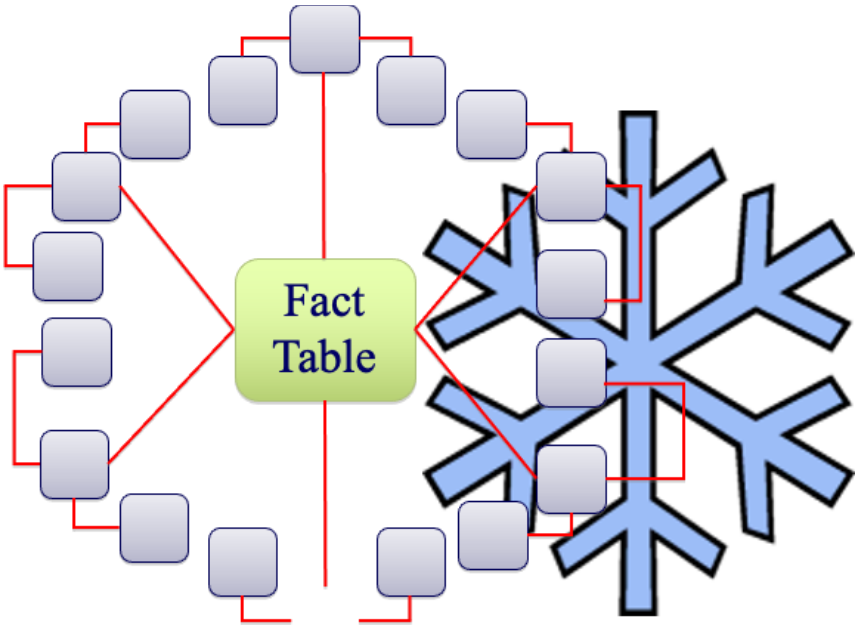
- Tidak boleh memiliki nilai yang NULL
- Perincian tabel fakta Anda harus berada pada butir atom terendah dan terinci yang ditangkap oleh proses bisnis.
- Setiap fakta harus Aditif, atau dirancang ulang menjadi aditif sebanyak mungkin.
- Setiap fakta harus memiliki perincian yang sama.
- Aturan dalam membentuk Dimension Table
- Nilai atribut verbose harus sedeskriptif mungkin.
- Kolom deskriptif – harus mudah untuk mengetahui arti kolom tersebut.
- Lengkap – tidak ada nilai null / kosong di salah satu atribut.
- Dinilai secara terpisah – satu nilai entitas bisnis per baris.
- Kualitas Terjamin – data bersih dan konsisten.
- Harus selalu berisi kunci bisnis, atau warisan PK dari sistem sumber.
- Selalu miliki *surrogate key.* Anda tidak memasukkan ketergantungan pada kunci eksternal.

Gudang data yang dibangun biasa menggunakan data dalam multidimensional dengan beberapa *fact table*. Beberapa modelling yang digunakan dalam data warehouses antara lain ;

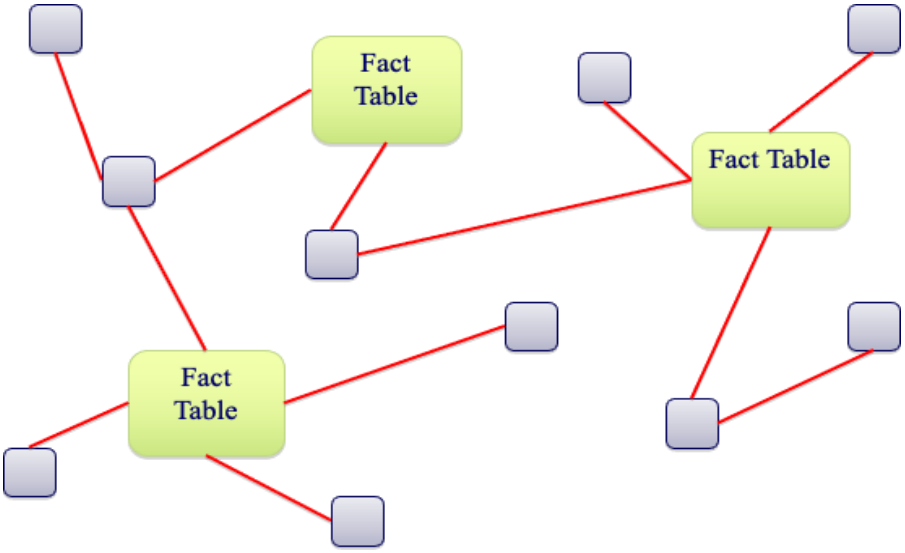
1. *Star schema* : Tabel fakta di tengah terhubung ke sekumpulan tabel dimensi



2. *Snowflake Schema*: Penyempurnaan skema bintang di mana beberapa hierarki dimensi dinormalisasi menjadi satu set tabel dimensi yang lebih kecil, membentuk bentuk yang mirip dengan kepingan salju



3. Fact Constellations : Beberapa tabel fakta berbagi tabel dimensi, dipandang sebagai kumpulan bintang, oleh karena itu disebut skema galaksi atau konstelasi fakta



Berikut langkah yang digunakan untuk mendesain dimensional model antara lain :

- Pilih business proses berdasarkan kebutuhan bisnis.

- Tentukan butir fakta *(fact grain)* tipe rincian
- Identifikasi dimensi dari proses bisnis
- Identifikasi fact dari proses bisnis

Data dimensi jarang berubah tetapi saat perubahan pada diperlukan untuk mengatasi perubahan tersebut. Terdapat 4 strategi paling populer yang sering digunakan dalam mengatasi perubahan pada dimensions atau disebut *slowly changing dimensions*.

- type 1 : Menimpa nilai yang telah ada sebelumnya dengan menggunakan nilai yang baru.
- type 2 : Menambahkan sebuah row dimensi yang baru
- type 3 : Menambahkan attribut dimensi
- Mini dimension : Membuat dimensi baru

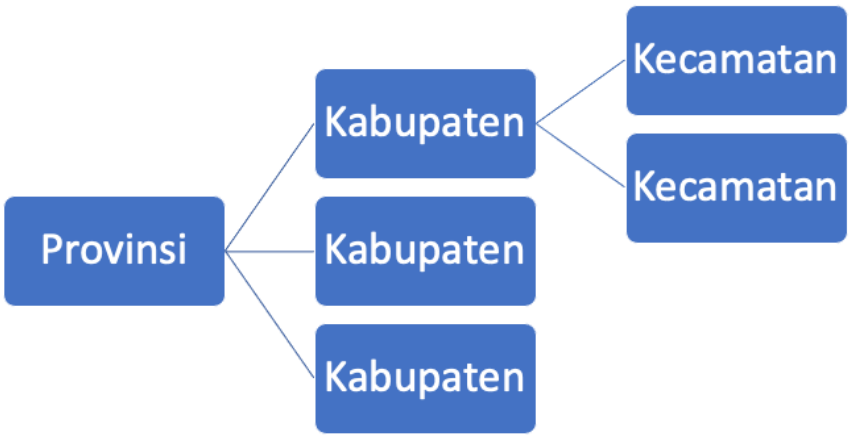
Strategi di atas dapat saling melengkapi atau dikombinasikan.

F. Operasi OLAP (Online Analytical Processing)

Dalam multidimensional model, data diorganisasikan kedalam beberapa dimensi. Setiap dimensi memiliki hirarki tersendiri. Data yang telah disusun ini memudahkan pengguna untuk melihat data dari berbagai persepektif. Terdapat beberapa operasi yang dapat digunakan untuk melihat data dari berbagai posisi. Beberapa operasi yang sering digunakan dalam Online Analytical Processing diantaranya;

1. Rool-up : operasi roll-up atau sering disebut drill-up oleh sebagai pengguna melakukan aggregasi pada cube atau dapat dikatakan mengurangi dimensi atau menanjak pada hirarki suatu data.

   Misalkan kita memiliki dimensi dengan hirarki sebagai berikut;



Operasi roll-up akan akan melakukan agregasi data by *ascending* dari level kecamatan ke level provinsi atau dengan kata lain daripada *grouping* data berdasarkan kecamatan, data akan *digrouping* berdasarkan negara.

2. Drill-down : drill-down merupakan kebalikan dari roll-up dimana ppengguna melakukan aggregasi data dari kurang detail menjadi lebih detail.

   Berlawanan dengan operasi roll-up drill-down akan melakukan grouping dari level provinsi menjadi ke level kecamatan.

   3. Slice and dice : operasi slice atau dice melakukan seleksi ke dalam satu dimensi atau cube yang menghasilkan subcube.

   Misalkan kita memiliki dimensi dengan hirarki sebagai berikut;

4. Pivot : Pivot atau sering disebut *rotate*; merupakan operasi visualisasi yang melakukan rotasi data untuk menyediakan alternatif presentasi data.

G. Rangkuman

Gudang data merupakan pengumpulan data yang berorientasi subjek, terintegrasi, berdasarkan varian waktu serta tidak mudah berubah dalam mendukung proses pengambilan keputusan manajemen (inmon, 1993)

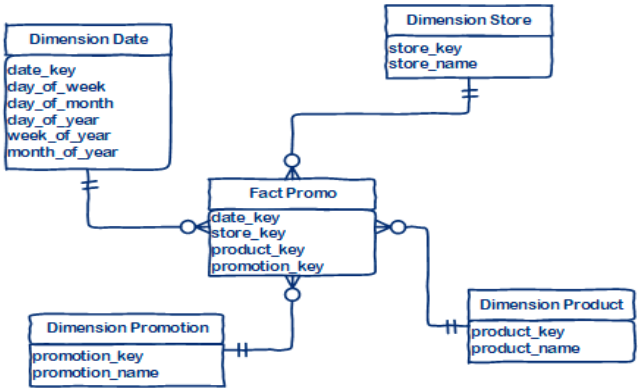
Dalam membangun gudang data terdapat beberapa komponen yang perlu diketahui, berdasarkan Corporate Information Factory (CIF) yang dikembangkan oleh (inmon, 1993). Komponen tersebut dapat kita bagi menjadi beberapa bagian diantaranya *External world & Applications, dan Integrasi dan Transformasi Layer*

Pembentukan *Data Marts dan Enterprise Data Warehouse* memerlukan perancangan model yang digunakan nantinya sebagai sebuah penyimpanan dari data yang sudah diintegrasikan. Dalam merancang dimensi dari data yang dibutuhkan biasanya akan merujuk kepada metodologi *Kimball Life Cycle.*

H. Evaluasi

Untuk menguji pemahaman Anda terkait materi *Datawarehouse*, silahkan kerjakan latihan di bawah ini.

1. Data yang tersimpan dalam *gudang data* didapatkan dari sistem operasional yang sedang berjalan secara reguler, dan tidak dapat diperbaharui oleh pengguna. Dalam kata lain, setelah data disimpan ke dalam *gudang data*, data tidak diperbaharui ataupun dihapus. Ini menunjukan bahwa gudang data menunjukan karakteristik ?
2. Pendekatan apakah yang dilakukan ketika transformasi data terjadi di gudang data dan staged data disimpan di gudang data ?
3. Sebutkan perbedaan antara datamart dan gudang Data.
4. Diketahui dimensional model sebagai berikut, menurut anda apakah bentuk dari skema dimensional model yang ditampilkan ?



DAFTAR PUSTAKA

Peraturan Presiden No. 39 Tahun 2019 tentang Satu Data Indonesia. Jakarta : Republik Indonesia.

Badan Pusat Statistik. (2020). *Petunjuk Teknis Standar Data Statistik*. Jakarta: Badan Pusat Statistik.

DAMA International. (2015). *DAMA-DMBOK data management body of knowledge 2nd edition*. Basking Ridge, USA: Technics Publications LLC.

Design (R. Adams & D. Bevans (Eds.); Fifth Edit). Morgan Kaufmann Publishers is an imprint of Elsevier. https://www.ptonline.com/articles/how-to-get-better-mfi-results

Elmasri, Ramez.Fundamentals of database systems / Ramez Elmasri, Shamkant B. Navathe.—6th ed.p. cm.

Garcia-molina, H., Ullman, J. D., Widom, J., Hall, P., Ullman, J. D., & Ascherman, S. W. (n.d.). *DATABASE SYSTEMS The Complete Book*.

Han, J., Kamber, M., & Pei, J. (2011). Data Mining: Concepts and Techniques (3rd ed.). Morgan Kaufmann.

Loshin, D. (2010). *The practitioner‟s guide to data quality improvement*. Elsevier, Morgan Kaufmann OMG Press.

Morgan Kaufman "Database Modeling and Design (Fifth Edition)"

Normalisasi Database: Pengertian, Tujuan dan Cara Melakukannya. (n.d.). Retrieved December 14, 2022, from https://www.jojonomic.com/blog/normalisasi-database/

Puspitasari, D., Rahmad, C., & Astiningrum, M. (2016). Normalisasi Tabel Pada Basisdata Relasional. *Jurnal Prosiding SENTIA | ISSN: 2085-2347*, *8*(1), 340–345.

REPOSITORY UNIVERSITAS DIAN NUSWANTORO. (n.d.). Retrieved December 14, 2022, from https://repository.dinus.ac.id/

R. Kimball, M. Ross., W. Thornthwaite, J. Mundy, and B. Becker. (2007). The Data Warehouse Lifecycle Toolkit (2nd Edition), Wiley Publishing

Teorey, T., Lighstone, S., Nadeau, T., & Jagadish, H. V. (2011). *Database Modeling and Design* (R. Adams & D. Bevans (Eds.); Fifth Edit). Morgan Kaufmann Publishers is an imprint of Elsevier. https://www.ptonline.com/articles/how-to-get-better-mfi-results

Thomas Connolly and Carolyn Begg. (-). Database Systems: A Practical Approcah to Design. Implementation, and Management 06. Pearson education. USA. ISBN: 978-1-292-06118-4

Vercellis, C. (2009). Business intelligence: Data mining and optimization for decision making. John Wiley & Sons.

http://achmatim.net/2013/05/19/mysql-perintah-query-untuk-mencari-record-yang-tidak-ada-di-tabel-lain/

https://aws.amazon.com/id/relational-database/

https://www.geeksforgeeks.org/difference-between-database-and-data-structure/

https://blog.rumahweb.com/query-adalah/

https://dqlab.id/mengenal-macam-macam-fungsi-join-table-sql-dan-perbedaannya

https://kelasprogrammer.com/2-cara-membuat-database-perpustakaan/

https://www.niagahoster.co.id/blog/apa-itu-struktur-data/#:~:text=Struktur%20data%20adalah%20cara%20menyimpan,kolom%2Dkolom%20dan%20susunan%20tertentu.

https://rangkumanmatkul.wordpress.com/2012/12/12/struktur-data-pada-basis-data/

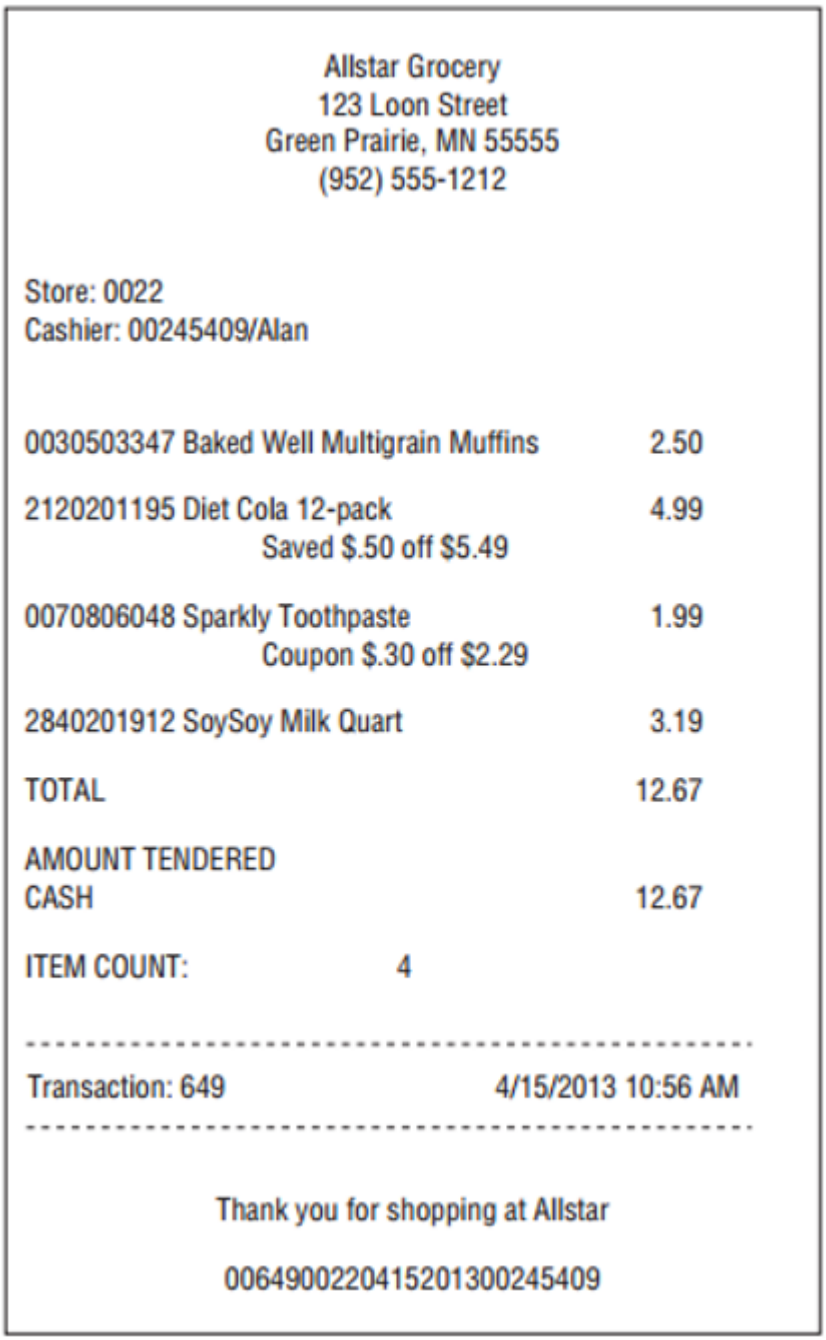
https://www.techtarget.com/searchdatamanagement/definition/data-structure

https://sis.binus.ac.id/2021/07/23/transaction-management-part-1-introduction/

LAMPIRAN-LAMPIRAN

1. Lampiran Praktikum Data Warehouse / Gudang Data

a. Studi Kasus Retail



Allstar Grocery
123 Loon Street
Green Prairie, MN 55555
(952) 555-1212

Store: 0022
Cashier: 00245409/Alan

0030503347 Baked Well Multigrain Muffins	2.50
2120201195 Diet Cola 12-pack	4.99
Saved $.50 off $5.49	
0070806048 Sparkly Toothpaste	1.99
Coupon $.30 off $2.29	
2840201912 SoySoy Milk Quart	3.19
TOTAL	12.67
AMOUNT TENDERED	
CASH	12.67
ITEM COUNT: 4	

Transaction: 649 4/15/2013 10:56 AM

Thank you for shopping at Allstar

00649002204152013002454O9

Berikut ini merupakan tanda terima yang biasa ditemukan saat berbelanja di minimarket maupun supermarket. Sebagai sebuah minimarket dan supermarket manajemen sangat memperhatikan alur logistik dari barang,

stok dan penjualan untuk memaksimalkan keuntungan. Keuntungan yang didapat merupakan harga yang diberikan pada produk dan diskon yang diberikan untuk menarik konsumen untuk lebih banyak berbelanja. Berdasarkan data yang diatas, manajemen meminta untuk dibuatkan model dimensi yang sesuai untuk dapat melihat bagaimana prilaku pembelian konsumen yang dapat dilihat oleh aplikasi *point of sales* supermarket tersebut.

b. Melakukan ETL sederhana dengan menggunakan Python

Pada praktikum ini, kita akan melakukan praktikum sederhana dengan melakukan proses ETL dari sumber data ke sebuah penampungan data. Anda saat ini bekerja sebagai seorang data scientist pada perusahaan XYZ saat ini anda ditugaskan untuk mengumpulkan daftar artikel atau berita dari media yang ada di Indonesia, sehingga anda dapat menjawab pertanyaan dibawah. Untuk itu silahkan ikut langkah - langkah berikut untuk menjawab pertanyaan dibawah.

1. Dapatkan API Key dengan mendaftar pada alamat berikut : https://newsapi.org/.
2. Install docker di komputer anda melalui link berikut https://docs.docker.com/engine/install/
3. Install postgresql pada local machine dengan memanfaatkan docker compose sebagai berikut.

```
version: '3.8'
services:
  db:
    image: postgres:14.1-alpine
    restart: always
    environment:
      - POSTGRES_USER=adi2022
      - POSTGRES_PASSWORD=adi2022
    ports:
      - '5432:5432'
    volumes:
      - db:/var/lib/postgresql/data
volumes:
  db:
    driver: local
```

4. Setelah mendapatkan API Key silahkan download, lengkapi dan jalankan source code di alamat berikut ini : https://github.com/jfadibrin/modul2022/tree/main/M04_BasisData .

Setelah data tersebut terkumpul, anda ditugaskan untuk mengolah data tersebut untuk menjawab beberapa pertanyaan sebagai berikut.

i) Website apakah yang menghasilkan sumber artikel terbanyak ?
ii) Siapakah penulis/ pengarang paling produktif ?
iii) Urutkan jumlah setiap artikel yang diterbitkan berdasarkan tanggal waktu penerbitan dari yang lama sampai yang terbaru.