BASIS DATA LANJUT BAGIAN KEDUA

Pada bagian kedua Basis Data Lanjut ini akan dibahas :
Basis data Terdistribusi, Basis data Internet, ***DSS***, Data ***Warehouse*** dan Data ***Mining***

Basis data Terdistribusi

Pada sistem basis data terdistribusi data di simpan tersebar di beberapa tempat. Setiap tempat penyimpanan dikelola oleh suatu ***DBMS*** yang mandiri. Agar tampilan ***view*** basis data terdistribusi transparan,, maka harus memenuhi dua hal, yaitu independensi data terdistribusi dan atomisitas transaksi terdistribusi.

Dengan independensi data terdistribusi, pengguna dapat melakukan ***query*** secara sederhana tanpa menyebutkan tempat data atau replika data atau fragmen data itu disimpan. Ini memenuhi prinsip independensi data fisik dan data logik atau data logik tidak tergantung data fisik. Lebih jauh lagi proses ***query*** juga harus sudah memperhitungkan biaya antara penyimpanan data fisik melalui komunikasi data atau disimpan sebagai data lokal (replika).

Dengan atomisitas transaksi terdistribusi pengguna harus dapat melakukan transaksi tulis, ***update*** atau akses data terhadap data terdistribusi, seolah-olah data disimpan secara lokal. Efek transaksi terhadap data terdistribusi harus bersifat atomik, yaitu perubahan secara persisten terhadap data remote dan data lokal jika transaksi telah ***commit***, atau tidak terjadi perubahan sama sekali jika transaksi gagal (tidak dapat ***commit***).

Walaupun secara umum kedua hal tersebut harus dipenuhi, tetapi pada situasi jika terjadi lalu lintas data yang padat dan terjadi kelambatan transmisi, maka diperlukan mekanisme khusus untuk menanganinya yang berkaitan dengan overhead administrasi dan performansi ***DBMS***

Jenis basis data terdistribusi

Jika ***DBMS*** yang menangani data untuk semua ***server*** sejenis, maka sistem basis data tersebut disebut sebagai sistem basis data terdistribusi homogen, tetapi sebaliknya jika ***DBMS*** yang menangani data terdistribusi beragam, maka disebut sistem basis data terdistribsi heterogen atau disebut juga sebagai sistem multi basis data.

Kunci keberhasilan membangun sistem yang heterogen adalah pada standarisasi protokol ***gateway***. Protokol gateway adalah API (***application programming interface***) yang memungkinkan ***DBMS*** berfungsi untuk aplikasi eksternal, sebagai contoh seperti : ***ODBC*** dan ***JDBC***. Akses basis data terdistribusi melalui protokol ***gateway*** menjadi mungkin, karena perbedaan format data dan perbedaan antar ***server*** dijembatani.

Arsitektur DBMS terdistribusi

Terdapat tiga alternatif pendekatan untuk membedakan fungsi ***DBMS***, yaitu ***client-server***, kolaborasi ***server*** dan ***middleware***. Sistem ***client-server*** memiliki satu atau banyak proses pada ***client*** dan satu atau banyak proses pada ***client***. ***Client*** berurusan dengan antaramuka dengan user dan ***server*** mengelola data dan eksekusi transaksi.

Arsitektur ini populer, karena relatif sederhana untuk diiplemetasikan karena ada pemisahan fungsi yang jelas dan ***server*** yang tersentralisasi. Biaya yang tinggi hanya untuk satu ***server***, selain pengguna akan lebih nyaman menggunakan antarmuka grafis pada client. Untuk memperlancar layanan ***server*** dan mengurangi ***overhead*** komunikasi, maka diperlukan ***caching*** pada ***server***.

Sistem kolaborasi server

Pada sistem client ***server*** tidak memungkinkan satu ***query*** pada client dijalankan oleh multipel ***server***, karena tidak ada mekanisme kolaborasi antar ***server***. Pada sistem kolaborasi ***server***, ***query*** didekomposisi menjadi sub-***query*** dan disebar ke ***server*** yang berbeda sesuai fungsinya. Secara idealnya dekomposisi sub-***query*** harus memperhatikan biaya komunikasi jaringan dan biaya pengolahan lokal.

Sistem middleware

Sistem middleware dirancang untuk memungkinkan satu ***query*** dijalankan pada multipel ***server*** dengan ***server-server*** basis data tidak mengelola eksekusi pada multi tempat. Untuk koordinasi sub-***query*** dan eksekusi join dilakukan oleh perangkat lunak tersendiri yang disebut middleware

Penyimpanan data pada DBMS terdistribusi

Pada ***DBMS*** terdistribusi relasi disimpan pada beberapa tempat yang berbeda. Akses ke relasi secara remote menimbulkan biaya kirim pesan, oleh karena itu untuk mengurangi biaya satu relasi dipartisi atau dibuat fragmen atau replika yang tersebar pada beberapa tempat sedemikian sehingga penyimpanan dilakukan berdasarkan frekuensi penggunaan lokal, agar biaya overhead komunikasi berkurang.

Fragmentasi

Framentasi dari data dalam bentuk fragmen data yang disimpan pada tempat yang berbeda ada dua kemungkinan, yaitu fragmentasi horizontal dan vertikal. Fragmen vertikal terdiri dari fragmen atribut untuk semua rekord, sedangkan fragmen horizontal terdiri dari fragmen rekord untuk semua atribut. Fragmen horizontal tergantung pada frekuensi akses data berdasar salah satu atribut dengan nilai tertentu, sedangkan fragmen vertikal dibuat berdasarkan jenis atribut tertentu.

Contoh fragmentasi vertikal

Tid	pid	nama	kota	umur	saldo
T1	23412	Joni	Bogor	18	4300
T2	43252	Susi	Jakarta	18	3450
T3	43211	Susi	Jakarta	19	4220
T4	36123	Maya	Bandung	21	5340
T5	36056	Gani	Bandung	22	2500

Fragmen horizontal

Fragmen vertikal

Replikasi

Replika adalah hasil replikasi satu relasi data atau fragmen relasi yang dapat disimpan pada lebih dari satu tempat, jumlah replika fragmen relasi tidak harus ama untuk satu relasi. Contoh jika relasi R dijadikan tiga fragmen R1, R2, R3, mungkin R1 tidak dibuat replikanya, tetapi R2 dibuat replika di satu tempat lain dan R3 dibuat replika di semua tempat.

Tujuan replikasi ada dua motivasi, yaitu meningkatkan availabilitas data dan mempercepat evaluasi ***query*** jika ada replika fragmen atau satu relasi pada tempat lokal. Ada dua jenis replikasi, yaitu replika sinkron dan asinkron jika dikaitkan dengan proses pemutakhiran (***update***) antara replika dan data aslinya.

Replikasi sinkron

Ada dua teknik dasar untuk menjamin transaksi menghasilkan satu hasil dan tidak bergantung pada akses terhadap data atau replika data yang digunakan dalam perhitungan transaksi.

Teknik pertama disebut voting. Transaksi harus menulis mayoritas data dan replikanya dan membaca minimal satu replika yang dianggap paling mutakhir. Contoh jika ada 10 replika data dan 7 replika ditulis oleh transaksi ***update***, maka 4 data lainnya juga harus ditulis. Setiap replika mempunyai nomor versi. Replika dengan nomor versi tertinggi dianggap paling mutakhir. Teknik ini kurang menarik, karena akan terjadi banyak proses baca, padahal proses baca sangat diperlukan pada transaksi berikutnya.

Teknik kedua disebut ***read-any write-all***, artinya untuk proses baca cukup melibatkan satu replika, tetapi ketika proses tulis harus melibatkan semua replika. Proses baca dapat dilakukan dengan cepat apalagi baca data lokal, tetapi proses tulis lebih lama. Teknik ini lebih populer, karena proses baca lebih sering dibutuhkan dibandingkan proses tulis.

Replikasi asinkron

Replikasi sinkron memerlukan biaya lebih tinggi dibanding asinkron, karena selama transaksi ***update*** belum ***commit***, maka semua replika harus di kunci secara eksklusif. Untuk teknik ***read-any write-all***, maka jika ada kelambatan atau kegagalan komunikasi, maka transaksi tidak bisa ***commit*** karena harus menunggu sampai semua replikasi di tulis, sehingga replikasi sinkron kurang realistis

Sebaliknya replikasi asinkron lebih realistis, walaupun melanggar prinsip independensi data terdistribusi selama interval waktu tertentu sampai dilakukan transaksi ***update*** secara berkala. Jadi pengguna harus berhati-hati dan harus dapat mengenali replika yang paling mutakhir. Tentu saja replikasi asinkron tidak cocok untuk aplikasi yang ***real time*** (waktu nyata)

Pada replikasi asinkron terdapat dua pilihan, yaitu replikasi asinkron situs primer dan replikasi asinkron peer-to-peer. Replikasi asinkron situs primer memiliki satu replika yang dianggap sebagai master atau data primer. Replika lainnya disebut replika sekunder. Tidak seperti replika primer, replika sekunder tidak dapat di-***update*** langsung. Mekanisme pemilihan replika primer dan sekunder melalui mekanisme pendaftaran oleh pengguna dan penentuan relasi di situs tertentu yang dijadikan replika primer dan replika lainnya harus mengacunya.

Pada replikasi asinkron peer-to-peer, beberapa replika bisa di-***update*** (mungkin tidak semua) dan dijadikan replika master. Dalam hal terjadi konflik, karena masalah keterlambatan propagasi, maka harus diterapkan salah satu strategi penanganan konflik. Secara umum konflik biasanya dapat diselesaikan, bahkan lebih sering tidak terjadi konflik, sehingga jenis replikasi ini banyak digunakan. Salah satu strategi pencegah konflik adalah waktu proses transaksi ***update*** tidak bersamaan dan pada satu saat hanya dilakukan terhadap salah satu replika (yang lain tidak dapat di-***update***), kemudian perubahan terhadap replika master itu dipropagasi ke replika yang lain. Jika ada kegagalan ***update*** terhadap salah satu replika, maka diambil alih oleh salah satu replika yang lain yang biasa dijadikan sebagai ***backup***.

Transaksi terdistribusi

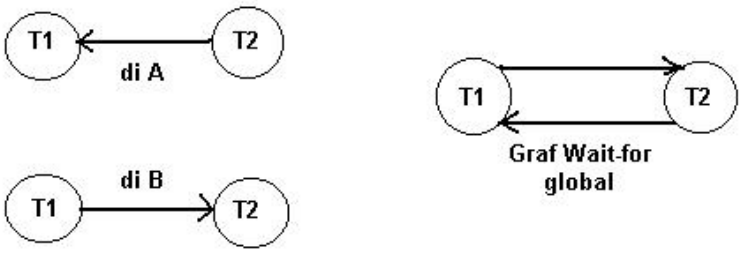
Pada sistem terdistribusi transaksi dapat dilakukan pada suatu tempat tetapi dapat akses data di tempat lain. Setiap transaksi dipecah menjadi beberapa sub-transaksi yang dijalankan secara tersebar melalui manajer transaksi pada setiap tempat sub-transaksi dijalankan untuk dikoordinasikan. Untuk kasus kontrol proses yang terjadi bersamaan (konkuren), maka ada mekanisme penguncian objek yang digunakan yang ada di tempat lain, juga bagaimana cara mendeteksi jika terjadi ***deadlock***

Kontrol konkurensi terdistribusi

Pengelolaan penguncian objek yang terdistribusi dapat dilakukan dengan beberapa cara, yaitu secara terpusat (sentralisasi), replika primer dan terdistribusi penuh. Pada cara terpusat, penanganan penguncian dilakukan dari satu tempat. Pada cara replika primer, penanganan penguncian dilakukan pada tempat replika primer berada. Dan pada cara terdistribusi penuh, maka penanganan penguncian dilakukan pada tempat replika yang akan dikunci. Cara ketiga ini lebih banyak digunakan.

Deadlock terdistribusi

Selain skema pencegahan ***deadlock***, terdapat skema deteksi ***deadlock*** yang lebih banyak digunakan pada cara selain terpusat. Untuk dapat mendeteksi terjadinya ***deadlock***, maka pada tempat lokal terdapat informasi graf wait-for. Ada tiga jenis algoritma deteksi ***deadlock***, yaitu algoritma terpusat, algoritma hirarki dan algoritma sederhana



DEADLOCK TERDISTRIBUSI

Pemulihan terdistribusi

Pemulihan terhadap kegagalan pada sistem terdistribusi lebih rumit dibandingkan dengan pada sistem terpusat, karena jenis kegagalan pada sistem terdistribusi lebih beragam dan karena transaksi di bagi menjadi beberapa sub-transaksi, maka untuk penanganan ***commit*** diperlukan bantuan protokol ***commit***. Protokol ***commit*** yang paling banyak digunakan adalah protokol ***commit*** dua fasa (2PC).

.

Eksekusi normal dan protokol commit

Ketika eksekusi ***commit*** akan dilakukan, maka perintah ***commit*** dikirim kepada koordinator transaksi. Koordinator mengirimkan pesan ke sub-ordinat. Sub-ordinat kemudian memutuskan apakah akan abort atau ***commit*** sub-transaksi, kemudian mengirimkan kembali jawaban (no atau yes). Hanya jika koordiantor menerima jawaban yes dari semua sub-ordinat, maka ***commit*** dilakukan dan mengirimkan pesan ***commit*** ke semua sub-ordinat, jika tidak demikian maka abort dilakukan dan mengirimkan pesan abort ke semua sub-ordinat. Sub-ordinat akan memberi pesan ***commit*** atau abort ke koordinator setelah sub-ordinat melakukan ***commit*** atau abort pada sub-transaksi.

Terakhir koordinator setelah menerima pesan dari semua sub-ordinat, maka menuliskan kejadian transaksi pada ***log*** .

Restart setelah kegagalan

Jika terjadi crash kemudian kembali normal, maka proses pemulihan dilakukan dengan membaca log, jika pada log tercatat ***commit*** maka semua lakukan ***commit*** pada transaksi yang belum ***commit***. Selain protokol ***commit*** dua fasa, juga dikenal protokol ***commit*** tiga fasa.

Acuan :

Ramakrishnan, Raghu; Gehrke, Johannes; “Database Management Systems”; McGraw-Hill, edisi-2

Basis Data Internet

Dengan adanya teknologi internet, memungkinkan pengguna dari berbagai tempat terpisah mengakses data suatu organisasi secara remote. Mulai dari data berupa file teks sampai basis data, mulai dari program aplikasi browser ***Web*** atau aplikasi client sampai aplikasi ***server*** yang dijalankan di tempat organisasi, mulai dari format teks sederhana seperti ***HTML*** sampai format basis data yang dapat dikirim melalui internet, mulai dari akses informasi sampai transaksi bisnis melalui Internet (e-***Commerce***).

Web

Dengan ***World Wide Web*** (***WWW***) atau disingkat menjadi ***Web*** memungkinkan file berformat ***HTML*** (***hypertext markup language***) dapat ditampilkan atau dapat dilakukan transfer file ke berbagai tempat yang mengakses internet melalui browser ***Web*** seperti ***Internet Explorer*** dan sarana penghubung (***link***) yang diberi nama ***URL*** (universal resource locator) dengan menggunakan protokol ***HTTP*** (***hypertext transport protocol***) atau ***FTP*** (***file transport protocol***).

HTML

Dengan format ***HTML*** file teks dapat ditampilkan di layar melalui browser ***Web***, karena adanya tambahan mark berbentuk ***tag*** yang terdiri dari ***tag*** awal dan ***tag*** akhir dalam bentuk <***TAG***> dan </***TAG***>. Secara umum format ***HTML*** terdiri dari dua bagian yaitu ***header*** dan body yang dipisahkan dengan ***tag header*** (<***HEADER***>,</***HEADER***>) dan body (<***BODY>,</BODY***>) di dalam ***tag HTML*** (<***HTML***>,</***HTML***>), seperti contoh di bawah ini:

<u>Contoh halaman statik yang menampilkan data teks:</u>

```
<HTML>
 <HEAD>
 <TITLE>
  References
 </TITLE>
 </HEAD>
 <BODY>
  XML:
  <UL>
   <LI>Author  :Elliotte Rusty Harold</LI>
   <LI>Title   :XML in a Nutshell</LI>
   <LI>Publisher:O'Reilly</LI>
  </UL>
  Database:
```

```
 <UL>
  <LI>Author   :Raghu Ramakrishnan</LI>
  <LI>Title    :Database Management Systems</LI>
  <LI>Publisher:McGraw-Hill</LI>
 </UL>
 Data Warehouse:
 <UL>
  <LI>Author   :W.H Inmon</LI>
  <LI>Title    :Building The Data Warehouse</LI>
  <LI>Publisher:John-Wiley & Sons</LI>
 </UL>
</BODY>
</HTML>
```

Tampilan file ***HTML*** di atas pada browser ***Web*** seperti :

XML:

- Author :Elliotte Rusty Harold
- Title :***XML*** in a Nutshell
- Publisher:O'Reilly

Database:

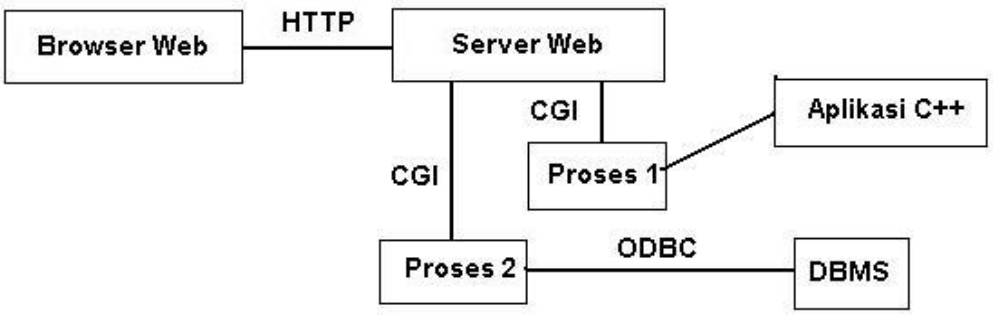
- Author :Raghu Ramakrishnan
- Title :Database Management Systems
- Publisher:McGraw-Hill

Data ***Warehouse***:

- Author :W.H Inmon
- Title :Building The Data ***Warehouse***
- Publisher:John-Wiley & Sons

Basis data dan Web

Kebutuhan pengguna melalui ***Web*** tidak hanya untuk akses file teks atau transfer file, tetapi berkembang juga untuk transaksi bisnis melalui tampilan form yang dapat diisi data dan form itu dikirim dari browser melalui jaringan Internet, sampai diterima di tempat perusahaan yang mempunyai situs ***Web***, kemudian diolah dan dikirim kembali melalui protokol ***CGI*** (***common gateway interface***) dan menampilkan hasilnya pada browser masing-masing pengguna yang berbeda-beda (terdistribusi) secara dinamik



Struktur Proses dengan CGI

Contoh halaman dinamik berbentuk form dalam format ***HTML*** :

```
<HTML>
 <HEAD>
 <TITLE>
  Form
 </TITLE>
 </HEAD>
 <BODY>
  <FORM action="Send.cgi" method=post>
   Type your name:
   <INPUT type="text" name="userName" size=30>
   <INPUT type="submit" value="Send">
   <INPUT type="reset" value="Clear">
  </FORM>
 </BODY>
</HTML>
```

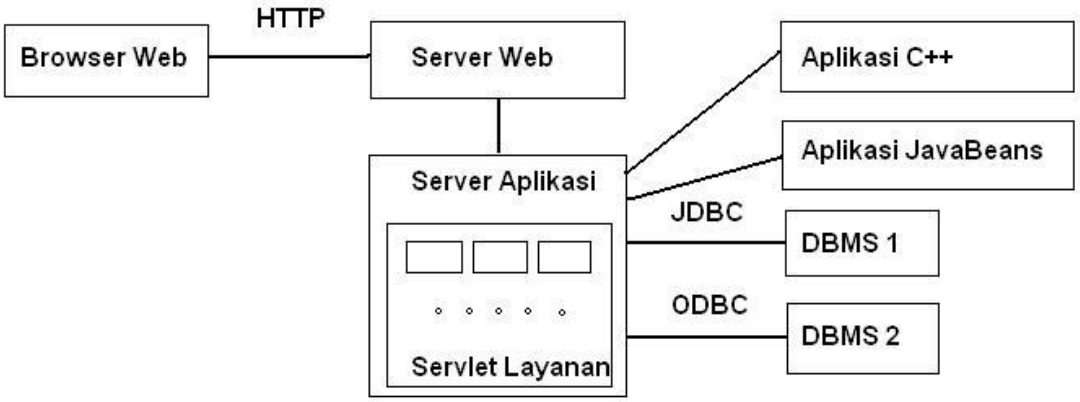
Tampilan pada browser:

Type your name: [ ] Send Clear

Pengolahan data base di tempat perusahaan membutuhkan peran ***DBMS*** dan protokol ***CGI***, tetapi untuk setiap halaman dinamik yang ditampilkan diperlukan satu proses tersendiri dan jika jumlah pengguna yang mengakses dan melakukan transaksi banyak, maka akan mempengaruhi performansi sistem.

Untuk itu maka diperlukan aplikasi ***server*** yang menangani proses-proses transaksi menjadi satu proses dan membaginya dalam beberapa thread. Aplikasi ini dapat dipisahkan dari ***server*** dan disebut sebagai middle tier. Antara aplikasi ***server*** dan ***DBMS*** diperlukan konektor berupa driver seperti ***JDBC*** untuk aplikasi Java atau ***ODBC*** untuk aplikasi lainnya.

Untuk aplikasi ***server*** dibutuhkan integrasi sumber data yang heterogen, transaksi yang melibatkan beberapa sumber data, keamanan atau sekuritas dan manajemen sesi. Untuk integrasi diperlukan sistem objek relasional. Untuk teransaksi terdistribusi diperlukan constraint atomisitas transaksi, konsistensi isi data, isolasi proses dan durabilitas. Untuk sekuritas diperlukan protokol yang aman seperti ***SSL*** (***secure socket layer***). Untuk manajemen sesi diperlukan identifikator sesi seperti ***cookies***



ARSITEKTUR SERVER APLIKASI

XML

Format ***HTML*** hanya dapat menampilkan data dalam bentuk teks dan tidak dapat merepresentasikan struktur data yang rumit karena ***tag*** pada format ***HTML*** bersifat statik. ***HTML*** dapat merepresentasikan ***view*** berorientasi dokumen, padahal untuk basis data diperlukan representasi ***view*** berorientasi skema. Untuk tujuan itu, maka dibuat format ***XML*** yang merupakan perpaduan ***HTML*** dengan ***SGML*** (***standard generalized markup language***). Dengan ***SGML*** memungkin mendefinisikan bahasa markup untuk data dan dokumen yang dapat digunakan untuk semua kebutuhan yang umum, sehingga menjadi relatif terlalu kompleks.

XML terdiri dari elemen, atribut, acuan entitas, komentar dan ***DTD*** (document type declaration). Elemen atau ***tag*** yang dimulai dengan ***tag*** awal dan diakhiri dengan ***tag*** akhir adalah unit atau blok pembangun utama dokumen ***XML***. Setiap elemen dapat bersarang di dalam elemen lain. Setiap elemen dapat teridiri dari beberapa atribut yang nilainya di-set pada awal ***tag***. Acuan entitas adalah shortcut teks atau simbol yang dimulai dengan '&' dan diakhiri dengan ';'. Ada 5 karakter &, >, <, ", ' yang juga menggunakan shortcut, yaitu amp, gt, lt, quot dan apos. Komentar dimulai dengan '<!-' dan diakhiri dengan '->'. ***DTD*** adalah aturan definisi atau tata bahasa penentu elemen, atribut dan entitas.

DTD XML

DTD ditandai dengan <!DOCTYPE nama_elemen_root [deklarasi DTD]>.
Elemen ditandai dengan <!ELEMENT nama_elemen (jenis_isi_elemen)>. Pada akhir ***tag*** setiap elemen terdapat tiga jenis karakter pilihan, yaitu karakter '*' untuk jumlah elemen nol atau banyak, karakter '+' untuk jumlah elemen satu atau banyak, atau karakter '?' untuk jumlah elemen nol atau satu. Setiap elemen yang mempunyai elemen lagi secara bersarang menggunakan tanda kurung: '(' dan ')' dengan nama elemen di dalamnya yang dipisah ',' jika lebih dari satu. Jika suatu elemen merupakan elemen primitif maka isi di dalam kurung diisi dengan simbol khusus: #PCDATA. Jika elemen tidak mempunyai isi gunakan simbol EMPTY. Atribut elemen di tandai dengan <!ATTLIST nama_atribut (nilai_atribut) nilai_default>. Jika nilai atribut harus ada isinya (tidak null) maka tambahkan simbol khusus: #REQUIRED sebelum ***tag*** akhir: '>'.

Contoh file ***XML*** :

```
<?XML version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE BOOKLIST SYSTEM "Book.DTD">
<BOOKLIST>
 <BOOK genre="DB" format="Hardcover">
  <AUTHOR>
   <FIRSTNAME>Raghu
   </FIRSTNAME>
   <LASTNAME>Ramakrishnan
   </LASTNAME>
  </AUTHOR>
  <TITLE>Database Management Systems
  </TITLE>
  <PUBLISHER>McGraw-Hill
  </PUBLISHER>
 </BOOK>
 <BOOK genre="DW" format="Paperback">
```

```
  <AUTHOR>
   <FIRSTNAME>W.H.
   </FIRSTNAME>
   <LASTNAME>Inmon
   </LASTNAME>
  </AUTHOR>
  <TITLE>Building The Data Warehouse
  </TITLE>
  <PUBLISHER>John-Wiley &amp;amp; Sons
  </PUBLISHER>
 </BOOK>
</BOOKLIST>
```

Contoh file ***DTD*** :

```
<!DOCTYPE BOOKLIST [
 <!ELEMENT BOOKLIST (BOOK)*>
 <!ELEMENT BOOK (AUTHOR,TITLE,PUBLISHER?)>
  <!ELEMENT AUTHOR (FIRSTNAME,LASTNAME)>
   <!ELEMENT FIRSTNAME (#PCDATA)>
   <!ELEMENT LASTNAME (#PCDATA)>
  <!ELEMENT TITLE (#PCDATA)>
  <!ELEMENT PUBLISHER (#PCDATA)>
 <!ATTLIST BOOK genre (DB | DW) #REQUIRED>
 <!ATTLIST BOOK format (Paperback | Hardcover) "Paperback">
]>
```

Query data XML

Pengelolaan data ***XML*** yang didefinisikan melalui dokumen ***XML*** memungkinkan aplikasi lebih mendekati pada pengelolaan basis data dari pada pengelolaan berorientasi teks yang didefinisikan melalui dokumen ***HTML*** dengan tetap memudahkan juga untuk ditampilkan pada browser ***Web***. Tidak semua jenis browser ***Web*** kompatibel dengan format ***XML***. ***Browser Netscape*** dan ***Mozila*** versi 6 atau yang lebih tinggi sudah mengakomodasi format ***XML***.

Untuk memudahkan dalam pengelolaan data yang terstruktur melalui Internet diperlukan bahasa ***query*** yang dapat mempertukarkan data dengan definisi struktur yang berbeda berformat ***XML*** yang ada pada file ***DTD***. Ada beberapa bahasa ***query XML*** (***XML-QL***) yang sesuai standar ***W3C*** seperti yang ada pada situs ***HTTP***://www.w3.org/***XML***/***Query***

Pada spesifikasi standar ***W3C*** yang ditetapkan terdiri dari empat spesifikasi yang saling bergantung secara hirarki mulai dari spesifikasi kebutuhan, model data, aljabar dan sintak. Bahasa ***query*** didefinisikan pada spesifikasi model data yang definisi operasinya ditentukan pada spesifikasi aljabar dan bagaimana tata tulis (sintak) ***query*** didefinisikan pada spesifikasi sintak.

Contoh sintak salah satu jenis ***XML***-QL (bagian klausa ***WHERE***):

Contoh ***query***-1:

```
WHERE <BOOK>
        <NAME>
         <LASTNAME>$ln</LASTNAME>
         <FIRSTNAME>$fn</FIRSTNAME>
        </NAME>
        <PUBLISHER>McGraw-Hill</PUBLISHER>
       </BOOK> IN www.amazon.com/books.XML
CONSTRUCT <RESULTNAME>
              <FIRST>$fn</FIRST>
              <LAST>$ln</LAST>
          </RESULTNAME>
```

Contoh ***query***-2

```
WHERE <BOOK>$b<BOOK> IN www.amazon.com/books.XML ,
        <AUTHOR>$a</AUTHOR>,
        <PUBLISHER>$p</PUBLISHER> IN $b
CONSTRUCT <RESULT><PUBLISHER>$p</PUBLISHER>
       WHERE <LASTNAME>$l</LASTNAME> IN $a
        CONSTRUCT <LASTNAME>$l</LASTNAME>
       </RESULT>
```

Hasil ***query***-1:

```
<RESULTNAME><FIRST>Raghu</FIRST><LAST>Ramakrishnan</LAST></R
ESULTNAME>
```

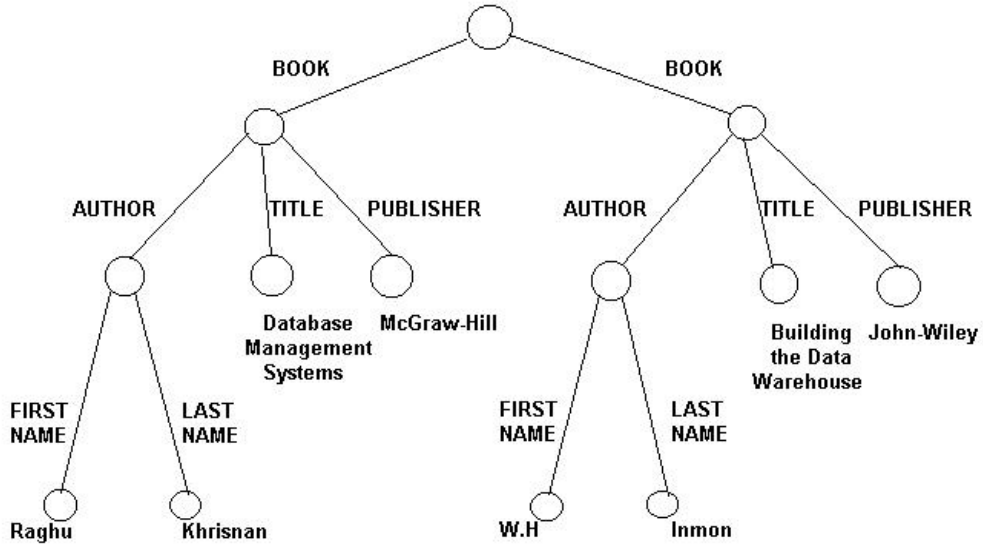
Hasil ***query***-2:

```
<RESULT> <PUBLISHER>MacGraw-Hill</PUBLISHER>
 <LASTNAME>Ramarishnan</LASTNAME>
<RESULT> <PUBLISHER>John-Wiley &amp;amp; Sons</PUBLISHER>
 <LASTNAME>Inmon</LASTNAME>
```

Model data semi terstruktur

Dengan adanya data yang terstruktur dan data tidak terstruktur, maka skema struktur data harus diketahui agar dapat diintegrasikan, jika tidak maka akan ada masalah dalam mengintegrasikan struktur data yang heterogen dan dalam ***query*** data, selain data terstruktur mempengaruhi performansi pertukaran data karena harus ada proses transformasi struktur data.

Kebutuhan untuk penyelesaian masalah tersebut diatas memicu diciptakannya pemodelan data semi terstruktur yang lebih fleksibel, yaitu menggunakan representasi graf berarah berbentuk pohon semantik. Setiap simpul menyatakan satu objek primitif atau objek komposit dan setiap garis penghubung simpul menyatakan atribut.



MODEL DATA SEMI TERSTRUKTUR

Salah satu model data semi terstruktur adalah ***OEM*** (***object exchange model***) yang merepresentasikan setiap objek dalam bentuk tripel, yaitu label, tipe dan nilai objek. Pada implementasinya juga ada identifikator objek yang bernilai unit. Label pada tripel yang pertama harus dipilih sedemikian sehingga informatif.

Contoh-1 : <lastName, string, “Ramakhrisnan”>

Contoh-2 :
<authorName, set, {firstname1, lastname1}>
fistname1 is <firstName, string, “Raghu”>
lastname1 is <lastName, string, “Ramakhrisnan”>

Contoh-3 :

<bookList, set, {book1,book2,book3}>
book1 is <book, set, {author1,title1,publisher1}>
book2 is <book, set, {author2,title2,publisher2}>
book3 is <book, set, {author3,title3,publisher3}>
author1 is <author, set, {firstname1,lastname1}>
title1 is <title, set, “Database Management Systems”>
publisher1 is <publisher, string, “McGraw-Hill”>
fistname1 is <firstName, string, “Raghu”>
lastname1 is <lastName, string, “Ramakhrisnan”>

Acuan :

Ramakrishnan, Raghu; Gehrke, Johannes; “Database Management Systems”; McGraw-Hill, edisi-2

DSS

Sistem Pendukung Keputusan atau ***Decision Support System*** (***DSS***) adalah aplikasi-aplikasi yang dapat mendukung pembuatan keputusan di tingkat atas berdasarkan rangkuman data historis sampai data saat ini dan identifikasi kecenderungan (trend) melalui ekstraksi data. Kelompok aplikasi kategori ***DSS*** ini terus berkembang yang dipicu oleh teknologi pengolahan analitik secara online sesuai kriteria yang ditentukan pengguna, seperti ***OLTP, OLAP*** dan Data ***Warehouse*** yang didukung oleh teknologi Data ***Mining***.

Dengan pesatnya pertumbuhan teknologi ***PC*** dan teknologi ***4GL***, menyebabkan adanya kebutuhan pemisahan basis data operasional dengan basis data informasional, dengan alasan :

- Adanya kebutuhan analisis untuk memberikan informasi yang secara fisik berbeda dengan data pada tingkat operasional
- Teknologi pendukung pengolahan tingkat operasional berbeda dengan teknologi untuk pendukung kebutuhan analisis data atau pendukung kebutuhan informasi
- Komunitas pengguna data operasional berbeda dengan pengguna terhadap data yang telah dianalisis
- Karakteristik pengolahan pada lingkungan operasional sangat berbeda dengan pengolahan pada lingkungan analisis, dsb.

Jadi diperlukan adanya arsitektur baru untuk penyimpanan data yang berbeda dengan basis data yang dapat menunjang aplikasi yang mengandung proses analisis data dan dapat menunjang sistem pengambilan keputusan atau ***Decision Support System*** (***DSS***). Arsitektur baru itu disebut Data ***Warehouse*** (***DW***) yang merupakan jantung dari ***DSS***, atau ***DW*** merupakan bagian utama ***DSS***. ***DW*** adalah ***DSS***, tetapi ***DSS*** tidak hanya ***DW***.

Ciri-ciri DW

DW dibangun dengan metodologi pengembangan yang berbeda dengan aplikasi biasa, ***DW*** secara fundamental sangat berbeda dengan data mart, ***DW*** sedang pada fase perkembangan yang pesat, karena memberikan manfaat, ***DW*** membutuhkan dan dibangun dari berbagai jenis dan jumlah data yang sangat besar. Manfaat utama yang dibutuhan pengguna ***DW*** di kalangan bisnis adalah memungkinkan organisasi mengurangi biaya untuk memperoleh informasi dengan cepat.

Evolusi DSS

Pada awal tahun 1970-an berkembang teknologi penyimpanan data yang dapat diakses secara langsung yang disebut ***DASD*** (direct access storage divice) menggantikan teknologi penyimpan data sekuensial seperti tape.

Kemudian setelah itu berkembang juga perangkat lunak sistem untuk mengelola basis data yang disebut ***DBMS*** (database management system) yang dapat mepermudah pengguna mengakses atau menyimpan dan mengambil kembali data pada ***DASD***, termasuk membuat indeks data. Seiring dengan itu muncul teknologi basis data relasional.

Setelah itu pada pertengahan tahun 1970-an berkembang teknologi pengolahan transaksi secara online (***OLTP***) yang dapat mempercepat akses data dan memicu aplikasi sistem reservasi, sistem teler bank dan sistem kontrol manukatur dll.

Pada awal tahun 1980-an muncul teknologi baru di bidang perangkat keras seperti ***PC*** (***microchip***) dan perangkat lunak seperti bahasa pemrograman generasi ke-4 (***4GL***). Dengan adanya ***PC*** dan ***4GL***, maka tidak hanya pengolahan transaksi online yang dapat dilakukan tetapi juga pengolahan analisis data pendukung sistem informasi dapat dilakukan sehingga dapat dipakai untuk keputusan operasional

Arsitektur ***DSS*** berevolusi untuk menghadapi tantangan seperti masalah kredibilitas data, produktifitas dan masalah transformasi data menjadi informasi. Contoh masalah kredibilitas data pada suatu organisasi yang mendapatkan kesimpulan yang berbeda yang diambil oleh departemen yang berbeda untuk suatu topik yang sama, hal ini bisa terjadi karena basis data tidak berdasarkan waktu yang sama, pendekatan algoritma dan tingkat ekstraksi data yang berbeda, eksternal data yang berbeda dan bahkan sumber data umum sejak dari awal yang sudah berbeda ..

Masalah produktifitas atau kinerja organisasi sangat dipengaruhi oleh kendala teknologi yang heterogen dan adanya teknologi baru yang harus dikuasai oleh SDM yang ada atau harus mencari SDM yang lebih baik diperlukan waktu adaptasi. Secara teknis ada kendala definisi data yang mempunyai kesamaan semantik diberi label yang berbeda, sehingga diperlukan waktu yang lebih lama untuk melakukan analisis., selain juga masalah waktu yang dibutuhkan untuk kompilasi data dan menulis program untuk dapat menghasilkan resume data dalam bentuk laporan

Masalah transformasi data menjadi informasi disebabkan karena masalah kesulitan dalam mngintegrasikan data yang heterogen dan kurangnya informasi historis data. Secara mendasar terdapat dua jenis data, yaitu data primitif dan data turunan. Data primitif diperoleh dari data operasional, sedangkan data turunan adalah hasil rangkuman atau hasil kalkulasi yang berasal dari data primitif untuk keperluan manajemen organisasi.

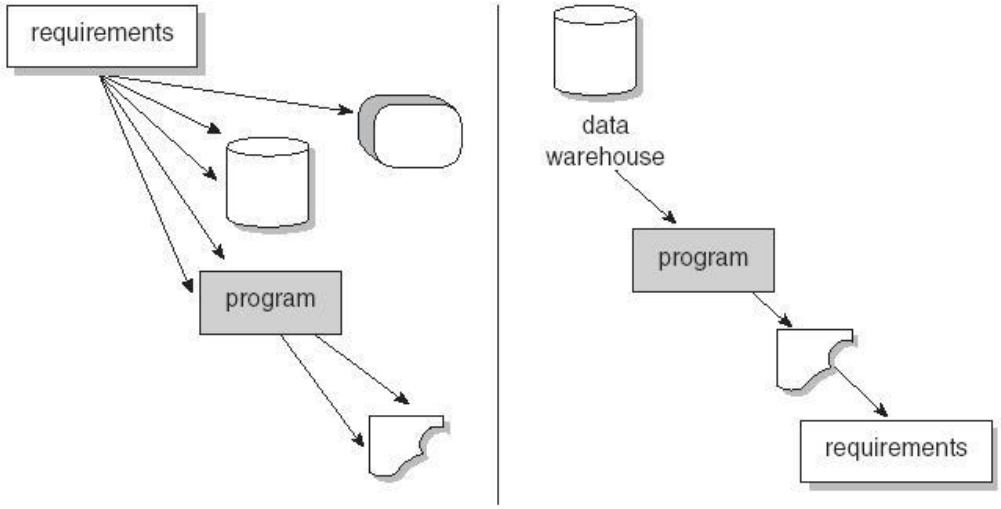
Kedua jenis data ini mempunyai perbedaan mendasar. Data primitif mudah untuk di-***update***, sedangkan data turunan tidak dapat di-***update*** secara langsung, karena harus dilakukan re-kalkulasi berdasarkan data primitif yang telah di-***update***. Status primitif data dinamis dan selalu baru, sedangkan data turunan bersifat statis dan mencerminkan historis data, sehingga penyimpanan kedua jenis data ini tidak dapat disimpan dalam satu lingkungan basis data yang sama. Data primitif mendukung fungsi administrasi, sedangkan data turunan mendukung fungsi manajemen.

Tingkat Arsitektur data

Arsitektur data dalam organisasi mempunyai beberapa tingkatan, mulai dari tingkat operasional, tingkat data ***warehouse*** (atomik), tingkat departemen (***data mart***) sampai tingkat individu (eksekutif). Sifat data operasional sangat rinci, up todate, mudah diakses dan berubah tiap hari dan berorientasi aplikasi. Sifat data ***warehouse*** terintegrasi, bergantung waktu, bebentuk rangkuman dan berorientasi subjek. Sifat data departemen konteks lebih sempit (***parochial***) dan berorientasi kebutuhan departemen. Sifat data individu sementara, heuristik dan berorientasi pada ***PC***. Jika semua tingkat arsitektur data ini ada, maka tingkat redundansi data juga tinggi.

Siklus hidup DSS

Siklus hidup pengembangan aplikasi klasik (***SDLC***) pada tingkat operasional dan ***DSS*** pada tingkat data ***warehouse*** sangat berbeda. Pada ***SDLC*** tingkat operasional dimulai dari kebutuhan dan berakhir pada data, sehingga disebut siklus hidup yang dipicu kebutuhan. Siklus hidup pada tingkat data ***warehouse*** (disebut ***CLDS***) terbalik mulai dari data dan berakhir pada kebutuhan, setelah melalui integrasi data, program aplikasi, hasil analisis, sehingga disebut siklus hidup yang dipicu data. ***CLDS*** umumnya menggunakan metodologi pengembangan spiral



SIKLUS HIDUP KLASIK & DSS (DATA WAREHOUSE)

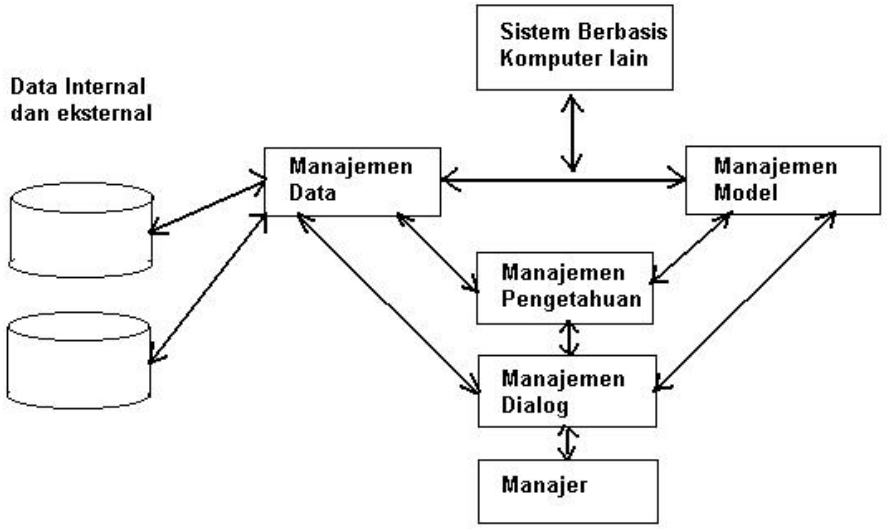
Acuan :

Inmon, WH ;"Building The Data Warehouse"; John Wiley & Sons; edisi-3; 2002

Karakteristik dan Kapabilitas DSS

- Mendukung untuk penentuan keputusan semi terstruktur
- Dukungan untuk tingkatan manajer yang berbeda-beda mulai dari tingkat terbawah sampai puncak
- Dukungan untuk unit atau kelompok organisasi dan individu
- Mendukung keputusan yang interdependen atau sekuensial
- Adaptif dan fleksibel terhadap perubahan waktu
- Mudah digunakan karena dukungan antarmuka yang ***user-friendly***
- Lebih menekankan efektifitas (akurasi, ketepatan dan kualitas) bukan efisiensi
- Kontrol tergantung pengguna
- Kontinuitas pembelajaran melalui evolusi pertambahan kapabilitas sistem
- Mudah dikonstruksi karena sederhana dan dapat dibangun secara terorganisasi
- Menggunakan pemodelan untuk keputusan berdasar analisis dengan berbagai strategi
- Untuk masalah yang rumit dapat dibantu dengan komponen pengetahuan

Komponen ***DSS*** dapat dimodelkan secara konsep yang terdiri dari empat komponen utama atau subsistem, yaitu manajemen data, manajemen model, manajemen dialog dan manajemen pengetahuan



KOMPONEN MODEL DSS KONSEPTUAL

Kapabilitas ***DSS*** secara keseluruhan mempunyai dua kapabilitas, yaitu : Pertama, dapat membuat berbagai ***DSS*** yang spesifik secara cepat dan mudah. Kedua, dapat memfasilitasi proses disain yang iteratif. Secara umum (***general***) ***DSS*** mempunyai tiga kapabilitas, yaitu : mudah digunakan untuk konstruksi termasuk modifikasi ***DSS*** melalui manajemen dialog, dapat mengakses berbagai jenis sumber data melalui manajemen data dan dapat mengakses berbagai aplikasi analisis melalui manajemen model. Berkaitan dengan kapabilitas umum tersebut, maka terdapat tiga jenis kapabilitas komponen.

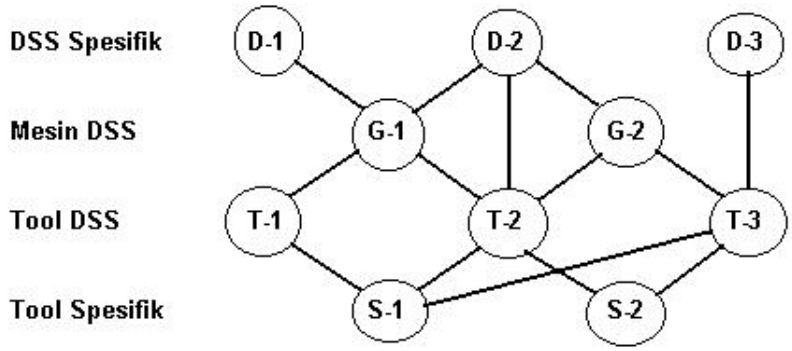
Kapabilitas komponen manajemen dialog, yaitu dapat menghasilkan output dengan berbagai piranti output, dapat menggunakan berbagai jenis piranti input pengguna, dapat menggunakan berbagai cara dialog, menyediakan sarana komunikasi antara pengguna, mendukung dokumentasi pengetahuan dari pengguna, dapat menyimpan dan menganalisis dialog dan mendukung manajemen dialog secara fleksibel dan adaptif.

Kapabilitas komponen manajemen data, yaitu dapat akses ke berbagai jenis format data, mendukung ekstraksi dan agregasi data. Mendukung fungsi akses ***query*** dan display, mendukung pengelolaan basis data, mendukung ***view*** data secara logik, mendukung dokumentasi data, mendukung penelusuran penggunaan data dan mendukung manajemen data secara fleksibel dan adaptif.

Kapabilitas komponen manajemen model, yaitu dapat akses ke pustaka basis model melalui katalog, mempunyai fasilitas pembangun model, fasilitas manipulasi dan menggunakan model, mendukung fungsi manajemen berbasis model, mendukung dokumentasi model, mendukung penelusuran penggunaan model dan mendukung manajemen model secara fleksibel dan adaptif.

Tingkat teknologi

Secara tingkatan teknologi terdapat tiga tingkat teknologi ***DSS***, yaitu Aplikasi ***DSS*** yang spesifik, Pembangkit (***generator***) atau mesin ***DSS*** dan Tool atau kakas ***DSS***. Tool ***DSS*** digunakan untuk membangun mesin ***DSS***. Mesin ***DSS*** digunakan untuk membangun aplikasi ***DSS*** yang spesifik. Secara lebih rinci kakas ***DSS*** dibangun dari berbagai kakas lainnya yang lebih spesifik. Membangun aplikasi ***DSS*** spesifik dapat dilakukan secara langsung dari kakas ***DSS***, tetapi kurang efisien, karena membutuhkan waktu pembangunan yang lebih lama.



Tingkat Teknologi DSS

Acuan :

Efraim Turban; “Decision Support Systems and Expert Systems”; Prentice Hall International Edition; edisi-4

Data Warehouse

OLTP

Sistem manajemen basis data banyak digunakan oleh organisasi untuk mengelola data operasional setiap hari, seperti data reservasi, data deposit yang harus reliabel dan efisien. Aplikasi ***OLTP*** (online transaction processing) pada 3 dekade terakhir ini terus berkembang.

Aplikasi yang dapat menghasilkan rangkuman data dari data lama dan baru ini termasuk aplikasi untuk pendukung keputusan atau, karena dapat digunakan untuk membantu dalam pengambilan keputusan organisasi secara tepat.

Kebutuhan melihat data sesuai kebutuhan (***view***) dan sesuai kriteria pengguna (***query***) yang diekstrak dari kumpulan data yang komplek memicu untuk mempelajari bagaimana cara mendefinisikan ***query*** yang efisien sebelum ***view***. Tahap penentuan ***query*** ini disebut pra-komputasi ***view***.

Untuk tujuan pra-komputasi ***view***, maka informasi harus dikonsolidaikan dari beberapa jenis basis data (***multi-database***) menjadi data ***warehouse*** dengan cara ***copy*** tabel dari berbagai lokasi ke satu lokasi.

Suatu organisasi agar dapat mengambil keputusan dengan tepat memerlukan ***view*** semua aspek organisasi secara komprehensif, oleh karena itu diperlukan adanya data ***warehouse*** yang mengandung data yang diambil dari semua basis data organisasi yang dikelola oleh masing-masing unit organisasi.

Banyak ciri karakter ***query*** pendukung keputusan yang tidak dapat ditangani oleh sistem ***SQL*** (***structural query language***) secara memadai, seperti: klausa WHERE yang terlalu banyak mengandung operator kondisional (***AND*** dan ***OR***) sehingga menjadi rumit.

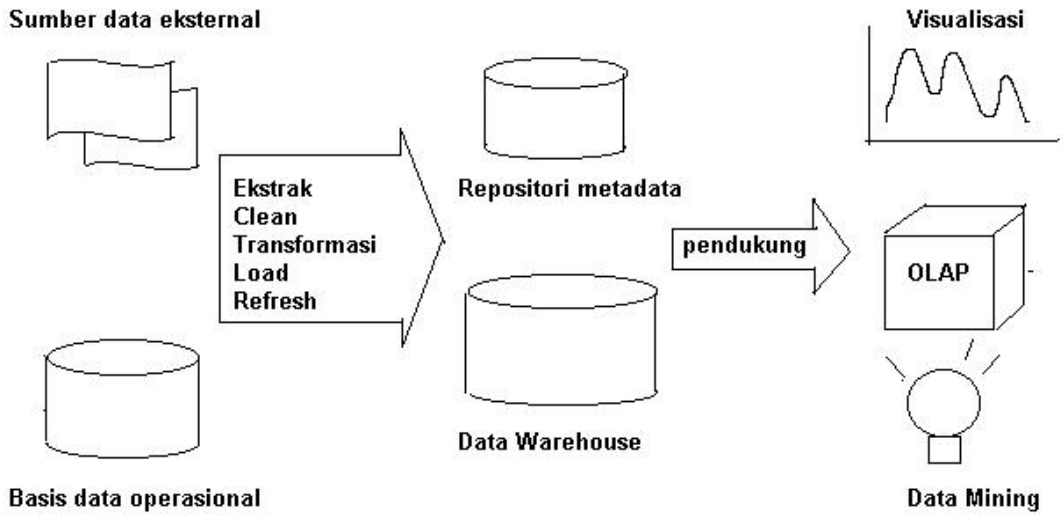
Selain itu banyak aplikasi yang membutuhkan fungsi statistik seperti deviasi standar yang belum dapat ditangani oleh ***SQL***, sehingga ***query SQL*** harus di-embed-kan ke dalam bahasa pemrograman lain. Beberapa aplikasi yang berkaitan dengan waktu membutuhkan basis data temporal, juga ada beberapa aplikasi yang memerlukan pendekatan deduktif dari beberapa ***query*** yang berkaitan untuk mendukung suatu keputusan.

Aplikasi-aplikasi analisis yang banyak mengunakan agregasi didominasi oleh analisis ***query*** secara online yang disebut ***OLAP*** (***online analytic processing***). Sistem ini mendukung jenis ***query*** dalam bentuk array multi dimensi dan dipengaruhi oleh aplikasi end-user seperti spreadsheet selain ***SQL*** basis data.

Selain O***L***AP terdapat aplikasi analisis trend atau pattern dari kumpulan data yang besar yang disebut exploratory data analysis yang sangat sulit di formulasikan dengan ***SQL***, maka diperlukan disiplin ilmu tersendiri yang disebut data ***mining***. Jadi kebutuhan untuk mendukung aplikasi ***OLAP*** dan data ***mining*** mendorong dibangunnya data ***warehouse***.

Data Warehouse (DW)

Data ***warehouse*** terdiri dari data dari berbagai sumber yang terkonsolidasi ditambah informasi rangkuman yang meliputi perioda yang panjang. Ukuran ***DW*** sangat besar dibandingkan basis data yang berkisar mulai dari beberapa Giga byte (***GB***) sampai beberapa Tera byte (***TB***). Ciri yang membedakan ***DW*** dengan ***OLTP*** adalah dalam kerumitan ***query*** dan kecepatan waktu respon. Untuk mendukung ***DW*** diperlukan sistem manajemen basis data (***DBMS***) yang terdistribusi dengan skalabilitas dan availabilitas yang baik.



ARSITEKTUR DATA WAREHOUSE

Tantangan yang dihadapi dalam pembuatan ***DW*** adalah bagaimana membuat suatu ***DW*** yang ukurannya besar yang merupakan integrasi koleksi data yang berasal (***copy***) dari berbagai sumber dari unit organisasi yang tersebar. Koleksi data ini mempunyai ketidakcocokan semantik, seperti cara penamaan suatu atribut, struktur normalisasi yang berbeda.

Tantangan lainnya setelah skema ***DW*** berhasil dibuat adalah bagaimana melakukan populasi data melalui transformasi data dan mempertahankan konsistensi isinya dengan basis data sumber.

Transformasi data dilakukan mulai dari ekstraksi data dari basis data operasional dan sumber eksternal, kemudian diolah sedemikian sehingga meminimalisasi kesalahan dan ketidakcocokan semantik. Transformasi data umumnya dilakukan dengan melalui pendefinisian ***view*** relasional dari berbagai tabel sumber, kemudian disimpan dalam basis data format baru (***DW***).

Dalam pemeliharaan ***DW*** diperlukan proses ***refresh*** yang diambil dari data asal yang up todate dan proses purge data lama yang sudah dianggap tidak relevan lagi. Proses ***refresh*** dilakukan terhadap replika data secara asinkron, walaupun melanggar prinsip data terdistribusi yang harus independen.

Hal penting untuk menjaga ***DW*** adalah dengan selalu memantau kekinian data yang tersimpan, dengan membuat katalog data dan disimpan secara terpisah dalam suatu sistem repositori meta data yang menyimpan informasi sumber data terkini.

OLAP

Aplikasi ***OLAP*** didominasi oleh ***query*** temporer yang kompleks. Secara ***SQL query*** ini melibatkan klausa ***group by*** dan agregasi dari model data multi dimensi. Sebagai contoh data penjualan yang terdiri dari dimensi produk, lokasi dan waktu. ***OLAP*** yang menggunakan basis data multi dimensi dalam bentuk array multi dimensi disebut ***MOLAP*** (***multidimension OLAP***).

Implementasi array multi dimensi dapat diwujudkan secara relasional melalui relasi tabel, seperti contoh di bawah ini:

Tabel Lokasi

lokid	Kota	prop	Negara
1	Jakarta	DKI	Indonesia
2	Penang	Penang	Malaysia
3	Surabaya	Jatim	Indonesia

Tabel Produk

pid	produk	kategori	Harga
11	Batik	Pakaian	200
12	Gelang	Aksesoris	150
13	Sepatu	sepatu	250

Tabel Penjualan

pid	waktuid	lokid	Penjualan
11	1	1	25
11	2	1	10
11	3	1	15
12	1	1	30
12	2	1	20
12	3	1	50
13	1	1	10
13	2	1	12
13	3	1	12
11	1	2	35
11	2	2	22
11	3	2	12

Masing-masing dimensi dapat diwujudkan secara hirarki, misalnya negara terdiri dari propinsi, setiap propinsi terdiri dari beberapa kota. Waktu dalam tahun terdiri dari bulan atau minggu yang terdiri dari hari atau tanggal. Sistem ***OLAP*** yang menyimpan informasi relasi tabel disebut ***ROLAP*** (relational ***OLAP***). Dimensi waktu berupa tabel waktu merupakan aplikasi ***OLAP*** yang tidak dapat ditangani oleh tipe data date dan timestamp pada ***SQL***

Query OLAP

Data dapat dilihat secara ***query*** dan juga dapat dimanipulasi oleh end user dengan spreadsheet. Sebagai contoh ***query*** yang melibatkan beberapa dimensi adalah : total penjualan per kota, propinsi dan rangking penjualan terbesar serta grand total penjualan. Secara ***OLAP*** untuk mendapatkan agregasi total penjualan propinsi dari total penjualan per kota dilakukan proses roll-up, dan sebaliknya disebut proses drill-down, tergantung dimensi, apakah lokasi atau produk.

Secara dua dimensi total penjualan per lokasi dan waktu dapat dilakukan secara pivot, proses ini disebut cross-tabulation. Dimensi lokasi dapat diganti dengan dimensi produk, sehingga secara cross-tabulation diperoleh tabel dua dimensi produk dan waktu.

Dimensi waktu dalam ***OLAP*** sangat penting, seperti total penjualan per bulan, perbulan dan per kota, perubahan total penjualan produk dalam persen, dan penjualan rata-rata untuk akhir perioda waktu tertentu. Jenis ***query*** terakhir ini tidak dapat dilakukan secara ***SQL***.

gambar tabulasi silang penjualan per tahun dan dua propinsi :

Tahun	Jabar	Banten	Jumlah
2000	81	63	144
2001	107	38	145
2002	35	75	110
Total	223	176	399

contoh ***query*** (***SQL***) dengan group by dari tiga tabel penjualan S, waktu T dan lokasi L :

Dengan tiga jenis roll up sbb:
roll-up dimensi waktu
roll-up dimensi lokasi
roll-up dimensi waktu dan lokasi, yaitu

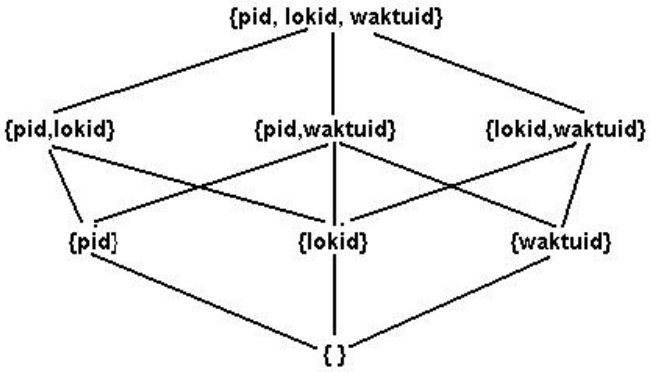
```
SELECT      SUM (S.penjualan)
FROM        Penjualan S, Waktu T
WHERE             S.waktuid=T.waktuid
GROUP BY    T.tahun
```

```
SELECT      SUM (S.penjualan)
FROM        Penjualan S, Lokasi L
WHERE             S.lokid=L.lokid
GROUP BY    L.prop
```

```
SELECT      SUM (S.penjualan)
FROM        Penjualan S, Waktu T, Lokasi L
WHERE             S.waktuid=T.waktuid AND S.lokid=L.lokid
GROUP BY    T.tahun, L.prop
```

Jika ada k dimensi, maka untuk roll-up dibutuhkan dua pangkat k ***query SQL***. Untuk menghasilkan ***query*** tiga dimensi digunakan ***query*** CUBE (kubus) yang dengan menggunakan klausa GROUP BY terhadap list grup. Ada delapan list grup atau subset ***query*** roll up dari set {pid, lokid, waktuid} termasuk subset kosong, dalam bentuk ***query*** seperti:

```
SELECT      SUM (S.penjualan)
FROM        Penjualan S
GROUP BY    list_grup
```

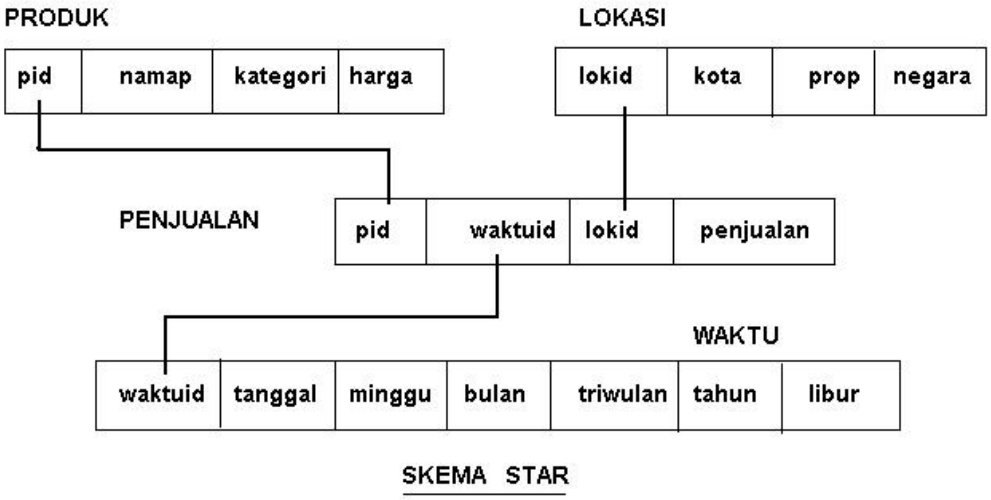


QUERY CUBE

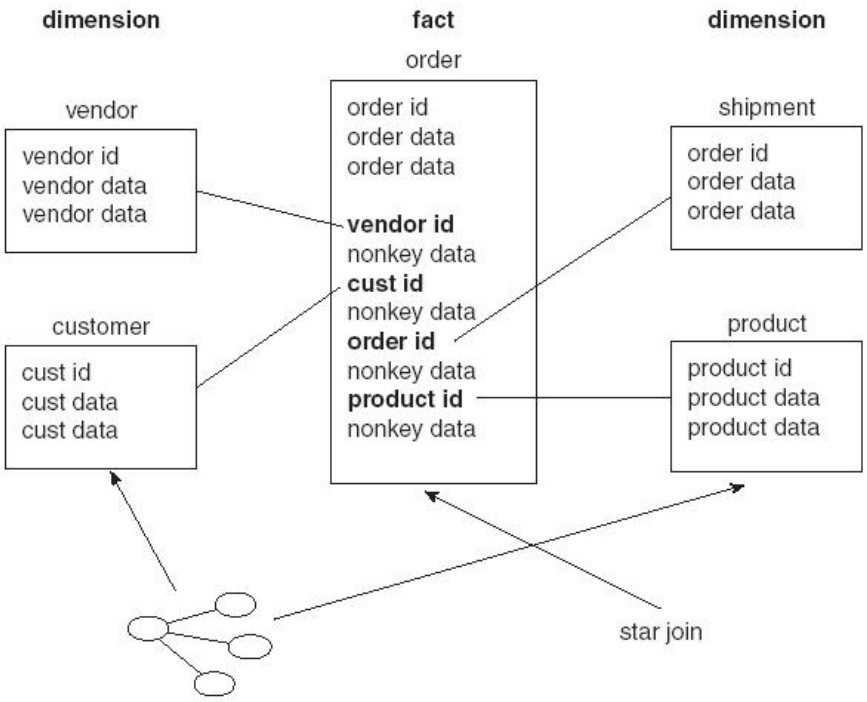
Query kubus

Disain basis data OLAP

Contoh skema star, tabel penjualan sebagai konektor tabel produk, lokasi dan waktu :



SKEMA STAR



TABEL FAKTA : SKEMA STAR

Tabel data (bulk) dalam bentuk tabel fakta (dalam bentuk normal ***BCNF***) dengan identifier yang dibangkitkan oleh sistem untuk mengurangi ukuran tabel data dan berelasi dengan tabel dimensi, seperti contoh pada gambar di atas

Informasi dimensi disimpan pada tabel dimensi yang tidak ternormalisasi dan bersifat statik agar tidak diperlukan operasi join untuk menjaga performansi. Secara disain tabel rangkuman yang mana saja yang perlu disimpan secara sementara (materialisasi) akan mempengaruhi efisiensi penggunaan memori. Secara implementasi ***OLAP*** dilengkapi dengan indeks.

Implementasi Teknik OLAP

Penggunaan indeks dalam format bitmap dapat mempercepat ***query*** karena tabel tidak perlu lengkap, tetapi cukup dalam bentuk sparse (jarang), tergantung bit yang aktif, dibandingkan dengan indek pohon biner (***B+Tree***)

Contoh rating (1 sampai 5) kustomer berdasar gender (L/P). tabel rating memerlukan 5 bit dan tabel gender 2 bit

kustid	nama	gender	rating
112	Joni	L	3
115	Rama	L	5
119	Susi	P	5
122	Bowo	L	4

L	P	1	2	3	4	5
1	0	0	0	1	0	0
1	0	0	0	0	0	1
0	1	0	0	0	0	1
1	0	0	0	0	1	0

Indeks Join

Komputasi join dengan waktu respon yang singkat sulit dilakukan jika ukuran data sangat besar. Dengan menggunakan indeks join untuk operasi ***query*** join dua tabel yang berelasi seperti kustomer dan pembelian menjadi lebih cepat. Secara umum join dapat melibatkan lebih dari dua relasi. Kelemahan indeks join adalah jika join melibatkan lebih dari satu kolom untuk setiap tabel dimensi. Jika data pada kolom jarang, maka dapat digunakan indek join bitmap.

Untuk efisiensi ***OLAP*** yang melibatkan basis data berukuran besar memerlukan teknik kompresi, apalagi kalau menggunakan indeks bitmap yang memudahkan untuk dikompresi.Selain itu penentuan pra-komputasi ***view*** untuk tujuan evaluasi ***query***, seperti ***query*** agregat juga sangat mementukan efisiensi.

Perkembangan ***SQL*** menuju perbaikan ke arah pendukung aplikasi ***OLAP*** multi dimensi seperti data mart, melalui struktur data yang menggunakan join star, yaitu adanya tabel fakta sebagai konektor dari beberapa tabel lain yang saling berelasi. Data mart bukan sebagai pengganti tetapi sebagai komplemen dan pendukung ***DW***.

Modifikasi Query

Contoh :

```
CREATE VIEW PenjualanRegional (kategori, penjualan, prop)
        AS SELECTP.kategori, S.penjualan, L.prop
            FROM        Produk P, Penjualan S, Lokasi L
            WHERE               P.pid = S.pid AND S.lokid = l.lokid
```

Untuk menghitung total penjualan lakukan ***query*** sbb :

```
SELECT      R.kategori, R.prop, SUM(R.penjualan)
FROM        PenjualanRegional R
GROUP BY    R.kategori, R.prop
```

Hasil modifikasi ***query*** menjadi :

```
SELECT      R.kategori, R.prop, SUM(R.penjualan)
FROM        (SELECT     P.kategori, S.penjualan, L.prop
             FROM              Produk P, Penjualan S, Lokasi L
             WHERE      P.pid = S.pid AND S.lokid = l.lokid) AS R
GROUP BY    R.kategori, R.prop
```

Materialisasi View

Contoh :
Sebelum materialisasi ***view*** :

```
SELECT      P.kategori, SUM(S.penjualan)
FROM        Produk P, Penjualan S
WHERE             P.pid = S.pid
GROUP BY    P.kategori

SELECT      L.prop, SUM(S.penjualan)
FROM        Lokasi L, Penjualan S
WHERE             L.pid = S.pid
GROUP BY    L.prop
```

Materialisasi ***view*** : (alternatif)

```
CREATE ***VIEW*** TotalPenjualan (pid, lokid, total)
        AS SELECTS.pid, S.lokid. SUM (S.penjualan)
             FROM        Penjualan S
             GROUP BY    S.pid, S.lokid
```

Setelah materialisasi ***view*** :

```
SELECT      P.kategori, SUM(T.total)
FROM        Produk P, TotalPenjualan T
WHERE             P.pid = T.pid
GROUP BY    P.kategori
```

```
SELECT      L.prop, SUM(T.penjualan)
FROM        Lokasi L, TotalPenjualan T
WHERE             L.lokid = T.lokid
GROUP BY    L.prop
```

Ada tiga hal yang harus diperhatikan ketika melakukan materialisiasi ***view***, yaitu: pertama ***view*** yang mana dan indeks yang mana yang harus dibuat materialsasinya, kedua apakah kumpulan material ***view*** dapat menjawab ***query***, ketiga seberapa sering material ***view*** harus di-***refresh*** agar tetap konsisten dan up todate dengan data base sebenarnya.

Untuk ***refresh*** material ***view*** ada tiga pilihan, yaitu : pertama secara ***lazy*** materi ***view*** di-***refresh*** pada saat ***query*** yang akan memperlambat proses ***query***, kedua secara periodik, dan ketiga secara forced yaitu ***refresh*** dilakukan setiap sejumlah tertentu perubahan pada basis data sebenarnya.

Query Cepat
Untuk kebutuhan ***query*** secara cepat seperti pada search engine di internet, maka diperlukan teknik pendekatan ***query*** cepat, seperti ***query top*** N, dan agregasi online secara aproksimasi seperti teknik ***quick count***.

Contoh ***query top*** N: (menggunakan DB2 IBM: Fitur ***Optimize***)

```
SELECT      P.pid, P.namap, S.penjualan
FROM        Penjualan S, Produk P
WHERE             S.pis=P.pid AND S.lokid=1 AND S.waktuid=3
ORDER BY    S.penjualan DESC
OPTIMIZE    FOR 10 ROWS
```

Contoh hasil agregasi ***online*** (prioritas secara ***online***)

Status(%)	Prioritas	prop	AVG(penjualan)	Konfiden(%)
72	Yes	DKI	4534	97
43	No	Jabar	2345	93
81	Yes	Jateng	6543	98
37	No	Jatim	3976	92

Acuan :
Silberschatz, Abraham; Korth, Henry; Sudharsan; "Database System Concepts"; McGraw-Hill, edisi-3.
Ramakrishnan, Raghu; Gehrke, Johannes; “Database Management Systems”; McGraw-Hill, edisi-2

Data Mining

Data ***Mining*** (***DM***) adalah ilmu yang berkaitan dengan bagaimana mencari informasi atau pengetahuan yang relevan dari basis data yang berukuran besar sekali dengan menggunakan aturan statistik dan pattern secara otomatis. Berbeda dengan pencarian pengetahuan dengan mesin pembelajaran pada bidang kecerdasan buatan ukuran data yang tersimpan pada disk tidak terlalu besar, maka pada ***DM*** ukuran data yang tersimpan pada disk sangat besar.

Dari aspek pencarian pengetahuan ada dua model ***DM***, yaitu Model ***DM*** yang masih melibatkan user secara langsung dalam pencarian pengetahuan, dan model ***DM*** yang pencarian pengetahuannya sepenuhnya ditangani sistem secara otomatis.

Sebagai alat analisis dan eksplorasi, terdapat kesinambungan antara ***query SQL***, OLAP dan ***DM***. ***Query SQL*** berada pada ujung, OLAP berada di tengah dan ***DM*** berada pada ujung yang lainnya. ***Query SQL*** relatif sederhana dibandingkan OLAP, sedangkan ***query*** pada ***DM*** sangat kompleks sehingga memerlukan parameter yang didefinisikan user serta penggunaan algoritma tertentu. Pada implementasinya data pada ***DM*** tidak lengkap atau mengandung noise.

Proses eksplorasi pengetahuan atau ***knowledge discovery process*** (***KDD***) dapat dibagi dalam beberapa tahap.
Seleksi data adalah tahap awal untuk memilah data mentah, kemudian data cleaning merupakan tahap membuang noise dan transformasi nilai data ke dalam unit yang umum, membangkitkan atribut baru dari kombinasi atribut yang ada sesuai skema relasional.
Pada tahap data cleaning mungkin juga dilakukan denormalisasi.
Pada tahap berikutnya yaitu tahap data ***mining*** dilakukan ekstraksi pattern sesungguhnya. Pada tahap terakhir, yaitu tahap evaluasi representasi pattern divisualisasi agar mudah dimengerti user, jika diperlukan terjadi umpan balik ke tahap sebelumnya.

Representasi pengetahuan pada ***DM*** menggunakan aturan asosiasi: if antiseden then konsekuen ({antiseden} => {konsekuen}), yang artinya jika ada predikat pada antiseden yang bernilai true, maka dianggap predikat pada konsekuen juga true. Berbeda dengan aturan statistik pada ilmu eksakta yang menuntut nilai persentase yang tinggi (mendekati 100%), maka biasanya pada ***DM*** nilai persentase dukungan sekitar 50% sudah dianggap tinggi.

Terdapat dua hal yang berkaitan dengan aturan asosiasi, yaitu pertama dukungan sebagai syarat perlu jika ada walaupun hanya satu kejadian dari sekian banyak kejadian yang bernilai true, kedua konfiden sebagai syarat cukup, misalnya jika ada sebanyak 80% transaksi pembelian roti, yang juga melakukan transaksi pembelian susu maka konfiden sudah dianggap tinggi.

Sebagai contoh ***DM*** kita ambil contoh analisis keranjang pasar (***market basket analysis***) yang merupakan koleksi transaksi dari setiap kustomer.

transid	kustid	tanggal	barang	jumlah
111	201	5/1/06	pulpen	2
111	201	5/1/06	tinta	1
111	201	5/1/06	susu	3
111	201	5/1/06	juice	6
112	105	6/3/06	pulpen	1
112	105	6/3/06	tinta	1
112	105	6/3/06	susu	1
113	106	6/3/06	Pulpen	1
113	106	6/3/06	susu	1
114	201	6/3/06	pulpen	2
114	201	6/3/06	tinta	2
114	201	6/3/06	juice	4

Analisis menunjukkan dukungan {pupen,tinta} ada 75%, artinya umumnya jika membeli pulpen, juga membeli tinta, sedangkan dukungan {susu,juice} hanya 25%, artinya pembelian susu dan juice jarang bersamaan

Untuk menghitung frekuensi transaksi dapat menggunakan algoritma properti a priori (bergantung pada keadaan sebelumnya) secara bertahap, mulai dari transaksi per item, dua item dst. Jika pada tahap sebelumnya frekuensi transaksi sudah dibawah batas kriteria yang ditetapkan, maka secara a priori algoritma tidak melanjutkan ke tahap selanjutnya untuk kombinasi (pasangan) item yang mengandung item dengan frekuensi di bawah batas tadi.

Sebagai contoh pada analisis keranjang pasar jika kita set kriteria minimal pembelian 70%, maka pembelian per item {pulpen}, {tinta}, dan {susu} pada tahap awal lolos, sedangkan {juice} tidak lolos. Oleh karena itu pada tahap kedua pasangan {pulpen,tinta}, {pulpen,susu} dan {tinta,susu} yang dianalisis, sedangkan pasangan {pulpen, juice} dan {tinta,juice} tidak dianalisis. Demikian pula pada tahap ketiga pasangan {pulpen, tinta, susu} juga tidak dianalisis, karena pasangan {tinta,susu} jarang terjadi.

Query iceberg

Query iceberg manghasilkan output yang lebih sedikit dari yang sesungguhnya (sifat gunung es), yaitu dengan manambahkan kondisi setelah klausa ***Having***.
Algoritma properti ***a priori*** dapat dijalankan dengan ***query iceberg*** per masing-masing item dengan kondisi pada klausa ***Having***, sebelum dengan ***query*** iceberg terhadap kombinasi item dengan klausa ***Having*** yang sama

Contoh ***query*** iceberg dengan operator Having

```
SELECT    P.kustid, P.barang, SUM(P.jumlah)
FROM      Pembelian P
GROUP BY  P.kustid, P.barang
HAVING    SUM (P.jumlah) > 5
```

Contoh ***query*** a priori terhadap pembeli

```
SELECT    P.kustid
FROM      Pembelian P
GROUP BY  P.kustid
HAVING    SUM (P.jumlah) >5
```

Contoh ***query*** a priori terhadap barang

```
SELECT    P.barang
FROM      Pembelian P
GROUP BY  P.barang
HAVING    SUM (P.jumlah) >5
```

Pada model ***DM*** yang dipandu user, user sebagai penanggungjawab utama dalam pembentukan aturan, dan sistem sebagai pendukung. Biasanya user melakukan hipotesa yang diuji oleh sistem. Biasanya diperlukan beberapa iterasi pendefinisian hipotesa yang dilakukan user, sebelum, sampai sistem memberikan nilai konfiden yang tinggi.

Sistem visualisasi data memudahkan user mengenali pattern berdasarkan warna yang berbeda, misalnya dengan warna yang lebih cerah. Display visual data dapat berbentuk peta, chart atau grafik dengan kuantisasi melalui warna.

Pencarian otomatis aturan dilakukan melaui dua cara, yaitu klasifikasi atau hirarki ***ISA*** dan aturan asosiasi. Pencarian melalui klasifikasi dimulai dari sampel data yang disebut training sebagai hipotesa, kemudian dilakukan pengujian secara partisi terhadap data sesungguhnya seperti pohon keputusan, misalnya untuk mencari pola kartu kredit dimulai dari partisi gelar, kemudian jumlah pendapatan dst.

Pada contoh analisis keranjang pasar di atas dengan melalui pendekatan generalisasi, maka cenderung ada peningkatan frekuensi. Misalnya yang membeli pulpen dan tinta (ATK) juga membeli minuman (mungkin juice atau susu). Jika pada mulanya : {tinta}=>{juice} diubah menjadi {tinta} => {minuman} maka frekuensi meningkat dari 50% ke 75%

Contoh Relasi pembelian dengan hirarki ISA

transid	kustid	tanggal	barang	jumlah
111	201	05/01/06	ATK	3
111	201	05/01/06	minuman	9
112	105	06/03/06	ATK	2
112	105	06/03/06	minuman	1
113	106	06/03/06	ATK	1
113	106	06/03/06	minuman	1
114	201	06/03/06	ATK	4
114	201	06/03/06	minuman	4

Aturan asosiasi generalisasi dapat dilakukan berdasarkan salah satu atribut. Pada contoh analisis keranjang pasar di atas, kalau ***query*** dilakukan berdasarkan kustomer, maka hasil analisis menunjukkan {pulpen} => {susu} mempunyai frekuensi 100% baik sebagai dukungan {pulpen} atau {susu} dan konfiden ({pulpen} => {susu})

Contoh:

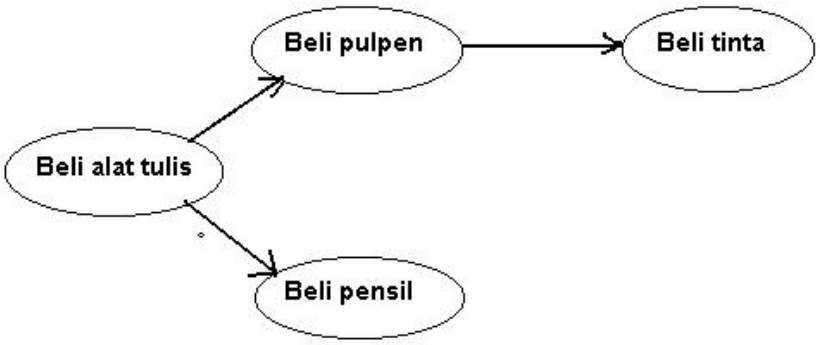
transid	kustid	Tanggal	barang	Jumlah
112	105	06/03/06	pulpen	1
112	105	06/03/06	tinta	1
112	105	06/03/06	susu	1
113	106	06/03/06	pulpen	1
113	106	06/03/06	susu	1
114	201	06/03/06	pulpen	2
114	201	06/03/06	tinta	2
114	201	06/03/06	juice	4
111	201	05/01/06	pulpen	2
111	201	05/01/06	tinta	1
111	201	05/01/06	susu	3
111	201	05/01/06	juice	6

Dengan cara yang sama asosiasi generalisasi dapat dilakukan berdasarkan tanggal, misalnya perhari, setiap awal bulan, setiap hari minggu, dsb. Analisis ini disebut analisis kalendrik

Pola sekuen dapat disimpulkan berdasarkan urutan kejadian transaksi per kustomer berdasarkan frekuensi seperti frekuensi pembelian suatu barang

Aturan asosiasi dapat juga digunakan untuk prediksi, seperti contoh aturan asosiasi {pulpen} => {tinta} dapat dijadikan prediksi untuk peningkatan penjualan tinta di masa mendatang dengan memberikan diskon pada pembelian pulpen, tetapi jika dikembangkan menjadi aturan {pensil} => {tinta}, mungkin prediksi akan gagal.

Jaring ***Bayesian*** adalah graf berarah yang dapat mengambarkan hubungan sebab dan akibat, simpul menggambarkan kejadian, sedangkan garis menggambarkan hubungan sebab akibat. Sebagai contoh pembelian ATK seperti pensil dan pulpen mungkin sering bersamaan, tetapi penyebab pembelian tinta terjadi akibat pembelian pulpen, dapat digambarkan sbb:



JARINGAN BAYESIAN

Aturan klasifikasi dan regresi mirip seperti aturan asosiasi tergantung pada dua hal, yaitu dukungan dan konfiden. Untuk aturan C1 => C2, maka dukungan sama dengan kondisi C1 dan C2. Konfiden kondisi C1 untuk aturan C1 => C2 adalah persentase data yang juga memenuhi kondisi C2.

Contoh informasi asuransi yang terdiri dari atribut umur, jenis mobil dan resiko, seperti: InfoAsuransi (umur:integer; jenis_mobil:string; resiko_tinggi:boolean) dapat menggunakan aturan klasifikasi dan regresi untuk memprediksi aplikasi baru asuransi.

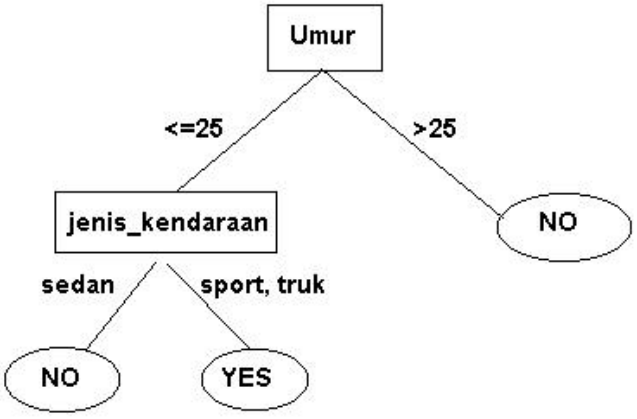
Pada aturan klasifikasi dan regresi terdapat atribut dependen dan atribut prediktor. Bentuk umumnya adalah : p1(X1) dan p2(X2) dan … pn(Xn) => Y = c.
P1, p2, …, pn adalah predikat; X1, X2, …, Xn adalah atribut prediktor dan Y adalah atribut dependen.

Ada dua jenis atribut dependen, yaitu kategori (non-numerik) dan numerik. Aturan yang melibatkan atribut kategori disebut aturan klasifikasi, sebaliknya aturan yang melibatkan atribut dependen numerik disebut aturan regresi. Aturan regresi membutuhkan komputasi.

Untuk contoh kasus asuransi kendaraan di atas, umur adalah atribut dependen numerik, sedangkan jenis mobil adalah atribut dependen dan atribut resiko_tinggi adalah atribut prediktor. Contoh aturan hasil dari penggunaan aturan klasifikasi dan regresi seperti : (10<=umur<=25) dan (jenis_mobil adalah{sport, truk}) => resiko_tinggi=true

Aturan berstruktur pohon

Aturan klasifikasi dan regresi dapat digambarkan dalam bentuk pohon keputusan. Pohon yang menggambarkan aturan klasifikasi disebut pohon klasifikasi atau pohon keputusan, sedangkan pohon yang menggambarkan aturan regresi disebut pohon regresi. Contoh pohon keputusan, seperti contoh kasus asuransi :



Pohon Keputusan Asuransi Kendaraan

Path dari akar sampai salah satu simpul pada pohon keputusan menggambarkan satu aturan klasifikasi. Pohon keputusan ini memudahkan untuk interpretasi.
Pada contoh gambar di atas, terdapat tiga aturan, yaitu :
Pertama, jika umur pemilik mobil di bawah 25 tahun dan jenis mobilnya sedan, maka kemungkinan tidak ikut asuransi.

Kedua, jika umur pemilik mobil di atas 25 tahun juga kemungkinan tidak tertarik asuransi. Ketiga, jika umur pemilik mobil di bawah 25 tahun dan jenis mobil adalah sport atau truk, maka kemungkinan akan tertarik asuransi.
Pada pohon keputusan terdapat atribut pencabangan yang digambarkan sebagai simpul, jika dari simpul itu ada pencabangan ke simpul berikutnya. Simpul pencabangan dan garis berlabel disebut kriteria pencabangan, sedangkan simpul yang tidak bercabang disebut simpul daun. Jadi pada pohon keputusan terdapat aturan klasifikasi yang berakhir pada simpul daun.

Pohon keputusan dibuat dengan dua fase, yaitu fase tumbuh, yaitu ketika semua kejadian digambarkan dalam bentuk pohon yang umumnya berukuran besar. Fase pemotongan adalah fase berikutnya dengan mengabaikan kejadian dengan frekuensi yang kecil di bawah kriteria yang ditetapkan. Metoda pemotongan pohon keputuan disebut metoda seleksi split (cabang), yaitu dengan memilih atribut tertentu. Pada contoh kasus asuransi atribut yang dipilih adalah umur dan jenis_kendaraan.

Contoh relasi informasi asuransi:

umur	Jenis_kendaraan	Resiko_tinggi
23	Sedan	False
30	Sport	False
36	Sedan	False
25	Truk	False
30	Sedan	False
23	Truk	True
30	Truk	False
25	Sport	True
18	Sedan	False

Acuan :
Silberschatz, Abraham; Korth, Henry; Sudharsan; "Database System Concepts"; McGraw-Hill, edisi-3.
Ramakrishnan, Raghu; Gehrke, Johannes; “Database Management Systems”; McGraw-Hill, edisi-2