BAB II

LANDASAN TEORI

Pada bab ini penulis akan menguraikan landasan teori yang menjadi acuan dalam penelitian ini yaitu landasan teori mengenai *data warehouse* dan *data mining*. Pada sub bab *data warehouse* penulis akan menjelaskan pengertian *data warehouse*, model arsitektur *data warehouse* yang akan digunakan berikut teknik dan proses yang akan dilakukan untuk membentuknya sedangkan pada sub bab *data mining* penulis akan menjelaskan mengenai pengertian dan kegunaan *data mining* berikut teknik dan proses untuk membuatnya.

2.1 *Data Warehouse*

2.1.1 Definisi *Data Warehouse*

Data warehouse adalah sesuatu yang berorientasi subyek (*subject-oriented*), terintegrasi (*integrated*), bergantung terhadap waktu (*time variant*) dan tidak berubah (*non-volatile*) yang berguna untuk mendukung proses pembuatan keputusan oleh manajemen (Inmon, 1993). Penjelasan lebih rinci mengenai karakteristik data dalam *data warehouse* adalah sebagai berikut:

- *Subject-oriented*

  Data berorientasi subjek karena *data warehouse* yang di atur lebih mengenai subjek utama dari perusahaan (seperti *customer, product* dan *sales*) daripada area aplikasi utama (seperti *customer invoicing,*

stock control dan *product sales*). Hal ini merefleksikan keperluan untuk menyimpan *decision-support data* daripada *application-oriented data*

- *Integrated*

  Data memiliki karakteristik *integrated* dikarenakan datangnya sumber data secara bersamaan, berasal dari sistem aplikasi besar perusahaan yang berbeda. Sumber data yang ada sering tidak konsisten, contohnya format yang berbeda antara data yang satu dengan lainnya. Sumber data yang ada harus dibuat konsisten untuk mempresentasikan suatu tampilan yang dapat memberikan informasi yang tepat dari bagi pengguna

- *Time-variant*

  Data memiliki karakteristik *time-variant* karena data hanya akurat dan valid pada suatu waktu atau interval waktu

- *Non-volatile*

  Data tidak bisa di *update* pada saat itu juga tetapi di *refresh* dari sistem operasional secara teratur. Data baru yang akan ditambahkan tidak menimpa atau mengganti suatu data yang sudah ada melainkan ditambahkan pada *database* sebagai *record* baru. *Database* secara berkesinambungan menggabungkan data baru ini dengan cara *incremental* dan mengintegrasikannya dengan data sebelumnya.

2.1.2 Keuntungan *Data Warehouse*

Implementasi *data warehouse* yang tepat dapat memberikan keuntungan-keuntungan antara lain:

- Potensi ROI (*Return On Investment)* yang besar

  Suatu perusahaan akan mengeluarkan sumber daya yang cukup besar untuk mengimplementasikan *data warehouse* dan pengeluaran yang dapat berbeda-beda sesuai dengan variasi solusi teknikal yang akan diterapkan pada perusahaan. Bagaimana pun juga, suatu studi oleh *International Data Corporation* (IDC) pada tahun 1996 melaporkan bahwa rata-rata tiga tahun *return of investment* (ROI) dalam *data warehouse* mencapai 401%, dengan lebih dari 90% dari perusahaan yang di survei mencapai lebih dari dari 40% ROI, setengah dari perusahaan mencapai lebih dari 160% ROI, dan seperempat lebih mendapat lebih dari 600% ROI (IDC, 1996)

- *Competitive Advantage*

  Return on investment yang besar dari perusahaan yang berhasil mengimplementasikan suatu *data warehouse* adalah bukti dari sangat besarnya *competitive advantage* yang dapat diperoleh dengan menggunakan teknologi ini. *Competitive advantage* diperoleh dengan mengijinkan si pengambil keputusan untuk mengakses data tersembunyi yang sebelumnya tidak tersedia, tidak diketahui, dan tidak dimanfaatkan seperti data mengenai pelanggan, tren dan permintaan

- Meningkatkan produktifitas dari pengambil keputusan perusahaan

  Data warehouse meningkatkan produktifitas dari pengambil keputusan perusahaan dengan membuat integrasi *database* yang konsisten, berorientasi subyek dan *historical data. Data warehouse* mengintegrasikan data dari banyak sistem yang tidak kompatibel menjadi suatu bentuk yang menyediakan satu tampilan yang konsisten mengenai perusahaan. Dengan mentransformasikan data menjadi informasi yang berguna, *data warehouse* mengijinkan si pengambil keputusan untuk melakukan analisis lebih sesuai dengan kenyataan, akurat dan konsisten.

2.1.3 Kategori Data pada *Data Warehouse*

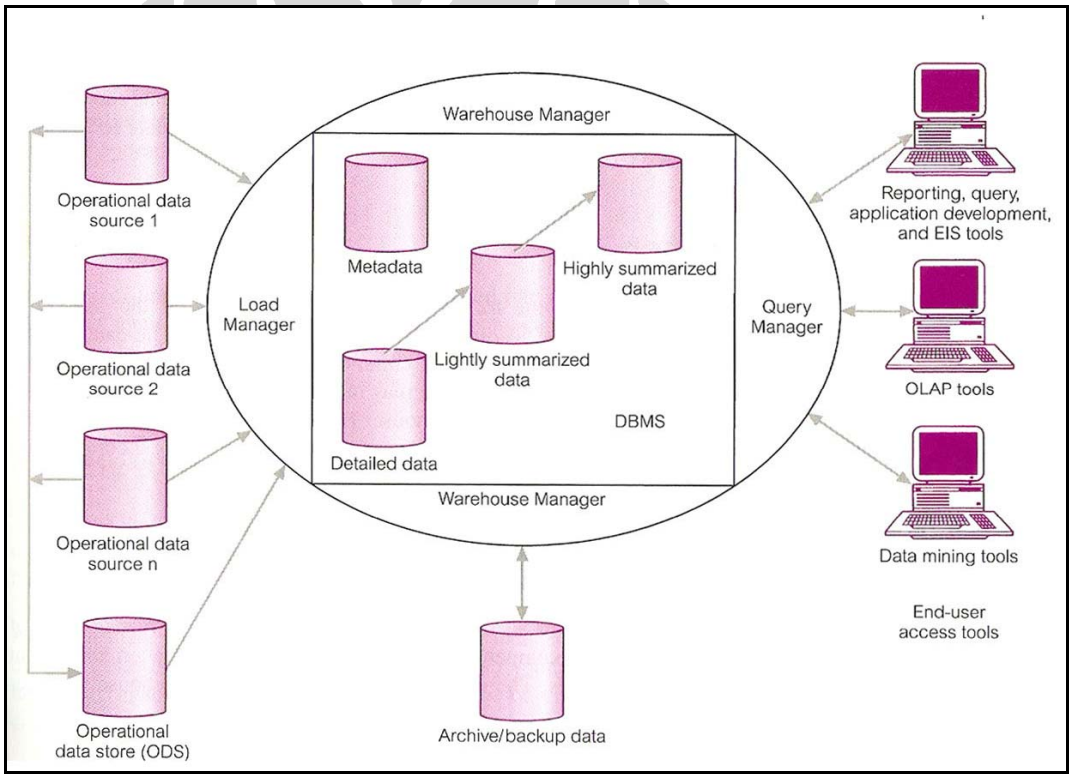
Untuk memahami *data warehouse* lebih dalam, ada dua aspek penting yang harus dipahami yaitu pertama adalah memahami tipe spesifik (*classification)* dari data yang akan disimpan di *data warehouse* dan kedua mengenai tahapan proses dalam pembuatan *data warehouse*. Mengenai kategori pada *data warehouse*, kategori ini diakomodasi berdasarkan *time-dependent data sources*.

Adapun klasifikasinya adalah sebagai berikut ini: (Kantardzic, 2003)

- *Old detail data* (data lama)
- *Current (new) detail data* (data saat ini atau baru)
- *Lightly summarized data* (data yang disimpulkan secara ringan)
- *Highly summarized data* (data yang disimpulkan secara berat)
- *Metadata* (direktori data atau panduan tentang data)

2.1.4 Arsitektur *Data Warehouse*

Untuk mempermudah proses pembangunan suatu *data warehouse* diperlukan pemilihan arsitektur yang tepat dan pemahaman yang baik terhadap arsitektur *data warehouse*. Pada sub bab ini dijelaskan mengenai arsitektur dan komponen utama dari *data warehouse* (Anahory dan Murray, 1997) beserta proses, *tools*, dan teknologi yang berhubungan dengan *data warehouse*. Untuk lebih jelasnya dapat dilihat pada Gambar 2.1 berikut ini.



Gambar 2.1 – Arsitektur *data warehouse* (Connoly and Begg, 2005)

Penjelasan untuk lebih detailnya dari Gambar 2.1 adalah sebagai berikut:

- *Operational Data*

  Sumber data dari *data warehouse* di peroleh dari operasional data perusahaan yang disimpan pada *database*, data operational yang disimpan pada masing-masing *workstation* dan data eksternal dari *vendor*

- *Operational Data Store*

  Suatu Operational Data Store (ODS) adalah penyimpanan dari integrasi data operasional untuk analisis. ODS memiliki struktur dan cara memperoleh data yang sama dengan *data warehouse,* tetapi ODS dalam pelaksanaannya hanya sebagai *staging area* untuk data yang akan dipindahkan ke dalam d*ata warehouse*. ODS biasanya dibuat ketika sistem operasional ditemukan tidak mampu untuk mencapai kebutuhan dari *report*. ODS memberikan pengguna kemudahan dari *database* relasional tanpa melampaui fungsi *decision support* dari *data warehouse*. Membangun suatu ODS sangat membantu dalam pembangunan suatu d*ata warehouse* karena ODS dapat menyediakan data yang telah di *extract* dari sumber data dan telah dibersihkan. Hal ini berarti, pekerjaan integrasi dan restrukturisasi data pada *data warehouse* dibuat lebih sederhana

- *Load Manager*

  Load Manager melakukan semua operasi yang berhubungan dengan proses *extract* dan *loading* semua data ke dalam d*ata warehouse.* Data di *extract* dari sumber data atau dari *operational data store*. Operasi yang dilakukan oleh *load manager* termasuk transformasi dari data yang akan disiapkan untuk dimasukkan ke dalam *data warehouse*. Besar dan

kompleksitas dari komponen data yang ditranformasi dan *laading* akan berbeda antara *data warehouse* dan akan dikonstruksi menggunakan kombinasi dari *data loading tools* milik *vendor* dan program yang telah dikustomisasi

- *Warehouse Manager*

  Warehouse manager (biasa dikatakan *front-end component)* melakukan semua operasi yang berhubungan dengan manajemen dari data pada *data warehouse.* Komponen ini dibangun menggunakan *data management tools* milik *vendor* dan program yang telah dikustomisasi. Operasi yang dilakukan oleh *warehouse manager* termasuk:

  - Menganalisis data untuk meyakinkan konsistensi data yang ada
  - Transformasi dan menggabungkan sumber data dari penyimpan sementara ke dalam tabel *data warehouse*
  - Membuat *index* dan *view* pada tabel basis
  - Melakukan denormalisasi (jika diperlukan)
  - Membuat agregasi (jika diperlukan)
  - *Backup data* dan *archiving data*

- *Query Manager*

  Query Manager (biasa dikatakan *back-end component)* melaksanakan semua operasi yang berhubungan dengan manajemen *user queries*. Komponen ini dikonstruksi menggunakan *end-user data access tools* milik vendor, *data warehouse monitoring tools, database facilities,* dan program yang sudah dikustomisasi. Kompleksitas *query manager* ditentukan oleh fasilitas yang disediakan oleh *end-user tools* dan *database*

- *Detailed Data*

  Detailed data adalah area pada *data warehouse* yang menyimpan semua data rinci dalam skema *database*. Dalam banyak kasus, data yang rinci tidak disimpan secara *online* tetapi akan tersedia dengan melakukan aggregasi data ke *level* berikutnya yang lebih rinci. Pada basis standar, data rinci dapat ditambahkan ke *data warehouse* sebagai suplemen data

- *Lightly and Highly Summarized Data*

  Area ini menyimpan semua data keseluruhan dari *lightly dan highly summarized data* yang dihasilkan oleh *warehouse manager*. Tujuan dari disimpannya informasi yang disimpulkan ini adalah untuk mempercepat kinerja *query*. Meskipun akan ada peningkatan pengelolaan data dikarenakan hal ini, tetapi hal ini diimbangi dengan menghilangkan kebutuhan untuk secara terus menerus melakukan *summary operation*s (seperti *sorting* dan *grouping*) untuk memenuhi suatu *query* pengguna

- *Archive or Backup Data*

  Ini adalah area dimana *data warehouse* menyimpan data rinci dan menyimpulkan data untuk tujuan *archiving* dan *backup*. Meskipun *summary data* berasal dari data rinci, sebaiknya *online summary data* tetap diperlukan untuk di *backup* jika data ini disimpan melebihi periode penyimpanan untuk *detailed data*. Data ini di kirim ke penyimpanan seperti *magnetic tape* atau *optical disc*

- *Metadata*

  Pada area *metadata* dilakukan proses menyimpan semua definisi *metadata* (data mengenai data) yang digunakan proses pada *data warehouse*. *Metadata* digunakan untuk tujuan yang bervariasi termasuk:

  - Proses *extractng* dan *loading*

    Metadata digunakan untuk memetakan sumber data ke tampilan umum data didalam *warehouse*

  - Proses manajemen *warehouse*

    Metadata digunakan untuk mengotomatisasi hasil dari tabel *summary*

  - Sebagai bagian dari proses manajemen *query*

    Metadata digunakan untuk *query* secara langsung ke semua sumber data yang tepat

- *End-User Access Tools*

  End-User Access Tools digunakan oleh pengguna untuk berinteraksi dengan *data warehouse*. Untuk pengkategorian *tools* ini maka akan dibagi menjadi lima grup utama (Berson dan Smith, 1997) yaitu:

  - *Reporting and query tools*

    Reporting tools adalah *tools* yang digunakan untuk membuat laporan regular operasional sedangkan *query tools* digunakan untuk *relational data warehouse* yang dibentuk untuk menerima *structure query language* (SQL) atau menghasilkan SQL untuk melakukan *query* ke data yang disimpan di *data warehouse*. Contoh *tools* nya adalah *query- by-example* (QBE)

- *Application development tools*

  Yaitu pengembangan aplikasi yang menggunakan *graphical data access tools* yang didesain secara primer untuk lingkungan *client-server*

- *Executive information system* (EIS) *tools*

  Yaitu suatu alat yang berhubungan dengan *mainframe* yang memungkinkan pengguna untuk membuat kustomisasi, aplikasi *graphical decision-support* untuk menyediakan suatu gambaran mengenai data perusahaan dan akses ke sumber data *eksternal*

- *Online Analytical Processing* (OLAP) *tools*

  Yaitu suatu alat yang berdasarkan konsep *multi-dimensional database* dan mengijinkan pengguna untuk menganalisa data menggunakan kompleks dan *dimensional views*

- *Data Mining Tools*

  Yaitu suatu alat yang berguna untuk menemukan sesuatu yang baru dan bermanfaat mengenai korelasi, pola, dan tren dengan menggali sejumlah data besar menggunakan teknik statistikal dan matematikal. Contohnya seperti *SQL Server Business Intelligence Development Studio* dan *RapidMiner*.

2.1.5 Tahapan *Data Warehouse*

Memilih tahapan atau proses yang tepat untuk mengkonstruksi *data warehouse* adalah langkah yang kritis dalam pembuatan suatu *data warehouse*. Data pada *data warehouse* harus distandarisasi terlebih dahulu sebelum dimasukkan. Proses yang digunakan dalam memproses *data* sebelum dimasukkan ke dalam suatu *data warehouse* adalah proses ETL (*extract, transform and load*). Penjelasan dari masing-masing proses adalah sebagai berikut:

- *Extract*

  Proses *extract* adalah proses mengekstrak (*extracting)* dan pengambilan data dari sumber pada sistem untuk di load ke *data warehouse.* Sumber data dapat diperoleh secara alternatif melalui ODS (*operational data storage)*. Data harus dikonstruksi ulang sebelum dimasukkan ke dalam *data warehouse*.

  Proses konstruksi ini melibatkan proses:

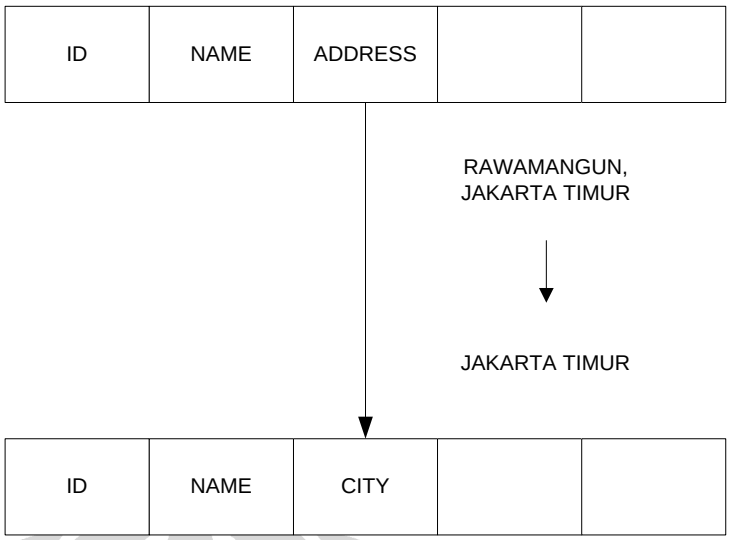
  - *Cleansing data* yaitu proses pembersihan data kotor. Kotor dalam hal ini adalah data berkualitas rendah seperti ketidak-konsistenan penulisan nama, kode id, duplikasi data, data tidak lengkap dan lain-lain
  - Restrukturisasi data pada *data warehouse* untuk memenuhi kebutuhan yang ada. Contoh: penambahan atau pengurangan *fields* dan *denormalisasi data*
  - Memastikan sumber data konsisten dengan data yang sudah ada di dalam *data warehouse*

- *Transform*

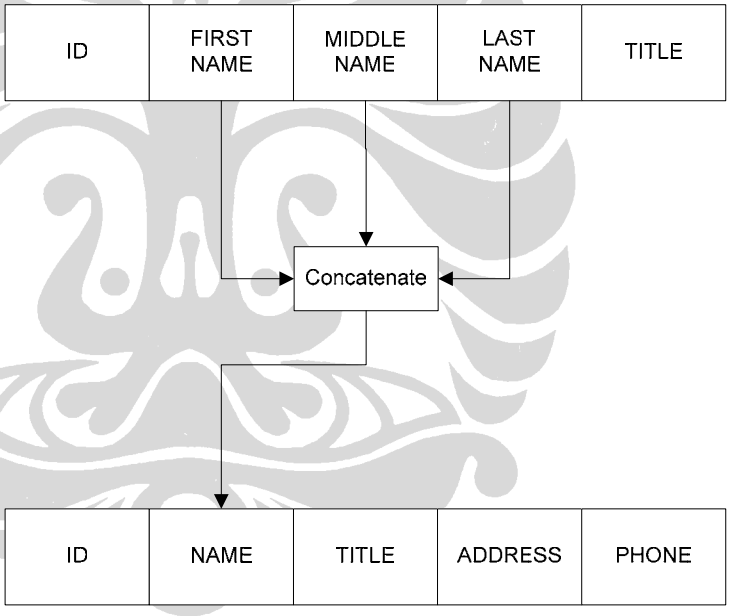
  Proses *transform* adalah proses pengubahan data, dimana data yang diperoleh dari proses *extract* (dalam format operasional) menjadi data dalam bentuk *data warehouse*. Proses *transform* ini melibatkan proses:

  - *Summarizing the data,* dengan cara pemilihan *(selection),* proyeksi *(projecting)*, penggabungan *(joining)*, normalisasi *(normalization)*, agrerasi *(aggregation)* dan *grouping* relasional data menjadi *views* yang lebih nyaman dan berguna bagi pengguna
  - *Packaging the data*, dengan mengkonversi *detail data* atau *summarized data* menjadi format yang lebih berguna seperti *spreadsheets, text document, private database* dan lainnya

  Terdapat dua cara transformasi yaitu dengan menggunakan fungsi *record-level*, dan fungsi *field-level*. Fungsi *record-level* melibatkan proses *summarizing the data* sedangkan fungsi *field-level* melibatkan proses *packaging the data*. Dua tipe dari *field-level* adalah *single-field* dan *multi-field*. Transformasi dengan menggunakan *field-level* mengubah data dari satu *field* menjadi satu *field* yang lain. Berbeda dengan *multi-field* dimana satu data atau lebih diubah menjadi field baru. Gambar mengenai *single-field* dan *multi-field* dapat dilihat pada Gambar 2.2 dan 2.3.



Gambar 2.2 – Contoh transformasi s*ingle-field*



Gambar 2.3 – Contoh transformasi m*ulti-field*

- *Load*

  Proses *load* adalah tahapan terakhir dari proses ETL. Pada proses ini akan dilakukan proses pemuatan data dari proses *transform* ke dalam suatu *data warehouse.* Pada proses ini dilakukan juga proses *indexing* untuk memberikan indeks ke masing-masing data untuk mempercepat proses

query. Terdapat dua mode *loading* ke dalam *data warehouse* yaitu *refresh* dan *update*. Mode *refresh* yaitu proses menuliskan kembali keseluruhan data di dalam *data warehouse* pada suatu interval waktu. Sedangkan untuk mode *update* yaitu suatu proses untuk meng-*update* (tidak menghapus atau menimpa data lain) data yang berubah ke tempat tujuan pada *data warehouse*. Mode *refresh* digunakan pertama kali ketika *data warehouse* berjalan dan data hendak dimuat, sedangkan mode *update* umumnya digunakan ketika pemeliharaan data atau ketika *data warehouse* sedang *running*.

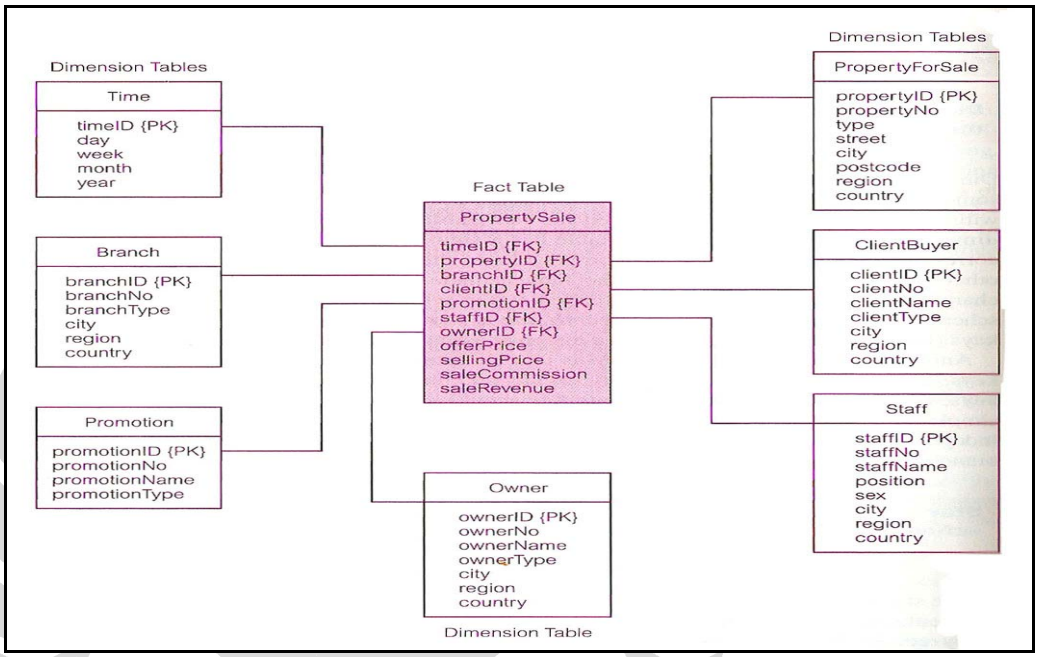
2.1.6 Desain *Data Warehouse*

Mendesain suatu *data warehouse database* sangatlah komples. Untuk memulai pembuatannya harus memperhatikan keperluan yang utama dan memilih data yang harus didahulukan terlebih dahulu, baru setelah itu bisa diperoleh komponen-komponen *database* yang akan digunakan untuk pembuatan *database* untuk *data warehouse*. Teknik yang digunakan untuk mendeskripsikan komponen dari *database* dari *data warehouse* adalah *dimensional modeling* (DM). Pengertian dari *dimensional modeling* adalah suatu teknik desain secara logikal yang memiliki sasaran untuk mempresentasikan data dengan standar, bentuk intuitif yang mengijinkan akses secara sangat cepat. Setiap tabel *dimensional model* memiliki komposisi dari satu tabel dengan *composite key* yang dinamakan *fact table* dan sekumpulan set tabel yang lebih kecil yang dinamakan *dimension table.*

Dimensional modeling memiliki beberapa struktur skema, yaitu:

- *Star schema*

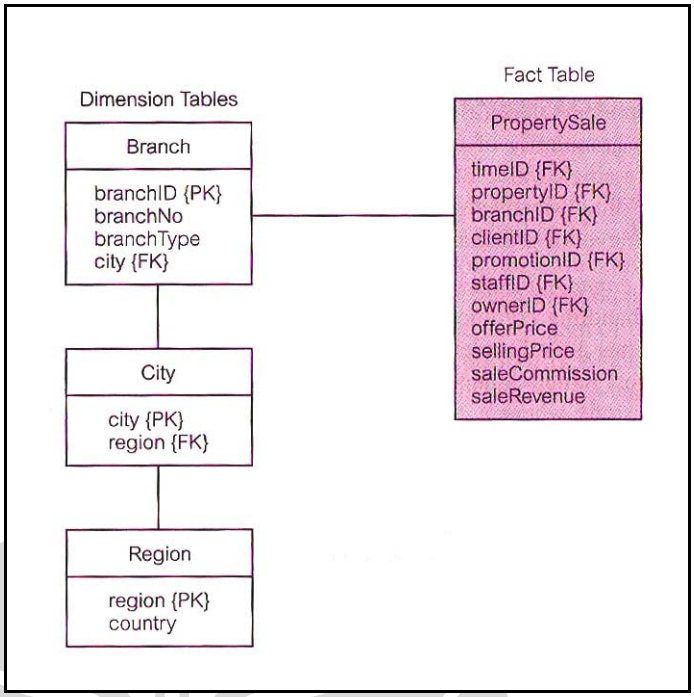
  Struktur *logical* yang memiliki *fact table* mengandung data fakta pada posisi tengah, dikelilingi oleh *dimension tables* mengandung referensi data (yang bisa di denormalisasi). Contoh dapat dilihat pada Gambar 2.4



Gambar 2.4 – *Star schema* (Connoly and Begg, 2005)

- *Snow schema*

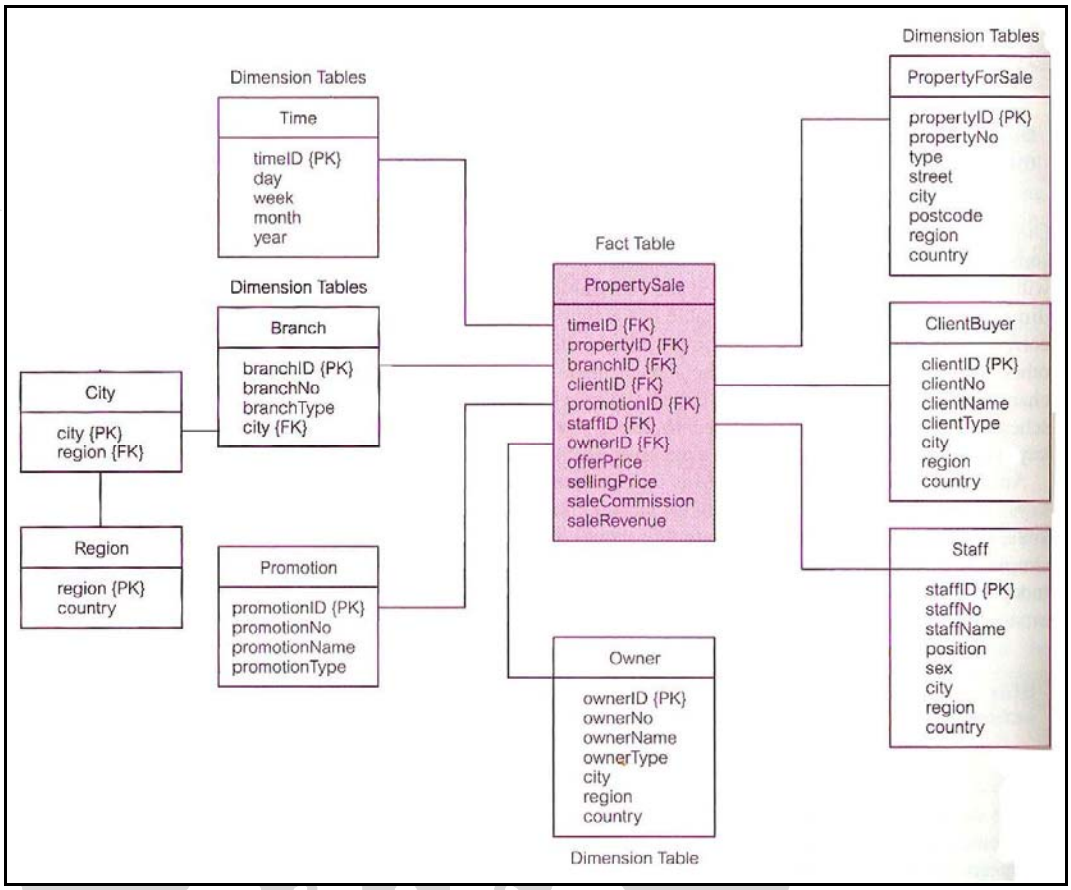
  Varian dari *star schema* dimana tabel dimensi tidak mengandung denormalisasi data. Skema ini memperbolehkan dimensi memiliki dimensi. Sebagai contoh kita bisa melakukan normalisasi *location data(city, region* dan *country attributes)* pada *branch* di *dimension table* dari Gambar 2.4 untuk membuat *dimension tables* baru yang dinamakan *city* dan *region*. Versi normalisasi dari *branch dimension table* dari *property sales shema* ditampilkan pada Gambar 2.5



Gambar 2.5 – *Snow schema* (Connoly and Begg, 2005)

- *Starflake schema*

  Starflake shema adalah skema *database* yang paling cocok untuk *database* yaitu dengan menggabungkan *denormalized star* dengan *normalized snowflake schemas*. Penggunaan *starflake schema* ini tidak selalu harus digunakan tetapi disesuaikan dengan kebutuhan yang diperlukan dari sistem *database* yang akan dikembangkan. Contoh dari gambar *Starflake schema* dapat dilihat pada Gambar 2.6 berikut ini



Gambar 2.6 – *Starflake schema* (Connoly and Begg, 2005)

2.2 *Data Mining*

2.2.1 Definisi *Data Mining*

Data mining adalah suatu proses mengekstraksi secara valid, sebelumnya belum diketahui, komprehensif dan informasi yang dapat memberikan aksi dari *database* besar dan menggunakannya untuk membuat keputusan bisnis yang krusial (Simoudis 1996). *Data mining* berhubungan dengan analisis dari data dan penggunaan teknik *software* untuk menemukan pola yang tersembunyi dan tidak diharapkan dan relasinya dalam bentuk set suatu data. Fokus dari *data mining*

adalah untuk memunculkan informasi yang tersembunyi dan tidak diharapkan. Informasi yang tersembunyi tersebut dapat memberikan nilai tambah pada bisnis perusahaan. Selain alasan di atas terdapat pula alasan-alasan lain mengapa diperlukannya penggunaan *data mining* berikut ini:

1. Data yang tersedia berjumlah sangat besar

   Dalam dekade terakhir ini, harga dari perangkat keras terutama *harddisk* telah turun secara drastis. Disamping itu perusahaan telah mengumpulkan sejumlah data yang sangat besar dari banyak aplikasi yang dimiliki. Dengan sejumlah data ini, perusahaan melakukan eksplorasi data untuk mencari pola tersembunyi sebagai panduan untuk membantu strategi bisnis yang akan dijalankan
2. Kompetisi yang meningkat

   Kompetisi yang ada sangat tinggi sebagai hasil dari marketing dan dengan adanya saluran distribusi seperti internet dan telekomunikasi. Perusahaan akan menghadapi kompetisi dunia karena itu kunci suksesnya bisnis adalah kemampuan untuk membina pelanggan yang sudah ada dan mendapatkan yang baru. Teknologi *data mining* dapat membantu perusahaan untuk menganalisa faktor yang mempengaruhi hal tersebut
3. Kemampuan Teknologi

   Teknologi *data mining* sebelumnya hanya ada pada lingkungan akademik tetapi saat ini banyak teknologi seperti ini semakin canggih dan siap untuk diterapkan pada industri. Algoritma yang ada semakin akurat, efisien dasn dapat menangani komplikasi data yang meningkat. Sebagai tambahan,

data mining application programming interfaces (APIs) telah distandarisasi, sehingga mengijinkan pengembang untuk membuat aplikasi *data mining* yang lebih baik

2.2.2 Teknik *Data Mining*

Ada empat operasi utama yang dapat dilakukan pada teknik *data mining* yaitu *predictive modeling, database segmentation, link analysis* dan *deviation detection*. Meskipun salah satu dari operasi utama dapat digunakan untuk mengimplementasikan aplikasi bisnis apapun, ada keterhubungan yang ditemukan antara aplikasi dan operasi yang bersangkutan. Teknik adalah implementasi secara spesifik dari operasi *data mining*. Bagaimanapun juga masing-masing operasi memiliki kekuatan dan kelemahannya masing-masing. Untuk lebih jelasnya mengenai teknik yang berasosiasi dengan salah satu dari empat operasi utama *data mining* (Cabena 1997) dapat dilihat pada Tabel 2.1 berikut ini:

Operations	***Data Mining Techniques***
Predictive modeling	*Classsification* *Value Prediction*
Database segmentation	*Demographic clustering* *Neural clustering*
Link analysis	*Association discovery* *Sequential pattern discovery* *Similar time sequence discovery*
Deviation detection	*Statistics* *Visualization*

Tabel 2.1 – *Data mining operations and associated techniques*
(Connoly and Begg, 2005)

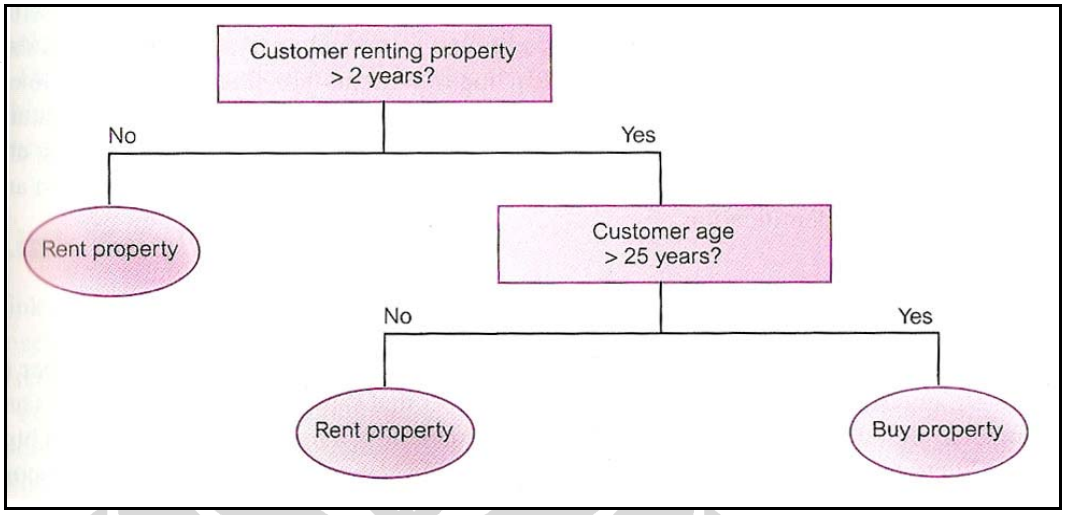
2.2.3 *Predictive Modeling*

Predictive Modeling menggunakan pendekatan generalisasi dari *'real world'* dan kemampuan untuk menempatkan data baru ke kerangka utama. *predictive modeling* bisa digunakan untuk menganalisa *database* yang ada untuk menentukan karakteristik (model) mengenai data set. Model ini dikembangkan menggunakan pendekatan *supervised learning* yang terdiri dari dua fase: *training* dan *testing*. *Training* membuat model menggunakan sampel besar dari data yang dinamakan *training set*, sedangkan *testing* mencoba model baru, data yang sebelumnya tidak terlihat untuk menentukan keakurasian dan karakteristik performa fisik. Ada dua teknik yang berasosiasi dengan *predictive modeling*: *classification* dan *value prediction*.

2.2.4 *Classification*

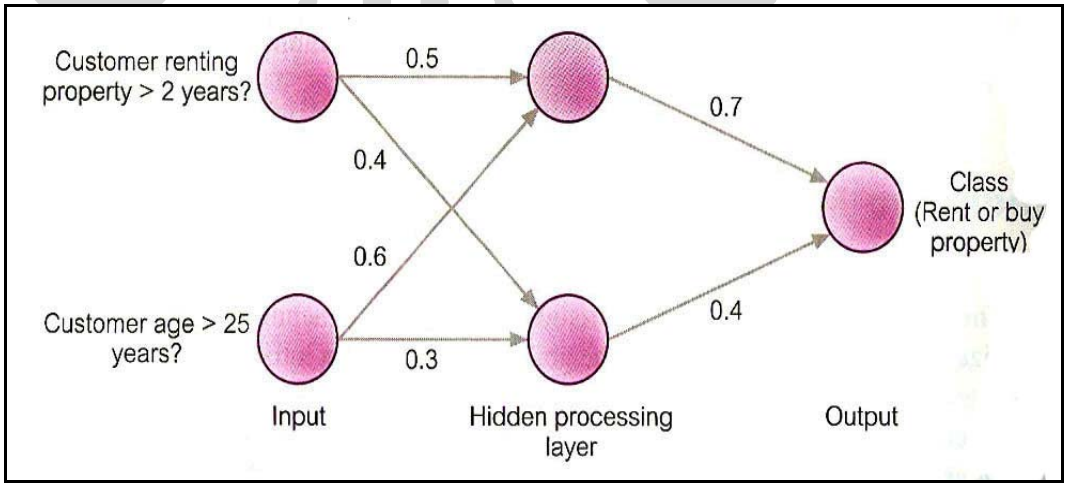
C*lassification* digunakan untuk membangun spesifik kelas yang telah ditentukan sebelumnya untuk masing-masing *record* di *database,* dari suatu set terbatas ke nilai kelas yang memungkinkan. Ini adalah dua spesialisasi dari klasifikasi: *tree induction* dan *neural induction*. Contoh dari klasifikasi menggunakan *induction* ada pada Gambar 2.7. Pada contoh ini menggambarkan apakah pelanggan yang sedang menyewa properti tertarik untuk membeli properti. *Predictive model* telah menentukan bahwa hanya dua variabel yang digunakan yaitu: lama waktu penyewaan oleh pelanggan dan umur dari pelanggan. Model ini

memprediksi pelanggan yang telah menyewa lebih dari dua tahun dan berusia lebih dari 25 tahun lebih tertarik untuk membeli properti.



Gambar 2.7 - Contoh c*lassification* menggunakan *tree induction*
(Connoly and Begg, 2005)

Contoh dari c*lassification* menggunakan *neural induction* pada Gambar 2.8



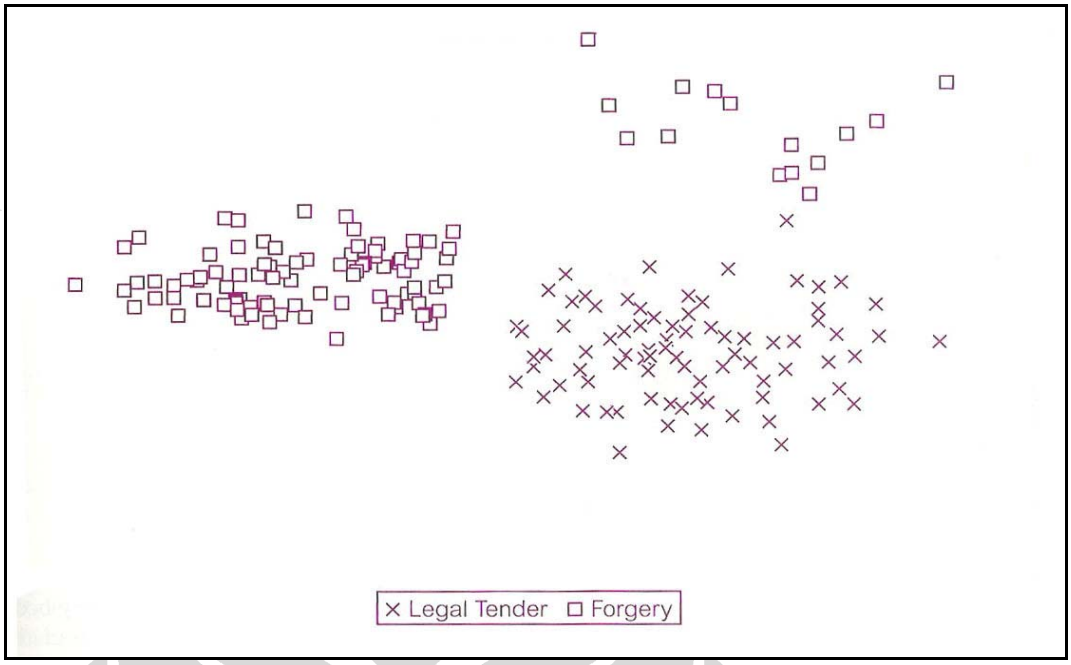
Gambar 2.8 – Contoh c*lassification* menggunakan *neural induction*
(Connoly and Begg, 2005)

Pada kasus ini, c*lassification* dari data diperoleh dengan menggunakan *neural network*. *Neural network* mengandung koleksi dari *node-node* yang terkoneksi demgan *input, output* dan *processing* pada masing-masing *node.* Antara lapisan

input dan *output* mungkin sebagai sejumlah lapisan proses tersembunyi. Masing-masing proses unit dalam satu lapisan saling berhubungan dengan proses unit di lapisan berikutnya oleh *weighted value* yang menggambarkan kekuatan hubungan.

2.2.5 *Clustering*

Sasaran *clustering* adalah untuk membagi *database* menjadi sejumlah *segments, clusters* dan *records* yang sama. Yaitu *records* yang membagi sejumlah *properties* dan dapat dianggap sebagai *homogeneous* (*Segments* memiliki *internal homogeneity* yang tinggi dan *external homogeneity* yang tinggi. Pendekatan ini menggunakan *unsupervised learning* untuk menemukan *homogeneous* sub-populasi di *database* untuk meningkatkan akurasi dari profil yang ada. Aplikasi dari *database segmentation* menggunakan *scatterplot* ditampilkan pada Gambar 2.9.



Gambar 2.9 – Contoh *database segmentation* menggunakan *scatterplot* (Connoly and Begg, 2005)

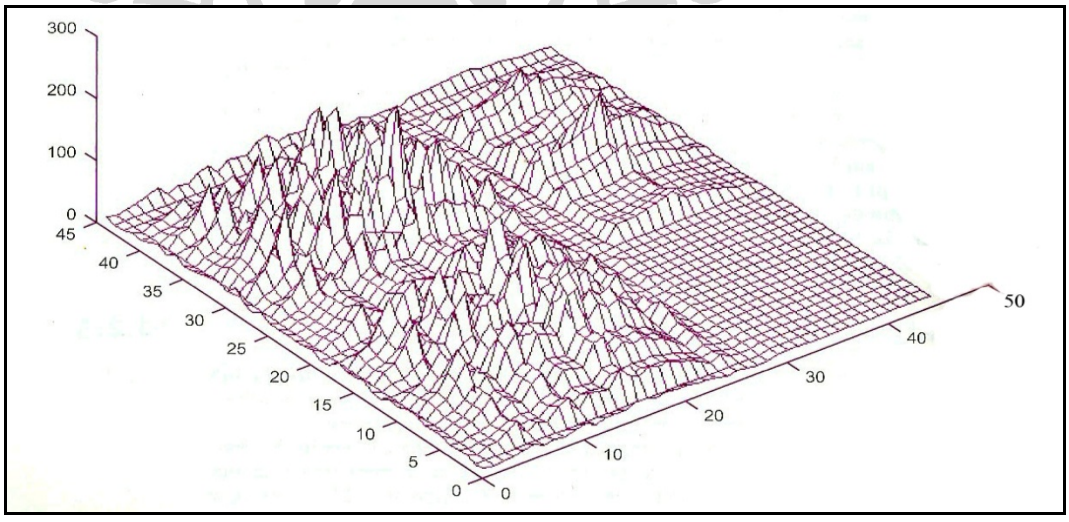
2.2.6 *Link Analysis*

Link Analysis memiliki sasaran untuk membangun jaringan yang dinamakan *associations* antara *individual records* atau *sets of records* di dalam *database*. Ada tiga spesialisasi dari analisis jaringan: *associations discovery*, *sequential pattern discovery* dan *similar time sequence discovery*. *Associations discovery* digunakan untuk menemukan item yang menyatakan keberadaan dari item yang lain didalam *event* yang sama. *sequential pattern discovery* menemukan *pattern* antara *event* seperti keberadaan satu set dari sekelompok *item* yang diikuti oleh satu set dari sekelompok *item* didalam *database* dalam beberapa periode waktu. *Similar time sequence discovery* digunakan seperti contoh: dalam

discovery of links antara dua set data yang bergantung terhadap waktu dan berdasarkan derajat kesamaan antara pola dalam suatu seri waktu.

2.2.7 *Deviation Detection*

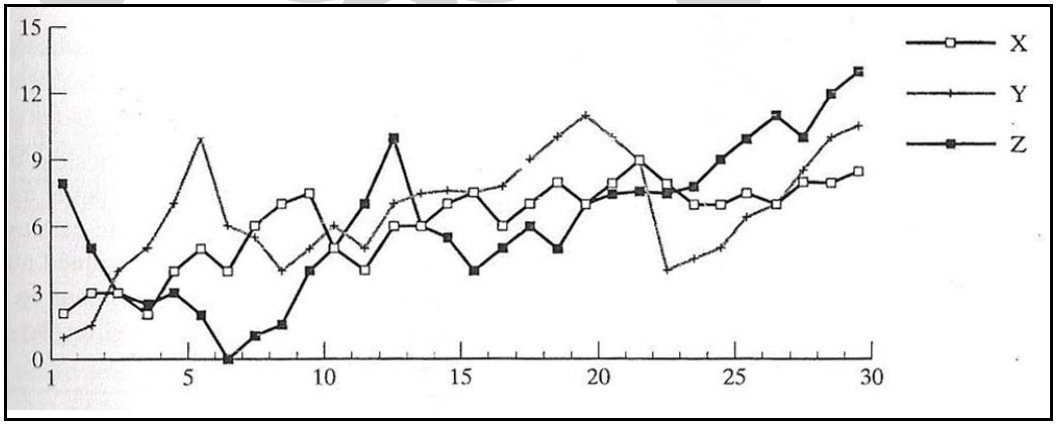
Deviation Detection adalah teknik yang relatif baru dalam teknik *data mining*. Namun *deviation detection* sering kali menjadi sumber dari penemuan baru karena teknik ini mengidentifikasi *outlier* yang mengekspresikan deviasi dari penemuan sebelumnya. Operasi ini ditampilkan menggunakan teknik *statistics* dan *visualizations* atau sebagai suatu produk dari *data mining.* Sebagai contoh regresi linear memfasilitasi pengidentifikasian data dalam teknik visualisasi modern yang menampilkan kesimpulan dan representasi grafik yang membuat deviasi mudah untuk dideteksi. Pada Gambar 2.10 ditampilkan gambar mengenai data yang diperlihatkan pada Gambar 2.9 dengan teknik visualisasi.



Gambar 2.10 – Contoh visualisasi dari data pada Gambar 2.9
(Connoly and Begg, 2005)

2.2.8 *Time Series Analysis*

Dengan menggunakan *Time Series Analysis,* nilai dari suatu atribut di amati dimana nilai tersebut berubah-ubah sesuai dengan waktu. Nilai ini biasanya diperoleh secara *time point* (harian, mingguan dan perjam dan lainnya). Suatu *time series plot* (Gambar 2.11). biasanya digunakan untuk mem-visualisasikan seri waktu. Pada Gambar 2.11 ini kita dengan mudah dapat melihat bahwa *plot* untuk Y dan Z memiliki kebiasaan yang sama, sedangkan X tampak memiliki kerentanan yang lebih sedikit. Ada tiga fungsi dasar yang ditampilkan pada *time series analysis*. Pada fungsi pertama perhitungan jarak digunakan untuk menentukan kesamaan antara *time series* yang berbeda. Pada fungsi kedua, struktur dari garis diamati untuk menentukan (dan mungkin mengklasifikasi kebiasaannya). Pada fungsi ketiga, struktur garis digunakan untuk *historical time series plot* untuk memprediksi nilai di masa akan datang.



Gambar 2.11 – *Time series plots*
(Dunham, 2003)

Definisi dari *time series* adalah suatu set dari atribut value dalam suatu periode waktu. Tipikal aplikasi *data mining* dengan *time series* termasuk menentukan kesamaan antara dua *time series* yang berbeda dan memprediksi nilai yang akan datang dari suatu atribut sesuai *time series* dari suatu nilai yang diketahui.

2.2.9 *Time Series Analysis Pattern*

Time Series Analysis dianalisis untuk menemukan pola dari data yang berurutan melalui observasi dan untuk memprediksi (*forecasting)* nilai yang akan datang melalui *time series variable*. Pola yang dideteksi termasuk:

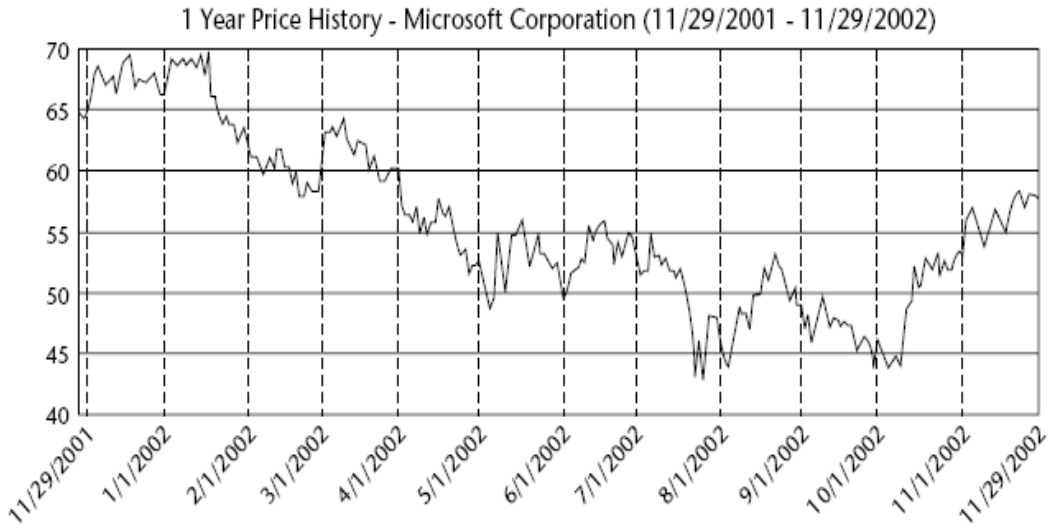
- *Trends*: suatu tren dapat dilihat sebagai suatu *systematic nonrepetitive changes (linear* atau *nonliniear)* kepada suatu nilai dalam suatu waktu.
- *Cycles*: disini perilaku yang diobservasi adalah siklus
- *Seasonal*: Pola yang terdeteksi berdasarkan waktu (tahun, bulan, atau hari)
- *Outliers*: untuk membantu mendeteksi pola, teknik mungkin digunakan untuk membuang atau mengurangkan pengaruh dari *outliers*

Mengidentifikasi pola pada *real data* sangat sulit, karena pengaruh dari *noise, outlier, errors* dan *missing data*. Banyak pola dapat diobservasikan ke data yang sama.

Suatu *time series* terdiri dari seri data yang dikumpulkan secara berturut-turut berdasarkan *incremental* waktu atau berdasarkan indikator yang berurutan lainnya. Banyak variabel lain yang merubah nilainya berdasarkan waktu. Nilai yang berurutan dari suatu variabel dalam suatu periode waktu membentuk *time*

series. Sebagai contoh, penutupan nilai saham harian *microsoft* adalah *time series* (lihat Gambar), nilai hasil penjualan perusahaan dalam periode kuartal dan nilai pendapatan dalam periode bulanan termasuk *time series*. Secara umum, kenaikan waktu dalam *time series* bisa diskrit atau berkelanjutan. Sebagai tambahan, nilai yang diobservasi dalam *time series* bisa diskrit atau berkelanjutan. Nilai penjualan dan nilai penghasilan perusahaan memiliki observasi yang berkelanjutan. *Time series* dari prakiraan cuaca dengan nilai cerah, mendung dan hujan memiliki observasi yang diskrit. Dalam berbagai literatur kebanyakan orang menggunakan *time series* untuk merujuk ke kasus dimana observasi berkelanjutan dan menggunakan *sequence* untuk merujuk ke suatu kasus dimana observasi diskrit.

Tujuan utama dari mengumpulkan data *time series* yaitu untuk *forecast* yaitu membuat prediksi terhadap nilai masa depan. Pada perusahaan telekomunikasi diperlukan suatu prediksi mengenai *traffic* dari jaringan yang dimiliki untuk keperluan perencanaan maintenance jaringan.



Gambar 2.12 – *Microsoft Stock Value*

2.2.10 Algoritma *Microsoft Time Series*

Yaitu algoritma untuk *forecasting* dalam bentuk hibrid yang menggabungkan teknik *autoregression* dan *decision tree.* Algoritma ini dinamakan ART (*AutoRegression Tree*). Dibawah ini akan dijelaskan lebih lanjut mengenai prinsip dari algoritma ini.

- ***Autoregression***

Autoregression adalah teknik popular berkaitan dengan *time series*. Suatu proses *autoregressive* merupakan nilai x dan waktu t(xt) yang merupakan fungsi dari nilai x pada waktu sebelumnya, sebagai contoh:

$$X_t = f(X_{t-1}, X_{t-2}, X_{t-3}, X_{t-n}) + \varepsilon_t$$

Dimana xt adalah *time series* yang diinvestigasi, dan n adalah urutan dari *autoregression* yang secara umum lebih sedikit dari panjang seri nya. Simbol yang terakhir *epsilon* merepresentasikan *noise*. Salah satu langkah kunci dari ART yaitu melakukan transformasi satu kasus *time series* menjadi beberapa kasus secara mendalam. Proses yang diilustrasikan pada Gambar 2.14 tabel sebelah kiri mengandung dua bentuk *time series* (2 kasus) dari penjualan bulanan susu dan roti. Tabel sebelah kanan adalah data yang sudah ditransformasi. Ada tujuh kolom didalam tabel: kolom pertama adalah ID, kemudian kolom kedua menunjukkan penjualan susu pada waktu t-2, kolom ketiga menunjukkan penjualan susu pada waktu t-1, kolom keempat menunjukkan penjualan susu pada waktu t. Tiga kolom terakhir berisikan informasi yang sama mengenai penjualan roti. Masing-masing baris dari tabel kanan merepresentasikan suatu kasus. Susu (t0) dan roti (t0) adalah dua

kolom yang diprediksi. Karena *decision tree* mendukung *regression*, kita bisa menggunakan teknik ini untuk memprediksi kedua kolom ini. Susu (t-1), susu (t-2), roti (t-1) dan roti (t-2) dianggap sebagai *regressor*.

Salah satu keuntungan dari kasus transformasi adalah semua *time series* di dalam model *mining* yang sama dikonversi ke kolom dalam tabel yang sama. Ketika menggunakan teknik *decison tree* untuk memprediksi susu (*t0*), semua kolom selain susu atau roti dianggap sebagai kolom input. Jika ada korelasi yang kuat antara penjualan roti dan susu, korelasi ini akan muncul pada fungsi *f*.

Tujuan dari algoritma *time series* adalah untuk menemukan fungsi *f*. Jika *f* adalah fungsi linear, kita akan memiliki rumus:

$$X_t = a_1 X_{t-1} + a_2 X_{t-2} + a_3 X_{t-3} + \ldots + a_n X_{t-n} + \varepsilon_t$$

Dimana ai adalah *autoregression coefficients*

Case Transform

Bulan	Susu	Roti
Jan-2005	5000	4500
Feb-2005	5200	4600
Mar-2005	5240	5130
Apr-2005	6390	6280
Mei-2005	6750	6160
Jun-2005	6280	6560
Jul-2005	7680	7200

Case Id	Susu (t-2)	Susu (t-1)	Susu (t-0)	Roti (t-2)	Roti (t-1)	Roti (t0)
1	5000	5200	5240	4500	4560	5130
2	5200	5240	6390	4600	5130	6280
3	5240	6390	6750	5130	6280	6120
4	6390	6750	6280	6280	6120	6560
5	6750	6280	6120	6160	6560	7200
…						

Gambar 2.13 – *Case Transformation*

Model ini sering digambarkan secara sederhana untuk menggambarkan *autoregression* (AR). Pertama kali dipublikasikan dan diselesaikan oleh Yuel pada tahun 1927. Secara verbal, *time series* saat ini bisa diperkirakan dengan

penjumlahan bobot linear dari kondisi *series* sebelumnya. Bobotnya adalah *autoregression coefficients*.

Banyak cara untuk menyelesaikan *autoregression coefficients*. Metode yang paling popular adalah untuk menyelesaikan ini adalah dengan meminimalisasikan rata² kuadrat selisih antara deret waktu yg dimodelkan Xnmodel dan deret waktu yg terpantau Xn. Proses meminimalisasi menghasilkan sistem persamaan linear dengan koefisien an, yang diketahui sebagai persamaan Yule-Walker berikut ini. Persamaan ini digunakan untuk menghitung koefisioen AR yang diberikan matrix yang diberikan pada Gambar 2.14.

$$\begin{bmatrix} 1 & r_1 & r_2 & r_3 & r_4 & \dots & r_{n-1} \\ r_1 & 1 & r_1 & r_2 & r_3 & \dots & r_{n-2} \\ r_2 & r_1 & 1 & r_1 & r_2 & \dots & r_{n-3} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ r_{n-1} & r_{n-2} & r_{n-3} & r_{n-4} & r_{n-5} & \dots & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \cdot \\ \cdot \\ \cdot \\ a_n \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ \cdot \\ \cdot \\ \cdot \\ r_n \end{bmatrix}$$

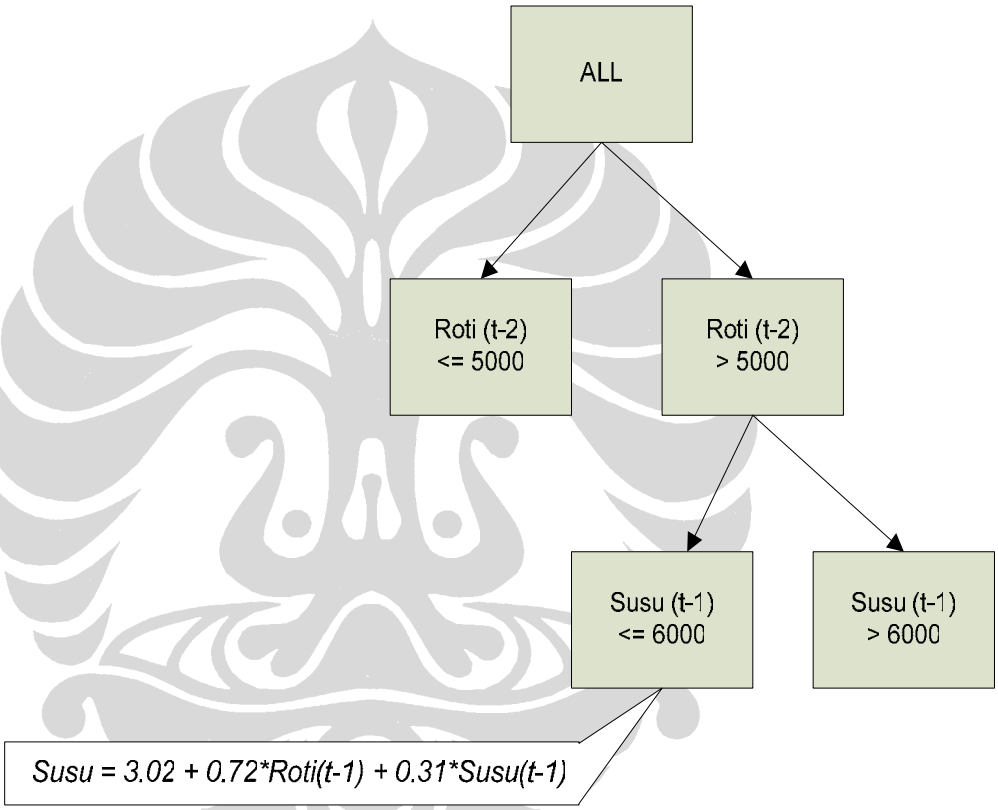
Gambar 2.14 – Matrix koefisien dimana r_d *autocorrelation coefficient* dengan delay d

- ***Autoregression Tress***

Algoritma microsoft time series adalah model *autoregressive* dimana fungsi *f* berkorespondensi kepada suatu *regression tree*. Sebagaimana telah disampaikan sebelumnya, publikasi dari algoritma ini adalah *AutoRegression Trees* (ART). Didalam ART fungsi *f* direpresentasikan dengan *regression tree*. Gambar 2.15 menampilkan *regression tree* yang dibangun menggunakan data

time series pada Gambar 2.13. Pembagian *tree* pertama terjadi pada penjualan roti 2 bln lalu. Jika penjualan roti lebih dari 110, dua bulan lalu akan ada pembagian *tree* yang lain terdapat lagi pembagian *tree* selanjutnya pada penjualan susu bulan lalu. Jika dalam kasus bulan lalu, penjualan susu kurang dari 120 dan formula regresi untuk susu yaitu:

3.02 + 0.72 * Bread(t-1) + 0.31 * Milk(t-1).



Gambar 2.15 – *Regression tree* pada data *time series*

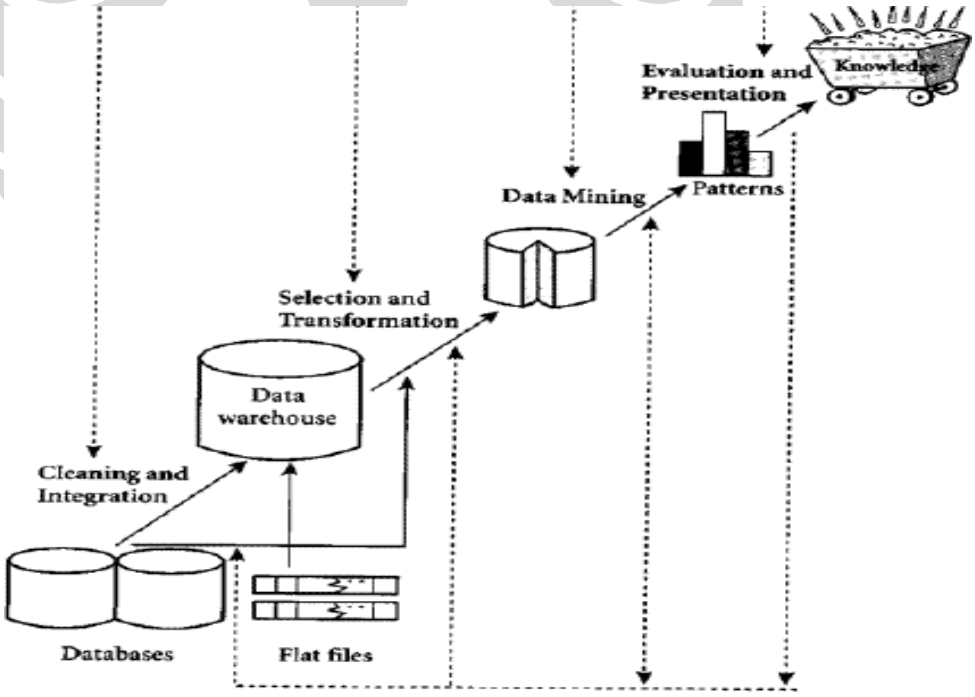
2.2.11 RapidMiner

RapidMiner adalah suatu *data mining tools* yang menggunakan konsep pengelolaan data dan pemodalan proses yang memberikan kemudahan dalam melakukan konfigurasi proses kepada penggunanya. Dengan adanya tampilan yang jelas dan penggunaan bahasa *script* menggunakan XML membuat *RapidMiner* menjadi suatu lingkungan pengembangan aplikasi yang baik untuk *data mining* dan *machne learning*.

Untuk melakukan *time series analysis* menggunakan *RapidMiner* dapat menggunakan referensi operator *data preprocessing* yaitu menggunakan operator *Series2WindowExamples* dan operator *MultivariateSeries2WindowExamples*. *Series2WindowExamples* digunakan utnuk melakukan prediksi dengan mentransformasi suatu contoh set yang mengandung suatu seri data ke suatu nilai tunggal. Untuk tujuan ini *windows* dengan nilai *window* dan *step size* (besar nilai lompatan) yang telah ditentukan sebelumnya akan digunakan untuk menganalisis data input (seri data) satu persatu untuk memprediksi nilai yang akan diprediksi. Sedangkan untuk *MultivariateSeries2WindowExamples* fungsi dan kegunaannya sama dengan *Series2WindowExamples*, perbedaannya hanya pada *MultivariateSeries2 WindowExamples* dapat menangani *multivariate series data.* Baru kemudian diterapkan filter yang akan digunakan untuk mendapatkan nilai prediksi dengan menggunakan *time series enalysis*, seperti menerapkan filter *linear regression*, *neural network,* SVM dan lainnya.

2.2.12 Tahapan *Data Mining*

Untuk mendapatkan hasil yang baik dan bermanfaat dalam proses menggali informasi yang tersembunyi di *data warehouse* perusahaan, diperlukan suatu proses *data mining* yang terstruktur dengan baik. Dimana proses *data mining* ini sendiri terdiri dari tahapan-tahapan yang mempunyai umpan balik dari setiap tahapan ke tahapan sebelumnya sehingga dalam setiap tahapan terjadi evaluasi terhadap output yang dihasilkan. Jika terjadi hasil yang tidak diinginkan maka dapat melakukan tahapan yang dilakukan sebelumnya sampai memperoleh hasil yang terbaik dari setiap tahapan. Lebih jelasnya mengenai tahapan *data mining* diilustrasikan pada Gambar 2.16 berikut ini:



Gambar 2.16 – Tahapan *Data Mining*

Berdasarkan ilustrasi diatas terlihat bahwa *data mining* adalah satu tahapan dari beberapa tahapan yang tergabung dalam tahapan *data mining*, berikut penjelasan dari setiap tahapan *data mining* (Jiawei Han and Micheline Kamber, 2001):

- *Cleansing Data*

  Cleansing Data atau pembersihan data yaitu tahapan untuk membuang data yang tidak konsisten dan menghilangkan *noise*. Data yang diperoleh dengan melakukan ekstraksi dari sumber data seperti *operational data store* dan *flat file* (contoh: *sheet file*) biasanya memiliki isian yang tidak sempurna seperti data yang tidak lengkap, data yang tidak valid ataupun data yang tidak konsisten seperti perbedaan penulisan nama, kode id dan lainnya. Data seperti ini sudah sebaiknya dibuang karena dengan keberadaannya dapat mengurangi kualitas data pada tahap-tahap berikutnya sehingga memberikan hasil yang tidak akurat pada tahap akhir (*garbage in garbage out*). Melalui pembersihan data ini juga akan mempengaruhi kinerja dari proses tahapan *data mining* karena data yang diproses akan berkurang jumlahnya dan kompleksitasnya

- Integrasi Data

  Data yang diperlukan dalam tahapan *data mining* tidak berasal dari satu sumber saja melainkan dari beberapa *database* dan *sheet file*. Proses yang harus dilakukan untuk menggabungkan sejumlah sumber data yang berbeda memerlukan pengerjaan yang cermat karena *fiekd* dari masing-masing sumber data bisa saja dalam penamaan yang berbeda tetapi maksudnya adalah sama. Tahap integrasi data juga melibatkan proses

transformasi yang kemudian dilanjutkan dengan proses *loading* ke *data warehouse*.

- Seleksi Data

  Hasil dari tahap sebelumnya *cleansing* dan integrasi data yang sudah disimpan di dalam *data warehouse*, pada tahap seleksi data dilakukan pemilihan dan pengambilan terhadap data yang relevan terhadap proses analisis yang akan dilakukan

- Transformasi Data

  Pada transformasi data sebelum data bisa langsung diaplikasikan pada tahap *data mining* hal yang dilakukan adalah data ditransformasikan ke bentuk yang tepat. Transformasi data dapat melibatkan hal-hal berikut ini:

  - *Data type transform*: hal ini adalah transformasi data yang paling sederhana. Sebagai contoh transformasi tipe kolom *Boolean* ke *integer*. Alasan untuk transformasi ini adalah beberapa algoritma *data mining* kinerjanya lebih baik jika menggunakan data *integer*, dilain pihak lebih baik menggunakan *boolean*
  - *Smoothing*: berguna untuk membuang *noise* dari data seperti *binning* yang membagi-bagi angka numerik dalam beberapa interval
  - *Aggregation*: dimana *summary* atau *aggregation* di aplikasikan ke data contohnya data *traffic* daily diaggregasikan untuk menghitung total *traffic* dalam satu bulan

- *Generalization*: generalisasi data dengan konsep hierarki, seperti atribut kategori *city* yang bisa digeneralisasi ke level lebih tinggi yaitu *country*
- *Missing value handling*: kebanyakan set data memiliki nilai yang hilang, banyak hal yang menyebabkan hal ini seperti ketika data datang dari dua sumber yang berbeda, menggabungkan dua sumber data ini dapat menyebabkan adanya nilai yang hilang karena kedua sumber tersebut tidak memiliki definisi atribut yang sama. Contoh lain adalah nilai yang ada memang kosong karena memang tidak ada data pengukuran yang diterima. Menghadapi permasalahan seperti ini dapat mengganti nilai yang kosong dengan nilai konstanta yang umum atau menggantinya dengan perhitungan tertentu.

- *Data Mining*

Dalam *data mining* terdapat tiga tahap yang dilakukan yaitu:

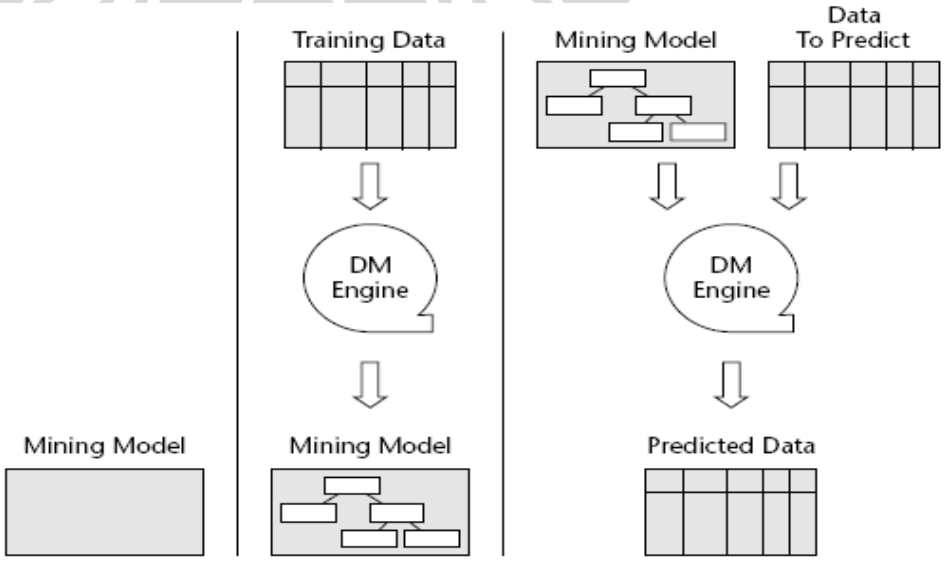
1. Tahap pertama adalah membuat *mining model*. Hal ini serupa dengan membuat tabel di dalam *database* relasional. Definisi dari *mining model* termasuk menentukan jumlah kolom yang akan digunakan sebagai input, kolom yang diprediksi dan algoritma yang digunakan. *Mining model* sendiri adalah suatu kontainer yang sama dengan tabel relasional. *Mining model* digunakan untuk menyimpan pola yang ditemukan oleh algoritma *data mining*

2. Tahap yang kedua dari *data mining* adalah *model training* (disebut juga pemrosesan). Pada tahap ini kita memberikan data historical

ke *data mining engine*. Contoh nya adalah data elemen jaringan dan informasi *traffic* jaringannya selama satu tahun. Dalam tahap *training*, algoritma *data mining* memulai untuk menganalisa masukan data. Bergantung dari efisiensi dari suatu *algoritma,* hal ini akan mempengaruhi proses *scanning* set data yang digunakan apakah memerlukan satu iterasi saja atau lebih untuk menemukan korelasi antar nilai atribut

3. Tahap ketiga dari *data mining* adalah *prediction*. Dalam rangka untuk memprediksi, diperlukan *trained mining model* dan data set yang baru. Ketika prediksi, *data mining engine* menerapkan aturan yang ditemukan dari tahap *training* ke data set yang baru dan menempatkan hasil prediksi untuk masing-masing kasus inputan.

Untuk lebih jelasnya tiga tahap tersebut dapat dilihat pada Gambar 2.17 berikut ini



Gambar 2.17 – Tiga Tahap *Data Mining*

- Evaluasi Pola

  Yang dilakukan pada tahapan ini adalah melakukan identifikasi dan evaluasi terhadap penemuan pola-pola menarik yang dapat memberikan gambaran pengetahuan terhadap sesuatu variabel pengukuran yang menarik.

- Presentasi

  Pada tahap ini teknik visualisasi dan teknik untuk merepresentasikan suatu pengetahuan digunakan untuk menyampaikan pengetahuan yang digali melalui *data mining* kepada pengguna.

2.3 Penelitian di Bidang *Data Warehouse* dan *Data Mining*

Penggunaan *data warehouse* dan *data mining* telah diterapkan pada sejumlah penelitian yang dilakukan di dunia. Seperti penelitian dengan judul "*Data Warehouse*: *A Telecommunications Business Solution"* (Papaiacovou, 2000) yang menjadikan *data warehouse* sebagai solusi bisnis bagi perusahaan telekomunikasi yang kemudian diterapkan pada perusahaan *Pacific Bell* yang merupakan salah satu perusahaan telekomunikasi terbesar di Amerika.

Permasalahan pada *Pacific Bell* terjadi ketika ada kebutuhan pengguna untuk mengakses sejumlah data dengan jumlah banyak yang tersebar pada *database* yang berbeda atau dalam bentuk *file* sedangkan hanya memiliki waktu yang terbatas. Masalah menjadi semakin rumit ketika sejumlah data yang diperoleh tersebut diharuskan digabungkan menjadi satu laporan seperti dalam bentuk laporan bulanan. *Data warehouse* menjadi solusi yang paling tepat untuk

mengatasi permasalahan ini. Permasalahan serupa juga terjadi pada perusahaan *Telecom Italia*. Pada penelitian dengan judul "*Data Integration and Warehousing in Telecom Italia"* (Trisolini, 1999) dijabarkan mengenai kebutuhan perusahaan *Telecom Italia* untuk mengintegrasikan 50 *internal database* yang berisikan data telekomunikasi ke *data warehouse* agar analisis dapat dilakukan lebih mendalam dan hasil analisis dapat dilakukan dengan lebih detil dan cepat.

Penelitian yang menggunakan *data mining* salah satunya adalah penelitian dengan judul "*Data Mining for Managing Quality of Service in Digital Mobile Telecommunications Networks*" (Vehvilainen, 2004). Penelitian ini membahas mengenai pengaplikasian teknik *data mining* pada data kinerja jaringan GSM untuk mengelola dan meningkatkan *quality of service* dari jaringan telekomunikasi. Penelitian serupa dengan judul "*Data Mining in Telecommunications"* (Weiss, 2004) membahas bagaimana *data mining* bisa digunakan untuk memunculkan informasi berharga yang tersembunyi didalam data telekomunikasi yang berguna untuk mengidentifikasi *telecommunication fraud,* meningkatkan efektifitas pemasaran dan mengidentifikasi *network fault*. Beberapa penelitian *data mining* yang lain membahas lebih spesifik bagaimana suatu teknik *data mining* dapat memberikan manfaat pada perusahaan. Seperti penelitian dengan judul "*Sequence Mining for Customer Behavior Predictions in Telecommunications"* (Eichnger, 2006) yang membahas mengenai penggunaan *data mining* untuk memprediksi perilaku pelanggan perusahaan telekomunikasi dan penelitian dengan judul "*Data Mining and Forecasting in Large-Scale Telecommunication Networks*" (Sasisekharan, 1996) mengenai penggunaan *data mining* untuk memprediksi kondisi jaringan perusahaan telekomunikasi AT&T.