BAB IV

HASIL DAN PEMBAHASAN

4.1 *Requitment Definition*

4.1.1 Pendefinisian Proyek

Pendefinisian proyek meliputi pengertian dari perangkat lunak atau proyek yang akan dibuat. Perangkat lunak yang akan dibuat adalah suatu aplikasi yang mampu mengelompokkan data barang sesuai dengan tingkat kecenderungannya muncul bersamaan dalam suatu transaksi dengan menggunakan Algoritma Apriori, selain itu juga sebagai bahan keputusan dalam menganalisa *market basket* agar meningkatkan penjualan. Sehingga aplikasi ini dapat membantu pemilik lembaga bisnis dan manajer perusahaan untuk menganalisis pola konsumsi yang dimiliki pelanggannya.

Sistem ini menggunakan analisis asosiasi, yaitu mendefinisikan suatu proses untuk menemukan semua aturan asossiatif yang memenuhi syarat minimum untuk *support* (*minimum support*) dan syarat minimum untuk *confidence* (*minimum confidence*). *Support* adalah suatu ukuran yang menunjukan seberapa besar tingkat dominasi suatu *item/itemset* dari keseluruhan transaksi. Ukuran ini menentukan apakah suatu *item/itemset* layak untuk dicari *confidence*-nya (misal, dari keseluruhan transaksi yang ada, seberapa besar tingkat dominasi

yang menunjukan bahwa item A dan B dibeli bersamaan), sedangakan *Confidence* adalah suatu ukuran yang menunjukkan hubungan antar 2 item secara *conditional* (misal, seberapa sering item B dibeli jika orang membeli item A).

4.1.2 Kebutuhan Sistem

Sistem yang dibangun ini diharapkan dapat membantu pemilik lembaga bisnis dan manajer perusahaan dalam menganalisa *market basket* agar dapat meningkatkan penjualan. Kebutuhan sistem ini untuk user adalah:

1. Sistem ini dapat mengolah *database* transaksi penjualan dengan menggunakan Algoritma Apriori.
2. Sistem ini dapat mengkombinasikan *item-item* yang dapat dibeli secara bersamaan dan menghitung banyaknya transaksi yang terjadi dalam tiap kombinasi tersebut.
3. Sistem ini menghasilkan *output* dari *item-item* yang saling berasosiasi dengan menghitung besarnya *support* dan *confidence*.
4. Sistem ini dapat memberikan informasi *item-item* atau produk yang sering dibeli konsumen, sehingga dapat mengetahui pola konsumsi konsumen.

Dalam membangun system ini dibutuhkan:

1. Perangkat Lunak / *Software*

   Dalam mengimplementasikan penelitian ini dibutuhkan *software* yang dapat mengkoneksikan langsung ke *database*. Pemrograman yang digunakan adalah Lazarus 0.9.28.2. Lazarus adalah pemrograman yang bersifat gratis (*free*) dan *open source* yang berfungsi sebagai alat untuk kompilator *Free* Pascal. IDE Lazarus (*screenshot*) adalah lingkungan pemrograman yang kaya *fitur* untuk

membuat aplikasi *stand alone* grafis dan konsol serta stabil. Lazarus dapat dipakai pada *Linux*, *Mac OS X* dan *Win32* dan menyediakan sumber editor yang disesuaikan dan bentuk (*form*) visual diciptakan sesuai dengan paket manajer, *debugger* dan memiliki integrasi GUI yang lengkap dengan *compiler Free Pascal.* (Wiki. 2010)

Free Pascal adalah sebuah kompiler yang berjalan pada banyak sistem operasi. Didesain untuk mengkompilasi *source code* dalam bahasa *Object Pascal* sebuah penambahan dari bahasa pemrograman Pascal. Berbeda dengan Java yang dirancang supaya *write once* dan *run anywhere* sedangkan Lazarus dan *Free* Pascal dirancang supaya *write once* dan *compile everywhere*. Karena kompiler yang sama tersedia untuk semua sistem operasi di atas sehingga tidak dibutuhkan *coding* ulang untuk menghasilkan produk untuk *platform-platform* yang berbeda, kecuali jika menggunakan *fitur* yang tergantung pada sistem operasi tertentu, serta didukung dengan *Cross-compiling*.

Lazarus mulai versi 0.9.26.2 sudah sangat stabil dan bisa dibandingkan dengan Delphi 7. Kelebihan Lazarus jika dibandingkan dengan Delphi adalah sebagai berikut:

1. *Open Source* dan *Free*.
2. *Multiplatform*, mendukung *Windows*, *Linux*, *Mac OS* dan *Pocket PC*.
3. Bisa menghasilkan *code* 64-bit.
4. Dikembangkan oleh komunitas *open source* sehingga berkembang dengan sangat pesat.

(Sujatmiko, Arief. 2009)

Selain menggunakan Pemrograman Lazarus, penelitian ini membutuhkan aplikasi *database* yang berguna untuk menyimpan data. Aplikasi *database* yang digunakan adalah SQL Server 2000. SQl Server 2000 merupakan salah satu produk DBMS (*Database Management System*) yang dibuat oleh Microsoft. SQL Server 2000 menawarkan beberapa *fitur* di dalam mengelola *database*, ada 2 *fitur* yang biasa digunakan untuk mengelola *database* di dalam SQL Server 2000, yaitu:

1. Menggunakan *Enterprise Manager*

   Fitur ini relatif mudah digunakan karena mode pengelolaannya berbasis GUI (*Graphical User Interface*). Oleh karena itu, cukup dengan metode *clik* dan *drag*. Dapat membuat *database* dan tabel serta manajemen database yang lain dengan mudah.

2. Menggunakan SQl *Query Analyzer*

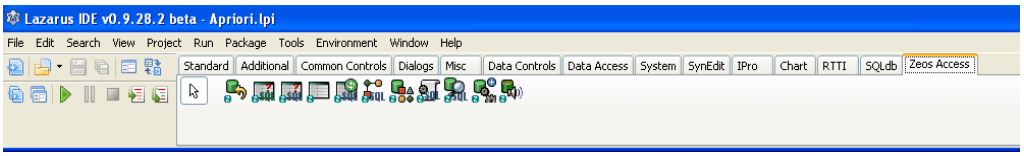
   Fitur ini menggunakan *Transact* SQL (perintah-perintah SQL) untuk mengelola *database* di dalam SQL Server 2000. Perintah-perintah *Transact* SQL merupakan pengembangan dari perintah-perintah SQL standar yang disesuaikan dengan manajemen *database* pada SQL Server. *Transact* SQL memungkinkan untuk membuat *database*, membuat tabel, mengubah struktur tabel, menghapus *database*, menghapus tabel, menyisipkan data, mengubah data, dan lain-lain.

   (Bunafit Nugroho dan Indah Indriyanna, 2007)

Untuk mengkoneksikan pemrograman Lazarus dan SQL Server 2000 dibutuhkan komponen pustaka dengan menginstal paket tambahan pada Lazarus 0.9.28. Komponen tersebut adalah ZeosDBO, fungsi dan cara

penggunaannya sama dengan BDE atau ADO di Delphi. ZeosDBO adalah pustaka komponen Lazarus yang berguna untuk koneksi ke *database*. (Sujatmiko, Arief. 2010)

Berikut ini gambar ZeosDBO yang telah terinstal:

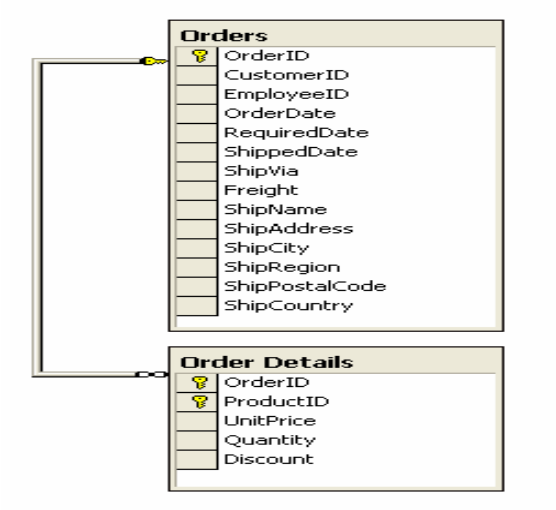


Gambar 4.1 Komponen ZeosDBO

2. Perangkat Keras / *Hardware*, dengan spesifikasi:
   a. Processor Intel Core 2 Duo
   b. RAM 1 GB dan Harddisk 120 GB
   c. Monitor dan VGA
   d. Mouse dan Keyboard

4.2 Software Requitment Analysis

Penelitian ini mencoba untuk mengimplementasikan Algoritma Apriori dalam mendapatkan hubungan antar data dengan aturan assosiasi. Penelitian ini menggunakan Northwind, yaitu *database* yang sudah ada di dalam SQL Server 2000. Hasil dari penelitian ini berupa aplikasi yang dapat menghasilkan hubungan antar produk dalam *database* Northwind dengan aturan assosiasi. Data yang diolah dalam *database* ini adalah data pada tabel *Orders* dan tabel *Order Details* yang saling berelasi. Berikut ini adalah gambar relasi dari ke dua tabel:



Gambar 4.2 Relasi Tabel

Tabel *Orders* menyimpan transaksi yang terjadi dalam suatu perusahaan sedangkan tabel *Order Details* menjelaskan *procedur* apa saja yang terbeli dalam masing-masing transaksi pada tabel *Orders*. Berikut ini adalah data pada tabel *Orders*:

	OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	Freight	ShipName
1	10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000	1996-07-16 00:00:00.000	3	32.3800	Vins et alcools
2	10249	TOMSP	6	1996-07-05 00:00:00.000	1996-08-16 00:00:00.000	1996-07-10 00:00:00.000	1	11.6100	Toms Spezialität
3	10250	HANAR	4	1996-07-08 00:00:00.000	1996-08-05 00:00:00.000	1996-07-12 00:00:00.000	2	65.8300	Hanari Carnes
4	10251	VICTE	3	1996-07-08 00:00:00.000	1996-08-05 00:00:00.000	1996-07-15 00:00:00.000	1	41.3400	Victuailles en s
5	10252	SUPRD	4	1996-07-09 00:00:00.000	1996-08-06 00:00:00.000	1996-07-11 00:00:00.000	2	51.3000	Suprêmes délices
6	10253	HANAR	3	1996-07-10 00:00:00.000	1996-07-24 00:00:00.000	1996-07-16 00:00:00.000	2	58.1700	Hanari Carnes
7	10254	CHOPS	5	1996-07-11 00:00:00.000	1996-08-08 00:00:00.000	1996-07-23 00:00:00.000	2	22.9800	Chop-suey Chines
8	10255	RICSU	9	1996-07-12 00:00:00.000	1996-08-09 00:00:00.000	1996-07-15 00:00:00.000	3	148.3300	Richter Supermar
9	10256	WELLI	3	1996-07-15 00:00:00.000	1996-08-12 00:00:00.000	1996-07-17 00:00:00.000	2	13.9700	Wellington Impor
10	10257	HILAA	4	1996-07-16 00:00:00.000	1996-08-13 00:00:00.000	1996-07-22 00:00:00.000	3	81.9100	HILARION-Abastos
11	10258	ERNSH	1	1996-07-17 00:00:00.000	1996-08-14 00:00:00.000	1996-07-23 00:00:00.000	1	140.5100	Ernst Handel
12	10259	CENTC	4	1996-07-18 00:00:00.000	1996-08-15 00:00:00.000	1996-07-25 00:00:00.000	3	3.2500	Centro comercial
13	10260	OTTIK	4	1996-07-19 00:00:00.000	1996-08-16 00:00:00.000	1996-07-29 00:00:00.000	1	55.0900	Ottilies Käselad
14	10261	QUEDE	4	1996-07-19 00:00:00.000	1996-08-16 00:00:00.000	1996-07-30 00:00:00.000	2	3.0500	Que Delícia
15	10262	RATTC	8	1996-07-22 00:00:00.000	1996-08-19 00:00:00.000	1996-07-25 00:00:00.000	3	48.2900	Rattlesnake Cany
16	10263	ERNSH	9	1996-07-23 00:00:00.000	1996-08-20 00:00:00.000	1996-07-31 00:00:00.000	3	146.0600	Ernst Handel
17	10264	FOLKO	6	1996-07-24 00:00:00.000	1996-08-21 00:00:00.000	1996-08-23 00:00:00.000	3	3.6700	Folk och fä HB
18	10265	BLONP	2	1996-07-25 00:00:00.000	1996-08-22 00:00:00.000	1996-08-12 00:00:00.000	1	55.2800	Blondel père et
19	10266	WARTH	3	1996-07-26 00:00:00.000	1996-09-06 00:00:00.000	1996-07-31 00:00:00.000	3	25.7300	Wartian Herkku
20	10267	FRANK	4	1996-07-29 00:00:00.000	1996-08-26 00:00:00.000	1996-08-06 00:00:00.000	1	208.5800	Frankenversand
21	10268	GROSR	8	1996-07-30 00:00:00.000	1996-08-27 00:00:00.000	1996-08-02 00:00:00.000	3	66.2900	GROSELLA-Restaur
22	10269	WHITC	5	1996-07-31 00:00:00.000	1996-08-14 00:00:00.000	1996-08-09 00:00:00.000	1	4.5600	White Clover Mar
23	10270	WARTH	1	1996-08-01 00:00:00.000	1996-08-29 00:00:00.000	1996-08-02 00:00:00.000	1	136.5400	Wartian Herkku
24	10271	SPLIR	6	1996-08-01 00:00:00.000	1996-08-29 00:00:00.000	1996-08-30 00:00:00.000	2	4.5400	Split Rail Beer
25	10272	RATTC	6	1996-08-02 00:00:00.000	1996-08-30 00:00:00.000	1996-08-06 00:00:00.000	2	98.0300	Rattlesnake Cany
26	10273	QUICK	3	1996-08-05 00:00:00.000	1996-09-02 00:00:00.000	1996-08-12 00:00:00.000	3	76.0700	QUICK-Stop
27	10274	VINET	6	1996-08-06 00:00:00.000	1996-09-03 00:00:00.000	1996-08-16 00:00:00.000	1	6.0100	Vins et alcools
28	10275	MAGAA	1	1996-08-07 00:00:00.000	1996-09-04 00:00:00.000	1996-08-09 00:00:00.000	1	26.9300	Magazzini Alimen

Tabel 4.1 Tabel *Orders*.

Pada tabel *Orders* ini menyimpan banyaknya data transaksi yang terjadi. Banyaknya data transaksi yang terjadi sebanyak 830 transaksi, dengan *primary key* terletak di *field* OrderID. Tabel ini memiliki 14 *field*, yaitu *OrderID*, *CostumerID*, *EmployeeID*, *OrderDate*, *RequiredDate*, *ShippedDate*, *ShipVia*, *Freight*, *ShipName*, *ShipAddress*, *ShipCity*, *ShipRegion*, *ShipPostalCode*, dan *ShipCountry*.

Berikut ini adalah data pada tabel *Order Details*:

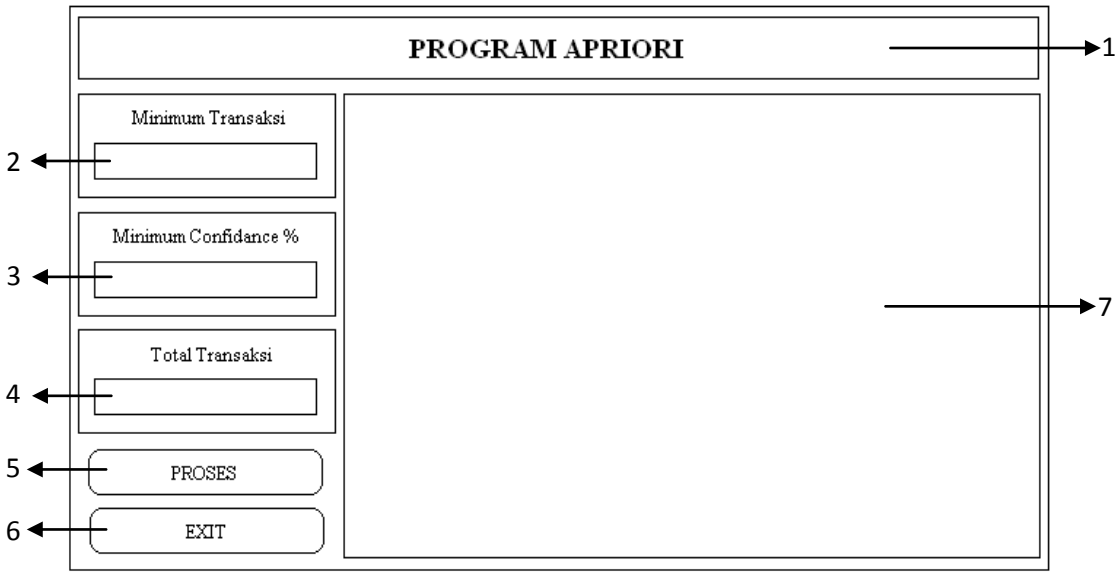
	OrderID	ProductID	UnitPrice	Quantity	Discount
1	10248	11	14.0000	12	0,
2	10248	42	9.8000	10	0,
3	10248	72	34.8000	5	0,
4	10249	14	18.6000	9	0,
5	10249	51	42.4000	40	0,
6	10250	41	7.7000	10	0,
7	10250	51	42.4000	35	0,15
8	10250	65	16.8000	15	0,15
9	10251	22	16.8000	6	5,e-002
10	10251	57	15.6000	15	5,e-002
11	10251	65	16.8000	20	0,
12	10252	20	64.8000	40	5,e-002
13	10252	33	2.0000	25	5,e-002
14	10252	60	27.2000	40	0,
15	10253	31	10.0000	20	0,
16	10253	39	14.4000	42	0,
17	10253	49	16.0000	40	0,
18	10254	24	3.6000	15	0,15
19	10254	55	19.2000	21	0,15
20	10254	74	8.0000	21	0,
21	10255	2	15.2000	20	0,
22	10255	16	13.9000	35	0,
23	10255	36	15.2000	25	0,
24	10255	59	44.0000	30	0,
25	10256	53	26.2000	15	0,
26	10256	77	10.4000	12	0,
27	10257	27	35.1000	25	0,
28	10257	39	14.4000	6	0,

Tabel 4.2 Tabel *Order Details*

Pada tabel *Order Details* ini berisi data apa saja yang terbeli dalam masing-masing transaksi pada tabel *Orders*. Tabel ini berelasi dengan tabel *Orders* dengan *foreign key* pada *field* OrderID, dan memiliki *primary key* pada *field ProductID*. Tabel ini memiliki 5 *field*, yaitu *OrderID*, *ProductID*, *UnitPrice*, *Quantity*, dan *Discount*.

4.3 Desain

Tahap ini mendesain program yang akan dibuat sesuai dengan analisis proyek dan kebutuhan yang ada. Program yang akan dibuat ini akan mengolah atau memproses data menjadi sumber informasi yang dapat digunakan untuk pengambilan keputusan dalam menganalisa *market basket* menggunakan Algoritma Apriori. Berikut ini adalah rancangan form yang digunakan dalam implementasi Algoritma Apriori.



Gambar 4.3 Desain Program Apriori

Keterangan desain:

1. Tampilan judul dari form yaitu 'PROGRAM APRIORI'
2. Tampilan Minimum Transaksi, digunakan sebagai batasan untuk mengetahui jumlah kombinasi item yang mungkin dalam *itemset*.
3. Tampilan Minimum *Confidence* %, digunakan sebagai batasan untuk menentukan aturan asosiasi yang terbentuk dalam bentuk %.

4. Tampilan Total Transaksi, digunakan untuk menampilkan banyaknya transaksi yang dilakukan.
5. Tampilan dari *button* Proses, digunakan untuk melakukan perintah pemrosesan data.
6. Tampilan dari *button Exit*, digunakan untuk menutup aplikasi.
7. Tampilan memo, digunakan sebagai keluaran atau hasil *output* dari proses yang dilakukan.

4.4 *Coding*

Setelah mendesain, tahap selanjutnya yaitu menerjemahkan desain tersebut ke dalam kode-kode dengan menggunakan bahasa pemrograman yang sudah ditentukan. Dalam pembuatan perangkat lunak ini menggunakan bahasa pemrograman Lazarus 0.9.28.2 dengan *database*-nya Northwind pada SQL Server 2000. Di bawah ini akan dijelaskan mengenai bagaimana cara implementasi *data mining* pada pengambilan keputusan dalam menganalisis *market basket* menggunakan Algoritma Apriori dengan menggunakan bahasa pemrograman Lazarus dan *database* Northwind pada SQL Server 2000.

4.4.1 Algoritma Apriori

Algoritma Apriori adalah algoritma paling terkenal untuk menemukan pola frekuensi tinggi. Pola frekuensi tinggi adalah pola-pola item di dalam suatu *database* yang memiliki frekuensi atau *support* di atas ambang batas tertentu yang disebut dengan istilah *minimum support*. Pola frekuensi tinggi ini digunakan

untuk menyusun aturan assosiatif dan juga beberapa teknik data mining lainnya. Algoritma Apriori dibagi menjadi beberapa tahap pada iterasi ke-*k*, yaitu:

1. Pembentukan kandidat *itemset.*

   Kandidat *k-itemset* dibentuk dari kombinasi (k-1)-*itemset* yang didapat di itersi sebelumnya. Salah satu ciri Algoritma Apriori adalah adanya pemangkasan kandidat *k-itemset* yang subset-nya yang berisi k-1 *item* tidak termasuk dalam pola frekuensi tinggi dengan panjang k-1.

2. Perhitungan *support* dari tiap kandidat *k-itemset*.

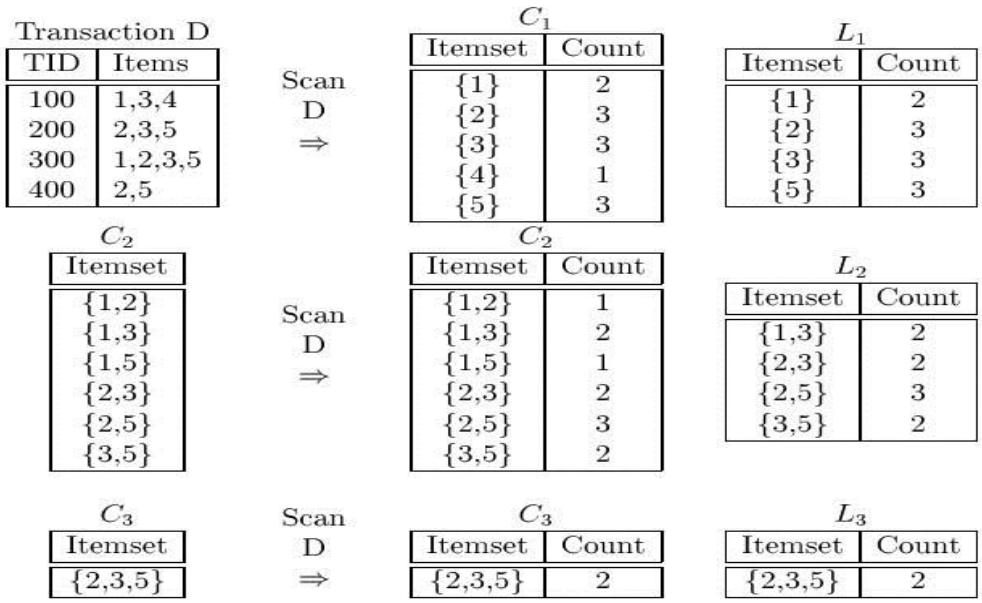
   Support dari setiap kandidat *k-item* set didapat dengan men-*scan database* untuk menghitung jumlah transaksi yang memuat semua *item* di dalam kandidat *k-itemset* tersebut. Ini adalah juga termasuk ciri dari Algoritma Apriori dimana diperlukan perhitungan dengan *scan* seluruh *database* sebanyak *k-itemset* terpanjang.

3. Tetapkan pola frekuensi tinggi.

   Pola frekuensi tinggi yang memuat k *item* atau *k-itemset* ditetapkan dari kandidat *k-itemset* yang *support*-nya lebih besar dari minimum *support*.

4. Bila tidak didapat pola frekuensi tinggi baru maka seluruh proses dihentikan. Bila tidak, maka k ditambah satu dan kembali ke bagian pertama.

Contoh dari penerapan Algoritma Apriori diilustrasikan di gambar di bawah ini:

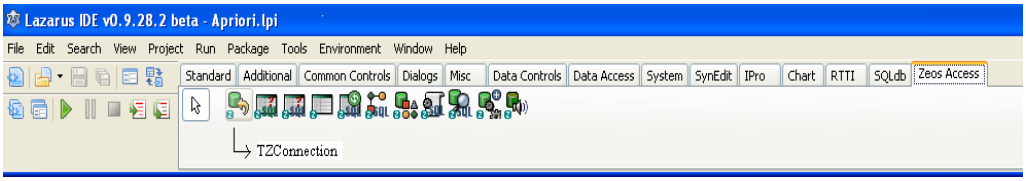


Gambar 4.4 Ilustrasi Penerapan Algoritma Apriori

4.4.2 Pengkoneksian Lazarus ke *database* Northwind pada SQL Server 2000

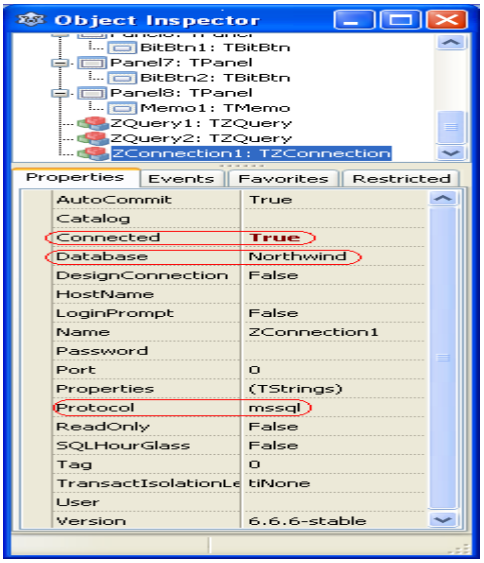
Untuk mengkoneksikan pemrograman Lazarus dan SQL Server 2000 dibutuhkan komponen tersendiri dengan menginstal paket tambahan pada Lazarus 0.9.28. Komponen tersebut adalah ZeosDBO. Komponen ini diinstal ke pemrograman lazarus dengan cara mengkompile file 'zcomponent.lpk'. Setelah menginstal komponen ini maka dapat dikoneksikan pemrograman yang akan dibuat dengan *database* Northwind pada SQL Server 2000 dengan cara sebagai berikut:

1. Mengklik komponen ZeosDBO yaitu *Zeos Access* pada *palete*, kemudian *doubleclick tool TZConnection*. Berikut ini tampilan dari *tool* TZConnection:



Gambar 4.5 Tool TZConnection

2. Kemudian klik *ZConnection* yang telah dipilih, selanjutnya *edit properties* dari *ZConnection* tersebut yang ada di *Object Inspector*. Ganti *Protocol* dengan 'mssql' karena aplikasi *database* yang digunakan adalah SQL Server 2000.dan pilih *database* Northwind, kemudian ganti *Connected* dengan nilai '*true*'. Jika hasilnya '*true*' maka pemrograman berhasil terkoneksi dengan *database* Northwind pada SQL Server 2000. Berikut ini tampilan dari *properties* dari *ZConnection*:



Gambar 4.6 Cara Mengkoneksikan Lazarus 0.9.28 dengan SQL Server 2000

4.4.3 Procedure Pemrograman

Pemrograman apriori ini dibagi menjadi 3 *procedure*, yaitu:

1. *Procedure* untuk *button*1 atau 'proses'.

   Procedure ini melakukan perintah dalam mengolah *database*, melakukan perintah *query* yang dimanfaatkan untuk mendapatkan kombinasi item yang mungkin dalam *itemset* dengan menerapkan Algoritma Apriori.

2. *Procedure* untuk *button*2 atau ‘exit’.

   Procedure ini melakukan perintah keluar atau menutup aplikasi. Berikut ini *coding* dari *procedure exit*:

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
  Close;

end;
```

3. *Procedure* untuk menampilkan from

   Procedure ini melakukan perintah untuk menampilkan *form*, di dalam program ini yang akan ditampilkan adalah *form* ‘Memo’. Berikut ini *coding* dari *procedure* tampil *form*:

```
procedure TForm1.FormShow(Sender: TObject);
begin
  C := 0;
  Memo1.Clear;

end;
```

4.4.4 Coding Input

Pemrograman Apriori yang akan dibuat ini memiliki 3 inputan, yaitu:

1. Input nilai minimum transaksi

   Pengguna atau *user* menginput nilai minimum transaksi sebagai batasan untuk mengetahui jumlah kombinasi item yang mungkin dalam *itemset*. Dalam *coding* program untuk minimum transaksi diberi nilai integer.

```
min_transaksi : integer;

min_transaksi := StrToInt(EdtMinTransaksi.text);
```

2. Input berapa persen minimum *confidence*

   Minimum *confidence* digunakan sebagai batasan untuk menentukan aturan asosiasi yang terbentuk. Dalam *coding* program untuk minimum *confidence* diberi nilai real.

```
min_confidence : real;
min_confidence := StrToFloat(EdtMinConfidence.Text);
```

3. Input banyaknya total transaksi yang terjadi

   Banyaknya total transaksi yang terjadi ini diinput langsung dari total transaksi yang ada di tabel *Orders*. Jadi program langsung membaca data, pengguna atau *user* tidak perlu menginput total transaksi. *Coding* program untuk total transaksi adalah:

```
EdtTotalTransaksi.Text := Fields[0].AsString;
total_transaksi := Fields[0].AsInteger;
```

4.4.5 Coding Output

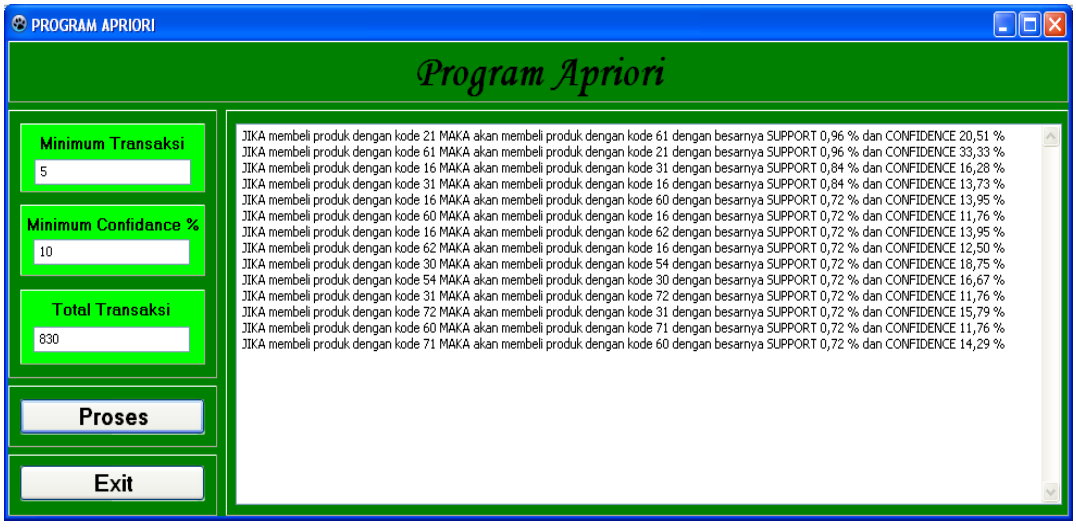
Output dari pemrograman apriori ini adalah perhitungan *support* dan *confidence*, dengan syarat jika minimum *confidence* terpenuhi. *Support* adalah suatu ukuran yang menunjukan seberapa besar tingkat dominasi suatu *item/itemset* dari keseluruhan transaksi. Ukuran ini menentukan apakah suatu *item/itemset* layak untuk dicari *confidence*-nya (misal, dari keseluruhan transaksi yang ada, seberapa besar tingkat dominasi yang menunjukan bahwa item A dan B dibeli bersamaan). Sedangkan *Confidence* adalah suatu ukuran yang menunjukkan hubungan antar 2 item secara *conditional* (misal, seberapa sering item B dibeli jika orang membeli item A).

Berikut ini adalah *coding* untuk *output*:

```
if FieldByName('jumlah').AsInteger*100/
ZQuery2.FieldByName('jumlah').AsInteger >= min_confidence then
begin
  antecedent := 'JIKA membeli produk dengan kode ';
  for k := 1 to i do
  begin
    if k = j then
    konklusi := ' MAKA akan membeli produk dengan kode '+ Fields[k-1].AsString +
    ' dengan besarnya SUPPORT '+
     FormatFloat('0.00', FieldByName('jumlah').AsInteger/
     total_transaksi*100) + ' %' +
    ' dan CONFIDENCE '+
     FormatFloat('0.00',FieldByName('jumlah').AsInteger * 100 /
     ZQuery2.FieldByName('jumlah').AsInteger) + ' %'
    else if ((j = 1) and (k>2)) or ((j>1) and (k>1)) then
     antecedent := antecedent + ', ' + Fields[k-1].AsString
    else antecedent := antecedent + Fields[k-1].AsString;
    end;
Memo1.Lines.Add(antecedent + konklusi);
end;
```

4.5 *Testing*

Pemrograman yang telah dibuat kemudian diuji secara keseluruhan (*system testing*). Berikut ini adalah tampilan hasil *running* serta *testing* dari *execution* Apriori.exe:



Gambar 4.7 *Running* dan *Testing* Program Apriori

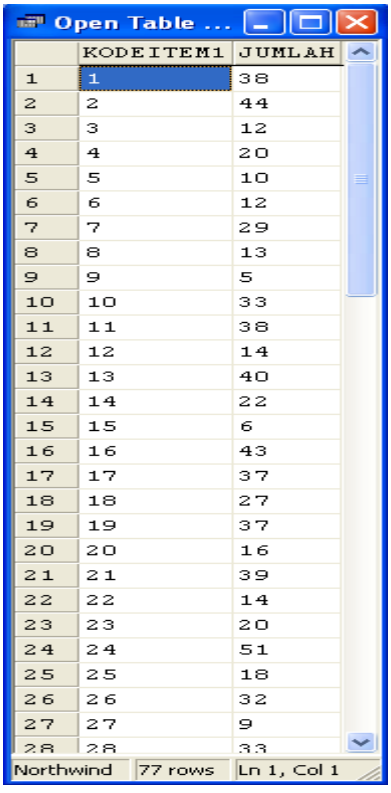
Aturan asosiasi yang diperoleh adalah sebagai berikut:

1. JIKA membeli produk dengan kode 21 MAKA akan membeli produk dengan kode 61 dengan besarnya SUPPORT 0,96 % dan CONFIDENCE 20,51 %
2. JIKA membeli produk dengan kode 61 MAKA akan membeli produk dengan kode 21 dengan besarnya SUPPORT 0,96 % dan CONFIDENCE 33,33 %
3. JIKA membeli produk dengan kode 16 MAKA akan membeli produk dengan kode 31 dengan besarnya SUPPORT 0,84 % dan CONFIDENCE 16,28 %
4. JIKA membeli produk dengan kode 31 MAKA akan membeli produk dengan kode 16 dengan besarnya SUPPORT 0,84 % dan CONFIDENCE 13,73 %
5. JIKA membeli produk dengan kode 16 MAKA akan membeli produk dengan kode 60 dengan besarnya SUPPORT 0,72 % dan CONFIDENCE 13,95 %
6. JIKA membeli produk dengan kode 60 MAKA akan membeli produk dengan kode 16 dengan besarnya SUPPORT 0,72 % dan CONFIDENCE 11,76 %
7. JIKA membeli produk dengan kode 16 MAKA akan membeli produk dengan kode 62 dengan besarnya SUPPORT 0,72 % dan CONFIDENCE 13,95 %
8. JIKA membeli produk dengan kode 62 MAKA akan membeli produk dengan kode 16 dengan besarnya SUPPORT 0,72 % dan CONFIDENCE 12,50 %
9. JIKA membeli produk dengan kode 30 MAKA akan membeli produk dengan kode 54 dengan besarnya SUPPORT 0,72 % dan CONFIDENCE 18,75 %
10. JIKA membeli produk dengan kode 54 MAKA akan membeli produk dengan kode 30 dengan besarnya SUPPORT 0,72 % dan CONFIDENCE 16,67 %
11. JIKA membeli produk dengan kode 31 MAKA akan membeli produk dengan kode 72 dengan besarnya SUPPORT 0,72 % dan CONFIDENCE 11,76 %

12. JIKA membeli produk dengan kode 72 MAKA akan membeli produk dengan kode 31 dengan besarnya SUPPORT 0,72 % dan CONFIDENCE 15,79 %
13. JIKA membeli produk dengan kode 60 MAKA akan membeli produk dengan kode 71 dengan besarnya SUPPORT 0,72 % dan CONFIDENCE 11,76 %
14. JIKA membeli produk dengan kode 71 MAKA akan membeli produk dengan kode 60 dengan besarnya SUPPORT 0,72 % dan CONFIDENCE 14,29 %

Aturan-aturan tersebut diperoleh dengan langkah-langkah sebagai berikut:

1. Mengambil *ProductId* dan frekuensi transaksi terhadap produk tersebut dari tabel *Order Details* dengan syarat memenuhi minimum transaksi yang ditentukan oleh *user* pada *form* aplikasi, dalam hal ini nilai yang di-*input* sebesar 5 kemudian hasilnya dimasukkan ke dalam tabel C1. Berikut ini tampilan dari tabel C1 :

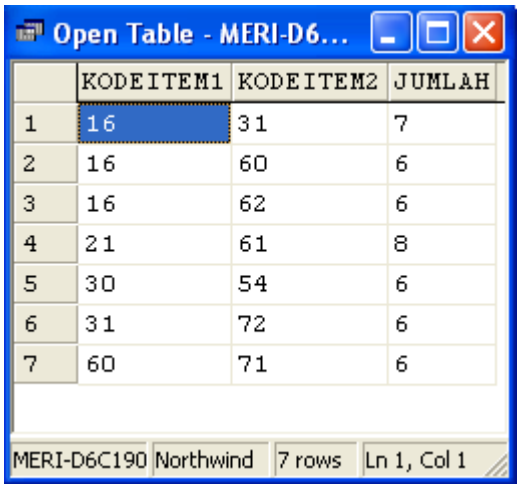

Open Table ...

	KODEITEM1	JUMLAH
1	1	38
2	2	44
3	3	12
4	4	20
5	5	10
6	6	12
7	7	29
8	8	13
9	9	5
10	10	33
11	11	38
12	12	14
13	13	40
14	14	22
15	15	6
16	16	43
17	17	37
18	18	27
19	19	37
20	20	16
21	21	39
22	22	14
23	23	20
24	24	51
25	25	18
26	26	32
27	27	9
28	28	33

Northwind | 77 rows | Ln 1, Col 1

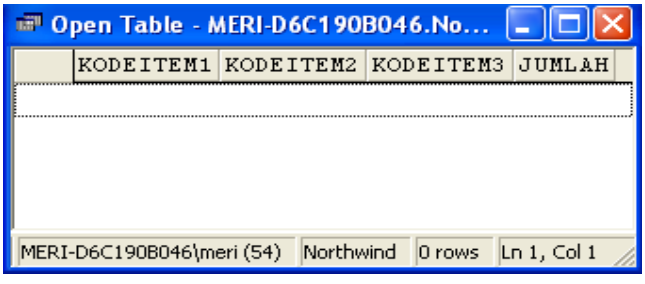
Tabel 4.3 Tabel C1

2. Membuat kombinasi *item-item* pada tabel C1 dan dimasukkan ke dalam tabel C2 menjadi calon 2-*itemset*. Pada langkah ini dilakukan pula penghitungan frekuensi transaksi yang mengandung kombinasi *item-item* tersebut. Kombinasi *item* yang memiliki frekuensi transaksi kurang dari nilai minimum_transaksi dihapus dari tabel C2. Berikut ini tampilan dari tabel C2 :



Tabel 4.4 Tabel C2

3. Membuat kombinasi *item-item* pada tabel C2 dan dimasukkan ke tabel C3 menjadi calon 3-*itemset*. Seperti pada langkah 2, dilakukan penghitungan frekuensi transaksi yang mengandung kombinasi *item* dari calon k-*itemset*, dan kombinasi yang tidak memenuhi nilai minimum transaksi dihapus dari tabel C3. Namun, isi dari tabel C3 kosong, yang artinya tidak ada kombinasi *item* yang memuhi syarat minimum transaksi. Berikut ini tampilan dari tabel C3 :



Tabel 4.5 Tabel C3

4. Karena yang memenuhi syarat minimum transaksi ada pada tabel C2 maka dari tabel C2 dibentuk aturan asosiasi. Aturan yang berbentuk:
   jika membeli [kodeitem1] maka akan membeli [kodeitem2] dan sebaliknya
   jika membeli [kodeitem2] maka akan membeli [kodeitem1]

5. Dari masing-masing bentuk aturan asosiasi pada seluruh *record* yang ada di tabel C2 dilakukan perhitungan nilai *confidence*. Bagi aturan yang memenuhi syarat minimum *confidence* akan ditampilkan, sedangkan yang tidak memenuhi tidak ditampilkan. Dalam hal ini, minimum *confidence* ditentukan sebesar 10%.

4.6 *Maintenance*

Setelah jadi dan diujicobakan serta sesuai yang diinginkan, maka diperlukan pengoperasikan program di lingkungannya dan melaksanakan pemeliharaan, seperti penyesuaian atau perubahan karena adaptasi dengan situasi sebenarnya. Dalam penelitian ini, tidak dilakukan *maintenance* dikarenakan pemrograman yang telah dihasilkan ini belum disebarluaskan atau dipersentasikan ke swalayan, perusahaan atau bidang transaksi penjualan lainnya yang membutuhkan pengolahan data sebagai bahan keputusan dalam menganalisa *market basket* agar meningkatkan penjualan.