DAFTAR PUSTAKA

1. Andayani Sri. *Pembentukan Cluster dalam Knowledge Discovery In Database dengan Algoritma K-Means.* Yogyakarta: Jurusan Pendidika Matematika FMIPA UNY, 2007.
2. Architect Chieft, Guerra Joseph, President Vice. *Why You Need a Data Warehouse*. Chesire: Andrews Consulting Groups, 2011.
3. Bank Data Kesehatan Nasional: *Pusat Data dan Indormasi Departemen Kesehatan Nasional*. Tersedia (www.bankdata.depkes.go.id), 2013.
4. Canlas Jr Ruben D. *Data Mining In Healthcare : Current Application And Issues.* Australia: Carniege Mellon University, 2009.
5. Chen Ming-Syan, Han Jiawei, Yu Philip S. *Data Mining : A Overview From Database Perpective.* Canada: IEEE Transactions on Knowledge and Data Engineering, 1996.
6. Dharmaputeri Endah. *Aplikasi Sistem Informasi Geografis Pelayanan Kesehatan Kota Depok Berbasis Web Menggunakan Quantum GIS.* Depok: Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Gunadarma, 2009.
7. Endang Indriasih. *Sistem Informasi Geografis Dalam Bidang Kesehatan Masyarakat*. Jakarta: Buletin Penelitian Sistem Kesehatan Pada Pusat Penelitian dan Pengembangan Sistem dan Kebijakan Kesehatan, 2008.

8. Erwin. *Analisis Market Basket Dengan Algoritma Apriori dan FP-Growth*. Ogan Ilir: Jurnal Generic Vol.4 No.2 , 2009.
9. Esri. *Geographic Information Systems and Environmental Health: Incorporating Esri Technology and Services.* New York St.: Esri, 2011.
10. Gerber Jason, Glass Michael K, Naramore Elizabeth, Scouarnec Yann Le, Stolz Jeremy. *PHP 5, Aphace, MySQL Web Development*. Indianapolis: Wiley Publishing, 2005.
11. Hariadi Mochammad, Ratnasari Dwi, Sumpeno Surya. *Klasifikasi Text Mining Dalam Game Tutor Cerdas "Dasar Listrik" Menggunakan Metode Naïve Bayes Classifier*. Surabaya: Seminar Nasional Pascasarjana XI Program Pasca Sarjana Institut Sepuluh November, 2011.
12. Hermawati Fajar Astuti. *Data Mining.* Yoyakarta: Andi Publiser, 2013.
13. Jackson Joyce. *Data Mining : A Conceptual Overview.* South Carolina: Management Science Departement University Of South Carolna, 2002.
14. Kadir Abdul. *Mastering Ajax dan PHP.* Yoyakarta: Andi Publiser, 2009.
15. Kropla Bill. *Beginning MapServer. USA: Apress, 2005.*
16. Kusrini, Lutfhi Taufiq Emha. *Algoritma Data Mining*. Yogyakarta: CV. Andi Offset, 2009.

17. Lukito Hendra. *Perumusan Pola Penyebaran Demam Berdarah Melalui Data Mining Pada Database Dinas Kesehatan DKI Jakarta*. Bogor: Institut Pertanian Bogor, 2007.

18. Map Server: *Open Source Web Mapping*. Tersedia (www.mapserver.org), 2013.

19. OpenLayers: *Free Map for the Web*. Tersedia (www.openlayers.org),2013.

20. Prabhu C.S.R. *Data Warehousing Concepts, Techniques, Products and Applications Second Edition*. India: Rajkarnal Electric Press, 2006.

21. Ruliansyah Andri.*Sistem Informasi Geografis Untuk Pemetaan Daerah rawan Demam Berdarah Dengue.* Yogyakarta: Universitas Gajah Mada, 2011.

22. Santosa Budi.*Aplikasi Data Mining Dalam Bidang Kesehatan Studi Kasus Demam Berdarah*. Surabaya: Institut Teknologi Sepuluh November, 2009.

23. Shadiq M. Ammar. *Keoptimalan Naïve Bayes Dalam Klasifikasi.* Universitas Pendidikan Indonesia, 2009.

24. Sugiarti Yuni. *Analisis dan Perancagan UML (Unified Modeling Language).* Yoyakarta: Graha Ilmu, 2013.

25. Syaukani Muhammad. *Mengelola Data Pada MySQL Server*. Jakarta: PT Elex Media Komputindo, 2005.

26. Velicanu Manole. *Building a Data Warehouse step by step.* Romania: Academia de Studii Economice din Bucureşti. - Vol. XI., 2007.

27. Wu Juinjie. *Advances In K-Means Clustering.* Beijing: *Departement Of Management Science and Engineering School Of Economics And Management Tsinghua University*, 2012.

LAMPIRAN

1. Source Code *Class KMean*

```
class KMean{
   public $objek = array();
   public $cluster = array(array());
  public function __construct($daftarObjek,$jumlahCluster) {
      //Membuat Objek Untuk Setiap Baris Data dan Penentuan Centroid Cluster Awal
      $c=0;
      $in=0;
      $counterCentroid = floor(count($daftarObjek)/$jumlahCluster);
      $tmpObjek1 = array(array());
      $tmpObjek2 = array(array());
      //Pembuatan Objek
      for ($i=0;$i<count($daftarObjek);$i++){
         $this->objek[$i] = new ObjekKMean($daftarObjek[$i]);
         for ($j=0;$j<count($daftarObjek[$i]);$j++){
            $tmpObjek1[$j][$i] = $daftarObjek[$i][$j];
         }
      }
      //Penentuan Centroid Awal
      for ($i=0;$i<count($tmpObjek1);$i++){
         sort($tmpObjek1[$i]);
      }
      for ($i=0;$i<count($tmpObjek1);$i++){
         for ($j=0;$j<count($tmpObjek1[$i]);$j++){
            $tmpObjek2[$j][$i] = $tmpObjek1[$i][$j];
         }
      }
      for ($i=0;$i<count($tmpObjek2);$i++){
         if ($i==$c){
            if ($in<$jumlahCluster){
               for ($j=0;$j<count($tmpObjek2[$i]);$j++){
                  $this->cluster[$in][$j] = $tmpObjek2[$i][$j];
               }
               $in++;
            }
            $c+=$counterCentroid;
         }
      }
      ObjekKMean::setUpdateCluster($this->cluster);
      //Melakukan Iterasi Untk Penentuan Cluster Objek
      $this->setClusterObjek(0);
   }
```

```
//Menset Ulang Centroid Cluster Berdasarkan Objek Terbarunya
 private function setCentroidCluster(){
   for ($i=0;$i<count($this->cluster);$i++){
      $countObj = 0;
      $x = array();
      for ($j=0;$j<count($this->objek);$j++){
         if ($this->objek[$j]->clusterObjek==$i){
            for ($k=0;$k<count($this->objek[$j]->data);$k++){
               $x[$k] += $this->objek[$j]->data[$k];
            }
            $countObj++;
         }
      }
      for ($k=0;$k<count($this->cluster[$i]);$k++){
         if ($countObj>0){
            $this->cluster[$i][$k] = $x[$k]/$countObj;
         }
      }
   }
   ObjekKMean::setUpdateCluster($this->cluster);
 }
//Iterasi Ubtuk Penentuan Cluster Objek
 private function setClusterObjek($itr){
   $cek = TRUE;
   for ($i=0;$i<count($this->objek);$i++){
      //Cluster Awal Objek
      $clusterAwal = $this->objek[$i]->clusterObjek;
      //Cluster terbaru
      $this->objek[$i]->setClusterObjek();
      //Membanding cluster awal objek dengan cluster terbaru
      if ($clusterAwal!=  $this->objek[$i]->clusterObjek){
         $cek = FALSE;
      }
   }
   //Jika Cluster Awal Seluruh Objek <> Cluster terbaru
   if ((!($cek))&&($itr<20)){
      //Menentukan Centroid Cluster
      $this->setCentroidCluster();
      //Melakukan Rekursif
      $this->setClusterObjek($itr+1);
   }
 }
}
```

2. Source Code *Class ObJekKMean*

```
class ObjekKMean{
   static $cluster = array();
   public $data = array();
   public $clusterObjek;
   function __construct($d) {
      $this->data = $d;
   }
   static function setUpdateCluster($c){
      ObjekKMean::$cluster = $c;
   }
   function setClusterObjek(){
      //Menentukan Cluster dari Objek
      $jml = 0;
      $tmpCluster = 0;
      $tmpC = 0;
      $c = null;
      for ($i=0;$i<count(ObjekKMean::$cluster);$i++){
         $jml = 0;
         for ($j=0;$j<count($this->data);$j++){
            $jml += pow(($this->data[$j] - ObjekKMean::$cluster[$i][$j]),2);
         }
         $tmpC = sqrt($jml);
         if ($c==null){
            $c = $tmpC;
            $tmpCluster = $i;
         }
         if ($tmpC < $c){
            $c = $tmpC;
            $tmpCluster = $i;
         }
      }
      $this->clusterObjek=$tmpCluster;
   }
}
```

3. Source Code *Class NaiveBayes*

```
class NaiveBayes {
   var $dataTraining=array(array());
   var $dataIndikator=array();
   var $dataKategori=array();
   var $kategoriDataTraining=array();
   var $kategoriMean = array(array());
   var $kategoriStandarDeviasi = array(array());
   var $kmean;
   function __construct($data, $kategori, $jmlKelas){
      //Inisialisasi Indikator
      for($i=0;$i<count($data[0]);$i++){
         $this->dataIndikator[$i] = $i;
      }
      //Inisialisasi Data Training
      $this->dataTraining = $data;
      //Inisialisasi Daftar Kelas Kategori
      for ($i=0;$i<$jmlKelas;$i++){
         $this->dataKategori[$i] = $i;
      }
      //Membuat Kelas Kategori dari Data Kontinu dengan K-Mean
      $kat=array();
      for ($i=0;$i<count($kategori);$i++){
         $kat[$i] = array($kategori[$i]);
      }
      $this->kmean = new KMean($kat, $jmlKelas);
      //Mengatur Ulang Kelas Dengan Data Kategori
      for($i=0;$i<count($kategori);$i++){
         $this->kategoriDataTraining[$i] = $this->kmean->objek[$i]->clusterObjek;
      }
      //Mengatur Mean dan Standar Deviasi Masing2 Indikator terhadap Masing2 Kategori
      $this->setKategoriMeanStandarDeviasi();
   }
   function getKategoriDataUji($data){
      //Mencari Likelyhood dari setiap kategori
      $likelyhood = array();
      for ($i=0;$i<count($this->dataKategori);$i++){
         $lh = 0;
         for ($j=0;$j<count($this->dataIndikator);$j++){
            if ($j==0)
               $lh = $this->getDentitasGauss($data[$j], $this->kategoriMean[$i][$j], $this-
>kategoriStandarDeviasi[$i][$j]);
            else
               $lh *= $this->getDentitasGauss($data[$j], $this->kategoriMean[$i][$j], $this-
>kategoriStandarDeviasi[$i][$j]);
         }
         $likelyhood[$i] = $lh;
      }
```

```
    //Mencari Probabilitas setiap kategori
    $probabilitas = array();
    for($i=0;$i<count($this->dataKategori);$i++){
        $akum=0;
        for ($j=0;$j<count($likelyhood);$j++){
            if ($i<>$j) $akum +=$likelyhood[$j];
        }
        if ($akum!=0){
            $probabilitas[$i] = $likelyhood[$i]/$akum;
        }else{
            $probabilitas[$i] = $likelyhood[$i]/1;
        }
    }
    //Penentuan Kategori
    $kat=0;
    $tmp=0;
    for($i=0;$i<count($this->dataKategori);$i++){
        if ($i==0){
            $tmp = $probabilitas[$i];
        }else{
            if ($probabilitas[$i]>$tmp){
                $kat = $i;
                $tmp = $probabilitas[$i];
            }
        }
    }
    return $kat;
}

function setKategoriMeanStandarDeviasi(){
    for($i=0;$i<count($this->dataKategori);$i++){
        for ($j=0;$j<count($this->dataIndikator);$j++){
            $jml=0;
            $n=0;
            $dataStandarDeviasi=array();
            for($k=0;$k<count($this->dataTraining);$k++){
                if ($this->kategoriDataTraining[$k]==$this->dataKategori[$i]){
                    $jml += $this->dataTraining[$k][$j];
                    $dataStandarDeviasi[$n]=$this->dataTraining[$k][$j];
                    $n++;
                }
            }
            if ($n==0){
                $mean = 0;
                $standarDeviasi=0;
            }else{
                $mean = $jml/$n;
                $standarDeviasi = $this->getStandarDeviasi($dataStandarDeviasi);
            }
            $this->kategoriMean[$i][$j] = $mean;
            $this->kategoriStandarDeviasi[$i][$j] = $standarDeviasi;
        }
    }
}
```

```
function getStandarDeviasi($daftarData){
    $n = count($daftarData);
    $jumlah=0;
    for ($i=0;$i<count($daftarData);$i++){
      $jumlah += $daftarData[$i];
    }
    $rata2 = $jumlah/$n;
    $akum=0;
    for($i=0;$i<count($daftarData);$i++){
      $akum += (pow(($daftarData[$i]-$rata2),2));
    }
    return (sqrt($akum/($n-1)));
  }

  function getDentitasGauss($data,$mean,$standarDeviasi){
    //e = 2,7183
    //Phi = 3,1416
    if ($standarDeviasi==0){
      return 0;
    }else{
      $ret = (1/((sqrt(2*3.1416))*$standarDeviasi))*
          (pow((2.7183),(-0.5*(pow(($data-$mean)/$standarDeviasi,2)))));
      return $ret;
    }
  }
}
```

4. Source Code *Class FpGrowth*

```
class FpGrowth {
    var $dataCluster = array(array());
    var $countIndikator = array();
    var $dataTransaction = array(array());
    var $indikatorTransaction = array(array());
    var $path = array();
    var $fpTree;
    var $asosiasiIndikator=array();
    var $kmean;
    function __construct($data, $minimumSupport) {
        //Tahap Cluster Data Indikator Menjadi 2 Cluster
        $this->dataCluster = $data;
        for ($i=0;$i<count($data[0]);$i++){
            //Membuat Data Untuk K-Mean
            $tmpData = array(array());
            for ($j=0;$j<count($data);$j++){
                $tmpData[$j][0] = $data[$j][$i];
            }
            $this->setClusterData($tmpData, $i);
        }
        //Set Support Count Masing-masing Indikator;
        $this->setSupportCountIndikator();
        //Mengurutkan Indikator Berdasarkan Support Count Indikator
        arsort($this->countIndikator);
        //membuat Data Transaksi Baru Berdasarkan Minimum Support dan Urutan Support Count Indikator
        for ($i=0;$i<count($this->dataCluster);$i++){
            $j=0;
            foreach($this->countIndikator AS $index=>$count){
                if ($minimumSupport<=($count/count($this->dataCluster))){
                    $this->dataTransaction[$i][$j] = $this->dataCluster[$i][$index];
                    $j++;
                }
            }
        }
        $j=0;
        foreach($this->countIndikator AS $index=>$count){
            if ($minimumSupport<=($count/count($this->dataCluster))){
                $this->indikatorTransaction[$j] = $index;
                $j++;
            } }
        //Membuat FP-Tree
        FpTree::setTree($this->dataTransaction);
        //Buat Path
        $this->setAsosiasiPath(FpTree::$node);
        //Asosiasi Setiap Indikator
        $this->setAsosiasiIndikator($minimumSupport);
    }
```

```
function setClusterData($data,$kolom){
    $kmean = new KMean($data, 2);
    for ($i=0;$i<count($data);$i++){
      $this->dataCluster[$i][$kolom] = $kmean->objek[$i]->clusterObjek;
    }
  }
  function setSupportCountIndikator(){
    //Inisialisasi Jumlah Indikator
    for ($i=0;$i<count($this->dataCluster[0]);$i++){
      $this->countIndikator[$i] = 0;
    }
    for ($i=0;$i<count($this->dataCluster[0]);$i++){
      for ($j=0;$j<count($this->dataCluster);$j++){
        if ($this->dataCluster[$j][$i]==1){
          $this->countIndikator[$i]++;
        }
      }
    }
  }
  function setAsosiasiPath($tree){
    $cekSinglePath=TRUE;
    for($i=0;$i<count($this->indikatorTransaction);$i++){
      $tmp = 0;
      for ($j=0;$j<count($tree);$j++){
        if ($tree[$j]->level==$i){
          $tmp++;
        }
      }
      if ($tmp>1){
        $cekSinglePath=FALSE;
        break;
      }
    }
    if ($cekSinglePath){
      $tmpPath = array();
      $tmpSupportCount=0;
      for ($i=(count($this->indikatorTransaction)-1);$i>=0;$i--){
        for ($k=0;$k<count($tree);$k++){
          if ($tree[$k]->level==$i){
            if (count($tmpPath)==0) $tmpSupportCount = $tree[$k]->supportCount;
            array_push($tmpPath,$tree[$k]->indexIndikator);
          }
        }
      }
      if (count($tmpPath)>1){
        $cekPath=FALSE;
        for ($t=0;$t<count($this->path);$t++){
          if ($this->path[$t]->indexIndikator==$tmpPath){
            $this->path[$t]->supportCount+=$tmpSupportCount;
            $cekPath=TRUE;
            break;
          }
        }
        if ($cekPath==FALSE){
          $index = count($this->path);
          $this->path[$index] = new Path($tmpPath,$tmpSupportCount);
        }
      }
```

```
}else{
        $tmpTree=array();
        $label = array("a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p");
        for ($i=(count($this->indikatorTransaction)-1);$i>=0;$i--){
           for ($j=0;$j<count($tree);$j++){
              if (($tree[$j]->level==$i)&&(count($tmpTree)==0)){
                 array_push($tmpTree, $tree[$j]);
                 //Menghapus Node
                 unset($tree[$j]);
                 break;
              }
              if (($tree[$j]->level==$i)&&($tmpTree[count($tmpTree)-1]->parent==$tree[$j]->id)){
                 array_push($tmpTree, $tree[$j]);
                 break;
              }

           }
        }
        $tree = array_values($tree);
        //Devide And Conqure
        $this->setAsosiasiPath($tmpTree);
        $this->setAsosiasiPath($tree);
     }
  }
  function setAsosiasiIndikator($minimumSupportCount){
     for ($i=0;$i<count($this->path);$i++){
        //Cek Indikator
        $cekIndikator=FALSE;
        $index=0;
        for ($j=0;$j<count($this->asosiasiIndikator);$j++){
           if ($this->asosiasiIndikator[$j]->index==$this->path[$i]->indexIndikator[0]){
              $index=$j;
              $cekIndikator=TRUE;
           }
        }
        if (!($cekIndikator)){
           $index = count($this->asosiasiIndikator);
           $this->asosiasiIndikator[$index] = new AsosiasiIndikator($this->path[$i]->indexIndikator[0]);
        }

        for ($j=1;$j<count($this->path[$i]->indexIndikator);$j++){
           $cekIndexAsosiasi=FALSE;
           for ($k=0;$k<count($this->asosiasiIndikator[$index]->indexAsosiasi);$k++){
              if ($this->asosiasiIndikator[$index]->indexAsosiasi[$k]==$this->path[$i]->indexIndikator[$j]){
                 $indexSupportCount=$k;
                 $cekIndexAsosiasi = TRUE;
                 break;
              }
           }
           if (!($cekIndexAsosiasi)){
              $this->asosiasiIndikator[$index]->addIndexAsosiasi($this->path[$i]->indexIndikator[$j]);
              $this->asosiasiIndikator[$index]->addSupportCount($this->path[$i]->supportCount);
           }else{
              $this->asosiasiIndikator[$index]->editSupportCount($indexSupportCount,$this->path[$i]-
>supportCount);
           }
        }
     }
```

```
        //Normalkan Index Indikator
        for ($i=0;$i<count($this->asosiasiIndikator);$i++){
            $this->asosiasiIndikator[$i]->index = $this->indikatorTransaction[$this->asosiasiIndikator[$i]->index];
            for ($j=0;$j<count($this->asosiasiIndikator[$i]->indexAsosiasi);$j++){
                $this->asosiasiIndikator[$i]->indexAsosiasi[$j] = $this->indikatorTransaction[$this->asosiasiIndikator[$i]->indexAsosiasi[$j]];
            }
        }
        //Menghapus Asosiasi yang lebih kecil dari minimum support count
        for ($i=0;$i<count($this->asosiasiIndikator);$i++){
            $loop = count($this->asosiasiIndikator[$i]->indexAsosiasi);
            for ($j=0;$j<$loop;$j++){
                if (($this->asosiasiIndikator[$i]->supportCount[$j])/count($this->dataCluster)<$minimumSupportCount){
                    unset($this->asosiasiIndikator[$i]->supportCount[$j]);
                    unset($this->asosiasiIndikator[$i]->indexAsosiasi[$j]);
                }
            }
            $this->asosiasiIndikator[$i]->supportCount = array_values($this->asosiasiIndikator[$i]->supportCount);
            $this->asosiasiIndikator[$i]->indexAsosiasi = array_values($this->asosiasiIndikator[$i]->indexAsosiasi);

        }
    }
}
```

5. Source Code *Class FpTree*

```
Class FpTree{
   static $node = array();
   static function setTree($dataTransaction){
      for ($i=0;$i<count($dataTransaction);$i++){
         FpTree::setPush($dataTransaction[$i]);
      }
   }
   static function setPush($data){
      $l=0;
      $level=0;
      $parent=null;
      for ($s=0;$s<count($data);$s++){
         if ($data[$s]==1){
            $tmp=FALSE;
            for ($j=0;$j<count(FpTree::$node);$j++){
               if ((FpTree::$node[$j]->level==$level)&&(FpTree::$node[$j]-
>indexIndikator==$s)&&(FpTree::$node[$j]->parent==$parent)){
                  FpTree::$node[$j]->supportCount++;
                  $parent = FpTree::$node[$j]->id;
                  $tmp=TRUE;
                  break;
               }
            }
            if ($tmp==FALSE){
               $index = count(FpTree::$node);
               FpTree::$node[$index]= new Node($index, $s, $level, $parent);
               $parent  = $index;
            }
            $level++;
         }
      }
   }
}
```

6. Source Code *Class Node*

```
Class Node{
   var $id;
   var $indexIndikator;
   var $level;
   var $parent;
   var $supportCount;

   public function __construct($i,$in,$le,$pa) {
      $this->id=$i;
      $this->indexIndikator=$in;
      $this->level=$le;
      $this->parent=$pa;
      $this->supportCount=1;
   }
}
```

7. Source Code *Class Asosiasi Indikator*

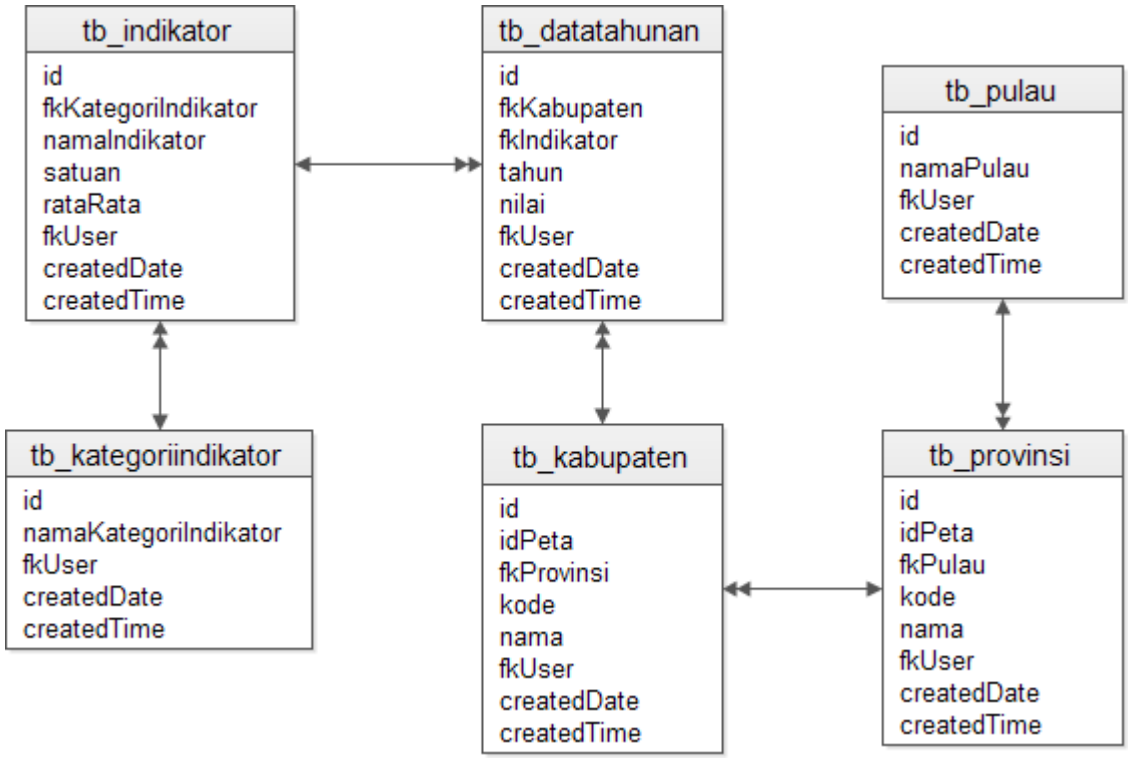
```
Class AsosiasiIndikator{
   var $index;
   var $indexAsosiasi=array();
   var $supportCount=array();

   public function __construct($i) {
      $this->index=$i;
   }
   public function addIndexAsosiasi($ia){
      array_push($this->indexAsosiasi,$ia);
   }
   public function addSupportCount($sc){
      array_push($this->supportCount,$sc);
   }
   public function editSupportCount($i,$sc){
      $this->supportCount[$i]+=$sc;
   }
}
```

8. Source Code *Class Path*

```
Class Path{
   var $indexIndikator = array();
   var $supportCount;
   function __construct($index, $count) {
      $this->indexIndikator = $index;
      $this->supportCount = $count;
   }
}
```

9. Relasi Database



10. Struktur Tabel

Tabel Pulau				
No	**Attribut**	**Tipe**	**Ukuran**	**Keterangan**
1	id	Integer	11	*Primary Key*
2	namaPulau	String	50	Nama Pulau
3	fkUser	Integer	11	*Foreign Key User* Pembuat
4	createdDate	Date		Tanggal Pembuatan
5	createdTime	Time		Pukul Pembuatan

Tabel Provinsi				
No	**Attribut**	**Tipe**	**Ukuran**	**Keterangan**
1	id	Integer	11	*Primary Key*
2	idPeta	String	2	Id Peta
3	fkPulau	Integer	11	*Foreign Key* Tabel Pulau
4	kode	String	2	Kode Pulau
5	nama	String	50	Nama Provinsi
6	fkUser	Integer	11	*Foreign Key User* Pembuat
7	createdDate	Date		Tanggal Pembuatan
8	createdTime	Time		Pukul Pembuatan

Tabel Kabupaten				
No	**Attribut**	**Tipe**	**Ukuran**	**Keterangan**
1	id	Integer	11	*Primary Key*
2	idPeta	String	2	Id Peta
3	fkProvinsi	Integer	11	*Foreign Key* Tabel Provinsi
4	kode	String	2	Kode Kabupaten
5	nama	String	50	Nama Kabupaten / Kota

6	fkUser	Integer	11	*Foreign Key User* Pembuat
7	createdDate	Date		Tanggal Pembuatan
8	createdTime	Time		Pukul Pembuatan

Tabel Kategori Indikator				
No	**Attribut**	**Tipe**	**Ukuran**	**Keterangan**
1	id	Integer	11	*Primary Key*
2	namaKategoriIndikator	String	50	Nama Kategori Indikator
3	fkUser	Integer	11	*Foreign Key User* Pembuat
4	createdDate	Date		Tanggal Pembuatan
5	createdTime	Time		Pukul Pembuatan

Tabel Indikator				
No	**Attribut**	**Tipe**	**Ukuran**	**Keterangan**
1	Id	Integer	11	*Primary Key*
2	fkKategoriIndikator	Integer	11	*Foreign Key User* Pembuat
3	namaIndikator	String	50	Nama Indikator
4	Satuan	String	20	Satuan Indikator
5	rataRata	Enum	0,1	Jenis Nilai Apakah Rata-rata
6	fkUser	Integer	11	*Foreign Key User* Pembuat
7	createdDate	Date		Tanggal Pembuatan
8	createdTime	Time		Pukul Pembuatan

Tabel Data Tahunan				
No	**Attribut**	**Tipe**	**Ukuran**	**Keterangan**
1	Id	Integer	11	*Primary Key*
2	fkKabupaten	Integer	11	*Foreign Key* Tabel Kabupaten
3	fkIndikator	Integer	11	*Foreign Key* Tabel Indikator
4	tahun	Integer	4	Tahun
5	fkUser	Integer	11	*Foreign Key User* Pembuat
6	createdDate	Date		Tanggal Pembuatan
7	createdTime	Time		Pukul Pembuatan