RANCANGAN DATABASE STORAGE ENGINE MENGGUNAKAN DATA DALAM FORMAT XML

Ashari Imamuddin
Jurusan Teknik Informatika, Fakultas Ilmu Komputer, Universitas Bina Nusantara
Jl. KH Syahdan 9 Kemanggisan, Jakarta Barat 11480, Jakarta, Indonesia
e-mail: ashari1844@lecturer.binus.ac.id

ABSTRAKSI

Database storage engine merupakan komponen yang sangat penting dalam RDBMS (relational database namagement system). Komponen ini berfungsi melakukan organisasi data dan akses file dalam physical storage. Format XML digunakan karena XML memiliki struktur bahasa yang sederhana dan fleksibel sehingga mendukung pembuatan bahasa atau tag sendiri sesuai dengan fungsinya. Rancangan ini meliputi pembuatan struktur basis data, tabel, indeks, pengubahan struktur basis data, tabel, penghapusan basis data, tabel, indeks, pengambilan, penambahan, pengubahan, dan penghapusan data. Pencarian data dilakukan dengan menggunakan indeks baik clustered maupun nonclustered index.

Kata kunci: *database, database storage engine, dan XML.*

1. PENDAHULUAN

Sistem manajemen file ini terdiri dari banyak aplikasi dimana masing-masing aplikasi mendefinisikan dan mengatur data secara terpisah. Beberapa kelemahan dari sistem ini:

1. Data bersifat terpisah dan relasi antardata tidak ada sehingga data menjadi sulit diakses dan relasi antardata yang tidak jelas.
2. Dapat terjadi duplikasi atau redudansi dalam pengambilan dan penyimpanan data.
3. Format dan struktur *file* yang tidak sama karena bergantung pada masing-masing aplikasinya.
4. Terdapatnya aplikasi yang tidak cocok dalam *query* dari aplikasi lain sehingga tidak ada ketentuan untuk keamanan dan integritas data, *recovery*, serta pembagian akses dalam file.

Keterbatasan dan permasalahan dalam sistem *File-Based Approach* inilah yang kemudian melahirkan konsep mengenai sistem basis data dimana kumpulan data memiliki hubungan logika, penjelasan data, dan dirancang untuk kebutuhan informasi perusahaan. Sistem basis data mendefinisikan kendali dan akses yang jelas dan disebut juga *Relational Database Management System* (RDBMS). Sebuah RDBMS mengijinkan *user* untuk mendefinisikan sistem basis data atau *Data Definition Language* (DDL) dan memanipulasi data dengan meng-*insert*, meng-*update*, men-*delete*, serta me-*retrieve* data dari sistem basis data atau *Data Manipulation Language* (DML). Data yang disimpan dalam media yang besar dapat digunakan secara berkesinambungan. Data-data disimpan dalam organisasi *file* yang teratur, dan informasi dari setiap data dicatat dengan jelas dalam *system catalog*. Sebuah RDBMS berfungsi membaca data dari media penyimpanan, kemudian menampilkan data kepada *user*, atau menambahkan, menghapus, mengubah data dari *user* melalui bahasa umum yang dinamakan *Structured Query Language* (SQL).

Penyimpanan dan pengaturan data dalam organisasi *file* memiliki banyak teknik dan metode. Untuk itu banyak pembahasan mengenai bagaimana penyimpanan dan pengaturan data yang efisien. Berdasarkan latar belakang inilah sehingga dibuatlah suatu penelitian dan perancangan aplikasi untuk merancang sebuah arsitektur untuk komponen RDBMS yang berfungsi mengatur *file* yang efisien secara fisikal pada komputer dan penyimpanan data.

2. LINGKUP RANCANGAN

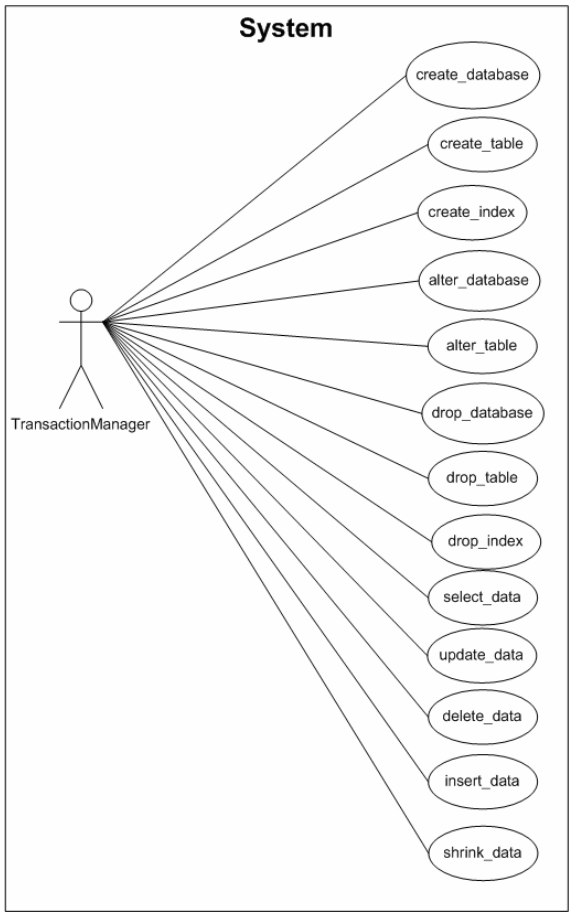
Komponen *storage engine* ini berfungsi mengatur organisasi data dan akses *file* dalam *physical storage*. Penyimpanan data dibuat dalam format XML. Tipe data yang digunakan yaitu *integer*, *float*, *char*, dan *date*.

Perancangan ini akan diimplementasikan ke dalam sebuah aplikasi yang ditujukan untuk sistem operasi *Windows*. Aplikasi dari rancangan ini akan menerima masukan dari *Transaction Manager*. *Query* yang diterima adalah *query Data Definition Language* (DDL) dan *Data Manipulation Language* (DML).

Query untuk DDL melakukan proses pembuatan struktur basis data, yang meliputi basis data, tabel, kolom, dan indeks. Transaksi untuk *query* DDL pada bagian pembuatan indeks dibatasi hanya untuk membuat *clustered index* pada *primary key* dan *nonclustered index* (*non-unique*).

Query untuk DML dapat melakukan proses manipulasi data yang meliputi menambah data, menghapus data, mengubah data, dan melihat data. Transaksi *query* DML untuk menghapus dan mengubah data dibatasi hanya untuk satu tabel dan memiliki satu kondisi. Transaksi *query* DML untuk mengubah data dibatasi hanya dapat mengganti nilai baru dengan nilai yang tetap (bukan hasil perhitungan). Transaksi *query* DML untuk melihat data dibatasi hanya dapat menampilkan data untuk sejumlah kolom tertentu dari satu tabel dan satu

buah kondisi serta tidak membahas mengenai pengelompokan (*group by*), pengurutan (*order by*), memasukkan ke tabel lain (*into*), menggunakan nilai hasil pengelompokan (*having*), menghilangkan nilai yang memiliki duplikasi (*distinct*), ataupun memilih baris berdasarkan urutan (*top*).



Gambar 1. Use Case Diagram - Rancangan Aplikasi

3. RANCANGAN STORAGE ENGINE

3.1 Rancangan Use Case

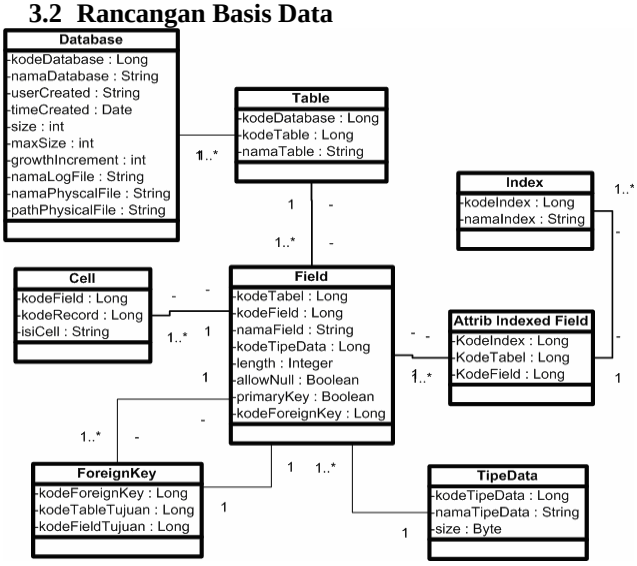
Pada Gambar 1 dapat dilihat use case yang terdapat dalam system. Di sini storage engine dibagi menjadi dua bagian, yaitu:

Dibagi menjadi dua bagian yaitu fasilitas *Data Definition Language* (DDL) dan *Data Manipulation Language* (DML).

a. *Data Definition Language* (DDL) yang menyediakan fasilitas untuk:
   - Mangatu pembuatan basis data baru.
   - Perubahan struktur pada basis data yang ada.
   - Menghapus basis data yang ada.
   - Membuat tabel baru dalam sebuah basis data.
   - Mengubah struktur pada tabel yang ada.
   - Menghapus tabel yang ada.
   - Membuat indeks baru.
   - Menghapus indeks yang sudah ada.

b. *Data Manipulation Language* (DML) yang menyediakan fasilitas untuk:
   - Menambahkan data pada sebuah tabel yang ada.
   - Menampilkan data pada sebuah tabel yang ada.
   - Menghapus data pada sebuah tabel yang ada.
   - Mengubah data pada sebuah tabel yang ada.

3.2 Rancangan Basis Data



Gambar 2. Rancangan Basis Data

Pada gambar 2 adalah rancangan basis data dari RDBMS ini. Gambar tersebut menunjukkan data yang disimpan di dalam database system, yaitu database, table, field, cell, index, foreignkey, dan tipe data.

3.3 Rancangan Format File XML dan Deskripsi Data

Organisasi *File* yang diharapkan adalah di root *folder* terdapat beberapa *file* yang berfungsi sebagai *template*, serta menyimpan informasi basis data dan tipe data yang ada, yaitu:

a. *SysCatalog*

Menerangkan format data penyimpanan dan deskripsi tipe data setiap objek yang ada. Di dalamnya terdapat:
- *SysDatabase* untuk mendeskripsikan tentang format basis data.
- *SysTable* untuk mendeskripsikan tentang format tabel.
- *SysField* untuk mendeskripsikan tentang format kolom.
- *SysRecord* untuk mendeskripsikan tentang format dari baris.
- *SysTipeData* untuk mendeskripsikan tentang format tipe data.
- *SysIndex* untuk mendeskripsikan tentang format indeks.

```
<SystemCatalog>
  <sysDatabase kodeDatabase=Long>
  <namaDatabase>String</namaDatabase>
    <userCreated>String</userCreated>
    <timeCreated>Date</timeCreated>
    <timeLastModified>Date</timeLastModified>
    <namaLogFile>String</namaLogFile>

<namaPhysicalFile>String</namaPhysicalFile>

<pathPhysicalFile>String</pathPhysicalFile>
    <countTable>Integer</countTable>
  </sysDatabase>
    <sysTable kodeTable=Long status=Byte>
    <namaTable>String</namaTable>
    <userCreated>String</userCreated>
    <timeCreated>Date</timeCreated>
    <timeLastModified>Date</timeLastModified>
    <countRow>Integer</countRow>
    <countField>Integer</countField>
  </sysTable>
  <sysField kodeField=Long status=Byte>
    <namaField>String</namaField>
    <kodeTipeData>Long</kodeTipeData>
    <length>Integer</length>
    <precision>Integer</precision>
    <allowNull>Boolean</allowNull>
    <primaryKey>Boolean</primaryKey>
    <kodeForeignKey kode=Long>
      <kodeTableTujuan>Long</kodeTableTujuan>
      <kodeFieldTujuan>Long</kodeFieldTujuan>
   </kodeForeignKey>
  </sysField>
  <sysCell kodeRecord=Long status=Byte>
    <isiCell>String</isiCell>
  </sysCell>
  <sysTipeData kodeTipeData=Long>
    <namaTipeData>String</namaTipeData>
    <size>Byte</size>
  </sysTipeData>
  <sysIndex kodeIndex=Long>
    <namaIndex>Long</namaIndex>
    <kodeTable>Long</kodeTable>
    <kodeField>Long</kodeField>
  </sysIndex>
</SystemCatalog>
```

b. ***SysTypes***

Berisi tipe data-tipe data yang terdapat pada sistem dengan detilnya antara lain:

- Kode Tipe Data, dengan kisaran nilai dari 1 sampai 99
- Nama Tipe Data
- *Size*, ukuran dari tipe data

```
<DataTypeObjects>
  <TipeData kodeTipeData="1">
    <namaTipeData>int</namaTipeData>
    <size>4</size>
  </TipeData>
    <TipeData kodeTipeData="2">
    <namaTipeData>float</namaTipeData>
    <size>12</size>
  </TipeData>
    <TipeData kodeTipeData="3">
    <namaTipeData>datetime</namaTipeData>
    <size>10</size>
  </TipeData>
    <TipeData kodeTipeData="4">
    <namaTipeData>char</namaTipeData>
    <size>1</size>
  </TipeData>
</DataTypeObjects>
```

c. ***SysTypes***

Menerangkan format untuk penyimpanan informasi dari setiap basis data.

```
<Database kodeDatabase="">
    <Table kodeTable="" status="">
    <namaTable></namaTable>
    <userCreated></userCreated>
    <timeCreated></timeCreated>
    <timeLastModified></timeLastModified>
    <countField></countField>
    <countRow></countRow>
  </Table>
  </Database>
```

d. ***SysIndexes***

Menerangkan format untuk penyimpanan indeks yang dibuat di dalam satu tabel dari suatu basis data.

```
<IndexObjects>
  <Index kodeIndex="  ">
    <namaIndex></namaIndex>
    <IndexedField kodeTable="  ">
      <kodeField></kodeField>
      <kodeField></kodeField>
    </IndexedField>
  </Index>
</IndexObjects>
```

e. ***SysIndexValues***

Menerangkan format untuk penyimpanan data tabel sesuai indeks yang ada.

```
<Index kodeIndex="  " kodeTable="  ">
  <Field kodeField="  ">
    <Cell>
      <isiCell>   </isiCell>
      <kodeRecord>   </kodeRecord>
    </Cell>
  </Field>
</Index>
```

f. ***SysTables***

Menerangkan format untuk penyimpanan informasi dari setiap tabel.

```
<Table kodeTable="  ">
  <Field kodeField="  " status=" ">
    <namaField></namaField>
    <kodeTipeData></kodeTipeData>
    <length></length>
    <precision></precision>
    <allowNull></allowNull>
    <primaryKey></primaryKey>
    <ForeignKey kodeForeignKey="  ">
      <kodeTableTujuan></kodeTableTujuan>
      <kodeFieldTujuan></kodeFieldTujuan>
    </ForeignKey>
  </Field>
</Table>
```

g. ***SysValues***

Menerangkan format untuk penyimpanan data dari setiap baris dari suatu tabel.

```
<Table kodeTable="  ">
  <Record kodeRecord="    " status=" ">
    <Field    kodeField="        "    length="
"></Field>
  </Record>
</Table>
```

h. ***Databases***

Berisi objek-objek basis data yang ada, dengan detilnya antara lain:

- Kode *Database*, sebagai atribut, dengan kisaran nilai dari 1 sampai 99.
- Nama *Database*, dengan panjang nama maksimal 59 karakter.
- *User Created*, dengan panjang nama maksimal 16 karakter.
- *Time Created*, dengan format mm/dd/yyyy.
- *Time Last Modified*, dengan format mm/dd/yyyy.
- Nama *Log File*, dengan panjang nama maksimal 64 karakter (default : nama *log file* didapat dari nama basis data + _Log).
- Nama *Physical File*, dengan panjang nama maksimal 64 karakter (default : nama *physical file* didapat dari nama basis data + _Data).
- *Path Physical File*, dengan panjang nama maksimal 128 karakter.
- *Count Table*, jumlah tabel yang dimiliki basis data, dengan kisaran nilai dari 1 sampai 99.

```
<DatabaseObjects>
  <Database kodeDatabase="  ">
    <namaDatabase></namaDatabase>
    <userCreated></userCreated>
    <timeCreated></timeCreated>
    <timeLastModified></timeLastModified>
    <namaLogFile></namaLogFile>
    <namaPhysicalFile></namaPhysicalFile>
    <pathPhysicalFile></pathPhysicalFile>
    <countTable></countTable>
   </Database>
</DatabaseObjects>
```

Selanjutnya setiap basis data dibuatkan *folder* atau direktori untuk menyimpan informasi basis data tersebut, informasi setiap tabel, nilai setiap tabel, indeks, yang terdiri atas:

a. "Info" + (Nama Basis Data)

Menerangkan informasi dari tabel-tabel yang terdapat dalam database tersebut, dengan detil dari tabel sebagai berikut:

- *kodeTable*, sebagai atribut, dengan kisaran nilai dari 1 sampai 99.
- status, sebagai atribut, dengan nilai 0 (deleted), 1 (active), 2 (updated).
- *namaTable*, dengan panjang nama maksimal 64 karakter.
- *userCreated*, dengan panjang nama maksimal 16 karakter.
- *timeCreated*, dengan format mm/dd/yyyy.
- *timeLastModified*, dengan format mm/dd/yyyy.
- *countField*, jumlah kolom yang dimiliki setiap tabel, dengan kisaran nilai dari 1 sampai 99.
- *countRecord*, jumlah baris yang dimiliki setiap tabel, dengan kisaran nilai dari 1 sampai 9999.

b. "Info" + "-" + (Kode Basis Data) + "-" + (Kode Tabel)

Menerangkan informasi dari kolom-kolom yang terdapat dalam suatu tabel, dengan detil dari kolom adalah sebagai berikut:

- *kodeField*, sebagai atribut, dengan kisaran nilai dari 1 sampai 99.
- *status*, sebagai atribut, dengan nilai 0 (deleted), 1 (active), 2 (updated).
- *namaField*, dengan panjang nama maksimal 64 karakter.
- *kodeTipeData*, dengan kisaran nilai antara 1 sampai 99.
- *precision*, dengan kisaran nilai 0 sampai 4.
- *allowNull*, dengan nilai antara true atau false.
- *primaryKey*, dengan nilai antara true atau false.
- *foreignKey*, dengan detailnya yaitu:
  - *kodeForeignKey*, dengan kisaran nilai 0 sampai 99, nilai 0 berarti tidak terdapat hubungan ke tabel lain.
  - *kodeTableTujuan*, dengan kisaran nilai 0 sampai 99, nilai 0 berarti tidak terdapat hubungan ke tabel lain.
  - *kodeFieldTujuan*, dengan kisaran nilai 0 sampai 99, nilai 0 berarti tidak terdapat hubungan ke kolom dari tabel lain.

c. "Data" + "-" + (Kode Basis Data) + "-" + (Kode Tabel)

Menerangkan nilai data dari suatu tabel dengan data tiap baris, detilnya sebagai berikut:

- *kodeRecord*, sebagai atribut, dengan kisaran nilai antara 1 sampai 9999.
- *status*, sebagai atribut, dengan nilai 0 (deleted), 1 (active), 2 (updated).
- Setiap baris memuat nilai-nilai dari kolom, panjang data dari setiap kolom dialokasikan sesuai dengan tipe data awal kolom itu dibuat. Untuk informasi dalam kolom ditambahkan:
  - *kodeField*, dengan kisaran nilai antara 1 sampai 99.
  - *length*, panjang nilai data, dengan kisaran nilai antara 1 sampai 999.

d. "Index" + (Nama Basis Data)

Menerangkan indeks yang dimiliki dalam sebuah basis data, detilnya sebagai berikut:

- *kodeIndex*, sebagai atribut, dengan kisaran nilai antara 1 sampai 99.
- *namaIndex*, dengan panjang nama maksimal 64 karakter
- *kodeTable*, sebagai atribut, dengan kisaran nilai antara 1 sampai 99 dan menerangkan kolom apa saja yang digunakan dalam indeks. Disimpan dengan detil berupa:
  - *kodeField*, dengan kisaran nilai antara 1 sampai 99.

e. "Index" + "-" + (Kode Basis Data) + "-" + (Kode Indeks)

Menerangkan nilai data dari suatu indeks, dimana nilai data sudah dikelompokkan menurut nilai dari kolom yang dijadikan indeks, detilnya sebagai berikut:

- *kodeIndex*, sebagai atribut, dengan kisaran nilai antara 1 sampai 99.
- *kodeTable*, sebagai atribut, dengan kisaran nilai antara 1 sampai 99.
- *kodeField*, dengan kisaran nilai antara 1 sampai 99, di dalam setiap kolom yang dijadikan indeks, disimpan *distinct* data, data unik dengan detil sebagai berikut:
  - *isiCell*, nilai dari kolom.

4. KESIMPULAN

Makalah ini merupakan hasil penelitian pengembangan sebuah RDBMS *storage engine*. Rancangan ini telah dikonstruksi dan telah diimplementasikan ke dalam bahasa pemrograman yang telah ditentukan.

Storage engine ini dapat mengatur organisasi data dan akses *file*. *Storage manager* ini dibuat dengan *format file* XML. *Format file* XML dipilih karena dengan menggunakan *format* ini dapat membuat *tag* sendiri sesuai keinginan *user*, mendukung integritas data dan pengambilan data yang akurat karena menggunakan *tag*. Eksekusi perintah untuk *Data Definition Language* (DDL) mempunyai kemampuan untuk membuat basis data baru, tabel baru, dan indeks baru, mengubah struktur basis data, struktur tabel yang terdapat pada sistem, serta menghapus basis data, tabel, indeks yang terdapat pada sistem. Eksekusi perintah untuk *Data Manipulation Language* (DML) mempunyai kemampuan untuk melihat data, menambah data baru, mengubah serta menghapus data yang ada. Eksekusi perintah DML untuk melihat, mengubah dan menghapus data dapat memiliki *filter* satu kondisi berdasarkan masukan nilai sesuai jenis tipe data kolomnya atau menampilkan sejumlah kolom tertentu.

Pencarian data dilakukan menggunakan indeks yaitu *clustered index* yang terdapat pada *primary key*. *Clustered index* dilakukan dengan mengurutkan data langsung pada *physical storage*. Pencarian data juga dilakukan menggunakan *nonclustered index*. *Nonclustered index* dilakukan dengan mengambil nilai *distinct* dari nilai-nilai dari kolom, mengurutkan dan menyimpan kode baris berdasarkan masing-masing nilai *distinct* tersebut. Pembacaan data dilakukan untuk menampilkan sejumlah baris tertentu dari *file* pada *physical storage*.

Proses DDL dan DML menghasilkan data yang akurat dan dapat digunakan untuk proses selanjutnya. Semakin banyak jumlah kolom ataupun jumlah baris, maka pemakaian waktu, memori dan CPU untuk transakai *select*, *update* dan *delete* semakin besar. Pemakaian waktu dan memori mengalami peningkatan eksponensial terhadap jumlah baris karena setiap proses akan melakukan pembacaan ulang untuk semua data dari tiap tabel yang mengalami proses dan ditampung dalam memori. Untuk pemakaian CPU, hampir semua proses menggunakan 99 persen terutama jika proses tersebut memakan waktu yang cukup lama. Hal ini dapat terjadi karena salah satu prinsip sistem operasi yang akan memaksimalkan penggunaan semua *resource* yang ada jika tidak ada proses lain yang memiliki prioritas sama atau lebih tinggi.

PUSTAKA

[1] Britton, Carol and Doake, Jill. (2001). *Object-Oriented System Develompent: A Gentle Introduction*. McGraw-Hill Companies, Inc.

[2] Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; and Stein Clifford. (2001). *Introduction to Algorithms*, Second Edition. MIT-Press and Mc-GrawHill. USA.

[3] Connolly, Thomas and Begg, Carolyn. (2002). *Database Systems*: *A Practiocal Approach to Design, Implementation and Management*. 3rd Edition. Pearson Education Limited, United States of America.

[4] Date, C. J.(2000). *An Introduction to Database System.* Addison Wesley Longman, Inc, USA.

[5] Elmasri, Ramez and Navathe, Shamkant B. (2000). *Fundamentals of Database Systems*. 3rd Edition. Addison Wesley.

[6] Ramakrishnan, Raghu and Gehrke, Johannes. (2003). *Database Management Systems*. 3rd Edition. McGraw-Hill Companies, Inc.

[7] Silbreschatz, Abraham, Korth, Henry F. and Sudarshan, S. (2006). *Database System Concepts*. 5th Edition. McGraw-Hill Companies, Inc.

[8] Whitten, Jeffery L., Bentley, Lonnie D. and Dittman, Kevin C. (2004). *System Analysis Design Methods*. 6th Edition. Mc-GrawHill, USA.