QUERY BUILDER PADA RDBMS ORACLE MENGGUNAKAN XML DAN ACTIVEX BERBASIS WEB

Darlis Heru Murti, Yudhi Purwananto, M. Rifqi Febrianto
Jurusan Teknik Informatika,
Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember
Kampus ITS, Jl. Raya ITS, Sukolilo – Surabaya 60111, Telp. + 62 31 5939214, Fax. + 62 31 5913804
Email: darlis@its-sby.edu , yudhi@its-sby.edu

ABSTRAK

Dalam penelitian ini akan dibahas mengenai pembuatan aplikasi pembangkit query berbasis web pada RDBMS Oracle 9i yang memiliki kemampuan untuk melakukan pembuatan query baik secara forward(diagram ke teks query) maupun reverse(teks query ke diagram) dengan menggunakan teknologi IIS, ISAPI Ekstension, activex dan XML sebagai data interchange.

Penelitian ini bertujuan untuk membuat sebuah arsitektur dimana instalasi Oracle hanya dilakukan sekali, yaitu pada komputer server. Sedangkan komputer client tidak perlu diinstall, namun tetap harus terkoneksi pada server melalui jaringan LAN. Dengan teknologi maka semua perintah (syntax SQL) yang dibentuk pada server akan dikirm pada server untuk dieksekusi pada level database Dan record hasil dari eksekusi tersebut akan dikembalikan nilainya pada komputer client..

Uji coba dan evaluasi dilakukan dengan menggunakan 2 cara, yaitu uji coba kebenaran dalam membentuk syntax query dan uji coba kecepatan aplikasi yang sudah dikembangkan. Untuk uji coba kebenaran, aplikasi sudah bisa melakukan pembentukan syntax SQL baik secara forward maupun reverse namun hanya terbatas pada fungsi-fungsi standart pada ANSI SQL. Untuk uji coba kecepatan, menunjukkan bahwa kinerja perangkat lunak dipengaruhi 3 parameter yaitu jumlah tabel, jumlah kolom pada masing-masing tabel, dan jumlah record hasil eksekusi syntax SQL. Dimana peningkatan waktu yang dibutuhkan bersifat eksponensial.

Kata kunci :*Pembangkit Query, IIS, ISAPI Extension, ActiveX, XML, Oracle 9i.*

1. PENDAHULUAN

Dalam lingkungan proses Sistem Informasi, untuk menghasilkan kecepatan yang tinggi dalam proses pelayanan dan pemberian laporan–laporan yang akurat pada user dibutuhkan *query*. Perangkat lunak yang ada pada umumnya hanya melakukan pembuatan SQL dari diagram namun pembentukan visualisasi *syntax* SQL belum diterapkan. Selain itu masih membutuhkan instalasi pada masing-masing komputer untuk berkoneksi pada basis data *server*. Oleh karena itu diperlukan sebuah mekanisme dimana instalasi hanya dilakukan pada komputer server saja tidak pada masing-masing komputer *client* dan memiliki kemampuan untuk melakukan proses pembuatan SQL dengan forward (diagram ke teks SQL) maupun reverse (teks SQL ke diagram).

Untuk membuat arsitektur tersebut maka pada proses perancangan dan pembuatan digunakan ActiveX Control pada lingkungan *client* dan IIS serta ISAPI Ekstensions pada lingkungan *server*. Dimana aplikasi *client* berupa komponen akan diletakkan pada sebuah kontainer, yaitu *Web Browser* Internet Explorer. Sedangkan aplikasi *server* berupa DLL yang akan menerima *request-request* dari aplikasi *client* untuk dieksekusi pada level *dabatase*. Dengan menerapkan teknologi tersebut maka instalasi Oracle hanya dilakukan sekali yaitu pada komputer *server*, namun masing-masing *client* yang ingin memiliki koneksi pada basis data *server* harus terkoneksi pada komputer *server* melalui jaringan LAN/Internet. Dan aplikasi yang dikembangkan akan berbasis web karena komponen yang sudah dibuat pada lingkungan *client* memiliki kontainer *Web Browser*.

2. DASAR TEORI

ISAPI EXTENSIONS

Internet Server Application (ISAPI) merupakan sebuah kumpulan dari ekstensi-ekstensi *Microsoft Foundation Class* (MFC) yang memungkinkan untuk dapat bekerja secara langsung dengan *Internet Information Server* (IIS) [MUE-98]. ISA yang telah dibuat akan menjadi sebuah DLL yang akan di-*load* pada sever. Dan DLL tersebut akan berbagi alamat memori yang sama dengan yang dimiliki oleh server HTTP [MIC-01], dan kita dapat me-*unload*-nya bila kita memerlukan kebutuhan akan *space* memori pada aplikasi yang lain.

ISAPI *Extensions* merupakan sebuah DLL yang dapat di-*load* ataupun dipanggil oleh *server* HTTP. Dengan itu maka user dapat mengisi *form* dan *click* pada sebuah tombol untuk mengirimkan data pada sebuah *Web server* dan membangkitkan ISA-nya, yang bisa memproses informasi untuk

menyediakan isi yang berlainan atau menyimpannya pada *database*. *Extesion* Web *server* dapat menggunakan informasi dalam sebuah database untuk membangun sebuah halaman web yang dinamis kemudian mengirimkan informasi tersebut pada komputer *client* untuk ditampilkan.

ACTIVEX CONTROL

ActiveX Control merupakan sebuah obyek COM (*Component Object Model*) yang dapat mengimplementasikan sebuah antarmuka tertentu (*interface*) yang dapat dilihat dan bertindak seperti sebuah kontrol. *Interface* yang telah disebutkan di atas adalah *IUnknown*, yaitu sebuah *interface* khusus yang dibutuhkan oleh semua komponen yang mengimplementasikan fungsi-fungsi penting ActiveX Control biasanya mendukung banyak *interface* dalam rangka untuk memberikan lebih banyak kemampuan, namun semuanya bisa dianggap sebagai *optional*. Selain yang telah disebutkan di atas *ActiveX Control* juga merupakan sebuah obyek yang kompleks dimana tidak hanya bisa mengimplementasikan 1 macam *interface* COM saja, namun banyak.

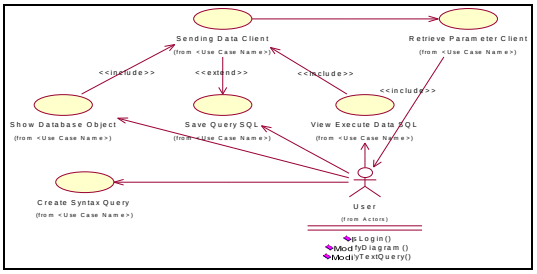
XML

XML adalah sebuah markup language yang menyediakan format untuk mendeskripsikan data terstruktur dan menyediakan semacam aturan-aturan untuk membuat susunan yang terstruktur. XML dibuat untuk mengatasi kekurangan dari SGML (Standard Generalized Markup Language) pada proses pengiriman melalui web. Pada XML dapat mendefinisikan kumpulan tag yang tak terbatas. Sebuah elemen dapat mendeklarasikan isi (content) sebagai nama, alamat, tempat tanggal lahir, dan lain-lain. Karena XML merupakan standar, maka data XML dapat diambil dan dimanipulasi tanpa harus memperhatikan aplikasi yang menjalankannya

3. ARSITEKTUR SISTEM

Pada dasarnya perangkat lunak yang dibuat merupakan sebuah editor untuk melakukan pembuatan SQL pada RDBMS Oracle 9i. Perangkat lunak yang dikembangkan tersebut merupakan aplikasi *client*, dimana semua perintah dan kondisi yang terjadi pada lingkungan *client* akan diteruskan menuju ke lingkungan *server*. Pada lingkungan *server* terdapat juga sebuah perangkat lunak yang akan bertugas untuk meneruskan perintah dan kondisi sesuai dengan input yang telah diberikan pada lingkungan *client*. Perangkat lunak yang dikembangkan lebih menitikberatkan pada *query* yang bertipe ANSI SQL.

Gambar 1 menunjukkan use case diagram dari sistem aplikasi dan hubungan antara perangkat lunak dalam lingkungan *server* dan *client*.



Gambar 1. Use Case Diagram Sistem

4. PERANCANGAN APLIKASI *CLIENT - SERVER*

Perangkat lunak untuk *client* dirancang dengan menggunakan teknologi ActiveX. Aktifitas-aktifitas yang dilakukan pada lingkungan perangkat lunak *client* adalah sebagai berikut :

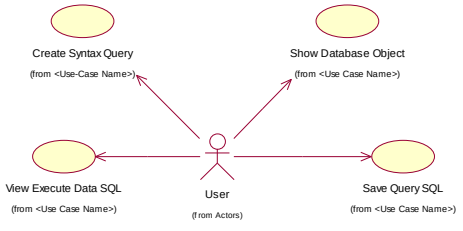
Menampilkan obyek basis data server
Pembentukan syntax SQL
Menampilkan data hasil eksekusi syntax SQL
Menyimpan syntax SQL

Perangkat lunak untuk *server* dirancang dengan menggunakan teknologi ISAPI Ekstensions yang merupakan sebuah DLL yang dapat di-*load server* HTPP. Dalam perancangan perangkat lunak untuk kebutuhan *server*, ada fungsi yang sangat penting, yaitu menerima dan mengirimkan kembali parameter ataupun data yang diterima dari *server*.

PERANCANGAN APLIKASI *CLIENT*

Perangkat lunak untuk *client* dirancang dengan menggunakan teknologi ActiveX yang merupakan sebuah obyek COM (*Component Object Model*) yang dapat mengimplementasikan sebuah antar muka tertentu (interface) yang dapat dilihat dan bertindak sebagai sebuah *container*. Karena yang dikembangkan merupakan perangkat lunak untuk melakukan pembuatan *query*, maka operasi yang dilakukan dalam editor tersebut banyak sekali, mulai membuat diagram, mengubah, dan menghapus obyek pada editor tersebut.

Proses-proses interaksi antara user dan perangkat lunak yang terjadi dalam lingkungan *client* dapat dijelaskan pada gambar 2.



Gambar 2. Interaksi antara user dengan perangkat lunak *client*

Dalam desain ActiveX Control yang dikembangkan dalam perangkat lunak *client*, ada beberapa obyek penting yang digunakan, antara lain :

QRYTable

Merupakan obyek diagram berupa table yang di dalamnya terdapat kolom-kolom (obyek QRYColumn). Dan masing-masing dari obyek tersebut dikumpulkan menjadi satu collection, QRYTables.

QRYRelation

Merupakan obyek diagram yang berupa relasi yang didalmnya merupakan reference yang menghubungkan 2 tabel. di dalamnya terdapat obyek QRYRelColumn yang menyimpan nama kolom-kolom yang saling berelasi. Dan masing-masing dari obyek tersebut dikumpulkan menjadi satu collection, QRYRelations.

QRYQueryCol

Merupakan obyek yang menyimpan nama-nama kolom yang dipilih ataupun termasuk dalam sebuah query yang telah dibangun dengan bantuan diagram. Dan masing-masing dari obyek tersebut dikumpulkan menjadi satu collection, QRYQueryCols.

QRYCondition

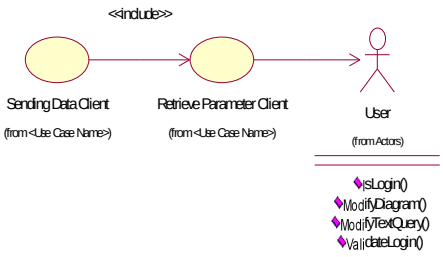
Merupakan obyek yang menyimpan kondisi-kondisi yang terdapat pada sebuah query yang telah dibangun dengan bantuan diagram. Dan masing-masing dari obyek tersebut dikumpulkan menjadi satu collection, QRYConditions.

PERANCANGAN APLIKASI *SERVER*

Perangkat lunak untuk server dirancang dengan menggunakan teknologi ISAPI Ekstensions yang merupakan sebuah DLL yang dapat di-load ataupun dipanggil oleh server HTPP. Perangkat lunak yang telah di-load dalam web server akan terus hidup dalam web server tersebut hingga servis dalam web server tersebut dimatikan.

Dalam perancangan perangkat lunak untuk kebutuhan server, ada fungsi yang sangat penting dalam perangkat lunak tersebut, yaitu menerima dan mengirimkan kembali parameter ataupun data yang diterima dari server.

Proses-proses yang terjadi dalam lingkungan server dapat dijelaskan pada gambar 3.



Gambar 3. Interaksi antara *user* dengan perangkat lunak *server*

Proses yang terjadi di *server* berikut merupakan proses yang memegang peranan sangat penting, karena proses inilah yang mengatur jalannya setiap perubahan yang dilakukan pada perangkat lunak *client*. Setiap proses *transfer* data antara perangkat lunak *client* dengan perangkat lunak *server* sebagian besar menggunakan fasilitas XML, karena kemudahannya untuk dibaca dalam *platform* apapun. Oleh karena itu mekanisme berikut di buat untuk mengatur proses yang telah disebutkan di atas. Ada 2 proses pada mekanisme tersebut, yaitu :
Menerima Parameter dari *Client*
Mengirim Data yang sudah diolah ke *Client*

KOMUNIKASI PERANGKAT LUNAK *CLIENT-SERVER*

Perangkat lunak yang dibuat terdiri dari 2 buah, yaitu perangkat lunak *client* dan perangkat lunak *server*. Oleh karena perangkat lunak yang dibuat ada 2 buah maka dibutuhkan sebuah mekanisme pengiriman data/komunikasi data antara kedua perangkat lunak tersebut. Metode yang digunakan dalam mekanisme pengiriman tersebut adalah metode Polling Data, dimana pihak *client* yang aktif melakukan *polling* ke *server*.

Selain itu, proses komunikasi tersebut juga menggunakan file-file XML yang memiliki struktur berbeda untuk masing-masing. Ada 3 macam struktur XML yang dipergunakan dalam sistem ini, yaitu :

1. Struktur XML Obyek Basis Data, strukturnya adalah sebagai berikut :

```
<file>
 <tablelist>
     <table>
     → properti dari table
         <columnlist>
             <column>
             → prxoperti dari column
             </column>
         </columnlist>
     </table>
</tablelist>
 <rellist>
     <relation>
     → properti dari relation
         <relcolumns>
             <relcolumn>
             → properti dari relcolumn
             </relcolumn>
```

```
            </relcolumns>
        </relation>
    </rellist>
    </file>
```

2. Struktur XML Data, strukturnya adalah sebagai berikut :

```
<dataxml>
        <row>
        → nilai dari tiap-tiap field
    </row>
</dataxml>
```

3. Struktur XML Penyimpanan file lokal, strukturnya adalah sebagai berikut :

```
<filesave>
 <select>
    <fieldname>
    → properti dari field pada query
    </fieldname>
 </select>
 <from>
    <tablename>
    → properti dari tabel pada query
    </tablename>
 </from>
 <where>
    <condition>
    → properti dari kondisi pada query
    </condition>
 </where>
 <globalwhere>
    <global>
    → properti dari kondisi global
    </global>
 </globalwhere>
 <globalhaving>
    <global>
    → properti dari kondisi global
    </global>
 </globalhaving>
</filesave>
```

5. APLIKASI *CLIENT*

Perangkat lunak yang dibuat pada lingkungan *client* merupakan sebuah ActiveX Control yang ditempatkan pada kontainer di Internet Explorer. Perangkat lunak ini dibuat dengan menggunakan bahasa pemrograman Visual Basic 6.

Untuk editor, tempat membuat *query* dengan bantuan diagam menggunakan komponen *third party*, yaitu AddFlow 4.2. Dengan menggunakan komponen tersebut maka proses penggambaran obyek-obyek seperti tabel dan relasi akan lebih mudah dilakukan karena tiap obyek yang digambarkan dibagi menjadi 2 jenis yaitu *node* dan *link*.

PEMBUATAN DIAGRAM

Pada bagian ini digunakan untuk melakukan beberapa perubahan pada diagram maka diperlukan adanya proses penambahan ataupun penghapusan obyek-obyek basis data pada editor. Dalam proses penambahan maupun penghapusan obyek-obyek tersebut dibagi menjadi beberapa bagian, yaitu prosedur yang berhubungan dengan obyek tabel dan prosedur yang berhubungan dengan obyek tabel relasi.

```
//Inisialisasi obyek AddFlow dan
QRYTable
Set vNewAddFlow = AddFlow4Lib.AddFlow
Set vTable = mTable

//Membuat Node Tabel dan
//Header(Nama tabel)
Set BodyNode = vNewAddFlow.Nodes.Add()
With BodyNode
 Set HeaderNode =
vNewAddFlow.Nodes.Add
 //Membuat field tabel
 AddColumn vNewAddFlow, vTable,
```

Gambar 4. *Pseudocode* tambah tabel (pada diagram)

PENANGANAN PROPERTY QUERY

Bagian ini digunakan untuk melakukan perubahan terhadap syntax query yang telah dibuat. Gambar 5 menunjukkan pseudocode dari prosedur penambahan field yang ditampilkan pada syntax query.

```
//Cek key dari field = key pada
diagram
If field = SelectedNode then
 For Each mTable In glbAFTables
    For i = 1 To mTable.Columns.Count
        //Tambah field query
        Set mQCol = New QRYQueryCol
        glbQueryCols.Add mQCol
        Exit For
        End If
    Next
    Next
```

Gambar 5. *Pseudocode* tambah field (pada query)

```
//Hapus kondisi yang ada
DelAllCondition()

//Tambah kondisi untuk relasi
//yang dibuat secara manual
For Each mRel In glbAFRelationsMan
 Set mCond = New QRYCondition
 glbConditions.Add , , , , , , , mCond
Next

//Tambah kondisi untuk relasi
//yang otomatis terbentuk
For Each mRel In glbAFRelations
 Set mCond = New QRYCondition
 glbConditions.Add , , , , , , , mCond
Next
```

Gambar 6. *Pseudocode* tambah kondisi (pada query)

KOMUNIKASI DATA XML

Untuk kebutuhan komunikasi antara client-server maka fungsi dan prosedur di bawah ini dipakai untuk mengirimkan XML ataupun menerima XML dan mengkonversi tag-tag XML ke dalam bentuk teks biasa. Gambar 8 menunjukkan pseudocode dari

prosedur yang digunakan untuk mengirimkan parameter-parameter dari client.

```
//Inisialisasi variabel XMLHTTP30
Dim mXMLReq As New XMLHTTP30

//Mengirim parameter-parameter client
mXMLReq.open "POST", url, bAsync
mXMLReq.send xmlData
//Return value dari proses
//yang telah dilakukan
SendXML = mXMLReq.responseText
```

Gambar 8. *Pseudocode* Mengirim Parameter ke *Server*

6. APLIKASI *SERVER*

Perangkat lunak yang dibuat pada lingkungan *server* merupakan sebuah ISAPI Extensions yang akan berjalan pada IIS. Perangkat lunak ini dibuat dengan menggunakan bahasa pemrograman Visual C++ 6. *Class* utama yang digunakan dalam ISAPI DLL ini adalah CQRYBuilderServer yang merupakan turunan dari *class* ChttpServer.

```
m_provUser = new CDataProvider();
dbProvider = "OraOLEDB.Oracle.1";
a = m_provUser->
Connect(dbProvider,mUser,mPass,mServic
e);
if (a.GetLength() == 0){
 OpenQueryTable();
 CreateTableObjectXML();
 OpenQueryRelation();
 CreateRelationObjectXML();

 strSchema.Format(docPtr->xml);
 *pCtxt << strSchema;
}
```

Gambar 9. *Pseudocode* proses ambil obyek-obyek basis data *server*

Pseudocode di atas ada 4 fungsi yang dijalankan, yaitu untuk fungsi OpenQueryTable dan OpenQueryRelation digunakan untuk mengambil obyek-obyek basis data server yang berhubungan dengan tabel dan relasi. Untuk kemudian obyek-obyek tersebut dikirim pada client melalui tag-tag XML yang di-generate oleh fungsi-fungsi CreateTableObjectXML dan CreateRelationObject XML.

Sedangkan Gambar 10 menunjukkan *pseudocode* dari proses pengiriman data hasil eksekusi *query* yang telah dikirm oleh perangkat lunak *client*.

```
m_provUser = new CDataProvider();
dbProvider = "OraOLEDB.Oracle.1";
a = m_provUser->
Connect(dbProvider,mUser,mPass,mServi
ce);
if (a.GetLength() == 0){
 CreateXMLData();

 strQuery.Format(docPtr->xml);
 *pCtxt << strQuery;
}
```

Gambar 10. *Pseudocode* proses pengiriman data pada client

QUERY METADATA

Untuk melakukan proses pembuatan query maka kita perlu mengetahui metadata atau obyek basis data yang terdapat dalam sebuah basis data Dimana obyek-obyek tersebut akan dikonversi menjadi tag-tag XML. Untuk mendapatkan seluruh nilai dari obyek basis data yang ada maka digunakan query untuk mendapatkan nilai-nilai tersebut, yaitu query untuk mendapatkan obyek tabel dan obyek relasi.

7. UJI COBA DAN EVALUASI

Uji coba terhadap perangkat lunak dilakukan dengan menggunakan basis data SCOTT, TA'ERS, TA1, SH, ES_MAIL dan PENDIDIKAN. Dari masing-masing tabel yang berada di basis data *server* tersebut akan diuji coba dengan melakukan login untuk mengetahui waktu yang dibutuhkan dalam mengambil obyek tabel dan *view* yang ada dalam sebuah basis data. Dan membuka data yang ada dalam masing-masing tabel untuk kemudian diketahui waktu yang dibutuhkan dalam proses eksekusi *query* yang ada. Uji coba yang akan dilakukan adalah sebagai berikut :

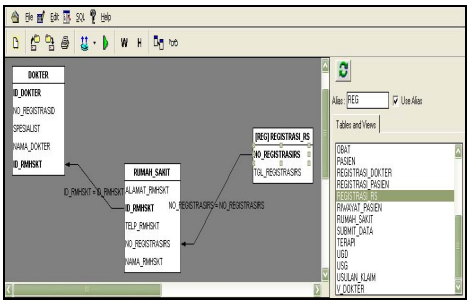
1. Uji coba kebenaran
2. Uji coba login dengan basis data yang berbeda
3. Uji coba login dengan peningkatan jumlah tabel
4. Uji coba login dengan peningkatan jumlah relasi
5. Uji coba eksekusi query dengan peningkatan jumlah record

UJI COBA KEBENARAN

Proses pelaksanaan uji coba kebenaran perangkat lunak dibagi menjadi beberapa bagian, yang akan dijelaskan sebagai berikut :

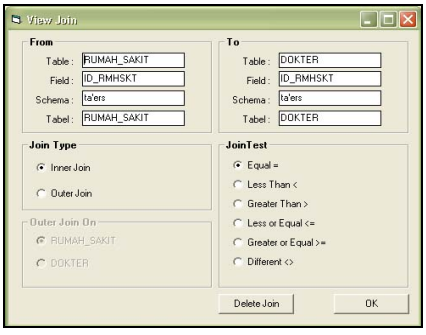
1. Uji coba menggambar diagram

Uji coba ini dilakukan dengan menggambar dan menghapus obyek diagram yang ada, yaitu tabel dan relasi. Penambahan tabel pada diagram dapat dilakukan dengan memilih daftar tabel pada bagian kiri aplikasi. Gambar 11 menunjukkan antar muka dari editor diagram pada perangkat lunak.



Gambar 11. Editor diagram

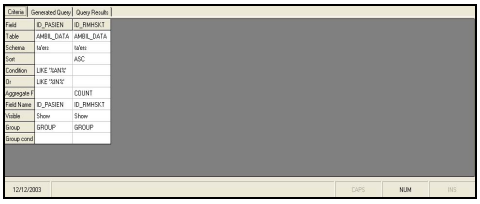
Selain itu properti dari masing-masing relasi yang terbentuk dapat diubah seperti pada gambar 12. Properti yang dapat dirubah tersebut, adalah tipe join apakah *inner join* atau *outer join*. Bila *outer join*, maka terhadap tabel apa serta operator dari relasi tersebut juga dapat dirubah. Gambar 12 menunjukkan antar muka dari properti relasi.



Gambar 12. Properti Relasi

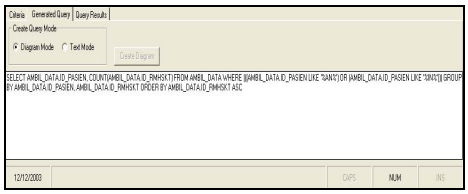
2. Uji coba kustomisasi kriteria field

Setelah melakukan proses penggambaran struktur SQL yang diinginkan maka selanjutnya dapat melakukan beberapa kustomisasi terhadap *field-field* yang telah dipilih. Bila tidak ada field yang dipilih maka tidak terdapat kolom tambahan pada editor field seperti yang tampak pada gambar 13, namun bila ada maka kolom bertambah sesuai dengan jumlah *field* yang dipilih.



Gambar 13. Editor kriteria field basis data

Dari beberapa perubahan yang telah dilakukan maka syntax SQL yang dibentuk dapat dilihat pada gambar 14. SQL tersebut kemudian akan dieksekusi untuk mendapatkan data.



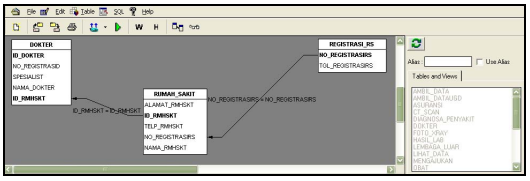
Gambar 14. Syntax SQL yang dihasilkan editor

3. Uji coba pembentukan visualisasi query dan sub query

Syntax SQL yang terdapat pada gambar 15 dapat dibentuk diagramnya dengan menekan tombol "*Create Diagram*". Namun sebelumnya untuk menjalankan proses ini maka *radio button* harus berada dalam posisi "*Text Mode*".

Gambar 15. Teks query

Diagram yang dihasilkan dari teks *query* pada gambar 15 ditunjukkan pada gambar 16.



Gambar 16. Diagram yang dihasilkan dari query

4. Uji coba eksekusi query

Uji coba kebenaran ini dilakukan untuk mengetahui apakah *syntax* SQL yang sudah dibentuk dapat dieksekusi pada level basis data dengan perangkat lunak yang terdapat pada *server*. Gambar 17 menunjukkan sebagian data hasil eksekusi *query* tersebut.

ID_RMHSKT	ID_DOKTER	NO_REGIS	NAMA_DOK	SPESIALIS	ID_RMHSK	NO_REGIS	NAMA_RMI	ALAMAT_R	TELP_RMH	NO_REGIS	TGL_REGIS
	80317	76332	88999	PUNGKY	KANDUNG/	80317	33333	RUMAH SA	JL. RAYA PI	2344029408	33333	12/3/2003
	80317	60961	11122	ADE	KANDUNG/	80317	33333	RUMAH SA	JL. RAYA PI	2344029408	33333	12/3/2003
	80317	34062	11122	ARDI	KANDUNG/	80317	33333	RUMAH SA	JL. RAYA PI	2344029408	33333	12/3/2003
	80317	60942	11122	FIRMAN	KANDUNG/	80317	33333	RUMAH SA	JL. RAYA PI	2344029408	33333	12/3/2003
	80317	35095	11122	MAMAN	KANDUNG/	80317	33333	RUMAH SA	JL. RAYA PI	2344029408	33333	12/3/2003
	67889	23580	44444	PUDJO	KANDUNG/	67889	44444	RUMAH SA	JL. RAYA K	2344029408	44444	12/4/2003
	67889	96635	44444	MIA	KANDUNG/	67889	44444	RUMAH SA	JL. RAYA K	2344029408	44444	12/4/2003
	67889	60268	44444	NIKEN	KANDUNG/	67889	44444	RUMAH SA	JL. RAYA K	2344029408	44444	12/4/2003
	66527	34579	11111	RIFQI	KANDUNG/	66527	44444	RUMAH SA	JL. RAYA CI	2344029408	44444	12/4/2003
	66527	36588	11111	BUDI	KANDUNG/	66527	44444	RUMAH SA	JL. RAYA CI	2344029408	44444	12/4/2003

Gambar 17. Data hasil eksekusi query

UJI COBA LOGIN DENGAN BASIS DATA YANG BERBEDA

Uji coba ini dilakukan untuk mengetahui waktu yang dibutuhkan pada saat login aplikasi. Uji coba ini dilakukan dengan menggunakan beberapa basis data yang memiliki jumlah tabel dan relasi yang berbeda serta ukuran masing-masing tabel yang berbeda (dalam MB). Spesifikasi basis data yang digunakan dalam uji coba ini ditunjukkan pada tabel 1 sebagai berikut:

Tabel 1. Spesifikasi basis data uji coba login

Nama Basis Data	Tbl	Rls	Klm	Ukuran (MB)
SCOTT	4	1	18	0.24
TA1	15	22	56	0.9
SH	15	8	160	89.18
TA'ERS	23	24	106	1.38
ES_MAIL	52	0	342	5.51
PENDIDIKAN	66	1	838	3.96

Tabel 2 menunjukkan waktu yang dibutuhkan dalam 5 kali proses login.

Tabel 2. Hasil uji coba login

Nama Basis Data	I	II	III	IV	V	Rata-rata (dtk)
SCOTT	1	1	0.77	0.77	0.78	0.864
TA1	16	7	16	6	4	9.8
SH	22	21	29	20	19	22.2
TA'ERS	17	14	31	15	20	19.4
ES_MAIL	92	203	88	106	90	115.8
PENDIDIKAN	501	561	511	499	491	512.6

Dari tabel 2 dapat diambil kesimpulan bahwa basis data dengan jumlah tabel yang besar membutuhkan waktu lebih lama untuk melakukan koneksi dengan basis data. Selain jumlah tabel yang besar, ukuran dari tabel yang besar juga mempengaruhi waktu yang dibutuhkan untuk melakukan koneksi terhadap basis data meskipun faktor tersebut tidak signifikan.

Pada contoh 3 dan 4 tabel 1 dapat dilihat bahwa basis data TA'ERS memiliki jumlah tabel yang lebih banyak daripada basis data SH. Namun pada data hasil uji coba tabel 2 dapat dilihat bahwa basis data SH membutuhkan waktu yang lebih lama daripada basis data TA'ERS untuk melakukan koneksi. Ini disebabkan karena selain faktor jumlah tabel, jumlah kolom dalam masing-masing tabel juga menentukan lamanya proses koneksi yang dilakukan, pada basis data TA'ERS jumlah kolom sebanyak 106 sedangkan pada basis data SH jumlahnya sebanyak 160 kolom.

UJI COBA LOGIN DENGAN PENING-KATAN JUMLAH TABEL

Uji coba ini dilakukan untuk mengetahui waktu yang dibutuhkan pada saat proses login dengan menggunakan faktor jumlah tabel. Basis data yang digunakan adalah PENDIDIKAN yang memiliki jumlah tabel sebanyak 66 buah. Namun untuk memenuhi keperluan uji coba maka dicoba dengan peningkatan jumlah tabel secara linier.

Uji coba login dengan peningkatan jumlah tabel secara linier ditunjukkan pada tabel 3. Dan untuk masing-masing peningkatan jumlah tabel dilakukan sebanyak 5 kali percobaan yang kemudian diambil hasil rata-ratanya.

Tabel 3. Uji coba peningkatan linier pada jumlah tabel

Jml Tbl	Percobaan					Rata rata (dtk)
	I	II	III	IV	V	
12	13	7	12	4	10	9.2
18	11	35	12	12	10	16
24	25	35	33	50	26	33.8
30	65	61	58	49	67	60
36	91	74	73	69	89	79.2
42	75	83	107	77	91	86.6
48	81	112	78	95	86	90.4
54	224	229	224	226	221	224.8
60	303	292	283	288	391	311.4
66	501	561	511	499	491	512.6

Dari tabel 3 dapat diambil kesimpulan bahwa dengan semakin banyaknya tabel atau *view* dalam sebuah basis data maka waktu yang dibutuhkan dalam proses koneksi pada basis data *server* juga lebih lama.

UJI COBA LOGIN DENGAN PENING-KATAN JUMLAH RELASI

Uji coba ini dilakukan untuk mengetahui waktu yang dibutuhkan pada saat proses login dengan menggunakan faktor jumlah relasi. Uji coba ini dilakukan dengan menggunakan basis data TA'ERS yang memiliki jumlah relasi sebanyak 24 buah. Namun untuk memenuhi keperluan uji coba maka dicoba dengan peningkatan jumlah relasi secara linier.

Uji coba login dengan peningkatan jumlah relasi secara linier ditunjukkan pada tabel 4. Dan untuk masing-masing peningkatan jumlah relasi dilakukan sebanyak 5 kali percobaan yang kemudian diambil hasil rata-ratanya.

Tabel 4. Uji coba peningkatan linier pada jumlah relasi

Jml Rel	Percobaan					Rata-rata (dtk)
	I	II	III	IV	V	
4	8	14	8	13	13	11.2
8	10	9	13	14	13	11.8
12	15	13	11	15	19	14.6
16	14	12	15	20	15	15.2
20	16	14	14	14	20	15.6
24	17	14	31	15	20	19.4

Dari tabel 4 dapat diambil kesimpulan bahwa semakin banyaknya relasi dalam sebuah basis data maka waktu yang dibutuhkan dalam proses koneksi pada basis data *server* juga lebih lama meskipun jumlah peningkatan waktu yang dibutuhkan tidak terlalu signifikan.

UJI COBA EKSEKUSI QUERY DENGAN PENINGKATAN JUMLAH RECORD

Uji coba ini dilakukan untuk mengetahui waktu yang dibutuhkan untuk melakukan eksekusi terhadap sebuah *query* ataupun *sub query*. Uji coba ini dilakukan dengan menggunakan basis data SH yang memiliki variasi dalam jumlah *records*. Tabel 5 menunjukkan spesifikasi masing-masing tabel pada basis data yang digunakan dalam uji coba.

Tabel 5. Spesifikasi data uji coba eksekusi query

Nama Tabel	Jml Klm	Jml Rec	Ukuran (MB/B)
CHANNELS	4	5	0.06 / 65536
COUNTRIES	5	19	0.06 / 65536
CAL_MONTH _SALES_MV	2	35	0.06 / 65536
PROMOTIONS	8	501	0.13 / 131072
TIMES	31	1461	0.38 / 393216
PRODUCTS	15	10000	3 / 3145728
CUSTOMERS	16	50000	9 / 9437184
FWEEK_PSCA T_SALES_MV	5	149325	7 / 7340032
COSTS	4	787766	26 / 27262976
SALES	7	1016271	43.25 / 45350912

Query yang digunakan dalam uji coba ini hanya merupakan *query*, SELECT * FROM (Nama Tabel). Pada Tabel 6 akan ditunjukkan waktu eksekusi bagi masing-masing tabel tersebut.

Tabel 6. Hasil uji coba eksekusi query

Nama Tabel	Waktu (detik)
CHANNELS	0.59
COUNTRIES	0.77
CAL_MONTH_SALES_MV	3.27
PROMOTIONS	24.51
TIMES	149.77
PRODUCTS	668.89
CUSTOMERS	5172.28 (5 record)
FWEEK_PSCAT_SALES_MV	12243.88 (2 record)
COSTS	- (tidak bisa)
SALES	- (tidak bisa)

Dari tabel 6 dapat diambil kesimpulan bahwa semakin banyak jumlah *record* yang dihasilkan dalam eksekusi sebuah *query*, maka waktu yang dibutuhkan untuk menampilkan data tersebut juga semakin lama. Selain faktor jumlah *record*, jumlah field yang ditampilkan dalam *query* juga merupakan faktor penting yang mempengaruhi waktu yang dibutuhkan untuk menampilkan data. Sebagai contoh pada tabel 6 untuk nama tabel Customers, banyaknya *field* yang ditampilkan hanya sebatas 5 *record* karena alokasi memori yang dibutuhkan untuk menampilkan data tersebut terbatas.

8. KESIMPULAN

Dalam implementasi perangkat lunak ini, telah dilakukan evaluasi terhadap proses perancangan dan pengembangan sistem. Berdasarkan evaluasi tersebut maka dapat diambil beberapa kesimpulan, yaitu :

Perangkat lunak dapat mengimplementasikan pembuatan syntax query dengan menggunakan diagram dan pembentukan diagram dengan menggunakan syntax query pada editor.

Dengan adanya perangkat lunak ini maka instalasi Oracle hanya dilakukan pada web server saja sehingga tiap-tiap komputer client tidak memerlukan instalasi tersebut namun harus terhubung pada web server jika ingin mendapatkan koneksi pada basis data server.

Faktor yang paling dominan dalam mempengaruhi kinerja dari perangkat lunak adalah jumlah tabel, jumlah kolom pada masing-masing tabel, dan jumlah record hasil eksekusi query yang terdapat dalam basis data. Dimana peningkatan waktu yang mempengaruhi kinerja dari perangkat lunak bersifat eksponensial.

9. DAFTAR PUSTAKA

1. ActiveX Control Basics.doc, 2003.
2. Dawes, Chip, & Thomas, Biju, *Introduction to Oracle 9iTM SQL*. Sybex Incoperation. London.
3. Hudson, Kurt, *Internet Infor-mation Server Overview and Architecture.*, http://www.window-sitlibrary.com/Content/141/01/, 1998.
4. Hunter, David. *Beginning XML*. Wrox Press. United States,
5. Lewis, Morris. *How to Use ActiveX Data Object*, http:// www. sqlmag.com/Articles/, 1999.
6. Lorentz, Diana, *Oracle 9i SQL Reference Release 2 (9.2)*, Oracle Corporation, Oktober 2002.
7. Microsoft Corporation, *Microsoft Developer Network (MSDN)*. http://www.msdn.microsoft.com., Juli 2001.
8. Rezin, K, *Perancangan dan Pembuatan Aplikasi Case Tool Untuk Melakukan Pemodelan Basis Data Fisik untuk RDBMS Oracle Berbasis Web*, Tugas Akhir Jurusan Teknik Informatika, Fakultas Teknologi Informasi Institut Teknologi Sepuluh Nopember Surabaya, 2003.
9. *Samba Server Documentation, ISAPI Extensions.* http: //www.samba.com/, 1998.
10. Skonnard, Aaron, *Inside Knowledge: Understanding the IIS Architecture.* http://www.microsoft. com/mind/1099/inside/inside1099.asp, 1999.
11. XML, *Document Object Model (DOM) Level 1 Spesification.pdf*, 1998.
12. Young, Michael, *Step By Step XML*, PT. Elex Media Komputindo, Jakarta, 2000.