OTOMATISASI SQL QUERY UNTUK DATABASE ENGINE

Arie Nugroho[1)], Arik Sofan Tohir[2)]

[1), 2)] Sistem Informasi Universitas Nusantara PGRI Kediri
Jl Mojoroto Gang I No.6 Kediri
Email : arieunp81@gmail.com[1)], arik.sofan.tohir@gmail.com[2)]

Abstrak

Desain database dalam sebuah aplikasi sistem informasi merupakan salah satu komponen yang sangat penting untuk menampung data dan menghasilkan informasi untuk kebutuhan pengambil keputusan. Dalam aplikasi sistem informasi, antara pengguna satu dengan yang lain biasanya menginginkan jenis database yang berbeda, sehingga pihak pengembang software dalam pembuatan aplikasi database membuat desain database ulang sesuai jenis database yang diinginkan. Hal tersebut membutuhkan waktu dan tenaga lebih untuk menyelesaikan aplikasi sistem informasi yang hampir sama. Hal ini bisa dihindari dengan melakukan generate otomatis dari file xml ke dalam bentuk perintah Structured Query Language(SQL) untuk mendeskripsikan desain table sehingga bisa diterapkan ke dalam jenis database yang berbeda sesuai kebutuhan. Hasil otomatisasi berupa bahasa basis data yaitu Data Definition Language(DDL) yang menjelaskan skema basis data dan Data Manipulation Language(DML) yang menjelaskan manipulasi data berupa stored procedure untuk perintah insert,update,delete,select. Database engine yang disupport adalah MySQL,SQLite dan Microsoft SQL Server.

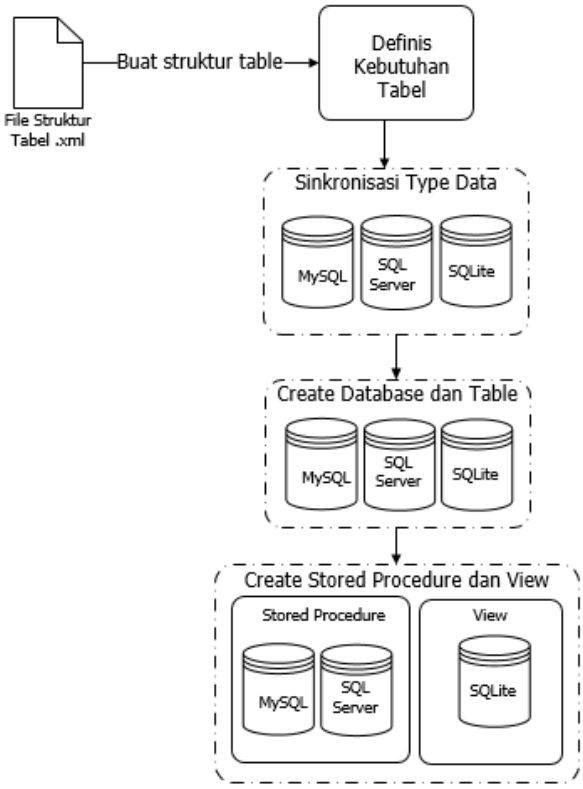
Kata kunci: *xml,database,SQL,DDL,DML*

1. Pendahuluan

Dalam menjalankan proses bisnis suatu instansi atau perusahaan membutuhkan aplikasi *database* agar dapat mempermudah pengolahan data. Data-data yang diolah dengan tepat dapat menghasilkan informasi yang dibutuhkan. Untuk membuat aplikasi *database* bisa menggunakan *database* versi gratris atau berbayar. Pihak pengguna membutuhkan aplikasi *database* sesuai dengan kebutuhan atau anggaran perusahaannya. Tentunya pihak pengembang harus mengikuti kebutuhan pengguna atau bisa memberikan masukkan terkait dengan kebutuhan pengguna. Pihak pengembang dalam waktu yang hampir bersamaan bisa menerima permintaan aplikasi *database* dengan desain sistem yang sama. Dengan perbedaan jenis *database* yang dibutuhkan membuat pihak pengembang membuat ulang desain *database*, hal ini tentunya membutuhkan waktu dan tenaga. Dalam penelitian ini akan dibahas solusi untuk memudahkan pengembang sistem dalam membuat *database* untuk beberapa jenis *database engine* dalam sistem yang hampir sama. SQL adalah bahasa *database* untuk membuat *query* dan memanipulasi *database* relasional. Menulis dan menjalankan *query* di SQL adalah bagian dari pembahasan *database* relasional. SQL telah distandartkan oleh American National Standards Institute (ANSI) and the International Organization for Standardization (ISO)[1].Otomatisasi secara signifikan dapat mengurangi biaya testing yang dapat meningkatkan efektivitas *testing* secara signifikan[2]. Data Definition Language(DDL) adalah struktur basis data yang menggambarkan skema basis data secara keseluruhan dan didesain dengan bahasa khusus[3]. Data Manipulation Language(DML) adalah bentuk bahasa basis data yang digunakan untuk melakukan manipulasi dan pengambilan data pada suatu baris data[3]. XML adalah bahasa *markup* yang dirancang khusus untuk penyampaian informasi melalui *World Wide Web* (WWW). XML merupakan suatu format data berbasis teks yang membantu *Developer* dan *Provider* dalam menggambarkan, mengirimkan dan menukarkan data yang terstruktur dari antara berbagai macam aplikasi kepada pengguna untuk keperluan manipulasi[4].

2. Pembahasan

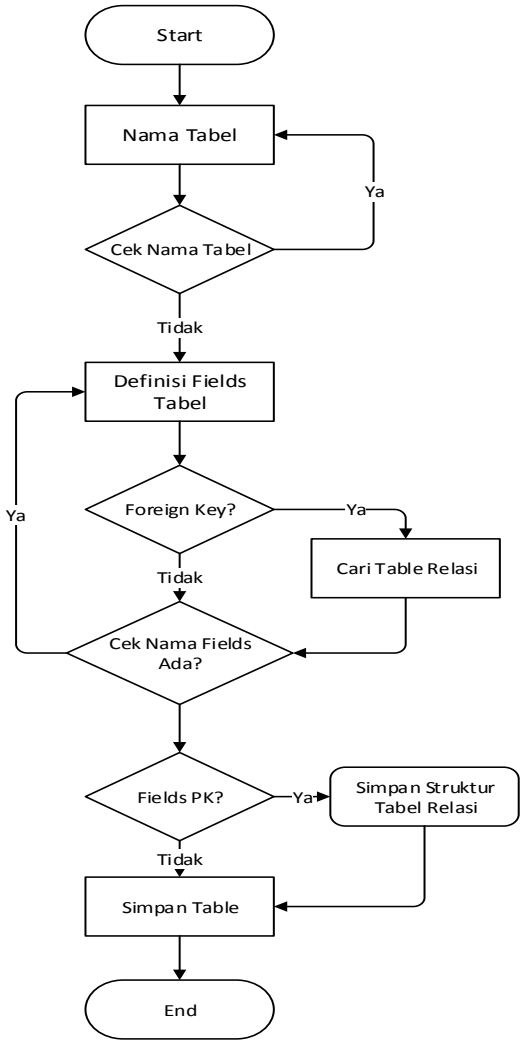
Dalam penelitian ini dibuat aplikasi untuk otomatisasi sql *query* untuk beberapa *database engine*,yaitu MySQL,SQLite dan Microsoft SQL Server. Ini akan sangat membantu pengembang sistem menghemat waktu dan tenaga dalam pembuatan *database*, *table*, *stored procedure* untuk sistem yang sama atau yang hampir sama dalam beberapa *database engine*. Penelitian ini tidak membahas normalisasi *table* sehingga pengembang sistem diasumsikan sudah mempunyai rancangan *table* yang dibutuhkan. Proses dari otomatisasi sql *query* dibagi menjadi beberapa tahap, yaitu mendefinisikan kebutuhan *table* dari *file* struktur *table* yang berupa *file* xml, sinkroninsasi *type* data sesuai dengan *database engine* yang akan digunakan, *create database* dan *table*, *create stored procedure*(untuk *database engine* MySQL dan SQL server) dan *view*(untuk *database engine* SQLite) sesuai dengan *database engine* yang akan digunakan. Proses otomatisasi sql *query* ditunukkan pada gambar 1.



Gambar 1.*Proses Otomatisasi SQL Query*

File struktur *table* .xml merupakan *file* dasar untuk membentuk struktur *table*, dimana dalam *file* terserbut terdapat struktur untuk membentuk sebuah *table* yang lengkap dengan definisi (*Fields Name*, *Data Type*, *Length*, *Primary key*, *Foreign key*). Dari hasil pendefinisian *table* akan dilakukan proses sinkronisasi *type* data sesuai dengan *database engine* yang digunakan. Proses selanjutnya setelah sinkronisasi *type* adalah proses untuk membuat *database* dan *table* (Create *Database*, Create table) sesuai dengan *database engine* yang digunakan. Setelah pembuatan *database* dan table berhasil maka selanjutnya dibuatlah *stored procedure* dan *view*. Pada proses ini ada pebedaan untuk *database Engine* MySQl, SQL Server dan SQLite. Untuk *database engine* MySQl dan SQL Server akan dibuat untuk *Stored procedure* (Insert, Update, Delete dan Select). Untuk *database engine* SQLite tidak dibuat *stored procedure* karena *database engine* ini belum mendukung fasilitas *stored procedure*, sehingga khusus untuk *database engine* ini hanya dibuatkan *view*.

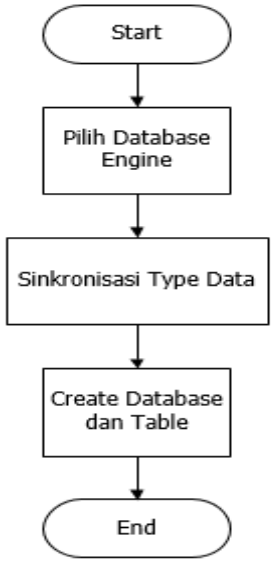
Berikutnya akan dijelaskan secara detil berupa *flowchart* dari masing-masing proses di atas. Pertama adalah *flowchart* untuk proses definisi kebutuhan *table* di sebuah sistem yang akan dibangun.



Gambar 2.*Flowchart* Proses Definisi Kebutuhan *Table*

Pada gambar 2 menunjukkan proses pembuatan *table* yang diperlukan dalam pembangunan sebuah sistem dengan Otomatisasi *query*. Pertaman dengan menentukan nama *table*, apabila nama *table* sudah pernah dibuat maka proses akan dikembalikan lagi ke proses penamaan *table* dan apabila nama *table* belum ada maka dikukan proses pendefinisian nama *field*s atau atribut-atribut yang diperlukan pada oleh *table*. Pada proses pendifinisian *field*s atau atribut jika sebuah atribut merupakan *foreign key* atau ada relasi dengan *table* lain, maka dilakukan proses pemilihan *table* yang menjadi relasi. Setelah *field*s dibuat dan dilakukan proses pengecekan nama, berikut nya dicek lagi apakah *field* yang baru dibuat merupakan *field*s *foreign key* atau tidak, jika berupa *field*s *foreign key* maka secara otomatis akan terbentuk *table* relasi dan semua struktur *table* disimpan.

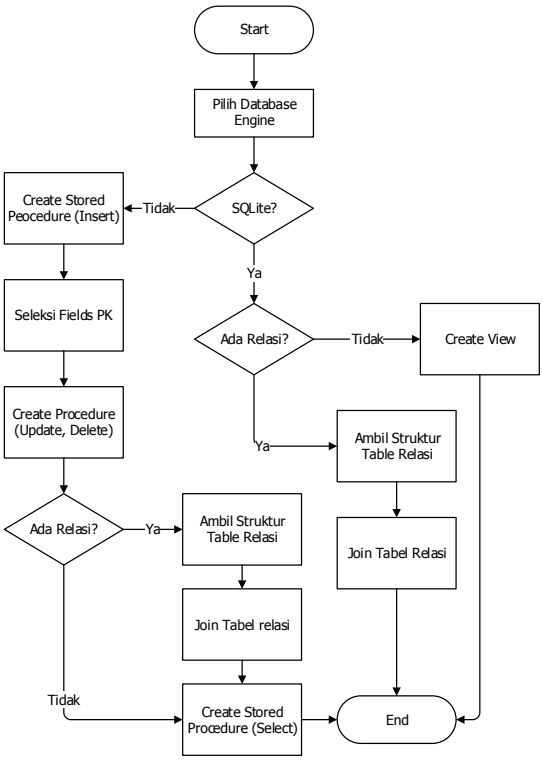
Berikutnya *flowchart* untuk proses sinkronisasi *type* data serta create *database* dan *table* dari definisi kebutuhan *table* ke dalam *database engine* yang ditentukan.



Gambar 3.*Flowchart* Sinkrinisasi *type* data serta proses create *database* dan *table*

Proses create *database* dan *table* merupakan proses untuk membuat membuat sebuah perintah *Data Definition Languange* (DDL) yang menghasilkan *query* untuk membuat *database* dan *table*. Pada proses ini pelu di tentukan terlebih dahulu jenis *database engine* yang digunakan. Di karekan setiap *database engine* memiliki type data yang berbeda maka perlu dilakukan sinkrosiniasi jenis *type* data yang sesuai dengan *database engine* yang digunakan.

Berikutnya *flowchart* untuk proses create *stored procedure* dan *view*

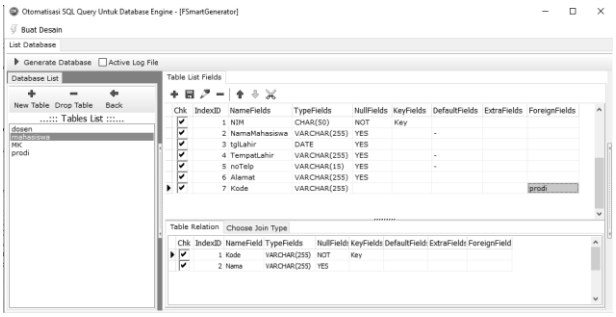


Gambar 4.*Flowchart Stored procedure* dan *view*

Proses berikutnya setelah *database* dan *table*e terbuat adalah pembuatan *Stored procedure* dan *View*. Dari gambar proses tersebut dapat kita lihat terdapat dua proses secara umum yaitu untuk *database* SQLite dan bukan SQLite. Untuk proses *database* selain SQLite proses yang akan dijalankan pertama kali adalah pembuatan *Stored procedure Insert* yang digunakan untuk perintah memasukasukkan data baru pada *table*. Proses selanjut nya adalah pembuatan *Stored procedure Update* dan *Delete*, pada pembuatan *stored Update* dan *delete* diperlukan proses untuk memflter *field*s yang menjadi *Primary key* pada *table*, hal ini perlu dilakukan untuk membentuk DDL yang diakan dijadikan acuan untuk melakukan proses updata dan delete pada *table*. Proses selanjut nya adalah pembuatan *stored procedure* untuk select. Untuk proses ini dibagi menjadi dua proses yaitu untuk *table* yang memiliki relasi dan *table* yang tidak memiliki relasi. Untuk *table* yang tidak memiliki relasi, procedure select langsung di create. Untuk *table* yang memiliki relasi perlu lakukan proses pembacaan struktur *file* yang menjadi relasi, hal ini perlu untuk mendapatkan *field*s-*field*s yang di definisikan untuk dimasukkan dalam *join* pada *stored procedure select*. Untuk jenis *database engine* SQLite karena belum memiliki fasilitas *stored procedure* maka khusus untuk *database* SQLite hanya akan membentuk sebuah *view* dari masing-masing *table*, baik *table* yang memiliki relasi maupun *table* yang tidak memiliki relasi.

Data uji yang digunakan adalah sistem informasi akademik. Berikut ini langkah pengujian dari proses otomatisasi sql *query*.

1. Pembuatan struktur atau desain *table*, yaitu pembuatan definisi *table* berupa nama *table*, nama *field*, *type* data, panjang *field*, definisi *primary key*. Setelah semua *table* dibuat, berikutnya adalah pengaturan relasi antar *table*. Hasil dari proses ini adalah *file* xml setiap *table* dan relasi antar *table*. Contoh desain *table* ditunjukkan pada gambar 5.



Gambar 5.*Desain Database*

Pada gambar 5 dicontohkan *database* akademik dengan empat *table*, yaitu dosen, mahasiswa, MK(matakuliah) dan prodi. *File* xml yang dihasilkan adalah struktur setiap *table* dan relasinya. Contoh *file* mahasiswa.xml ditunjukkan dengan gambar 6

Gambar 6.*File mahasiswa.xml*

Pada gambar 6 adalah *file* mahasiswa.xml yang mempunyai struktur *field* nim, nama, tempattgllahir, tgllahir,notelp,alamat,kodeprodi.

2. Dari *file* xml setiap *table* yang dihasilkan kemudian digenerate menjadi *file script* untuk definisi *table* (DDL), berupa *file* sql setiap *table* lengkap dengan relasinya. Contoh *file script* sql ditunjukkan gambar 7

Log File

```
DROP TABLE IF EXISTS dosen;
CREATE TABLE dosen( NIDN CHAR(50) NOT NULL,
NamaDosen VARCHAR(255) NULL,
Gelar VARCHAR(255) NULL,
Kode VARCHAR(255) NULL ,
PRIMARY KEY (NIDN));

DROP TABLE IF EXISTS mahasiswa;
CREATE TABLE mahasiswa(
NIM CHAR(50) NOT NULL,
NamaMahasiswa VARCHAR(255) NULL DEFAULT '-' ,
tglLahir DATE NULL  ,TempatLahir VARCHAR(255) NULL DEFAULT '-' ,
noTelp VARCHAR(15) NULL DEFAULT '-' ,
Alamat VARCHAR(255) NULL,
Kode VARCHAR(255) NULL ,
PRIMARY KEY (NIM)
);

DROP TABLE IF EXISTS MK;
CREATE TABLE MK(
KodeMK CHAR(50) NOT NULL,
NamaMK VARCHAR(255) NULL,
SKS INTEGER NULL ,
PRIMARY KEY (KodeMK)
);

DROP TABLE IF EXISTS prodi;
CREATE TABLE prodi(
Kode VARCHAR(255) NOT NULL,
Nama VARCHAR(255) NULL,
PRIMARY KEY (Kode)
);
```

Gambar 7.*script sql*

Pada Gambar 7 merupakan *script* sql yang dihasilkan, untuk *table* dosen, mahasiswa, MK dan prodi untuk *database* MySQL

3. Dari *file* sql kemudian digenerate menjadi *stored procedure* (*insert,update, delete* dan *select join table*). Pada gambar 8 adalah contoh *file stored procedure* untuk *table* dosen

Log File

```
DROP PROCEDURE IF EXISTS dosen_insert;
CREATE PROCEDURE dosen_insert(fNIDN CHAR(50),fNamaDosen VARCHAR(255),
fGelar VARCHAR(255),fKode VARCHAR(255))
BEGIN
INSERT INTO akademik.dosen (NIDN,NamaDosen,Gelar,Kode)
VALUES (fNIDN,fNamaDosen,fGelar,fKode);
END;

DROP PROCEDURE IF EXISTS dosen_update;
CREATE PROCEDURE dosen_update(fNIDN CHAR(50),
fNamaDosen VARCHAR(255),fGelar VARCHAR(255),
fKode VARCHAR(255),idNIDN CHAR(50))
BEGIN
UPDATE akademik.dosen SET NIDN=fNIDN,NamaDosen=fNamaDosen,
Gelar=fGelar,Kode=fKode
WHERE NIDN=idNIDN;
END;

DROP PROCEDURE IF EXISTS dosen_delete;
CREATE PROCEDURE dosen_delete(f_ppi_id int)
BEGIN
DELETE FROM akademik.dosen
WHERE NIDN=fNIDN;
END;


DROP PROCEDURE IF EXISTS dosen_select;
CREATE PROCEDURE dosen_select()
BEGIN
SELECT
dosen.NIDN,dosen.NamaDosen,dosen.Gelar,dosen.Kode,
prodi.Kode,prodi.Nama
FROM
akademik.dosen
LEFT OUTER JOIN prodi ON prodi.Kode=dosen.Kode_prodi;
END
```

Gambar 8.*file stored procedure table dosen*

Pada gambar 8 adalah contoh *stored procedure* yang dihasilkan untuk *table* dosen, yaitu *storerd procedure insert*, *update*, *delete* dan *select(join)*

Hasil pengujian otomatisasi sql *query* untuk *stored procedure* di *database engine* SQL Server ditunjukkan pada gambar 9

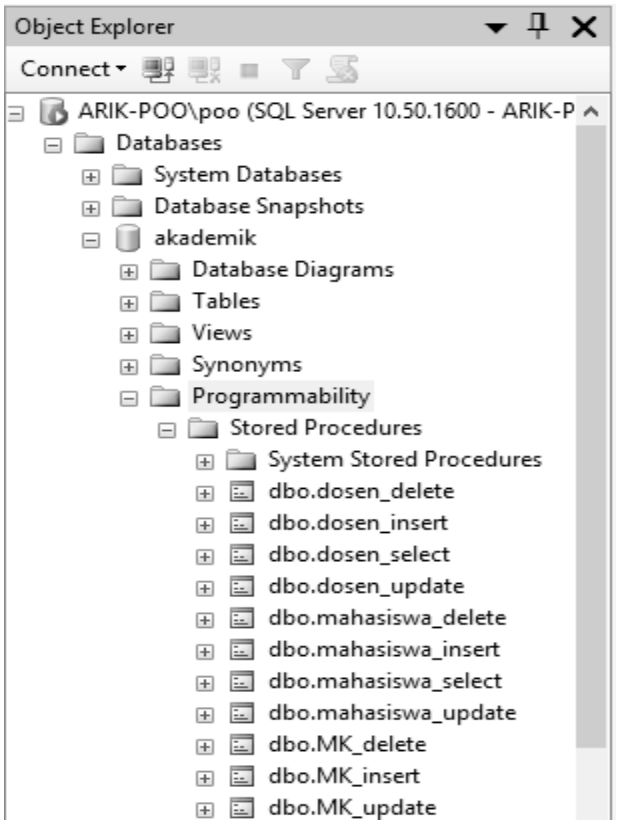
```
if not exists ( select * from sys.objects
where name='dosen_insert'
and objectproperty(object_id,'IsProcedure')=1 )
exec('CREATE PROCEDURE [dosen_insert](
@NIDN CHAR(50),
@NamaDosen VARCHAR(255),
@Gelar VARCHAR(255),
@Kode_prodi VARCHAR(255))
AS
SET NOCOUNT ON;
BEGIN
INSERT INTO .dosen (NIDN,
NamaDosen,Gelar,Kode_prodi)
VALUES (@NIDN,@NamaDosen,
@Gelar,@Kode_prodi);
SELECT SCOPE_IDENTITY()
END;')

if not exists ( select * from sys.objects
where name='dosen_update'
and objectproperty(object_id,'IsProcedure')=1 )
exec('CREATE PROCEDURE [dosen_update](
@NIDN CHAR(50),
@NamaDosen VARCHAR(255),
@Gelar VARCHAR(255),
@Kode_prodi VARCHAR(255),@idNIDN CHAR(50))AS
SET NOCOUNT ON;
BEGIN
UPDATE dosen SET
NIDN=@NIDN,
NamaDosen=@NamaDosen,
Gelar=@Gelar,
Kode_prodi=@Kode_prodi
WHERE NIDN=@idNIDN;
SELECT SCOPE_IDENTITY()
END;')
```

Gambar 9. Stored Procedure Database Engine SQL Server

Pada gambar 9 ditunjukkan perintah query untuk membuat stored procedure pada database engine SQL Server. Pada otomatisasi query ini berfungsi untuk menghasilkan sebuah perintah query yang digunakan untuk membuat strored procedure pada SQL Server. Hasil stored procedure yang dihasilkan adalah stored procedure untuk Insert data, Update data, delete data data dan select. Untuk tabel yang memiliki relasi dengan tabel lain stored procedure select akan mengasilkan join dengan tabel lain dan menggunakan penghubung *Left Outer Join.*

Hasil struktur database (*table*,*stored procedure*) di *database engine* SQL Server, dari SQL server Management Studio ditunjukkan pada gambar 10



Gambar 10. SQL Server Management Studio

Pada gambar 10 ditunjukkan hasil pembuatan *stored procedure* untuk masing-masing kebutuhan tabel (*Insert*, *Update*, *Delete*, *Select*). Dengan hasil tersebut dapat ditarik kesimpulan bahwa hasil desain tabel yang tersimpan pada file xml dapat digunakan untuk membuat atau membentuk *stored procedure* pada *database engine* SQL Server. Baik untuk kebutuhan penambahan data, ubah data, penghapusan data dan untuk menampilkan data pada sebuah tabel.

3. Kesimpulan

Dari hasil penelitian ini file XML dapat di implementasikan untuk membuat otomatisasi SQL untuk membentuk database dan struktur tabel, stored procedure (*insert*, *update*, *delete* dan *select join table*) untuk database engine MySQL dan Microsoft SQL Server. File XML juga dapat digunakan untuk membentuk database dan stuktur tabel dan membuat view pada database engine SQLite.

Saran untuk penelitian ini adalah menambahkan *database engine* yang lain serta jika ada perubahan *database* (*field* dan *table*) didesain awal, setelah diotomatisasi tidak akan mempengaruhi data di *table database engine* yang telah dibuat.

Daftar Pustaka

[1] Quan Do,Rajeev K Agrawal,Dhana Rao,Venkat N Gudivada, "Automatic Generation of SQL Queries," American Society for *Engine*ering Education, Paper ID #8958,June 2014.
[2] Abdul Shadi Khalek and Sarfraz Khurshid. "Automated sql *query* generation for systematic testing of *database engine*s" IEEE/ACM international conference on Automated software *engine*ering, ASE '10, pages 329–332, New York, NY, USA, 2010. ACM
[3] Fathansyah, "Basis Data" Edisi Revisi, Hal.18, Juni, 2012.
[4] F.Samopa,D.H.Murti,O.Oktanio."Sistem *Query* Pada Dokumen Xml Dengan Menggunakan Bahasa Sql". Jurnal Ilmiah Teknologi Informasi, Vol.4,Hal 133-140,Juli 2005

Biodata Penulis

ARIE NUGROHO,memperoleh gelar Sarjana Komputer (S.Kom), Jurusan Sistem Informasi STMIK KADIRI Kediri, lulus tahun 2006. Memperoleh gelar Magister Manajemen (M.M.) Program Pasca Sarjana Magister Manajemen Konsentrasi Teknologi dan Informasi Universitas ISLAM Malang, lulus tahun 2011.Saat ini menjadi Dosen di Universitas Nusantara PGRI Kediri.

ARIK SOFAN TOHIR,memperoleh gelar Sarjana Komputer (S.Kom), Jurusan Sistem Informasi STMIK KADIRI Kediri, lulus tahun 2011.Saat ini menjadi Dosen di Universitas Nusantara PGRI Kediri.

ISSN : 2302-3805