# ANALISA OPTIMALISASI BAHASA SQL BERDASARKAN RELATIONAL ALGEBRA PADA KASUS REKAPITULASI MAHASISWA LAYAK WISUDA

**Eko Darmanto**
Fakultas Teknik, Program Studi Sistem Informasi
Universitas Muria Kudus
Email: bi_anto97@yahoo.com

**ABSTRAK**

Aljabar relasional sangat membantu adanya kemungkinan penggunaan sintaks bahasa SQL yang berlainan dalam masalah yang sama. Masalah yang digunakan sebagai eksperimen adalah kasus rekapitulasi data mahasiswa layak wisuda pada suatu perguruan tinggi. Penggunaan sintaks bahasa SQL yang optimal dapat dianalisa menggunakan teknik *query tree* untuk mendapatkan hal utama dalam pemrosesan basis data yaitu kecepatan dan ketepatan komputasi yang berhubungan dengan pemuatan data ke dalam memori komputer. Hasil dari penelitian menunjukkan bahwa pemuatan data ke dalam memori komputer lebih optimal jika menggunakan *join* dibandingankan dengan *Cartesian product*. Dimungkinkan gabungan keduanya menjadi lebih cepat jika aturan penggunaan seleksi kemudian proyeksi didahulukan ketimbang penggabungan menggunakan *Join* atau *Cartesian product* terlebih dulu, hal ini bertujuan untuk mengurangi pemuatan baris dan kolom data yang berlebih. Ditinjau dari kecepatan komputasi untuk operasi himpunan yang melibatkan lebih dari satu tabel, *Cartesian product* dengan didahului proyeksi dan seleksi lebih cepat dibandingkan dengan *Join* untuk kasus yang sama.

**Kata kunci:** optimalisasi sintaks SQL, aljabar relasional, *query-tree*.

***ABSTRACT***

*Relational algebra is helpful for the possible use of the SQL syntax of different languages in the same problem. Issues that are used as experimental data recapitulation is a case worthy student at a college graduation. The use of optimized SQL language syntax can be analyzed using query-tree techniques to get the main thing in database processing speed and accuracy that computation associated with loading data into the computer's memory. Results from the study showed that loading data into the computer's memory if you use a more optimal than the Cartesian product join. Possible combination of the two to be faster if the usage rules take precedence over the selection and then projected merger using Join or Cartesian product first, it aims to reduce the loading of rows and columns of data that excess. In terms of computing speed for the set of operations involving more than one table, the Cartesian product preceded projection and selection Join faster than for the same case.*

***Keywords:*** *SQL syntax optimization, relational algebra, query-tree.*

## 1. PENDAHULUAN

Rekapitulasi mahasiswa layak wisuda pada suatu perguruan tinggi biasanya memiliki beberapa persyaratan. Persyaratan yang umum digunakan adalah persyaratan akademik dan administrasi (non akademik). Persyaratan akademik misalnya sudah menyelesaikan seluruh mata kuliah yang dimiliki oleh setiap program studi. Penyelesaian dalam menempuh mata kuliah memiliki persyaratan didalamnya yaitu indeks prestasi minimal setiap mahasiswa sudah terpenuhi, telah menempuh mata kuliah muatan lokal, jumlah nilai minimal yang diperbolehkan tidak dilanggar dan semua mata kuliah yang telah ditempuh memiliki kode yang sama atau pada kurikulum yang sama (jika terdapat beberapa kurikulum yang jalan bersamaan).

Makalah ini akan menjawab permasalahan persyaratan akademik yang berupa mata kuliah sudah ditempuh semuanya, atau SKS minimal program studi terpenuhi, nilai D tidak lebih dari 3 (tiga), Indeks Prestasi Kumulatih (IPK) lebih dari atau sama dengan 2.5, sudah menempuh mata kuliah Metodologi Penelitian, Praktek Kerja Lapangan, lulus mata kuliah Skripsi, setiap mata kuliah yang di ambil harus memiliki kode mata kuliah yang sejenis (memiliki kesamaan kurikulum), pernah menempuh kuliah muatan lokal yang ditandai dengan memiliki sertifikat. Pengecekan yang dilakukan oleh bagian administrasi untuk kelayakan wisuda biasanya dilakukan permahasiswa dan seluruh mahasiswa yang telah mendaftar wisuda sebagai rekapitulasi akhir.

Persyaratan non akademik misalnya penyelesaian administrasi keuangan, tidak memiliki pinjaman buku perpustakaan, dan persayaratan lainnya tidak dibahas dalam makalah ini. Permasalahan yang diangkat adalah yang hanya berkaitan erat dengan data-data persyaratan akademik yang jelas keberadaannya dalam sebuah basis data sistem akademik suatu perguruan tinggi. Berdasarkan basis data tersebut dapat dianalisa bagaimana menggunakan bahasa SQL (*structured query language*) yang hemat dan cepat yang dianalisa menggunakan aljabar relasi (*relational algebra*). Data-data sebagai penunjang eksperimen digunakan data simulasi yang mendekati dengan data sebernarnya, karena data yang sesungguhnya bersifat privasi.

## 2. METODOLOGI PENELITIAN

Langkah-langkah yang dilakukan pada penelitian ini meliputi:

1) Analisa objek, objek yang digunakan adalah sebuah perguruan tinggi yang akan mewisuda mahasiswanya dengan kriteria memiliki sejumlah mahasiswa, sejumlah mata kuliah inti, sejumlah mata kuliah muatan lokal yang dijadikan sebagai syarat wisuda. Objek memiliki sejumlah aturan yang harus dipatuhi untuk melaksanakan wisuda, baik aturan akademik maupun non akademik.
2) Penggalian data, data-data asli digali dan dilakukan analisa awal tentang kecenderungan adanya keterhubungan antar data. Kecenderungan hubungan ini harus bisa dimodelkan menggunakan alat pemodelan data berupa ER-Diagram. Pada saat implementasi menggunakan data simulasi, dengan alasan tetap menjaga privasi dari data-data sebuah perguruan tinggi.
3) Penemuan akar masalah, berdasarkan analisa objek dan data yang ada, akar masalah harus bisa diidentifikasi dengan baik. Misalnya masalah data yang sudah terintegrasi dengan baik tetapi belum dapat ditemukan langkah optimal pemrosesannya sehingga memperlama proses *query*.
4) Menentukan solusi sementara, solusi sementara ini memberikan gambaran jika langkah awal penggunaan data sudah dilakukan analisa optimalisasi menggunakan metode tertentu, maka akan menghemat biaya proses dan lebar pita koneksitas untuk akses data.
5) Pengumpulan teori formula Aljabar Relasi, metode yang digunakan untuk analisa optimalisasi adalah dengan menggunakan perumusan aljabar relasi yang kemudian diterjemahkan dalam bahasa SQL.
6) Perancangan basis data, tahap ini merupakan tahap yang paling penting dalam pemodelan data relasional. Alat perancangan yang digunakan adalah ER-Diagram dengan notasi model Chen [1] dan dipetakan dalam bentuk skema relasional basis data.
7) Implementasi basis data, untuk menganalisa kecepatan proses, basis data relasional hasil rancangan diimplementasikan dalam sebuah DBMS. DBMS yang dipakai adalah MySQL karena bersifat bebas pakai non komersial dengan lisensi publik.
8) Analisa menggunakan Aljabar Relasional dan Bahasa SQL, untuk mendefinisikan struktur data, manipulasi, dan pengontrolan akses diperlukan DBMS yang memiliki bahasa SQL yang kuat. Penerjemahan dari Aljabar Relasi menjadi sintaks dalam bahasa SQL dilakukan menggunakan analisa dari eksperimen model *trial and error*.
9) Analisa Optimalisasi Proses, setiap hasil dari pemrosesan *query* dianalisa lama prosesnya sebagai faktor biaya. Semakin lama/lambat prosesnya maka diasumsikan akan memakan banyak biaya sumber daya komputasi yang ada. Sumber daya komputasi yang dipakai adalah kapasitas penyimpan memori sementara RAM, kecepatan prosesor dan lebar pita untuk transmisi data dan medianya.

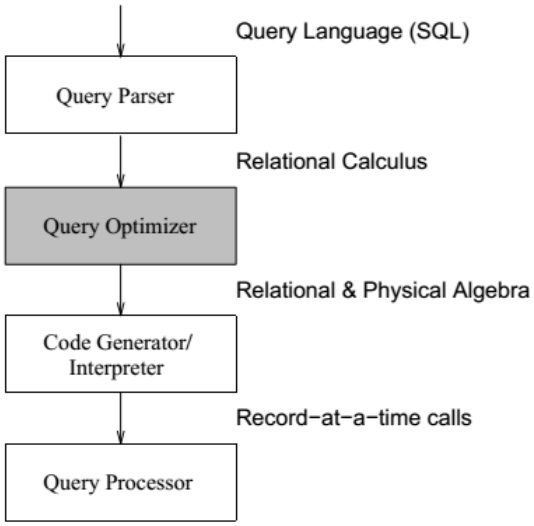
## 3. TINJAUAN PUSTAKA

SQL atau disebut juga dengan SEQUEL (*Structured English Query Language*) merupakan bahasa pemrograman yang memiliki tujuan khusus dan dirancang untuk mengelola data dalam sistem manajemen database relasional (RDBMS-*Relational Database Management Systems*), atau untuk pengolahan aliran data dalam sistem manajemen basis data relasional [2]. SQL memiliki tiga bagian utama yaitu bahasa pemrograman untuk mendefinisikan data (*Data Definition Language*-DDL), untuk manipulasi dan akses data (*Data Manipulation Language*-DML) dan bagian yang digunakan untuk pengawasan/kontrol pemakai (Data Control Language). Bahasa SQL masih memiliki beberapa kekurangan diantaranya adalah ketiga bahasa tersebut harus terintegrasi menggunakan suatu bahasa pemrograman tertentu [3] [4]. Bahasa SQL atau SEQUEL dibangun atas dasar *Relational Algebra*. Secara luas pemakaiannya telah distandarkan dalam sebuah kerangka kerja yang terdaftar dalam *International Organization for Standardization* (ISO) pada tahun 1987 [5].

Aljabar relasional (*relational algebra*) telah dikemukakan oleh EF Codd yang merupakan keluarga aljabar dengan semantik yang dibangun untuk memodelkan data yang disimpan dalam basis data relasional dan mendefinisikan *query*-nya. Aljabar Relasional (AR) pertama kali digunakan untuk menganalisa secara matematis data perbankan yang sangat besar dengan 5 (lima) operasi dasar yaitu seleksi (*selection*), proyeksi (*projection*), perkalian (*Cartesian product*), penggabungan himpunan (*set union*) dan selisih himpunan (*set difference*) [6]. *Cartesian product* dalam beberapa literatur disebut juga *cross product* atau *cross join*, sedangkan *set difference* terdapat empat bagian yaitu *intersection*, *left outer join*, *right outer join* dan *outer join* [7][8]. Disamping aljabar relasi juga terdapat *relational calculus*. Kalkulus relasional terdiri dari dua kalkuli yaitu kalkulus relasional tupel dan kalkulus relasional domain, yang merupakan bagian dari model basis data relasional yang menyediakan cara deklaratif untuk menentukan *query* basis data [3] [8].

Pemodelan basis data dapat dilakukan dengan berbagai macam cara, diantaranya dapat dilakukan dengan teknik Normalisasi dan langsung menggunakan ER-Diagram [3] [8]. Pada makalah ini hanya akan menggunakan pemodelan data menggunakan ER-Diagram dengan notasi gambar seperti yang diperkenalkan oleh Chen [1]. Meskipun menggunakan pemodelan ER-Diagram, masih tunduk terhadap aturan-aturan untuk memperoleh basis data yang baik menggunakan teknik Normalisasi. Teknik Normalisasi sendiri memiliki beberapa tahapan setiap tahap memiliki bentuk normal (*Normal Form*) [3] [8] [9]. ER-Diagram hanya memiliki 2 (dua) komponen utama yaitu himpunan entitas (*entity set*) dan relasinya (*relationships*). Namun demikian notasi yang digunakan berupa persegi panjang untuk entitas, belah ketupat untuk relasi, lingkaran atau elips berupa atribut dan garis sebagai penghubung sekaligus sebagai informasi kardinalitas hubungan relasionalnya [1].

Pemodelan data menggunakan ER-Diagram diterjemahkan dalam skema relasional tanpa memerlukan definisi dari struktur data dari tabel atau relasi tersebut. Sampai pada tahap pemetaan/penterjemahan ini istilah tabel sama dengan relasi, dengan ciri yang sama-sama memiliki kolom (*attribute*) dan baris data. Skema relasional ini dijadikan dasar untuk operasi-operasi dalam aljabar relasional. Hasil dari aljabar relasional diterjemahkan dalam bahasa SQL, langkah selanjutnya yaitu mencari alternatif hasil terjemahan lainnya yang memiliki arti dan hasil yang sama kemudian membandingkan hasilnya. Hasil perbandingan ini dianalisa berdasarkan lama waktu proses dengan asumsi berdasarkan jumlah data yang tersedia berjumlah sangat banyak. Hasil aljabar relasional yang telah diterjemahkan dalam bahasa SQL dioptimalisasi sesuai dengan alur proses dalam eksekusi sintaks bahasa SQL. Eksekusi ini memiliki urutan yang digunakan sebagai acuan untuk optimasi yang disajikan pada Gambar 1 berikut [10];




**Gambar 1. Alur Pemrosesan Dalam Optimasi *Query* [10]**

## 4. NOTASI ALJABAR RELASIONAL

Terdapat 5 (lima) notasi aljabar relasional dasar yaitu, seleksi (*selection*), proyeksi (*projection*), perkalian (*Cartesian product*), penggabungan himpunan (*set union*) dan selisih himpunan (*set difference*) [7] [8]. Namun untuk memperagakan analisa yang lebih mendalam diperlukan notasi lengkap yang ada dalam aljabar relasional dengan rincian adanya teori operasi himpunan, operasi khusus dan operasi tambahan [3] [8].

A. *Projection* $(\pi)$
Digunakan untuk memilih dan menampilkan atribut-atribut dari suatu tabel atau relasi.
Misalkan terdapat $R = (A_1, A_2, \dots, A_n)$. (1)
Dimana $R$ adalah relasi atau tabel yang diperoleh dari entitas dan atau relationship. Sedangkan $A_1, A_2, \dots, A_n$ adalah atribut ke-1, sampai dengan atribut ke-n. Maka bentuk proyeksinya seperti pada notasi persamaan 2 berikut;

$$\pi_{A_i}(R) \quad (2)$$

Keterangan: $A_i$ : merupakan atribut-atribut dan datanya yang akan ditampilkan. Sedangkan R adalah satu atau lebih tabel yang digunakan. Pada sintaks bahasa SQL bentuk proyeksi dari aljabar relasional $\pi_{A_1,A_2,A_3}(R)$ adalah SELECT **A1, A2, A3** FROM R.

B. Selection $(\sigma)$
Digunakan untuk menyaring data berdasarkan suatu kriteria tertentu. Kriteria yang digunakan dapat digabung dengan logika AND $(\wedge)$, OR $(\vee)$dan NOT $(\neg)$. Selain logika juga terdapat operator pembanding yang terdiri dari $=, \neq, <, >, \leq dan \geq$. Bentuk umumnya pada persamaan 3.

$$\sigma_K(R) \quad (3)$$

Keterangan: $K$: merupakan kondisi pembatas atau penyaring data, yang terdiri dari nama atribut, operator pembanding dan nilai batasnya. Sedangkan $R$ adalah satu atau lebih tabel yang digunakan. Dalam bahasa SQL seleksi ditempatkan pada klausa WHERE atau HAVING jika seleksinya setelah terjadi GROUP BY.

C. *Union* $(\cup)$
Digunakan untuk menggabungkan dua atau lebih tabel yang memiliki kolom dengan jumlah yang sama. Misalkan terdapat relasi $R$ dan $S$, maka operasi unión adalah $R \cup S$. Dalam terjemahan bahasa SQL tabel R dan S terbentuk sintaksis sebagai berikut;
Maka hasil dari $R \cup S$ adalah
`SELECT * FROM R UNION SELECT * FROM S`

D. *Intersection* $(\cap)$
Digunakan untuk mencari data yang sama yang terdapat pada dua atau lebih tabel atau relasi, operasi *intersect* mengharuskan tabel-tabel tersebut memiliki jumlah atribut yang sama. Bentuk umum dari operasi ini adalah $R \cap S$. Bahasa SQL dari operasi tersebut adalah `SELECT * FROM R INTERSECT SELECT * FROM S.`

E. *Difference* $(-)$
Digunakan untuk mencari nilai tabel di tabel sebelah kiri yang sama dengan sebelah kanan, atau lebih umum digunakan untuk operasi pengurangan. Bentuk umumnya adalah $R - S$. Bahasa SQL dari operasi tersebut adalah `SELECT * FROM R MINUS SELECT * FROM S.`

F. *Cartesian product* $(\times)$
Digunakan untuk perkalian atau kombinasi data dari dua atau lebih tabel. Operasi ini umumnya digunakan untuk kombinasi penggabungan dari beberapa tabel atau relasi. Bentuk umumya adalah $R \times S$. hasil perkalian dari relasi R dan S adalah gabungan dari atribut-atribut dari kedua relasi dan jumlah datanya terjadi kombinasi perkalian. Misalnya relasi R = (3,4) dan S = (3,3) artinya R memiliki 3 kolom atribut dan 4 baris data dan relasi S memiliki 3 kolom atribut dan 3 baris data, maka hasil dari $R \times S = (6{,}12)$. Sedangkan untuk sintaks bahasa SQL-nya adalah `SELECT * FROM R, S.`

G. *Join* $(\bowtie)$
Digunakan untuk menggabungkan multi tabel dengan kunci relasi secara otomatis. *Join* terdapat 3 jenis *join* yaitu *Inner join* atau *natural join*, *left outer join* dan *right outer join*. Bentuk umum dari *join* yaitu $R \bowtie S$. sedangkan sintaks bahasa SQL-nya adalah `SELECT * FROM R NATURAL JOIN S;.`

H. *Generalized Projections* $(\pi)$
Digunakan untuk perhitungan pada atribut turunan (atribut yang nilainya diperoleh dari proses perhitungan). Secara umum sama dengan proyeksi biasa. Misalnya untuk menampilkan tanggal saat ini $\pi_{Date(NOW())}$ maka bentuk sintaks bahasa SQL-nya adalah `SELECT Date(NOW());.`

I. *Aggregate Functions* ($\mathcal{g}$)
Digunakan untuk mencari nilai data pada suatu atribut atau kumpulan atribut yang berupa, rata-rata, total, jumlah data, nilai minimal dan nilai maksimal. Dengan bentuk umum seperti pada persamaan 4 berikut:

$$\mathcal{g}_{f(A)}(\mathbf{R}) \quad (4)$$

Keterangan: A: Atribut yang akan diagregasikan, dan R: Tabel/relasi
$f$: *fungsi agregasi yang berupa*: SUM: menghitung total jumlah, AVG: menghitung rata-rata, COUNT: menghitung jumlah data, MAX : mencari nilai tertinggi, MIN : mencari nilai terendah. Misalnya ingin menghitung jumlah data dari relas R, maka aljabar relasionalnya adalah $\mathcal{g}_{Count(\ )}(R)$ sedangkan sintaks bahasa SQL-nya adalah `SELECT Count(*) FROM R;.`

J. *Rename* ($\boldsymbol{\rho}$)
Digunakan untuk merubah nama tabel secara permanen dan sementara. Bentuk Aljabar relasi untuk merubah nama tabel memiliki persamaan. Perbedaan perubahan nama tabel secara permanen maupun sementara hanya terletak pada perintah SQL-nya. Bentuk umum dari perintah rename seperti pada formulasi pada persamaan 5.

$$\boldsymbol{\rho}_{NewTable}(OldTable) \quad (5)$$

Sebagai Keterangan diberikan contoh merubah nama tabel secara permanen tabel Coba menjadi Contoh. Maka aljabar relasionalnya adalah $\boldsymbol{\rho}_{Contoh}(Coba)$ dengan sintaks bahasa SQL-nya `RENAME TABLE Coba TO Contoh.`

K. *Assignment* ($\leftarrow$)
Digunakan untuk penugasan perintah pada DDL dan untuk menyingkat penulisan Aljabar Relasi yang panjang.

*L.* *Insert* ($\cup$)
Digunakan untuk menambahkan data pada suatu tabel/relasi. Jika terdapat suatu tabel/relasi r dengan suatu ekspresi E yang berisi data-data yang akan dimasukkan pada tabel maka bentuk umumnya pada persamaan 6.

$$\boldsymbol{r \leftarrow r \cup E} \quad (6)$$

Contoh menambahkan data pada tabel MataKuliah dengan data-data KodeMK='SIS-302', NamaMK='Sistem Basis Data', SKS=3, Teori=3, Praktek=0, maka Aljabar relasinya adalah $MataKuliah \leftarrow MataKuliah \cup \{('SIS-302','Sistem\ Basis\ Data',3,0,0)\}$, dengan sintaks bahasa SQL-nya `INSERT INTO Product VALUES ('SIS-202','Sistem Basis Data',3,3,0).`

M. *Update* ($\boldsymbol{\delta}$)
Menggunakan simbol Delta ($\delta$) yang digunakan untuk merubah data. Jika terdapat tabel/relasi r dengan ekspresi perubahan data E pada suatu atribut A, maka bentuk umum *update* adalah (persamaan 7).

$$\boldsymbol{\delta_A \leftarrow E(r)} \quad (7)$$

Contoh merubah jumlah SKS Praktek menjadi 1 pada mata kuliah dengan kode 'SIS-302', maka aljabar relasinya adalah $\delta_{\text{Praktek}} \leftarrow \text{Praktek} = 1\ (\sigma_{\text{KodeMK}='SIS-302'})^{(\text{MataKuliah})}$ sedangkan sintaks bahasa SQL-nya adalah `UPDATE MataKuliah SET Praktek=1 Where KodeMK='SIS-302'`.

N. *Delete* (–)
Digunakan untuk menghapus suatu data tertentu pada suatu tabel. Jika terdapat suatu tabel atau relasi r dengan suatu kondisi ekspresi E yang akan menentukan data mana yang akan dihapus, maka bentuk umumnya adalah (persamaan 8);

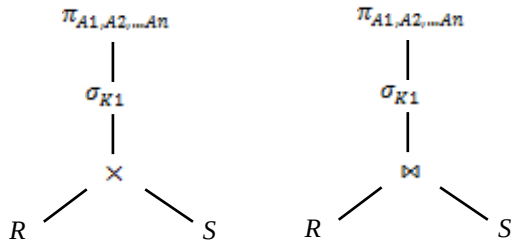
$$\boldsymbol{r \leftarrow r - E} \quad (8)$$

Contoh hapus seluruh data product yang warnanya biru, dengan aljabar Relasi $\boldsymbol{Product \leftarrow Product - \sigma_{color='Blue'}(Product)}$, dengan sintaks bahasa SQL-nya adalah `DELETE FROM Product WHERE Color='Blue'.`

## 5. OPTIMASI BAHASA SQL

Teori optimasi dapat menggunakan pendekatan *heuristic* dan *cost-based*. Metode yang sering digunakan pada pendekatan *heuristic* adalah Greedy Method yang menggunakan *query tree* untuk menganalisanya. Pendekatan *Cost-Based* menggunakan statistik dalam database yang disimpan dalam

*data dictionary* yang berupa meta data [10]. Analisa menggunakan *query tree* dapat menggunakan 2 pendekatan yaitu pendekatan menggunakan *Cartesian product* dan *Join*. Setiap pendekatan dicari efisiensinya dengan pendekatan jumlah baris data dan kolom yang dimuat ke dalam memori komputer. Bentuk optimalisasi menggunakan *query tree* secara sederhana disajikan seperti pada Gambar 2.



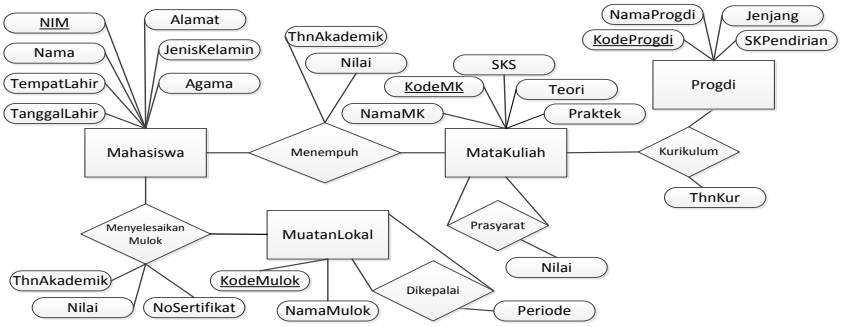

**Gambar 2. Diagram Pohon Optimasi Aljabar Relasi Menggunakan *Query Tree***

Berdasarkan Gambar 2 dapat dijelaskan bahwa diagram pohon pada bagian kanan dan kiri memiliki kesamaan dari setiap *node*-nya, yaitu dimulai dari bagian *root* berupa proyeksi, kemudian seleksi, cabang (*branch*) berupa *Cartesian product* atau *Join*, sedangkan bagian terluar atau daun (*leaf*) berupa tabel atau relasi.

Pada makalah ini optimasi *query* yang dilakukan memiliki langkah-langkah sebagai acuan pengerjaan baik berupa aljabar relasi maupun penulisan sintaksis dalam bahasa SQL yaitu; 1). Menentukan jumlah tabel/relasi yang terlibat, 2). Menentukan cara penggabungan tabel-tabel jika lebih dari sebuah tabel/relasi, 3). Menentukan proyeksi dan atau seleksi jika ada, 4). Menentukan sintaks bahasa SQL, 5). Analisa kecepatan dan banyaknya muatan data dalam memori komputer, 6). Prediksi hasil.

## 6. PERANCANGAN BASIS DATA

Berdasarkan prinsip-prinsip dan urutan langkah dalam menggambar ER-Diagram [1], maka diperoleh entitas **Mahasiswa**, **MataKuliah**, **Progdi**, dan **MuatanLokal**. Hubungan relasional yang terjadi yaitu **Menempuh**, **Kurikulum**, **Prasyarat**, **Dikepalai** dan **MenyelesaikanMulok**. ER-Diagram yang menggambarkan hubungan antar entitas tersebut memiliki hubungan *binary relationships* dan *unary relationship* seperti pada Gambar 3. ER-Diagram untuk basis data mahasiswa layak wisuda.




**Gambar 3. ER-Diagram Untuk Basis Data Mahasiswa Layak Wisuda**

Berdasarkan gambar ER-Diagram pada Gambar 2, dapat diterjemahkan menjadi skema relasi yang menganut model $R = (A_1, A_2, \ldots, A_n)$. Dimana $R$ adalah relasi atau tabel yang diperoleh dari entitas dan atau relationship. Sedangkan $A_1, A_2, \ldots, A_n$ adalah atribut ke-1, sampai dengan atribut ke-n. Sebuah atribut atau gabungan beberapa atribut yang berperan sebagai kunci relasional diberikan garis bawah jika merupakan *primary key* (PK). Hasil transformasi atau penterjemahan ER-Diagram pada Gambar 2 menjadi skema relasional berikut :

**Mahasiswa**=(NIM, Nama, TempatLahir, TanggalLahir, Alamat, JenisKelamin, Agama).
**Menempuh**=(NIM, KodeMK, ThnAkademik, Nilai).
**Matakuliah**=(KodeMK, NamaMK, SKS, Teori, Praktek).
**Kurikulum**=(KodeMK, KodeProgdi, ThnKur).
**Progdi**=(KodeProgdi, NamaProgdi, Jenjang, SKPendirian).

**Prasyarat**=(KodeMK, Prasyarat, Nilai).
**MuatanLokal**=(KodeMulok, NamaMulok).
**KepalaMulok**=(KodeMulok, Dikepalai, Periode).
**MenyelesaikanMulok**=(NIM, KodeMulok, ThnAkademik, Nilai, NoSertifikat).

Berdasarkan skema relasi yang telah terbentuk, langkah selanjutnya adalah menentukan struktur dari setiap skema relasi sehingga terlihat jelas kolom dan tipe datanya. Berikut ini adalah struktur tabel dari skema relasi yang telah terbentuk dalam sintaks bahasa SQL dengan kategori bahasa DDL.

```
CREATE TABLE `mahasiswa`(`NIM` varchar(9) NOT NULL,`Nama` char(50),`TempatLahir` char(50
),`TanggalLahir` date,`Alamat` char(100),`JenisKelamin` char(1),`Agama` char(50),PRIMARY
 KEY (`NIM`))
CREATE TABLE `menempuh`(`NIM` varchar(9),`KodeMK` varchar(7),`ThnAkademik` varchar(20),`
Nilai` char(2),KEY `FK_menempuh` (`NIM`))
CREATE TABLE `matakuliah`(`KodeMK` varchar(7) NOT NULL,`NamaMK` char(50),`SKS` int(11,`T
eori` int(11),`Praktek` int(11),PRIMARY KEY (`KodeMK`))
CREATE TABLE `kurikulum`(`KodeMK` varchar(7) NOT NULL,`KodeProgdi`varchar(2) NOT NULL,`T
hnKur` varchar(4))
CREATE TABLE `progdi`(`KodeProgdi` varchar(2) NOT NULL,`NamaProgdi` char(50),`SKPendiria
n` char(50),`Jenjang` char(5),PRIMARY KEY (`KodeProgdi`))
CREATE TABLE `prasyarat`(`KodeMK` varchar(7),`Prasyarat`varchar(7),`Nilai`char(2))
CREATE TABLE `muatanlokal`(`KodeMulok` varchar(7) NOT NULL,`NamaMulok` char(50),PRIMARY
KEY  (`KodeMulok`))
CREATE TABLE `kepalamulok`(`KodeMulok` varchar(7),`Dikepalai` char(30),`Periode` varchar
(9))
CREATE TABLE `menyelesaikanmulok`(`NIM` varchar(9),`KodeMulok` varchar(7),`ThnAkademik`
varchar(20),`Nilai` char(2),`NoSertifikat` char(30))
```

Berdasarkan DDL struktur data tersebut perlu suatu asumsi bahwa tabel telah berada dalam sebuah DBMS dengan data-data simulasi yang mendekati dengan kenyataan. Jika perguruan tinggi tersebut memiliki sejumlah entitas data maka perlu simulasi jumlah data seperti pada Tabel 1 berikut.

**Tabel 1. Simulasi jumlah data dari entitas dan relasional tabel**

| *No* | *Nama Tabel* | *Jumlah Kolom* | *Jenis Tabel* | *Jumlah Data* |
|---|---|---|---|---|
| 1 | Mahasiswa | 7 | Entitas | 8,000 |
| 2 | MataKuliah | 5 | Entitas | 900 |
| 3 | Progdi | 4 | Entitas | 15 |
| 4 | MuatanLokal | 2 | Entitas | 3 |
| 5 | Menempuh | 4 | Relasi | 7,200,000 |
| 6 | Kurikulum | 3 | Relasi | 60 |
| 7 | Prasyarat | 3 | Relasi | 45 |
| 8 | KepalaMulok | 3 | Relasi | 1 |
| 9 | MenyelesaikanMulok | 5 | Relasi | 24,000 |

## 7. HASIL DAN PEMBAHASAN

Berdasarkan dari Tabel 1, simulasi jumlah data dapat diperoleh data bahwa setiap tahun rata-rata jumlah mahasiswa adalah 2000, jika dibagi dengan jumlah program studi maka rata-rata jumlah mahasiswa perangkatan setiap program studi adalah 133 mahasiswa. Jika jumlah SKS rata-rata dari setiap program studi yang harus ditempuh adalah 150 SKS maka analisisnya adalah jumlah tersebut merupakan jumlah dari setiap kurikulum yang sedang berjalan harus sejenis yang ditempuh oleh setiap mahasiswa. Dengan kata lain, setiap mahasiswa tidak boleh menempuh kelompok kurikulum yang berbeda dengan kurikulum yang telah ditentukan pada angkatannya. Aturan-aturan yang dimunculkan dari permasalahan umum yang ada di lapangan.

### *7.1 Analisa SKS Minimal Setiap Program Studi Terpenuhi*

Untuk mengetahui SKS minimal dari setiap mahasiswa yang berada dalam sebuah program studi maka tabel/relasi yang terlibat adalah Tabel Mahasiswa diambil NIM-nya untuk mengetahui Tahun kurikulum, nama program studi, mata kuliah yang ditempuh. Hasil proyeksi yang diharapkan dalam bentuk skema relasional adalah SKStempuh=(NIM, JumlahSKS).

a.1. Aljabar relasinya dalam operasi *JOIN* tampak sebagai berikut;

$$\pi_{NIM,\mathcal{G}_{sum(SKS)}}(Menempuh \bowtie MataKuliah)$$

Sintaks bahasa SQL-nya dalam *JOIN* adalah:

```
SELECT menempuh.NIM, SUM(matakuliah.SKS FROM publikasi_simetris.matakuliah
INNER JOIN publikasi_simetris.menempuh ON (matakuliah.KodeMK =
menempuh.KodeMK) GROUP BY menempuh.KodeMK;
```

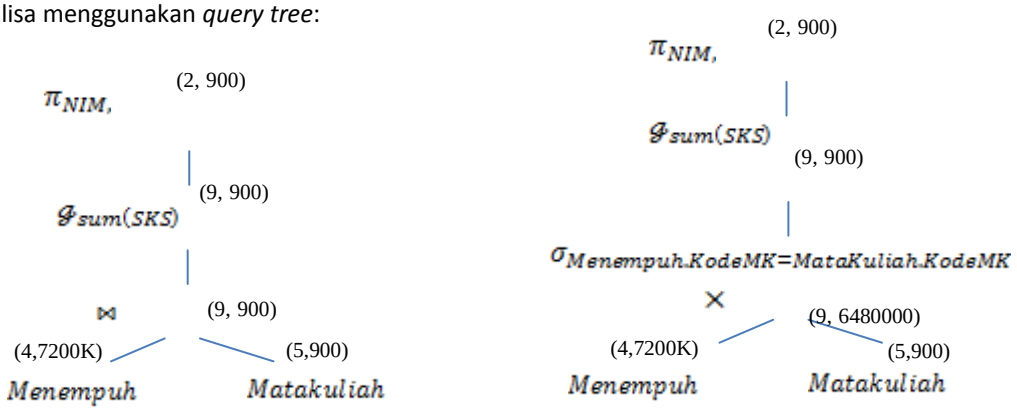
a.2 Aljabar relasinya dalam operasi *Cartesian product* tampak sebagai berikut;

$$\pi_{NIM,\mathcal{G}sum(SKS)}\sigma_{Menempuh.KodeMK=MataKuliah.KodeMK}(Menempuh \times MataKuliah)$$

Sintaks bahasa SQL-nya dalam *Cartesian product*:

```
SELECT menempuh.NIM, SUM(matakuliah.SKS) FROM publikasi_simetris.menempuh
    ,publikasi_simetris.matakuliah
WHERE (menempuh.KodeMK =matakuliah.kodemk)GROUP BY menempuh.KodeMK;
```

Analisa menggunakan *query tree*:

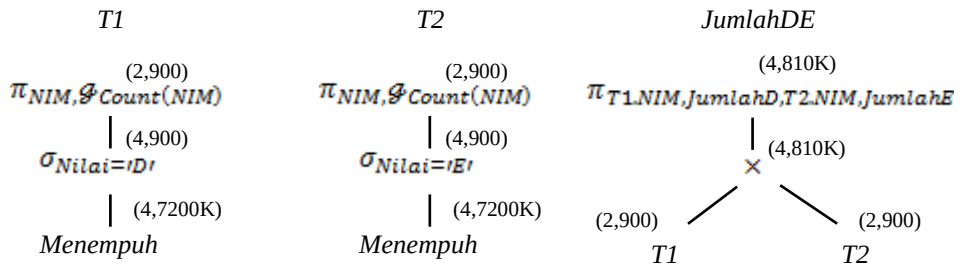



**Gambar 4. Optimasi Untuk Mencari Sintaks Optimal Jumlah SKS**

Berdasarkan Gambar 4. Diperoleh hasil yang hampir mirip antara menggunakan *join* dan *Cartesian product*, namun terlihat bagian kiri (*join*) lebih kecil pemuatan ke dalam memori komputer terutama pada bagian *join*, karena pemuatannya sesuai dengan data yang memiliki kesamaan kunci relasi secara alami.

### *7.2 Batas Untuk Nilai Tertentu*

Batas untuk nilai tertentu misalnya adalah jumlah nilai D tidak lebih dari 4, dan tidak boleh ada nilai E dari seluruh mata kuliah yang pernah ditempuhnya. Analisa untuk kasus ini adalah jika mahasiswa pernah mendapatkan nilai pasti terrekam dalam basis data terutama pada tabel mahasiswa menempuh mata kuliah. Jadi jumlah tabel yang berhubungan adalah satu tabel saja yaitu tabel **Menempuh**=(NIM, KodeMK, ThnAkademik, Nilai) karena seluruh informasi kolom data sudah tercakup didalamnya. Dari tabel **Menempuh** tabel luaran yang diharapkan dalam bentuk skema relasi adalah **JumlahDE**=(NIM, JumlahD, NIM, JumlahE). Penyelesaian batas nilai ini dilakukan dengan pendekatan *cartesian product* Pendekatan *cartesian product* dengan teknik *query-tree* yang telah dioptimasi tampak seperti pada Gambar 5 berikut :




**Gambar 5. *Query-Tree* Menggunakan *Cartesian Product* Untuk Batasan Jumlah Nilai**

Berdasarkan Gambar 5 maka dapat dijelaskan bahwa tabel Menempuh dipecah menjadi 2 tabel yaitu tabl T1 dan T2. Ketika pemuatan kolom dan baris data ke dalam memori komputer T1 dan T2 memiliki perbedaan pada data yang disaring. Data yang disaring pada tabel T1 adalah jumlah nilai D, sedangkan pada tabel T2 adalah nilai E. Hasil dari kedua tabel tersebut masih disimpan sementara dalam memori

komputer dengan jumlah kolom dan baris (kolom, baris) keduanya sama yaitu (2,900). Jumlah 900 data merupakan asumsi jika yang dimuat adalah jumlah seluruh mata kuliah yang ada di perguruan tinggi tersebut. Pemuatan data berupa kolom dan baris digabungkan dengan menggunakan *cartesian product*. Hasil akhir yang dimuat dalam memori komputer yaitu (4,810K), yang artinya sebanyak 4 kolom data dengan maksimal baris data yang dimuat dan ditampilkan adalah 810,000 data. Hal ini prosesnya lebih cepat dibandingkan dengan mengguanakan *Join*, karena data dari setiap tabel disaring/diseleksi terlebih dulu baru dilakukan penggabungan dan diproyeksikan sesuai dengan kolom data yang diinginkan. Hasil pemetaan *Query-tree* kedalam aljabar relasional dan sintaks bahasa SQL tampak sebagai berikut;

Aljabar Relasi

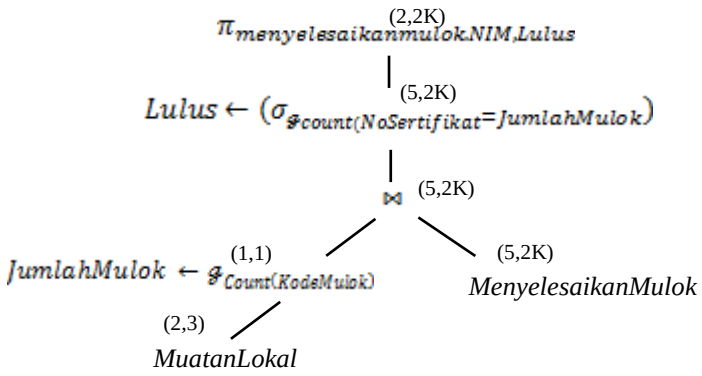
$$JumlahD \leftarrow \sigma_{Nilai='D'}$$
$$JumlahE \leftarrow \sigma_{Nilai='E'}$$
$$T1 \leftarrow \pi_{NIM,\mathcal{G}_{Count(NIM)}}\sigma_{Nilai='D'}(Menempuh)$$
$$T2 \leftarrow \pi_{NIM,\mathcal{G}_{Count(NIM)}}\sigma_{Nilai='E'}(Menempuh)$$
$$JumlahDE = \pi_{T1.NIM,JumlahD,T2.NIM,JumlahE}(T1 \times T2)$$

Sintaks bahasa SQL

```
SELECT T1.NIM, JumlahD, T2.NIM, JumlahE
FROM (SELECT NIM, COUNT(NIM) AS JumlahD FROM menempuh
WHERE (Nilai ='D')
GROUP BY NIM) AS T1,(SELECT NIM, COUNT(NIM) AS JumlahE FROM menempuh
WHERE (Nilai ='E') GROUP BY NIM) AS T2;
```

### *7.3 Pernah Menempuh Kuliah Muatan Lokal*

Mata kuliah muatan lokal memberikan keterampilan khusus bagi mahasiswanya, sehingga mata kuliah muatan lokal ini bersifat wajib. Banyak juga yang menyebutnya sebagai keterampilan wajib. Dikatakan sudah menempuh mata kuliah wajib, jika mahasiswa tersebut telah mendapatkan sertifikast sebagai bukti keikutsertaannya. Tabel yang terlibat adalah tabel **MenyelesaikanMulok**=(NIM, KodeMulok, ThnAkademik, Nilai, NoSertifikat), tabel **Mahasiswa**=(NIM, Nama, TempatLahir, TanggalLahir, Alamat, JenisKelamin, Agama) dan tabel **MuatanLokal**=(KodeMulok, NamaMulok). Tabel luaran yang diharapkan adalah **LulusMuLok**=(NIM, Lulus). Bentuk query-tree yang telah optimal disajikan pada Gambar 6.




**Gambar 6. *Query-Tree* Pernah Menempuh Kuliah Muatan Lokal**

Berdasarkan Gambar 6. Dapat dijelaskan bahwa mula-mula kolom dan baris data yang dimuat dalam memori sebanyak (2,3) dan (5,2000). Diasumsikan bahwa perguruan tinggi hanya memiliki 3 mata kuliah muatan lokal. Setiap tahun meluluskan mahasiswa sebanyak inputannya yaitu 2000 baris data mahasiswa yang telah menyelesaikan kuliah muatan lokal. Sebelum dilakukan *join* dicari dulu jumlah mata kuliah muatan lokalnya, sehingga tidak berpengaruh pada muatan memori ketika *join*. Sampai pada tahap proyeksi jumlah data maksimal yang mungkin dimuat adalah 2 kolom dan 2000 baris data atau dinotasikan dengan (2,2K). Hasil dari optimasi menggunakan Query-Tree dituliskan dalam bentuk aljabar relasional dan sintaks bahasa SQL berikut;

Aljabar Relasi

$$JumlahMulok \leftarrow \mathcal{F}_{Count\,(KodeMulok)}\,(MuatanLokal)$$
$$Lulus \leftarrow (\sigma_{\mathcal{F}_{count\,(NoSertifikat}=JumlahMulok})$$
$$LulusMulok = \pi_{menyelesaikanmulok.NIM,Lulus}(MuatanLokal \bowtie MenyelesaiakanMulok)$$

Sintaks bahasa SQL

```
SELECT menyelesaikanmulok.NIM, IF(COUNT(menyelesaikanmulok.NoSertifikat)=3, "Ya",
"Tidak") AS Lulus
FROM muatanlokal NATURAL JOIN menyelesaikanmulok
GROUP BY menyelesaikanmulok.NIM;
```

## 8. KESIMPULAN

Kesesuaian antara teori dalam aljabar relasi dengan implementasi ke dalam sintaks bahasa SQL terlihat dari hasil luaran eksekusi program. Aljabar relasional sangat membantu adanya kemungkinan penggunaan sintaks bahasa SQL yang berlainan dalam masalah yang sama. Penggunaan sintaks bahasa SQL yang optimal setelah dianalisa menggunakan teknik *query tree* mendapatkan dua hal utama dalam pemrosesan basis data yaitu kecepatan komputasi dan pengoptimalan pemuatan data ke dalam memori komputer. Pemuatan data ke dalam memori komputer lebih optimal jika menggunakan *join* dibandingankan dengan *Cartesian product*. Dimungkinkan gabungan keduanya menjadi lebih cepat jika aturan penggunaan seleksi kemudian proyeksi didahulukan ketimbang penggabungan mengguanakan *Join* atau *Cartesian product* terlebih dulu, hal ini bertujuan untuk mengurangi pemuatan baris dan kolom data yang berlebih. Ditinjau dari kecepatan komputasi untuk operasi himpunan yang melibatkan lebih dari satu tabel, *Cartesian product* dengan didahului proyeksi dan seleksi lebih cepat dibandingkan dengan *Join* untuk kasus yang sama.

## REFERENSI

[1] Chen PPS. English, Chinese and ER-Diagrams. Data & Knowledge Engineering. 1997;: p. 5-16.

[2] Chamberlin DD, Boice RF. SEQUEL: Structured English Query Language. Proceedings of the 1974 ACM SIGFIDET Workshop on Data Description, Access and Control (Association for Computing Machinery). 1974;: p. 249-264.

[3] Date CJ. An Introduction to Database Systems. 8th ed. International: Pearson; 2004.

[4] Codd EF. The Relational Model for Database Management: Version 2. In Serious Flaws in SQL.: Addison-Wesley; 1990. p. 371–389.

[5] ISO/IEC 9075-1:2008 Information technology -- Database languages -- SQL -- Part 1: Framework (SQL/Framework). [Online].; 2008 [cited 2015 October 11. Available from: HYPERLINK "http://www.iso.org/iso/catalogue_detail.htm?csnumber=45498"

[6] Codd EF. A Relational Model of Data for Large Shared Data Banks. Communications of the ACM. 1970; 13(6): p. 377–387.

[7] Fuhr N, Rolleke T. A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems. ACM Transactions on Information Systems. 1997; XV(1): p. 32-66.

[8] Silberschatz A, Korth HF, Sudarshan S. Database System Concept. In. International: McGraw-Hill; 2010. p. 218.

[9] Codd EF. Further Normalization of the Data Base Relational Model. 1971 May 24-25..

[10] Ioannidis YE. Query Optimization. National Science Foundation. Madison: University of Wisconsin, Computer Sciences Department; 1996. Report No.: Grants IRI-9113736 and IRI-9157368.