ANALISA OPTIMASI BAHASA SQL DENGAN *INDEXING* PADA KASUS MAHASISWA LAYAK MENERIMA BEASISWA PADA PERGURUAN TINGGI

Chavid Syukri Fatoni[1)], Dwi Astuti[2)], Kusrini[3)]

[1,2,3)] Magister Teknik Informatika, Universitas AMIKOM Yogyakarta
Jl. Ring Road Utara, Condong Catur, Depok, Sleman, Yogyakarta 55281
Email : fatoni.work@gmail.com[1)], dwi.uwi.finance@gmail.com[2)], kusrini@amikom.ac.id[3)]

Abstrak

Optimalisasi dalam suatu pemrosesan basis data adalah kecepatan dan ketepatan komputasi dimana hal tersebut berkaitan dengan pemuatan data ke dalam memori komputer. Pemrosesan basis data sendiri erat kaitannya dengan penggunaan sintak bahasa SQL. Penggunaan sintak bahasa SQL yang berlainan pada kasus permasalahan basis data yang sama bisa diatasi menggunakan Aljabar relasional untuk pemodelan notasi. Eksperimen yang dijadikan masalah pada penggunaan sintak SQL ini yaitu kasus mahasiswa yang layak menerima beasiswa di suatu perguruan tinggi. Implementasi kasus tersebut pada basis data bisa diperoleh hasil maksimal dengan menggunakan teknik indexing database. Pengunaan teknik indexing database tersebut didapatkan hasil yang menunjukkan bahwa penggunaan Cartesian product lebih efektif dibandingkan menggunakan Join untuk kecepatan dalam pemrosesan sintaks dan menampilkan datanya. Penggabungan Join dan Cartesian product dimungkinkan pemuatan data bisa lebih cepat. Namun, penggabungan tersebut, tidak lebih cepat apabila menggunakan cartesian product dengan dengan didahului seleksi dan proyeksi pada proses pemuatan baris dan kolom dengan jumlah data yang besar. Hal tersebut bisa dibuktikan pada operasi himpunan yang melibatkan lebih dari satu tabel untuk kasus yang sama.

Kata kunci : *Indexing, Aljabar, Cartesian product, Join.*

1. Pendahuluan

Rekapitulasi mahasiswa layak menerima beasiswa pada suatu perguruan tinggi biasanya memiliki beberapa persyaratan. Direktorat Jenderal Pendidikan Tinggi Kementerian Pendidikan Nasional (DIKTI) memberikan pedoman bahwa beasiswa diberikan terdapat dua jenis yaitu PPA (Peningkatan Prestasi Akademik) dan Beasiswa BBM (Bantuan Belajar Mahasiswa).

Persyaratan yang umum digunakan adalah persyaratan akademik dan non akademik. Persyaratan akademik memiliki IPK tertinggi dengan jumlah SKS yang sedikit. Persyaratan non akademik diantaranya data orang tua yang paling tidak mampu, tidak memiliki rumah, dll.

Tujuan Penelitian ini dimaksudkan untuk mengetahui perbandingan antara *Cartesian product* dan *Join* yang diterapkan pada *database* dengan *indexing* dan tanpa *indexing*, sehingga diharapkan pembaca dapat mengetahui perbandingan secara tepat saat melakukan optimasi antara *Cartesian product* dan *Join* sehingga dapat menjadi referensi untuk penelitian selanjutnya.

Basis data merupakan sekumpulan data yang berisi informasi yang disimpan dalam suatu pengorganisasian [1][2].

Relasi Aljabar merupakan keluarga aljabar dengan semantik yang dibangun untuk memodelkan data yang disimpan dalam basis data relasional dan mendefinisikan *query*-nya [3].

SQL merupakan bahasa komputer standar ANSI (*American National Standard Institute*) untuk mengakses dan memanipulasi sistem basis data. Sistem SQL dipergunakan untuk mengambil dan atau mengupdate data dalam basisdata [4].

Cartesian product yang didahului dengan proyeksi dinilai lebih optimal daripada *Join*. Kelemahan dari penelitian ini belum adanya eksekusi dengan jumlah data yang berbeda [6].

Penelitian yang kedua menyimpulkan bahwa metode optimasi untuk web pembelajaran yang paling sesuai adalah M2S *Crossover* karena nilai *fitness* yang lebih tinggi dan waktu akses lebih cepat, namun belum dijelaskan penggunaan *database* dan jumlah data yang diseleksi [8].

Mekanisme pengindeksian mempengaruhi kinerja *database* sistem, Dengan indeks, data bisa cepat ditemukan tanpa melintasi tabel *database* [7].

Penelitian ketiga menjelaskan pengenalan *index* dan T-Tree untuk mengoptimasi *main memory database*, algoritma optimasi mirip dengan T-tree, perbedaanya terletak pada pencarian *precursor* ke *node*. Dengan optimasi algoritma kita dapat langsung mencari *node precursor* dan menghindari banyak langkah berulang, namun belum diperbandingkan terhadap *database* yang tidak *terindex*ing [9].

Penelitian keempat menemukan satu set kebijakan optimal dibawah beban kerja jaringan yang berbeda yang dapat digunakan sebagai model keamanan yang berbeda. Model optimasi otonom mengintegrasikan keamanan dan QoS sumber komputasi yang ada, namun belum dipaparkan karakteristik *database*-nya[10].

Penelitian ini akan membahas perbandingan antara *Cartesian product* dengan *Join* yang di terapkan ke dalam dua *database*, yang pertama *database* dengan *indexing* yang kedua *database* tanpa *indexing* dengan jumlah *database* yang berbeda untuk mengetahui

kecepatan akses data.antara Cartesian product dan join yang diterapkan pada *database* dengan *indexing* dan *noindexing*.

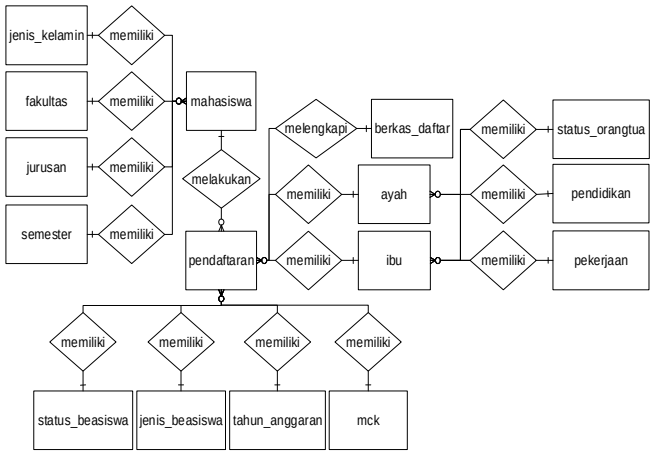
1.1 Metodologi Penelitian

Langkah-langkah yang dilakukan pada penelitian ini meliputi:

1. Analisa objek yang digunakan adalah suatu perguruan tinggi yang memberikan beasiswa PPA maupun BBM kepada mahasiswa yang memenuhi kriteria persyaratan yang telah ditentukan.
2. Pengkajian data dilakukan analisa awal tentang kecenderungan adanya hubungan antar data, pada saat implementasi menggunakan data simulasi.
3. Pengumpulan teori formula Aljabar Relasi menggunakan perumusan aljabar relasi yang diterjemahkan dalam bahasa SQL.
4. Perancangan basis data, tahap ini merupakan tahap yang paling penting dalam pemodelan data relasional. Alat perancangan yang digunakan adalah *ER-Diagram*.
5. Implementasi basis data menggunakan MySQL karena bersifat bebas pakai non komersial dengan lisensi publik. Terdapat 2 basis data yaitu yang di *indexing* dan yang tanpa *indexing*.
6. Analisa Optimalisasi Proses, setiap hasil dari pemrosesan *query* dianalisa lama proses eksekusi. Data hasil analisa direkam kemudian dianalisa sebagai dasar pengambilan simpulan.

1.2 Entity Relational *Database*

Pemodelan data relasional *database* akan disajikan pada Gambar 1.



Gambar 1. *ER-Diagram Beasiswa*

1.3 Notasi Aljabar

Notasi aljabar didefinisikan memiliki 5 (lima) notasi aljabar relasional dasar antara lain seleksi (*selection*), proyeksi (*projection*), perkalian (*Cartesian product*), penggabungan himpunan (*set union*) dan selisih himpunan (*set difference*) [5] [2].

Proses analisa yang lebih mendalam diperlukan notasi lengkap yang ada dalam aljabar relasional dengan rincian adanya teori operasi himpunan, operasi khusus dan operasi tambahan [7] [2].

1. *Projection* (π)
   Notasi *Projection* berfungsi untuk memilih dan menampilkan atribut-atribut pada suatu tabel atau relasi. Misal terdapat $R = (F_1, F_{2,\ldots} F_n)$. (1) Dimana R adalah relasi atau tabel yang diperoleh dari entitas dan atau relationship. Sedangkan (F_1, $F_{2,\ldots}$ F_n) adalah atribut ke-1 sampai dengan atribut ke-n. Maka notasi proyeksinya seperti pada notasi persamaan (1) berikut :

   $$\pi_{F_i}(R) \quad \text{.....(1)}$$

   Keterangan : F_i merupakan atribut-atribut dan datanya yang akan ditampilkan. Sedangkan R adalah satu atau lebih tabel yang digunakan. Pada sintaks bahasa SQL notasi proyeksi dari aljabar relasional :

   $$\pi_{F_1, F_2, F_3, F_4}(R)$$

   adalah `SELECT F1, F2, F3, F4 FROM R.`
2. *Selection* (σ)
   Notasi *Selection* berfungsi 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 =, $\neq$, <, >, $\leq$, $\geq$. Notasi aljabarnyta pada persamaan (2).

   $$\sigma_C(R) \quad \text{.....(2)}$$

   Keterangan: C 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.
3. *Union* ($\cup$)
   Notasi *Union* berfungsi untuk menggabungkan dua atau lebih tabel yang memiliki kolom dengan jumlah yang sama. Misalkan terdapat relasi A dan B, maka operasi union adalah $A \cup B$. Dalam terjemahan bahasa SQL tabel A dan B terbentuk sintaksis sebagai berikut, maka hasil dari $A \cup B$ adalah :
   ```
   SELECT * FROM A UNION SELECT * FROM B
   ```
4. *Intersection* ($\cap$)
   Notasi *Intersection* berfungsi 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. Notasi aljabar dari operasi ini adalah $A \cap B$. SQL dari operasi tersebut adalah
   ```
   SELECT * FROM A INTERSECT SELECT * FROM B.
   ```
5. *Difference* ($-$)
   Notasi *Difference* berfungsi untuk mencari nilai tabel di tabel sebelah kiri yang sama dengan sebelah kanan, atau lebih umum digunakan untuk operasi pengurangan. Notasi aljabarnya adalah $A - B$
   Bahasa SQL dari operasi tersebut adalah
   ```
   SELECT * FROM A MINUS SELECT * FROM B.
   ```

6. *Cartesian product* (×)
   Notasi *Cartesian product* berfungsi untuk perkalian atau kombinasi data dari dua atau lebih tabel. Operasi ini umumnya digunakan untuk kombinasi penggabungan dari beberapa tabel atau relasi. Notasi aljabarnya adalah : $A \times B$. Hasil perkalian dari relasi A dan B adalah gabungan dari atribut-atribut dari kedua relasi dan jumlah datanya terjadi kombinasi perkalian. Misalnya relasi A = (1,2) dan B = (1,4) artinya A memiliki 1 kolom atribut dan 2 baris data dan relasi B memiliki 1 kolom atribut dan 4 baris data, maka hasil dari $A \times B$ = (2,8). Sedangkan untuk sintaks bahasa SQL-nya adalah `SELECT * FROM A, B.`
7. *Join* (⋈)
   Notasi *Join* berfungsi 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*. Notasi aljabar dari *join* yaitu $A \bowtie B$. Sedangkan sintaks bahasa SQL-nya adalah `SELECT * FROM A NATURAL JOIN B.`
8. *Generalized Projections* (π)
   Notasi *Generalized Projections* berfungsi 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()).`
9. *Aggregate Functions* (g)
   Notasi *Aggregate Functions* berfungsi untuk mencari nilai data pada suatu atribut atau kumpulan atribut yang berupa, ratarata, total, jumlah data, nilai minimal dan nilai maksimal. Dengan notasi aljabar seperti pada persamaan (3) berikut:

   $$g_{f(A)}(R) \quad \text{.....(3)}$$

   f : fungsi agregasi yang berupa : SUM : menghitung jumlah total, AVG: menghitung ratarata, COUNT: menghitung jumlah data, MAX : mencari nilai tertinggi, MIN : mencari nilai terendah. Misalnya ingin menghitung jumlah data dari relas R, maka aljabar relasionalnya adalah

   $g_{Count()}(R)$ sedangkan sintaks bahasa SQL-nya adalah `SELECT Count(*) FROM R.`
10. *Rename* (ρ)
    Notasi *Rename* berfungsi untuk mengubah 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. Notasi aljabar dari perintah rename seperti pada formulasi pada persamaan (4).

    $$\rho_{NewTable}(OldTable) \quad \text{.....(4)}$$

    Sebagai Keterangan diberikan contoh merubah nama tabel secara permanen tabel Coba menjadi Contoh. Maka aljabar relasionalnya adalah

    $\rho_{Sample}(Testing)$ dengan sintaks bahasa SQL-nya `RENAME TABLE Sample TO Testing.`
11. *Assignment* (←)
    Notasi *Assignment* berfungsi untuk penugasan perintah pada DDL dan untuk menyingkat penulisan Aljabar Relasi yang panjang.
12. *Insert* (∪)
    Notasi *Insert* berfungsi 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 notasi aljabarnya pada persamaan (5).

    $$r \leftarrow r \cup E \quad \text{.....(5)}$$

    Contoh menambahkan data pada tabel Mahasiswa dengan data-data yaitu kolom NIM=”17.52.0927”, nama_lengkap=”Chavid Syukri Fatoni”, tempat_lahir=”Surakarta”, tanggal_lahir=1991-10-09, alamat=”Jl. Ir. Juanda No.30 Purwodiningratan Jebres Surakarta”, maka Aljabar relasinya adalah
    Mahasiswa ← *Mahasiswa* ∪ {(“17.52.0927”, ”Chavid Syukri Fatoni”, ”Surakarta”, 1991-10-09, ”Jl. Ir. Juanda No.30 Purwodiningratan Jebres Surakarta”)} dengan sintaks bahasa SQL-nya
    `INSERT INTO Mahasiswa VALUES (“17.52.0927”, ”Chavid Syukri Fatoni”, ”Surakarta”, 1991-10-09, ”Jl. Ir. Juanda No.30 Purwodiningratan Jebres Surakarta”).`
13. *Update* (δ)
    Notasi *Update* yang menggunakan simbol Delta (δ) yang berfungsi untuk merubah data. Jika terdapat tabel/relasi r dengan ekspresi perubahan data W pada suatu atribut Q, maka notasi aljabar *update* adalah persamaan (6).

    $$\delta_{Q \leftarrow W(r)} \quad \text{.....(6)}$$

    Contoh merubah tanggal lahir menjadi 2017-01-24 pada mahasiswa dengan NIM “17.52.0927”, maka aljabar relasinya adalah

    $\delta_{tanggal\_lahir \leftarrow tanggal\_lahir=2017\text{-}01\text{-}24}$

    $(\sigma_{NIM=”17.52.0927”})^{(Mahasiswa)}$
    sedangkan sintaks bahasa SQL-nya adalah `UPDATE Mahasiswa SET tanggal_lahir=2017-01-24 WHERE NIM=”17.52.0927”.`
14. *Delete* (−)
    Notasi *Delete* berfungsi untuk menghapus suatu data tertentu pada suatu tabel. Jika terdapat suatu tabel atau relasi r dengan suatu kondisi ekspresi W yang akan menentukan data mana yang akan dihapus, maka notasi aljabarnya seperti persamaan (7).

    $$r \leftarrow r - W \quad \text{.....(7)}$$

    Contoh menghapus semua data pendaftaran yang tanggal pendaftarannya sesudah tanggal 2017-12-30, dengan aljabar relasi *pendaftaran* ← *pendaftaran* − $\sigma_{tanggal\_daftar < 2017\text{-}12\text{-}30}^{(pendaftaran)}$, dengan

sintaks bahasa SQL-nya adalah `DELETE FROM pendaftaran WHERE tanggal_daftar<2017-12-30`.

1.4 Aturan Pemberian Beasiswa

1. PPA (Peningkatan Prestasi Akademik)
   a. Mahasiswa yang mempunyai IPK paling tinggi.
   b. Mahasiswa yang mempunyai SKS paling banyak (jumlah semester paling sedikit).
   c. Mahasiswa yang memiliki prestasi di kegiatan ko/ekstra kurikuler (olahraga, teknologi, seni/budaya tingkat internasional/ dunia, Regional/ Asia/ Asean dan Nasional).
   d. Mahasiswa yang (orang tuanya) paling tidak mampu.
2. BBM (Bantuan Belajar Mahasiswa)
   a. Mahasiswa yang (orang tuanya) paling tidak mampu.
   b. Mahasiswa yang memiliki prestasi di kegiatan ko/ekstra kurikuler (olahraga, teknologi, seni/budaya tingkat internasional/ dunia, Regional/ Asia/ Asean dan Nasional).
   c. Mahasiswa yang mempunyai IPK paling tinggi.
   d. Mahasiswa yang mempunyai SKS paling banyak (jumlah semester paling sedikit).

1.5 Data Beasiswa

Jumlah data yang dipergunakan untuk menganalisa *Cartesian product* dan *Join* akan disajikan pada Tabel 1.

Tabel 1. *Data Beasiswa*

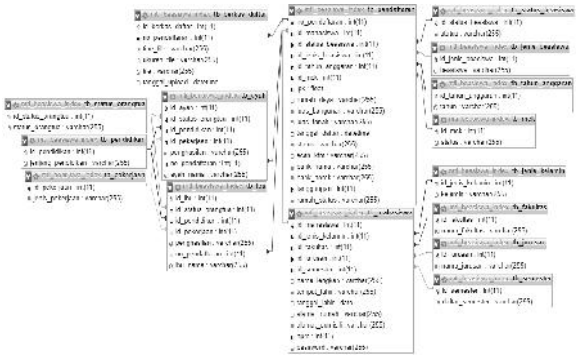
No	Nama Tabel	Jml Kolom	Jml Data
1	Ayah	7	35.840
2	Berkas Daftar	6	25.494
3	Fakultas	2	10
4	Ibu	7	35.840
5	Jenis Beasiswa	2	4
6	Jenis Kelamin	2	2
7	Jurusan	2	64
8	Mahasiswa	12	35.840
9	Mck	2	4
10	Pekerjaan	2	4
11	Pendaftaran	17	25.494
12	Pendidikan	2	5
13	Semester	2	2
14	Status beasiswa	2	2
15	Status orang tua	2	2
16	Tahun anggaran	2	4

1.6 Relasi *Database*

Berdasarkan data di atas, akan disusun dua *database* sebagai alat penelitian yaitu *database* tanpa *indexing* pada Gambar 2 dan dengan *indexing* pada Gambar 3.

Gambar 2. *Relasi tabel noindexing*

Gambar di atas menerangkan terdapat 16 tabel relasi tabel *database* yang tidak di *indexing* setiap kolom pada masing-masing tabel dengan jumlah data yang sama.



Gambar 3. *Relasi tabel dengan indexing*

Gambar di atas menerangkan relasi antara 16 tabel dengan *database* yang telah di *indexing*.

2. Pembahasan

Berdasarkan dari Tabel 1, data beasiswa dapat diketahui bahwa data mahasiswa setiap tahun rata-rata jumlah mahasiswa tiap angkatan mencapai kurang lebih 8.000 data, jika ditotal semua fakultas maka akan mencapai kurang lebih 35.000, selain itu untuk pemberian beasiswa kepada mahasiswa yang sumber dana dari DIKTI hanya dibagi menjadi 2 jenis beasiswa yaitu PPA (Peningkatan Prestasi Akademik) dan Beasiswa BBM (Bantuan Belajar Mahasiswa) yang secara terpusat melalui Bagian Kemahasiswaan, jika perguruan tinggi sistem pendidikannya semakin meningkat maka semakin banyak pula mahasiswa yang mendapatkan prestasi dan semakin banyak mahasiswa yang ingin mandapatkan beasiswa.

Melalui proses pemberian beasiswa mahasiswa diwajibkan melengkapi persyaratan administrasi untuk pendaftaran setiap periode semester ganjil, jika dilihat dari data sekitar 75% atau kurang lebih 25.000 mahasiswa dari total mahasiswa aktif ikut serta mendaftar beasiswa, maka dari itu dalam proses penentuan pemberian beasiswa ini memerlukan aturan-aturan atau *rule* yang digunakan untuk mendapatkan data-data mahasiswa yang layak untuk menerima beasiswa.

2.1 Analisa Mahasiswa Menerima Beasiswa PPA

Analisa ini berdasarkan aturan-aturan untuk menentukan penerima beasiswa PPA, antara lain sebagai berikut :

1. Mahasiswa yang telah mendaftar Beasiswa PPA
2. Mahasiswa memiliki IPK paling tinggi
3. Mahasiswa sudah melengkapi upload berkas
4. Mahasiswa yang total penghasilan kedua orangtuanya paling rendah
5. Mahasiswa yang tidak memiliki memiliki MCK pribadi

6. Mahasiswa yang memiliki rumah daya listrik ≤ 450W
7. Mahasiswa yang memiliki luas tanah & luas rumah ≤ 50m²
8. Mahasiswa yang orangtuanya memiliki tanggungan saudara paling banyak
9. Mahasiswa yang tidak memiliki rumah tempat tinggal kepemilikan sendiri

2.1.1 Analisa dengan operasi *Cartesian product*

Proyeksi Aljabar dalam operasi *Cartesian product* (×) sebagai berikut :

π *NO.DAFTAR, NAMA LENGKAP, IPK, BERKAS, PENGHASILAN ORANGTUA, MCK PRIBADI, STATUS RUMAH, LUAS TANAH ≤ 50m², LUAS BANGUNAN ≤ 50m², RUMAH DAYA LISTRIK ≤ 450W, JUMLAH TANGGUNGAN, BEASISWA PPA*

(*tb_pendaftaran* × *tb_mahasiswa* × tb_berkas_daftar × *tb_ayah* × *tb_ibu* × *tb_mck* × *tb_jenis_beasiswa*)

Penerapan sintaks bahasa SQL-nya sebagai berikut :

```
SELECT tb_pendaftaran.no_pendaftaran AS `NO. DAFTAR`,
tb_mahasiswa.nama_lengkap AS `NAMA LENGKAP`,
tb_pendaftaran.ipk AS `IPK`,
IF(tb_berkas_daftar.id_berkas_daftar!="", "Lengkap", "Belum
Lengkap") AS `BERKAS`,
(`tb_ayah`.penghasilan+`tb_ibu`.penghasilan) AS `PENGHASILAN
ORANGTUA`, IF(tb_mck.status LIKE "%PRIBADI%", "TIDAK", "YA")
AS `MCK PRIBADI`, tb_pendaftaran.rumah_status AS `STATUS
RUMAH`, IF(REPLACE(tb_pendaftaran.luas_tanah, "m", "")<=50,
"YA", "TIDAK") AS `LUAS TANAH <= 50M`,
IF(REPLACE(tb_pendaftaran.luas_bangunan, "m", "")<=50, "YA",
"TIDAK") AS `LUAS BANGUNAN <= 50M`,
IF(REPLACE(tb_pendaftaran.rumah_daya, "w", "")<=450, "YA",
"TIDAK") AS `RUMAH DAYA LISTRIK <= 450W`,
tb_pendaftaran.tanggungan AS `JUMLAH TANGGUNGAN`,
IF(tb_jenis_beasiswa.beasiswa LIKE "%PPA%", "YA", "TIDAK") AS
`BEASISWA PPA` FROM `tb_pendaftaran` tb_pendaftaran,
`tb_mahasiswa` tb_mahasiswa,`tb_berkas_daftar`
tb_berkas_daftar, `tb_ayah` tb_ayah, `tb_ibu` tb_ibu,
`tb_mck` tb_mck, `tb_jenis_beasiswa` tb_jenis_beasiswa WHERE
`tb_pendaftaran`.id_mahasiswa = `tb_mahasiswa`.id_mahasiswa
AND `tb_berkas_daftar`.no_pendaftaran =
`tb_pendaftaran`.no_pendaftaran AND `tb_ayah`.no_pendaftaran
= `tb_pendaftaran`.no_pendaftaran AND `tb_ibu`.no_pendaftaran
= `tb_pendaftaran`.no_pendaftaran AND `tb_pendaftaran`.id_mck
= `tb_mck`.id_mck AND `tb_pendaftaran`.id_jenis_beasiswa =
`tb_jenis_beasiswa`.id_jenis_beasiswa ORDER BY `IPK` DESC,
`PENGHASILAN ORANGTUA` ASC, `MCK PRIBADI` ASC,  `STATUS
RUMAH` ASC, `LUAS TANAH <= 50M` ASC,  `LUAS BANGUNAN <= 50M`
ASC,  `RUMAH DAYA LISTRIK <= 450W` ASC,  `JUMLAH TANGGUNGAN`
ASC, `BEASISWA PPA` DESC;
```

2.1.2 Analisa dengan operasi *Join*

Proyeksi Aljabar dalam operasi *Join* (⋈) sebagai berikut:

π *NO.DAFTAR, NAMA LENGKAP, IPK, BERKAS, PENGHASILAN ORANGTUA, MCK PRIBADI, STATUS RUMAH, LUAS TANAH ≤ 50m², LUAS BANGUNAN ≤ 50m², RUMAH DAYA LISTRIK ≤ 450W, JUMLAH TANGGUNGAN, BEASISWA PPA*

(*tb_pendaftaran* ⋈ *tb_mahasiswa* ⋈ tb_berkas_daftar ⋈ *tb_ayah* ⋈ *tb_ibu* ⋈ *tb_mck* ⋈ *tb_jenis_beasiswa*)

Penerapan sintaks bahasa SQL-nya sebagai berikut :

```
SELECT tb_pendaftaran.no_pendaftaran AS `NO. DAFTAR`,
tb_mahasiswa.nama_lengkap AS `NAMA LENGKAP`,
tb_pendaftaran.ipk AS `IPK`,
IF(tb_berkas_daftar.id_berkas_daftar!="", "Lengkap", "Belum
Lengkap") AS `BERKAS AS
(`tb_ayah`.penghasilan+`tb_ibu`.penghasilan) AS `PENGHASILAN
ORANGTUA`, IF(tb_mck.status LIKE "%PRIBADI%", "TIDAK", "YA")
AS `MCK PRIBADI`, tb_pendaftaran.rumah_status AS `STATUS
RUMAH`, IF(REPLACE(tb_pendaftaran.luas_tanah, "m", "")<=50,
"YA", "TIDAK") AS `LUAS TANAH <= 50M`,
IF(REPLACE(tb_pendaftaran.luas_bangunan, "m", "")<=50, "YA",
"TIDAK") AS `LUAS BANGUNAN <= 50M`,
IF(REPLACE(tb_pendaftaran.rumah_daya, "w", "")<=450, "YA",
"TIDAK") AS `RUMAH DAYA LISTRIK <= 450W`,
tb_pendaftaran.tanggungan AS `JUMLAH TANGGUNGAN`,
IF(tb_jenis_beasiswa.beasiswa LIKE "%PPA%", "YA", "TIDAK") AS
`BEASISWA PPA` FROM `tb_pendaftaran` tb_pendaftaran INNER
JOIN `tb_mahasiswa` tb_mahasiswa ON
`tb_pendaftaran`.id_mahasiswa = `tb_mahasiswa`.id_mahasiswa
INNER JOIN `tb_berkas_daftar` tb_berkas_daftar ON
`tb_pendaftaran`.no_pendaftaran =
`tb_berkas_daftar`.no_pendaftaran INNER JOIN `tb_ayah`
tb_ayah ON `tb_pendaftaran`.no_pendaftaran =
`tb_ayah`.no_pendaftaran INNER JOIN `tb_ibu` tb_ibu ON
`tb_pendaftaran`.no_pendaftaran = `tb_ibu`.no_pendaftaran
INNER JOIN `tb_mck` tb_mck ON `tb_pendaftaran`.id_mck =
`tb_mck`.id_mck INNER JOIN `tb_jenis_beasiswa`
tb_jenis_beasiswa ON `tb_pendaftaran`.id_jenis_beasiswa =
`tb_jenis_beasiswa`.id_jenis_beasiswa ORDER BY `IPK` DESC,
`PENGHASILAN ORANGTUA` ASC, `MCK PRIBADI` ASC, `STATUS RUMAH`
ASC, `LUAS TANAH <= 50M` ASC, `LUAS BANGUNAN <= 50M` ASC,
`RUMAH DAYA LISTRIK <= 450W` ASC, `JUMLAH TANGGUNGAN` ASC,
`BEASISWA PPA` DESC;
```

2.2 Analisa Mahasiswa Menerima Beasiswa BBM

Analisa ini berdasarkan aturan-aturan untuk menentukan penerima beasiswa BBM, antara lain sebagai berikut :

1. Mahasiswa yang telah mendaftar Beasiswa BBM
2. Mahasiswa sudah melengkapi upload berkas
3. Mahasiswa yang total penghasilan kedua orangtuanya paling rendah
4. Mahasiswa yang tidak memiliki memiliki MCK pribadi
5. Mahasiswa yang memiliki rumah daya listrik ≤ 450W
6. Mahasiswa yang memiliki luas tanah & luas rumah ≤ 50m²
7. Mahasiswa yang orangtuanya memiliki tanggungan saudara paling banyak
8. Mahasiswa yang tidak memiliki rumah tempat tinggal kepemilikan sendiri
9. Mahasiswa memiliki IPK paling tinggi

2.2.1 Analisa dengan operasi *Cartesian product*

Proyeksi Aljabar dalam operasi *Cartesian product* (×) sebagai berikut :

π *NO.DAFTAR, NAMA LENGKAP, IPK, BERKAS, PENGHASILAN ORANGTUA, MCK PRIBADI, STATUS RUMAH, LUAS TANAH ≤ 50m², LUAS BANGUNAN ≤ 50m², RUMAH DAYA LISTRIK ≤ 450W, JUMLAH TANGGUNGAN, BEASISWA PPA*

(*tb_pendaftaran* × *tb_mahasiswa* × tb_berkas_daftar × *tb_ayah* × *tb_ibu* × *tb_mck* × *tb_jenis_beasiswa*)

Penerapan sintaks bahasa SQL-nya sebagai berikut :

```
SELECT tb_pendaftaran.no_pendaftaran AS `NO. DAFTAR`,
tb_mahasiswa.nama_lengkap AS `NAMA LENGKAP`,
IF(tb_berkas_daftar.id_berkas_daftar!="", "Lengkap", "Belum
Lengkap") AS `BERKAS`,
(`tb_ayah`.penghasilan+`tb_ibu`.penghasilan) AS `PENGHASILAN
ORANGTUA`, IF(tb_mck.status LIKE "%PRIBADI%", "TIDAK", "YA")
AS `MCK PRIBADI`, tb_pendaftaran.rumah_status AS `STATUS
RUMAH`, IF(REPLACE(tb_pendaftaran.luas_tanah, "m", "")<=50,
"YA", "TIDAK") AS `LUAS TANAH <= 50M`,
IF(REPLACE(tb_pendaftaran.luas_bangunan, "m", "")<=50, "YA",
"TIDAK") AS `LUAS BANGUNAN <= 50M`,
IF(REPLACE(tb_pendaftaran.rumah_daya, "w", "")<=450, "YA",
"TIDAK") AS `RUMAH DAYA LISTRIK <= 450W`,
tb_pendaftaran.tanggungan AS `JUMLAH TANGGUNGAN`,
tb_pendaftaran.ipk AS `IPK`, IF(tb_jenis_beasiswa.beasiswa
LIKE "%BBM%", "YA", "TIDAK") AS `BEASISWA BBM` FROM
`tb_pendaftaran` tb_pendaftaran, `tb_mahasiswa` tb_mahasiswa,
`tb_berkas_daftar` tb_berkas_daftar, `tb_ayah` tb_ayah,
`tb_ibu` tb_ibu, `tb_mck` tb_mck, `tb_jenis_beasiswa`
tb_jenis_beasiswa WHERE `tb_pendaftaran`.id_mahasiswa =
`tb_mahasiswa`.id_mahasiswa AND
`tb_berkas_daftar`.no_pendaftaran =
`tb_pendaftaran`.no_pendaftaran AND `tb_ayah`.no_pendaftaran
= `tb_pendaftaran`.no_pendaftaran AND `tb_ibu`.no_pendaftaran
= `tb_pendaftaran`.no_pendaftaran AND `tb_pendaftaran`.id_mck
= `tb_mck`.id_mck AND `tb_pendaftaran`.id_jenis_beasiswa =
`tb_jenis_beasiswa`.id_jenis_beasiswa ORDER BY `PENGHASILAN
ORANGTUA` ASC, `MCK PRIBADI` ASC, `STATUS RUMAH` ASC, `LUAS
TANAH <= 50M` ASC, `LUAS BANGUNAN <= 50M` ASC, `RUMAH DAYA
LISTRIK <= 450W` ASC, `JUMLAH TANGGUNGAN` ASC, `IPK` DESC,
`BEASISWA BBM` DESC;
```

2.2.2 Analisa dengan operasi *Join*

Proyeksi Aljabar dalam operasi *Join* (⋈) sebagai berikut:

$$\pi_{NO.DAFTAR,\ NAMA\ LENGKAP,\ IPK,\ BERKAS,\ PENGHASILAN\ ORANGTUA,\ MCK\ PRIBADI,\ STATUS\ RUMAH,\ LUAS\ TANAH \le 50m^2,\ LUAS\ BANGUNAN \le 50m^2,\ RUMAH\ DAYA\ LISTRIK \le 450W,\ JUMLAH\ TANGGUNGAN,\ BEASISWA\ PPA}$$

$$(tb\_pendaftaran \bowtie tb\_mahasiswa \bowtie tb\_berkas\_daftar \bowtie tb\_ayah \bowtie tb\_ibu \bowtie tb\_mck \bowtie tb\_jenis\_beasiswa)$$

Penerapan sintaks bahasa SQL-nya sebagai berikut :

```
SELECT tb_pendaftaran.no_pendaftaran AS `NO. DAFTAR`,
tb_mahasiswa.nama_lengkap AS `NAMA LENGKAP`,
IF(tb_berkas_daftar.id_berkas_daftar!="", "Lengkap", "Belum
Lengkap") AS `BERKAS`,
(`tb_ayah`.penghasilan+`tb_ibu`.penghasilan) AS `PENGHASILAN
ORANGTUA`, IF(tb_mck.status LIKE "%PRIBADI%", "TIDAK", "YA")
AS `MCK PRIBADI`, tb_pendaftaran.rumah_status AS `STATUS
RUMAH`, IF(REPLACE(tb_pendaftaran.luas_tanah, "m", "")<=50,
"YA", "TIDAK") AS `LUAS TANAH <= 50M`,
IF(REPLACE(tb_pendaftaran.luas_bangunan, "m", "")<=50, "YA",
"TIDAK") AS `LUAS BANGUNAN <= 50M`,
IF(REPLACE(tb_pendaftaran.rumah_daya, "w", "")<=450, "YA",
"TIDAK") AS `RUMAH DAYA LISTRIK <= 450W`,
tb_pendaftaran.tanggungan AS `JUMLAH TANGGUNGAN`,
tb_pendaftaran.ipk AS `IPK`, IF(tb_jenis_beasiswa.beasiswa
LIKE "%BBM%", "YA", "TIDAK") AS `BEASISWA BBM` FROM
`tb_pendaftaran` tb_pendaftaran INNER JOIN `tb_mahasiswa`
tb_mahasiswa ON `tb_pendaftaran`.id_mahasiswa =
`tb_mahasiswa`.id_mahasiswa INNER JOIN `tb_berkas_daftar`
tb_berkas_daftar ON `tb_pendaftaran`.no_pendaftaran =
`tb_berkas_daftar`.no_pendaftaran INNER JOIN `tb_ayah`
tb_ayah ON `tb_pendaftaran`.no_pendaftaran =
`tb_ayah`.no_pendaftaran INNER JOIN `tb_ibu` tb_ibu ON
`tb_pendaftaran`.no_pendaftaran = `tb_ibu`.no_pendaftaran
INNER JOIN `tb_mck` tb_mck ON `tb_pendaftaran`.id_mck =
`tb_mck`.id_mck INNER JOIN `tb_jenis_beasiswa`
tb_jenis_beasiswa ON `tb_pendaftaran`.id_jenis_beasiswa =
`tb_jenis_beasiswa`.id_jenis_beasisw ORDER BY `PENGHASILAN
ORANGTUA` ASC, `MCK PRIBADI` ASC, `STATUS RUMAH` ASC, `LUAS
TANAH <= 50M` ASC, `LUAS BANGUNAN <= 50M` ASC, `RUMAH DAYA
LISTRIK <= 450W` ASC, `JUMLAH TANGGUNGAN` ASC, `IPK` DESC,
`BEASISWA BBM` DESC;
```

Menguraikan hasil analisis antara *Cartesian product* dan *Join* dengan *database* yang di *indexing* dan tanpa di *indexing*, dengan catatan waktu *query* dalam satuan *seconds* dan pengambilan data yang berbeda mulai dari 1000, 5.000 dan 10.000 data *query* berdasarkan uji coba peneliti untuk mendapatkan hasil perbedaan catatan waktu saat *query* data, maka dapat diperoleh data yang disajikan pada Tabel 2.

Tabel 2. *Hasil Analisa*

Kriteria	*Database Indexing*			*Database Noindexing*		
	1.000	5.000	10.000	1.000	5.000	10.000
PPA						
CP	0.0029	0.0713	0.1203	0,0424	59.2394	98.7897
Join	0,0773	0.1023	0.1650	0,0430	72.8736	112.6559
BBM						
CP	0,0024	0.3498	0.7200	0,0257	35.9287	96.9062
Join	0,0026	0.2647	0.6657	0,0224	47.3282	100.8839

3. Kesimpulan

Berdasarkan uraian di atas dapat disimpulkan bahwa :

1. Optimasi menggunakan *Cartesian product* & *Join* diperoleh data bahwa eksekusi *Cartesian product* lebih cepat dibandingkan *Join (Inner Join)* dalam kasus ini dalam mengambil data Mahasiswa yang berhak mendapatkan beasiswa yang memiliki beberapa aturan yang dimodelkan dalam bentuk notasi aljabar.
2. Optimasi menggunakan *Indexing* MySQL untuk *field* yang dibuatkan *index* untuk pencarian atau relasi akan lebih meningkatkan optimalisasi eksekusi *database* dalam jumlah yang besar.
3. Penerapan *Cartesian product* akan lebih tepat jika diterapkan pada kasus yang memiliki *rule* pengambilan data yang melibatkan banyak tabel untuk sekali *query*, selain itu *Cartesian product* juga akan lebih optimal jika menerapkan *Indexing* pada setiap *field* dalam *rule* pencarian atau pengambilan data dengan jumlah besar.

Daftar Pustaka

[1] Forta B.2002. Belajar Sendiri SQL Dalam 10 Menit. Yogyakarta : Andi.
[2] Silberschatz, Abraham, H.F. Korth, and S. Sudarshan. 2011. Database System Concepts - 6th. Ed.
[3] Codd EF. A Relational Model of Data for Large Shared Data Banks. Communications of the ACM. 1970.
[4] Simarmata J,Paryudi I. 2006. Basis Data.Yogyakarta:Andi Offset.
[5] 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.
[6] Darmanto, Eko. 2015. Analisa Optimalisasi Bahsa SQL Berdasarkan Relational Algebra pada Kasus Rekapitulasi Mahasiswa Layak Wisuda. Jurnal Simetris Vol.6 No.2.
[7] Date CJ. An Introduction to Database Systems. 8th ed. International: Pearson; 2004.
[8] Rahmadyani, Eka. 2016. Analisa Optimasi Query dalam Pembelajaran Berbasis Web dengan Metode M2S Crossover dan Chunk Crossover. Jurnal Times, Vol.V No. 2.
[9] Chen, Zhuoyan, Lu, Xunming, Qiu, Liqing. 2016. T-Tree Index Optimization Based on Main Memory Database. International Conference on Information China: Technology for Manufacturing System (ITMS 2016).
[10] Zhao, Suanchai. Lin, Quizhen. Chen, Jianyong. Wang, Xiaoming, Yu, Jianping, Ming, Zhong. 2016. Optimizing Security and Quality of Service in a real time database system using Multi object genetic Algorithm.

Biodata Penulis

Chavid Syukri Fatoni, memperoleh gelar Sarjana Komputer (S.Kom), Jurusan Teknik Informatika STMIK Sinar Nusantara Surakarta, lulus tahun 2014. Tengah menempuh study lanjut Magister Komputer (M.Kom) Program Pasca Sarjana Magister Teknik Informatika Universitas AMIKOM Yogyakarta.

Dwi Astuti, memperoleh gelar Sarjana Komputer (S.Kom), Jurusan Sistem Informasi STMIK Bina Patria Magelang, lulus tahun 2014. Tengah menempuh study lanjut Magister Komputer (M.Kom) Program Pasca Sarjana Magister Teknik Informatika Universitas AMIKOM Yogyakarta.

Kusrini, memperoleh gelar Sarjana Komputer (S.Kom) pada Program Studi Ilmu Komputer Universitas Gadjah Mada Yogyakarta tahun 2002. Memperoleh gelar Magister Komputer (M.Kom) dari Program Studi Ilmu Komputer Universitas Gadjah Mada Yogyakarta tahun 2006. Memperoleh gelar Doktor Program Studi Ilmu Komputer Universitas Gadjah Mada Yogyakarta tahun 2010. Saat ini sebagai Dosen di Universitas AMIKOM Yogyakarta.