# ANALISIS OPTIMASI *QUERY SQL* MENGGUNAKAN TEKNIK *HEURISTIC* PADA KASUS DATA TRANSAKSI PELANGGAN YANG LAYAK MENDAPATKAN REKOMENDASI PRODUK

GENTISYA TRI MARDIANI, HANI IRMAYANTI
Program Studi Teknik Informatika, Fakultas Teknik dan Ilmu Komputer
Universitas Komputer Indonesia

*Suatu perusahaan saat ini perlu melakukan inovasi dalam mempromosikan produk yang baik untuk meningkatkan penjualan produknya, salah satu caranya adalah dengan memberikan rekomendasi produk kepada pelanggan. Cara yang dilakukan untuk mendapatkan rekomendasi tersebut dilakukan dengan mengidentifikasi data transaksi penjualan kemudian dibagi berdasarkan jenis produk yang banyak dibeli oleh pelanggan, kemudian memberikan rekomendasi produk berdasarkan jenis produk yang paling sering dibeli oleh pelanggan pada periode tertentu.*

*Permasalahan yang terjadi adalah untuk menampilkan data rekomendasi produk membutuhkan waktu yang cukup lama, disebabkan oleh banyak baris data yang diolah dalam database, selain itu masalah lainnya adalah pemilihan query yang optimal dalam menampilkan data rekomendasi produk juga dapat mempengaruhi kecepatan akses terhadap data. Sehingga perlu dilakukan analisis untuk mengetahui query mana yang lebih optimal untuk menampilkan rekomendasi produk. Sebuah query dikatakan optimal apabila waktu akses terhadap data memiliki waktu paling minimum ketika melakukan operasi dengan menggunakan teknik tertentu.*

*Teknik yang digunakan di penelitian ini yaitu Teknik Heuristik. Teknik ini dipilih karena dapat mengurangi kompleksitas optimasi, selain itu juga dapat memecahkan masalah dengan sedikit mengabaikan apakah solusi dibuktikan dengan benar, namun teknik ini dapat menghasilkan solusi yang paling optimal. Penelitian ini menghasilkan rekomendasi query yang dapat digunakan, sehingga diharapkan dalam proses mengolah data rekomendasi produk di dalam database perusahaan dapat lebih cepat dilakukan.*

***Keywords : Optimasi Query, Heuristic, Aljabar Relasional, MySQL***

## PENDAHULUAN

### 1. Latar Belakang

Dalam dunia industri saat ini perusahaan perlu melakukan inovasi dalam mempromosikan produk yang baik sehingga perusahaan dapat meningkatkan penjualannya dan pelanggan dapat tertarik untuk dapat melakukan pembelian berbagai produk. Salah satu cara yang dapat dilakukan yaitu dengan mengidentifikasi data transaksi penjualan dengan membagi jenis produk yang sering dibeli oleh pelanggan dan menyediakan rekomendasi produk yang tepat. Penentuan rekomendasi produk pun memiliki beberapa persyaratan, yaitu jenis produk yang pernah dibeli oleh pelanggan, jenis produk yang paling banyak terjual di perusahaan, kemudian produk yang paling banyak terjual dan pernah dibeli oleh pelanggan dalam suatu periode tertentu yang akan dijadikan rekomendasi untuk pelanggan. Kemudian pelanggan yang layak mendapatkan rekomendasi produk tersebut adalah pelanggan yang sering bertransaksi dalam periode tertentu.

Data transaksi pelanggan yang tersimpan di dalam *database* memiliki baris yang cukup banyak, sehingga dalam proses menampilkan data rekomendasi produk pun membutuhkan waktu yang cukup lama. Hal tersebut dipengaruhi selain jumlah data yang tersimpan juga kecepatan akses data berdasarkan *query* yang digunakan untuk menampilkan data tersebut.

Permasalahan yang dibahas adalah bagaimana

mengetahui query mana yang optimal untuk menghasilkan data rekomendasi produk yang layak bagi pelanggan. Proses pengelolaan *database* tersebut menggunakan *Database Management System* (DBMS), dengan menggunakan Bahasa *Structure Query Language* (SQL). Kehandalan dari suatu *database* dapat diketahui dari proses untuk mencari perencanaan eksekusi *query* yang terbaik atau yang lebih dikenal dengan proses optimasi *query,* dalam memproses *statement – statement* SQL yang dibuat oleh user maupun aplikasinya [1]. Perencanaan *query* yang optimal, dilihat dari waktu akses yang paling minimum ketika melakukan proses pada *statement-statement* SQL dengan teknik tertentu.

Dalam menganalisis optimasi dari *query*, salah satu teknik yang dapat digunakan yaitu teknik *heuristic optimization.* Teknik ini digunakan untuk mengurangi kompleksitas optimisasi, memecahkan masalah dengan sedikit mengabaikan apakah solusinya dapat dibuktikan dengan benar, namun biasanya menghasilkan solusi yang mendekati optimal. Tujuan dari optimasi adalah untuk mengurangi sebanyak mungkin baris data yang tidak dibutuhkan dalam prosesnya. Setelah menganalisis optimasi dari *query* yang digunakan diharapkan dalam proses mengolah data rekomendasi produk di dalam *database* perusahaan dapat lebih cepat dilakukan. Data yang akan digunakan dalam penelitian ini menggunakan data simulasi yang mendekati dengan data sebenarnya, karena data yang sesungguhnya bersifat privasi.

## 2. Rumusah Masalah

Berdasarkan permasalahan tersebut, maka dapat dirumuskan perumusan masalah yaitu bagaimana proses analisis optimasi SQL menggunakan teknik *heuristic* pada kasus data transaksi pelanggan yang layak mendapatkan rekomendasi produk.

## 3. Tujuan dan Manfaat

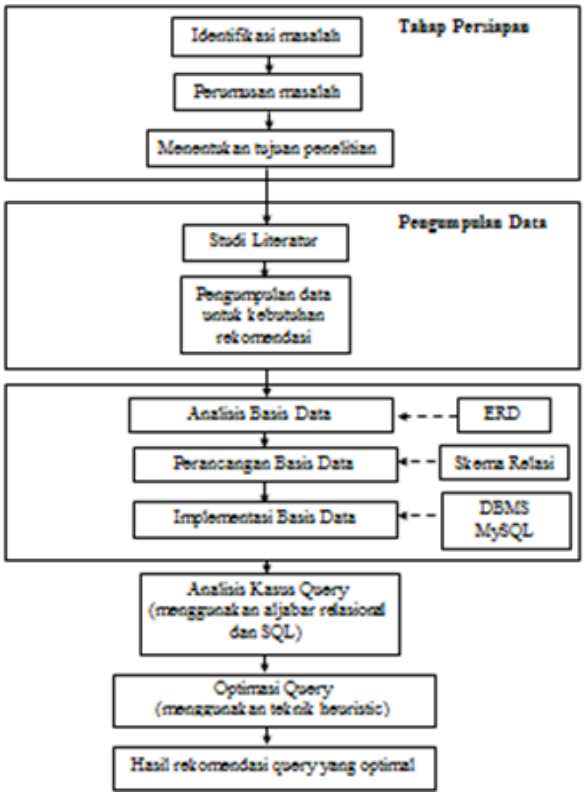
Tujuan dari penelitian ini adalah untuk mengetahui penggunaan *query* mana yang lebih optimal dalam menampilkan data pelanggan yang layak mendapatkan rekomendasi produk.

Hasil yang dicapai pada penelitian ini adalah memberikan rekomendasi penggunaan query yang akan digunakan dalam mencari data pelanggan yang layak mendapatkan rekomendasi produk.

Manfaat yang ingin dicapai adalah dengan diketahui optimasi dari suatu *query*, maka dapat membantu perusahaan mencari data pelanggan yang layak mendapatkan rekomendasi produk akan lebih cepat dalam pengaksesan pada *database* perusahaan.

## 4. Metodologi Penelitian

Proses atau alur penelitian akan disajikan dalam berupa gambar yang menjelaskan setiap tahap penelitian. Alur penelitian yang dilakukan dijelaskan dalam gambar berikut.




Gambar 1. Metodologi Penelitian

## 5. Tinjauan Pustaka

### a. Aljabar Relasional

Aljabar relasional merupakan kumpulan operasi terhadap relasi dimana setiap operasi menggunakan satu atau lebih relasi untuk menghasilkan suatu relasi yang baru. Aljabar relasional juga menyediakan seperangkat operator untuk memanipulasi data. (Fathansyah, 2012)

Operasi dasar aljabar relasional adalah sebagai berikut:

1) *Selection* (σ): operasi ini digunakan untuk menyeleksi atau mencari *record/* baris data yang memenuhi predikat atau syarat yang ditentukan. Syarat tersebut biasanya disimbolkan dengan operator perbandingan dan beberapa predikat dapat dikombinasikan dengan operator penghubung.
2) *Projection* (p): operasi ini digunakan untuk mendapatkan kolom-kolom tertentu untuk

ditampilkan. Operasi proyeksi adalah operasi unary yang mengirim relasi argumen dengan kolom- kolom tertentu.

3) *Union* (U): operasi untuk menghasilkan gabungan tabel, dengan syarat kedua tabel terdapat atribut yang sama. Operasi ini memungkinkan untuk menggabungkan data dari dua baris yang sejenis.
4) *Set-difference* (-): operasi untuk mendapatkan record-record yang berada pada suatu tabel tetapi tidak pada tabel lainnya.
5) *Cartesian-product* (X, disebut juga cross product): digunakan untuk merelasikan semua *record-record* yang berasal dari dua tabel. Operasi *cartesian product* umumnya tidak berdiri sendiri, tetapi dapat digunakan bersama dengan operasi lainnya seperti select dan project
6) *Rename* (r): digunakan untuk menyalin tabel lama ke dalam tabel baru

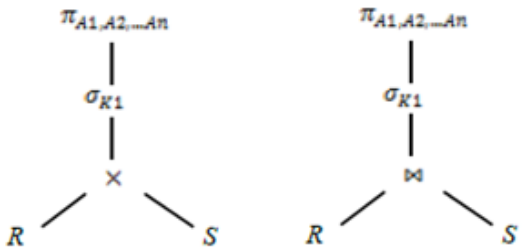
Sedangkan operasi tambahan di aljabar relasional adalah sebagai berikut:

1) *Set intersection* (Ç): digunakan untuk mendapatkan irisan (kesamaan anggota) dari dua kelompok data dari suatu tabel atau relasi. Kumpulan tuple-tuple yang berada di E1 dan berada di E2.
2) *Natural join* (∞): dilakukan jika kedua relasi memiliki satu atau lebih atribut sekutu. Kolom atribut sekutu bersifat tunggal (diambil salah satu).
3) *Theta join* (q): Kumpulan tuple-tuple E1xE2 yang nilai atribut i memenuhi relasi θ terhadap nilai atribut j
4) *Division* ( ¸ ) : Operasi yang banyak digunakan dalam query yang mencakup frase "setiap" atau "untuk semua", operasi ini juga merupakan pembagian atas tuple-tuple dari dua relasi

### b. 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 menganalisisnya. Pendekatan *Cost-Based* menggunakan statistik dalam database yang disimpan dalam data *dictionary* yang berupa meta data (Darmanto. 2015). Analisis menggunakan *query tree* dapat menggunakan 2 pendekatan yaitu pendekatan menggunakan operasi *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. Perbandingan *query tree* operasi *cartessian product* dan join

### 3. Teknik *Heuristic*

Teknik *Heuristic* atau yang biasa disebut dengan *rule based optimization* adalah optimisasi *query* dengan menggunakan aturan-aturan heuristik dan dijalankan pada *logical query plan* (rencana query secara logika) yang terdiri dari urutan operasi-operasi relasional yang biasanya digambarkan sebagai *query tree*. *Query Optimizer* mendapatkan sebuah inisial plan dari parser dan menggunakan aturan-aturan heuristik untuk mentransformasikan sebuah query ke dalam sebuah bentuk yang sama sehingga dapat diproses dengan lebih efisien. Adapun tujuan dari transformasi tersebut adalah (Sudaryanto, 2007):

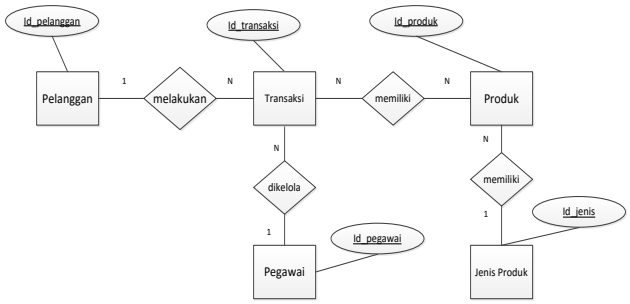
a. Standarisasi, yaitu mentransformasikan sebuah query ke dalam sebuah bentuk standar tanpa optimisasi.
b. Simplifikasi, yaitu mengeliminasi kelebihan dalam sebuah query.

Ameliorasi, yaitu menyusun ekspresi-ekspresi yang sudah dihasilkan dengan baik untuk mengevaluasi bentuk.

## PEMBAHASAN

### 1. Analisis Basis Data

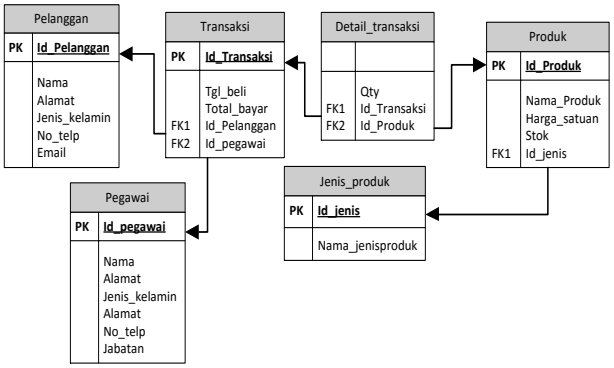
Berdasarkan kebutuhan data yang akan digunakan dalam menentukan pelanggan yang layak mendapatkan rekomendasi produk, maka dapat digambarkan relasi antar setiap entitas data menggunakan model *Entity Relationship Diagram* (ERD). ERD dapat dilihat pada gambar 3.




Gambar 3. Entity Relationship Diagram

### 2. Perancangan Basis Data

Berdasarkan hasil analisis data yang sudah dibuat sebelumnya, maka akan dilakukan perancangan data meliputi pembuatan model skema relasi dan struktur tabel untuk mengetahui relasi setiap tabel dan mengetahui strutur data dari setiap tabel yang akan dibuat dalam menentukan pelanggan yang layak mendapatkan rekomendasi produk.




Gambar 4. Skema Relasi

### 3. Implementasi *Database*

Berdasarkan analisis dan perancangan basis data yang sudah dilakukan, maka langkah selanjutnya adalah melakukan implementasi database. DBMS yang digunakan adalah MYSQL. Berikut akan dijelaskan contoh *query* dalam pembuatan *database*, pembuatan tabel, dan contoh data yang sudah dimasukkan dalam database.

#### a. Pembuatan *database*

Tabel 1 menjelaskan pembuatan *database* pada DBMS MySQL.

Tabel 1. Pembuatan Database

| Nama Database | *Query* |
|---|---|
| db_rekomendasi | Create database db rekomendasi; |

#### b. Pembuatan Tabel Pelanggan

Tabel 2 menjelaskan contoh pembuatan tabel pelanggan.

Tabel 2. Pembuatan Tabel

| Nama Tabel | *Query* |
|---|---|
| pelanggan | CREATE TABLE IF NOT EXISTS `pelanggan` ( `Id_pelanggan` int(10) NOT NULL AUTO_INCREMENT, `Nama` varchar(30) NOT NULL, `Alamat` varchar(50) NOT NULL, `Jenis_kelamin` enum ('L','P') NOT NULL, `No_telp` varchar(12) NOT NULL, `Email` varchar(30) NOT NULL, PRIMARY KEY (`Id_pelanggan`) ) ENGINE=InnoDB DEFAULT CHARSET=latin1; |

Pembuatan tabel lain memiliki perintash yang sama dengan tabel pelanggan, hanya nama kolom dan ukuran, serta tipe data yang digunakannya berbeda, mengikuti hasil perancangan basis data pada tahap sebelumnya.

c. Pemasukkan data pada tabel pelanggan

Gambar 5 berikut mejelaskan contoh data yang sudah terisi pada tabel pelanggan.

| | Id_pelanggan | Nama | Alamat | Jenis_kelamin | No_telp | Email |
|---|---|---|---|---|---|---|
| Ubah Salin Hapus | 1 | Annisa Amelia | Jalan Proklamasi No 2 Banjar | P | 08578683437 | annisaamelia@gmail.com |
| Ubah Salin Hapus | 2 | Ria | Jalan Iskandardinata no 10 Banjar | P | 081214478990 | ria@gmail.com |
| Ubah Salin Hapus | 3 | Asri Nurmala | Jalan Guntur no 38 Banjar | P | 085727604692 | asrinurmala@gmail.com |
| Ubah Salin Hapus | 4 | Hardo Pratama | Jalan Dago No 10 Bandung | L | 085270950747 | hardopratama@gmail.com |
| Ubah Salin Hapus | 5 | Asep Suparman | Jalan Tubagus Ismail 167 Bandung | L | 087867675456 | asep.s@gmail.com |
| Ubah Salin Hapus | 6 | Udin | Bandung | L | 085786834372 | udin@gmail.com |
| Ubah Salin Hapus | 7 | Scott Smith | Ap #384-2374 In St. | L | 6114-5736-54 | Sed.nulla@estmollis.co.uk |
| Ubah Salin Hapus | 8 | Yasir Goodwin | P.O. Box 867, 4136 Tristique Road | L | 1065-0095-67 | in.cursus.et@eunibh.edu |
| Ubah Salin Hapus | 9 | Iris Collier | Ap #446-2278 Pretium Rd. | L | 8122-4447-22 | lacinia.mattis@acorci.com |
| Ubah Salin Hapus | 10 | Colorado Jacobson | 739-2149 Vitae St. | P | 9478-8503-49 | vehicula.et@etcommodoat.ca |
| Ubah Salin Hapus | 11 | Scarlett Knox | Ap #288-371 Lobortis, Avenue | P | 1315-6625-32 | lacus@augue.org |
| Ubah Salin Hapus | 12 | Timothy Rojas | 319-9629 Sed Av. | P | 6503-5019-88 | Morbi.accumsan@Mauriseuturpis. |
| Ubah Salin Hapus | 13 | Quemby Neal | 1300 Eget Ave | L | 8062-4040-73 | varius.ultrices.mauris@arteliec |
| Ubah Salin Hapus | 14 | Amber Eaton | P.O. Box 102, 6661 Eget Street | P | 4560-9083-95 | convallis.convallis@penatibus. |
| Ubah Salin Hapus | 15 | Phyllis Haley | 4296 Gravida St. | L | 9763-9964-12 | in@lacinia.ca |
| Ubah Salin Hapus | 16 | Shea Parks | Ap #330-828 Tellus. Rd. | L | 1487-9369-10 | et@mi.co.uk |

Gambar 5. Data Pelanggan

### 1. Analisis Kasus

Berdasarkan basis data yang sudah dibuat dan data yang sudah terisi pada setiap tabel, langkah selanjutnya adalah melakukan rekapitulasi jumlah kolom dan jumlah baris data yang tersimpan sebelum melakukan perhitungan nilai estimasi dari *query* yang dibuat. Tabel 3 menjelaskan jumlah data dari setiap tabel.

Tabel 3. Jumlah Kolom dan Baris pada Setiap Tabel

| No | Nama Tabel | Jumlah Kolom | Jumlah Data |
|---|---|---|---|
| 1 | Pelanggan | 6 | 1005 |
| 2 | Produk | 5 | 73 |
| 3 | Transaksi | 5 | 152 |
| 4 | Detai transaksi | 3 | 304 |
| 5 | Jenis Produk | 2 | 4 |
| 6 | Pegawai | 6 | 5 |

Batasan yang diberikan untuk mendapatkan rekomendasi produk adalah sebagai berikut:

a. Produk yang direkomendasikan adalah beberapa produk yang jenis produknya pernah dibeli oleh pelanggan
b. Produk yang direkomendasikan kepada pelanggan adalah produk yang jenis produknya paling laris terjual dan pernah dibeli oleh pelanggan tersebut dalam periode transaksi 3 bulan terakhir
c. Pelanggan yang layak mendapatkan rekomendasi produk merupakan pelanggan yang melakukan lebih dari 1 transaksi dalam periode 3 bulan terakhir

Berdasarkan batasan yang sudah ditentukan, maka perlu menganalisis *query* yang digunakan untuk menampilkan data tersebut. Teknik yang akan dilakukan adalah dengan cara menuliskan query SQL kemudian diterjemahkan ke dalam bahasa Aljabar Relasional, lalu akan dihitung estimasi query menggunakan teknik *Heuristic* dan akan digambarkan struktur *tree* dari setiap query yang dibuat. Operasi yang digunakan untuk menampilkan data menggunakan 2 operasi, yaitu operasi *cartessian product* dan operasi *join*.

### a. Analisis Data Produk yang direkomendasikan

Berikut ini hasil analisis data produk yang direkomendasikan :

1) Operasi *cartessian product*

Berikut akan dijelaskan tahapan menampilkan data menggunakan operasi *cartessian product*:

- Sintaks SQL menggunakan operasi *cartessian product*:

Tabel 4. *Query SQL* menggunakan *cartessian product*

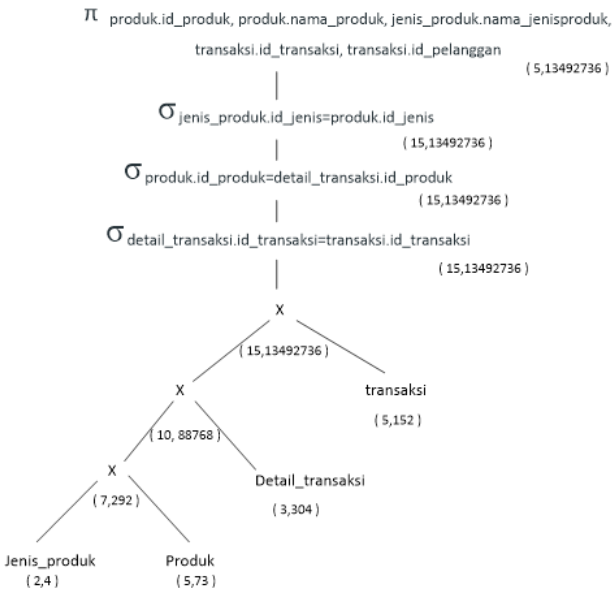
| Perintah | *Query* |
|---|---|
| Tampilkan data produk, data jenis produk, dan data transaksi, dimana produk tersebut pernah dibeli oleh pelanggan | **SELECT** produk.id_produk, produk.nama_produk, nama_jenisproduk, transaksi.Id_transaksi, transaksi.id_pelanggan **FROM** jenis_produk, produk, detail_transaksi, transaksi **WHERE** jenis_produk.id_jenis=produk.id_jenis **AND** produk.id_produk=detail_transaksi.Id_produk **AND** transaksi.id_transaksi=detail_transaksi.id_transaksi; |

- Aljabar Relasional menggunakan operasi *cartessian product:*

  $\rho$ produk.id_produk, produk.nama_produk, jenis_produk.nama_jenisproduk, transaksi.id_transaksi, transaksi.id_pelanggan ($\sigma$jenis_produk.id_jenis=produk.id_jenis $\wedge$ produk.id_produk=detail_transaksi.id_produk $\wedge$ detail_transaksi.id_transaksi=transaksi.id_transaksi (jenis_produk x produk x detail_transaksi x transaksi ) )

- Struktur *Query Tree* menggunakan operasi *cartessian product*:

  Berikut ini merupakan struktur *tree Cartessian product* untuk mendapatkan rekomendasi produk:

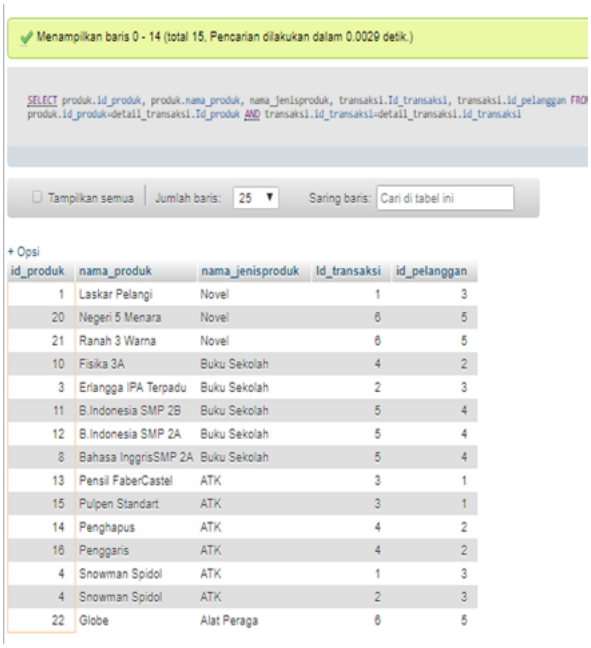



Gambar 6. Struktur *tree operasi cartessian product*

Berdasarkan *tree* yang sudah dibuat terlihat bahwa hasil estimasi *query* menggunakan operasi *cartessian product* adalah menghasilkan 5 kolom dan 13.492.736 baris.

- Hasil menampilkan data pada *database*:

  Berikut tampilan dari *database* yang menampilkan data produk, data jenis produk, dan data transaksi, dimana produk tersebut pernah dibeli oleh pelanggan.




| id_produk | nama_produk | nama_jenisproduk | Id_transaksi | id_pelanggan |
|---|---|---|---|---|
| 1 | Laskar Pelangi | Novel | 1 | 3 |
| 20 | Negeri 5 Menara | Novel | 6 | 5 |
| 21 | Ranah 3 Warna | Novel | 6 | 5 |
| 10 | Fisika 3A | Buku Sekolah | 4 | 2 |
| 3 | Erlangga IPA Terpadu | Buku Sekolah | 2 | 3 |
| 11 | B.Indonesia SMP 2B | Buku Sekolah | 5 | 4 |
| 12 | B.Indonesia SMP 2A | Buku Sekolah | 5 | 4 |
| 8 | Bahasa InggrisSMP 2A | Buku Sekolah | 5 | 4 |
| 13 | Pensil FaberCastel | ATK | 3 | 1 |
| 15 | Pulpen Standart | ATK | 3 | 1 |
| 14 | Penghapus | ATK | 4 | 2 |
| 16 | Penggaris | ATK | 4 | 2 |
| 4 | Snowman Spidol | ATK | 1 | 3 |
| 4 | Snowman Spidol | ATK | 2 | 3 |
| 22 | Globe | Alat Peraga | 6 | 5 |

Gambar 7. Tampilan hasil eksekusi *query* operasi *cartessian product*

Berdasarkan hasil eksekusi *query* pada DBMS MYSQL menghasilkan waktu akses sebesar 0,0029 detik jika menggunakan operasi *cartessian product*.

### 2) Operasi *JOIN*

Berikut akan dijelaskan tahapan menampilkan data menggunakan operasi *JOIN*:

- Sintaks SQL menggunakan operasi *JOIN*:

Tabel 5. *Query SQL* menggunakan operasi *join*

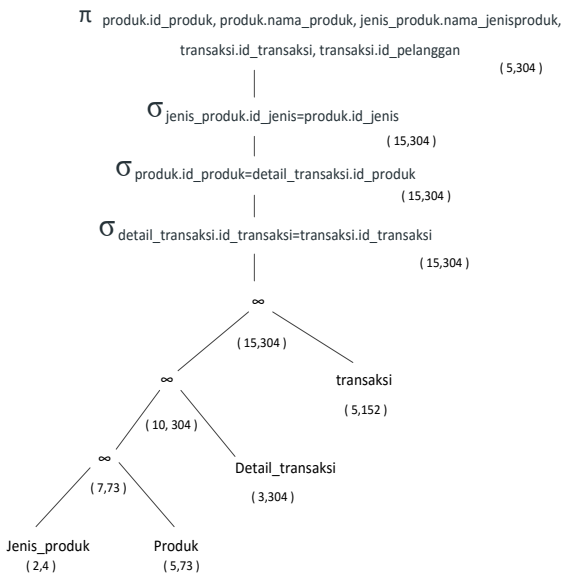
| Perintah | *Query* |
|---|---|
| Tampilkan data produk, data jenis produk, dan data transaksi, dimana produk tersebut pernah dibeli oleh pelanggan | **SELECT** produk.id_produk, produk.nama_produk, nama_jenisproduk, transaksi.Id_transaksi, transaksi.id_pelanggan **FROM** jenis_produk **JOIN** produk **ON** jenis_produk.id_jenis=produk.id_jenis **JOIN** detail_transaksi **ON** produk.id_produk=detail_transaksi.id_produk **JOIN** transaksi **ON** detail_transaksi.id_transaksi =transaksi.id_transaksi; |

- Aljabar Relasional menggunakan operasi *JOIN:*

$$\pi_{\text{produk.id\_produk, produk.nama\_produk, jenis\_produk.nama\_jenisproduk, transaksi.id\_transaksi, transaksi.id\_pelanggan}} ( \text{jenis\_produk} \infty_{\text{jenis\_produk.id\_jenis=produk.id\_jenis}} \text{produk} \infty_{\text{produk.id\_produk=detail\_transaksi.id\_produk}} \text{detail\_transaksi} \infty_{\text{detail\_transaksi.id\_transaksi=transaksi.id\_transaksi}} \text{transaksi} )$$

- Struktur *Query Tree* menggunakan operasi *JOIN*:

  Berikut ini merupakan struktur *tree JOIN* untuk mendapatkan rekomendasi produk:

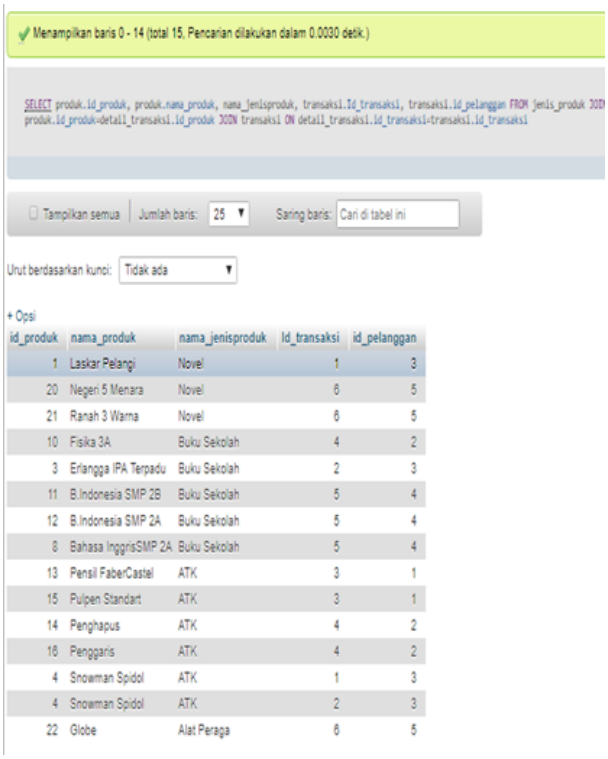



Gambar 8. *Struktur tree* operasi join

Berdasarkan *tree* yang sudah dibuat terlihat bahwa hasil estimasi *query* menggunakan operasi *JOIN* adalah menghasilkan 5 kolom dan 304 baris.

- Hasil menampilkan data pada *database*:

  Berikut tampilan dari database yang menampilkan data produk, data jenis produk, dan data transaksi, dimana produk tersebut pernah dibeli oleh pelanggan.




| id_produk | nama_produk | nama_jenisproduk | Id_transaksi | id_pelanggan |
|---|---|---|---|---|
| 1 | Laskar Pelangi | Novel | 1 | 3 |
| 20 | Negeri 5 Menara | Novel | 6 | 5 |
| 21 | Ranah 3 Warna | Novel | 6 | 5 |
| 10 | Fisika 3A | Buku Sekolah | 4 | 2 |
| 3 | Erlangga IPA Terpadu | Buku Sekolah | 2 | 3 |
| 11 | B.Indonesia SMP 2B | Buku Sekolah | 5 | 4 |
| 12 | B.Indonesia SMP 2A | Buku Sekolah | 5 | 4 |
| 8 | Bahasa InggrisSMP 2A | Buku Sekolah | 5 | 4 |
| 13 | Pensil FaberCastel | ATK | 3 | 1 |
| 15 | Pulpen Standart | ATK | 3 | 1 |
| 14 | Penghapus | ATK | 4 | 2 |
| 16 | Penggaris | ATK | 4 | 2 |
| 4 | Snowman Spidol | ATK | 1 | 3 |
| 4 | Snowman Spidol | ATK | 2 | 3 |
| 22 | Globe | Alat Peraga | 6 | 5 |

Gambar 9. Hasil tampilan eksekusi *query* operasi *join*

Berdasarkan hasil eksekusi query pada DBMS MYSQL menghasilkan waktu akses sebesar 0,0030 detik jika menggunakan operasi *join*.

### b. Analisis Produk yang direkomendasikan pada periode 3 bulan terakhir

Berikut ini hasil analisis data produk yang direkomendasikan:

#### 1) Operasi *cartessian product*

Berikut akan dijelaskan tahapan menampilkan data menggunakan operasi *cartessian product*:

- Sintaks SQL menggunakan operasi *cartessian product*:

Tabel 6. *Query SQL* operasi *cartessian product*

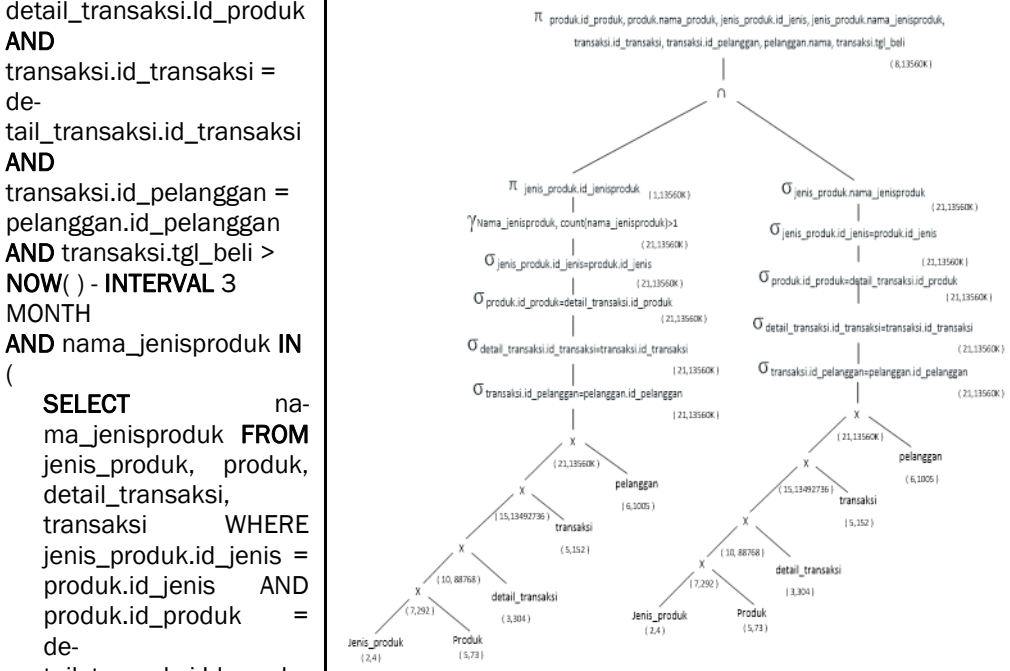
| Perintah | *Query* |
|---|---|
| Tampilkan data produk, data jenis produk, data transaksi, dan data pelanggan, dimana jenis produknya pernah dibeli oleh pelanggan lebih dari 1 kali dalam periode transaksi 3 bulan terakhir | **SELECT** produk.id_produk, produk.nama_produk, jenis_produk.id_jenis, jenis_produk.nama_jenisproduk, transaksi.Id_transaksi, transaksi.id_pelanggan, pelanggan.nama, transaksi.tgl_beli **FROM** jenis_produk, produk, detail_transaksi, transaksi, pelanggan **WHERE** jenis_produk.id_jenis = produk.id_jenis **AND** produk.id_produk = detail_transaksi.Id_produk **AND** transaksi.id_transaksi = detail_transaksi.id_transaksi **AND** transaksi.id_pelanggan = pelanggan.id_pelanggan **AND** transaksi.tgl_beli > **NOW**( ) - **INTERVAL** 3 MONTH **AND** nama_jenisproduk **IN** ( **SELECT** nama_jenisproduk **FROM** jenis_produk, produk, detail_transaksi, transaksi WHERE jenis_produk.id_jenis = produk.id_jenis AND produk.id_produk = detail_transaksi.Id_produk **AND** transaksi.id_transaksi = detail_transaksi.id_transaksi GROUP BY nama_jenisproduk **HAVING COUNT**( nama_jenisproduk)>1); |

- Aljabar Relasional menggunakan operasi *cartessian product:*

p produk.id_produk, produk.nama_produk, jenis_produk.id_jenis, jenis_produk.nama_jenisproduk, transaksi.id_transaksi, pelanggan.nama transaksi.id_pelanggan, pelanggan.nama, transaksi.tgl_beli
(σjenis_produk.id_jenis=produk.id_jenis
Λ produk.id_produk=detail_transaksi.id_produk
Λ detail_transaksi.id_transaksi=transaksi.id_transaksi
Λ transaksi.id_pelanggani=pelanggan.id_pelanggan
Λ transaksi.tgl_beli> ( NOW() – INTERVAL 3 MONTH )
Λ jenis_produk.nama_jenisproduk (jenis_produk x produk x detail_transaksi x transaksi x pelanggan))
∩ (pnama_jenisproduk (σjenis_produk.id_jenis=produk.id_jenis
Λ produk.id_produk=detail_transaksi.id_produk
Λ detail_transaksi.id_transaksi=transaksi.id_transaksi
Λ transaksi.id_pelanggan=pelanggan.id_pelanggan
(γNama_jenisproduk, count(nama_jenisproduk)>1 (jenis_produk x produk x detail_transaksi x transaksi x pelanggan ) )

- Struktur *Tree* menggunakan operasi *cartessian product*

Berikut ini merupakan struktur *tree Cartessian product* untuk mendapatkan rekomendasi produk:




Berdasarkan *tree* yang sudah dibuat dapat dilihat bahwa estimasi *query* menggunakan operasi *cartessian product* adalah menampilkan 8 kolom dan 13.560.000 baris.

- Hasil menampilkan data pada *database*

Berikut tampilan dari database yang menampilkan data produk, data jenis produk, data transaksi, dan data pelanggan, dimana jenis produknya pernah dibeli oleh pelanggan lebih dari 1 kali dalam periode transaksi 3 bulan terakhir.

Menampilkan baris 0 - 13 (total 14, Pencarian dilakukan dalam 0.0078 detik.)

```
SELECT produk.id_produk, produk.nama_produk, jenis_produk.id_jenis, jenis_produk.nama_jenisproduk, transaksi.Id_transaksi, transaksi.id_pela
transaksi, pelanggan WHERE jenis_produk.id_jenis = produk.id_jenis AND produk.id_produk = detail_transaksi.Id_produk AND transaksi.id_transa
(transaksi.tgl_beli > NOW( ) - INTERVAL 3 MONTH) AND nama_jenisproduk IN ( SELECT nama_jenisproduk FROM jenis_produk, produk, detail_transak
detail_transaksi.Id_produk AND transaksi.id_transaksi = detail_transaksi.id_transaksi GROUP BY nama_jenisproduk HAVING COUNT( nama_jenisprod
```

Tampilkan semua | Jumlah baris: 25 | Saring baris: Cari di tabel ini

+ Opsi

| id_produk | nama_produk | id_jenis | nama_jenisproduk | Id_transaksi | id_pelanggan | nama | tgl_beli |
|---|---|---|---|---|---|---|---|
| 1 | Laskar Pelangi | 1 | Novel | 1 | 3 | Asri Nurmala | 2017-08-28 |
| 20 | Negeri 5 Menara | 1 | Novel | 6 | 5 | Asep Suparman | 2017-09-10 |
| 21 | Ranah 3 Warna | 1 | Novel | 6 | 5 | Asep Suparman | 2017-09-10 |
| 3 | Erlangga IPA Terpadu | 2 | Buku Sekolah | 2 | 3 | Asri Nurmala | 2017-08-28 |
| 10 | Fisika 3A | 2 | Buku Sekolah | 4 | 2 | Ria | 2017-09-08 |
| 11 | B.Indonesia SMP 2B | 2 | Buku Sekolah | 5 | 4 | Hardo Pratama | 2017-09-11 |
| 12 | B.Indonesia SMP 2A | 2 | Buku Sekolah | 5 | 4 | Hardo Pratama | 2017-09-11 |
| 8 | Bahasa InggrisSMP 2A | 2 | Buku Sekolah | 5 | 4 | Hardo Pratama | 2017-09-11 |
| 4 | Snowman Spidol | 3 | ATK | 1 | 3 | Asri Nurmala | 2017-08-28 |
| 4 | Snowman Spidol | 3 | ATK | 2 | 3 | Asri Nurmala | 2017-08-28 |
| 13 | Pensil FaberCastel | 3 | ATK | 3 | 1 | Annisa Amelia | 2017-09-01 |
| 15 | Pulpen Standart | 3 | ATK | 3 | 1 | Annisa Amelia | 2017-09-01 |
| 14 | Penghapus | 3 | ATK | 4 | 2 | Ria | 2017-09-08 |
| 16 | Penggaris | 3 | ATK | 4 | 2 | Ria | 2017-09-08 |

Gambar 10. tampilan hasil eksekusi *query operasi cartessian product*

Berdasarkan hasil eksekusi *query* pada DBMS MYSQL menghasilkan waktu akses sebesar 0,0078 detik jika menggunakan operasi *cartessian product*.

### c. Analisis data pelanggan yang layak mendapatkan rekomendasi product

Proses yang dilakukan untuk mengetahui pelanggan yang layak mendapatkan rekomendasi produk merupakan pelanggan yang melakukan lebih dari 1 transaksi dalam periode 3 bulan terakhir adalah dengan menyimpan *query* dari tahap sebelumnya ke dalam *view*.

Berikut adalah sintaks SQL yang digunakan untuk membuat view:

**CREATE VIEW** Namaview **AS** query;

Nama *view* yang dibuat adalah ***rekomendasi***, dan *query* yang dibuat adalah query yang berasal dari tahap sebelumnya.

#### 1) Operasi *cartessian product*

Berikut akan dijelaskan tahapan menampilkan data menggunakan operasi *cartessian product*:

- Sintaks SQL menggunakan operasi *cartessian product*:

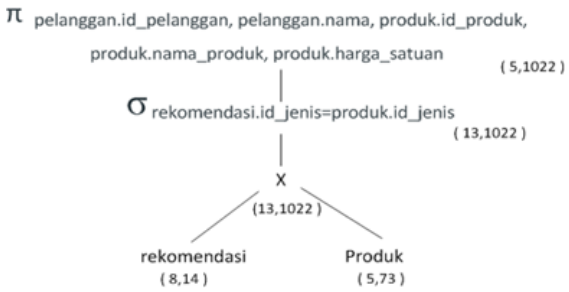
Tabel 7. Query SQL operasi cartessian product

| Perintah | Query |
|---|---|
| Tampilkan data pelanggan, nama jenis produk, dan data produk yang direkomendasikan, berdasarkan tabel view rekomendasi dan tabel produk. | **SELECT** id_pelanggan, nama, nama_jenisproduk, produk.id_produk, produk.nama_produk, produk.harga_satuan **FROM** rekomendasi, produk **WHERE** rekomendasi.id_jenis **IN** (produk.id_jenis) |

- Aljabar Relasional menggunakan operasi *cartessian product:*

$$\rho_{\text{pelanggan.id\_pelanggan, pelanggan.nama, produk.id\_produk, produk.nama\_produk, produk.harga\_satuan}} \left( \sigma_{\text{rekomendasi.id\_jenis=produk.id\_jenis}} (\text{rekomendasi} \times \text{produk}) \right)$$

- Struktur *tree* menggunakan operasi *cartessian product*:

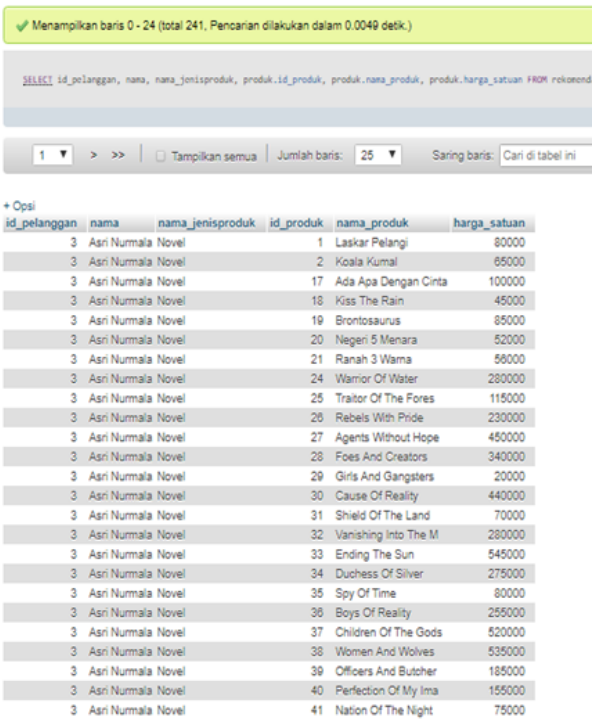



Gambar 11. struktur *tree* operasi *cartessian product*

Berdasarkan *tree* yang sudah dibuat dapat dilihat bahwa estimasi query menggunakan operasi *cartessian product* adalah menampilkan 5 kolom dan 1022 baris.

- Hasil menampilkan data pada *database*

Berikut tampilan dari *database* yang menampilkan data pelanggan, nama jenis produk, dan data produk yang direkomendasikan, berdasarkan tabel *view* rekomendasi dan tabel produk.



Menampilkan baris 0 - 24 (total 241, Pencarian dilakukan dalam 0.0049 detik.)

```
SELECT id_pelanggan, nama, nama_jenisproduk, produk.id_produk, produk.nama_produk, produk.harga_satuan FROM rekomend
```

1 > >> Tampilkan semua Jumlah baris: 25 Saring baris: Cari di tabel ini

+ Opsi

| id_pelanggan | nama | nama_jenisproduk | id_produk | nama_produk | harga_satuan |
|---|---|---|---|---|---|
| 3 | Asri Nurmala | Novel | 1 | Laskar Pelangi | 80000 |
| 3 | Asri Nurmala | Novel | 2 | Koala Kumal | 65000 |
| 3 | Asri Nurmala | Novel | 17 | Ada Apa Dengan Cinta | 100000 |
| 3 | Asri Nurmala | Novel | 18 | Kiss The Rain | 45000 |
| 3 | Asri Nurmala | Novel | 19 | Brontosaurus | 85000 |
| 3 | Asri Nurmala | Novel | 20 | Negeri 5 Menara | 52000 |
| 3 | Asri Nurmala | Novel | 21 | Ranah 3 Warna | 56000 |
| 3 | Asri Nurmala | Novel | 24 | Warrior Of Water | 280000 |
| 3 | Asri Nurmala | Novel | 25 | Traitor Of The Fores | 115000 |
| 3 | Asri Nurmala | Novel | 26 | Rebels With Pride | 230000 |
| 3 | Asri Nurmala | Novel | 27 | Agents Without Hope | 450000 |
| 3 | Asri Nurmala | Novel | 28 | Foes And Creators | 340000 |
| 3 | Asri Nurmala | Novel | 29 | Girls And Gangsters | 20000 |
| 3 | Asri Nurmala | Novel | 30 | Cause Of Reality | 440000 |
| 3 | Asri Nurmala | Novel | 31 | Shield Of The Land | 70000 |
| 3 | Asri Nurmala | Novel | 32 | Vanishing Into The M | 280000 |
| 3 | Asri Nurmala | Novel | 33 | Ending The Sun | 545000 |
| 3 | Asri Nurmala | Novel | 34 | Duchess Of Silver | 275000 |
| 3 | Asri Nurmala | Novel | 35 | Spy Of Time | 80000 |
| 3 | Asri Nurmala | Novel | 36 | Boys Of Reality | 255000 |
| 3 | Asri Nurmala | Novel | 37 | Children Of The Gods | 520000 |
| 3 | Asri Nurmala | Novel | 38 | Women And Wolves | 535000 |
| 3 | Asri Nurmala | Novel | 39 | Officers And Butcher | 185000 |
| 3 | Asri Nurmala | Novel | 40 | Perfection Of My Ima | 155000 |
| 3 | Asri Nurmala | Novel | 41 | Nation Of The Night | 75000 |

Gambar 12. Tampilan hasil eksekusi query

Berdasarkan hasil eksekusi *query* pada DBMS MYSQL menghasilkan waktu akses sebesar 0,0049 detik jika menggunakan operasi *cartessian product*.

### 2) Operasi *join*

Berikut akan dijelaskan tahapan menampilkan data menggunakan operasi *JOIN*:

- Sintaks SQL menggunakan operasi *join*:

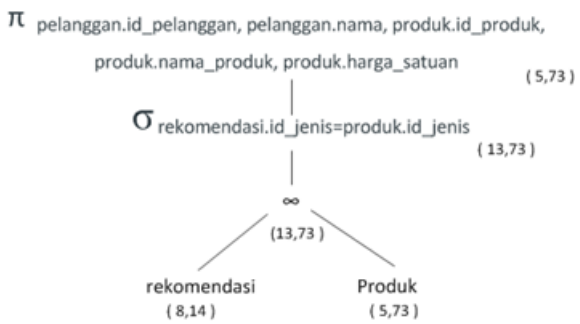
Tabel 8. Query SQL menggunakan operasi join

| Perintah | Query |
|---|---|
| Tampilkan data pelanggan, nama jenis produk, dan data produk yang direkomendasikan, berdasarkan tabel view rekomendasi dan tabel produk. | **SELECT** id_pelanggan, nama,nama_jenisproduk,produk.id_produk, produk.nama_produk, produk.harga_satuan **FROM** rekomendasi **JOIN** produk **WHERE** rekomendasi.id_jenis **IN** (produk.id_jenis) |

- Aljabar Relasional menggunakan operasi *cartessian product:*

$p$ *pelanggan.id_pelanggan, pelanggan.nama, produk.id_produk, produk.nama_produk, produk.harga_satuan* (*rekomendasi* $\infty$ *rekomendasi.id_jenis=produk.id_jenis produk* )

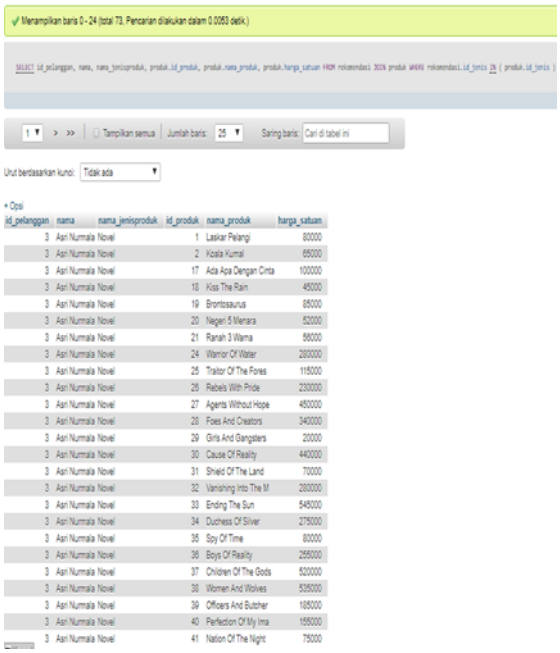
- Struktur *tree* menggunakan operasi *join*:




Berdasarkan tree yang sudah dibuat dapat dilihat bahwa estimasi query menggunakan operasi *Join* adalah menampilkan 5 kolom dan 73 baris.

- Hasil menampilkan data pada *database*

Berikut tampilan dari *database* yang menampilkan data pelanggan, nama jenis produk, dan data produk yang direkomendasikan, berdasarkan tabel view rekomendasi dan tabel produk.



Menampilkan baris 0 - 24 (total 73, Pencarian dilakukan dalam 0.0053 detik.)

```
SELECT id_pelanggan, nama, nama_jenisproduk, produk.id_produk, produk.nama_produk, produk.harga_satuan FROM rekomendasi JOIN produk WHERE rekomendasi.id_jenis IN ( produk.id_jenis )
```

1 > >> Tampilkan semua Jumlah baris: 25 Saring baris: Cari di tabel ini

Urut berdasarkan kunci: Tidak ada

+ Opsi

| id_pelanggan | nama | nama_jenisproduk | id_produk | nama_produk | harga_satuan |
|---|---|---|---|---|---|
| 3 | Asri Nurmala | Novel | 1 | Laskar Pelangi | 80000 |
| 3 | Asri Nurmala | Novel | 2 | Koala Kumal | 65000 |
| 3 | Asri Nurmala | Novel | 17 | Ada Apa Dengan Cinta | 100000 |
| 3 | Asri Nurmala | Novel | 18 | Kiss The Rain | 45000 |
| 3 | Asri Nurmala | Novel | 19 | Brontosaurus | 85000 |
| 3 | Asri Nurmala | Novel | 20 | Negeri 5 Menara | 52000 |
| 3 | Asri Nurmala | Novel | 21 | Ranah 3 Warna | 56000 |
| 3 | Asri Nurmala | Novel | 24 | Warrior Of Water | 280000 |
| 3 | Asri Nurmala | Novel | 25 | Traitor Of The Fores | 115000 |
| 3 | Asri Nurmala | Novel | 26 | Rebels With Pride | 230000 |
| 3 | Asri Nurmala | Novel | 27 | Agents Without Hope | 450000 |
| 3 | Asri Nurmala | Novel | 28 | Foes And Creators | 340000 |
| 3 | Asri Nurmala | Novel | 29 | Girls And Gangsters | 20000 |
| 3 | Asri Nurmala | Novel | 30 | Cause Of Reality | 440000 |
| 3 | Asri Nurmala | Novel | 31 | Shield Of The Land | 70000 |
| 3 | Asri Nurmala | Novel | 32 | Vanishing Into The M | 280000 |
| 3 | Asri Nurmala | Novel | 33 | Ending The Sun | 545000 |
| 3 | Asri Nurmala | Novel | 34 | Duchess Of Silver | 275000 |
| 3 | Asri Nurmala | Novel | 35 | Spy Of Time | 80000 |
| 3 | Asri Nurmala | Novel | 36 | Boys Of Reality | 255000 |
| 3 | Asri Nurmala | Novel | 37 | Children Of The Gods | 520000 |
| 3 | Asri Nurmala | Novel | 38 | Women And Wolves | 535000 |
| 3 | Asri Nurmala | Novel | 39 | Officers And Butcher | 185000 |
| 3 | Asri Nurmala | Novel | 40 | Perfection Of My Ima | 155000 |
| 3 | Asri Nurmala | Novel | 41 | Nation Of The Night | 75000 |

Gambar 13. Tampilan hasil eksekusi query

Berdasarkan hasil eksekusi *query* pada DBMS MYSQL menghasilkan waktu akses sebesar 0,0053 detik jika menggunakan operasi *join*.

## KESIMPULAN

Berdasarkan hasil analisis dan implementasi eksekusi *query* yang sudah dilakukan, maka dapat disimpulkan bahwa penggunaan operasi yang lebih optimal dalam menampikkan data yang dibutuhkan dari beberapa tabel adalah menggunakan operasi *join*, karena operasi *join* menghasilkan jumlah estimasi kolom dan baris yang lebih kecil jika dibandingkan dengan operasi *cartessian product*. Namun untuk beberapa kasus operasi *cartessian product* akan menghasilkan waktu akses yang lebih kecil jika dibandingkan dengan operasi *join*. Hal tersebut terjadi karena ada beberapa faktor yang mempengaruhi, seperti jumlah baris data yang tersimpan, kecepatan *processor* yang digunakan, serta faktor lain yang perlu dilakukan analisis lebih lanjut.

## DAFTAR PUSTAKA

S.Sudaryanto, N. 2007. Techno.com, Vol.7 No.2, Mei 2007.

Fathansyah.2012. Basis Data. Bandung: Penerbit Informatika

E.Darmanto. 2015. Jurnal SIMETRIS, Vol.6 No.2, November 2015, ISSN:2252-4983