PQL: Operasi komposisi dan jaringan semantik data

Benhard Sitohang

Laboratorium Basis Data, Departemen Teknik Informatika, ITB
Jl. Ganesa 10, Bandung 40132
Telp. 022-2503031, Fax 022-2502746, E-mail : benhard@informatika.org

Sari

Public Query Language (PQL) adalah bahasa penelusuran (*query*) pada basis data model relasi, yang lebih bersifat *user friendly* relatif terhadap bahasa penelusuran pada Generasi ke-IV (SQL, QBE, QUEL, dll.). Dari sisi struktur bahasa, PQL didasarkan pada struktur sintaks linier (tidak terdapat struktur blok rekursif). Pada sisi tampilan, hasil eksekusi dibentuk berdasarkan definisi entitas (bukan tuppel, seperti pada SQL). Dengan demikian duplikasi data pada tampilan dapat dihilangkan. Proses *query* dan prinsip tampilan yang dimaksudkan pada PQL dapat diimplementasikan dalam DBMS, dengan menerapkan teknik penggabungan (operator komposisi), yang didasarkan pada prinsip operator join dan navigasi pada jaringan semantik data. Operator komposisi dan jaringan semantik data adalah topik utama penelitian yang dijelaskan pada tulisan ini.

Kata kunci:bahasa penelusuran, basis data, ketergantungan fungsional, model relasi, navigasi, operator relasi, Public Query Language (PQL).

Abstract

PQL: Composition operator and semantic data network

Public Query Language (PQL) is the syntax of query for relational database, which is categorized as highly user friendly compared to the 4-th generation language (SQL, QBE, QUEL, etc.). As a query language, PQL is constructed as linear syntax (without block recursive). Visualization of the results is constructed as an entity (not tupple, as happened in SQL). As a consequence, duplication of data on the results could be rejected. Query process and the principle of visualization of results of PQL can be adapted as part of DBMS, using composition method (Composition Operator), developed as an intepretation of join operator and navigation, supported by Data Semantic Network. Both, Composition Operator and Data Semantic Network are main topic discussed in this article, as the result of this research on PQL.

Key words: database, functional dependency, navigation, Public Query language (PQL), query language, relational model, relational Operator.

1 Pendahuluan

Konsep Basis data model relasi memungkinkan penggunaan struktur bahasa penelusuran (*query*) yang bersifat *non-procedural* (dikategorikan sebagai bahasa Generasi ke-IV). Sebagai contoh, adalah SQL (*Structured Query Language*)[2], yang dikenal dengan tiga kata kunci *Select-From-Where*, dan dianggap sebagai standar bahasa penelusuran (*de facto*, OSI-9075). Contoh lain adalah QUEL (termasuk dalam kategori *tupple relational calculus*)[7], QBE (menggunakan tabel dua dimensi/skeleton)[17]. Sesuai dengan strukturnya, bahasa penelusuran generasi ke-IV menyajikan kemudahan bagi pemakai yang memahami prinsip skema logis basis data, relasi, skema relasi, atribut, atribut kunci, dan tuppel. Untuk pemakai dalam arti luas (publik/casual, yang kurang memahami skema lojik basis data), maka bahasa *non-procedural* generasi ke-IV ternyata relatif belum dapat mendukung kemudahan yang dimaksudkan (*user friendly*). Kesulitan timbul sebagai akibat dari tuntutan atas pemahaman terhadap skema relasi, atribut serta atribut kunci, di samping kompleksitas struktur *query* yang meningkat, bila jumlah relasi yang berperan juga bertambah (berdampak pada penggunaan blok rekursif).

Untuk meningkatkan kemudahan bagi pemakai dalam menggunakan bahasa penelusuran, serta meningkatkan kemudahan dalam menginterpretasikan hasil penelusuran, dikembangkan struktur bahasa dan metode pemrosesan, yang disebut sebagai *Public Query Language (PQL)*, yang dimaksudkan untuk basis data model relasi.

Tulisan ini dimaksudkan untuk menjelaskan hasil penelitian yang dilakukan secara khusus untuk prinsip pengolahan pendukung PQL (terutama operator komposisi dan jaringan semantik data). Untuk mendukung penjelasan, digunakan beberapa contoh yang didasarkan pada skema basis data yang terdapat pada Diagram-1. Tulisan ini diawali dengan penjelasan ringkas struktur PQL, yang dimaksudkan untuk lebih mempermudah pemahaman, dengan membandingkannya secara langsung dengan SQL.

2 Sintaks PQL

Bertitik tolak dari sintaks PQL[12], pemakai hanya perlu menyatakan dan mengetahui atribut yang diperlukan, relatif terhadap seluruh atribut yang terdapat pada basis data. Bila diperlukan, juga dapat dinyatakan kondisi dalam bentuk sederhana yang relatif terhadap atribut yang juga terdapat pada basis data.

Struktur dasar PQL adalah sebagai berikut (menggunakan kata kunci **TAMPILKAN-JIKA, DAN** dan **ATAU**) :

TAMPILKAN < atribut-1, atribut-2 ....., atribut-n >
JIKA < predikat > (*) ;

(*) : **<predikat >** berfungsi sebagai kondisi, berupa kondisi sederhana yang dibangun pada tingkat atribut (mis. : KOTA = 'JAKARTA').

Penjabaran lebih lengkap sintaks PQL dan diagram sintaks PQL[12] disertakan pada Diagram 2. Untuk perbandingan, diagram sintaks SQL disertakan pada Diagram 3.

Berikut ini beberapa contoh PQL yang didasarkan pada skema basis data INDUSTRI pada Diagram 4.

SKEMA BASIS DATA "INDUSTRI" (Standard)

KOMODITAS	(KODE_KOMOD#, NAMA_KOMOD, ICGS, CCN)
INDUSTRI	(KODE_IND#, NAMA_IND)
LOKASI	(KODE_LOK#, NAMA_LOK)
USAHA	(KODE_KOMOD#, KODE_LOK#, TAHUN#, JML_USH)
TENAGA	(KODE_KOMOD, KODE_LOK#, TAHUN#, WNI_A#, PENDDKAN#, AKTIFITAS#, GAJI_TOT, BONUS, LEMBUR, JML_PEG)
HASIL_1	(KODE_KOMOD#, KODE_LOK#, TAHUN#, JENIS_PROD#, PASAR#, VOLUME, SATUAN, NILAI_PROD)

Catatan : atribut dengan tanda # berarti atribut kunci.

Diagram 1 : Skema basis data "Industri" (standard)

SINTAKS PQL

Khusus untuk RETRIEVE data :

TAMPILKAN {<atribut>} "| {**JIKA**< eksp>} ;
<eksp> : : = <eksp1> | (< eksp1>) { <oplojbin> <eksp> }"0
| **TIDAK** (<eksp1>) { < oplojbin > < eksp>}"0 ;
<eksp1> : : = < eksp2 > | <eksp2> < oplojbin> < eksp> } ;
<eksp2> : : = < atribut> < oprel> { < atribut>< konstanta> } ;
<oplojbin > : : = **DAN** | **ATAU** ;
<oprel> : : = = | < > | < | > | < = | < = ;
<atribut> : : = < huruf > < huruf> | < angka> | } "0 ;
<konstanta> : : = < string > | < bilangan> ;
<string > : : = { < huruf > | < angka> | < spesial > } "0 ;
<bilangan > : : = { | + }' 0 { < bulat > { < bulat> }' 0 ;
<bulat > : : = { < angka > } "1 ;
<angka> : : = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 ;
<huruf > : : = a | b| c | d| ..... |z | A | B| C |.... Z| ;
<spesial> : : = / | / | * | (|) | ^ | % | $ | # | @ | ! | ? | . | , | " | ` | { | | |
{ | } | | | ` | ~ | : | ; | + | − | _ ;

Diagram 2 : Sintaks PQL

Contoh 1 : Predikat Sederhana

PQL : **TAMPILKAN** kode_komod, jml_peg, gaji_tot
JIKA kode_komod = 'k1' **DAN** kode_lok = 'k2' ;

Contoh 2 : Predikat Sederhana, untuk merepresentasikan predikat dengan struktur blok pada SQL

PQL : **TAMPILKAN** kode_komod, jml_peg, gaji_tot, penddkan
JIKA pasar = 'jepang' ;

SINTAKS SQL

Khusus untuk RETRIEVE data :
SELECT [DISTINCT] < list-atribut >
[INTO <nama-relasi>]
[FROM { <Nama-relasi | < nama-view > }
[, {< Nama-relasi > | < nama-view >}] .....
[WHERE <eksp>]
[GROUP BY [ALL] <atribut> [, <atribut >]
[**HAVING** <eksp>]
[ORDER BY [{ <Nama-relasi> | < nama-view > }]
<atribut> |
<list-nomor > | <eksp> } [**ASC** | **DESC**]
[, { [<nama-relasi> | <nama-view> <atribut> |
<list-nomor> | <eksp> } [**ASC** | **DESC**] } ...] ;

Diagram 3 : Sintaks SQL

SKEMA BASIS DATA "INDUSTRI" (untuk PQL)

DAFTAR ATRIBUT :

KODE_KOMOD	NAMA_KOMOD	ICGS	CCN
KODE_IND	NAMA_IND	KODE_LOK	NAMA_LOK
TAHUN	JML_USH	WNI_A	PENDDKAN
AKTIFITAS	GAJI_TOT	BONUS	LEMBUR
JML_PEG	JENIS_PROD	PASAR	VOLUME
SATUAN	NILAI_PROD		

Diagram 4 : Skema Basis Data "Industri (Untuk PQL)

Contoh 3 : Seperti contoh 2

PQL : **TAMPILKAN** kode_komod, jml_peg, gaji_tot, penddkan
JIKA kode_komod = 'k1' **DAN**
(nama_lok = 'jakarta' **ATAU** nama_lok = 'jatim') ;

Jika dibandingkan dengan SQL, beberapa hal yang tidak perlu ditampilkan pada PQL, yaitu: nama relasi baik untuk atribut yang akan ditampilkan, maupun untuk kondisi, serta struktur blok dalam blok (blok rekursif).

3 Tampilan hasil penelusuran PQL

Dilihat dari sisi hasil penelusuran, PQL menerapkan prinsip entitas, bukan tuppel (seperti yang terjadi pada SQL, QUEL maupun QBE). Dengan demikian, hasil akhir PQL lebih mudah diinterpretasi oleh pemakai.

Sebagai ilustrasi, bahasa penelusuran Generasi ke-IV (SQL) dapat menghasilkan tampilan sebagai berikut :

SQL:	NRP	NAMA	NILAI MATA-KULIAH		DOSEN	
	n-1	ALI	B	basis data	d-1	→ 1 tuppel
	n-1	ALI	B	basis data	d-2	→ 1 tuppel
	n-1	ALI	C	sistem operasi	d-2	→ 1 tuppel

Interprestasi yang tepat untuk contoh di atas, harus didasarkan pada pemahaman terhadap tuppel, sehingga harus dipahami bahwa:

- < n1, ALI > pada baris ke 1, 2 dan 3 merepresentasikan 1 entitas mahasiswa (bukan 3 entitas),
- < basis data > pada baris 1 dan 2 merepresentasikan 1 entitas mata kuliah, dan
- < d-2 > pada baris 2 dan 3 merepresentasikan 1 entitas dosen.

Sesuai dengan prinsip operasi komposisi yang dimaksudkan untuk mendukung eksekusi PQL (akan dijelaskan pada bagian 5), maka hasil yang sama dengan contoh sebelumnya akan disajikan dalam bentuk tampilan sebagai berikut:

PQL:	NRP	NAMA	NI-LAI	MATA-KULIAH	DOSEN	
	n-1	ALI	B	basis data	d-1	→ 1 tuppel
					d-2	
			C	sistem operasi	d-2	

Beberapa bentuk tampilan untuk contoh penelusuran sebelumnya, disajikan berikut ini.

Tampilan 1: tampilan hasil untuk contoh 1 (PQL)

PQL	KODE KOMOD	JML_ PEG	GAJI_ TOT	PEN-DIDIK-AN	
	k 1	356	24 juta	SD	→ 1 tuppel
		400	36 juta	SMP	
		500	57 juta	SMA	
		300	80 juta	S-1	

Tampilan yang sama untuk SQL adalah sebagai berikut :

SQL	KODE KOMOD	JML_ PEG	GAJI_ TOT	PEN DIDIK-AN	
	k 1	356	24 juta	SD	→ 1 tuppel
	k 1	400	36 juta	SMP	→ 1 tuppel
	k 1	500	57 juta	SMA	→ 1 tuppel
	k 1	300	80 juta	S-1	→ 1 tuppel

Interprestasi yang tepat dari tampilan ini akan dicapai, jika pemakai memahami definisi tuppel dan ketergantungan fungsional. Akibat dari definisi tersebut, pemakai harus memahami bahwa 'k1' pada baris ke-1, 2, 3 dan 4 menyatakan 1 entitas, bukan 4 entitas.

Tampilan 2 : tampilan hasil untuk contoh 2 (PQL)

PQL:	KODE_ KOMOD	JML_ PEG	GAJI_ TOT	
	k 1	1556	197 juta	→ 1 tuppel
	k 2	...	.....	→ 1 tuppel
	k 3	...	.....	→ 1 tuppel
		dst.		

Tampilan yang sama untuk SQL adalah sebagai berikut:

SQL:	KODE_ KOMOD	JML_ PEG	GAJI_ TOT	
	k 1	356	24 juta	→ 1 tuppel
	k 1	400	36 juta	→ 1 tuppel
	k 1	500	57 juta	→ 1 tuppel
	k 1	300	80 juta	→ 1 tuppel
	k 2	.....	.....	→ 1 tuppel
		dst		

Hasil tersebut dapat diubah menjadi seperti tampilan 2 (PQL), dengan menggunakan kata kunci SUM (jml_peg, gaji_tot) dan GROUP BY (kode_komod). Contoh lain yang lebih signifikan adalah sebagai berikut:

Penelusuran dan tampilan PQL:

PQL : **TAMPILKAN** kode_komod, jml_peg, gaji_tot, penddkan, nilai_prod **JIKA** kode_komod = 'k1' **DAN** kode_lok = 'k2' ;

PQL :

KODE_ KOMOD	JML_ PEG	GAJI_ TOT	PEN-DIDIK-AN	NILAI_ PROD	
k 1	356	24 juta	SD	950 juta	→ 1 tuppel
	400	36 juta	SMP		
	500	57 juta	SMA		
	300	80 juta	S-1		

Penelusuran dan tampilan pada SQL untuk maksud yang sama dengan contoh di atas:

SQL : **SELECT** AA.kode_komod, AA.jml_peg, AA.gaji_tot, AA.penddkan, BB.nilai_prod
FROM tenaga AA

WHERE AA.kode_komod = 'k1' **AND** AA.kode_lok = 'k2' **AND** AA.kode_komod = (**SELECT** BB.kode_komod **FROM** hasil1 BB **WHERE** AA.tahun = BB.tahun) ;

SQL :

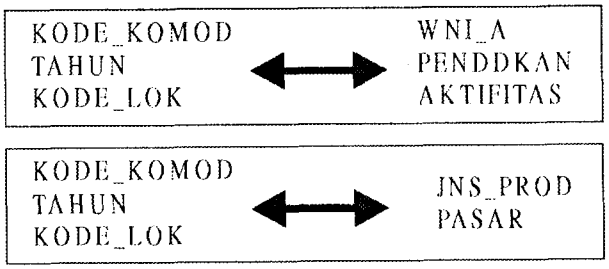
KODE_ KOMOD	JML_ PEG	GAJI_ TOT	PEN-DIDIK-AN	NILAI_ PROD	
k 1	356	24 juta	SD	950 juta	→ 1 tuppel
k 1	400	36 juta	SMP		→ 1 tuppel
k 1	500	57 juta	SMA		→ 1 tuppel
k 1	300	80 juta	S-1		→ 1 tuppel

Sesuai dengan maksud penelusuran, interprestasi hasil di atas adalah sebagai berikut :

> Di lokasi 'k2' terdapat industri yang menghasilkan komoditas (kode komoditas adalah 'k1'), dengan total nilai produksi industri sebesar 950 juta. Pada industri tersebut terdapat pegawai sejumlah :
>
> 356 (gaji total sebesar 24 juta), berpendidikan SD
> 400 (gaji total sebesar 36 juta), berpendidikan SMP
> 500 (gaji total sebesar 57 juta), berpendidikan SMA
> 300 (gaji total sebesar 80 juta), berpendidikan S-1

Untuk contoh yang terakhir ini, yang menjadi masalah dan perlu mendapat perhatian khusus dari pemakai adalah interpretasi dari atribut NILAI_PROD. Pada hasil PQL, NILAI-PROD adalah 950 juta (hanya muncul 1

kali). Sedangkan pada hasil SQL, NILAI_PROD 950 juta muncul pada setiap tuppel (dampak dari operator join, yang mengakibatkan adanya duplikasi pemunculan data). Untuk mendapatkan interpretasi yang benar, maka pemakai harus memahami, bahwa NILAI_PROD pada tampilan hasil SQL tidak mempunyai ketergantungan fungsional (*functional dependency*) dengan PENDIDIKAN, tetapi hanya mempunyai ketergantungan fungsional dengan KODE_KOMOD. Kesalahan interpretasi semantik dari hasil SQL akan timbul, jika dilakukan operasi SUM (JML_PEG, GAJI_TOT dan NILAI_PROD) serta GROUP BY (KODE_KOMOD). Terjadinya duplikasi penampilan pada hasil SQL (seperti pada contoh di atas: KODE_KOMOD dan NILAI_PROD) adalah dampak dari adanya hubungan yang banyak antara entitas yang diwakili atribut kunci. Hal ini timbul, karena hubungan antara entitas yang diwakili atribut kunci tidak didefinisikan pada basis data model relasi serta operator join[1]. Pada contoh terakhir, terdapat hubungan antara kelompok atribut kunci berikut:



4 Operasi komposisi pada PQL

Pada butir 2 dijelaskan prinsip sintaks PQL, serta bentuk tampilan hasil pengolahan yang diharapkan (butir 3). Kedua hal di atas akan dapat dicapai, dengan menerapkan prinsip operasi (operator) komposisi, yang dijelaskan berikut ini.

Pengaksesan basis data untuk mendukung PQL didasarkan pada prinsip komposisi, yaitu memanfaatkan jalur sintaks yang ada pada tingkat data untuk membentuk jaringan semantik data, yang dimaksudkan untuk menggantikan peran operator join [1]. Penerapan prinsip komposisi (operator komposisi) dimaksudkan untuk menghindari adanya kemungkinan duplikasi tampilan data, sebagai akibat hubungan fungsional yang ada antara entitas yang diwakili atribut kunci (lihat contoh terakhir). Penggabungan antara dua relasi pada PQL dilakukan oleh operator komposisi, yang prinsip kerjanya didasarkan pada penelusuran (navigasi) jalur jaringan semantik data yang terkelompok pada semua atribut join.

Operator komposisi yang dimaksudkan pada PQL merupakan penjabaran pembentukan jaringan semantik data (ekuivalen dengan operator join), serta penelusuran (navigasi) semua jalur pada jaringan semantik data yang memenuhi syarat, relatif terhadap jaringan semantik data yang dapat dibangun berdasarkan eksistensi tuppel pada relasi. Penjabaran jaringan semantik data tersebut dimaksudkan untuk menghilangkan definisi tuppel. Sebagai contoh, gambar pada Diagram 6 adalah jaringan semantik data yang ada (khusus untuk data AK1 pada atribut AK), untuk basis data yang dijabarkan pada Diagram 5.

Pada saat eksekusi *query* (PQL), untuk mendapatkan hasil yang akan ditampilkan, aktivitas penelusuran (secara logis) jaringan semantik data (sebagai bagian dari operator komposisi) diterapkan sesuai dengan langkah berikut:

BASIS DATA

Relasi R-1 :	AK#	BK#	C	D
	AK1	BK1	C1	D1
	AK1	BK2	C2	D2
	AK2	BK1	C3	D3
	AK2	BK2	C4	D4
	AK3	BK3	C5	D5

Relasi R-2 :	BK#	XK#	E	F
	AK1	XK1	E1	F1
	AK1	XK2	E2	F2
	AK2	XK1	E3	F3
	AK2	XK2	E4	F4
	AK3	XK3	E5	F5

Relasi R-3 :	XK#	YK#	G	H
	XK1	YK1	G1	H1
	XK1	YK2	G2	H2
	XK2	YK1	G3	H3
	XK2	YK2	G4	H4
	XK3	YK3	G5	H5

Relasi R-4 :	YK#	AK#	I	J
	YK1	AK1	I1	J1
	YK1	AK2	I2	J2
	YK2	AK1	I3	J3
	YK2	AK2	I4	J4
	YK3	AK3	I5	J5

Catatan : atribut dengan tanda # berarti atribut kunci.

Diagram 5 : Basis Data "Industri"

Langkah-1	Identifikasi semua relasi yang harus diaktifkan (atribut yang akan ditampil-kan, serta atribut yang berperan pada predikat). Hal ini dilakukan dengan mencari posisi setiap atribut yang akan ditampilkan dan yang berperan pada predikat.
Langkah-2	Tentukan semua atribut join antara masing-masing relasi (semua atribut join dari setiap relasi dianggap sebagai satu kesatuan).
Langkah-3	Mulai dari salah satu atribut yang diketahui kondisinya (jika terdapat predikat), atau Mulai dari salah satu atribut join yang ada antara dua relasi
Langkah-4	Telusuri jaringan semantik yang menuju ke atribut lain yang dimaksudkan sebagai salah satu atribut untuk hasil akhir.
Langkah-5	Teruskan penelusuran ke atribut lain (jika masih mungkin), atau Kembali ke data sebelumnya untuk menelusuri jalur akses lain (untuk mengulangi langkah-5).
Langkah-6	Ulangi langkah-5, sampai semua atribut yang diminta telah ditelusuri.

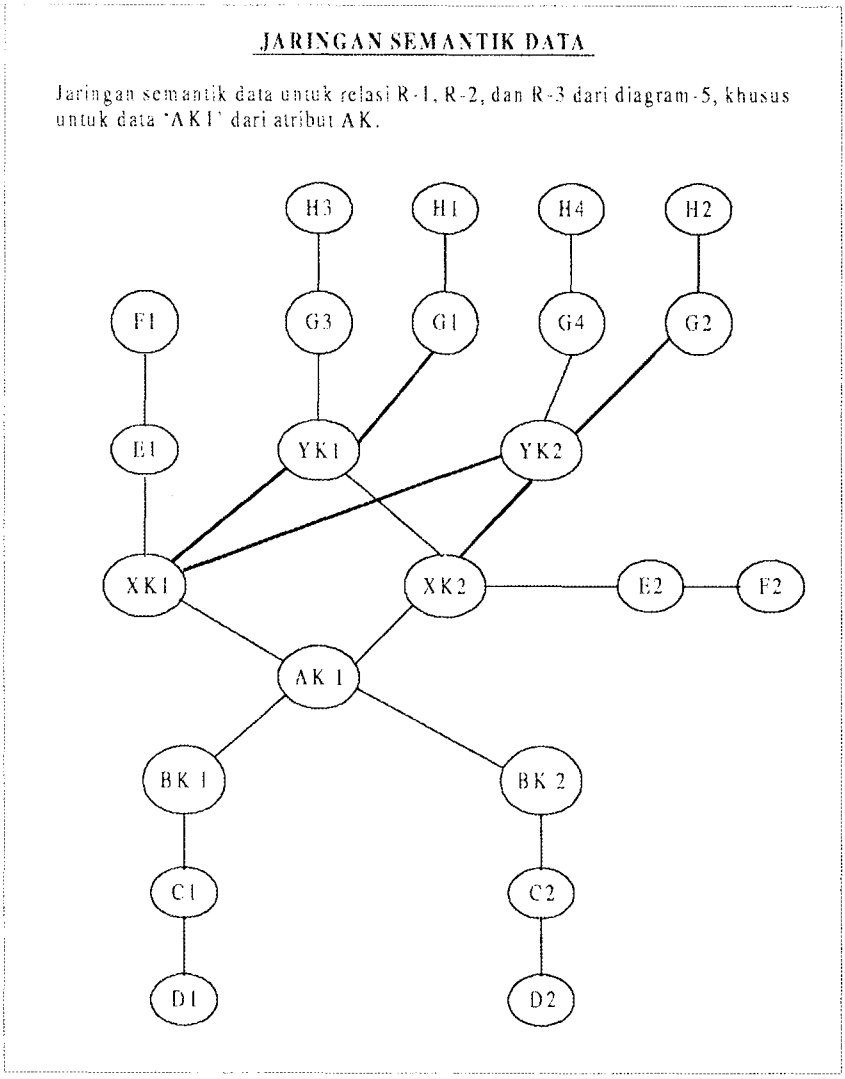


Diagram 6 : Jaringan semantik data

Untuk penelusuran PQL berdasarkan basis data pada Diagram 4 : **TAMPILKAN D, F, H JIKA C = 'C1'**, navigasi dimulai dari 'C1', sehingga jalur lengkap adalah seperti pada Diagram 7.

Catatan :

Data dengan hurup kecil (a,b,c, dst.) adalah data alias, untuk membedakan pengulangan penelusuran.

ATRIBUT	C	AK	XK	F	YK	H
	C1 →	AK 1 a →	D1			
		a →	XK 1 b →	F1		
			b →		YK 1 d →	H1
			b →		YK 2 c →	H2
		a →	XK 2 c →	F2		
			c →		d →	H3
			e →		c →	H4

Diagram 7 : Jalur penelusuran PQL

Dari penelusuran tersebut, akan didapatkan rekapitulasi penelusuran jalur semantik data sebagai berikut :

C	AK	D	XK	F	YK	H	
C1	AK 1	D1	XK1	F1	YK 1	H1	→ 1 tuppel
			XK2	F2	YK2	H2	
						H3	
						H4	

Sehingga hasil akhir adalah (merupakan hasil akhir dari aplikasi operator projeksi untuk D, F, H) :

D	F	H	
D1	F1	H1	→ 1 tuppel
	F2	H2	
		H3	
		H4	

Untuk penelusuran **PQL: TAMPILKAN D, F, H**; penelusuran dapat dimulai secara bebas. Salah satu alternatif yang mungkin, adalah jalur penelusuran pada diagram berikut.

ATRIBUT	AK	BK	D	XK	F	YK	H
	AK 1 a →	BK 1 b →	D1				
	a →			XK 1 d →	F1		
				d →		YK 1 e →	H1
				d →		YK 1 f →	H2
	a →			XK 2 g →	F2		
				g →		e →	H3
				g →		f →	H4
	a →	BK 2 c →	D2				
	AK 2 h →	b →	D3				
	h →						
	h →			d →	F3		
	h →			g →	F4		
	h →	c →	D4				
	AK 3 i →	BK 3 →	D5				
	i →			XK 2 k →	F5		
				k →		YK 3 →	H5

Dari penelusuran tersebut, akan didapatkan rekapitulasi penelusuran jalur semantik data sbb.:

AK	BK	D	XK	F	YK	H	
AK1	BK1	D1	XK1	F1	YK1	H1	
AK2	BK2	D2	XK2	F2	YK2	H2	
AK3	BK3	D3	XK3	F3	YK3	H3	→ 1 tuppel
		D4		F4		H4	
		D5		F5		H5	

Sehingga hasil akhir adalah (merupakan hasil akhir aplikasi operator projeksi untuk D, F, H):

D	F	H	
D1	F1	H1	
D2	F2	H2	
D3	F3	H3	→ 1 tuppel
D4	F4	H4	
D5	F5	H5	

Sebagai bahan perbandingan, navigasi yang dimaksudkan untuk mendukung operator komposisi pada PQL, adalah sama dengan prinsip aplikasi navigasi[10] pada basis data model hirarki. Perbedaan signifikan terdapat pada cara pemetaan jaringan semantik data. Pada model hirarki, struktur pohon didefinisikan pada tingkat *record type*[10], sedangkan pada PQL, struktur pohon didefinisikan pada tingkat primitif basis data, yaitu *field* (pembentukan struktur pohon pada PQL dijelaskan pada bagian berikut).

5 Jaringan semantik data untuk operasi komposisi

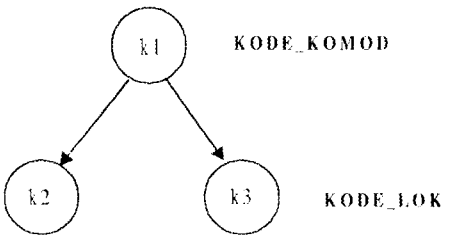
Pembentukan jaringan semantik data (contoh: gambar pada Diagram 6) sebagai pendukung navigasi pada operasi komposisi PQL dimaksudkan untuk menghilangkan duplikasi data (representasi dari entitas yang sama). Pada model relasi, duplikasi kemunculan data yang sama pada beberapa tuppel, adalah pendekatan untuk dapat menyatakan hubungan fungsional satu-banyak atau banyak-banyak yang ada antara entitas yang diwakili oleh atribut kunci. Sebagai contoh (dari skema relasi PEGAWAI pada diagram-1) adalah antara KODE_KOMOD dengan KODE_LOK:

KODE_KOMOD	KODE_LOK
k 1	k 2
k 1	k 3

Data di atas (k1 pada baris 1 dan 2 menyatakan satu enititas) dimaksudkan untuk dapat mereprentasikan adanya industri dengan hasil (kode komoditas k1) yang sama, tetapi pada lokasi yang berbeda (k2 dan k3).

Pada jaringan semantik data, usaha menghilangkan duplikasi data seperti di atas didekati dengan menyusun data dari satu relasi (interpretasi join dalam bentuk lain), sesuai dengan aturan hirarki (*tree structure*), dilihat dari atribut kunci yang dapat berperan sebagai atribut join.

Sebagai ilustrasi, maka data di atas (KODE_KOMOD dengan KODE_LOK) akan tersusun menjadi:



Catatan : kemunculan data 'k1' pada struktur di atas, hanya satu kali.

Pembentukan jaringan semantik data (interpretasi lain dari operator join) dapat dilakukan dengan urutan langkah berikut.

angkah-1	Tentukan skema relasi dimana semua atribut yang akan ditampilkan dan atribut pada predikat (kondisi sederhana) ditemukan.
Langkah-2	Tentukan kumpulan atribut kunci (dapat terdiri dari 1 atau beberapa atribut) maksimum pada setiap skema relasi yang dapat digunakan sebagai atribut join antara semua skema relasi hasil langkah-1(*).
Langkah-3	Tranformasi tuppel dari setiap relasi yang berperan dalam operasi komposisi, dengan menggunakan kumpulan atribut kunci sebagai akar (root)(**)
Langkah-4	Lakukan operasi komposisi antara sesama struktur pohon (hasil langkah-3), dengan memperhatikan kesamaan kumpulan atribut kunci. Hasil dari operasi komposisi relatif terhadap 2 relasi akan menghasilkan satu struktur pohon, yang merupakan penggabungan kedua struktur pohon yang merupakan representasi dari 2 relasi yang dikomposisikan.
Langkah-5	Ulangi langkah-4, sampai semua operator komposisi telah diaplikasikan. Hasil akhir dari pengulangan ini akan merupakan langkah untuk menggabungkan semua struktur pohon menjadi satu kesatuan (***).

keterangan :

(*) : *Hal ini dilakukan tidak terbatas hanya pada skema relasi hasil langkah-1, tetapi pada semua skema relasi yang terdapat pada basis data. Kepentingan pengikutsertaan semua skema relasi, dimaksudkan untuk mendapat-kan jumlah atribut kunci semaksimal mungkin, yang dapat digunakan sebagai atribut join.*

(**) : *Untuk meningkatkan efisiensi, maka struktur pohon hanya dibangun untuk data atribut kunci (root) yang memenuhi kondisi, relatif terhadap operasi komposisi sebelumnya. Dengan proses transformasi ini, maka kumpulan data yang merupakan penjabaran (mapping) dari kumpulan atribut kunci, akan unik.*

(***) : *Strategi penyimpanan struktur pada implementasi langkah ini akan sangat menentukan kinerja (performance) sistem PQL.*

Catatan :

Langkah untuk penelusuran jaringan semantik data (untuk mendapatkan hasil akhir) telah dijelaskan pada bagian 4.

Dengan jaringan semantik data yang dihasilkan dari langkah-langkah tersebut di atas, maka setiap penemuan satu cabang baru pada penelusuran struktur pohon, akan menghasilkan data yang baru. Hal ini terjadi, karena setiap cabang pada struktur pohon akan unik. Jika beberapa struktur pohon dari beberapa penelusuran disatukan (usaha menghilangkan duplikasi data pada tampilan), maka akan didapatkan suatu jaringan data yang membentuk geraf, yang dalam penjelasan sebelumnya disebut sebagai jaringan semantik data (sebagai contoh, lihat gambar pada diagram-6).

6 Uji-coba implementasi

Uji-coba pengembangan dan implementasi algoritma operator komposisi dan pembentukan jaringan semantik data berdasarkan prinsip yang telah dijelaskan pada butir 4 dan 5 telah dilakukan. Berdasarkan uji-coba, algoritma operator komposisi dan algoritma pembentukan jaringan semantik data dapat memberikan hasil yang sesuai dengan harapan, dengan algoritma yang lengkap sebagai penjabaran dari langkah tersebut pada butir 4 dan 5[13].

Permasalahan utama pada uji-coba penggunaan sistem termaksud, adalah pada konfigurasi perangkat keras yang digunakan, khususnya yang menyangkut dengan penggunaan memori. Pada saat pembentukan jaringan semantik data, maka semua data (*field*) yang didefinisikan pada atribut yang perlu diakses, harus dipetakan relatif terhadap data (*field*) lainnya, baik dari atribut yang sama, maupun dari atribut yang berbeda (pemetaan tersebut membentuk satu jaringan semantik data, yang didasarkan pada struktur geraf). Dengan demikian, jaringan semantik data akan terdiri dari satu struktur geraf, dengan jumlah node yang relatif besar, yaitu sama dengan jumlah data (*field*) yang memenuhi syarat. Sebagai ilustrasi, geraf pada diagram-6 adalah satu geraf yang dibentuk berdasarkan basis data pada diagram-5 untuk satu *query* tertentu, dimana geraf tersebut merupakan jaringan semantik data yang dibentuk dari relasi R-1, R-2, dan R-3 (diagram-5), khusus untuk data AK1 dari atribut AK. Jaringan semantik data seperti contoh tersebut harus terlebih dahulu dibentuk secara utuh (membutuhkan memori yang besar, relatif jauh lebih besar dari kebutuhan tuppel), baru kemudian operator komposisi dapat diterapkan, untuk mendapatkan hasil akhir. Selanjutnya, sesuai dengan kondisi atribut AK, maka pembentukan jaringan semantik data dan penerapan operator komposisi harus diulang beberapa kali, sesuai dengan jumlah data yang terdapat pada atribut AK.

Secara prinsip, pendekatan tersebut di atas sangat berbeda dengan prinsip penerapan operator join, dimana pengaksesan selalu dilakukan berdasarkan tuppel. Dengan demikian, kebutuhan memori untuk proses dengan operator komposisi (akses berdasarkan entitas) membutuhkan kapasitas memori yang jauh lebih besar daripada pengaksesan berdasarkan tuppel.

Dari sisi sistem operasi dalam arti fungsional, masalah kapasitas memori tersebut sebenarnya tidak menjadi kendala. Hal ini dapat diatasi dengan berbagai cara, antara lain: *swapping, virtual memory*. Tetapi dari sisi waktu tanggap (*time response*) untuk pemrosesan *query* pada basis data, biasanya pendekatan ini kurang memberikan hasil yang memuaskan. Oleh karena itu, masih diperlukan pengembangan alternatif lain, yang secara spesifik akan sesuai dengan kebutuhan pemrosesan *query* pada basis data.

7 Kesimpulan

Kemudahan interpretasi hasil (tampilan) dimungkinkan oleh penggunaan operator komposisi (kesatuan antara interpretasi operator join dalam bentuk lain, serta navigasi pada jaringan [geraf] data), yang didukung oleh penjabaran data dari setiap relasi menjadi struktur pohon (*tree*) berdasarkan semantik data (disebut sebagai jaringan semantik data).

Pengembangan struktur tampilan hasil PQL didasarkan pada analisis kebutuhan serta berbagai kendala yang dihadapi oleh pemakai awam SQL. Dengan SQL, interpretasi hasil pengolahan hanya akan benar, jika pemakai memahami tuppel dan ketergantungan fungsional. Hal yang terakhir ini, adalah konsiderasi pada struktur tampilan PQL, bagaimana menghilangkan adanya kemungkinan salah interpretasi hasil.

Pada penelitian tersebut, konsiderasi utama yang menjadi pertimbangan adalah, bagaimana menyediakan fasilitas interaksi bagi pemakai (*query*) yang tidak memahami konsep basis data. Dengan adanya sintaks PQL [13] dan prinsip solusi dalam pemrosesan yang dijelaskan dalam makalah ini, kiranya kendala yang ada pada pemakai, dapat diatasi dengan meningkatkan derajat otomatisasi pada sistem basis data.

Salah satu kendala implementasi PQL adalah pada konfigurasi perangkat keras yang digunakan, khususnya memori. Pembentukan jaringan semantik data memerlukan ukuran memori yang relatif besar, sesuai dengan jumlah node pada jaringan semantik data dan jumlah node tersebut akan sama dengan jumlah data (*field*) yang terkait, yang ada dalam basis data.

Usulan perbaikan selanjutnya yang diperlukan, sebagai kelanjutan dari penelitian ini, adalah evaluasi dan pengembangan algoritma pembentukan jaringan semantik data, sehingga dapat meredusir jumlah data (*field*) pada geraf yang mewakili. Alternatif pendekatan yang mungkin, yaitu menerapkan kondisi *query* sedini mungkin (sebelum penerapan operasi komposisi), sehingga data (*field*) yang tidak memenuhi syarat kondisi pada *query* tidak perlu disertakan pada saat pembentukan jaringan semantik data. Alternatif lain yang mungkin dapat diteliti lebih mendalam, yaitu menentukan batas pemilahan jaringan semantik data, sehingga pembentukan jaringan tersebut dapat dilakukan secara bertahap. Dua alternatif ini merupakan arah penelitian lanjutan yang penting dalam topik PQL, sebelum PQL termaksud dapat digunakan secara nyata untuk mendukung pemrosesan *query* pada basis data.

8 Pustaka

1. Aho, AV dkk, The theory of join in relational databases, *ACM Transaction on Database System*, **4**, 3, 297-314, 1979.
2. ANSI/X3/H2, *Comittee on American National Standart Database Language SQL*, (X3H2-84-117), ANSI/X3/SPARC Project 363 D, ACM, New York, 1984.
3. Chamberlin, DD dkk., SEQUEL 2: A unified approach to data definition, manipulation, and control, *IBM Journal Research & Development*, **20**, 6, 560-575, 1976.
4. Codd, EF, A relational model for large shared data banks, *Comm. ACM*, **13**, 6, 337-387, 1970.
5. Codd, EF, Extending the database relational model to capture more meaning, *ACM transaction on database system*, **4**, 4, 397-434, 1979.
6. Codd, EF, *The relational model for database management (version 2)*, Addison-Wesley Publishing Co.Inc., USA, 1990.
7. Date, CJ, and DARWEN Hugh, *Relational Database (writing 1989-1991)*, Addison-Wesley Publishing Co.Inc., USA, 1992.
8. Husni, S & Sitohang, B dkk, *Perancangan basis data interaktif untuk SKDP*, Laporan akhir kerja sama penelitian antara Perumtel dan Lembaga Penelitian-ITB, Lembaga Penelitian – ITB, Bandung, 1988.
9. Ozkaharan, EA, *Database Machine and Database Management*, Pretice Hall, USA, 1986.
10. Stonebaker, M, Rertrospective on a database system, *ACM Transaction on database System*, **5**, 2, 225-240, 1980.
11. Sitohang, B, *Manuskrip PQL*, Laboratorium Basis Data & Sistem Manajemen Informasi. Jurusan Teknik Informatika, FTI-ITB, 1991.
12. Sitohang, B, *PQL: Bahasa penelusuran [Query] basis data (Public Query Language/PQL)*, Jurusan Teknik Informatika, FTI, ITB, September 1995.
13. Rismauli, AV & Tridanarti, A, *Implementasi algoritma Public Query Language*, Laporan Kerja Praktek, Jurusan Teknik Informatika, FTI, ITB, Maret 1993.
14. Ullman, JD, *Database and knowledge–based systems*, Volume I, Computer Science Press, Maryland – USA, 1988.
15. Wachid Nur, *Studi antarmuka PQL*, Tugas Akhir Sarjana, Jurusan Teknik Informatika, FTI-ITB, 1994.
16. Wiederhold, G, *File Organization for database design*, Mc.Graw-Hill, Singapore, 1987.
17. Zloof, MM, Query By Example: a database language, *IBM System Journal*, **16**, 4, 324-343, 1977.