PENYIMPANAN DATA RDF DENGAN MENGGUNAKAN DATABASE RELASIONAL

Heri Kurniawan dan Wahyu C. Wibowo
Fakultas Ilmu Komputer, Universitas Indonesia
e-mail: hek50@ui.edu,wibowo@cs.ui.edu

ABSTRAKSI

Internet mempunyai lebih dari jutaan sumber daya yang tersebar didunia. RDF(Resource Description Framework) sebagai salah satu teknologi semantic web mempunyai peran besar dalam mendeskripsikan alamat sumber daya tersebut dalam dokumen teks yang terstruktur. Seiring dengan berjalannya waktu, jumlah sumber daya semakin hari semakin banyak. Hal ini membuat jumlah dan besar file dokumen RDF juga semakin meningkat. Peningkatan ini berdampak pada menurunnya kinerja proses pencarian dan efisiensi penyimpanan dokumen RDF. Beberapa cara dilakukan untuk menyelesaikan permasalahan tersebut, salah satunya dengan melakukan pemetaaan RDF dengan menggunakan database relasional. Pada paper ini kami mencoba mengusulkan desain skema database yang dapat merepresentasikan informasi dalam RDF. Desain ini diharapkan dapat meningkatkan efisiensi dan efektifitas pencarian dan penyimpanan informasi RDF.

Kata Kunci: *RDF, Semantic Web.*

1. PENDAHULUAN

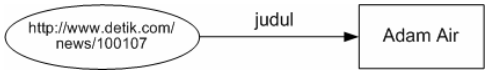
Sejak ditemukannya web pada tahun 1989 [4], banyak kalangan berlomba-lomba untuk mempublikasikan sumber daya yang mereka miliki kepada khalayak dunia. Sumber daya tersebut dapat berbentuk html, teks dan lain-lain yang tersebar dibanyak lokasi. Peningkatan sumber daya memang menguntungkan dari sisi pengguna internet, namun pada sisi lain peningkatan ini menyisakan masalah yang tidak kalah penting yaitu meningkatnya kompleksitas manajemen dan navigasi web. Berdasarkan permasalahan ini, W3C meluncurkan *Semantic Web* dengan salah satu bahasa pendukungnya yaitu RDF (Resource Description Framework)[7]. Semantic web merupakan ekstensi tambahan dari web yang akan membuat informasi menjadi lebih bermakna dan aktifitas pencarian, pertukaran data, dan penggabungan informasi menjadi lebih mudah [2]. Sebagai contoh ketika sebuah *web crawler* mengunjungi situs klinik maka informasi yang didapat bukan hanya sebatas kata kunci "terapi", "obat", "dokter" namun juga informasi lain seperti hubungan antar cabang klinik, penjaga klinik, hari jaga klinik, hingga format tanggal jaga klinik (misalnya dd-mm-yyyy).

Semantic web menggunakan RDF untuk memberi tambahan informasi yang tidak dapat ditemukan melalui kata kunci pencarian biasa. Dengan RDF keterkaitan antara satu artikel dengan artikel lain pada situs lain menjadi mudah didapatkan. Seiring berjalannya waktu, sumber daya internet makin bertambah dan hal ini mengakibatkan jumlah dan besar dokumen RDF ikut bertambah [1]. Pertambahan ini berdampak negatif pada performa kecepatan pencarian sumber daya dan efisiensi penyimpanan dokumen RDF. Berdasarkan permasalahan tersebut, para peneliti mulai melirik kemungkinan informasi RDF dipetakan dengan menggunakan database relasional [5][3]. Pada akhirnya implementasi ide ini membutuhkan desain skema database yang efektif dan efisien yang dapat merepresentasikan seluruh informasi dalam RDF. Pada paper ini kami mencoba mengusulkan skema database RDF beserta implementasinya dengan menggunakan database relasional.

2. RDF

RDF (*Resource Description Framework*) merupakan metadata yang digunakan untuk medeskripsikan alamat sumber daya pada web [7]. Metadata ini dapat berupa judul, pengarang, hak cipta, dan lisensi dalam dokumen web. Elemen pernyataan dalam RDF terdiri dari subjek, predikat dan objek. Subjek adalah sesuatu yang dideskripsikan dan biasanya berisikan alamat URI. Dalam hal ini alamat URI merepresentasikan sumber daya. Predikat merupakan properti dari sumber daya yang menjelaskan hubungan antara subjek dengan objek. Selain itu objek merupakan nilai dari sebuah predikat. Objek mempunyai dua tipe data yaitu objek yang mempunyai tipe URI misalnya "http://www.airplane.com/id/102932" dan objek yang bertipe literal misalnya "adam air". Subjek dan predikat berisikan data yang bertipe sumber daya sedangkan objek dapat bertipe sumber daya dan literal [7].

Dalam RDF, jika kita ingin mengekspresikan kalimat *Judul berita pada link http://www.detik.com/news/100107 adalah adam air* maka untuk menyesuaikan dengan struktur RDF, kalimat bisa kita ubah menjadi *http://www.detik.com/news/100107 mempunyai judul berita adam air*. Dari kalimat tersebut kita dapatkan "http://www.detik.com/news/100107" sebagai subjek, "judul" sebagai predikat dan "adam air" sebagai objeknya. Berikut representasi pernyataan dalam bentuk grafik:



Gambar 1. Graph pernyataan RDF

Bentuk elipse digambarkan sebagai subjek, panah sebagai predikat dan kotak sebagai objek. Contoh implementasi RDF dalam bentuk XML adalah sebagai berikut:

```
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-
rdf-syntax-ns#"
xmlns:berita="http://berita.org/elements/">
<rdf:Description about="
http://www.detik.com/news/100107">
<berita:judul>Adam Air</berita:judul>
</rdf:Description>
</rdf:RDF>
```

Kode 1. Kode RDF

Pada kode 1, RDF menggunakan prefiks rdf dan berita untuk menyingkat alamat penuh URI. Kata rdf merupakan nama namespace yang mengacu pada *syntax* RDF sedangkan berita untuk elemen-elemen berita. Kedua prefiks tersebut dinamakan juga dengan namespace. Dalam namespace, walaupun kita merujuk ke alamat URI sebuah schema, kenyataannya alamat tersebut tidak harus selalu ada. Tidak ada aktifitas validasi dilakukan terhadap alamat tersebut. Hal ini dilakukan lebih untuk menghindari terjadinya penggunaan elemen yang ambigu. Caranya yaitu dengan membuat prefiks yang yang berlainan. Prefiks namespace merupakan bagian dari *XML qualified name* (Qname) yang digunakan untuk memudahkan penulisan RDF [8]. Qname terdiri dari prefiks dan nama lokal. Misalnya teks berita:judul, kata berita merupakan nama pendek dari URI http://berita.org/elements/ sedangkan kata judul adalah elemen dari berita seperti pada link http://berita.org/elements/judul.

3. VARIASI BENTUK RDF

RDF mempunyai bentuk dan kode yang bervariasi. Variasi ini tergantung dari kalimat yang akan dideskripsikan. Variasi ditandai dengan penggunaan tag yang mempunyai fungsi yang berbeda satu sama lain. Berikut adalah beberapa variasi bentuk RDF:

a. RDF Anonim
   Sumber daya anonim tanpa URI digunakan untuk menghubungkan banyak elemen (predikat dan objek) dengan subjek. Elemen objek dan predikat dapat berjumlah lebih dari satu namun saling berkaitan dengan subjek asalnya.

b. RDF Container
   Suatu saat kita ingin mendefinisikan pernyataan tentang sebuah buku yang ditulis oleh beberapa penulis atau software yang dirancang oleh beberapa analis. Hal tersebut dapat dilakukan dengan menggunakan RDF Container. Container merupakan sumber daya yang dapat menampung banyak elemen. Elemen yang tergabung dapat terdiri dari sumber daya, literal ataupun sumber daya anonim (node kosong). RDF Container mempunyai tiga tipe yaitu rdf:bag, rdf:seq dan rdf:alt. Rdf:bag menyimpan banyak nilai tanpa memperhatikan urutan dan duplikasinya. rdf:seq meyimpan nilai secara terurut berdasarkan alfabet tanpa memperhatikan duplikasi nilainya sedangkan rdf:alt menyimpan nilai alternatif dari nilai yang sudah ada, sebagai contoh rdf:alt digunakan untuk mendefinisikan statement alternatif bahasa dari judul sebuah buku.

c. RDF Collection
   RDF Collection merupakan grup yang dapat menyimpan kumpulan sumber daya dalam bentuk list yang terstruktur. List ini dibentuk dengan menggunakan tipe rdf:list, sumber daya rdf:nil dan property rdf:first dan rdf:rest. Berbeda dengan RDF:Container, pada RDF Collection kita mendefinisikan anggota sebuah grup secara lengkap dan spesifik.

d. RDF Reification
   RDF mempunyai kosa kata internal yang digunakan untuk mendeskripsikan pernyataan-pernyataan yang ada pada RDF. Pendeskripsian ini disebut *Reification* [7]. Kosa kata *reification* terdiri dari tipe *rdf:statement* dan properti *rdf:subject*, *rdf:predicate* dan *rdf:object*. Penggunaan *reification* diharapkan dapat memperjelas informasi dari suatu pernyataan.

4. MANFAAT PEMETAAN RDF

Database mempunyai banyak fasilitas yang menguntungkan sebagai media penyimpan informasi bila dibandingkan dengan penyimpanan pada disk. Penyimpanan RDF dalam database akan memudahkan implementasi aplikasi berbasis RDF karena pencarian informasi cukup dengan menggunakan SQL. Database mempunyai sejumlah fasilitas yang dapat menjaga integritas dan efisiensi penyimpanan informasi RDF [6]. Penyimpanan RDF diluar database dapat memberi banyak masalah sebagai contoh dalam masalah kontrol akses file. Kontrol akses pada disk penyimpanan biasanya kurang terdefinisi dengan jelas sehingga realibilitas dan keamanan dokumen berkurang. Hal ini berbeda dengan mekanisme penyimpanan pada database. Pada database hak akses terhadap suatu informasi didefinisikan dengan jelas berdasarkan peran masing-masing user. Selain itu database dilengkapi dengan log akses yang dapat merekam setiap aktifitas akses pengguna.

Berikut ini adalah beberapa manfaat lain dari pemetaan RDF ke database:

a. Kinerja
Pencarian informasi dengan menggunakan SQL lebih cepat bila dibandingkan dengan melakukan parsing terhadap dokumen RDF untuk pengambilan informasi yang sama. Selain itu pencarian melalui SQL akan lebih meningkat jika database mengimplementasikan *query optimizer* dan mekanisme *caching* untuk kueri yang sama.

b. Penyimpanan yang efisien
Dalam metadata RDF terdapat banyak karakter yang muncul lebih dari satu kali yang mengakibatkan efisiensi penyimpanan data berkurang, contohnya seperti pada pernyataan namespace. Pada database, nama namespace cukup ditulis sekali dan selanjutnya cukup dengan menggunakan id namespace bila ada penggunaan namespace yang sama.

c. Pengaturan banyak pernyataan
Penggunaan kueri SQL memudahkan pencarian subset dari pernyataan yang relevan dari total pernyataan dalam RDF. Model RDF menggunakan konsep penggabungan beberapa pernyataan menjadi satu entitas yang dapat di integrasikan kedalam database.

d. Transparansi
Database mempunyai fasilitas untuk membuat VIEW sehingga mengurangi kerumitan dalam kueri SQL. Selain itu, pengembang aplikasi dapat menggunakan *stored procedure* untuk memproses sejumlah kode perintah yang rumit yang tersembunyi dari tampilan pengguna.

e. Keamanan dan Perbaikan
Database menjamin konsistensi transaksi informasi. Sebagai contoh, bila terjadi kesalahan dalam *stored procedure* yang mengakibatkan terhapusnya model RDF pada point tertentu, database dapat melakukan rollback pada semua transaksi hingga fase commit terakhir.

f. Informasi terstruktur pada database
Skema database menyimpan data secara terstruktur sehingga memudahkan pengaturan penyimpanan informasi RDF.

g. Referensi melalui ID
URI dan statement dapat diketahui melalui ID. Hal ini lebih memudahkan karena pengembang aplikasi tidak harus menuliskan teks panjang namespace dan URI dalam pencarian informasi pada SQL.

5. SKEMA

Skema database sangat menentukan performa penyajian informasi. Desain yang baik akan memberikan hasil yang optimal dalam pencarian informasi. Pada paper ini, pembuatan skema didasarkan pada karakteristik RDF seperti pernyataan (subjek, predikat, objek), sumber daya (URI), namespace, literal dan reification. Berdasarkan hal karakteristik tersebut, kami membuat tabel-tabel yang mampu merepresentasikan informasi RDF secara total.

Kami menggunakan field subjek, predikat dan objek pada tabel pernyataan untuk menyimpan seluruh pernyataan (kecuali pernyataan reification) pada RDF. Sebagai langkah efisiensi, beberapa field menggunakan konsep ID yang merujuk ke tabel lain. Metode ini dilakukan untuk menghindari penulisan informasi yang berulang seperti nama predikat, namespace dan alamat sumber daya. Secara umum, skema yang kami bangun terdiri dari 5 tabel yaitu sumberdaya, pernyataan, objekliteral, namespace, predikatref dan reification. Berikut adalah penjelasan dari masing-masing tabel:

a. sumberdaya
Tabel sumberdaya menyimpan semua alamat URI yang ada pada pernyataan RDF. Tabel ini terdiri dari field id, namespace, dan namars. Field id merupakan penanda alamat URI yang bersifat unik. Field namespace menyimpan pointer yang merujuk ke field id dari tabel namespace sedangkan namars menyimpan lokasi alamat URI.

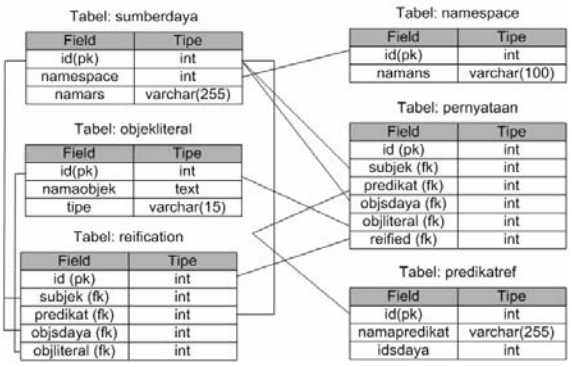
b. pernyataan
Tabel ini menyimpan semua pernyataan subjek, predikat, dan objek yang ada pada RDF. Tabel pernyataan terdiri dari field id, subjek, predikat, objsdaya, objliteral dan reified. Pada tabel ini terdapat dua field objek yaitu objsdaya dan objliteral. Field objsdaya digunakan untuk menyimpan alamat objek yang mempunyai format URI sedangkan field objliteral digunakan untuk menyimpan objek yang mempunyai format literal (teks). Field subjek dan objsdaya merujuk ke field id pada tabel sumberdaya, field predikat merujuk ke tabel id dari predikatref sedangkan objliteral merujuk ke field id pada tabel objekliteral. Field reified digunakan untuk menandakan sejumlah pernyataan yang mempunyai pernyataan *reification*.

c. objekliteral
Tabel ini menyimpan objek yang berbentuk literal. Field-fieldnya terdiri dari id, namaobjek dan tipe. Field namaobjek menyimpan nilai objek sedangkan tipe mendefinisikan tipe data dari nama objek.

d. namespace
Field-field pada tabel ini adalah id dan namans. Field namans menyimpan nama singkat dari alamat URI namespace yang digunakan pada metadata RDF.

e. predikatref
Tabel ini mempunyai field id, namapredikat dan idsdaya. Fungsi utama tabel ini adalah untuk menyimpan prefiks dan nama local (Qname) dari pernyataan RDF (seperti berita:judul) kedalam kolom namapredikat. Setiap nilai pada baris namapredikat merujuk ke tabel sumberdaya melalui field idsdaya. Tabel ini dibuat untuk menghindari penulisan berulang dari kode predikat yang muncul beberapa kali dalam dokumen RDF.

f. reification
Tabel ini menangani pernyataan RDF yang menggunakan *reification*. Struktur tabel ini tidak jauh berbeda dengan tabel pernyataan karena karakteristik pernyataan *reification* sama dengan pernyataan utama.



Gambar 2. Skema

6. IMPLEMENTASI

Pada penelitian ini, pemetaan data RDF kedalam database relasional diimplementasikan dengan menggunakan *Oracle Database Express Edition 10g Release 2* (10.2). Pemetaannya sendiri dilakukan dengan memanfaatkan struktur tabel pada Gb.2. Selain mengimplementasikan kode RDF standar, contoh-contoh kode RDF pada bab ini juga mengimplementasikan variasi bentuk RDF yang mengacu pada konvensi penulisan RDF [7]. Variasi tersebut antara lain RDF Anonim, RDF Container, RDF Collection dan RDF Reification. Standar kueri yang digunakan adalah SQL dan PL/SQL. PL/SQL digunakan khusus untuk pencarian informasi yang berkaitan dengan node kosong (*blank node*) kode RDF. Hal ini dilakukan karena penggunaan SQL biasa belum dapat memfasilitasi pencarian informasi yang cukup kompleks.

Setiap variasi kode RDF mempunyai karakter yang berbeda bila ditempatkan dalam database. Perbedaan ini terletak pada pengkodean datanya. Pengkodean yang mencolok terjadi pada node kosong dari masing-masing variasi. Identitas node kosong untuk setiap variasi RDF harus dibedakan agar tidak terjadi kesalahan pencarian informasi. Kami membuat identitas node kosong dengan format Bnode:zzz.yy.xx, dimana zzz merupakan 3 huruf untuk membedakan kepemilikan node kosong. Misalnya Bnode:nid, node kosong ini merupakan kepunyaan RDF anonim sedangkan Bnode:bag adalah kepunyaan RDF Container. Kemudian id yy pada Bnode:zzz.yy.xx merupakan id pembeda bila dalam satu database terdapat beberapa node kosong dari variasi RDF yang sama. Khusus untuk RDF Anonim dan RDF Container, nilai yy diambil dari id tabel sumberdaya dari baris node kosong tersebut. Sedangkan untuk RDF Collection yy diambil dari id tabel sumberdaya dari node kosong yang pertama. Pembeda terakhir adalah bilangan xx dari Bnode: zzz.yy.xx. Pembeda terakhir ini hanya digunakan pada RDF Collection. Hal ini mengingat RDF Collection mempunyai format kalimat dengan node kosong yang saling bertautan bila dikodekan kedalam database.

Tabel 1. Node kosong dari variasi RDF

Variasi	***ID***
RDF Anonim	Bnode:nid.yy
RDF Container	Bnode:bag.yy Bnode:alt.yy Bnode:seq.yy
RDF Collection	Bnode:sch.yy.xx

Setelah id dari setiap node kosong didefinisikan, maka selanjutnya node kosong tersebut digunakan untuk memverifikasi pencarian melalui PL/SQL. Bagi kode RDF yang tidak menggunakan node kosong misalnya pada contoh RDF Standar, maka pencarian informasi dapat dilakukan cukup dengan menggunakan kueri SQL.

7. KESIMPULAN

Pemetaan RDF menggunakan database relasional memberikan banyak keuntungan bagi pengembangan aplikasi berbasis RDF. Penyimpanan data berbasis database memberikan banyak fasilitas untuk optimalisasi penyajian informasi RDF. Perancangan skema database merupakan langkah awal sebelum penyimpanan RDF dilakukan. Berdasarkan hasil implementasi, skema database yang dirancang untuk mengakomodasi semua variasi RDF tidak dapat mengandalkan kueri SQL biasa untuk melakukan pencarian informasi. Skema ini membutuhkan PL/SQL untuk melakukan proses pengecekan terhadap sebuah nilai dan selanjutnya mengeksekusi serangkaian aksi setelah pengecekan. Hal ini dikarenakan beberapa variasi RDF menggunakan node kosong yang membuat pengkodean dan pencarian informasi menjadi lebih kompleks dan waktu pencarian bertambah.

PUSTAKA

[1] Alexaki, V; et al. The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases, *Proceedings of the Second International Workshop on the Semantic Web (SemWeb2001)*, May 2001. http://citeseer.ist.psu.edu/alexaki01icsforth.html

[2] *An Introduction to RDF*, http://www-128.ibm.com/developerworks/library/w-rdf/

[3] Chong, E.I., Das, S., Eadon,G., and Srinivasan, J., An Efficient SQL-based RDF Querying Scheme, *Proceedings of the 31st VLDB Conference*, Trondheim, Norway, 2005.

[4] Information Management: *A Proposal*, http://www.w3.org/History/1989/proposal.html

[5] K. Wilkinson, C. Sayers, H. Kuno, and D. Reynolds. Efficient RDF storage and retrieval in Jena2. *In Proceedings of VLDB Workshop on Semantic Web and Databases*, pages 131--150, 2003. 7. http://citeseer.ist.psu.edu/wilkinson03efficient.html

[6] *Persisten RDF Database*, http://www.w3.org/1999/07/13-persistant-RDF-DB

[7] RDF Primer, http://www.w3.org/TR/rdf-primer/

[8] RDF Schema, http://www.w3.org/TR/rdf-schema/