KONSTRUKSI SKEMA BASIS DATA VIRTUAL BERBASIS *VIEW* SQL UNTUK MEMBANTU PROSES EKSTRAKSI ONTOLOGI

Choerun Asnawi[1], Kusrini[2], Sudarmawan[3]

Magister Teknik Informatika
Universitas Amikom Yogyakarta

c.asnawi@gmail.com[1], kusrini@amikom.ac.id[2], sudarmawan@amikom.ac.id[3]

Abstrak

Salah satu upaya untuk mewujudkan visi web semantik adalah terkait dengan penyediaan data yang dapat dimanfaatkan oleh teknologi terkait. Salah satu sumber data yang tersedia secara melimpah adalah basis data relasional. Berbagai upaya dilakukan agar dapat mengkonversi data tersebut menuju format yang dipahami oleh teknologi web semantik. Termasuk di dalamnya adalah untuk mempelajari makna yang terkandung di dalam data sumber lalu mengekstraknya dalam bentuk ontologi. Salah satu kendala yang muncul adalah bahwa basis data sumber tidaklah selalu sesuai dengan asumsi awal yang dipakai pada metode ekstraksi ontologi, sehingga hasilnya tak sesuai dengan harapan.

Dalam penelitian ini, diusulkan suatu metode ekstraksi ontologi dari basis data relasional yang tidak mensyaratkan asumsi awal terhadap basis data sumbernya. Pada metode yang diusulkan ini, asumsi justru dibangun dan basis data sumber akan dibuat mengikuti asumsi tersebut, tanpa mengubah struktur yang sudah ada. Caranya adalah dengan mengkonstruksi skema virtual dari view-view khusus yang strukturnya dibuat menyesuaikan dengan asumsi yang ada. Selain itu, dengan metode ini maka view juga dapat dimanfaatkan untuk mengekspos semantik tersembunyi di dalam struktur basis data sumber.

Metode yang diusulkan tersebut mampu menghasilkan output berupa ontologi yang lebih baik dibandingkan dengan metode sejenis lainnya. Selain itu, dengan metode ini juga dapat mengungkap semantik tersembunyi yang gagal diekstrak oleh metode lainnya, tanpa harus memakai bahasa pemetaan baru.

Kata Kunci: *web* semantik, basis data relasional, ontologi, OWL, SQL, *view*.

1. Pendahuluan

Sebagai salah satu representasi masa depan dari *web* yang ada saat ini, visi *web* semantik menjadi salah satu yang diupayakan untuk terwujud. Upaya tersebut tak lepas dari berbagai kendala yang menghalangi, yang salah satunya terkait dengan ketersediaan data yang dapat dimanfaatkan. Memang benar bahwa *web* yang ada saat ini memuat data dan informasi yang amat sangat banyak, namun hanya sedikit saja yang dapat dimanfaatkan secara semantik. Berbagai data yang tersebar di *web* saat ini sebagian besar tersimpan dalam sistem basis data relasional (Malik dkk., 2015). Menurut solid IT gmbh (2018), basis data relasional tetap masih menjadi yang paling populer di antara berbagai jenis *database engine* yang ada. Oleh karenanya, menjadi sangat penting untuk mempelajari cara mengubah data relasional menjadi bentuk yang dipahami oleh teknologi *web* semantik. Selain itu, teknik untuk mengekstrak semantik (makna)

yang terkandung di dalam suatu basis data relasional juga penting untuk dikembangkan. Semantik yang terekstrak kemudian disajikan dalam bentuk ontologi untuk dapat digunakan dalam *web* semantik.

Sebenarnya sudah banyak penelitian dan usulan tentang metode ekstraksi ontologi dari basis data relasional, namun kebanyakan memiliki kelemahan yang sama. Semua metode tersebut menggunakan asumsi awal pada basis data yang dijadikan sebagai sumber, yang akan gagal diterapkan jika asumsi awal tersebut tidak terpenuhi. Contoh asumsi awal yang paling sering digunakan adalah bahwa basis data sumber minimal harus sudah berada dalam bentuk normal ketiga (3NF). Padahal pada kenyataannya, banyak basis data yang sudah mengalami proses denormalisasi untuk pengoptimasian. Asumsi lainnya yang juga sering ditemukan adalah terkait dengan penggunaan skenario struktur dalam skema basis data serta skenario penamaan objek di dalamnya. Selain itu, kebanyakan metode yang ada tidak memperhitungkan adanya ambiguitas yang sering ditemui dalam skema basis data relasional. Dua hal yang memiliki bentuk atau struktur sama di dalam model relasional dapat memiliki makna yang berbeda. Sebagai contoh adalah hubungan *one-to-one*, yang dapat bermakna kepemilikan, pewarisan, atau *vertical partition*.

Dalam penelitian ini, diusulkan suatu metode ekstraksi ontologi dari basis data relasional yang tidak mensyaratkan asumsi awal terhadap basis data sumbernya. Pada metode yang diusulkan ini, asumsi justru dibangun dan basis data sumber akan dibuat mengikuti asumsi tersebut, tanpa mengubah struktur yang sudah ada. Caranya adalah dengan mengkonstruksi skema basis data virtual menggunakan *view-view* khusus yang strukturnya dibuat sesuai dengan asumsi yang ada. Dengan skema virtual tersebut, bentuk normal dapat dibangun, skenario struktur tabel dan penamaan objek dapat diseragamkan dan disesuaikan dengan asumsi yang digunakan. Selain itu, dengan metode ini maka *view* juga dapat dimanfaatkan untuk mengekspos semantik tersembunyi yang ada di dalam struktur basis data sumber.

Makalah ini akan dibagi menjadi beberapa bagian, yang diawali dengan pemaparan tujuan yang akan dicapai, diikuti dengan sedikit peninjauan terhadap beberapa pustaka dan penelitian yang terkait. Berikutnya adalah bagian inti yang diawali dengan pemaparan konvensi penggunaan istilah, diikuti dengan pemaparan tentang teknik yang digunakan beserta contoh-contohnya menggunakan studi kasus. Bagian terakhir berisi kesimpulan dan saran untuk pengembangan teknik yang sudah dipaparkan.

2. Tinjauan Pustaka

Beberapa pelopor penelitian di bidang ekstraksi ontologi dari basis data relasional (RDB) adalah Stojanovic dkk. (2002) dan Astrova (2004). Pada penelitian-penelitian itu diperlihatkan bahwa semantik di dalam RDB dapat diungkap dengan jalan melakukan *database reverse engineering* (DBRE). Dengan DBRE, maka model logis pada RDB dapat dibawa menuju model konseptual yang lebih kaya akan semantik, semacam model Entity-Relationship (ER) atau Enhanced Entity-Relationship (EER). Ontologi sendiri merupakan sebuah model konseptual, sehingga dapat dijadikan target *reverse engineering*.

Gorskis dkk. (2016) menunjukkan bahwa teknik analisis data dapat digunakan selain analisis skema saja, untuk mengungkap keberadaan semantik tambahan. Boumlik dan Bahaj (2016) mengusulkan sebuah pendekatan otomatis yang lengkap untuk membentuk ontologi dari RDB berdasarkan sekelompok aturan yang mampu menangani kekangan (*constraint*) dan jenis hubungan (*relationship*) yang paling rumit. Hazber dkk. (2016) mengusulkan suatu pendekatan baru, yang memungkinkan aplikasi *web* semantik untuk mengakses RDB beserta isinya memakai metode semantik. Abbasi dan Kulathuramaiyer (2016) mengusulkan teknik transformasi untuk membangun ontologi OWL dari RDB dengan mengikuti versi tertulis dari SQL-DDL, lalu ontologi yang dihasilkan akan divalidasi dan ditingkatkan melalui perbandingan dengan EER yang telah dibangun sebelumnya. Sementara itu Sangeeta dan Rao (2017) memilih untuk tidak menggunakan pendekatan DBRE, namun langsung menggunakan ERD yang sudah tersedia. Dari penelitian ini dihasilkan suatu *tool* untuk mengekstrak ontologi dari diagram ER/EER, yang dinamai sebagai Onto Extractor.

Dari sekian penelitian yang dilakukan dan metode yang diusulkan, semuanya berasumsi bahwa basis data *input* sudah berada dalam bentuk normal, minimal 3NF. Ada sedikit penelitian lain yang menyadari hal tersebut dan fokus untuk dapat menangani basis data yang tidak normal. Di antara yang sedikit itu adalah Cui dkk. (2013) dan El Idrissi dkk. (2014). Cui dkk. (2013) fokus untuk mempelajari pendeteksian ketidaknormalan tersebut dan cara menginterpretasikan dan menanganinya. Sedangkan El Idrissi dkk. (2014) lebih fokus untuk mempelajari cara mentransformasi basis data *input* menuju basis data yang sudah dalam bentuk yang normal. Tak satu pun dari penelitian yang diulas di atas yang mempertimbangkan ambiguitas yang mungkin ada pada basis data relasional. Kecuali Sangeeta dan Rao (2017) yang memang murni langsung

menggunakan diagram ER/EER sebagai sumber ekstraksinya, semua penelitian yang lainnya akan menghasilkan ontologi yang keliru jika terjadi salah tafsir, atau jika struktur yang ada tidak sesuai dengan asumsi awal yang digunakan.

Perbedaan penelitian-penelitian yang diulas di atas dengan penelitian yang diusulkan adalah terletak pada dua hal berikut: (1) upaya otomasi proses konversi, serta (2) hal yang dianalisis untuk mengekstrak ontologi. Berbagai penelitian tersebut berupaya agar proses ekstraksi benar-benar dapat dilakukan secara otomatis tanpa campur tangan dari *user*, sementara metode yang digunakan dalam penelitian yang diusulkan ini membutuhkan campur tangan dari *user* untuk melakukan verifikasi sebelum dilakukan proses ekstraksi. Pada metode yang diusulkan, ontologi diekstrak dengan menganalisis struktur *view* dari skema virtual yang dikonstruksi khusus, bukan pada struktur tabel fisiknya.

Dari berbagai literatur yang ditinjau, ditemukan adanya beberapa kesamaan atau kesepakatan dalam penggunaan aturan translasi komponen RDB menuju komponen ontologi. Aturan-aturan yang bersifat umum ini juga akan digunakan pada metode yang diusulkan, seperti terlihat pada **Tabel 1**.

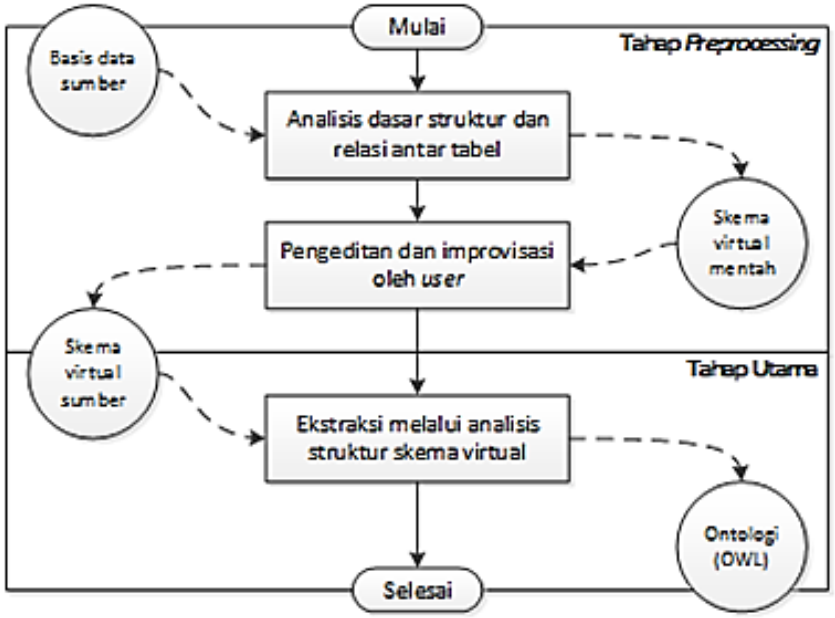
Tabel 1. Aturan-aturan umum translasi RDB ke ontologi

No	Komponen RDB	Hasil Translasi ke Ontologi (OWL)
1	Tabel entitas	Kelas OWL
2	Kolom non kunci tamu	Properti tipe data
3	Kolom kunci tamu	Properti objek
4	Tabel asosiatif atau *junction table*	Properti objek
5	*Junction table* dengan kolom tambahan	Kelas atau restriksi OWL
6	Hubungan *n-ary*	Kelas atau restriksi OWL
7	Baris (*record*)	Individual OWL
8	Domain (tipe data kolom)	Tipe data *XML Schema Definition* (XSD)
9	Tabel asal suatu kolom	Domain dari properti
10	Tabel yang direferensi kunci tamu	Range dari properti objek
11	Kunci utama yang sekaligus menjadi kunci tamu sepenuhnya (hubungan 1:1)	Hubungan pewarisan (*inheritance*) menggunakan `rdfs:subClassOf`

3. Pembahasan

Metode yang diusulkan ini memiliki alur proses yang secara garis besar digambarkan pada **Gambar 1**. Alur proses tersebut terbagi ke dalam dua tahap, yaitu tahap *preprocessing* dan tahap utama. Pada tahap *preprocessing* diperlukan *input* berupa basis data sumber dan dihasilkan *output* berupa skema virtual sumber. Tahap ini berisi dua proses, yaitu: (1) analisis basis data sumber

untuk membangun skema virtual yang masih mentah, serta (2) pengeditan dan improvisasi oleh *user* yang masih harus dilakukan secara manual untuk memperbaiki struktur skema virtual mentah tadi. Dari tahap *preprocessing* ini dihasilkan skema virtual sumber, yang akan digunakan sebagai *input* untuk tahap berikutnya. Pada tahap utama, dilakukan analisis terhadap struktur dari skema virtual, termasuk sintaks dari *view-view* pembentuknya dan menghasilkan *output* berupa ontologi hasil ekstraksi.



Gambar 1. Alur proses dari metode yang diusulkan

Untuk membentuk DDL yang digunakan untuk membangun skema virtual mentah, digunakan aturan dasar yang memiliki pola seperti terlihat pada **Tabel 2**. Secara singkat aturan dasarnya dapat dijelaskan sebagai berikut:

1. *View* yang dibentuk secara khusus untuk membangun skema virtual, untuk selanjutnya disebut sebagai tabel virtual, namanya diambil dari nama tabel asalnya dengan ditambahi awalan titik dua.
2. Kunci utama dari tabel virtual ditandai dengan awalan tanda titik dua juga pada setiap nama kolom penyusunnya.
3. Khusus untuk tabel yang mewakili suatu entitas, yang selanjutnya disebut sebagai tabel entitas, kunci utama pada tabel virtual yang dihasilkan darinya diwakili oleh sebuah kolom khusus yang bernama ":id". Nilainya diambil dari gabungan nilai kolom-kolom kunci utama pada tabel asalnya.
4. Pada tabel virtual yang berasal dari tabel entitas, kolom penyusun kunci utama yang sebenarnya hanya perlu disertakan setelah kolom khusus ":id" jika tabel asalnya tidak menggunakan kunci pengganti.

5. Kunci tamu pada tabel virtual diwakili oleh operasi *join* menuju tabel virtual yang direferensi. Kolom yang digunakan pada posisi kunci tamu ini adalah kolom ":id" dari tabel virtual yang direferensi.
6. Jika karena suatu hal terpaksa memecah sebuah tabel sumber menjadi dua tabel virtual namun masih tetap ingin menyatakannya sebagai satu tabel, kunci utama pada tabel virtual kedua tidak boleh memakai nama ":id" namun harus berupa kunci tamu ke tabel virtual utamanya.

Tabel 2. Pola aturan dasar pembuatan skema virtual mentah

Komponen	Pola Input	Pola Output
Tabel dasar tanpa kunci pengganti	Tabel (PK, Kolom)	CREATE VIEW `:Tabel` (`:id`, PK, Kolom) AS SELECT PK, PK, Kolom FROM Tabel;
Tabel dasar dengan kunci pengganti	Tabel (PK (auto), Kolom)	CREATE VIEW `:Tabel` (`:id`, Kolom) AS SELECT PK, Kolom FROM Tabel;
Tabel entitas dengan kunci komposit	Tabel (PK1, PK2)	CREATE VIEW `:Tabel` (`:id`, PK1, PK2) AS SELECT CONCAT(PK1, PK2), PK1, PK2 FROM Tabel;

Kunci tamu	TabelB (PKB)
	
	TabelA (
	  PKA,
	  FKB → TabelB(PKB)
	)
		CREATE VIEW `:TabelB` (`:id`, PKB) AS
		SELECT PKB, PKB FROM TabelB;
		
		CREATE VIEW `:TabelA` (`:id`, PKA, FKB) AS
		SELECT a.PKA, a.PKA, b.`:id` FROM TabelA a
		  JOIN `:TabelB` b ON a.FKB = b.`:id`

Junction table	TabelA (PKA)
	
	TabelB (PKB)
	
	TabelX (
	  FKA → TabelA(PKA),
	  FKB → TabelB(PKB)
	)
		CREATE VIEW `:TabelA` (`:id`, PKA) AS
		SELECT PKA, PKA FROM TabelA;
		
		CREATE VIEW `:TabelB` (`:id`, PKB) AS
		SELECT PKB, PKB FROM TabelB;
		
		CREATE VIEW `:TabelX` (`:FKA`, `:FKB`) AS
		SELECT a.`:id`, b.`:id` FROM TabelX x
		  JOIN `:TabelA` a ON x.FKA = a.`:id`
		  JOIN `:TabelB` b ON x.FKB = b.`:id`;

Self reference atau *circular reference*	Tabel (
	  PK,
	  FK → Tabel(PK)
	)
		CREATE VIEW `:Tabel` (`:id`, PK) AS
		SELECT PK, PK FROM Tabel;
		
		CREATE VIEW `:Tabel-FK` (`:Tabel`, FK) AS
		SELECT a.`:id`, b.`:id` FROM Tabel x
		  JOIN `:Tabel` a ON x.PK = a.`:id`
		  JOIN `:Tabel` b ON x.FK = b.`:id`;

Setelah skema virtual mentah dihasilkan, proses verifikasi, modifikasi, dan improvisasi dilakukan untuk mengubahnya menjadi skema virtual yang siap untuk proses ekstraksi ontologi. Proses verifikasi dilakukan untuk mengecek ketidaknormalan yang ada pada skema virtual, dengan mencari keberadaan tabel dan kolom sensitif, kolom redundan, grup kolom yang berulang, serta partisi tabel. Proses modifikasi digunakan untuk memperbaiki struktur skema virtual dengan menghilangkan tabel dan kolom sensitif atau redundan, serta melakukan transformasi struktur tabel. Proses improvisasi digunakan untuk mengungkap semantik yang tidak terlihat secara eksplisit pada basis data sumber.

Setelah skema virtual dipersiapkan dan sudah melalui proses verifikasi, modifikasi dan improvisasi oleh *user*, proses selanjutnya adalah mengekstrak ontologi darinya. Aturan-aturan dasarnya adalah sebagai berikut:

1. *Prefix* yang dipakai di ontologi yang dihasilkan adalah: “cls:” untuk kelas OWL, “dp:” untuk properti tipe data, dan “op:” untuk properti objek.
2. Setiap tabel entitas (tabel yang kunci utamanya diwakili oleh kolom “:id”) diekstrak menjadi sebuah kelas OWL.
3. Jika tabel entitas tidak menggunakan kunci utama pengganti, kolom yang diwakili oleh kolom “:id” diekstrak menjadi properti OWL dan dijadikan sebagai kunci dari kelas OWL terkait dengan `owl:hasKey`.
4. Setiap kolom non kunci diekstrak menjadi properti tipe data.
5. Setiap kunci tamu pada tabel entitas diekstrak menjadi properti objek.
6. Setiap *junction table* diekstrak menjadi properti objek.
7. Tabel virtual yang kunci utamanya menjadi kunci tamu secara penuh, jika nama kolomnya adalah “:id” maka diekstrak menjadi kelas OWL yang menjadi subkelas dari kelas OWL hasil ekstraksi tabel yang direferensi. Sedangkan yang nama kolom kunci utamanya bukan “:id” maka dianggap sebagai ekstensi dari tabel yang direferensi dan dianggap sebagai satu kesatuan, sehingga tidak diekstrak menjadi kelas OWL sendiri.
8. *Junction table* yang memiliki tambahan kolom non kunci atau yang mewakili hubungan *n-ary* (lebih dari dua entitas) akan menghasilkan restriksi OWL sejumlah banyaknya kolom dikurangi satu. Kolom pertama bagian dari kunci utama akan mewakili domain dari properti objek yang dihasilkan dari *junction table* tersebut. Kolom lainnya akan digunakan di masing-masing restriksi OWL menggunakan `owl:onProperty` dan `owl:allValuesFrom` yang nantinya digabungkan dalam sebuah kelas anonim yang dipakai sebagai *range* dari properti objek yang dihasilkan.
9. Klausa `WHERE` pada definisi tabel entitas virtual akan menghasilkan restriksi OWL yang menggunakan `owl:onProperty` dan `owl:hasValue`.

4. Studi Kasus

Untuk memperjelas pembahasan serta membuktikan bahwa metode yang diusulkan ini mampu menjawab permasalahan yang diangkat, berikut ini akan dijelaskan alur proses yang ada menggunakan sebuah studi kasus sederhana. Pertama, diberikan dua buah tabel sederhana yang kunci utamanya masing-masing berupa kunci alami dan kunci pengganti. Proses ektraksi ontologinya

dapat dilihat pada **Tabel 3**. Pada kasus ini diperlihatkan proses ekstraksi kelas OWL dari tabel entitas, pemanfaatan `owl:hasKey` untuk menandai properti yang menjadi kunci dari kelas OWL, ekstraksi properti tipe data dari kolom non kunci, serta pengubahan nama tabel dan nama kolom di tabel virtual yang dikonstruksi.

Tabel 3. Proses ekstraksi ontologi dari tabel sederhana

Sumber	buku (isbn, judul) kota (id (auto), nama)
Skema virtual mentah	CREATE VIEW `:buku` (`:id`, isbn, judul) AS SELECT isbn, isbn, judul FROM buku; CREATE VIEW `:kota` (`:id`, nama) AS SELECT id, nama FROM kota;
Skema virtual sumber	CREATE VIEW `:Buku` (`:id`, ISBN, judulBuku) AS SELECT isbn, isbn, judul FROM buku; CREATE VIEW `:Kota` (`:id`, namaKota) AS SELECT id, nama FROM kota;

Ontologi hasil ekstraksi	cls:Buku a owl:Class ; owl:hasKey (dp:ISBN) .
	cls:Kota a owl:Class .
	
	dp:ISBN a owl:DatatypeProperty ;
	  rdfs:domain cls:Buku ; rdfs:range xsd:string .
	dp:judulBuku a owl:DatatypeProperty ;
	  rdfs:domain cls:Buku ; rdfs:range xsd:string .
	dp:namaKota a owl:DatatypeProperty ;
	  rdfs:domain cls:Kota ; rdfs:range xsd:string .

Kasus berikutnya, akan diberikan dua buah tabel yang saling terkoneksi melalui sebuah kunci tamu. Proses ektraksi ontologi darinya dapat dilihat pada **Tabel 4**. Di sini diperlihatkan proses ektraksi properti objek dari kunci tamu, proses penghapusan kolom redundan pada tabel virtual, serta pemanfaatan `owl:unionOf` jika ada nama kolom yang sama di beberapa tabel yang berbeda.

Tabel 4. Proses ekstraksi properti objek dari kunci tamu

Sumber	prodi (kode, nama) mahasiswa (nim, nama, kd_prodi → prodi, nm_prodi)
Skema virtual mentah	CREATE VIEW `:prodi` (`:id`, kode, nama) AS SELECT kode, kode, nama FROM prodi; CREATE VIEW `:mahasiswa` (`:id`, nim, nama, kd_prodi, nm_prodi) AS SELECT a.nim, a.nim, a.nama, b.`:id`, a.nm_prodi FROM mahasiswa a JOIN `:prodi` b ON a.prodi = b.`:id`;
Skema virtual sumber	CREATE VIEW `:Prodi` (`:id`, kodeProdi, nama) AS SELECT kode, kode, nama FROM prodi; CREATE VIEW `:Mahasiswa` (`:id`, NIM, nama, terdaftarDi) AS SELECT a.nim, a.nim, a.nama, b.`:id` FROM mahasiswa a JOIN `:Prodi` b ON a.prodi = b.`:id`;

Ontologi hasil ekstraksi (cuplikan)	cls:Prodi a owl:Class ; owl:hasKey (dp:kodeProdi) .
	cls:Mahasiswa a owl:Class ; owl:hasKey (dp:NIM) .
	
	dp:nama a owl:DatatypeProperty ;
	  rdfs:domain [owl:unionOf (cls:Prodi cls:Mahasiswa)] ;
	  rdfs:range xsd:string .
	
	op:terdaftarDi a owl:ObjectProperty ;
	  rdfs:domain cls:Mahasiswa ; rdfs:range cls:Prodi .

Berikutnya akan diberikan contoh kasus yang memakai *junction table*. Di sini akan diperlihatkan proses ekstraksi properti objek dari *junction table* serta cara memperbaiki struktur *junction table* yang tidak sesuai dengan asumsi awal. Asumsi awal yang banyak digunakan untuk mendeteksi *junction table* adalah bahwa tabel ini memiliki kunci utama berupa *compound key*, yakni terdiri atas lebih dari satu kolom dan setiap kolom penyusunnya merupakan kunci tamu. Jika karena suatu alasan tertentu, diputuskan untuk menggunakan kunci pengganti pada sebuah *junction table*, maka tabel itu tidak akan terdeteksi sebagai *junction table* lagi. Proses ekstraksi ontologi untuk kasus ini diperlihatkan pada **Error! Not a valid bookmark self-reference.**.

Tabel 5. Proses ekstraksi properti objek dari *junction table*

Sumber	dosen (nidn, nama) makul (kode, nama, sks) pengampu (id (auto), nidn → dosen, kd_mk → makul)

Skema virtual mentah	`CREATE VIEW `:dosen` (`:id`, nidn, nama) AS`
	`SELECT nidn, nidn, nama FROM dosen;`
	
	`CREATE VIEW `:makul` (`:id`, kode, nama, sks) AS`
	`SELECT kode, kode, nama, sks FROM makul;`
	
	`CREATE VIEW `:pengampu` (`:id`, nidn, kd_mk) AS`
	`SELECT a.id, b.`:id`, c.`:id` FROM pengampu a`
	`  JOIN `:dosen` b ON a.nidn = b.`:id``
	`  JOIN `:makul` c ON a.kd_mk = c.`:id`;`

Skema virtual sumber	`CREATE VIEW `:Dosen` (`:id`, NIDN, nama) AS`
	`SELECT nidn, nidn, nama FROM dosen;`
	
	`CREATE VIEW `:Matakuliah` (`:id`, kode, nama, SKS) AS`
	`SELECT kode, kode, nama, sks FROM makul;`
	
	`CREATE VIEW `:mengampu` (`:dosen`, `:matakuliah`) AS`
	`SELECT a.`:id`, b.`:id` FROM pengampu x`
	`  JOIN `:Dosen` a ON x.nidn = a.`:id``
	`  JOIN `:Matakuliah` b ON x.kd_mk = b.`:id`;`

Ontologi hasil ekstraksi (cuplikan)	`cls:Dosen a owl:Class ; owl:hasKey (dp:NIDN) .`
	`cls:Matakuliah a owl:Class ; owl:hasKey (dp:kode) .`
	
	`op:mengampu a owl:ObjectProperty ;`
	`  rdfs:domain cls:Dosen ; rdfs:range cls:Matakuliah .`

Pada contoh kasus berikutnya akan diperlihatkan cara mengekspos dan mengekstrak hubungan pewarisan (*inheritance*). Sengaja untuk contoh kasus ini digunakan struktur basis data sumber yang tidak secara eksplisit memperlihatkan adanya hubungan tersebut. Pada contoh tersebut, tabel dosen dan tabel mahasiswa masing-masing mewakili sebuah entitas yang merupakan turunan dari entitas “orang”. Implementasi pada basis data sumber menggunakan teknik spesialisasi, dengan tanpa membuat tabel untuk entitas induknya namun dengan menambahkan atribut-atribut pada entitas induk sebagai kolom pada tabel-tabel turunannya. Selain itu teknik generalisasi dilakukan untuk mengimplementasikan turunan dari entitas dosen, dengan tanpa membuat tabel turunan tapi membuat

kolom “tipe” sebagai kolom pembeda. Prosesnya dapat dilihat pada **Error! Not a valid bookmark self-reference.**.

Tabel 6. Proses ekstraksi hubungan pewarisan

Sumber	dosen (nidn, nama, jk, tipe) mahasiswa (nim, nama, jk)
Skema virtual mentah	CREATE VIEW `:dosen` (`:id`, nidn, nama, jk, tipe) AS SELECT nidn, nidn, nama, jk, tipe FROM dosen; CREATE VIEW `:mahasiswa` (`:id`, nim, nama, jk) AS SELECT nim, nim, nama, jk FROM mahasiswa;

Skema virtual sumber	CREATE VIEW `:Orang` (`:id`, nama, jenisKelamin) AS
	SELECT nim, nama, jk FROM mahasiswa
	UNION SELECT nidn, nama, jk FROM dosen;
	
	CREATE VIEW `:Dosen` (`:id`, NIDN) AS
	SELECT b.`:id`, a.nidn FROM dosen a
	  JOIN `:Orang` b ON a.nidn = b.`:id`;
	
	CREATE VIEW `:Mahasiswa` (`:id`, NIM) AS
	SELECT b.`:id`, a.nim FROM mahasiswa a
	  JOIN `:Orang` b ON a.nim = b.`:id`;
	
	CREATE VIEW `:DosenTetap` (`:id`) AS
	SELECT a.`:id` FROM dosen x
	  JOIN `:Dosen` a ON x.nidn = a.`:id` WHERE x.tipe = 1;
	
	CREATE VIEW `:DosenLuarBiasa` (`:id`) AS
	SELECT a.`:id` FROM dosen x
	  JOIN `:Dosen` a ON x.nidn = a.`:id` WHERE x.tipe = 2;

Ontologi hasil ekstraksi (cuplikan)	cls:Orang a owl:Class .
	
	cls:Dosen a owl:Class ; owl:hasKey (dp:NIDN) ;
	  rdfs:subClassOf cls:Orang .
	cls:Mahasiswa a owl:Class ; owl:hasKey (dp:NIM) ;
	  rdfs:subClassOf cls:Orang .
	
	cls:DosenTetap a owl:Class ; rdfs:subClassOf cls:Dosen .
	cls:DosenLuarBiasa a owl:Class ; rdfs:subClassOf cls:Dosen .

Tabel 7. Proses ekstraksi dari tabel virtual ekstensi

Sumber	dosen (nidn, nama, homebase → prodi) prodi (id (auto), nama, jenjang, kaprodi → dosen)

Skema virtual (tanpa modifikasi)	CREATE VIEW `:dosen` (`:id`, nidn, nama) AS
	SELECT nidn, nidn, nama FROM dosen;
	
	CREATE VIEW `:prodi` (`:id`, nama, jenjang, kaprodi) AS
	SELECT a.id, a.nama, a.jenjang, b.`:id` FROM prodi a
	  JOIN `:dosen` b ON a.kaprodi = b.`:id`;
	
	CREATE VIEW `:dosen-ext` (`:dosen`, homebase) AS
	SELECT b.`:id`, c.`:id` FROM dosen a
	  JOIN `:dosen` b ON a.nidn = b.`:id`
	  JOIN `:prodi` c ON a.homebase = c.`:id`;

Ontologi hasil ekstraksi (cuplikan)	cls:dosen a owl:Class ; owl:hasKey (dp:nidn) .
	cls:prodi a owl:Class .
	
	op:kaprodi a owl:ObjectProperty ;
	  rdfs:domain cls:prodi ; rdfs:range cls:dosen .
	op:homebase a owl:ObjectProperty ;
	  rdfs:domain cls:dosen ; rdfs:range cls:prodi .

Untuk kasus selanjutnya, akan ditunjukkan adanya beberapa variasi hubungan antar tabel virtual yang memakai kunci utama secara penuh sebagai

kunci tamunya, disertai interpretasinya. Varian yang pertama sudah ditunjukkan pada kasus sebelumnya untuk mewakili hubungan pewarisan. Pada varian ini, tabel virtual turunan menggunakan kunci utama bernama ":id" yang sekaligus merupakan kunci tamu yang mengacu pada tabel virtual induknya, serta tabel pertama di bagian klausa FROM-nya merupakan tabel asli (bukan tabel virtual).

Varian kedua adalah yang tabel pengacunya tidak menggunakan nama ":id" sebagai kolom kunci utamanya. Tabel yang semacam ini diinterpretasikan sebagai ekstensi dari tabel virtual yang diacunya. Bentuk ini mirip dengan bentuk partisi vertikal, dalam artian bahwa kedua tabel itu sebenarnya merupakan satu kesatuan yang dipecah karena suatu hal. Salah satu alasan untuk memecah tabel virtual ini adalah karena adanya *circular reference* atau *self reference*. **Tabel 7** memperlihatkan proses ekstraksi ontologi dari contoh kasus ini.

Varian ketiga adalah yang kedua tabel menggunakan kunci utama berupa kolom bernama ":id", tetapi beda dengan varian pertama, pada varian ini tabel virtual pengacu tidak menggunakan operasi *join* dan pada bagian FROM hanya terdapat satu tabel saja yaitu tabel virtual yang diacunya. Varian ini tidak akan dijumpai pada skema virtual mentah hasil analisis basis data sumber, karena murni merupakan improvisasi dari *user*. Tujuan pembuatan tabel virtual dengan cara ini adalah untuk mengeksplisitkan semantik yang tidak akan terlihat dengan hanya melalui analisis struktur dan data pada basis data sumber. Salah satu penggunaannya adalah untuk mengekstrak bentuk khusus dari suatu kelas OWL, dengan memanfaatkan *filtering* sederhana, seperti terlihat pada **Error! Not a valid bookmark self-reference.**.

Tabel 8. Proses ekstraksi kelas tambahan dengan proses *filtering*

Sumber	`mahasiswa (nim, nama, jk)`
Skema virtual mentah	`CREATE VIEW `:mahasiswa` (`:id`, nim, nama, jk) AS` `SELECT nim, nim, nama, jk FROM mahasiswa;`

Skema virtual sumber	`CREATE VIEW `:Mahasiswa` (`:id`, NIM, nama, jenisKelamin) AS`
	`SELECT nim, nim, nama, jk FROM mahasiswa;`
	`CREATE VIEW `:Mahasiswi` (`:id`) AS`
	`SELECT `:id` FROM `:Mahasiswa` WHERE jenisKelamin = 'P';`

Ontologi hasil ekstraksi (cuplikan)	`cls:Mahasiswa a owl:Class ; owl:hasKey (dp:NIM) .`
	`_:x1 a owl:Restriction ;`
	`owl:onProperty dp.jenisKelamin ; owl:hasValue "P" .`
	`_:x2 a owl:Class ; owl:intersectionOf (cls:Mahasiswa _:x1) .`
	`cls:Mahasiswi a owl:Class ; owl:equivalentClass _:x2.`

Berkaitan dengan struktur basis data yang tidak normal, kasus berikutnya akan memuat tabel yang memiliki grup kolom berulang. Di sini akan diperlihatkan

cara mengatasi hal tersebut dengan membangun struktur tabel virtual yang lebih normal dari aslinya. Caranya adalah dengan memecah tabel menjadi dua tabel virtual yang saling terhubung, dengan tabel virtual kedua menggunakan operasi *union* untuk memperoleh data dari setiap kolom dalam grup kolom berulang. Hasil ekstraksi akhirnya berupa satu properti objek atau satu properti tipe data, tergantung dari status kolom yang diulang tersebut (kunci tamu atau bukan).

Di sini akan diperlihatkan dua kasus sekaligus, yang pertama berupa kolom non-kunci dan yang kedua berupa kolom kunci tamu. Selain itu akan diperlihatkan juga bahwa dengan memakai tabel virtual, dimungkinkan untuk mengekstrak satu atau lebih kolom tertentu dan membuat tabel entitas virtual darinya. Dengan teknik tersebut maka sebuah kolom non-kunci dapat diubah menjadi kolom kunci tamu dengan mengacu pada tabel virtual yang diekstrak dari kolom tersebut. Teknik ini dapat diterapkan pada kolom yang berisi nilai enumerasi atau kolom yang berupa hasil penggabungan tabel referensi dari suatu proses denormalisasi. Proses lengkapnya dapat dilihat pada **Error! Not a valid bookmark self-reference.**.

Tabel 9. Proses perbaikan struktur tabel yang memuat grup kolom berulang

Sumber	`pendaftar (nisn, nama, nilai1, nilai2, nilai3, pil1, pil2)`
Skema virtual mentah	`CREATE VIEW `:pendaftar`   (`:id`, nisn, nama, nilai1, nilai2, nilai3, pil1, pil2) AS SELECT nisn, nisn, nama, nilai1, nilai2, nilai3, pil1, pil2   FROM pendaftar;`
Skema virtual sumber	`CREATE VIEW `:Pendaftar` (`:id`, NISN, nama) AS SELECT nisn, nisn, nama FROM pendaftar;` `CREATE VIEW `:Jurusan` (`:id`, namaJurusan) AS SELECT DISTINCT pil1, pil1 FROM pendaftar UNION DISTINCT SELECT DISTINCT pil2, pil2 FROM pendaftar;` `CREATE VIEW `:Pendaftar-nilai` (`:pendaftar`, nilai) AS SELECT b.`:id`, a.nilai1 FROM pendaftar a   JOIN `:Pendaftar` b ON a.nisn = b.`:id` UNION ALL SELECT b.`:id`, a.nilai2 FROM pendaftar a   JOIN `:Pendaftar` b ON a.nisn = b.`:id` UNION ALL SELECT b.`:id`, a.nilai3 FROM pendaftar a   JOIN `:Pendaftar` b ON a.nisn = b.`:id`;` `CREATE VIEW `:memilihJurusan` (`:pendaftar`, `:jurusan`) AS SELECT a.`:id`, b.`:id` FROM pendaftar x   JOIN `:Pendaftar` a ON x.nisn = a.`:id`   JOIN `:Jurusan` b ON x.pil1 = b.`:id` UNION ALL a.`:id`, b.`:id` FROM pendaftar x   JOIN `:Pendaftar` a ON x.nisn = a.`:id`   JOIN `:Jurusan` b ON x.pil2 = b.`:id`;`
Ontologi hasil ekstraksi (cuplikan)	`cls:Pendaftar a owl:Class ; owl:hasKey (dp:NISN) . cls:Jurusan a owl:Class ; owl:hasKey (dp:namaJurusan) .` `dp:nilai a owl:DatatypeProperty ;   rdfs:domain cls:Pendaftar ; rdfs:range xsd:decimal .` `op:memilihJurusan a owl:ObjectProperty ;   rdfs:domain cls:Pendaftar ; rdfs:range cls:Jurusan .`

5. Penutup

Dari pembahasan dan studi kasus yang telah disajikan, terlihat bahwa metode yang diusulkan ini mampu mengatasi permasalahan yang diangkat pada penelitian ini. Konstruksi *view* khusus terbukti dapat digunakan untuk membangun sebuah skema basis data virtual yang disesuaikan dengan asumsi awal yang digunakan untuk ekstraksi ontologi. Selain itu, konstruksi *view* khusus terbukti juga mampu memecahkan permasalahan terkait dengan kemenduaan pada struktur dan hubungan antar tabel. Terakhir, konstruksi *view* khusus terbukti dapat mengungkap semantik yang semula tersembunyi karena memang bagian dari kekurangan model relasional, atau karena ketidaksesuaian asumsi awal dengan rancangan basis data yang dipakai.

Beberapa hal yang masih dirasa kurang dalam penelitian ini dapat dijadikan bahan untuk penelitian lanjutan. Proses pembentukan DDL untuk pembangunan skema virtual pada penelitian ini sifatnya masih sangat dasar dan belum memakai analisis yang mendalam. Hal itu membuat DDL yang dihasilkan masih sangat sederhana, sehingga apabila basis data sumber banyak memuat kondisi yang tidak sesuai asumsi maka akan membutuhkan upaya yang terlalu besar saat pengeditan dan improvisasi oleh *user*. Sebuah penelitian dapat dilakukan untuk melakukan analisis yang lebih mendalam sehingga DDL yang dihasilkan lebih lengkap dan membuat proses verifikasi oleh *user* menjadi lebih ringan. Selain itu, proses verifikasi dan modifikasi isi DDL yang dihasilkan masih bersifat manual dan tekstual tanpa visualisasi grafis. Kesalahan sintaks pada DDL hasil pengeditan masih sering terjadi. *Tool* berbasis GUI dapat dikembangkan untuk hal itu agar lebih memudahkan *user* dalam melakukan verifikasi dan modifikasi yang diperlukan.

Daftar Pustaka

Abbasi, A. A. dan Kulathuramaiyer, N., 2016, A Systematic Mapping Study of Database Resources to Ontology via Reverse Engineering, *Asian Journal of Information Technology*, Vol. 15 No. 4, pp. 730-737.

Abiteboul, S., Hull, R., dan Vianu, V., 1995, *Foundations of Databases: The Logical Level*, Addison-Wesley Longman Publishing, Reading, Massachusetts.

Al Khuzayem, L. dan McBrien, P., 2016, Extracting OWL Ontologies from Relational Databases Using Data Analysis and Machine Learning, *Databases and Information Systems IX*, pp. 43-56.

Astrova, I., 2004, Reverse Engineering of Relational Databases to Ontologies, In: Bussler C.J., Davies J., Fensel D., Studer R. (eds) *The Semantic Web: Research and Applications*, ESWS 2004, Lecture Notes in Computer Science, vol 3053. Springer, Berlin, Heidelberg, pp. 327-341.

Boumlik, A. dan Bahaj, M., 2016, Advanced Set of Rules to Generate Ontology from Relational Database, *Journal of Software* Vol. 11 No. 1, pp. 27-43.

Codd, E. F., 1970, A Relational Model of Data for Large Shared Data Banks, *Communications of the ACM*, Vol. 13 No. 6, pp. 377-387.

Cui, S., Li, Y., dan Wang, Y., 2013, Implementation of Ontology Extraction Oriented to Non-normalized Database, *Advanced Materials Research*, Vols. 756-759, pp. 1489-1493.

El Idrissi, B., Baïna, S., dan Baïna, K., 2014, A Methodology to Prepare Real-World and Large Databases to Ontology Learning, *Proceedings of the I-EISA Conferences 7*, pp. 175-185.

Geroimenko, V., 2004, *Dictionary of XML Technologies and the Semantic Web*, Springer-Verlag London Ltd.

Gorskis, H., Aleksejeva, L., dan Polaka, I., 2016, Database Analysis for Ontology Learning, *12th International Conference on Application of Fuzzy Systems and Soft Computing (ICAFS 2016)*, pp. 113-120.

Harrington, J.L., 2016, *Relational Database Design and Implementation*, 4th Edition, Morgan Kaufmann, Cambridge, USA.

Hazber, M. A. G., Li, R., Gu, X., dan Xu, G., 2016, Integration Mapping Rules: Transforming Relational Database to Semantic Web Ontology, *Applied Mathematics & Information Sciences*, Vol. 10 No. 3, pp. 1-21.

Hoffer, J., Venkataraman, R., dan Topi, H., 2016, *Modern Database Management*, Global Edition, 12th Edition, Pearson Education Limited, Essex, England.

Malik, K. R., Ahmad, T., dan Iqbal, M. M., 2015, Data Mapping for Transformation from RDB Schema to RDF Schema, *Technical Journal, University of Engineering and Technology (UET) Taxila*, Pakistan, Vol. 20 No. 3, pp. 99-105.

solid IT gmbh, 30 Agustus 2018, DB-Engines Ranking, *https://db-engines.com/en/ranking*.

Stojanovic, N., Stojanovik, L., dan Volz, R., 2002, A Reverse Engineering Approach for Migrating Data-Intensive Web Sites to the Semantic Web, In: Musen M.A., Neumann B., Studer R. (eds) *Intelligent Information Processing, IFIP – The International Federation for Information Processing*, vol 93. Springer, Boston, pp. 141-154.