Vol.2018-CH-117 No.8
2018/5/12

グラフデータベースによる 文書リポジトリ統合管理システムの設計

王　一凡[1,2]　永﨑　研宣[2]　下田　正弘[3]

概要：筆者らが近年参画している国際規格策定会議では、関連文書が公開リポジトリで一括管理されているが、膨大な文書がそれぞれ個別に登録されているため文書間の関係をたどることが難しい。また、毎回の審議で大量の対象を処理するため議事録は簡素で誤脱も多くならざるを得ず、議論の過程や履歴の理解は会議参加者の記憶によって補われている面が大きい。こうした中、審議の都度各参加者が行う予備調査にかかる負担は大きく、議論の正確さに及ぼす影響が懸念される。そこで、これら文書の関係や更新履歴を集約し、簡便に縦覧するために、文書やその内容をグラフデータベースである Neo4j 内で表現し、その入力・検索を行うシステムを開発した。特に会議の主題である漢字の審議については、詳細な情報が確認・記録可能なデータ構造を構築した。いわゆるグラフデータベースの特徴である（ソフトウェアによって形式が異なるものの）ノードとその関係のみに還元されるスキーマレスな構造は、データにアドホックな要請が多いこのようなデータベースの設計と管理に大変親和的であり、人文科学における概念構造をモデル化するうえで幅広い応用可能性が期待されることを、若干の知見とともに紹介する。

Design of Document Repository Management System Based on Graph Database

WANG YIFAN[1,2]　NAGASAKI KIYONORI[2]　SHIMODA MASAHIRO[3]

1. はじめに

筆者らSATテキストデータベースプロジェクト[*1]（代表：下田正弘）は仏典外字のUCS符号化を目指して、ISO/IEC JTC1/SC2/WG2 の漢字を担当する下部組織である Ideographic Rapporteur Group (IRG) に参加している。

IRG の主要な活動の一つは新規提案字の審査である [1] が、近年は各参加組織が提出した計数千文字のセットを 1 単位として行われている。通常、会議のたびに参加者が分担してこれらを調査し、次回会議で結果を持ち寄り符号化の可否やデータの正誤を議論することを繰り返す。この際、個々の参加組織に割り当てられる字数は近時の平均で1000 字を越える。

IRG で審議される文書群は公開リポジトリ[*2]で保管されているが、各文書は時系列順に配列されているのみで、相互の関連性をリポジトリ上で把握するのは必ずしも容易でない。また、毎回の会議の結果を基に議事録や更新された作業文書が公開されるが、大量の字を短期間で担当するため、記録が必ずしも正確かつ完全ではなく、編集時のミスが紛れこむことも少なくない。そのため、実際の議論の経過を理解するには、出席者としての記憶に頼るか、過去の文書を洗い直して補うしかなく、加えて現状ではその文書の内容の比較も目視が主で煩瑣であり、透明性の確保が困難である。

このような状況下で個々の字の調査者への負担は過大なものとなり、結果として見落としなどにより提案字の審査が不正確になることが懸念される。実際に、過去に収録した字の重複や文字情報の誤りはたびたび指摘されてい

1 東京大学大学院教育学研究科
Graduate School of Education, University of Tokyo
2 人文情報学研究所
International Institute for Digital Humanities
3 東京大学大学院人文社会系研究科
Graduate School of Humanities and Sociology, University of Tokyo

*1 http://21dzk.l.u-tokyo.ac.jp/SAT/

*2 http://appsrv.cse.cuhk.edu.hk/~irg/

る [2]。

この問題を踏まえ、筆者らは文書の内容をデータベース化し、特定の対象に対する議論履歴や関連する文書・内容を簡便に検索・参照・編集できるシステムの開発を試みた。

2. 設計および構成

2.1 目的

本システムの目標は IRG の公開する既存形式のデータを読み込み分析し、それらを IRG の実際のワークフローに即した形で整理しユーザーに提示すること、またユーザーからの同様の形式に沿ったデータの新規入力を可能とすることとした。

まず、IRG の概念モデル構築のためには関与する多様なエンティティ（文字・組織・文書・参加者など）とその間の複雑な関連をモデル化しなければならない。さらにそれぞれのエンティティごとに独自の情報が必要であったり、個別に考慮すべき事項があったりすることが少なくない。

これに加え、IRG では同一世代の字であっても議論の進展に伴って文字情報モデルが更新されたり [3]、ワークフローに変更が加わることが時折ある。とりわけ、IRG は 30 年近く活動しているグループであるのに対し、筆者らは近年参加し始めたに過ぎないため、過去のデータを十分理解しているわけではない。過去のデータを移行するにあたって想定しない構造が出現する可能性が高いと考えられる。したがってデータモデルにもその時々のモデルに対応できる柔軟性が必要とされる。

以上の要因を踏まえ、データの格納形式には局所的な関係をカスタマイズしやすいグラフ型モデルを採用するのが適しているのではないかと判断した。

なお、同様に漢字情報を扱う CHISE プロジェクト[*3]においても、実体となる Lisp データで記述された文字間の関係や文字オブジェクトの構造は非常にグラフ的である [4] ということも特筆しておきたい。

機能面については、画面上で情報を表示する際に一覧性が高く、時系列が把握しやすいことが求められる。また、特に文字情報を編集する際は入力がすぐに反映されることに加え、簡便に入力できるよう定型的な入力を補助する機能が必要と考えた。なぜならば、議事のスケジュールが密な IRG 会議中に可能な限りリアルタイムで情報を入力するためにも本システムを利用したいからである。

2.2 データベースの仕様

現在利用できるグラフデータベースの実装にはさまざまなものがあるが、本研究では検討の結果 Neo4j 3.x 系[*4]を使用することにした。理由として、

- 単純なグラフ構造を格納するソフトウェア（RDF ストアなど）に比べ、ノードや辺内部にプロパティを保持でき、型ラベルを付与できる [5]。これは純粋なグラフというより Key-Value ストアを組み込んだものに相当するが、位置付けが未確定な情報を保持可能であるなど、実用上の利点があると判断した。
- 必ずしも先進的な設計ではないが、仕様が成熟しておりライブラリやユーザーベースが充実しているため、比較的小規模なシステムの開発に際して不必要な困難が少ないと想像される。
- データベース本体が独立したディレクトリとなっており、SQLite のようにポータブルである。インターネット接続が必ずしも保証されない現地での会議中に使用するために環境を移す必要性が想定されるため、都合が良い。
- ACID 準拠のトランザクションが利用できる [5]。既存文字情報の一括インポート時には数千〜数万のノードを挿入することになるため、高速なクエリを安全に発行できる機能は極めて重要である。

執筆時現在では**図 1** のようなデータの関係モデルを構築している。現時点では主に文字情報の整理に注力しているが、文書には他に多様な情報が含まれており、今後の拡張によってそれらも何らかの形で取り込めるよう検討したい。

2.3 アプリケーションの仕様

ユーザーとの入出力を処理するプログラムは Ruby 2.4 系[*5]で動作する Sinatra 2[*6]フレームワークを用いて記述した、ウェブアプリケーションである。執筆時現在は特定少数の環境での動作のみ考慮しているため、フロントエンドには比較的対応ブラウザが新しい Spectre[*7] (CSS) と Zepto[*8] (JS) フレームワークを使用してページ記述を簡潔に留めている。

また、データベースとの通信には直接アクセスの代わりに、Neo4j の Ruby 用ライブラリである Neo4j.rb[*9]を使用し、Neo4j の入出力を Ruby オブジェクトにラッピングするようにした。このライブラリはスキーマレスな Neo4j に擬似的なスキーマ定義を提供しており、バリデーションや日時型といったいくつかの利便性機能を利用することができた。前述のようにスキーマレスであることがグラフデータベースの利点とはいえ、確定したデータをオブジェクト的に取り扱えることはアプリケーション的には有用である。このライブラリの「スキーマ」はインデックスや制約を除き Ruby 側が担保するため、形式の変更に追随するには定義の書き換えのみで済み、高い柔軟性は依然として維持される。

[*3] http://www.chise.org/
[*4] https://neo4j.com/
[*5] https://www.ruby-lang.org/ja/
[*6] http://sinatrarb.com/
[*7] https://picturepan2.github.io/spectre/
[*8] http://zeptojs.com/
[*9] http://neo4jrb.io/

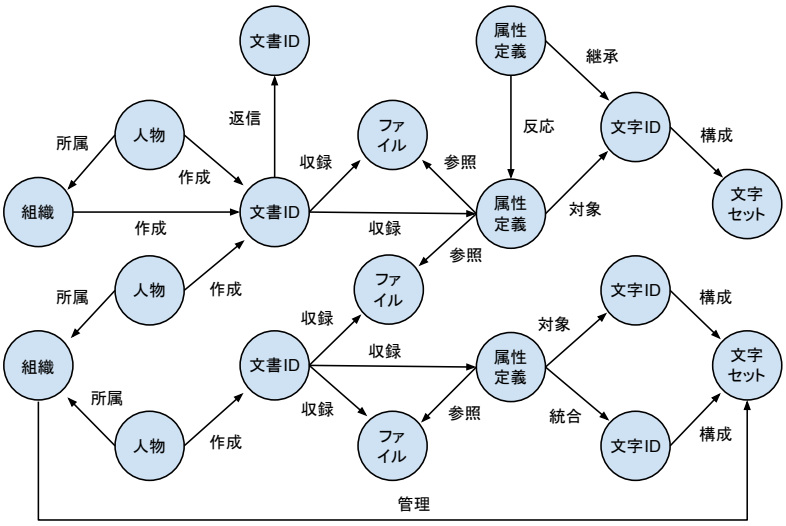

図 1　データモデルの概要

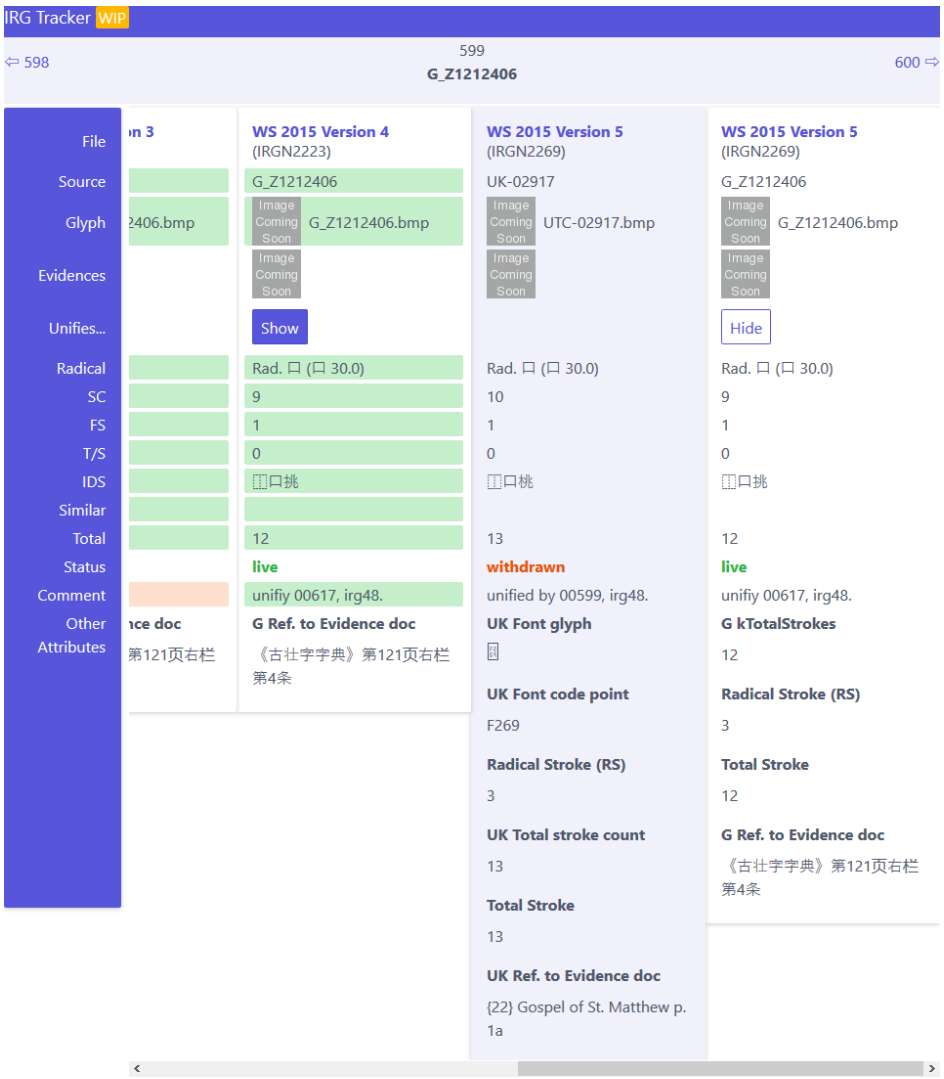

図 2　アプリケーションの文字情報表示画面（開発中）

アプリケーションの主要機能として、ある文字番号に対する議論記録（文書毎）の一覧表示および各記録の編集と新規追加、ある文書に含まれる文字関連情報の一覧表示、新規文書の定義を画面上で行うことができる（図 **2**）。UI に表示される情報は想定されるワークフローでの必要十分性を考慮した。表示機能においては一覧性を重視して記録をコンパクトに配列し、変更箇所をハイライトすることで目視での確認作業を補助している。編集機能は AJAX で最低限の通信のみを発行するようにして、できるだけ会議中の同時記録にも耐えうるよう応答性を重視している。

2.4 補助システム

Excel 文書やテキストファイル形式などで公開されている既存の大規模データを、アプリケーションをバイパスして一括インポートするための Ruby スクリプトを作成した。原ファイルに内容・形式面の誤りが存在することを前提としているため、テストランでのエラー検出機能を充実させた。また処理の効率とバリデーションの正確性を勘案し、Neo4j.rb のマッパーを利用しながらできるだけ Neo4j の提供するトランザクション機能を活用することで、1000 文字（分に相当するノードと関係群）あたり約 60〜70 秒程度で挿入できるようになっている。

3. 運用にあたって

3.1 ライブラリについて

今回使用した Neo4j の Ruby ライブラリはコミュニティプロジェクトであり、仕様とドキュメントが合致していなかったり、ウェブ上の情報に古いものが混在していたりして、正しい記述を特定するのに労力を要した場面がいくつかあった。Neo4j 自体はドキュメントも最新情報も充実していたが、やはり古い無効な情報が残存しているのは避けがたく、留意すべき点と思われた。

3.2 Reification

Neo4j では辺にもプロパティを付加することができるため、関係に追加情報に与えたい時も多くの場合はプロパティで事足りる [6]。しかし、ある関係に対して二つの頂点だけではない他のノードを介在させたい場面が生じた（図 **3**）ので、できるだけ検索効率を犠牲にせずに構造を記述するため、関係をノード化して対応することにした（図 **4**）。

この中間ノードはネイティブな関係としては扱われなくなり、クエリ上も特別な配慮が必要になる。どのような手法が適しているか今後も検討が必要である。

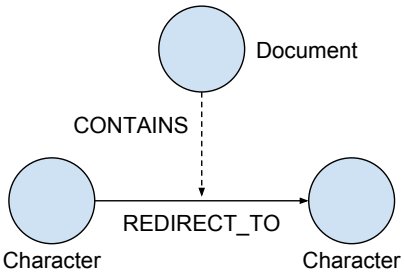

図 3　関係への関係づけ

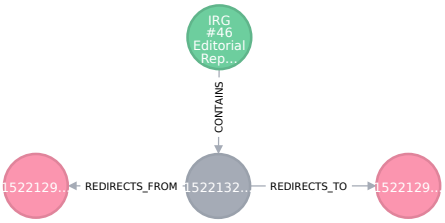

図 4　Neo4j 上の図 3 の表現

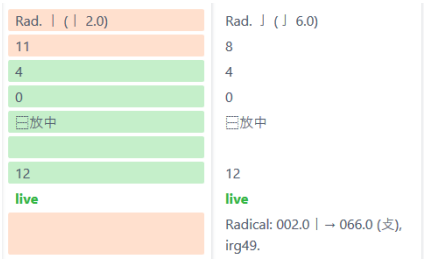

図 5　議論結果と変更が矛盾している例

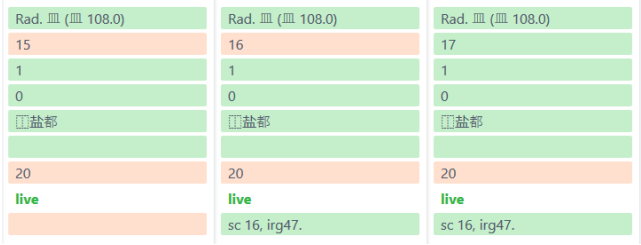

図 6　記録が抜けている変更の例

4. 成果

4.1 文書の整合性の確保

データの入力はまだ完全ではないが、本システムによって、文字情報と履歴を随時参照しながら目視によって、あるいはクエリの集計によって問題のあるデータの検出が可能になった。

これにより、図 **5** や図 **6** のような誤脱や矛盾を含む記述、あるいは毎回最新データのみ参照しているために気づかれにくかった議論の不整合（図 **7**）が直ちに確認できるようになった。

これは IRG における作業効率と信頼性の向上に貢献するだろう。

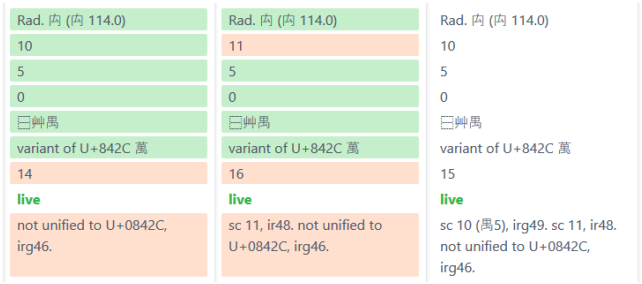

図 7　毎回議論が省略されるため修正内容が往復している例

4.2　応用の可能性

あらかじめ均質な、あるいは正規化されたデータセットを想定するリレーショナルデータベース (RDB) では、構造が安定したデータを操作するには適するが、パターン化できない不斉性を含むデータの処理は必ずしも容易ではない。人文学ではデータの性質があらかじめ明らかではなく、むしろ生のデータから構造や解釈を帰納することが研究の主題となることが多い。このような探索的な作業においては、各データ単位ごとに個別の制約や画一的に適用できない属性を考慮しなければならないことがしばしばある。この種の作業に RDB を使用すると、新たな発見やリモデルを反映するたびに、スキーマの加除や 構造に対するアドホックな修正を行わざるを得ず、管理の負担が増え操作も直感的でなくなる。

対して、グラフデータベース (GDB) の性質の一つである明示的なスキーマの欠如は、局所的な変更を全体に波及させることなく、必要な分だけ「とりあえず」入力しておくという操作を可能にする。また、GDB におけるノード間の辺という概念は、RDF にみられるように、オブジェクト間／内部の記述に兼用できる性質の良いものであり、概念モデルの大幅な変更にも耐えうる。さらに DB の実装上ネイティブ表現として取り扱われるため、特別なレイヤーなしに直接操作できることが保証される。

これにより、GDB 上では素データの中にある注目すべき点をその都度書き込むようにしてデータ構造を改変することができ、同時にそれを電子的に操作できるデータとすることができる。したがって、研究中のデータを試行錯誤するための基盤として、グラフ構造は非常に適しているのではないかと考えられる。

以上、本稿の研究も実例として、人文学的研究において既知データの保管庫としてのみならず、研究上のツールとして GDB を利用することの有益さを提言するものである。

4.3　成果物の公開

本システムのソースコードは、執筆時現在以下の URL で公開されており、自由に利用可能である。

`https://github.com/747/irg-neo`

また、仕様が安定した一部機能は人文情報学研究所の機関サーバーで実際に運用されており、データ入力と開発の進捗に伴い順次機能を公開していく予定である。

`https://irg.dhii.jp/` [*10]

なお、後者のリソースは SAT 大蔵経データベースの 2018 年版からも補足情報としてリンクされ、各提案文字ごとに IRG での議論の状況を確認できるようになっている。

5.　今後の展望

5.1　課題

実装に用いた Neo4j の制約として、1 ノードあたりの辺の数が多くなるとパフォーマンスに影響を及ぼすことが指摘されている [7]。現状では文字セットを表すノードに最大で数万の関係が繋がるケースがあるため、ユースケース次第では速度の低下が顕著になる可能性があるかもしれず、場合によっては構造の変更を検討したい。

不足データについて、現段階では IRG のリポジトリから容易に取得できないが、必要であり実在していると思われる一部情報が存在する。これらを利用できないか関係各所と調整、もしくは他プロジェクトのリソースを利用することなどを考えている。

5.2　発展

今後も引き続きデータを入力し、データセットをより完全にしていきたい。本システムの目的の一つである現場での記録は、実用に耐えうるか実地で検証する予定である。

また、本システムは現状機関内部で用いているのみだが、他の関係者の要望があれば機能を一般あるいは関係者に開放、もしくは IRG の作業プラットフォームと統合することも視野に入れている。それに応じて機能の拡張方針を定めていくことが必要となるだろう。

システムのデータベースはネイティブな RDF ではないが、RDF 化したデータを出力して Linked Open Data の提供に資することも構想している。ただし当面は作業上の実用機能の実装に注力したい。

謝辞　本研究は JSPS 科研費 15H05725 の助成を受けたものです。

参考文献

[1] IRG Rapporteur: *IRG Principles and Procedures (IRG PnP) Version 10* (online), 入手先 ⟨http://appsrv.cse.cuhk.edu.hk/~irg/irg/irg49/IRGN2275PnP10.pdf⟩（参照 2018-04-15）.

[2] CJK Editorial Group: “Editorial Report on Issues Related to Published Standard” (online), 入手先 ⟨http://appsrv.cse.cuhk.edu.hk/~irg/irg/irg48/IRGN2218.pdf⟩（参照 2018-04-15）.

[3] West A.: “UK Request to Rename UK Source Ref-

[*10] 執筆時現在、トップページが整備されていないため、例えば `https://irg.dhii.jp/browse/WS2015/1` や `https://irg.dhii.jp/source/SAT/USAT05803` にアクセスしてみていただきたい。

erences for IRG Working Set 2015” (online), 入手先 ⟨http://appsrv.cse.cuhk.edu.hk/~irg/irg/irg48/IRGN2244.pdf⟩（参照 2018-04-15）.

[4] 守岡知彦：CHISE のデータ形式 (Ver.0.1)（オンライン），入手先 ⟨http://git.chise.org/~tomo/character/chise-format.pdf⟩（参照 2018-04-15）.

[5] Robinson, I., Webber, J. and Eifrem, E.: *Graph Databases, Second Edition*, O’Reilly Media (2015).

[6] Barrasa, J.: “RDF Triple Stores vs. Labeled Property Graphs: What’s the Difference?”, *Neo4j Blog* (online), 入手先 ⟨https://neo4j.com/blog/rdf-triple-store-vs-labeled-property-graph-difference/⟩（参照 2018-04-15）.

[7] Weinberger, C.: “Index Free Adjacency or Hybrid Indexes for Graph Databases”, *ArangoDB Blog* (online), 入手先 ⟨https://www.arangodb.com/2016/04/index-free-adjacency-hybrid-indexes-graph-databases/⟩（参照 2018-04-15）.