Automated Generation of Graphs from Relational Sources to Optimise Queries for Collaborative Filtering

Ahmad Shahzad
School of Electrical Engineering and Computer Science
University of Liverpool, Liverpool, L69 3BX, U.K
Email: ahmads@liverpool.ac.uk

Frans Coenen
School of Electrical Engineering and Computer Science
University of Liverpool, Liverpool, L69 3BX, U.K
Email: coenen@liverpool.ac.uk

***Abstract*—Graph abstraction is an intuitive and effective approach for collaborative filtering as used in, for example, recommender engines. However, for many collaborative filtering applications, the transactional data is kept in a relational database and, through bespoke processes, is Exported, Transformed and Loaded (ETL) into a graph database where collaborative filtering algorithms can be applied. However, the ETL process requires knowledge of the source relational database, the target graph database and the application domain. The ETL process, therefore, tends to be expensive, non-optimised for graph queries and relies heavily on application domain knowledge and understanding of the property graph engine for the graph database. In this paper, a mechanism is presented whereby data in a relational format, which is normalised to 5^{th} normal form, can be automatically converted to a graph database format, through an automated process. The presented evaluation demonstrates, using the recommendation engine application domain as an example, that the proposed mechanism is more efficient than comparable approaches to reduce the execution time required for collaborative filtering.**

***Keywords*—Graph Construction; Collaborative Filtering; Query Optimization; Normalization; Cold Start.**

I. INTRODUCTION

Graphs are featured in a range of different application domains. They play a pivotal role for the solution of a variety of problems, from simple path finding problems to much more complex problems, such as collaborative filtering for *Recommendation Engines*. The principal advantage of representing data as graphs is that, at the physical level, a graph database satisfies the so called index-free adjacency property in which each node stores information about its neighbours only; there is no requirement for a global index of the connections between nodes. As a result, the traversal of an edge is independent of the size of the data. This makes it very efficient to conduct local analysis of the graph and means it is well suited to the processing of large data collections, or tasks, like collaborative filtering, where for any given vertex v it is required to find the most similar vertices from a set of vertices V. Although relational databases can provide a basis for collaborative filtering, they feature slow runtime, especially when there are many joins between entities. Graph databases are inherently faster in modelling and identifying associations between entities because they do not require expensive join operations and can be instantiated on distributed data frameworks. Hence, it is desirable to transform a relational database into a graph database for the purpose of performing graph analytical queries, for example collaborative filtering for recommender engines. However, the transformation process, which involves Export, Transform and Loading (ETL), is usually a bespoke process. Thus, the ETL processes tend to be expensive and require knowledge of the relational and graph databases used, and the mapping functions to associate relation and graph database entities. In this paper, a mechanism is proposed to automate the process of migration from a relational database format to a graph database format. The generated graph database shows better performance compared to other approaches in terms of the efficient execution of collaborative filtering algorithms.

A good example of a graph analytical query application is collaborative filtering in the context of *Recommendation Engines*. This is the exemplar graph analytical application domain used throughout this paper to illustrate the proposed approach. *Recommendation Engines* have a significant impact on

 ISBN: 978-1-61208-790-0

the success of business and diversity of sales [1]. The intuition underpinning *Recommendation Engines* is to take advantage of past history to make predictions. However, this means that users with little or no history cannot be provided with recommendations to any degree of accuracy. This is known as the *cold start recommendation engine* problem. There have been various techniques which have been effectively used to make recommendations given the *cold start* problem [2][3][4]. However, all these techniques rely on auxiliary information concerning the target user for whom recommendations need to be generated. This extra information is compared with other users to identify a "user group". Recommendations are then made based on the identified user group and candidate items for recommendation are ranked according to some criteria.

The problem of migrating from a relational database to a graph database, to support the faster execution of collaborative filtering algorithms, can be stated as follows. Let $S(R_1, R_2, ...R_n)$ be a relational schema which consists of a set of Relations. Each Relation R_i from schema S consists of a set of attributes, $A_i(A_{i1}, A_{i2}, \ldots, A_{im})$, which can be uniquely identified by a set of *Primary Keys*, $PK_i(PK_{i1}, PK_{i2}, \ldots, PK_{in})$. A primary key attribute $Pk_{ij} \in Pk_i$ can be used as a reference to another relation R_j, also known as a *Foreign Key*. The set of foreign keys for a relation R_j is defined as $FK_j(FK_{j1}, FK_{j2}, .., FK_{jn})$. Note that $FK_j \subset PK_j$, and that r is an instance of R and comprises a tuple of the form $\langle r_1, r_2, .., r_k \rangle$.

A graph G is defined as $G = (V, E, T_V, T_E)$, where V is a finite set of nodes, $E \subseteq V \times V$ is a finite multi-set of edges, T_V is a finite set of node types, and T_E is a finite set of edge types. Each node is mapped to a node type by a mapping function $\phi V : V \rightarrow T_V$, and each edge is mapped to an edge type by another mapping function $\phi E : E \rightarrow T_E$. A node $v_i \in V$, or an edge $ek(v_i, v_j) \in E$, has a set of $\langle attribute, value \rangle$ pairs which constitute the properties of the vertex or edge. The schema of a property graph G is defined as a directed graph $GS = (T_V, T_E)$, where T_V is a finite set of node types, and $T_E \subseteq T_V \times T_V$ is a finite set of edge types. The task is to transform S into GS.

In this paper, a comprehensive approach to the automated migration of relational to graph database storage is proposed. The proposed approach converts a relational database schema *S* into a graph database schema *GS*. As already noted, the advantage offered is that the execution of queries, which require filtering over values of low cardinality, will be more efficient than alternative relational to graph databases processes. The translation takes advantage of integrity constraints assuming 5^{th} normal form. This paper targets property graph databases to model graph data and the associated graph engine specific query language. This makes the approach independent of the specific graph engine implementation. In order to test the feasibility of the proposed approach, a complete software solution was developed for converting relational to graph databases. The solution was evaluated in the context of the *cold start* recommendation engine *collaborative filtering problem* using the *Movielens* data set. The evaluation demonstrated that there was no loss of data in translation and that the execution of queries was more efficient than in the case of other compatible approaches to achieve the same result.

The rest of this paper is organized as follows. An overview of relevant previous work is presented in Section II. This is followed by a description of the proposed approach in Section III. The evaluation of the approach is given in Section V. The paper is then completed with some conclusions presented in Section VI.

II. Related Work

There has been some previous work directed at automating the ETL process. In [5], an approach was presented to convert a relational database into a graph database founded on the property graph model. The significance of the property graph model was that it was expressive enough to cover many real world applications and it was applicable to a range of graph database realisations, such as Neo4J[6], Titan[7] and OrientDb[8]. The authors focused on speeding up the processing of queries over the constructed graph by building a graph structure based on joinable tuple aggregations. However, the approach had two main limitations: (i) a simplified version of a property graph was considered where only nodes have properties, while edges have labels that represent relationships between the data at nodes, and (ii) only relational database style queries were

considered. The work presented in [9] improved on the ideas presented in [5] by considering graph edge properties and n-way relationships when more than two foreign keys were involved. However, this led to a design which added redundant information to the edges, information that semantically did not exist. In [10], a scalable "map reduce" approach was proposed for converting relational databases into graph databases, however, the design and mapping of the relational to graph database remained a choice for the user; in other words, it was a semi-automated approach. In the context of collaborative filtering, Filho et al. [11] proposed topological analysis for the generation of heuristics in terms of betweenness, closeness and degree centrality, so as to identify nodes which could be considered to be *hubs* and/or *authorities*, so as to improve collaborative filtering results. However, the approach did not improve the run time performance. The approach presented in this paper addresses the disadvantages associated with this earlier work.

The *cold start* problem in context of *collaborative filtering* has been well studied. Suggested solutions can be categorised according to how the missing information is collected [12]: (i) explicit information collection and (ii) implicit information collection. However, in the "real world", it is not always possible to explicitly gather information about a user, hence, implicit information is typically the most feasible approach. Implicit information collection relies on using information about the user which is already available in the system or freely available in public space, such as social media. An implementation of Quantitative Association Rules (QAR) [13] for implicit information collection of *Recommendation Engines* was used with respect to the *cold start Recommendation Engine* problem presented in this paper.

III. Converting a Relational Data Model to a Graph Model

The proposed relational to graph transformation is founded on the application of a set of rules. Let K be the total number of foreign keys in a relation R_i. For any given relation R_i with a set of primary keys defined by PK_i, and any given attribute of this relation A_{ij}, let $|CA_{ij}|$ be the cardinality of that attribute. Thus, $|PK_i|$ is the total number of rows in a relation. The ratio of values to rows is:

$$\lambda = |CA_{ij}|/|PK_i| \quad (1)$$

Let T be the selected threshold for a given domain. Then the following proposed rule set can be applied to transform a relational schema S to a graph schema GS:

- Rule1: If $K == 1$ in R_i, which references R_j. For each $r_i \in R_i$ referencing $r_j \in R_j$ create two vertices for r_i and r_j linked by an edge with the property FK_{i1}.
- Rule2: If $K == 2$ in R_i, which references R_j and R_k. For each $r_i \in R_i$, referencing $r_j \in R_j$ and $r_k \in R_k$, create two vertices for r_j and r_k linked by an edge with properties equivalent to all the attributes of r_i.
- Rule3: If $K \geq 3$ in R_i, which references $R_j ... R_n$. For each $r_i \in R_i$, referencing $r_j \in R_j ... r_n \in R_n$, create $n+1$ vertices $r_i, r_j ... r_n$ with edge properties $FK_{i1} ... Fk_{in}$, respectively.
- Rule4: For any A_i at any node, if λ is less than threshold T, create a new node with a relationship to the node A_i where the property belongs. An automated Id will be generated for such new node and it will be connected to the parent node where it was originally located.
- Rule5: If a vertex r_i already exists, then it will not be created again; instead, it will be used with other vertices created using the first three rules.

The above can be applied in parallel as only one of the above rules will be applicable to each relational table contained in the relational database to be transformed.

Database normalisation ensures integrity of the data and prevents the chance of the duplication of data. A good database design should conform to at least 3^{rd} normal form. If a schema is in 3^{rd} normal form, then, in most cases, although not all, it can qualify to be in 5^{th} normal form. In practice, this is generally true for all transactional data. However, some applications do not enforce any constraints on the database side and, hence, do not have any foreign keys in the relations. In these special cases,

 ISBN: 978-1-61208-790-0

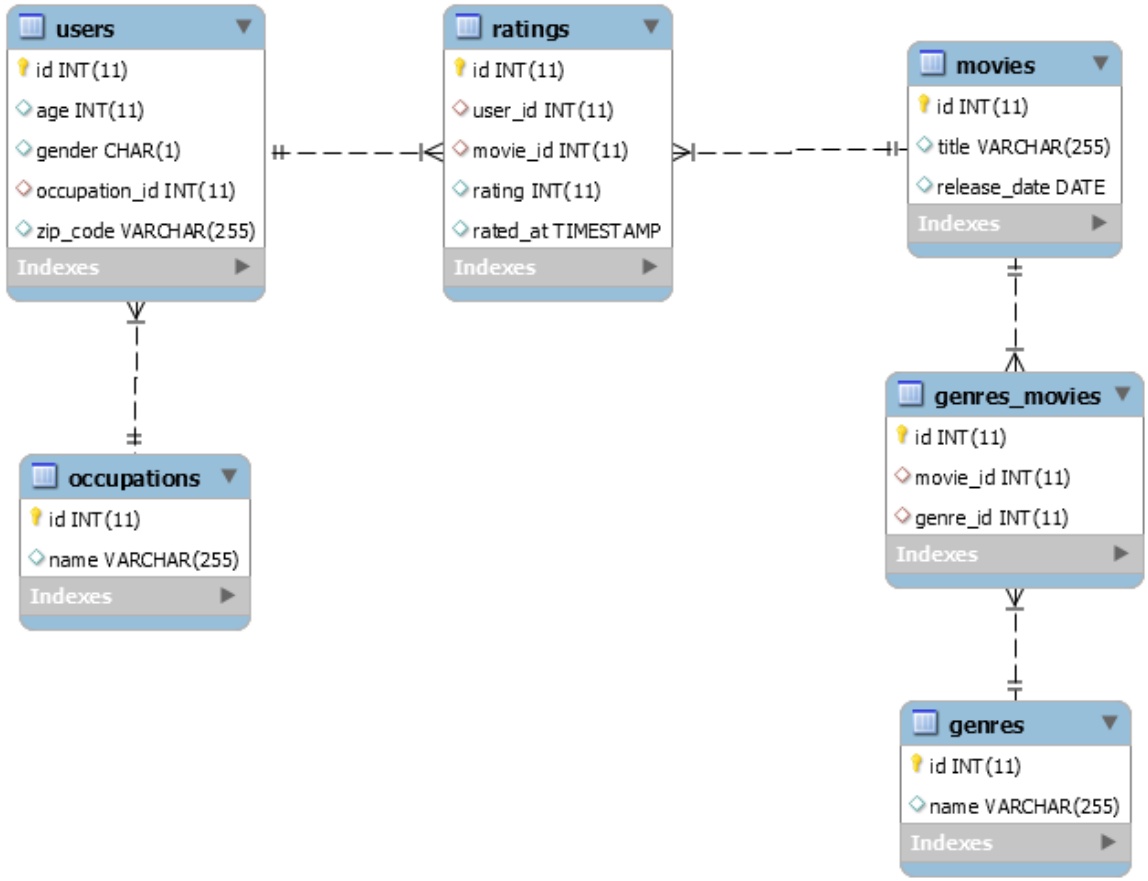

Figure 1. Entity Relationship Diagram of Movilens Database in 5^{th} normal form.

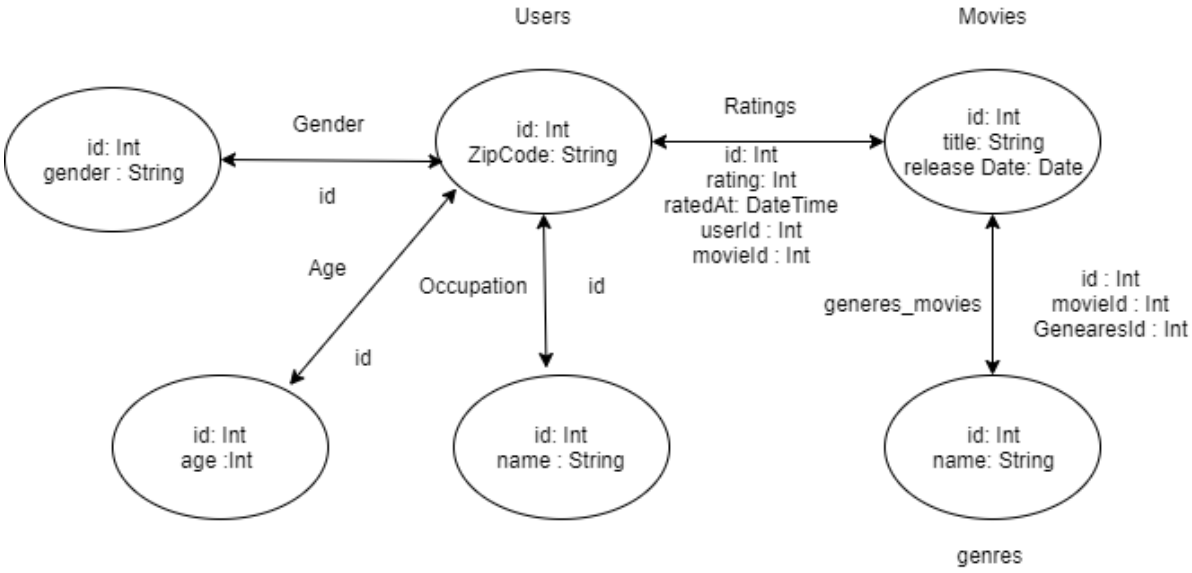

Figure 2. Generated Graph Schema for Movielens Dataset.

 ISBN: 978-1-61208-790-0

constraints are handled in the software application layers instead of the database. Also, in some cases, for the sake of faster insertion into the database, some transactional tables do not enforce normalisation principles. However, if data is not in 5^{th} normal form, then there are well-understood techniques through which the database design can be adjusted so that it is in 5^{th} normal form[14][15]. In this paper, in the context of the proposed mechanism for translating a relational database into a graph database, it is assumed that the input relational database is already in 5^{th} normal form. The reason that the relational database schema *S* is required to be in 5^{th} normal form is because, for any lower normalisation, there may exist a relation in which all columns could form a composite primary key; in other words, there will be no non-key columns. When migrating such a relational database to a graph database format, each relational row will be recorded as a new vertex, thus unnecessarily increasing the number of vertices and hence slowing down the resolution of any query that may be directed at the graph databases.

IV. MOVIELENS EXAMPLE

The Movielens database [16] is a popular choice for the study of collaborative filtering algorithms. MovieLens is a Web-based recommender system that can be used to recommend movies to its members according to their film preferences. It operates by applying collaborative filtering to its members' movie ratings and reviews. It contains some 11 million reviews for some 8500 movies. *Ratings* are expressed using the numeric range 1 to 5. For the evaluation presented in the following section, $100{,}000$ ratings, provided by 943 members with respect to 1682 movies, were used. Figure 1 shows the entity relationship diagram for the Movielens database. It shows the relational database in 5^{th} normal form. From figure1, it can be seen that, for the *User* and *Occupation* tables, Rule 1 will be applicable, because the *Users* table has only one foreign key. As a result, all rows within the *User* table become nodes in the graph that have edges linking to occupations with `occupation_id` as the edge property. Rule 2 is applicable to the *Ratings* table which features two foreign keys, therefore all user nodes have edges to movies nodes, and all the attributes of *Ratings* are set as properties of the edge between users and movies nodes. If $T = 0.1$ is assumed, then *age* and *gender* of *users* falls within the threshold T and, thus, are allocated generated Ids, taken out of the users node and placed in their own new nodes with a link to the *users* node. Figure 2 shows the resulting graph scheme, GS, after the proposed automated translation has been applied.

V. EXPERIMENTAL RESULTS

The performance evaluation of the proposed approach was conducted by comparing it against an approach founded on *Neo4j* (Neo4j ETL) which adopted the first three steps defined in Section III. An implementation of Quantitative Association Rules (QAR) [13] for recommendation engines was used with respect to cold start users. The QAR-based implementation only had limited information like *age*, *gender*, *zipcode* and *occupation* for cold start users. Based on this information, the graph database could be queried. A recommendation engine, written in Scala, was used to make recommendations according to the similarity between users based on the information provided. Experiments were conducted using five fold cross validation with 80% of users as the training data set and the remaining 20% as the test set. Both approaches produce a success rate of 67% for the recommended movies. The proposed approach, referred to as the *Optimised Collaborative Filtering (OCF)* approach, was faster by a substantial margin, as shown in Figure 3. The same results in terms of recommendation quality were produced by both approaches. However, the advantage offered by the proposed *OCF* approach was that it was more efficient in that it only selected a handful of nodes within the graph database to find similar nodes according to the input information, as opposed to scanning all user nodes and then filtering according to the input information. All property graphs can find a node in a constant time given its Id, resulting in a significant speed-up advantage for the proposed method with respect to recommender query resolution.

To further strengthen the idea, a number of example queries were executed to find the execution time taken to search the graph databases according to the given information for a simulated new MovieLens member. The results with respect to five of these queries are shown in Figure 4. With reference to the

 ISBN: 978-1-61208-790-0

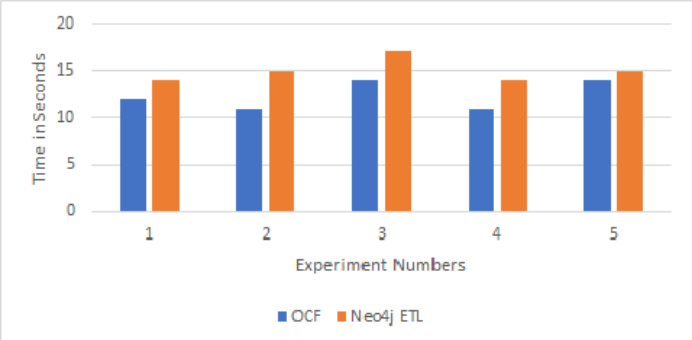

Figure 3. Five fold cross validation results comparing OCF operation with Neo4j ETL.

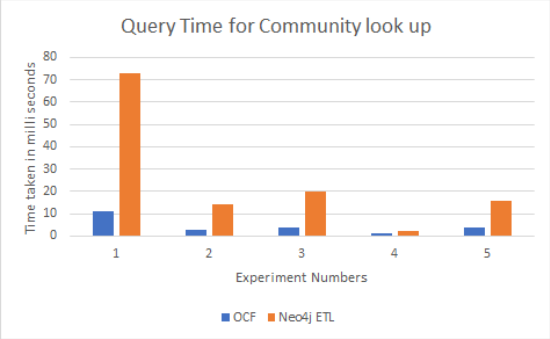

Figure 4. Runtime for five example community detection queries comparing OCF operation with Neo4j ETL.

figure, it should be noted that each "bin" represents a separate graph query. The aim was to find all users which matched the criteria for the specified member. Of course, for comparison purposes, the queries had to be expressed differently because the underlying graph schemas were different, however, the end result was the same. From the figure, it can be seen that the proposed *OCF* approach provided significant efficiency gains over the *Neo4j ETL* approach. It should also be noted that the efficiency gains were more marked when larger datasets were returned as a consequence of the query resolution.

VI. Conclusions and Further Suggestions

In this paper, an approach to the automated generation of a graph database from a given relational database has been described. The proposed approach operates in a more sophisticated manner than earlier approaches. Compared to an alternative current approach, the *Neo4j ETL* approach, the proposed approach operates much more efficiently while producing the same outcomes. This approach can be extended and tried with further splitting up the properties of edges into separate vertices if the edge property conforms to a particular threshold set in the same manner as for the proposed approach to vertex properties. The results can also be tried on distributed graph databases to reinforce the results presented in this paper.

References

[1] D. Fleder and K. Hosanagar, "Blockbuster culture's next rise or fall: The impact of recommender systems on sales diversity." *Management Science*, vol. 55, no. 5, pp. 697–712, 2009. [Online]. Available: https://pubsonline.informs.org/doi/10.1287/mnsc.1080.0974

[2] X. N. Lam, T. Vu, T. D. Le, and A. D. Duong, "Addressing cold-start problem in recommendation systems," in *Proceedings of the 2nd International Conference on Ubiquitous Information Management and Communication*, ser. ICUIMC '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 208âĂŞ211. [Online]. Available: https://doi.org/10.1145/1352793.1352837

[3] N. Mirbakhsh and C. X. Ling, "Improving top-n recommendation for cold-start users via cross-domain information," *ACM Trans.*

 ISBN: 978-1-61208-790-0

Knowl. Discov. Data, vol. 9, no. 4, Jun. 2015. [Online]. Available: https://doi.org/10.1145/2724720

[4] J. Zhu, J. Zhang, C. Zhang, Q. Wu, Y. Jia, B. Zhou, and P. S. Yu, "Chrs: Cold start recommendation across multiple heterogeneous information networks," *IEEE Access*, vol. 5, pp. 15 283–15 299, 2017.

[5] R. D. Virgilio, A. Maccioni, and R. Torlone, "Converting relational to graph databases." *Graph Data Management Experiences and Systems*, pp. 1–6, 2013. [Online]. Available: https://dl.acm.org/doi/10.1145/2484425.2484426

[6] J. J. Miller, "Graph database applications and concepts with neo4j," in *Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA*, vol. 2324, no. 36, 2013.

[7] R. Angles, "The property graph database model." in *AMW*, 2018.

[8] "Orient db, property graph model." [Online]. Available: https://orientdb.org/docs/3.0.x/datamodeling/Tutorial-Document-and-graph-model.html

[9] D. W. Wardani and J. Kiing, "Semantic mapping relational to graph model." in *Proceeding - 2014 International Conference on Computer, Control, Informatics and Its Applications, "New Challenges and Opportunities in Big Data", IC3INA 2014*, Sebelas Maret University, 2014, pp. 160–165. [Online]. Available: https://ieeexplore.ieee.org/document/7042620

[10] S. Lee, B. H. Park, S. Lim, and M. Shankar, "Table2graph: A scalable graph construction from relational tables using map-reduce," in *IEEE First International Conference on Big Data Computing Service and Applications*, 2015, pp. 294–301.

[11] S. P. L. Filho, M. C. Cavalcanti, and C. M. Justel, "Graph modeling for topological data analysis." *Enterprise Information Systems*, pp. 193–214, 2019. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-26169-6_10

[12] J. Gop and S. Jain, "A survey on solving cold start problem in recommender systems." in *Proceeding - IEEE International Conference on Computing, Communication and Automation, ICCCA 2017*, vol. 2017-January, no. Proceeding - IEEE International Conference on Computing, Communication and Automation, ICCCA 2017, Department of Computer Engineering, National Institute of Technology, 2017, pp. 133–138. [Online]. Available: https://ieeexplore.ieee.org/document/8229786

[13] S. Tyagi and K. K. Bharadwaj, "Enhanced new user recommendations based on quantitative association rule mining," *Procedia Computer Science*, vol. 10, pp. 102 – 109, 2012, aNT 2012 and MobiWIS 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050912003742

[14] M. L. Wilson, "A requirements and design aid for relational data bases," in *Proceedings of the 5th International Conference on Software Engineering*, ser. ICSE '81. IEEE Press, 1981, p. 283âĂŞ293.

[15] B. Lira, A. Cavalcanti, and A. Sampaio, "Automation of a normal form reduction strategy for object-oriented programming," in *Proceedings of the 5th Brazilian workshop on formal methods*, 2002, pp. 193–208.

[16] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context." *ACM Transactions on Interactive Intelligent Systems*, no. 4, 2015. [Online]. Available: https://dl.acm.org/doi/10.1145/2827872

 ISBN: 978-1-61208-790-0