This article proposes a specific graph database application for streamlining major knowledge management processes. The author develops a property graph data model to facilitate the process model of knowledge management.

With the advancement of emerging technologies, knowledge management continues to assume an important role in organizations. The latest technological developments have transformed the traditional landscape of organizations’ technological infrastructure, enabling them to seek innovative ways to manage their internal knowledge assets. For instance, the Internet of Things (IoT) enables ubiquitous connectivity among intelligent devices, allowing real-time data to be collected, shared, and stored. Big data poses challenges for organizations in collecting and utilizing huge amounts of data but also creates opportunities for them to capture and discover new knowledge to facilitate further development.

Among emerging technologies, NoSQL databases have become popular alternatives to traditional relational databases due to advantages such as flexibility, scalability, efficiency, and the ability to handle large amounts of unstructured, semistructured, and structured data. As a special type of NoSQL database, graph databases have the same advantages. In addition, graph databases are particularly useful in handling data elements that have complicated relationships (bit.ly/2hgsGa9). Due to their unique features, graph databases have been widely applied in various areas, such as fraud detection, real-time...
recommendation engines, master data management, network and IT operations, and identity and asset management.\textsuperscript{6} Despite the growing popularity of graph database applications in various areas, there is little research studying how graph databases can support knowledge management practices. This article attempts to address this gap by proposing a specific property graph database model to facilitate traditional knowledge management processes in organizations.

**Knowledge Management Process Model**

Knowledge management can be implemented as a series of processes in organizations.\textsuperscript{7,8} To be consistent with the traditional landscape of knowledge management, the following processes, moderated by a knowledge management system, are considered in this research. First, knowledge needs to be discovered—either from existing documents as explicit knowledge or as tacit knowledge from knowledge workers who have the expertise—and then captured and created. Second, knowledge must be codified and stored for future retrieval purposes. Third, knowledge needs to be transferred and shared; depending on the nature of knowledge, different processes will be used. Specifically, explicit knowledge contained in existing documents can be easily transferred to knowledge seekers, whereas tacit knowledge needs to be shared among knowledge workers. Finally, knowledge obtained by recipients will be applied in specific tasks, and knowledge used will be assessed for its quality and performance.

Figure 1 shows a dataflow diagram of the major processes the system supports. Process 1.0 allows knowledge workers to register their new expertise and update their existing expertise listed in the system. Datastore D1 maintains an expertise directory for all the knowledge workers in the organization. Process 2.0 enables knowledge workers to search for existing knowledge residing in organizations—either the explicit knowledge manifested by the documents stored in the...

![Figure 1. Dataflow diagram of a knowledge management system. The system supports several major processes, including maintaining an expertise directory and letting knowledge workers search for existing organizational knowledge.](image-url)
repository or the tacit knowledge associated with one or more knowledge workers. Keywords are attached to a document or a knowledge worker so that people can use them to search for a specific piece of knowledge. Datastores D2 and D3 are used to store the explicit knowledge and the metadata for tacit knowledge (that is, the knowledge workers who possess specific tacit knowledge). Processes 2.0 and 3.0 facilitate the matching process of locating a particular document or an expert based on the original request.

Although the system’s timing is not explicitly modeled in the dataflow diagram, the system will always try to match specific documents with the requested knowledge keywords. Only when such documents are not available will the system then recommend that potential knowledge experts share their tacit knowledge.

External knowledge is not considered in this system—that is, it is assumed that the system will not resort to external sources for knowledge capture and acquisition. However, in a future extension not modeled in this article, the system will be able to interface with other external systems to allow outside experts to interact with it.

The process model serves as the foundation for the development of a data model that can facilitate the required processes for the knowledge management system. The next section illustrates a conceptual data model based on the traditional relational database approach.

### Relational Data Model

Following on the process model, the conceptual data model can be developed to reflect the situation when a relational database design is adopted. To be consistent, the datastores modeled in Figure 1 should be considered in the conceptual data model.

Figure 2 demonstrates the relational conceptual data model, in which four entities are created to reflect the process facilitation needs, shown in Figure 1. The **Expertise** entity stores a list of expertise keywords and their descriptions. The **Knowledge Worker** entity contains a list of expertise keywords and their descriptions. The **Document** entity contains authored documents, and the **Worker’s Expertise** entity relates to the expertise keywords and their descriptions.

---

**Figure 2.** The relational data model of the knowledge management system. Four entities reflect the process facilitation needs: **Expertise**, **Knowledge Worker**, **Document**, and **Worker’s Expertise**.
Knowledge Worker entities capture important information about knowledge workers. The Document entity keeps the major characteristics of each document, such as its document ID, name, description, and type. The Worker’s Expertise entity is essentially an associative entity that matches a worker with his or her specific expertise keywords, which can be added and updated at a specific time.

The relationships among these four entities are quite complex. Worker’s Expertise reduces the original many-to-many relationship between Knowledge Worker and Expertise to two one-to-many relationships between Knowledge Worker and Worker’s Expertise and between Expertise and Worker’s Expertise, respectively.

To implement this conceptual data model in a relational database management system (RDBMS), all the many-to-many relationships need to be further reduced, which means additional relations need to be introduced. Because there are currently two binary and three unary many-to-many relationships, five additional relations should be constructed to fully implement the data model in an RDBMS. It is evident that the implementation of this data model is quite cumbersome. In addition, the newly introduced relations will negatively influence the response time of the queries searching the database because individual tables need to be joined to return the results of each query. The next section describes how a property graph data model can avoid these potential problems in a relational data model.

**Property Graph Data Model**
A graph database can be conceptually illustrated through a property graph data model. A property graph contains all the interconnected nodes used to hold the attributes that describe each node. One special aspect of graph data models is that all the nodes can be tagged with labels to indicate their roles in the domain (see tinkerpop.apache.org). The labels’ contextualizing role can be effective in modeling the different roles assumed by a knowledge worker in an organization, which is an additional benefit of applying graph databases in this situation.

In a property graph data model, relationships also exist but are used to connect nodes. Like those in a relational data model, relationships can have properties to describe how two nodes are related. However, a relationship in a property graph is directional.

A relational data model can be easily converted to a property graph data model. Figure 3 shows the property graph data model converted from the conceptual relational model in Figure 2. Because a many-to-many relationship can be easily implemented, there is no need to separately model the Worker’s Expertise node; instead, a many-to-many relationship can be directly modeled between the Knowledge Worker and Expertise nodes. Two properties (date created and date updated) can be attached to this relationship to indicate when a knowledge worker creates and updates his or her expertise items. Finally, only three nodes (Knowledge Worker, Expertise, and Document) are needed with a many-to-many relationship between any two of them. In addition, each node has a many-to-many relationship with itself.

**Implementation**
The implementation of the property graph data model can be straightforward. Neo4j Community Edition was chosen as the graph database management system to demonstrate the following example (see Figure 4), which includes three knowledge workers, three documents, and three expertise items. In addition, there are 22 arcs that describe how these 9 nodes relate to each other with many-to-many relationships. For instance, Expertise One (Exp One) relates to Exp Two, which further relates to Exp Three. Document One (Doc One) links to Doc Two and Doc Three, and Doc Two links to Doc Three. The nodes and arcs can be easily created using Neo4j query statements.

As an illustration, Figure 5 demonstrates some source query code to create the nodes of Document One, Knowledge Worker One, and Expertise One as well as the arcs between Worker One and Document One, between Worker One and Expertise One, between Document One and Expertise One, between Document One and Document Two, between Document One and Expertise One, between Expertise One and Expertise Two, and between Worker One and Worker Two. When new knowledge workers, expertise items, and documents need to be added, the query can be quickly expanded to insert the new nodes and arcs into the graph database.

Finally, two sample queries (shown in Figure 6) are used to illustrate the processes that can be facilitated to allow knowledge workers to search
**Figure 3.** The property graph data model of the knowledge management system. A relational data model can be easily converted to a property graph data model such as this one.

**Figure 4.** Graph database example using the Neo4j graph database management system. Nodes and arcs can be easily created using Neo4j query statements.
the graph database for specific expertise items. Query 1 can search for all the existing documents that match a particular expertise keyword (Exp One in this example). If no such document is available in the system, then Query 2 can be used to look for available knowledge workers who can provide such expertise. Additional queries can be developed to explore the relevant documents linked to each other and similar expertise available in the system, given that no such expertise items and documents are found in the current database.

Knowledge management continues to assume an important role despite the changing landscape of technological infrastructure in organizations. Emerging technologies provide opportunities for companies to manage their knowledge assets with more innovative and effective methods.

Relational databases have been the core components of knowledge management systems to facilitate common knowledge management processes in organizations, including knowledge capture, storage, transfer, and application. Due to the complex nature of these processes, it is cumbersome to design, develop, and implement a system based on relational databases to facilitate them.

Graph databases can be a good alternative to traditional relational databases as the core component of a knowledge management system because of their special features. First, graph databases are ideal for modeling scenarios in a social network context in which knowledge management processes are well represented. Second, the many-to-many relationships in a relational database can be easily implemented as arcs in a graph database without introducing additional nodes. Third, the data stored in a graph database can be quickly accessed through simple queries, whereas multiple tables must be joined in a relational database for the same purpose.

This article proposes a specific application of graph databases in streamlining the major knowledge management processes. A property graph data model is presented in accordance with
the conceptual relational data model to facilitate the process model of knowledge management. In addition, the property graph data model is implemented through Neo4j Community Edition to demonstrate the ease of designing, developing, and implementing a knowledge management system based on graph databases.

This research contributes to the literature by exploring how graph databases can support knowledge management, thus providing guidance for practitioners seeking alternative approaches to traditional methods as well as a foundation for researchers to further study those unique features of graph databases that can improve the efficacy of knowledge management.

References

Zuopeng (Justin) Zhang is the editor in chief of the Journal of Global Information Management and an associate professor of management information systems at State University of New York at Plattsburgh. His research interests include the economics of information systems, knowledge management, electronic business, business process management, information security, and social networking. Zhang’s research publications have appeared in several leading journals in the field. He received a PhD in business administration with a concentration on management science and information systems from Pennsylvania State University. Contact him at zzhan001@plattsburgh.edu.

IEEE Software seeks practical, readable articles that will appeal to experts and nonexperts alike. The magazine aims to deliver reliable information to software developers and managers to help them stay on top of rapid technology change. Submissions must be original and no more than 4,700 words, including 250 words for each table and figure.

Author guidelines: www.computer.org/software/author
Further details: software@computer.org
www.computer.org/software