Chapter 14

Integrating personal knowledge graphs into the enterprise

Dan McCreary[a]

This chapter covers the topic of integrating personal knowledge graphs (PKGs) into a large enterprise. When the term refers to a personal knowledge graph (PKG) that is integrated into the enterprise, it is used as integrated personal knowledge graph (IPKG). The proposed chapter consists of four parts: **Background** on PKGs to explore about the fundamentals of PKGs; **IPKG challenges** to explore the challenges of integrating PKGs into an enterprise knowledge graph and the role that machine learning can play in the future; **IPKG steps** to show the flow of steps in performing the integration; **Security** to find grain access control – using role and authorization.

14.1 Introduction of PKGs

A new generation of note-taking tools helps us quickly organize thoughts as knowledge graphs. The concept of a PKG goes back to 2019,[1] the term has recently become popular because of a new generation of note-taking software that uses knowledge graph representations. This chapter will introduce you to PKGs and then discuss how PKGs might impact overall corporate knowledge management strategy and how they might eventually work with enterprise knowledge graphs.[2]

What is a PKG? A PKG is a new class of software that allows you to efficiently take notes in the form of flexible non-linear knowledge graphs. PKGs have evolved from older linear note-taking and outlining software as presented in Figure 14.1. PKGs are quickly gaining popularity among students, researchers, software developers, bloggers, and creative content authors.

A brief history of note-taking using linked notecards. Back in the 1500s, researchers in Germany developed a knowledge capture system called Zettelkasten.[3] It was essentially a way of putting ideas on index cards that showed how ideas

[a]Kelly-McCreary and Associates LLC, USA

[1]https://krisztianbalog.com/files/ictir2019-pkg.pdf

[2]https://dmccreary.medium.com/a-definition-of-enterprise-in-ekgs-561283d37deb

[3]https://en.wikipedia.org/wiki/Zettelkasten

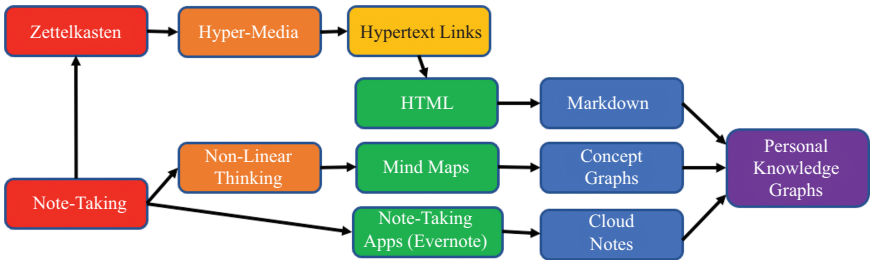


Figure 14.1 The evolution of PKGs

or concepts were related to other concepts. The ideas around One-Concept-Per-Card and Concept-Linking developed in Zettlekasten were then transferred to electronic form in the 1960s through the concept of HyperMedia,[4] where any part of a document could be linked to any other part of that document or any other document. HyperMedia evolved into hypertext which became the basis of the Hypertext Markup Language (HTML). In 2001, the authors of the worldwide web (Tim Berners-Lee) also attempted adding relationship types to these links, which became the basis for the Semantic Web.[5] The semantic web was an attempt to make it easier to store distributed but still connected information on web pages.

In parallel to the evolution of concept linking, we saw the growth of the personal note-taking[6] software industry. These electronic tools, such as outlining tools, mind-mapping tools, and cross-platform tools such as Evernote,[7] allowed individuals to create and retain their notes on many different devices such as their cell phones and tablets. These disparate devices were all tied to a cloud-connected storage system so all your notes on all your devices could stay in sync.

The rise of Markdown for knowledge capture and knowledge interchange: Although note-taking tools were handy, they lacked a simple way to capture and interchange linked knowledge between systems. Markup syntax such as HTML is portable, yet most users did not want to capture their notes in hypertext because it required too much work to format things such as links to other concepts. What was needed was a "lightweight" version of HTML that was easier to type in. In 2004, the Markdown[8] format was created to be just that as discussed in Figure 14.2. A lightweight way for users to quickly enter knowledge without requiring them to create highly structured markup. Any text editor such as Notepad can be used to create markup, and a few hundred lines of Python code are all that is needed to convert Markdown into HTML.

[4]https://en.wikipedia.org/wiki/Hypermedia
[5]https://en.wikipedia.org/wiki/Semantic_Web
[6]https://en.wikipedia.org/wiki/Note-taking
[7]https://en.wikipedia.org/wiki/Evernote
[8]https://en.wikipedia.org/wiki/Markdown

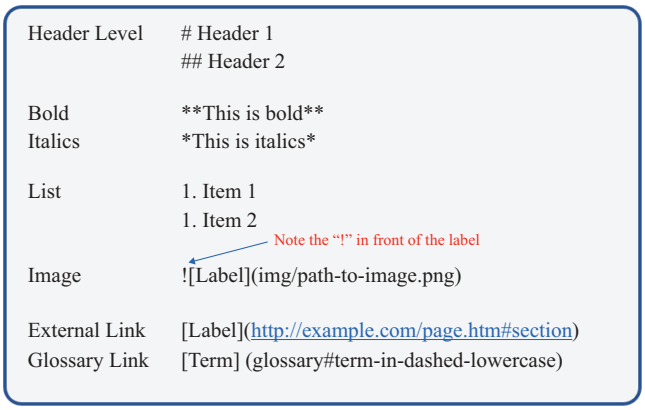


Figure 14.2 Sample cheat sheet showing common markdown syntax

Since 2004, Markdown has steadily grown to be the number one way software engineers and data scientists, create documentation about their code. It is now the default way to capture documentation on GitHub,[9] in Data Science with Jupyter Notebooks,[10] and create project documentation with Microsites.[11] Even traditional products such as Google Docs[12] are adding Markdown features. Adding "smart auto-complete" into IDEs such as VisualStudio makes it easier to create and edit markup. Want to point to an image in your repo? Just type a '![]("and VisualStudio will start to suggest path image files! AQ1

Combining hyperlinks, Markdown, and graphs

Another generation of personal note-taking tools has been explored to combine concept linking, Markdown, and knowledge graphs. These new tools are quickly evolving, and existing note-taking tools are beginning to add more graph-aware features. This new process and software category is called the personal knowledge graph (PKG) space. It started with innovative projects such as Roam Research[13] and the open-source Obsidian[14] product.

Why are PKGs so popular?

The amount of new information about almost any field of study grows exponentially. Doing a simple Google search on a topic is like drinking out of a firehose – it is hard to figure out what is essential in this continual stream of incoming information. PKG

[9]https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax
[10]https://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Working With Markdown Cells.html
[11]https://dmccreary.medium.com/auto-generating-ekg-documents-2c85f1e45f44
[12]https://support.google.com/docs/answer/12014036?hl=en
[13]https://roamresearch.com/
[14]https://obsidian.md/

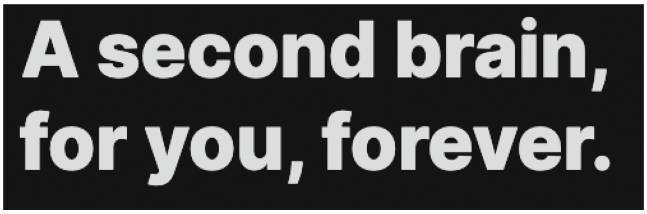


Figure 14.3 "A second brain" in the PKG community

community often repeated a word "second brain" as presented in Figure 14.3. The idea of retaining this knowledge over time is a popular selling point.[15]

This section covers a few common reasons people tell why people love their PKGs.

Advantage 1: Quickly organize thoughts

PKGs allow you to quickly organize the information coming out of these streams to figure out what is important, make new connections between existing topics, and create your insights. You can import any text, and just by highlighting a word or phrase and typing "[[" it magically turns into a vertex in the concept graph! As anyone type, the "autocomplete" can tell if they already have it in their PKG, and you can thus quickly link items together.

Advantage 2: Lowers your cognitive load

Before using PKGs, it was possible to frequently get stressed during and after constant back-to-back meeting schedules. In this situation, rarely had time between meetings to write down thoughts and try to re-prioritize tasks as user learned new content. Now user can both take notes during meetings and link the notes to my existing knowledge base. User stress levels have come way down! The problem is that our brain's short-term memory can only hold a fixed number of things. If we do not get them committed to documents, they disappear. User recall user neglected to write down these new ideas only days or weeks later. Many of these are related to the "offloading" of the task of remembering concepts and their relationships to another system. These are often referred to as the "Second Brain" arguments on why PKGs are useful. The PKG becomes a "persistent shadow" of real brain – it stores the key concepts and their relationships in a non-linear structure – a graph – similar to how our brain works.

Advantage 3: Promotes non-linear thinking

The third thing, noticed often is that PKGs help us break away from the "Tyranny of Linear Thinking." The order of traversal is usually fixed. This contrasts with traditional note-taking using documents that contain a fixed order of items lists. Document order requires us to have a series of sections and subsections. But the inherent order you encounter information while taking notes might not be the real way that you have insights into the world. Outlining tools can allow you to rearrange the order of topics

[15]https://obsidian.md

and pop down and pop up the order of sections. Still, it is nowhere near as flexible and powerful as connecting each topic to an existing knowledge graph!

Advantage 4: Long-term persistence

It seems like companies change the way they store knowledge every few months. It might be MS Word stored in file folders one year, a wiki the following year, then perhaps Microsoft SharePoint™, and now we are seeing a shift to microsites on GitHub. Every time an organization changes its tools, the knowledge is lost. Because they are built around standards like easy-to-edit Markdown, we now have hope that this data can persist for a long time. Although things like concept links, tags, and aliases are not yet standardized across platforms, our hope is that lightweight PKI inoperability standards without the need to implement a complete DocBook[16] compatibility checklist.

Advantage 5: Breaks writers' block

One of our goals is to try to write a blog every other week. Sometimes we have an idea, write the first paragraph, and we get blocked. We cannot figure out how to organize our thoughts in any way that will be somewhat coherent to our readers. If we keep lots of little graphs around for ideas, we can see them grow over time. The metaphor is to plant your "Digital Garden of Ideas" and let them grow naturally. When we have about a dozen good ideas in a subgraph it might be time to make it an actual blog.

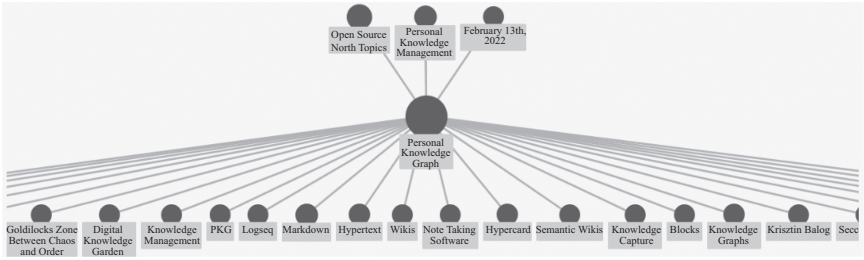
These advantages are their own personal view on how we use PKGs. If anyone searches on Google, it can be found that students, researchers, creative writers, and bloggers are all using PKGs in different ways.

Challenges with PKG tools

Although the PKG tools such as Roam Research[17] and Obsidian[18] are relatively new, they are adding new features (both free and for fee) quickly. My biggest challenge has been trying to share sub-sections of my PKG with others and publishing subgraph layouts on microsites. Right now, we need to do crude import/export operations to combine our knowledge graphs to a place where we can collaborate. However, it is suspected that these new features are coming quickly.

Our biggest challenge with the current generation of tools is the limited ways, we can generate visualizations of subgraphs. Like using the powerful GraphViz[19] libraries, we are able to try multiple constrained layout algorithms and play with top-down and left-to-right layouts. These layout rules are specified in a dot[20]-like language. Figure 14.4 shows that this rendering is "flat" and does not show how the concepts on the bottom are interrelated. It is planned to "pin" the arrangement of various nodes in the layout so they do not randomly change as we add new items. Definitely all these features will come with time. For now, we can write Python scripts

[16] https://en.wikipedia.org/wiki/DocBook
[17] https://roamresearch.com/
[18] https://obsidian.md/
[19] https://graphviz.org/docs/layouts/
[20] https://graphviz.org/doc/info/lang.html



AQ4

Figure 14.4 A screen image of an early graph on PKGs

that take the Markdown and convert them into Mermaid[21] structures for automatic layouts in our published microsites. These tools also do not allow us to add types or attributes to our relationships. That might also come in a future release. Remember that a little bit of semantics goes a long way!

Impact on the enterprise knowledge graph market

Vertex-level Role-based Access Control (RBAC) is a hallmark of enterprise-grade knowledge sharing systems like TigerGraph or Smartlogic's Ontology Editing tools. Right now, these PKG systems are all cloud-backed desktop tools or websites.They do not have fine-grain role-based access control rules so we can grant our teams read or write access to a subset of my PKG. However, it cannot be believed that the quickly growing PKG market will not find customers who want to combine note-taking's free-flowing nature and formal curated and approved knowledge graph systems. As APIs evolve, it could easily see functions such as "Publish this subgraph to the corporate ontology server for review."

Legal question: Who owns your PKG?

It cannot help but think how the complexity of people building their super-high-quality PKGs will overlap with the concerns of the intellectual property of organizations. During AT&T Bell Labs in the 1980s, all our written engineering notes were required to be done using the standardized AT&T Bell Laboratories blue-lined notebooks with individual page numbers. We were required to date each entry so that we could defend our intellectual property and patents carefully using these formal note-taking systems. We were told that adding a date, a few lines of text, and simple diagrams could save the company millions of dollars. Yet the word "Personal" implies that you own your own second brain. This will run against the concerns of corporate legal staff protecting their intellectual property.

Section summary

We have found it may take teams a while to get used to these new tools. They are a bit rough around the edges and do not yet have the same presentation and flexibility as more mature note-taking tools. My suggestion is to be patient and give yourself a few days/weeks to adjust the tool to meet your style of notetaking and publishing. Try out different ways of organizing concepts and see if you can find a pattern that works.

[21] https://squidfunk.github.io/mkdocs-material/reference/diagrams/#usage

14.2 The challenges of integrating PKGs into the enterprise

This section will focus on the challenges of integrating PKGs into larger knowledge ecosystems such as your company's enterprise knowledge graph (EKG) or your college and universities knowledge management system.

How AI is driving PKG technology

The intended perspective of this section is a solution architect that has been asked to understand the trade-offs between many isolated PKG silos and more integrated company knowledge graphs that share knowledge and avoids knowledge duplication and inconsistency. We look at the limited integration options available today and forecast large return-on-investments as PKG products integrate the latest recommendations of large machine learning models. Figure 14.5 shows that adding a new concept to a small PKG is much simpler than adding a new concept to a large complex EKG. If you are new to the area of PKGs, we strongly recommend reading the prior articles on PKGs, EKGs, and knowledge management to fully understand these concepts.

The personal vs. organizational reuse tension

Before we dive into terminology, let us remember that there are always two opposing forces driving PKGs in an organization. The first force is the need for each person to quickly capture notes that are consistent with their personal knowledge base. As you type, auto suggest lists allow you to quickly connect to your own personal concepts in your own PKG. When you type "[[" in the text editor, a list of existing concepts is presented. As you type, the list is automatically narrowed to match the prefix you have already started. The second force is the desire of many companies to capture this knowledge in a way that can be used by others in the company. This means that big organizations will often put in additional rules that prevent quick knowledge capture. Do you want to issue a ticket in the helpdesk system? You must tell us about your computer, what you are trying to do, what application you are using, and some information about how to reproduce the error. Without these required fields filled in, the Save button is disabled. Figure 14.6 shows a gap between free form of knowledge

Figure 14.5 Integration of concept

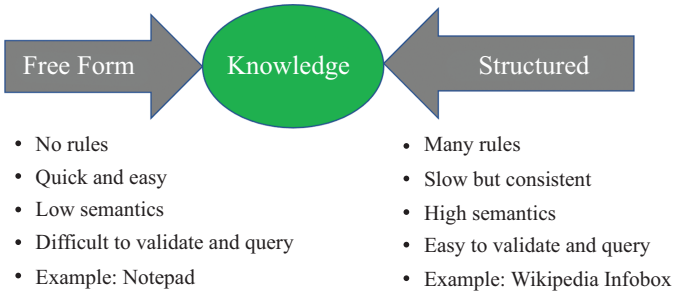


Figure 14.6 The inherent tension between free-form personal notetaking and the need for consistent structure across similarly typed knowledge

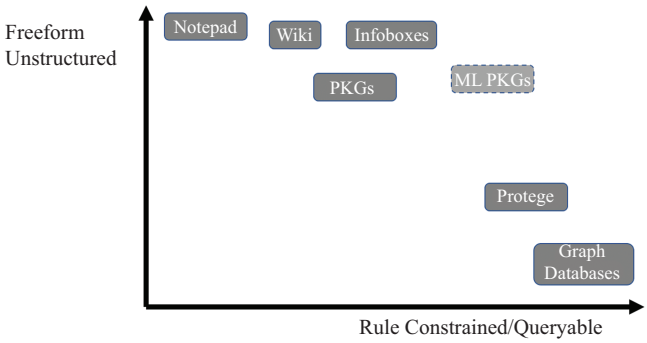


Figure 14.7 This chart shows a spectrum of tools for taking notes and collecting linked knowledge

and structured knowledge. Our experiences are that we will always see trade-offs. Who pays for the software, how the knowledge is captured, and how knowledge-sharing MBOs are written is often driving concern. There is no one-size-fits-all solution here, and the use of large machine-learning Natural Language Processing (NLP) tools is only making the decisions more complex. We will discuss this topic more later in the blog.

To begin our discussion, consider two desktop tools. The first is a standalone editing tool such as Microsoft Notepad™, which comes with your Windows™ desktop applications. Think of this as adding a new leaf of knowledge to a small new sapling of a tree. This tool allows you to type in any text from your keyboard and save it to your local file system. A Notepad is ideal for capturing new information that is disconnected from the rest of the world. You do not even need to be connected to your corporate network to use it. Figure 14.7 presents a spectrum of tools for taking notes and acquiring knowledge. Wikis also allow you to enter freeform text, but always

within a page that has a unique name. Your text may contain references to other named pages. Some Wiki pages can contain structured key-value pairs often called Infoboxes. The key difference between Notepad and Wiki is that Wiki pages are stored on a server and converted to web pages with links.

Now let us consider a PKG editor such as Roam or Obsidian. They are always connected to an existing knowledge base, and the focus is creating quick links to your internal personal graph. Here is a metaphor we want to understand clearly from Figure 14.1. PKGs are useful when we also want to capture new information that is an extension of the existing knowledge structure. For example, when you enter a simple concept name, the system checks to verify that it is a new concept and not a duplicate of an existing concept or an alias for an existing concept.

On the lower right corner of the chart are formal ontology editing tools such as Protege and graph databases such as Neo4j or TigerGraph. These graph databases have many constraints, such as formal types for every vertex and edge and strict rules about what types of vertices can connect to each other. Edges also must have a type.

The last box in Figure 14.7 is the Machine-Learning assisted PKG (ML-PKG), where NLP tools are used to automatically suggest text within your PKG editor. These tools do not exist yet, but with the fast evolution of large-language models such as BERT and GPT-3, we expect them to be available soon.

In summary, there are going to be lots of rules in an integrated Enterprise PKG. Many of these rules can be helpful. They can accelerate your work by adding knowledge using intelligent contextual autocomplete. Some of these rules will get in the way and slow down your knowledge capture rate.

Basic terminology

Let us start our discussion with some basic terms and concepts. In a PKG, the goal is to efficiently allow a typed stream of characters to efficiently capture knowledge concepts and make connections to prior concepts. In a PKG, we use a data structure called a knowledge graph. A graph stores not just a flat list of concepts but also the relationship between these concepts. The technical term we use for concept storage is a vertex, and the term for relationships is called an edge.

Untyped vs. typed systems

Many of us are familiar with the concept of a Wiki. The word "wiki" is a Hawaiian term for "quick." These were created so that the user could minimize the number of keystrokes that connect wiki "pages" to each other. In the Wiki data model, all concepts are stored on a wiki page, and the design goal is to follow the one-page-per-concept rule so that the concepts can be easily linked together. If you put two or more concepts on a page, a link to that page would not clearly show a single-concept-to-single-concept pattern.

Wiki's were a wonderful step forward in notetaking because the concept pages or concept cards were shared in a web server. As your peers added new concepts, you could create links to these concepts. As you added new concepts, your peers could also link to them. It was a huge breakthrough in shared knowledge management, and it is the basis for systems such as Wikipedia.

However, most people do not think of a Wiki as true "knowledge graphs" because they really are just collections of linked untyped documents, just like the world wide web. The designers of wiki wanted to extend the concept of "hyperlinks" and make the links easier to type than remembering the complex syntax of HTML anchor references. The fundamental difference between a wiki and a true knowledge graph is the fact that wiki pages are all of the same "type" (a document/card/concept type), and the relationships also all have a single type (is-related-to).

True knowledge graphs allow each vertex and edge to have a specific type. The list of possible types is often configured by a centralized department. For example, some vertices might represent an employee, some might represent products you sell, and some might describe business events such as a call made to your company helpdesk. The key thing about types is that we can associate rules with types, and these rules force consistency that makes knowledge graphs easy to query, just like we query a traditional database using SQL.

If you are adding a new city to a PKG, you can tell the system the vertex is of type "city" and also add a "located-in" relationship to a "state" vertex. Both vertices
AQ2 and relationships have typ

In the field of enterprise knowledge graph strategy, we use the Jim Hendler hypothesis "A little semantics goes a long way."[22] Adding a few simple types to both vertices and edges is the principal way to add meaning (semantics) to PKGs to supercharge their quearability and reuse. Many organizations have found that shared wikis are an ideal system for storing and linking knowledge. However, most organizations do not query the wikis because they put a priority on free-form editing over strict rules. You can always try to impose structure on a wiki page, but these structures are often optional, and few wiki systems prevent you from saving a wiki page if the required fields are missing. Wikipedia's Infoboxes[23] are an example of how wikis can be retrofitted to add structured data that can be queried.

Forcing name uniqueness

Although most wikis do not allow you to assign types to new pages at creation time, they do have some other rules that knowledge graphs do not have. When you create a new page, you need to give it a name. This name serves as the page identifier with a wiki. Now two pages can have the same name. In most PKGs, we have a similar rule. If you try to enter a new page with the same name, the software will tell you the page already exists. If you rename a page to be the same name as an existing concept, the system will ask you if you want to merge the names. Adding aliases (one or more labels for the same concept) is another challenge many PKGs do not do well.

Applying typed systems to notetaking event loops

Now that we have an idea of the critical difference between wikis and knowledge graphs, let us ask the question, how can we accelerate the ability to make connections as we enter text into a PKG page? We will start with some simple examples and then do a deeper dive into some more complex topics.

[22]https://www.cs.rpi.edu/ hendler/LittleSemanticsWeb.html
[23]https://en.wikipedia.org/wiki/Help:Infobox

The general pattern we use is to visualize characters typed on a keyboard. As the user types each new character, we apply a set of rules to assist the user in making the decision if there is an opportunity to add an edge to an existing concept. If there is, the users will use a character, such as a tab key, to confirm the relationships and auto-complete the remaining characters for the edge. If they do not want to accept the suggestion, they can just keep typing.

The key is that when the user is typing, they often want to signal that a link to a specific typed concept is needed. You might already be familiar with this process if you use social media and want to reference a person or a concept using "" to reference a person or a Twitter account. You might add hashtags using a "#" character at the end of your blog post to let recommender systems tie your blog post to people interested in a specific concept. Once you signal to the authoring system you want to reference an existing node with a type, it confirms this as you type or suggest a list of auto-completions.

The key is that organizations have many types of data that you want to include in your edge recommendation. Our focus is finding ways to integrate these organization-specific types into the as-you-type loop in the text editors for our PKG tools. If you want to reference employees in your company, the "@" might pull a list from your list of company employee names. Clicking on the link for that employee will allow the user to go to the PKG page for that employee.

Another common integration point is linking to a standard company glossary of terms and acronyms. Instead of using a special keyboard symbol, we can add a colon-separated prefix to a vertex name. For example, if we have a company glossary of terms, we can have your users use "g:" as a prefix, and the autocomplete will match those terms. If you are using an industry-specific glossary, we can reference those terminology standards in the prefix. For example, in healthcare, we use the "Just Plain Clear" glossary so that the prefix "jpc:" will auto-fill terms from that system.

Adding custom types

My suggestion when we are creating a PKG integration strategy, start small and build on your successes based on the value each integration adds to our user community. Start with simple employee lists and company glossary terms that are commonly referenced. Try experiments with things such as "g:" for a company glossary or "w:" for a Wikipedia term and see how frequently they are referenced. If there is low adoption of these suggestions, then they should be removed. Remember, people can always add external links to Wikipedia.

Acceptance rate monitoring

There have now been extensive studies about how large-language NLP models such as BERT and GPT-3 can be used to aid software developers by suggesting code. When a system makes recommendations to a user about a suggested autocompletion, if the user accepts the suggestion, this is logged in an event log. The parent of suggestions that a user accepts, called the acceptance rate, is critical for their adoption of these tools. In general, if your users do not accept around one-third of the autocomplete suggestions, the tools will become more annoying than helpful. Users will disable the

tools, and you will not get the feedback you need to be successful. So it is critical to aim for a 30% or higher acceptance rate before you roll these tools out into production.

Permalinks and PURLs

Adding full URL links from your notes to any internal corporate resources is also something to be wary of. Many internal document management systems, such as Sharepoint (TM), depend on the links to specific files within a hierarchical file system. When the permissions to these folders change, or the documents are moved, these links will no longer work. A better choice is to only make links in your PKGs to locations that your organization has made commitments to not changing – ever. We call these links Permalinks or PURLs. This can be done by careful management of your internal domain-name system so that a link to http://glossary.mycompany.com/#term will always work, even if the servers the glossary is hosted on changes.

The next step is to ensure that your PKG integration team has tools that can monitor broken links and work proactively to prevent links from being broken as content moves around your organization's servers. Consider creating a "value" of one dollar for a working link and help people under the knowledge value that is lost as links break and the time it takes to update broken links.

Most knowledge graphs support bidirectional links. If you change the name of a concept page, all the links to that page will automatically be updated. This is a HUGE win and amounts to super-consistent and low-cost relationship management.

As a general rule of thumb, 60% of the value of a knowledge graph is the nodes, and 40% of the value is the relationships. Broken links discourage authors from making future links, and users are more reluctant to trust systems with many broken links.

Integration with NLP frameworks

One of the key developments is the rise of low-cost real-time tools that can analyze incoming text and look for keywords and phrases that are relevant to an organization. These processes include automatic classification, named entity extraction, and fact extraction. For example, if we typed in the term "yesterday," a system could detect the current date and insert a link to yesterday's date in your timeline so that you could see what notes referenced that date.

AQ3 When the NLP integration occurs in the editing workflow is also relevant. Real-time inferences over large NLP models can be very expensive. As-you-type checks need to execute quickly. Adding keywords to a document at the end of the day is a much lower cost. In general, real-time autosuggest services need to run in the 1/10th of a second range. These are 100 ms response times, and organizations that provide service-level agreements might have onerous charge-back fees that your user does not want to pay for.

Other integration points Integration with the in-charter-stream autosuggest and integration with real-time NLP analytics are the two areas that will dominate your PKG integration architectures and drive your PKG product decision points. However, there are a few other areas to consider.

Converting Markdown extensions Almost all PKGs today center on extending standard Markdown formats. Unfortunately, different vendors have picked different formats for these extensions. When you import or export Markdown between systems, you may need to add converters between these formats. For example, converting from

Obsidian to Roam or vice versa will require you to load converter extensions to do this work. Fortunately, the conversion scripts are pretty simple syntax changes and are well documented. Many small Python programs are already available to do these conversions. Things to check for include external links, image links, metadata tags, text highlighting, and aliases.

Section summary There will always be a natural tension between quick free-form notetaking unencumbered by rules and the desire for the organization to reuse knowledge and make it consistent. PKGs, wikis, and enterprise knowledge graphs are all evolving in conjunction with the explosion of NLP tools and the growth of large-language models that suggest text within the context of a text editor.

14.3 Steps in building an integrating personal knowledge graphs into the enterprise

In Section 14.3, the key concepts and challenges used in an integrated personal knowledge graph (IPKG), this section will explore the steps to build the integrated system into a large organization. Let us examine how these concepts impact your PKG rollout plan.

The IPKG hypothesis The basis for much of the logic around these IPKGs efforts is the following hypothesis:

IPKGs will allow users of PKGs to create more valuable and helpful knowledge than if note-taking is done with siloed tools.

The key to this hypothesis is to quantify what valuable and helpful means in metrics that even a finance person with no background in formal knowledge management can understand.

Align your IPKG benefits with organization strategy

There will undoubtedly be many individuals that will be ultra-enthusiastic about finally having an excellent note-taking tool. And integrating PKG with company entity type systems is a huge bonus. However, companies rarely spend millions of dollars to make their employee happy. Employee happiness is a pleasant side effect but should never be considered a primary driver for corporate PKG initiatives.

Here are just a few of the benefits IPKGs can bring to an organization:

1. sharing knowledge
2. avoiding duplication of knowledge
3. making knowledge consistent
4. improving collaboration between teams within a department
5. improving collaboration between departments or business units
6. streamlining communication
7. increasing knowledge worker productivity
8. improving search results and search ranking

The key is not just to state these benefits but to specifically tie these objectives to your organization's annual business strategy.

Find and executive sponsor

My experience is that most successful IPKG projects are driven by empowered staff with formal training in enterprise knowledge management strategies. Not all organizations have these teams, or if they do have them, they do not have the enterprise mandate or sponsorship of an influential executive. But to be successful, you will need this long-term sponsorship and funding. Integration takes time and money. You will need to find staff with the right skills and the ability to get access to essential data services. Your business alignment document will help you convince executives that your Integrated PKG project will help them achieve their strategic objectives.

Define metrics for success

One of the critical challenges of integrating organization knowledge with any text editor is to agree on standard measures of productivity improvement. Let us take a simple example: adding a list of industry-specific terms or company product names to an individual spell checker.

We all know how annoying it is to "train" every new text editor you use to recognize company-specific terms for the correct spelling. The methods for modifying the dictionaries are often hidden or unavailable to the user. For example, the chat feature on your Zoom or Teams program might not allow you to add custom terms. Some people with dyslexia or where English is not their native language depend on good spelling checks. They might be reluctant to engage in any chat when they are not confident the tools will help them look professional.

For example, you open the "compose" window in your e-mail editor. You start writing and using terms specific to your industry and organization are all marked as misspellings. These words can be the names of products you use or products you sell. For each word, you need to find a way to add this term to your dictionary. But what if the text editor could automatically be updated with words that your peers have already confirmed are correct spellings or valid acronyms for systems you work with daily?

Example: customize spell checking

The key is to capture the small productivity gains if your word processor is smart enough to gather terms from the IPKGs of your peers. It takes only a few seconds to add a word to the dictionary of a local word processor. But if you have thousands of terms and each of your 10,000 employees needs to add these words, then lost productivity is measured. You will need to estimate the cost/hour of your knowledge workers and use the best estimates of the increased productivity without the distractions of your text editor constantly telling you there are terms that it needs to understand. These are the types of metrics that can help you build a business case for IPKGs.

We can also extend this checking to include the correct use of the case in an official term. For example, we use "Facebook" but not "FaceBook" and "LinkedIn," not "Linkedin."

Automatic hyperlinking

Let us take this one step further. What if the first time we created a document, and a highly technical term was used, the word processor would automatically add a link to your company business glossary? Then your new employees could quickly jump to the right place to find that term and understand the definition of that term and related terms.

An excellent example is an ability to reference individuals directly on social media sites such as Facebook, Twitter, and LinkedIn. A confirmation list is presented by referencing a person starting with an "@" character. After you click on that item on the list, a link to that person's home page is automatically inserted into the text.

The key to making these features useful is to limit the scope of these selection lists. You might only refer to people in your social network and rank-order the list based on criteria such as if they are on the current discussion or how often you have mentioned them in the past. The scope and sorting rules can be contextual and complex. The scope and sorting rules illustrate the effort required to build scalable IPKG autosuggest algorithms.

Integrating simple productivity features into any note-taking tool and tracking the measurable productivity gains is often the first step in your IPKG journey. Once you have done this alignment and got a sign-off from an executive sponsor, you are ready to create a formal PKG integration plan.

Create an inventory of current knowledge repositories One of the critical challenges of building good IPKGs is understanding where related information and knowledge exist within your organization today. Once we have an inventory of these knowledge stores, we can ask what autosuggest-related services could be designed using existing APIs. You will also need to estimate how frequently a note-taker will want to reference these concepts in their documents and prioritize the most commonly used terms. Here is an example of some of the knowledge sources within a large organization:

1. Approved business terms from a centralized glossary of terms.
2. List of industry-specific words or phrases from medical or legal dictionaries.
3. Names of wiki pages that are the most frequently referenced in other wiki pages or tagged with a specific label.
4. List of products and product attributes in a product catalog.
5. List of the approved desktop applications or tools that employees are using.
6. List of computer systems or databases.
7. List of employee names (both first and last names), including familiar person names from other countries and languages used in your organization.

In summary, anytime your users add new terms to their local spell checkers, you can monitor them and look for trends. Keeping quality levels high is key to building trust in your IPKG initiatives. Add a human-centric quality-checking process whenever you see quality issues enter a system. Sometimes common misspellings can creep into a list of approved terms. A subject–matter expert should review new terms. Ideally, any incorrect business terms are removed before other teams use them. Make sure that users can quickly provide feedback to the team if they dispute the correctness of a word or phrase.

Building a reference architecture diagram After your IPKG team starts to inventory knowledge sources, you should create a single-page "map" showing how knowledge flows into your IPKG system and its new services. These data flow diagrams start

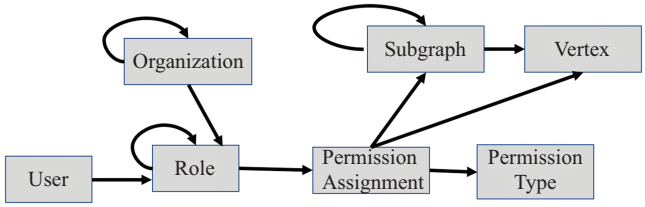


Figure 14.8 A high-level model of how roles are associated with permission assignments with a graph or vertex of a PKG

with a list of input sources on the left, your IPKG in the center, and consumers of this data on the right side.

14.4 Controlling access to IPKGs

Granting the proper access to knowledge to the right roles. Access control in a single person's PKG is simple. We have full rights to everything in our PKG; by default, no one else can see anything. This strategy makes the security design of most PKGs simple. Our PKG can easily link to external public knowledge and will be internally consistent. Figure 14.8 shows a high-level model of roles with permission assignment. The audience for your PKG is just yourself. You may not care about formatting rules, spelling checks, or consistency with organization standards. As you begin to share your knowledge with others, you may be more concerned with the spelling, formatting, and consistency with other knowledge. Before we dive into the security architecture, we need to define a few terms.

Basic graph security terminology

Let us start out with some basic terms and then progress to the more complex concepts.

- Each individual person creates one or more PKGs. Users always have full access to their own PKGs.
- PKGs consist of a set of named concepts represented as documents with distinct names and, optionally, a type. Each concept is called a vertex.
- PKGs may not have two concepts with exactly the same name. If you enter a duplicate concept, they can be renamed or merged together.
- The entirety of all the personal knowledge graphs is called the enterprise graph.
- Any part of the enterprise graph is called a subgraph.
- The public graph is a portion of the enterprise graph that all users can read. All users can always link to concepts in the public graph.
- Each user in an organization will be associated with one or more access roles in your organization.
- Access roles have access types such as read, rename, move, update, and delete.

- Users may also be granted access to subsections of the enterprise graph based on their role. This access is called role-based access control[24] or RBAC.
- Granting access to any resources for a role is called authorization.
- Verifying that a user is whom they say they are is called authentication. Most PKGs use either single-factor (password) or multi-factor authentication.
- The default access is the permissions that are applied when a user creates a new vertex in a graph. The default access may be different based on the location of a vertex in a subgraph or business rules.

User authentication

An enterprise will store a user's login credentials and policy in a centralized database. Most graph products use a protocol called LDAP to verify the user's credentials and, upon validation, send a list of the roles that users have to the application that logs them in. Because authentication processes are common to most databases, we do not spend much time on this topic here. Our focus is on granting access to the correct part of an enterprise group to a person.

Creation of shared subgraphs: Users may create a separate graph in their personal space and then grant access rights to individuals or roles within the organization. For example, you might create a subgraph for a specific project or team that works together. You can then grant full read and write permissions to other team members. This feature is easy to implement and available in most PKG products today, such as Roam Research.

The challenge is, what if some of your knowledge is in your personal space, and you want to share a read-only copy with a group of peers? You want them to be able to add relationships to your knowledge base but keep your foundational graph the same.

The ontology management problem: The ontology management problem requires many users to be able to change low-level ontologies but restricts change control on the upper levels of the ontology to those that understand the impact that these changes have on consumers of the ontology as shown in Figure 14.9. This type of problem comes up frequently in linked ontology design. Ontologies have multiple levels, such as upper ontology, middle ontology, and lower ontology. The higher up we go, the more lower-level components our changes might impact. Many small changes are frequently made to the lowest levels of ontology because a mistake can have a limited scope. Adding a new term, renaming a term, or fixing a typo in a name are examples of small changes. Changes to higher levels may ripple through many business processes that depend on consistent structures. To get around change-control concerns, enterprise-scale ontology management tools must allow you to make changes in a controlled environment and then run consistent regression tests that simulate the graph queries that downstream consumers will also run. Significant changes in upper levels of ontologies require an entirely new version of the ontology and allow downstream consumers to hold off updating their systems until they have modified their business logic to accommodate these changes. We introduced this example to demonstrate that personal notetaking is far from formal ontology management. The same knowledge

[24] https://en.wikipedia.org/wiki/Role-based_access_control

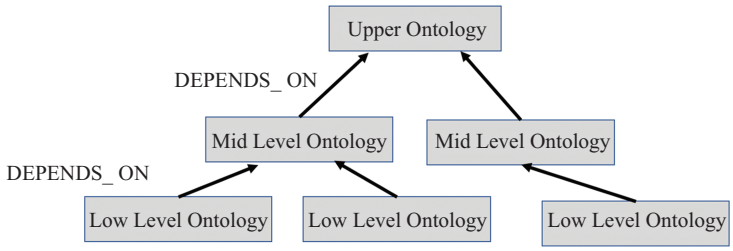


Figure 14.9 The ontology management problem

graph infrastructure can support both processes but having robust role-based access control is essential for high-stakes knowledge graph management.

Leveraging named groups

A better way to handle access to a subgraph is to grant access to a named group name such as "Team47". Then as individuals join or leave that team, we will not need to change our subgraph access permissions constantly. Because being able to modify the names in the group could impact access to the data, an approval process is often associated with adding new members to the group. Until now, the authorization problem has been similar to other database systems. What gets more complicated is when there are high-stakes vertices that need careful version control and testing.

Merging challenges

A typical scenario in PKG to EKG workflows is when informal note-talking evolves into formal knowledge bases designed to be used by a larger audience. A series of "Merge Conflicts" will naturally arise when this happens. If an author simply copies and pastes their page content into a shared graph, the links that used to work within your PKG may stop working. Although the links were consistent in your original PKG, a shared graph may not have access to the private concepts in your PKG.

After a user does the copy/paste, the editing tools might underline the broken links and suggest related links. If no matches are found, the user then has two options:

1. Remove the link
2. Copy the link destination page in your own personal graph
3. The latter option results in another problem. What if the new page you copied also has a set of broken links?

Moving knowledge from a private store to a shared subgraph is not trivial. It may be best to create the original document in the context of the shared group. Creating the new concept in a public subgraph avoids the broken links problem, but it adds a new challenge. What if your personal way of organizing knowledge diverges from how the group wants to organize knowledge?

Security summary

The bottom line is that there are no easy answers to the private-to-public merge problems. Migration of knowledge between subgraphs with different access rules must always deal with the link consistency problems. Role-based access control is

a mature design pattern that scales well over a large enterprise. The key is to avoid assigning permissions to individual users. Tying users directly to any resource results in much higher maintenance costs as users move around the organization.

14.5 Conclusion

Although current PKG software is designed for individual notetaking, the potential for leveraging the collective knowledge of all your organization's personal notes is immense. As these tools evolve with better APIs and robust role-based access control, the tools will be easier to integration into an enterprise knowledge management system.

Chapter 14

Integrating personal knowledge graphs into the enterprise

Author Queries

AQ1: Please confirm if "![](" is correct in the sentence "Just type a..."
AQ2: Please check the sentence "Both vertices and relationships ..." for completeness.
AQ3: Please check the sentence "When the NLP integraton occurs ..." for clarity.
AQ4: Please provide uncropped image for Figure 14.4.