Knowledge base & Rule engine

Contents

1 Definition

A knowledge base (KB) is a technology used to store complex structured and unstructured information used by a computer system. The initial use of the term was in connection with expert systems; which were the first knowledge-based systems.

Actually, a knowledge base is a database used for knowledge sharing and management.

It promotes the collection, organization, and retrieval of knowledge. Many knowledge bases are structured around artificial intelligence and not only store data but find solutions for further problems using data from previous experience stored as part of the knowledge base.

Knowledge management systems depend on data management technologies ranging from relational databases to data warehouses. Some knowledge bases are little more than indexed encyclopedic information; others are interactive and behave/respond according to the input prompted from the user.

2 Structured and Unstructured Knowledge Bases

Some knowledge bases can be quite 'unstructured', and the wiki is a good tool for this, with people adding knowledge topics freely, as they think fit. Many organizations find that the use of wiki's can spread rapidly throughout the organization.

Some knowledge bases need to be structured. For example a knowledge base for standard operating procedures in an organization, or a knowledge base for knowledge on good practices in the health sector, or legal topics, or customer knowledge etc. For structured knowledge bases, a process needs to be set up, and responsibilities assigned, for people to capture new learnings and ideas, as new knowledge nominations, for people to filter and edit these nominations, and for people to edit the knowledge topics. Some organizations even develop very complex knowledge bases based on their own innovative knowledge base processes.

So knowledge bases can be simple or complex, structured with simple or sophisticated knowledge processes, or unstructured, freely available on the web as wikis, for example, or developed as expensive proprietary software, depending on the needs of the organization.

3 Difference between a knowledge base and a database?

A database contains information that is structured in records, so that it can be sorted, categorized and accessed. Typically, a database is updated and maintained, centrally, by a database manager(s) or administrator(s). Typically, a database is centrally controlled and the information is 'one way', that is, from owner to user.

Databases first contained simple structured records of text and numbers. They then became more able to link to corresponding records as 'relational databases'.

In the 1980/1990 period, with the development of information management as a science, it became possible to populate databases with pictures and graphics, videos, tables, spreadsheets, and power point presentations etc The information became richer, even though it was still, typically, centrally managed and controlled. But, instead of calling them information bases, a term that never really caught on, we still tend to call them databases.

In the 1990/2000 period, with the development of collaborative team working tools, it became possible to create databases with far more collaborative team input, feedback and collaborative authoring. Centralization gave way to more 'participative development'. Furthermore, we learned how to better capture and store new learnings and ideas within these spaces, so that the knowledge base became more alive with 'continuous learning and ideas' and, even, 'continuous innovation'.

So, unlike a database, a knowledge base will typically develop knowledge as follows:

1. create new knowledge for a topic
2. expand the knowledge by discussions and feedback, new learnings and ideas
3. edit the expanded knowledge into better new knowledge
4. maintain history of revisions

In the context of knowledge management, these tools enable us to create knowledge bases, which are collaborative and participative databases that are structured to answer, for a given knowledge topic, the 'what, why, where, when, who and how' - the six components of knowledge.

4 Why knowledge base?

Before knowledge management tools and collaborative workspaces were available, people had to access centrally managed and controlled databases. New knowledge creation and knowledge sharing was based on the productivity of a few people in a central team, which, by comparison, is a slow process.

Knowledge bases now enable many more people in the organization to access, create, collaborate, develop new knowledge, more often as participants, to rapidly feed back and even create and edit new knowledge, where appropriate.

Knowledge bases give a full context for a knowledge topic by structuring the 'what, why, who, where, when, how'.

Knowledge bases, especially wiki's, do not normally require the involvement of the IT department, although their support is to be welcomed. This means that knowledge bases can be created rapidly by the users themselves.

5 How to use a Knowledge Base

5.1 Step 1 Identify what key area of knowledge you wish to better manage in a knowledge base.

Ideally, knowledge bases are most effective when they are used to better manage key knowledge areas. One way to identify a key knowledge areas is to:

Examine the organizational/business/project objectives that you wish to achieve

Ask the question 'What knowledge area(s), if we could better manage it, would make a big difference to our performance?

However, knowledge bases can, equally, also be very effectively used for each new project, or process, undertaken by the organization.

5.2 Step 2 Decide if the knowledge base is to be managed or open

Decide if this knowledge base needs to be managed by a knowledge base manager, or subject matter expert, to edit feedback and suggested knowledge improvements, or if it can be open to a wider audience to directly participate and edit themselves.

5.3 Step 3 Appoint a knowledge base manager

If it is to be a managed knowledge base, appoint a knowledge base manager (KB Manager) and develop the process to receive feedback, new learnings, new ideas, suggestions for improvement, measurements etc.

5.4 Step 4 Create the knowledge base

Consider using wiki's wherever possible, and the development of proprietary knowledge bases wherever there is a special need, beyond wiki functionality.

6 When to use knowledge bases

This question has already been partially answered, in the above description. Basically, wherever the need is to create new explicit knowledge and apply it, preferably as a team or collaborative community, collectively, there is a need for a knowledge base.

7 Where to use knowledge bases

Knowledge bases can be used anywhere in the organization. However, be cautious of 'over using' knowledge bases.

Think about the benefits of a knowledge base versus the costs, in financial terms, and in time and effort terms.

Think about the possible audiences and possible contributors.

8 Examples

8.1 Wikipedia

A wiki is one special type of knowledge base, with very powerful uses in an organization.

A wiki typically contains a page for each knowledge topic, and it contains a discussion page for each knowledge topic, and editing page for each knowledge topic, and a page to capture history of changes and revisions.

A wiki tends to be open to many/all to collaborate, develop and access new knowledge. The best example of a wiki is wikipedia, the encyclopedia created by mass collaboration throughout the world. Notice the four sections article, Talk, edit this page, and history.

For all types and sizes of organization, the wiki is an extremely powerful knowledge management tool for creating, maintaining and accessing knowledge bases. Since the introduction of the wiki technology in the early 2000's, many organizations have adopted the wiki for many of their knowledge bases. The wiki is a key knowledge management tool.

8.2 Wolframalpha

Another example is wolframAlpha. Users submit queries and computation requests via a text field. WolframAlpha then computes answers and relevant visualizations from a knowledge base of curated, structured data that come from other sites and books. The site "use[s] a portfolio of automated and manual methods, including statistics, visualization, source cross-checking, and expert review." The curated data makes Alpha different from semantic search engines, which index a large number of answers and then try to match the question to one.

9 Rule engine

9.1 What is a Rule Engine?

Two subparts of the expert system, knowledge base and rule engine.

Artificial Intelligence (A.I.) is a very broad research area that focuses on "Making computers think like people" and includes disciplines such as Neural Networks, Genetic Algorithms, Decision Trees, Frame Systems and Expert Systems. Knowledge representation is the area of A.I. concerned with how knowledge is represented and manipulated.

Expert Systems use Knowledge representation to facilitate the codification of knowledge into a knowledge base which can be used for reasoning, i.e., we can process data with this knowledge base to infer conclusions. Expert Systems are also known as Knowledge-based Systems and Knowledge-based Expert Systems and are considered to be "applied artificial intelligence". The process of developing with an Expert System is Knowledge Engineering.

Business Rule Management Systems build additional value on top of a general purpose.

Rule Engine by providing business user focused systems for rule creation, management, deployment, collaboration, analysis and end user tools. Further adding to this value is the fast evolving and popular methodology "Business Rules Approach", which is a helping to formalize the role of Rule Engines in the enterprise.

The term Rule Engine is quite ambiguous in that it can be any system that uses rules, in any form, that can be applied to data to produce outcomes. This includes simple systems like form validation and dynamic expression engines.

The book "How to Build a Business Rules Engine (2004)" by Malcolm Chisholm exemplifies this ambiguity.

9.2 High-level View of a Rule Engine

The high-level view of a Rule Engine is shown in Fig. 1. The Rules are stored in the Production Memory and the facts that the Inference Engine matches against are kept in the Working Memory. Facts are asserted into the Working Memory where they may then be modified or retracted.

A system with a large number of rules and facts may result in many rules being true for the same fact assertion; these rules are said to be in conflict.

The Agenda manages the execution order of these conflicting rules using a Conflict Resolution strategy.

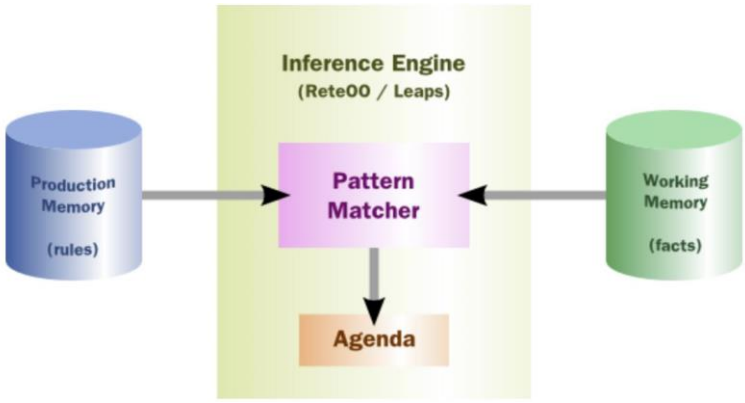

Fig. 1. High-level View of a Rule Engine

A Production Rule System is Turing complete, with a focus on knowledge representation to express propositional and first order logic in a concise, non-ambiguous and declarative manner. The brain of a Production Rules System is an Inference Engine that is able to scale to a large number of rules and facts. The Inference Engine matches facts and data against Production Rules - also called Productions or just Rules - to infer conclusions which result in actions. A Production Rule is a two-part structure

using First Order Logic for reasoning over knowledge representation as the Fig. 2. shows.

```
when
        <conditions>
then
        <actions>;
```

Fig. 2. A Production Rule

9.3 Pattern Matching

The process of matching the new or existing facts against Production Rules is called Pattern Matching, which is performed by the Inference Engine. There are a number of algorithms used for Pattern Matching by Inference Engines including:

- Linear
- Rete
- Treat
- Leaps

Execution methods for a rule system.

There are two methods of execution for a rule system: Forward Chaining and Backward Chaining; systems that implement both are called Hybrid Chaining Systems. Understanding these two modes of operation is the key to understanding why a Production Rule System is different and how to get the best from it.

Forward chaining is "data-driven" and thus reactionary, with facts being asserted into working memory, which results in one or more rules being concurrently true and scheduled for execution by the Agenda. In short, we start with a fact, it propagates and we end in a conclusion.

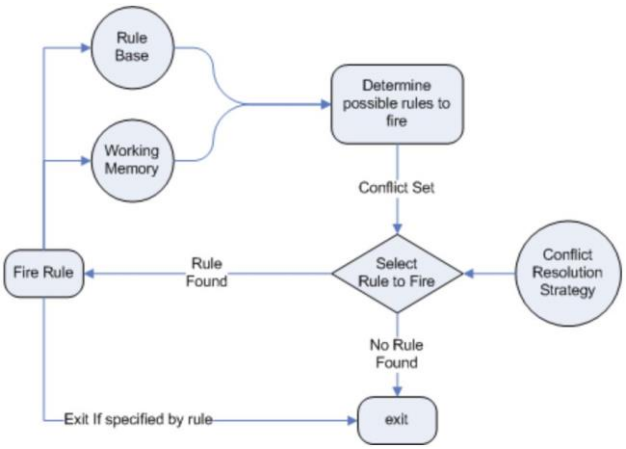

Fig. 3. Forward chaining

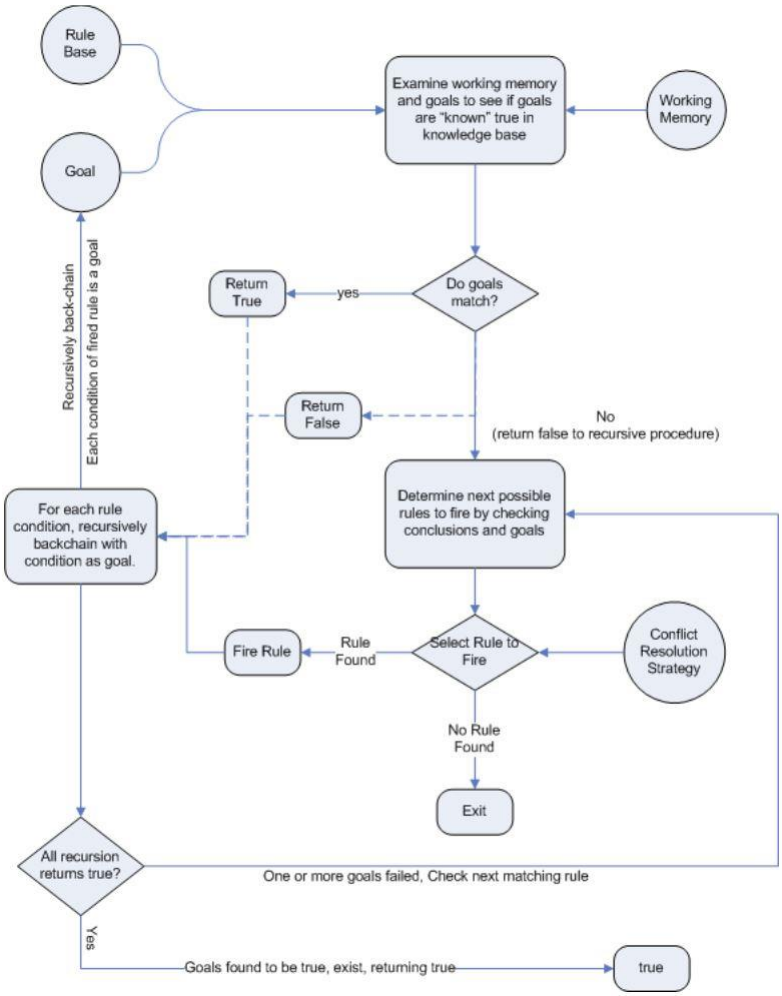

Fig. 4. Backward Chaining

Backward chaining is "goal-driven", meaning that we start with a conclusion which the engine tries to satisfy. If it can't it then searches for conclusions that it can satisfy; these are known as subgoals, that will help satisfy some unknown part of the current goal. It continues this process until either the initial conclusion is proven or there are no more subgoals. Prolog is an example of a Backward Chaining engine. Drools can also do backward chaining, which we refer to as derivation queries.

9.4 Advantages of a Rule Engine

Declarative Programming.

Rule engines allow you to say "What to do", not "How to do it".

The key advantage of this point is that using rules can make it easy to express solutions to difficult problems and consequently have those solutions verified. Rules are much easier to read than code.

Rule systems are capable of solving very, very hard problems, providing an explanation of how the solution was arrived at and why each "decision" along the way was made (not so easy with other of AI systems like neural networks or the human brain - I have no idea why I scratched the side of the car).

Logic and Data Separation.

Your data is in your domain objects, the logic is in the rules. This is fundamentally breaking the OO coupling of data and logic, which can be an advantage or a disadvantage depending on your point of view. The upshot is that the logic can be much easier to maintain as there are changes in the future, as the logic is all laid out in rules. This can be especially true if the logic is cross-domain or multi-domain logic. Instead of the logic being spread across many domain objects or controllers, it can all be organized in one or more very distinct rules files.

Speed and Scalability.

The Rete algorithm,the Leaps algorithm, and their descendants such as Drools' ReteOO, provide very efficient ways of matching rule patterns to your domain object data. These are especially efficient when you have datasets that change in small portions as the rule engine can remember past matches. These algorithms are battle proven.

Centralization of Knowledge.

By using rules, you create a repository of knowledge (a knowledge base) which is executable. This means it's a single point of truth, for business policy, for instance. Ideally rules are so readable that they can also serve as documentation.

Tool Integration.

Tools such as Eclipse (and in future, Web based user interfaces) provide ways to edit and manage rules and get immediate feedback, validation and content assistance. Auditing and debugging tools are also available.

Explanation Facility.

Rule systems effectively provide an "explanation facility" by being able to log the decisions made by the rule engine along with why the decisions were made.

Understandable Rules.

By creating object models and, optionally, Domain Specific Languages that model your problem domain you can set yourself up to write rules that are very close to natural language. They lend themselves to logic that is understandable to, possibly non-technical, domain experts as they are expressed in their language, with all the program plumbing, the technical know-how being hidden away in the usual code.

9.5 When should you use a Rule Engine?

when there is no satisfactory traditional programming approach to solve the problem:

- The problem is just too fiddle for traditional code.
- The problem may not be complex, but you can't see a non-fragile way of building a solution for it.
- The problem is beyond any obvious algorithmic solution.
- It is a complex problem to solve, there are no obvious traditional solutions, or basically the problem isn't fully understood.
- The logic changes often
- The logic itself may even be simple but the rules change quite often. In many organizations software releases are few and far between and pluggable rules can help provide the "agility" that is needed and expected in a reasonably safe way.
- Domain experts (or business analysts) are readily available, but are nontechnical.
- Domain experts often possess a wealth of knowledge about business rules and processes. They typically are nontechnical, but can be very logical. Rules can allow them to express the logic in their own terms. Of course, they still have to think critically and be capable of logical thinking. Many people in nontechnical positions do not have training in formal logic, so be careful and work with them, as by codifying business knowledge in rules, you will often expose holes in the way the business rules and processes are currently understood.

9.6 When not to use a Rule Engine

As rule engines are dynamic (dynamic in the sense that the rules can be stored and managed and updated as data), they are often looked at as a solution to the problem of deploying software. (Most IT departments seem to exist for the purpose of preventing software being rolled out.) If this is the reason you wish to use a rule engine, be aware that rule engines work best when you are able to write declarative rules. As an alternative, you can consider data-driven designs (lookup tables), or script processing engines where the scripts are managed in a database and are able to be updated on the fly.

9.7 Drools

Drools is a business-rule management system with a forward-chaining and backward-chaining inference-based rules engine, allowing fast and reliable evaluation of business rules and complex event processing.

Drools implements and extends the Rete algorithm; Leaps used to be provided but was retired as it became unmaintained. The Drools Rete implementation is called ReteOO, signifying that Drools has an enhanced and optimized implementation of the Rete algorithm for object oriented systems.

Drools engine.

The Drools engine is the rules engine in Drools. The Drools engine stores, processes, and evaluates data to execute the business rules or decision models that you define. The basic function of the Drools engine is to match incoming data, or facts, to the conditions of rules and determine whether and how to execute the rules.

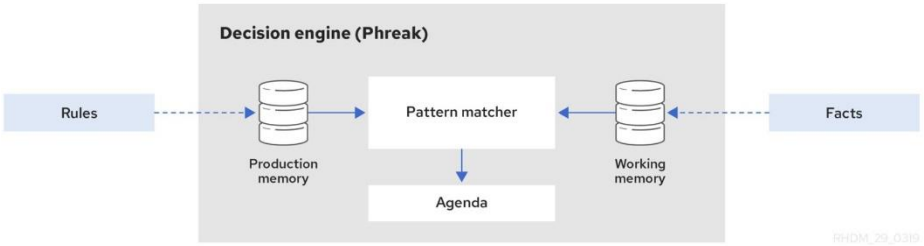

Fig. 5. The Drools engine

The Drools engine operates using the following basic components (Fig. 5.):

- Rules: Business rules or DMN decisions that you define. All rules must contain at a minimum the conditions that trigger the rule and the actions that the rule dictates.
- Facts: Data that enters or changes in the Drools engine that the Drools engine matches to rule conditions to execute applicable rules.
- Production memory: Location where rules are stored in the Drools engine.
- Working memory: Location where facts are stored in the Drools engine.
- Agenda: Location where activated rules are registered and sorted (if applicable) in preparation for execution.

When a business user or an automated system adds or updates rule-related information in Drools, that information is inserted into the working memory of the Drools engine in the form of one or more facts. The Drools engine matches those facts to the conditions of the rules that are stored in the production memory to determine eligible rule executions. (This process of matching facts to rules is often referred to as pattern matching.) When rule conditions are met, the Drools engine activates and registers rules in the agenda, where the Drools engine then sorts prioritized or conflicting rules in preparation for execution.