What is Rule-Engine?

Here I'm trying to explain rule-engine in a very simple way. Let's start with the problem. Suppose if I tell you that you have to build a bank application, where you have to implement the logic like mentioned below.

Logic:

A person is eligible for car loan **if**, he has monthly salary more than 70K and his credit score is more than 900 **then**, approve the car loan and sanction the 60% of requested amount.

You can easily implement these types of rules or logic in your application.

But If you will get some additional requirements like:

- If there are a large number of logics then, how you will search and apply them efficiently? (Good performance.)
- If logics are frequently changing and you generally code your logic in the application, then how you will manage or change the code that frequently? (Avoid frequent deployment.)
- Design the application such that, it can be easily maintained and understood by business people. (Use by non-technical members)
- If you have to keep your all business logic at a centralized place and separate from all the applications then, where you will keep it?

To achieve all these requirements in our application, we can use the **rule-engine**. But before starting the rules-engine, let's go through with few terminologies and background.

Terminologies:

- **Rule:** It is a set of the condition followed by the set of actions. It represents the logic of the system. The rules are mainly represented in the **if-then** form. It contains mainly two parts, **condition**, and **action**. The rule is also known as **production**.

  Rule = Condition + Action

  The condition also knows as a **fact or antecedents or patterns**. And action also knows as a **consequent**.

- **Human expert:** A person who is an expert in a corresponding business domain. This person provides knowledge in the form of rules.

For example:-The above-mentioned logic for a car loan is provided by a bank expert.

Knowledge in the form of rules:

Rule 1: A person is eligible for car loan?
if:
1. He has monthly salary more than 70K.
2. His credit score is more than 900.

then:
1. Approved car loan.
2. Sanctioned 80% of requested car loan amount.

Rule 2: A person is eligible for car loan?
if:
1. He has monthly salary more than 35K.
2. His credit score is more than 700.

then:
1. Approved car loan.
2. Sanctioned 60% of requested car loan amount.

- **Expert System:** It is a program that uses the knowledge of a human expert to solve the problems and giving a solution. It is also known as a **rule-based system or production system.**

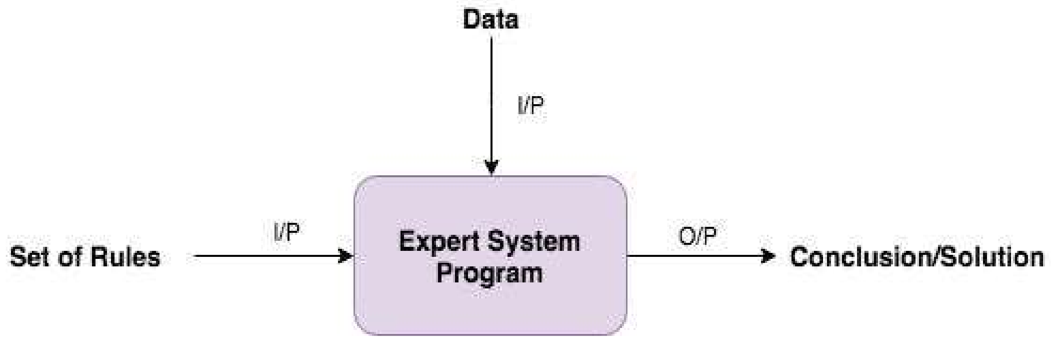

Fig: Expert System

- **Inference Engine:** It is a brain of expert system which manage a large number of rules and facts inside the expert system. Its job is picking rules and applying on data and generate a solution.

Now let's try to understand rule-engine.

Rule-Engine

It is an expert-system program, which runs the rules on the data and if any condition matches then it executes the corresponding actions.

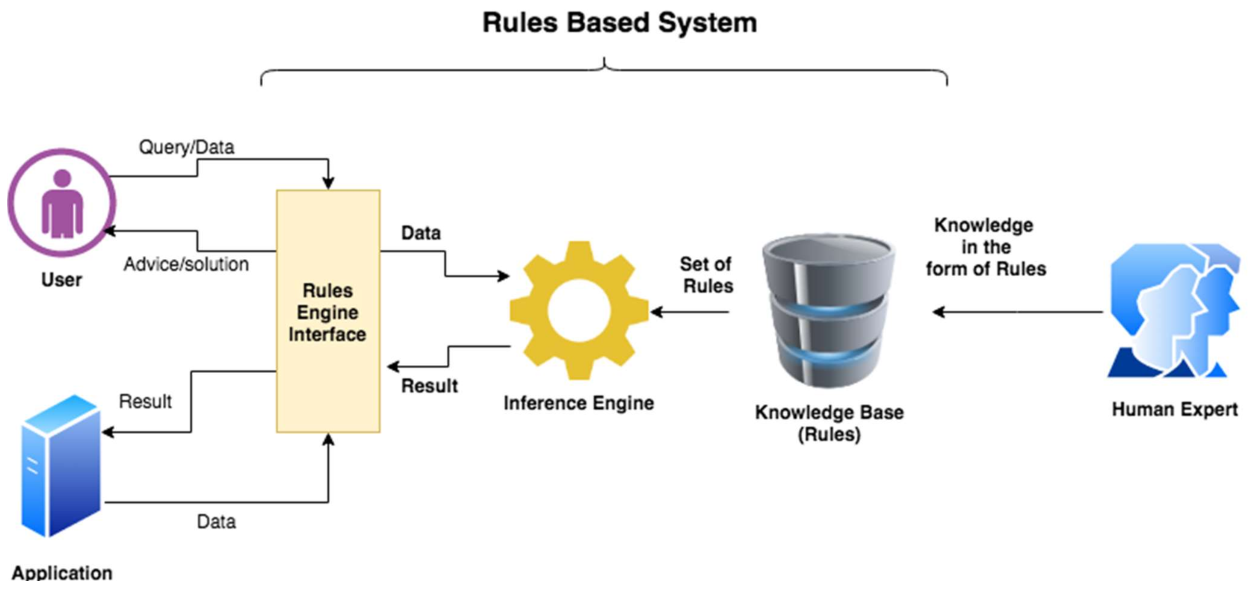

Fig: Rule-Engine

In the above diagram, it's showed that we collect knowledge in the form of rules (if-then form) and stored them in any store. The rules could be stored in any storage like files or databases. Now inference engine picks the rules according to requirements and runs them on input data or query. If any patterns/condition matches then it performs the corresponding action and returns the result or solution.

Inference-Engine

The inference engine is the component of the intelligent system in artificial intelligence, which applies logical rules to the knowledge base to infer new information from known facts. The first inference engine was part of the expert system.

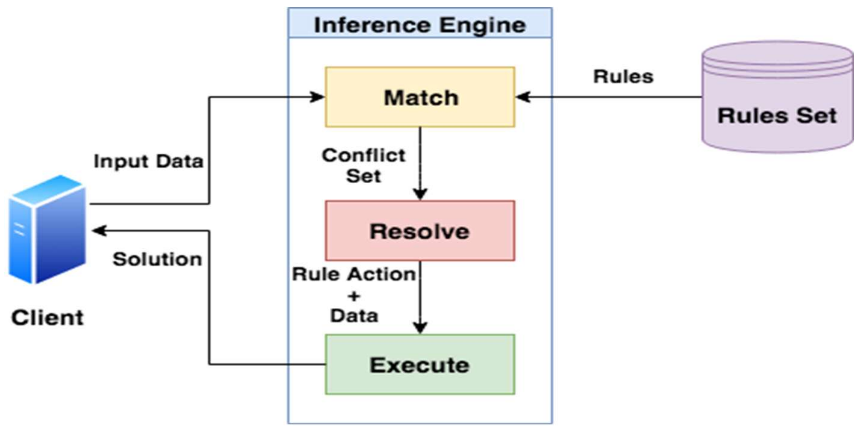

Fig: Inference-Engine

Inference-Engine's program works in three phases to execute the rule on given data.

Phase 1 — Match: In this phase, the inference engine matches the facts and data against the set of rules. This process called **pattern matching**.The output of the first phase is a **conflict set**.

- **Conflict set**: means, for the same fact or condition, it might be possible that more than one rule is satisfied. So, it returns the set of conflict rules.

Phase 2 — Resolve: In this phase, the inference engine manages the order of conflicting rules. It resolves the conflict and gives the selected one rules.

Phase 3 — Execute: In this phase, the inference engine simply runs the action of the selected rule on given data and return the output/result to the client.

Inference Methods

Rule engines generally use one of the following **inference methods** to implement an inference engine.

1. *Forward chaining*
2. *Backward chaining*

But before understanding the inference method, let's understand the ***reasoning***. There are two types of reasoning.

1. Goal-Directed/Backward Reasoning: It is working backward from the goal. Here we start from the main goal and then will go for sub-goals. So in goal-directed reasoning, if we want to achieve the main goal then we have to think that "*to achieve the main goal, what sub-goals we have to achieve."*

Example: If we plan for an evening out, and for this, we plan to go for a movie, outing, and dinner. Then ***evening out*** *is our main goal* and the ***movie, outing and dinner*** *are the sub-goals of the main goal*.

2. Data-Driven/Forward Reasoning: It starts with the available data and uses rules to extract more data until a goal is reached. Here we look at data and if we found some pattern then it performs respective action.

Example: Suppose we have to figure out the **colour of a pet named sunny** with given rules and data.

Rules:

1. If X has long legs and X eats Earthworms - Then X is a frog
2. If X has tail and X eats meat - Then X is a dog
3. If X is a frog - Then X is green
4. If X is a dog - Then X is yellow

Data:

1. sunny has long legs
2. sunny eats Earthworms

Here using given rules and data we can extract more data like:

1. sunny is a frog.
2. sunny is green.

So now let's discuss the Inference Methods:

Forward chaining:

- It is an implementation of *Forward Reasoning*.
- It is *data-driven*.
- Facts assert into the working memory.
- One or more rules could be concurrently true.

Backward chaining:

- It is an implementation of *Backward Reasoning*.
- It is goal-driven.
- Start with the conclusion (Goals) and if not found then search for sub-goals.

There is one more category called ***Hybrid chaining***. It is a combination of both forward and backward chaining.

Advantages of rule-engine :-

We can consider the all above specific requirements in the given example as the advantages of the rule engine.

1. Rules are very easy to read and code by any non-technical person like business analyst, client team, etc. Here you have to focus on "What to do", not "How to do".
2. You store your all rules at center storage. This means you have a central place where your all business rules and logic exists. It will be a source of truth for you.
3. Logic is managed separately from core application logic so it can be managed and reused.
4. In rule-engine, we use different pattern matching and conflict resolving algorithms, which give high performance.
5. For frequently changing requirements, we can easily update rules. No code changes are required.
6. The complexity of code is more if it contains many decision points. The rule engine can handle much better it because they use a consistent representation of business rules.
7. The different applications can use the same rule-engine for the same logic. It increases reusability.

Implementation of rule-engine

There are many implementations are available for rules-engine. Few are like:

- **Drools:** Drools is a Business Rules Management System (BRMS) solution. It is an Object-Oriented Rule Engine for Java.

References: -

- **https://medium.com/@er.rameshkatiyar/what-is-rule-engine-86ea759ad97d**