MC05 - Large Language Models and Agents

Short Course - JCD 2026

Bruno Menezes (LNCC), Eduardo Bezerra (CEFET/RJ)

Outline

From LLMs to LLM-based Agents

From Prompting Techniques to Interaction Patterns

Tool Calling

Didactic Material

- **Lecture notes:** detailed PDF (30 pages) with all concepts.
- **Slides:** concise and didactic version for class.
- **Jupyter notebooks:** hands-on examples to experiment.
- **GitHub repository:** `sbbd2025_course`

All resources are freely available.

From LLMs to LLM-based Agents

From LLMs to LLM-based Agents

An LLM is, in essence, a **probabilistic text generator**: it predicts the next token, given an input token sequence.

From LLMs to LLM-based Agents

An LLM is, in essence, a **probabilistic text generator**: it predicts the next token, given an input token sequence.
Yet at scale, LLMs exhibit reasoning-like abilities (planning, inference, language understanding).

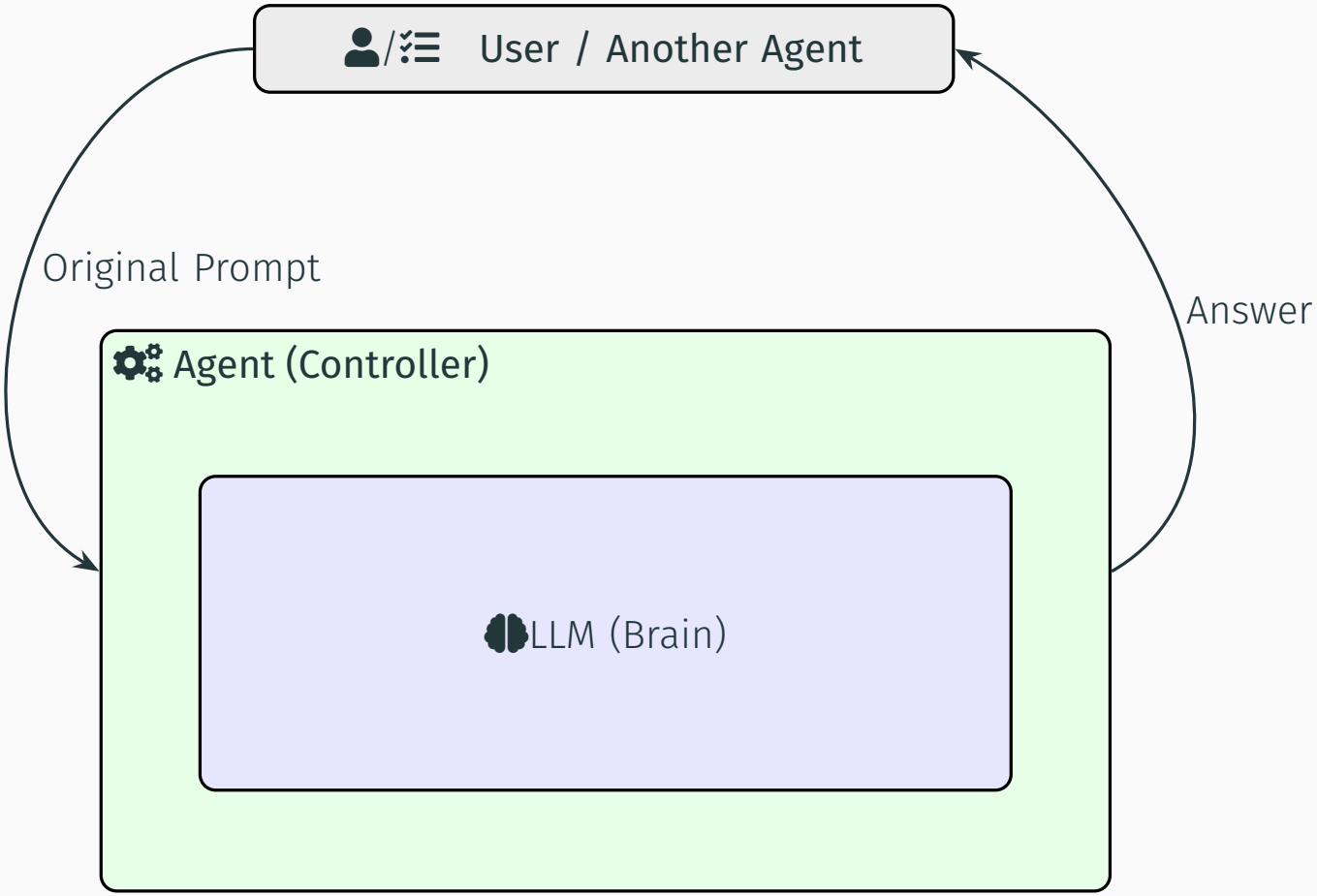
From LLMs to LLM-based Agents

An LLM is, in essence, a **probabilistic text generator**: it predicts the next token, given an input token sequence.
Yet at scale, LLMs exhibit reasoning-like abilities (planning, inference, language understanding).

An LLM-based agent combines:

- These **reasoning-like abilities** of the LLM,
- **Control logic** to orchestrate steps and manage context,
- Access to **external tools** (search, databases, APIs),
- Mechanisms for **perception and action**, so reasoning can affect the real world.

LLM-based Agent



Demo Time

From Prompting Techniques to Interaction Patterns

Prompting Techniques: Examples

Zero-shot

Prompt:

```
What is 47 + 35?
```

Output:

```
82
```

Few-shot

Prompt:

```
Examples:
12 + 7 = 19
5 + 9 = 14
23 + 18 = 41

Now, what
is 47 + 35?
```

Output:

```
82
```

Chain-of-Thought

Prompt:

```
What is 47 + 35?
Let's think
step by step.
```

Output:

```
First, add tens:
40 + 30 = 70
Then, ones:
7 + 5 = 12
Sum:
70 + 12 = 82
Answer: 82
```

From Prompting to Patterns

- Prompting techniques (Zero-shot, Few-shot, Chain-of-Thought) steer a **single LLM call**:
- But complex tasks require multi-step reasoning and tool use.

From Prompting to Agents

Examples of complex tasks that require multi-step reasoning and tool use:

- Answering questions over a large knowledge base (search + reasoning)
- Text-to-SQL (NL → SQL query → execution → formatted result)
- Planning a trip (dates, flights, hotels, budget)
- Solving math word problems (step-by-step reasoning + calculation)
- Data analysis (retrieve data → transform → summarize)

Interaction Patterns

Multi-step reasoning and tool use demand the use of **interaction patterns**.

Interaction Patterns

Multi-step reasoning and tool use demand the use of **interaction patterns**.

In the context of LLMs, an interaction pattern refers to the structured way a user or a system communicates with a model to achieve a specific outcome. Rather than just asking a single question, these patterns define the ”flow” of reasoning, the role the AI plays, and how it processes information to reach a solution.

Interaction Patterns

- **Interaction patterns** structure reasoning + action.
- Ensure consistency and integration with external tools.

Interaction Patterns

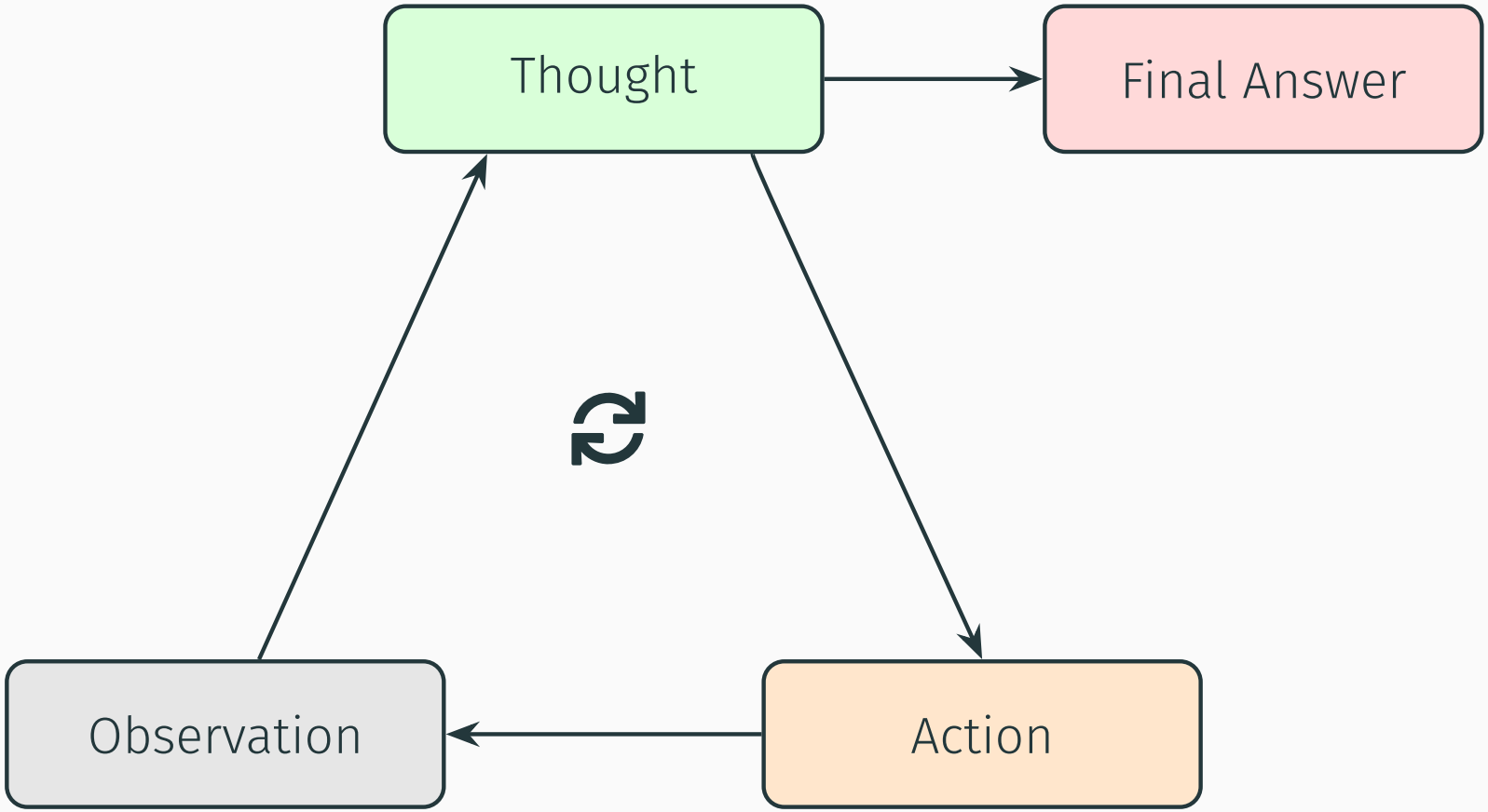
- **Interaction patterns** structure reasoning + action.
- Ensure consistency and integration with external tools.
- Examples:
  - ReAct (Reason + Act)
  - Plan-and-Act
  - Reflexion

Interaction Patterns

- **Interaction patterns** structure reasoning + action.
- Ensure consistency and integration with external tools.
- Examples:
  - ReAct (Reason + Act)
  - Plan-and-Act
  - Reflexion
- We focus on ReAct.

ReAct: Synergizing Reasoning and Acting in Language Models [**?**]

ReAct Pattern



The agent **thinks**, **acts**, and **observes** in a loop,
until enough information is gathered to deliver the final answer.

Example: Finding Payroll for Sales

- Scenario: A user asks for the total payroll expense of the Sales department for “last month”.

Example: Finding Payroll for Sales

- Scenario: A user asks for the total payroll expense of the Sales department for “last month”.
- Why this is non-trivial: “Last month” is *relative* and depends on the current date.

Example: Finding Payroll for Sales

- Scenario: A user asks for the total payroll expense of the Sales department for “last month”.
- Why this is non-trivial: “Last month” is *relative* and depends on the current date.
- Data requirement: We must query a payroll/financial database filtered by
  - month + year (resolved from “last month”)
  - department = `Sales`

Example: Finding Payroll for Sales

- Scenario: A user asks for the total payroll expense of the Sales department for “last month”.
- Why this is non-trivial: “Last month” is *relative* and depends on the current date.
- Data requirement: We must query a payroll/financial database filtered by
  - month + year (resolved from “last month”)
  - department = `Sales`
- Goal: Return a single, clear number (the total payroll expense) with the correct time reference.

Example: Finding Payroll for Sales

The next slide shows a step-by-step trace (Thought $\rightarrow$ Action $\rightarrow$ Observation) ending in the final answer.

Example: Finding Payroll for Sales

- Thought: I need to find the payroll data for the Sales department. First, I need to know which month was ”last month” relative to today (February 2026).

Example: Finding Payroll for Sales

- Thought: I need to find the payroll data for the Sales department. First, I need to know which month was ”last month” relative to today (February 2026).
- Action: get_current_date()

Example: Finding Payroll for Sales

- Thought: I need to find the payroll data for the Sales department. First, I need to know which month was ”last month” relative to today (February 2026).
- Action: get_current_date()
- Observation: Today is February 10, 2026. ”Last month” is January 2026.

Example: Finding Payroll for Sales

- Thought: I need to find the payroll data for the Sales department. First, I need to know which month was ”last month” relative to today (February 2026).
- Action: get_current_date()
- Observation: Today is February 10, 2026. ”Last month” is January 2026.
- Thought: Now I need to query the financial database for January 2026 payroll expenses specifically filtered for the ”Sales” department.

Example: Finding Payroll for Sales

- Thought: I need to find the payroll data for the Sales department. First, I need to know which month was ”last month” relative to today (February 2026).
- Action: get_current_date()
- Observation: Today is February 10, 2026. ”Last month” is January 2026.
- Thought: Now I need to query the financial database for January 2026 payroll expenses specifically filtered for the ”Sales” department.
- Action: query_payroll_db(month=”January”, year=2026, department=”Sales”)

Example: Finding Payroll for Sales

- Thought: I need to find the payroll data for the Sales department. First, I need to know which month was ”last month” relative to today (February 2026).
- Action: get_current_date()
- Observation: Today is February 10, 2026. ”Last month” is January 2026.
- Thought: Now I need to query the financial database for January 2026 payroll expenses specifically filtered for the ”Sales” department.
- Action: query_payroll_db(month=”January”, year=2026, department=”Sales”)
- Observation: Result: $142,500.00.

Example: Finding Payroll for Sales

- Thought: I need to find the payroll data for the Sales department. First, I need to know which month was ”last month” relative to today (February 2026).
- Action: get_current_date()
- Observation: Today is February 10, 2026. ”Last month” is January 2026.
- Thought: Now I need to query the financial database for January 2026 payroll expenses specifically filtered for the ”Sales” department.
- Action: query_payroll_db(month=”January”, year=2026, department=”Sales”)
- Observation: Result: $142,500.00.
- Thought: I have the information. I can now answer the user.

Example: Finding Payroll for Sales

- Thought: I need to find the payroll data for the Sales department. First, I need to know which month was ”last month” relative to today (February 2026).
- Action: get_current_date()
- Observation: Today is February 10, 2026. ”Last month” is January 2026.
- Thought: Now I need to query the financial database for January 2026 payroll expenses specifically filtered for the ”Sales” department.
- Action: query_payroll_db(month=”January”, year=2026, department=”Sales”)
- Observation: Result: $142,500.00.
- Thought: I have the information. I can now answer the user.
- Final Response: The total payroll expense for the Sales department in January 2026 was $142,500.00.

Example: Finding Payroll for Sales

But how the agent knows what tools it can invoke?!

Example: Finding Payroll for Sales

But how the agent knows what tools it can invoke?!

And how the agent knows how and when to invoke them?!

Example: Finding Payroll for Sales

But how the agent knows what tools it can invoke?!

And how the agent knows how and when to invoke them?!

To answer these questions, we need to talk about the mechanism of tool calling.

Tool Calling

Tool Calling

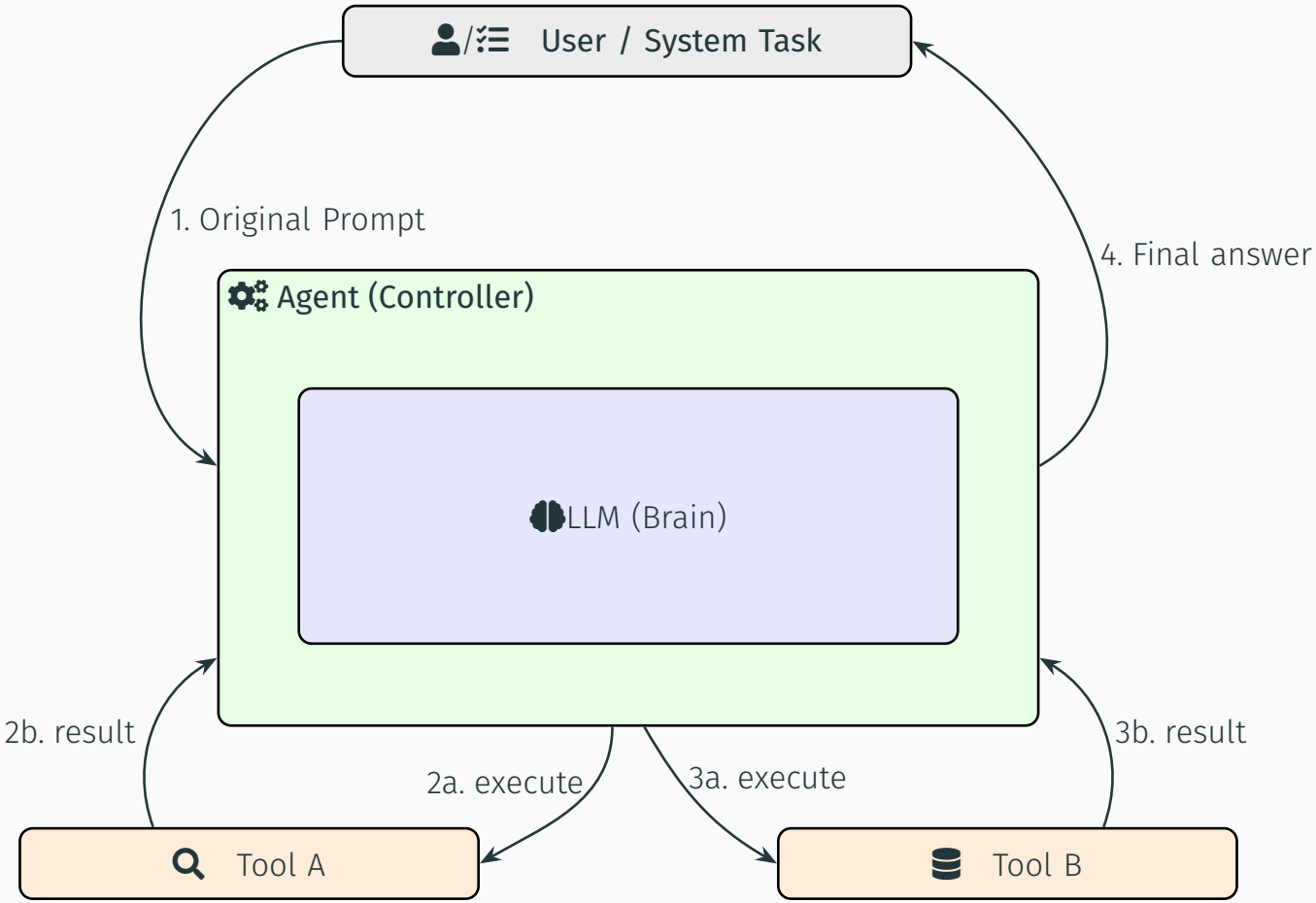
Multi-step tasks often require **external actions**, not just text generation.

Tool Calling

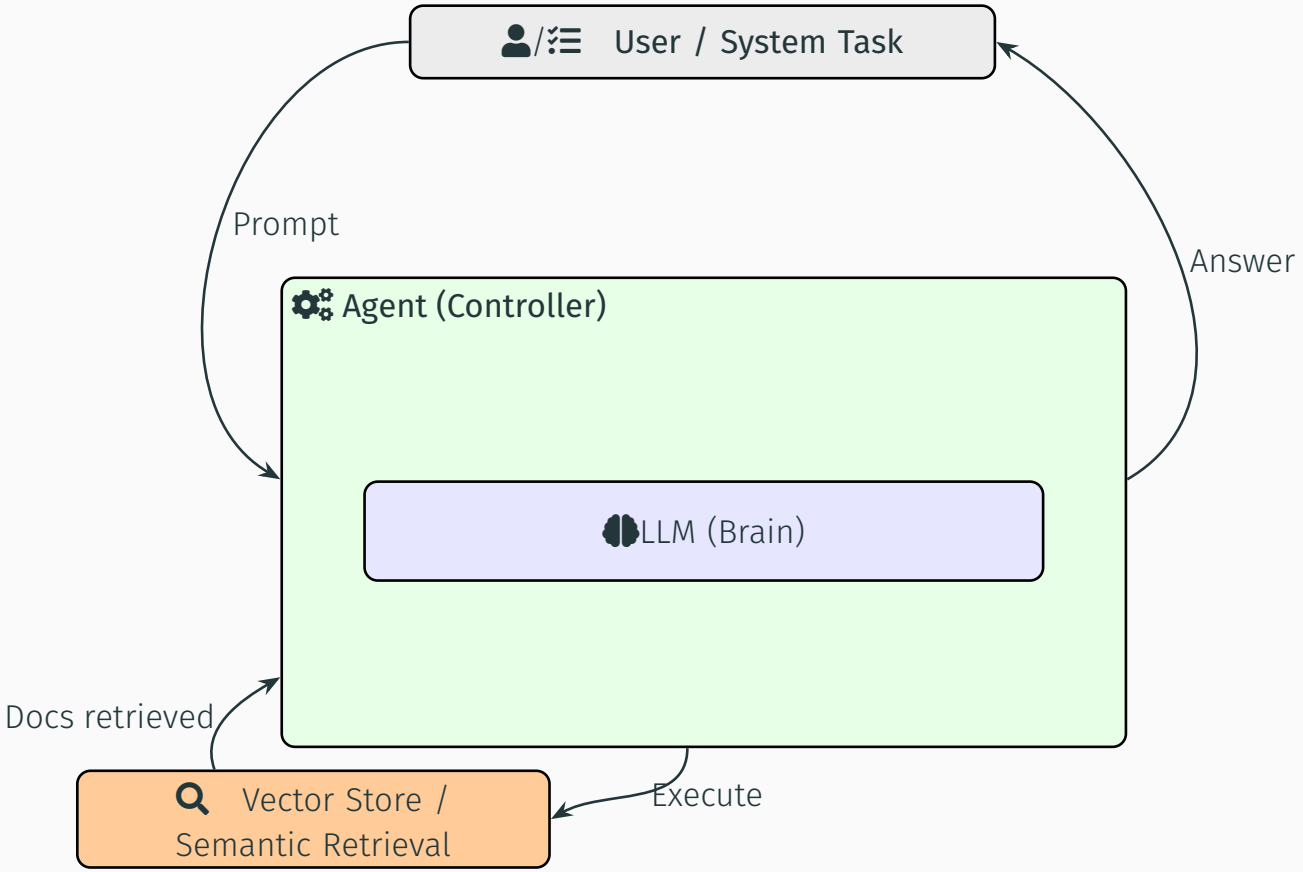
Multi-step tasks often require **external actions**, not just text generation.

In the context of LLMs, tool calling refers to the capability of a model to invoke external functions (tools) — such as APIs, databases, calculators, or search engines — as part of its reasoning process. Instead of relying only on internal knowledge, the model can retrieve fresh data, perform computations, and interact with systems to produce grounded and verifiable answers.

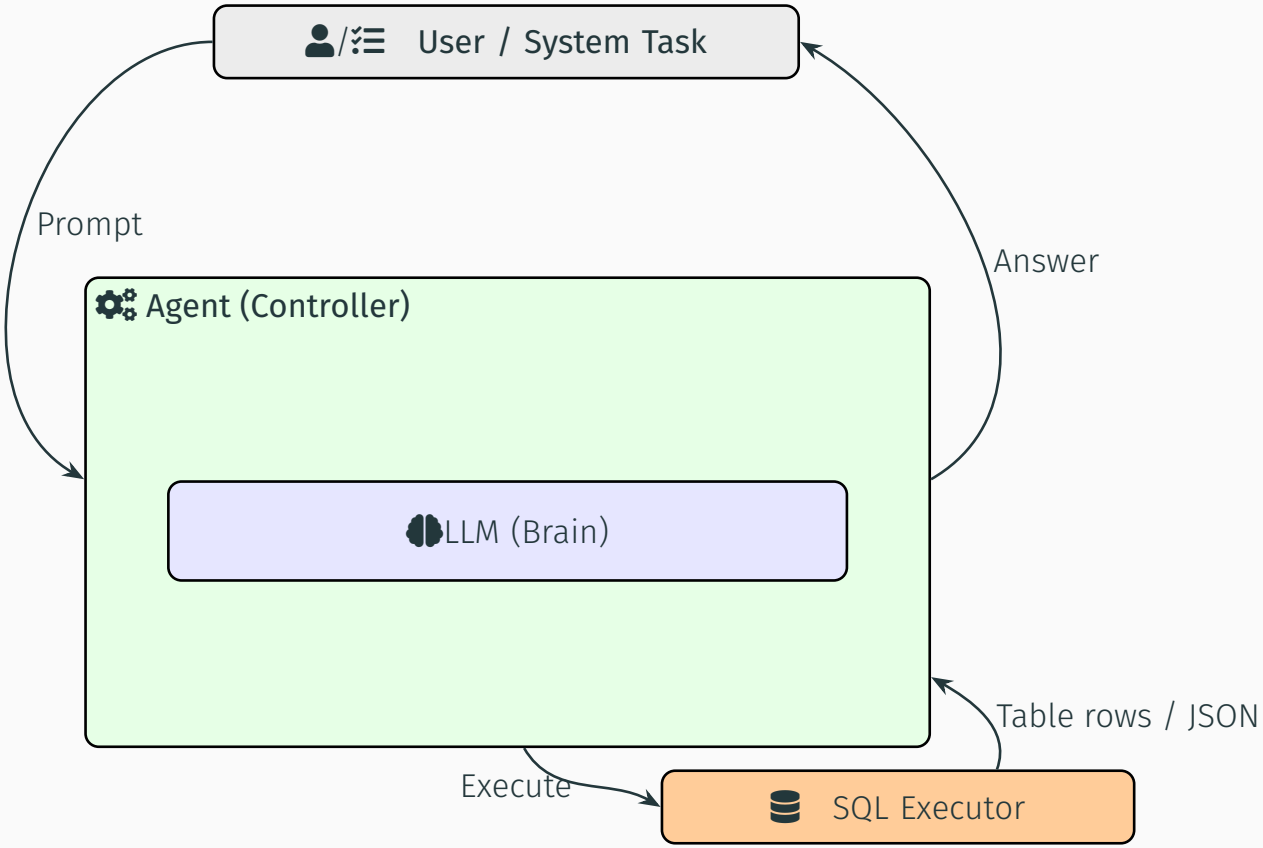
Tool Calling



Tool Calling Instance: RAG



Tool Calling Instance: Text-to-SQL



Tool Registration?

- LLMs do not “magically” know what tools exist.
- Each tool must be **registered** via a structured description that includes:
  - **Name** and **purpose** of the tool,
  - **Input parameters** (types, constraints, defaults),
  - **Expected outputs**.
- This information is added to the LLM context, so the model can decide when and how to use the tool.
- Tool registration is the gateway for RAG, Text-to-SQL, and many other applications.

Tool Registration?