Data Management for Agentic Memory: Foundations, Systems, and Challenges

Guoliang Li
Tsinghua University
liguoliang@tsinghua.edu.cn

Jiaqi Tian
Tsinghua University
tianjq25@mails.tsinghua.edu.cn

Xuanhe Zhou
Shanghai Jiao Tong University
zhouxuanhe@sjtu.edu.cn

ABSTRACT

Recent advances in large language models (LLMs) have led to the emergence of autonomous agents as a transformative paradigm for building intelligent AI systems, integrating reasoning, planning, tool use, and interaction capabilities to tackle complex, open-ended tasks. Despite their growing sophistication, most agents remain stateless, lacking the ability to retain, organize, and reuse past experiences, which limits their personalization, long-term adaptation, and lifelong learning. To address this limitation, agentic memory has emerged as a critical area of research, equipping agents with mechanisms to store, retrieve, update, and exploit information from prior interactions, reasoning trajectories, tool executions, and environmental feedback. From a data management perspective, agentic memory shares similarities with database systems but also presents opportunities for integration, enhancing query functionality, bridging structured and unstructured data, and optimizing execution efficiency through historical insights. This synergy between agentic memory and databases amplifies the capabilities of autonomous agents and redefines the role of databases in adaptive, intelligent data systems. In this tutorial, we systematically review agentic memory systems from a data management perspective. We introduce the workflow and architecture of agentic memory systems. We summarize their operators, storage, and optimization techniques. We also highlight open research challenges, aiming to inspire further innovation and progress in this exciting field.

PVLDB Reference Format:
Guoliang Li, Jiaqi Tian, and Xuanhe Zhou. Data Management for Agentic Memory: Foundations, Systems, and Challenges. PVLDB, 19(12): 4888 - 4892, 2026.
doi:10.14778/3827998.3828145

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/TsinghuaDatabaseGroup/AgenticMemory.

1 INTRODUCTION

Recent advances in large language models (LLMs) have driven the rapid emergence of autonomous agents as a transformative paradigm for building intelligent AI systems [22, 25, 28]. By integrating reasoning, planning, tool use, and interaction capabilities, these agents have shown immense potential in addressing complex, open-ended tasks [21, 24, 35]. However, despite their growing sophistication, most existing agents remain fundamentally stateless [26, 30] – processing each task or interaction in isolation, with little ability to retain, organize, or reuse past experiences over time. This lack of memory severely limits their ability to personalize, adapt over the long term, and engage in lifelong learning [4, 12]. Without mechanisms to accumulate experiences, reflect on past failures, or refine behavior across interactions, their potential for true autonomy remains constrained. To overcome this limitation, *agentic memory* has emerged as a critical area of research [8, 32]. Agentic memory equips agents with the ability to store, retrieve, update, and utilize information from prior interactions, internal reasoning trajectories, tool executions, and environmental feedback [27]. This enables agents to build a dynamic repository of knowledge, moving far beyond simple context buffers or append-only logs.

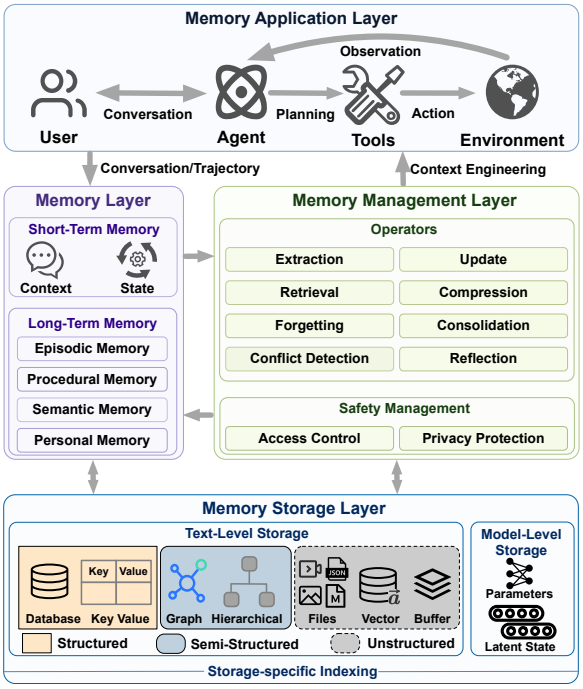

Figure 1: Overview of Data Management for Agentic Memory.

From a systems perspective, agentic memory shares striking parallels with database systems. Both are responsible for storing, organizing, retrieving, updating, and maintaining information to support reasoning and decision-making in dynamic environments. However, integrating agentic memory with database systems introduces a transformative opportunity [34]. While databases excel at storing and indexing structured data, the rise of LLMs and semantic analytics demands more flexible and dynamic memory management. Agentic memory complements databases by enhancing query functionality – enabling natural language queries with contextual

Proceedings of the VLDB Endowment, Vol. 19, No. 12 ISSN 2150-8097.
doi:10.14778/3827998.3828145

memory, dynamic query plan refinement, and semantic reasoning across structured and unstructured data sources. Additionally, it improves execution efficiency by leveraging historical insights to optimize query pipelines and reduce computation costs.

The integration of agentic memory and databases not only amplifies the capabilities of autonomous agents but also redefines the role of databases in adaptive, intelligent data systems. This synergy highlights a pressing need for the database community to advance agentic memory techniques. In this tutorial, we introduce the foundational techniques of agentic memory and aim to inspire further innovation at this exciting frontier. We systematically examine agentic memory systems from a data management perspective, as illustrated in Figure 1, with the goal of providing participants with a comprehensive understanding of the field and highlighting key research challenges for advancing agentic memory capabilities.

Tutorial Overview We will give a 1.5-hour tutorial.

Part I: Agentic Memory Workflow (20 minutes). The first part presents a system view of agentic memory. We describe how the memory system handles conversation and trajectory, constructs memory-augmented context for response generation, and maintains stored memories through a set of lifecycle operators. This perspective highlights agentic memory as a dynamic mechanism for organizing, updating, and reusing information over time, rather than a static store of interaction history.

Part II: Memory Categorization. (10 minutes). The second part organizes agentic memory into short-term and long-term components, and further decomposes long-term memory into episodic, procedural, semantic, and personal memory. This categorization clarifies memory functions and lays the foundation for discussing memory management and storage mechanisms.

Part III: Memory Management. (30 minutes) The third part covers two main aspects of agentic memory management: memory operators and safety management. For memory operators, we focus on existing implementation techniques of these operators. We then discuss access control and privacy protection in safety management, which are increasingly important as memory becomes persistent, shareable, and reusable across users and agents.

Part IV: Memory Storage. (15 minutes) The fourth part reviews the major storage formats used in existing systems, covering textual memory (structured, semi-structured, and unstructured) and model-level memory.

Part V: Existing Agentic Memory Systems. (10 minutes) We provide a comprehensive discussion of existing memory systems and summarize their techniques, including Mem0, MemOS, etc.

Part VI: Open Problems. (5 minutes) We conclude the tutorial by discussing open challenges and unresolved problems.

Target Audience and Learning Outcomes. The intended audience includes VLDB attendees from both academia and industry who are interested in data management and agentic memory. Our tutorial will be self-contained and accessible.

Related Tutorials. There has been no related tutorial in database conferences, and this is the first of its kind. We emphasize that **agentic memory is fundamentally a data management problem** and advocate for the database community to pay attention to this field and explore the open challenges in this exciting area.

2 TUTORIAL OUTLINE

2.1 Agentic Memory Workflow

Memory Input. During interactions between a user and an agent, a significant amount of information is generated, some of which may be valuable for future interactions and is worth recording in memory. When a user provides input, it is first routed through the agentic memory system instead of being sent directly to the agent. We define the external exchanges between the user and the agent as the **conversation**, and the agent's internal execution states – such as intermediate reasoning, tool calls, task progress, and environmental feedback – as the **trajectory**. These two types of information serve as the primary inputs to the memory system.

Memory Output. The raw input to the memory system is temporarily stored in *short-term memory* as part of the current interaction context [11, 14]. This input is asynchronously transformed into *long-term memory* using memory operators. The memory system then retrieves relevant memories from both short-term and long-term memory [16, 33]. The retrieved information are integrated with the original input using context-engineering techniques, creating a memory-augmented context. This enriched context is then passed to the agent, ensuring that the response generation is informed by both the current input and relevant historical memories.

Memory Maintenance. Without proper management, agentic memory can become inefficient, inconsistent, and overloaded due to the accumulation of redundant, outdated, or conflicting information. To address these challenges, memory management employs various operators to extract key information from inputs, as well as to insert, update, delete, and organize memories efficiently. These operators work together to ensure that agentic memory remains functional, organized, and aligned with the needs of the user-agent interaction.

Given a conversation or trajectory, an **extraction** operator is first applied to identify and derive *atomic memories* from short-term memory. An *atomic memory* is the fundamental storage unit in agentic memory systems – a minimal, self-contained piece of information extracted from a conversation or trajectory that represents a single, reusable data point. Next, the memory system applies a **retrieval** operator to search the existing memory store for related memories. Based on the retrieved results, an **update** operator is invoked to determine how the newly extracted atomic memories should be incorporated into the memory store. The decision is guided by **conflict detection** between the new atomic memories and retrieved ones. Depending on the results of the conflict detection, the system selects an appropriate strategy, such as inserting the new memory, merging it with existing ones, or deleting redundant or inconsistent memories [2, 30]. To maintain the coherence and consistency of the memory system, the conflict detection operator is periodically applied across the entire memory space to identify and resolve any inconsistencies. Once inserted into the memory store, memories enter a long-term maintenance stage. The **consolidation** operator organizes existing memory items into coherent structures by identifying, establishing, and maintaining explicit relationships among them. The **forgetting** operator removes memories that are no longer necessary, thereby limiting memory growth and preventing the accumulation of outdated or low-value information [33]. For memories that are less important but still above the removal threshold, a **compression** operator

can condense their content, reducing storage while retaining essential information [26]. A **reflection** operator analyzes existing memories to uncover latent relationships and derive higher-level knowledge [9]. This process transforms isolated memory items into more structured, generalized, and reusable representations.

2.2 Memory Categorization

According to CoALA [20], agentic memory can be categorized into short-term memory [7] and long-term memory, based on the temporal scope and functional role of stored information.

2.2.1 Short-Term Memory. Short-term memory, often aligned with the concept of conversational context, primarily stores information relevant to the ongoing user-agent interaction. This includes dialogue context, the agent's current task state, the progression of active tasks, intermediate reasoning steps, tool invocation states, and other transient execution signals. By retaining this information, short-term memory allows the agent to construct memory-augmented contexts that capture the critical elements of the immediate interaction. This ensures the agent does not lose track of the current context and supports the generation of coherent and contextually appropriate responses. Short-term memory is typically volatile, meaning it is continuously updated as the interaction evolves. It serves as a temporary workspace for the agent, facilitating real-time processing and responsiveness during the interaction.

2.2.2 Long-Term Memory. Long-term memory stores information that persists across interactions, enabling cross-session reasoning, knowledge accumulation, and personalized adaptation. Drawing from the classical distinctions in cognitive science, long-term memory is further categorized into episodic memory, procedural memory, and semantic memory. Additionally, personal memory is a distinct category focused on storing user-specific information.

Episodic Memory. Episodic memory records concrete interaction episodes and execution trajectories, capturing what occurred in specific situations and how the agent responded [1, 16]. It provides a basis for experience-driven reasoning and supports the retrieval of past outcomes to inform current decision making.

Procedural Memory. Procedural memory stores reusable task procedures, behavioral patterns, and action policies [7, 15, 27]. By encoding procedural rules and operational strategies, procedural memory enables the agent to perform tasks efficiently and replicate learned behaviors across different contexts.

Semantic Memory. Semantic memory encodes abstracted knowledge, facts, and generalized concepts acquired from interactions [2, 18, 33]. By serving as a durable repository of world knowledge and domain-specific concepts, it supports accurate concept understanding and mitigates errors or drift over multiple interactions.

Personal Memory. Personal memory is a dedicated memory module that retains user-specific information, such as preferences, habits, interaction patterns, and long-term constraints [5, 11, 33]. It enables long-term personalization by supporting user-aware reasoning.

2.3 Memory Management

2.3.1 Memory Operators. The operators govern memory management and collectively maintain the lifecycle (see Table 1).

Extraction. Early agentic memory systems often adopt raw trace logging, directly storing unprocessed interaction traces, typically in file systems [14, 16, 33]. This strategy introduces substantial storage overhead and makes memory retrieval difficult. To improve usability, subsequent approaches prompt LLMs to summarize the raw input into salient facts, decisions, or task-relevant information [2, 3]. More structured methods further define explicit memory schemas and extract typed fields or relations, enabling more effective memory management, retrieval, and downstream reasoning [2, 16].

Table 1: Summary of operators and key techniques.

Operators	Key Techniques
Extraction	Raw trace logging, LLM-based summary, structured extraction.
Update	Append-only, rule-driven, LLM-based, RL-based.
Retrieval	① Retrieval methods: text-based, structure-based. ② Post-retrieval processing: filtering, re-ranking, etc.
Consolidation	Similarity index, typed relation, hierarchical relation.
Forgetting	① Trigger conditions: capacity constraint, retention threshold. ② Utility-based deletion: temporal, usage, importance, validity.
Compression	Single-item compression, multi-item compression
Conflict Detection	Rule-based, LLM-based judgment on similar pairs.
Reflection	Abstraction: declarative knowledge, procedural knowledge.

Update. A basic strategy is append-only, in which newly extracted memories are directly appended to the memory store. Instead, some approaches first check whether a memory item is a duplicate of an existing one. If a duplicate is detected, the system uses the new observation to strengthen or refresh the existing memory [29]. This design primarily reduces redundancy but does not explicitly address contradictions between new and old memories. A more advanced approach treats an update as a conflict-aware write operation. The system first invokes a conflict detection operator. Based on the detection results, the system applies an appropriate update action.

Conflict Detection. To detect conflicts, the system periodically compares semantically related memories, followed by a consistency check. This check is typically performed through rule-based validation or LLM-based judgment [2, 30]. Once conflicts are identified, the system invokes the update operator to resolve them.

Retrieval. The implementation of the retrieval operator is highly dependent on the storage model (see Section 2.4). When memories are stored in structured formats, retrieval can leverage the explicit organization of the underlying data model. Database-based storage supports SQL-style querying [6], while graph-based storage enables relation-aware retrieval, where memory access can follow entity links, multi-hop connections, or temporal dependencies encoded in the graph [1, 16]. For unstructured memories, retrieval typically relies on embedding similarity search. Some systems employ hybrid pipelines that combine dense retrieval methods with lexical approaches like BM25, followed by a re-ranking step [16].

Consolidation. Consolidation methods differ in how they represent the relationships among memory items. Similarity-indexed methods infer connections between memories based on embedding similarity. Typed-relational methods explicitly maintain relationships such as entity-based, temporal, semantic, and causal links [10]. Hierarchical methods organize memories into nested structures at different levels of granularity through parent–child or group-membership relationships.

Forgetting. Forgetting mechanisms can be characterized by their trigger conditions and deletion policies. Common triggers include capacity overflow [14, 26] and the crossing of a retention threshold. Once triggered, the forgetting operator determines which memories

Table 2: Comparison of agentic memory systems. Abbreviations: STM. = Short-term Memory, Epis. = Episodic, Proc. = Procedural, Sem. = Semantic, Pers. = Personal, Ext. = Extract, Upd. = Update, Ret. = Retrieval, Cons. = Consolidation, Forg. = Forgetting, Comp. = Compression, Conf. = Conflict Detection, Refl. = Reflection, Str. = Structured, Semi. = Semi-structured, Unstr. = Unstructured.

	Categorization					Operators								Storage		
Systems	STM.	Epis.	Proc.	Sem.	Pers.	Ext.	Upd.	Ret.	Cons.	Forg.	Comp.	Conf.	Refl.	Str.	Semi.	Unstr.
Mem0 [2]	✓	✓	✗	✓	✗	✓	✓	✓	✗	✗	✗	✓	✗	✗	✗	✓
Mem0^g [2]	✗	✗	✗	✓	✗	✓	✓	✓	✗	✗	✗	✓	✗	✗	✓	✗
xMemory [9]	✓	✓	✗	✓	✗	✓	✓	✓	✓	✗	✓	✗	✗	✗	✓	✓
Memory-r1 [30]	✗	✓	✗	✓	✗	✓	✓	✓	✗	✗	✗	✓	✗	✗	✗	✓
MemLLM [13]	✗	✗	✗	✓	✗	✓	✓	✓	✗	✗	✗	✓	✗	✓	✗	✗
ChatDB [6]	✗	✓	✗	✓	✗	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✗
MemoryBank [33]	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	✗	✓	✗	✓	✓
MemGPT [14]	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	✗	✗	✗	✗	✓
MIRIX [26]	✗	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✗	✗	✗	✓	✓
ZEP [16]	✓	✓	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓

to remove based on their estimated utility, using signals such as recency, access frequency, importance, and validity. These signals may be considered individually or combined into a composite score. For example, temporal and usage-based signals may assign lower retention scores to older or infrequently accessed memories [33], while validity-based signals may identify and remove memories that have become outdated or been contradicted by newer information.

Compression. Methods for implementing the compression operator fall into two categories: single-item and multi-item compression. Single-item methods employ content-level techniques, such as summarization or the transformation of memory content into structured representations, to eliminate redundant details while preserving essential information. Multi-item methods, by contrast, consolidate multiple related memories into a single item.

Reflection. Depending on the type of knowledge they produce, reflection methods can be divided into two categories: declarative and procedural knowledge abstraction. The former synthesizes multiple memories into higher-level beliefs, patterns, relationships, or persistent states [9, 11], whereas the latter distills task-specific experiences into reusable strategies, workflows, rules, or skills.

2.3.2 Safety Management. It addresses the security challenges posed by agentic memory [23]. As agent memory becomes persistent and shareable across users and agents, it is crucial to explore multi-user and multi-agent memory scenarios [5, 17], where memories can be exchanged, reused, or collaboratively maintained across different entities. This evolution highlights the need for governance mechanisms to regulate memory access and safeguard sensitive content. *Access control* determines which memories can be shared and under what conditions [17]. *Privacy protection* prevents memory content from exposing or leaking user-sensitive information [19].

2.4 Memory Storage

Agentic memory can be implemented with multiple storage formats, which differ in their underlying memory representation, organizational structure, and support for memory operations.

Textual Memory. It adopts a human-readable format and supports structured, semi-structured, and unstructured memory.

(1) Structured Memory. It maintains memories in explicitly organized forms, e.g., relational databases [6, 13]. It is particularly suitable when memories can be normalized into explicit entities, attributes, and relations, such as user profiles and other well-defined memory records. By imposing explicit structure, it separates information clearly, avoiding the mixture of textual records.

(2) Semi-structured Memory. It uses graph-based structures [1, 2, 16], which can build indexes over entities, relations, and neighborhood structure, thereby enabling relation-aware retrieval, multi-hop reasoning, and memory organization based on entity connectivity.

(3) Unstructured Memory. Unstructured memory storage can be broadly viewed as file-based storage, where memories are preserved as flat files rather than being normalized into relational schemas or graph structures. This category covers both textual memories stored in free-form or weakly structured formats, such as Markdown and text chunks [2, 29], and multimodal memories stored as original files, such as images, audio, or video [26].

Model-Level Memory. Parametric memory stores knowledge implicitly in model parameters or latent states rather than in an external memory store [12, 31]. Its main advantage is tight integration with model inference, enabling low-latency access without external retrieval. However, it is harder to query, edit, or delete precisely.

2.5 Existing Agentic Memory Systems

We conduct a comprehensive survey of existing agentic memory systems and present a summary in Table 2.

2.6 Open Problems

We also highlight several open challenges in agentic memory, including human-like reflective memory, real-time memory optimization, collaborative memory across multiple agents, human-like dreaming, and benchmarking agentic memory systems.

3 BIOGRAPHY

Guoliang Li is a full professor in Department of Computer Science, Tsinghua University. His research interests include databases and data systems. He is an ACM Fellow and IEEE Fellow.

Jiaqi Tian is currently a PhD student in the Department of Computer Science, Tsinghua University. His research interest lies in agentic memory systems.

Xuanhe Zhou is an assistant professor in Department of Computer Science, Shanghai Jiao Tong University. His research interests lie in data systems. He received SIGMOD 2025 Jim Gray Dissertation Honorable Mention and VLDB 2023 Best Industry Paper Runner-up.

ACKNOWLEDGMENTS

Guoliang Li is the corresponding author. This paper was supported by NSF of China (62525202, 62232009), National Key R&D Program of China (2023YFB4503600), Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China(JYB2025XDXM509), Science and Technology Research and Development Plan of China Railway & National Engineering Research Center of System Technology for High-Speed Railway and Urban Rail Transit (L2024W001).

REFERENCES

[1] Petr Anokhin, Nikita Semenov, Artyom Y. Sorokin, Dmitry Evseev, Andrey Kravchenko, Mikhail Burtsev, and Evgeny Burnaev. 2025. AriGraph: Learning Knowledge Graph World Models with Episodic Memory for LLM Agents. In *IJCAI*. 12–20.

[2] Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory. In *ECAI*, Vol. 413. 2993–3000.

[3] Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, Huajun Chen, and Ningyu Zhang. 2025. LightMem: Lightweight and Efficient Memory-Augmented Generation. *CoRR* abs/2510.18866 (2025).

[4] Jinyuan Fang, Yanwen Peng, Xi Zhang, Yingxu Wang, Xinhao Yi, Guibin Zhang, Yi Xu, Bin Wu, Siwei Liu, Zihao Li, Zhaochun Ren, Nikos Aletras, Xi Wang, Han Zhou, and Zaiqiao Meng. 2025. A Comprehensive Survey of Self-Evolving AI Agents: A New Paradigm Bridging Foundation Models and Lifelong Agentic Systems. *CoRR* abs/2508.07407 (2025).

[5] Dongge Han, Camille Couturier, Daniel Madrigal Díaz, Xuchao Zhang, Victor Rühle, and Saravan Rajmohan. 2025. LEGOMem: Modular Procedural Memory for Multi-agent LLM Systems for Workflow Automation. *CoRR* abs/2510.04851 (2025).

[6] Chenxu Hu, Jie Fu, Chenzhuang Du, Simian Luo, Junbo Zhao, and Hang Zhao. 2023. ChatDB: Augmenting LLMs with Databases as Their Symbolic Memory. *CoRR* abs/2306.03901 (2023).

[7] Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. 2025. HiAgent: Hierarchical Working Memory Management for Solving Long-Horizon Agent Tasks with Large Language Model. In *ACL (1)*. 32779–32798.

[8] Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, Senjie Jin, Jiejun Tan, Yanbin Yin, Jiongnan Liu, Zeyu Zhang, Zhongxiang Sun, Yutao Zhu, Hao Sun, Boci Peng, Zhenrong Cheng, Xuanbo Fan, Jiaxin Guo, Xinlei Yu, Zhenhong Zhou, Zewen Hu, Jiahao Huo, Junhao Wang, Yuwei Niu, Yu Wang, Zhenfei Yin, Xiaobin Hu, Yue Liao, Qiankun Li, Kun Wang, Wangchunshu Zhou, Yixin Liu, Dawei Cheng, Qi Zhang, Tao Gui, Shirui Pan, Yan Zhang, Philip Torr, Zhicheng Dou, Ji-Rong Wen, Xuanjing Huang, Yu-Gang Jiang, and Shuicheng Yan. 2025. Memory in the Age of AI Agents. *CoRR* abs/2512.13564 (2025).

[9] Zhanghao Hu, Qinglin Zhu, Hanqi Yan, Yulan He, and Lin Gui. 2026. Beyond RAG for Agent Memory: Retrieval by Decoupling and Aggregation. *CoRR* abs/2602.02007 (2026).

[10] Dongming Jiang, Yi Li, Guanpeng Li, and Bingzhe Li. 2026. MAGMA: A Multi-Graph based Agentic Memory Architecture for AI Agents. In *ACL (1)*. 36848–36865.

[11] Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. 2025. Memory OS of AI Agent. In *EMNLP*. 25961–25970.

[12] Zhiyu Li, Shichao Song, Chenyang Xi, Hanyu Wang, Chen Tang, Simin Niu, Ding Chen, Jiawei Yang, Chunyu Li, Qingchen Yu, Jihao Zhao, Yezhaohui Wang, Peng Liu, Zehao Lin, Pengyuan Wang, Jiahao Huo, Tianyi Chen, Kai Chen, Kehang Li, Zhen Tao, Junpeng Ren, Huayi Lai, Hao Wu, Bo Tang, Zhenren Wang, Zhaoxin Fan, Ningyu Zhang, Linfeng Zhang, Junchi Yan, Mingchuan Yang, Tong Xu, Wei Xu, Huajun Chen, Haofeng Wang, Hongkang Yang, Wentao Zhang, Zhi-Qin John Xu, Siheng Chen, and Feiyu Xiong. 2025. MemOS: A Memory OS for AI System. *CoRR* abs/2507.03724 (2025).

[13] Ali Modarressi, Abdullatif Köksal, Ayyoob Imani, Mohsen Fayyaz, and Hinrich Schütze. 2024. MemLLM: Finetuning LLMs to Use An Explicit Read-Write Memory. *CoRR* abs/2404.11672 (2024).

[14] Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. 2023. MemGPT: Towards LLMs as Operating Systems. *CoRR* abs/2310.08560 (2023).

[15] Hongjin Qian, Zhao Cao, and Zheng Liu. 2026. MemoBrain: Executive Memory as an Agentic Brain for Reasoning. In *ACL (Findings)*. 2646–2662.

[16] Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. 2025. Zep: A Temporal Knowledge Graph Architecture for Agent Memory. *CoRR* abs/2501.13956 (2025).

[17] Alireza Rezazadeh, Zichao Li, Ange Lou, Yuying Zhao, Wei Wei, and Yujia Bao. 2025. Collaborative Memory: Multi-User Memory Sharing in LLM Agents with Dynamic Access Control. *CoRR* abs/2505.18279 (2025).

[18] Alireza Rezazadeh, Zichao Li, Wei Wei, and Yujia Bao. 2025. From Isolated Conversations to Hierarchical Schemas: Dynamic Tree Memory Representation for LLMs. In *ICLR*.

[19] Zitong Shi, Guancheng Wan, Wenke Huang, Guibin Zhang, Jiawei Shao, Mang Ye, and Carl Yang. 2025. Privacy-Enhancing Paradigms within Federated Multi-Agent Systems. *CoRR* abs/2503.08175 (2025).

[20] Theodore R. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. 2024. Cognitive Architectures for Language Agents. *Trans. Mach. Learn. Res.* 2024 (2024).

[21] Ji Sun, Guoliang Li, Peiyao Zhou, Yihui Ma, Jingzhe Xu, and Yuan Li. 2025. AgenticData: An Agentic Data Analytics System for Heterogeneous Data. *CoRR* abs/2508.05002 (2025).

[22] Zhaoyan Sun, Jiayi Wang, Xinyang Zhao, Jiachi Wang, and Guoliang Li. 2025. Data Agent: A Holistic Architecture for Orchestrating Data+AI Ecosystems. *IEEE Data Eng. Bull.* 49 (2025), 79–95.

[23] Bo Wang, Weiyi He, Shenglai Zeng, Zhen Xiang, Yue Xing, Jiliang Tang, and Pengfei He. 2025. Unveiling Privacy Risks in LLM Agent Memory. In *ACL (1)*. 25241–25260.

[24] Jiayi Wang and Guoliang Li. 2025. AOP: Automated and Interactive LLM Pipeline Orchestration for Answering Complex Queries. In *CIDR*.

[25] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. 2024. A survey on large language model based autonomous agents. *Frontiers Comput. Sci.* 18 (2024), 186345.

[26] Yu Wang and Xi Chen. 2025. MIRIX: Multi-Agent Memory System for LLM-Based Agents. *CoRR* abs/2507.07957 (2025).

[27] Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. 2025. Agent Workflow Memory. In *ICML*, Vol. 267.

[28] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, Qi Zhang, and Tao Gui. 2025. The rise and potential of large language model based agents: a survey. *Sci. China Inf. Sci.* 68 (2025).

[29] Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-MEM: Agentic Memory for LLM Agents. *CoRR* abs/2502.12110 (2025).

[30] Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schütze, Volker Tresp, and Yunpu Ma. 2026. Memory-R1: Enhancing Large Language Model Agents to Manage and Utilize Memories via Reinforcement Learning. In *ACL (1)*. 12805–12825.

[31] Hongkang Yang, Zehao Lin, Wenjin Wang, Hao Wu, Zhiyu Li, Bo Tang, Wenqiang Wei, Jinbo Wang, Zeyun Tang, Shichao Song, Chenyang Xi, Yu Yu, Kai Chen, Feiyu Xiong, Linpeng Tang, and Weinan E. 2024. Memory^3: Language Modeling with Explicit Memory. *CoRR* abs/2407.01178 (2024).

[32] Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2025. A Survey on the Memory Mechanism of Large Language Model-based Agents. *ACM Trans. Inf. Syst.* 43 (2025), 155:1–155:47.

[33] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. MemoryBank: Enhancing Large Language Models with Long-Term Memory. In *AAAI*. 19724–19731.

[34] Wei Zhou, Xuanhe Zhou, Shaokun Han, Hongming Xu, Guoliang Li, Zhiyu Li, Feiyu Xiong, and Fan Wu. 2026. Are We Ready For An Agent-Native Memory System? *arXiv preprint arXiv:2606.24775* (2026).

[35] Xuanhe Zhou, Guoliang Li, Zhaoyan Sun, Zhiyuan Liu, Weize Chen, Jianming Wu, Jiesi Liu, Ruohang Feng, and Guoyang Zeng. 2024. D-Bot: Database Diagnosis System using Large Language Models. *Proc. VLDB Endow.* 17 (2024), 2514–2527.