

DAO Project

数据优先本体论：世界模型在数据中，而非权重里

黎展波 / Atlas Lee*

2026 年 5 月

摘要

大语言模型的神经网络权重被默认为编码了世界模型。本文论证这一假设在需要精确性、可审计性和实时更新的场景中存在结构性局限，指出四种危机：幻觉、知识冻结、不可解释性和不可修改性。我们提出**数据优先本体论** (Data-First Ontology)，主张世界模型应以六种显式原语——实体 (Being)、定义 (Def)、类型 (Type)、关系 (Relation)、合约 (Contract) 和版本 (Version) ——存在，Def 与 Type 协作构成实体的完整行为。本文形式化了隐式与显式世界模型的区别，并介绍 **DaoQL** (DAO Project 核心引擎) 的教学版实现 DaoQL-Edu 作为阶段一验证：四种基础元语已实现跨引擎统一存储，且性能达到实用级别。完整六种元语的理论框架已阐述，其大规模验证作为后续工作。**关键词**：世界模型；本体论；大语言模型；数据结构；数据优先；显式知识表示

1 引言：世界模型的危机

1.1 默认假设

当代人工智能建立在一个极少被明确陈述的默认假设之上：**大语言模型的神经网络权重构成了世界模型**。当我们说 GPT-4 “知道” 巴黎是法国的首都，或者说它 “理解” 吸烟与肺癌之间的因果关系时，我们实际上在默许这一假设。按照这种看法，世界模型是一个函数 $f_\theta: \text{文本} \rightarrow \text{文本}$ ，其中参数 θ 编码了从数万亿 token 的预训练数据中提取的统计规律性。

这一假设支撑了该领域的核心研究纲领：让模型更大，用更多数据训练它，它的世界模型就会变得更准确、更全面、更鲁棒。扩展定律 (scaling laws) [1,2] 不仅被解释为损失下降的实证规律，更被当作世界模型随规模单调改善的证据。其隐含的推理是：如果一个 1750 亿参数的模型偶尔产生幻觉，那么一个万亿参数的模型就会很少幻觉；如果一个训练数据截止于 2024 年的模型缺乏 2025 年的知识，那么一个持续训练的模型就能保持与时俱进。

本文论证这一推理在**需要精确性、可审计性和实时更新的场景**中存在结构性局限。当前大语言模型面临的四种危机——幻觉、知识冻结、不可解释性和不可修改性——在纯权重编码的范式下难以根治，因为权重缺乏显式结构所具备的原子操作语义。

1.2 四种结构性危机

幻觉 (Hallucination)。大语言模型通过在其输入和训练数据所条件的概率分布中采样来生成文本。当训练数据包含矛盾信息，或者查询超出训练分布时，模型并不会 “知道” 自己不知道。它以高度自信输出看似合理但事实错误的陈述 [3]。其根本原因并非训练数据不足或优化

*DAO 取自中国哲学 “道” (Tao/Dao)，意为万物之本源与运行规律，与区块链中的 DAO (Decentralized Autonomous Organization) 无关。

不够完美，而是模型没有任何机制来验证其生成的陈述是否对应于架构中某个位置的显式事实。“巴黎是法国首都”这一“知识”并非存储在某个特定位置，而是作为统计关联分布于数十亿权重之中。当这一关联与另一关联——比如一条伪造的训练样本声称里昂是首都——发生竞争时，模型没有原则性的方法来解决冲突。

知识冻结 (Frozen Knowledge)。一旦训练完成，权重即被冻结。模型的世界模型是训练时刻世界状态的一张快照。检索增强生成 (Retrieval-Augmented Generation, RAG) [4] 和微调 (fine-tuning) [5] 等技术试图解决这一局限，但它们是被嫁接到冻结的权重架构之上，而非取而代之。RAG 将外部文档注入上下文窗口，但模型的根本世界模型——权重——并未改变。微调可以更新权重，但该过程计算成本高昂，影响整个模型的全局行为，且存在灾难性遗忘 (catastrophic forgetting) 的风险 [6]。权重中的世界模型在训练完成后难以增量更新。

不可解释性 (Uninterpretability)。当大语言模型给出一个答案时，我们无法检查其推理过程。注意力权重 (attention weights) 仅提供了一种粗略的事后可视化，显示哪些 token 影响了哪些其他 token [7]，但它们并不能揭示模型为何相信某个特定事实，也不能显示它考虑了哪些事实并将其排除。如果一个医疗大语言模型推荐某种治疗方案，我们无法追溯从训练数据到推荐结果的路径。这种不透明性并非当前可视化技术的临时局限，而是知识分布于数十亿参数之中的固有属性。权重中的世界模型缺乏可解释的推理轨迹。

不可修改性 (Uneditability)。假设我们发现一个大语言模型持续生成某种危险的错误陈述——例如向胃溃疡患者推荐阿司匹林。纠正这一错误需要微调，而微调影响整个权重矩阵，可能在别处引入新的错误 [8]。不存在“外科手术式”地移除或更新单一信念的机制。权重中的知识是全局耦合的：“阿司匹林”的表示与“止痛”、“血液稀释”、“胃部刺激”以及无数其他概念的表示纠缠在一起。修改一处，影响全局。权重中的世界模型难以进行精确的局部修改。

论点。这四种危机并非独立的工程问题，而是**纯权重编码范式**在缺乏显式结构操作语义时的系统性表现。在需要精确性、可审计性和实时更新的场景中，世界模型需要显式数据结构的支持。

1.3 思想实验

设想两座图书馆。

图书馆 A 雇了一位超级读者，他背下了馆藏中的每一本书。当你提出问题时，超级读者凭记忆作答。这位超级读者博学、善辩、反应迅速。但他偶尔会混淆相似的事实（幻觉），对培训日期之后出版的书籍一无所知（知识冻结），无法解释自己如何得出某个答案（不可解释），而且纠正一个错误可能扭曲其他记忆（不可修改）。

图书馆 B 将所有书籍都摆在书架上。当你提出问题时，图书管理员查阅目录，取回相关卷册，交叉核对资料来源，并给出附有引证的答案。书籍可以被更新（新版替换旧版），图书管理员的检索过程是可检查的（你可以看到查阅了哪些书），纠正一本书不会影响其他书。

在许多关键应用场景中，大语言模型单独作为知识仓库面临挑战。我们提出一种互补架构：显式数据结构作为世界模型，LLM 作为查询与推理引擎。

在图书馆 B 中，世界模型不在图书管理员的记忆里，而在书籍本身之中——它们的内容、关系、引证、修订历史。图书管理员（一个大语言模型）是一个复杂的查询引擎，而非知识的仓库。这种角色的倒转是本文的核心主张。

1.4 本文立场

我们提出**数据优先本体论** (Data-First Ontology): 一种将世界模型视为显式计算结构而非隐式统计分布的范式。具体而言, 我们主张世界模型应由六种原语构建而成:

- **实体 (Being)**: 具有唯一身份的状态集合, 是可追踪、可演化的存在单元;
- **定义 (Def)**: Being 的定义;
- **类型 (Type)**: Being 的分类标签;
- **关系 (Relation)**: Being 之间的关系;
- **版本 (Version)**: Being 的历史状态;
- **合约 (Contract)**: 决定 Being 的演化。

这六个原语共同回答”存在是什么、如何被定义、如何被分类、如何与他人关联、如何被追踪、如何演化”这六个本体论问题。它们的全形式化语义展开如下:

在此框架中, **类型 (Type)** 与 **定义 (Def)** 协作构成实体的完整行为。Def 回答”这是什么”——它规定实体的基础字段、约束和核心合约 (如”产品” Def 定义了名称、价格等字段), 类似于**类 (Class)**。Type 回答”这属于哪一类, 因此具有什么额外行为”——它是一个层级特化体系, 如”数码产品 → 手机 → 华为手机”, 类似于**接口 (Interface)**。Type 可以继承父类型的行为, 扩展新的合约, 或覆盖 Def 的默认行为。**字段定义在 Def 中, 行为由 Def 和 Type 共同组成**。一个实体同时具有 Def (类) 和 Type (接口), 二者协作而非正交。

这些原语不仅仅是一种数据模型, 它们本身就是世界模型。实体、定义、关系、合约和版本的结构就是世界的模型。大语言模型操作这一结构——查询它、遍历它、概括它——但模型本身独立于任何神经网络而持存。

本文的贡献如下:

1. **一个概念框架**, 区分隐式 (基于权重) 与显式 (基于结构) 世界模型, 并对其各自的局限性进行形式化刻画。
2. **六个本体论原语** (实体、定义、类型、关系、合约、版本), 它们共同构成显式世界模型, 具备操作语义和自举性质。Def 规定基础结构与核心行为, Type 提供层级特化与继承能力, 二者协作构成实体的完整行为。
3. **DaoQL-Edu** (DAO Project 核心引擎 DaoQL 的教学版), 一个仅包含四种核心元语的开源实现, 验证了阶段一 (工具层) 的主张: 跨引擎嵌套检索可以在单一进程内统一实现, 且在同机条件下达到或超越各专项系统的性能 (详见第 5 节)。完整版 DaoQL (六种元语, 62,000 行 Rust, 833 项测试) 已工程实现, Type 和 Contract 的验证作为后续工作。
4. **一个研究议程**, 指出必须解决方能释放该范式全部潜力的开放问题: 符号接地、完备性和演化一致性。

本文后续结构如下。第 2 节形式化隐式世界模型与显式世界模型的区别。第 3 节综述相关工作。第 4 节阐述数据优先本体论的核心论点, 包括六个原语及其哲学基础。第 5 节描述 DaoQL-Edu 教学版实现作为阶段一 (工具层) 的可行性证据。第 6 节将我们的立场置于科学哲学与心灵哲学的语境中。第 7 节讨论未来工作。第 8 节诚实声明本文的局限性与能力边界。第 9 节讨论开放问题, 第 10 节以对研究社区的邀请作结。

接洽”

() ()
* * * 能描述
0.333333
() ()
* * *

0.333333“订
行状单
为态状
(B依态
hay换机:
ior的待
动处
态理
→
已
付
款
→
已
发
货
→
已
送
达”

隐式世界模型将这些能力编码为神经网络权重中的统计模式。显式世界模型将它们编码为可操作的数据结构。

2.2 隱式世界模型

形式化地，一个隐式世界模型是一个函数：

$$W_{\text{implicit}} = f_{\theta} : \mathcal{X} \rightarrow \mathcal{Y}$$

其中 $\mathcal{X}$ 是输入查询的空间（通常为 token 序列）， $\mathcal{Y}$ 是输出响应的空间， $\theta \in \mathbb{R}^d$ 是神经网络的冻结参数（通常 $d \sim 10^{11}-10^{12}$ ）。

知识以分布式表示的形式存储:

- “巴黎是法国首都”这一“事实”并不存储在某个特定索引 θ_i 处。它编码为数千甚至数百万参数上的激活模式，与“柏林”、“东京”、“首都”和“法国”的表示纠缠在一起。
- “拥有地标”（HAS_LANDMARK）这一“关系”不是一等对象。它仅作为训练数据中“巴黎”和“埃菲尔铁塔”这两个 token 序列之间的统计相关性而存在。

- 关于胃溃疡和阿司匹林的”规则”不是一个逻辑蕴含。它是一个统计规律性，可能被更强的相关性覆盖——例如，如果”阿司匹林缓解疼痛”在训练数据中出现得更频繁。

后果：不存在”读取”特定事实、“修改”特定关系或”验证”特定规则的操作。唯一可用的操作是前向推理 ($f_{\theta}(x)$) 和微调 (更新 θ ，这是全局且近似的)。

2.3 显式世界模型

一个显式世界模型是一个结构化元组：

$$W_{\text{explicit}} = (\mathcal{B}, \mathcal{D}, \mathcal{T}, \mathcal{R}, \mathcal{C}, \mathcal{V})$$

其中：

- $\mathcal{B}$ 是**实体 (Being)** 的集合，每个实体具有唯一标识符 BeingId、Def (结构)、Type (特化行为，可选)、分为核心属性和扩展属性的状态，以及历史版本的链条；
- $\mathcal{D}$ 是**定义 (Def)** 的集合，每个定义规定实体之结构、约束、合约 (写入路径上的可执行钩子)、状态机和关系声明；
- $\mathcal{T}$ 是**类型 (Type)** 的集合，每个类型是 Def 的层级特化机制，提供继承能力，可扩展或覆盖 Def 的合约与约束；
- $\mathcal{R}$ 是**关系 (Relation)** 的集合，每个关系带类型、有向或无向，具有可选的时效窗口、基数约束和扩展属性；
- $\mathcal{C}$ 是**合约 (Contract)** 的集合，即写入操作前后被调用的可执行行为 (以脚本形式表达)；
- $\mathcal{V}$ 是一个版本化函数，将每个实体和时间戳映射到一个不可变快照。

与 EAVT 数据模型的根本区别。Datomic 等系统采用 EAVT (Entity-Attribute-Value-Time) 模型：“实体 E 在时刻 T 具有属性 A 的值 V”。这是一种**数据语义**——记录”发生了什么” (what happened)。一个 EAVT 事实是一条被动的命题记录。

DaoQL 的六原语是一种**本体论语义**——构造”是什么” (what is)。一个 Being 不是一条记录，而是一个**自治的存在单元**：它有身份 (BeingId)、有本质定义 (Def)、有特化分类 (Type)、有社会关系 (Relation)、有行为规则 (Contract)、有历史人格 (Version)。Datomic 问：“Alice 在 2024-01-01 的姓名是什么？” (查询历史事实)。DaoQL 问：“Alice 是什么？她如何行为？她与谁关联？她如何演化？” (存在论探索)。

这不是实现细节的不同，而是**根本的本体论承诺不同**：EAVT 承诺”世界由事实组成”，DaoQL 承诺”世界由存在组成”。

显式世界模型上的操作：

() () ()

* * *

操 语 示

0.3333333333333333例

() () ()

* * *

0.3333333333333333Berdye(城

d'将市,

De城

d 市

修_v2)

改

为

d',

附

带

迁

移

语

义

() () ()

* * *

0.3333333333333333contract(患

hook者,

script)

执新

行前,

行溃

为疡

附检

加查)

到

实

体

b

```

    () () ()
* * *
    操作语示
0.3333333333333333例
    () () ()
* * *
0.3333333333333333assign_type(Mate60,
t)将华
Type
t 手
分机)
配
给
实
体
b
    () () ()
* * *
0.3333333333333333type_contract(华
hood将,
script)
约机,
附创
加建
到后,
Typecheck_harmony)
t

```

后果：事实、关系、规则和行为都是一等对象。它们可以被独立地读取、修改、验证和审计。更新巴黎的人口不会影响柏林的人口。

2.4 对比分析

维度	隐式 (W_{implicit})	显式 (W_{explicit})
可读性	无法提取特定事实	<code>read(BeingId)</code> 返回精确状态
可修改性	微调影响整个模型	单一关系或事实更新
可验证性	无推理痕迹	完整的查询执行轨迹
可审计性	无法审计内部状态	版本链 = 不可变审计日志
可解释性	注意力权重不透明	类型、关系、合约人类可读
可演化性	需要重新训练	运行时模式演化 (<code>evolve</code>)
可组合性	权重难以局部分解	子模型可被提取和共享
反事实一致性	关联之间的统计竞争	修改结构后的确定性求值

维度	隐式 (W_{implicit})	显式 (W_{explicit})
----	------------------------------	------------------------------

最后一行——**反事实一致性** (counterfactual consistency) ——尤为关键，将成为我们计划中的实验评估的主题 (第 7 节)。当隐式世界模型被问到反事实问题 (“如果 X 不成立, Y 的后果会如何?”) 时, 答案取决于查询的具体措辞恰好激活了哪些统计关联。当显式世界模型被问到同一问题时, 答案可以通过修改结构并重新求值查询来确定性计算。

3 相关工作

在提出数据优先本体论之前, 我们需要明确本文与现有工作的关系。本节从三个维度梳理相关研究: 时序与不可变数据库、知识表示与知识图谱、以及大语言模型与结构化数据的结合。

3.1 时序与不可变数据库

Datomic [11] 是与本文理念最接近的先驱。由 Rich Hickey 于 2012 年提出, Datomic 的核心理念包括: (1) 时间作为数据模型的核心维度 (EAVT: Entity-Attribute-Value-Time); (2) 不可变追加——没有 UPDATE 或 DELETE, 只有新事实的追加; (3) Schema 即数据——Schema 以数据形式存储, 可被查询和演化; (4) 数据库作为值——数据库是不可变的值, 可以传递、比较和缓存。

Datomic 已经实现了 DaoQL 所主张的”时间作为本体”、“不可变追加”和”Schema 即数据”三个核心特性。然而, Datomic 采用单一的 EAVT 存储模型, 所有查询在 EAVT 索引上执行。DaoQL 在单一进程内统一了图、列、向量、索引和全文五种引擎, 通过 BeingId 实现零开销跨模态查询。此外, Datomic 的 Schema/Type 是一体的, 没有”同一实体类别可以有不同特化行为”的机制, 而 DaoQL 引入了 Def (Class) 和 Type (Interface) 的分离与协作。

Event Store [12] 和 **Temporal Tables** (SQL:2011) [13] 也在时间维度上做了重要工作。Event Store 采用事件溯源 (Event Sourcing) 模式, 将状态变化记录为不可变事件流。SQL:2011 的 Temporal Tables 在关系模型中内建了对有效时间 (valid time) 和事务时间 (transaction time) 的支持。这些系统记录”发生了什么”, 但不像 DaoQL 那样将时间作为存在的本体论维度——在 DaoQL 中, 时间不是属性, 而是 Being 的内在结构。

3.2 知识表示与知识图谱

RDF/OWL/SPARQL [14,15,16] 构成了语义网的技术栈。RDF 以三元组 (Subject-Predicate-Object) 表示知识, OWL 提供本体描述语言, SPARQL 提供查询机制。RDFS 的 **class** 和 **type** 机制允许简单的层级分类, 但缺乏 Def/Type 分离的灵活性。与 DaoQL 相比, RDF/OWL 缺乏内建的版本链、合约执行和多模态统一能力。知识图谱领域的综述工作 [30] 进一步揭示了现有知识表示系统在动态演化和多模态统一方面的结构性空白。

TypeDB [17] 是一个类型化超关系知识图谱数据库。TypeDB 支持类型继承、约束检查和规则推理, 其类型系统比 RDF/OWL 更严格。然而, TypeDB 的类型继承是静态的 Schema 定义, 不支持运行时的 Def/Type 协作演化, 也没有将合约嵌入存储引擎的写入路径。

Property Graph (如 Neo4j [18]、GQL [19]) 采用标签 (Label) 和属性的方式来组织图数据。Property Graph 的 Label 类似于 DaoQL 的 Type, 但 Label 没有行为语义——它不能定

义合约或约束。Property Graph 也缺乏不可变版本链和跨模态查询能力。图数据库模型的综述 [32] 指出，现有图数据库在类型系统和行为语义方面存在根本性限制。

3.3 大语言模型与结构化数据

GraphRAG [20] 由 Microsoft Research 提出，是 LLM 与知识图谱结合的代表性工作。GraphRAG 从非结构化文本中自动提取实体-关系三元组，构建知识图谱，然后在回答查询时检索相关的子图或社区结构来增强 LLM 的生成。**LightRAG** [21] 在此基础上优化了索引和检索效率，采用双层级检索系统。**FastGraphRAG** [22] 进一步使用个性化 PageRank 加速图遍历。

GraphRAG 及其变体的核心范式是“从文本构建图来辅助 LLM”——图是 LLM 的外部增强组件。DaoQL 的核心范式截然不同：图不是辅助 LLM 的工具，而是**世界模型的主存储**。LLM 只是查询和操作这个世界模型的推理引擎。

MemGPT [23] 由 UC Berkeley 提出，将操作系统的虚拟内存管理思想应用于 LLM。MemGPT 通过分层存储（核心内存、召回存储、归档存储）和显式函数调用，让 LLM 自主管理其记忆。MemGPT 解决了 LLM 上下文窗口有限的问题，但其记忆管理仍然以文本块为单位，缺乏结构化的实体、关系和版本语义。

MCP (Model Context Protocol) [24] 是一个开放协议，旨在标准化 LLM 与外部数据源、工具和服务的交互。MCP 是协议层，DaoQL 是存储层——二者互补而非竞争。一个基于 MCP 的 LLM 应用可以使用 DaoQL 作为其结构化记忆后端。

3.4 神经-符号结合

Neuro-symbolic AI [25] 致力于结合神经网络的感知能力与符号系统的推理能力。早期工作如 Neural Theorem Provers [26] 和 Logic Tensor Networks [27] 尝试将逻辑规则嵌入神经网络。近期工作如 AlphaProof [28] 展示了神经-符号结合在数学推理中的潜力。DaoQL 可以被视为神经-符号架构中的“符号端”基础设施——为 LLM（神经端）提供显式的、可验证的结构化知识库。

3.5 与现有工作的区别总结

	() () () () () ()
*	* * * * *
	维
0.1053	0.1053度
	() () () () () ()
*	* * * * *
0.1053	0.1053度
时	无无无Ver-
间	sion
作	链
为	
本	
体	

() () () () () ()
 * * * * *
 维 度
 0.10.13.70.10.07.10.10.228
 0.10.13.70.10.07.10.10.228
 不 WAL
 可 +
 变 版
 追 本
 加
 () () () () () ()
 * * * * *
 0.10.13.70.10.07.10.10.228
 0.10.13.70.10.07.10.10.228
 即 自
 数 举
 据
 () () () () () ()
 * * * * *
 0.10.13.70.10.07.10.10.228
 0.10.13.70.10.07.10.10.228
 协 部 Class/Interface
 作 分
 () () () () () ()
 * * * * *
 0.10.13.70.10.07.10.10.228
 0.10.13.70.10.07.10.10.228
 多 五
 模 引
 态 擎
 统
 一
 () () () () () ()
 * * * * *
 0.10.13.70.10.07.10.10.228
 0.10.13.70.10.07.10.10.228
 合 部 Rhai
 约 分 运
 嵌 行
 入 时
 写
 入
 路
 径

上表揭示了一个关键空白：现有系统要么专注于某一种数据模型（关系型、文档型、图型、向量型），要么专注于为 LLM 提供外部知识增强（GraphRAG、MemGPT），但没有系统同时具备所有数据模型能力并以”显式世界模型”为核心架构目标。

DaoQL 的定位：DaoQL 不是关系型数据库、不是文档数据库、不是图数据库、不是键值数据库、不是列式数据库、不是向量数据库、不是时序数据库、不是全文搜索引擎——但它**同时具备**所有这些数据库的能力：

() ()
 * *
 数DaoQL
 据的
 库实
 类现
 0.3929型方
 0.6071式

 () ()
 * *
 0.392971Def/Type
 关定
 系义
 型结
 数构
 据+
 库Con-
 tract
 约
 束
 (类
 似
 外
 键、
 触
 发
 器、
 检
 查
 约
 束)

() ()
 * *
 数DaoQL
 据的
 库实
 类现
 0.3929型方
 0.6071式

 () ()
 * *
 0.39290.6071BeingExt
 文键
 档值
 数扩
 据展,
 库支
 持
 任
 意
 嵌
 套
 属
 性
 () ()
 * *
 0.39290.6071BeingCore
 图+
 数Re-
 据la-
 库tion
 边
 链,
 指
 针
 直
 接
 邻
 接,
 O(1)
 图
 遍
 历

—
() ()
* *
数DaoQL
据的
库实
类现
0.3929型方
0.6071式
—
() ()
* *
0.392971BeingId
键→
值Be-
数ing
据的
库O(1)
查
找
() ()
* *
0.392971BeingExt
列按
式字
数段
据列
库式
存
储
+
LZ4
压
缩
+
SIMD
聚
合

—
() ()
* *
数 DaoQL
据的
库实
类现
0.3929型方
0.6071式
—
() ()
* *
0.39290.6071HNSW
向+
量标
数量
据量
库化,
BeingId
即
向
量
节
点
() ()
* *
0.39290.6071Version
时链
序+
数时
据间
库戳
索引,
O(1)
历
史
回
溯

() ()

* * 数 DaoQL

据的

库实

类现

0.3929型方

0.6071式

() ()

* *

0.3929071Tantivy

全倒

文排

搜索

索引

引+

擎jieba

分

词

关键洞察：传统数据库栈需要 8 个独立系统来满足世界模型的七种能力。DaoQL 在一个系统中统一了所有能力——不是通过“支持多种查询语言”的兼容层，而是通过**统一的原语集** (Being、Def、Type、Relation、Contract、Version)。这些原语不是 8 种数据模型的拼凑，而是一种**更底层的存在论结构**，它自然衍生出所有上层数据模型的能力。

DaoQL 填补了这一空白：它不是从文本构建图来辅助 LLM，而是将显式世界模型作为核心存储，让 LLM 成为这个模型的操作员而非拥有者。

4 核心论点：数据优先本体论

4.1 三个命题

我们提出三个相互嵌套的命题，它们共同定义了数据优先本体论范式。需要预先说明的是，命题中的“定义” (Def) 是结构规范原语，规定实体的字段、约束和核心行为；“类型” (Type) 是 Def 的层级特化机制，提供继承和多态能力，可扩展或覆盖 Def 的合约与约束，与 Def 协作构成实体的完整行为。

命题一（本体论）。计算世界模型中的基本存在单元不是符号（如一阶逻辑中所示），不是对象（如面向对象编程中所示），也不是进程（如 π 演算中所示）。它是**实体（Being）**：一种具有身份、状态、历史、行为以及与其他实体之关系的存在单元。

实体之“所是” (what it is) 不由外部模式规定，而由其**定义（Def）**规定。Def 不是实体的属性，而是实体的存在论前提：一个实体在成为可被操作的数据之前，必须先被定义为某种存在。Def 回答“这是什么”，而非“这属于哪一类”。

实体不是表中的一行（被动数据），也不是类的实例（行为在外部定义）。它是一种自治的存在单元，其行为内嵌于其定义（Def）之中，其历史是一条不可变的版本链条。实体是原语；其他一切都由此衍生。

命题二（认识论）。知识不是心智内容（连接主义），也不是命题集合（符号主义）。知识是**实体/定义/关系的结构**——而这一结构本身就是世界模型。

Type 提供了层级特化框架（如“数码产品 → 手机 → 华为手机”），可扩展和覆盖 Def 的合约与约束。知识的根基在 Def 的结构规范中：Def 定义了哪些字段存在、哪些约束生效、哪些核心合约触发。Type 在此基础上添加特化行为和约束，与 Def 共同构成完整的知识表示。

这一区分微妙但至关重要。在符号系统中，知识是关于世界的一组命题，独立于推理机制存储。在连接主义系统中，知识是推理机制本身的权重配置。在数据优先本体论中，知识是介于数据与推理之间的结构：Def 定义了可能存在什么，关系定义了实体如何连接，合约定义了结构如何演化。Type 则提供了分类框架，帮助组织和检索实体。结构不是关于世界的；在计算意义上，它就是世界。

命题三（方法论）。推理不是从权重中提取统计模式，而是对显式结构的**遍历与变换**。大语言模型是这一过程的加速器，而非过程内容的仓库。

当用户问道：“胃溃疡患者应该避免哪些药物？”，数据优先系统中的推理过程是：

1. 将查询解析为结构化遍历：实体（患者）→ 关系（诊断患有）→ Def（病症）→ 筛选（name="胃溃疡"）→ 关系（禁忌使用）→ Def（药物）。

在此遍历中，“患者”、“病症”、“药物”都是 Def（定义了相应实体的结构），而每个实体可能同时具有 Type（如“患者”的 Type 可能是“人类/病人”）。遍历操作基于 Def 规定的结构进行，Type 提供分类信息辅助检索。

2. 对显式结构执行遍历。
3. 收集结果（禁忌药物清单）。
4. 使用大语言模型从结构化结果生成自然语言响应。

大语言模型的角色是第 4 步，而非第 1-3 步。世界模型——关于哪些药物对胃溃疡禁忌的知识——栖居于显式结构之中，而非大语言模型的权重之内。

两阶段验证的分层。上述三个命题涉及的六种原语在验证策略上分为两个层次：

- **工具层**（阶段一，已验证）：Being、Def、Relation、Version 的交集足以解决“世界模型的多模态数据如何在单一系统内统一存储和查询”这一工程问题。第 5 节报告了 DaoQL-Edu 教学版对该层次的验证。
- **元工具层**（阶段二，理论定义）：Type 和 Contract 解决的是“世界模型如何描述和约束自身”这一本体论问题。这两个原语的理论框架已在下文完整阐述，其大规模工程验证作为后续工作。

这种分层是刻意的——即使仅有四种基础元语，也已经足够支撑一个比传统数据库碎片栈更强大的世界模型；Type 和 Contract 是增量增强，而非必要前提。

4.2 六个本体论原语

我们现在详细定义六个原语。

实体 (Being)。一个实体 b 是一个四元组：

$$b = (id, core, ext, history)$$

- $id \in \text{BeingId}$ 是一个唯一的、不可变的标识符（通常为 UUID）。
- $core$ 是一个固定大小的结构，包含实体的基本属性（如名称、定义标识符、创建时间戳）。
- ext 是一个可扩展的键值映射，包含该实体之定义所规定的任意属性。
- $history = [v_1, v_2, \dots, v_n]$ 是一条不可变的、仅追加的版本链，其中每个 $v_i = (timestamp_i, state_i, prev_hash)$ 。

实体不是从类”实例化”出来的；它是被**写入**存在的。创建与持久化之间没有区别。创建一个实体就是将其写入存储系统；读取一个实体就是通过其标识符检索它。

每个实体都有一个 **def** 字段，指向定义该实体之结构的 Def 实体。例如，一个”患者”实体的 **def** 指向定义”患者”之结构的 Def 实体。

定义 (Def)。一个定义 d 是一个六元组：

$$d = (name, fields, constraints, contracts, state_machine, relations)$$

- $name$ 是唯一的定义标识符（如”患者”、“药物”、“订单”）。
- $fields$ 是一组带类型的字段定义。
- $constraints$ 是一组不变谓词（如 `age >= 0`, `email MATCHES regex`）。
- $contracts$ 是一组可执行钩子（创建前、更新后、关联前等），每个钩子关联一段脚本（如 Rhai 脚本）。
- $state_machine$ 是一个有限状态机，定义该 Def 所辖实体的合法状态转换。
- $relations$ 是一组关系类型声明，该 Def 所辖实体可以参与这些关系。

Def 回答”这是什么”——它规定实体的完整结构。”患者”Def 包含姓名字段、年龄字段、诊断字段，以及相应的约束和合约。字段定义在 Def 中，不在 Type 中。

至关重要的是，**Def 本身也是实体**。Def 患者是一个实体，其 `core.def` 指向 `DAO_DEF`（即”定义之定义”）。不存在独立的”模式目录”或”系统表”。定义系统是**自举的**（bootstrapping）：Def 是什么的定义，本身作为一个 Def 存储。

类型 (Type)。一个类型 t 是 Def 的层级特化机制，类似于面向对象中的**接口 (Interface)**，但支持层级继承。它属于分类体系（taxonomy）中的节点，与 Def（类）协作构成实体的完整行为：

$$t = (name, parent, path, contracts, constraints, attributes)$$

- $name$ 是类型名称（如”华为手机”）。

- *parent* 是父类型（如“手机”），形成继承链。
- *path* 是从根到该节点的完整路径（如“数码产品/手机/华为手机”）。
- *contracts* 是该类型特有的可执行钩子，可扩展或覆盖 Def 的合约。
- *constraints* 是该类型特有的约束，可收紧 Def 的约束条件。
- *attributes* 是该类型特有的展示属性（如图标、颜色）。

字段定义在 Def 中，行为由 Def 和 Type 共同组成。Type 不定义字段，但它定义了额外的行为和约束。例如：

- Def “Product” = {fields: [name, price, brand], contracts: [check_price_positive]}
- Type “手机” (parent = “数码产品”) = {contracts: [check_has_imei]}
- Type “华为手机” (parent = “手机”) = {contracts: [check_harmony_os_support], constraints: [brand == “华为"]}

Being “Mate 60” (def = “Product”, type = “数码产品/手机/华为手机”) → 完整行为 = Def(Product) + Type(手机) + Type(华为手机) → 执行 check_price_positive + check_has_imei + check_harmony_os_support

Type 提供**多态特化**能力：同一 Def（类）的不同 Type（接口实现）可以有不同的行为。例如，Def “患者” 定义了基础字段，Type “急诊患者” 添加了紧急处理合约，Type “住院患者” 添加了床位管理合约。这类似于同一个类实现了不同的接口，每个接口规定了额外的行为约束。

Type 本身是可选的——一个 Being 可以没有 Type（仅使用 Def 的基础行为），但不能没有 Def。

关系 (Relation)。一个关系 r 是一个五元组：

$$r = (type, source, target, direction, attributes, validity)$$

- *type* 是关系类型（如 是首都、服用、禁忌使用）。
- *source* 和 *target* 是实体标识符。
- *direction* $\in$ {有向, 无向}。
- *attributes* 是一个可扩展的键值映射。
- *validity* = (t_{start}, t_{end}) 是可选的时效窗口。

关系是一等对象。它们可以被查询、版本化和约束。关系不是外键（被动的指针），而是一种活跃的连接，在创建、修改或遍历时可能触发合约。

注意：关系类型本身也是由 Def 定义的 (def = “DAO_RELATION”), 但关系实例是实体之间的实际连接。

合约 (Contract)。一个合约 c 是一个三元组：

$$c = (hook, script, ast_cache)$$

- *hook* $\in$ {创建前, 创建后, 更新前, 更新后, 关联前, 关联后, 删除前}。
- *script* 是合约的源代码（如一个 Rhai 函数）。
- *ast_cache* 是编译后的抽象语法树，缓存以提高性能。

合约嵌入写入路径。它们无法被客户端应用绕过，因为它们在存储引擎内部执行。例如，一个检查药物相互作用的合约在每次 关联（患者，药物，服用）操作时执行，无论哪个客户端发起了该操作。

合约在**本体论层面**是独立原语——它代表世界模型中”可执行行为规则”这一存在论维度，与 Being、Def、Type 等并列构成六种元语。但在**实现层面**，合约嵌入 Def 内部，作为 Def 的一个字段存储和执行。这种”本体独立、实现嵌入”的设计是有意的：它保证了合约无法被绕过（因为写入路径的唯一入口是 Def），同时保持了本体论的完整性。

版本 (Version)。一个版本 *v* 是一个四元组：

$$v = (timestamp, state, prev_pointer, next_pointer)$$

- *timestamp* 是逻辑或物理时间戳。
- *state* 是该版本时刻实体的完整状态。
- *prev_pointer* 是前一版本的存储偏移指针 (NodeOffset)。
- *next_pointer* 是后一版本的存储偏移指针 (NodeOffset)。

版本链使得每个实体的历史都可检查和可审计。它不是单独存储的可选审计日志，而是实体存在的内在时间维度。代码实现中以双向指针 (*prev_version* / *next_version*) 链接版本节点，功能等价于形式化定义中的版本链条。

4.3 自举：自我描述的系统

数据优先本体论最激进的性质是**系统描述自身**。不存在元层 (meta-level) 在数据层之外。考虑现有系统如何处理模式 (schema)：

```

_____
      () () ()
      * * *
      系模模
0.2308统式式
      存修
      0.3846改
_____
      () () ()
      * * *
0.2308统式式
      系TABLE
      统DDL
      表
  
```

() () ()

* * *

系模模

0.2308统式式

存修

0.3846储改

() () ()

* * *

0.230846goDB

无不

(无适

模用

式)

() () ()

* * *

0.230846CREATE

可 CONSTRAINT

选 Cypher

约

束

() () ()

* * *

0.230846mitlog

断事

言务

性

模

式

更

新

() () ()

* * *

0.230846DEF

定对

义 Def

的实

实体

体的

update

操

作

在 DaoQL 中, 命令 `define type 患者 { ... }` 不是由独立的 DDL 解析器解析并存储在独立目录中的。它是一个**写操作**, 创建一个 Def 为 `DAO_DEF` 的实体 (即"患者" 作为 Def 实体被写入)。定义就是数据; 数据带有定义; 系统是自我描述的。

具体而言, 自举层级如下:

Layer 0: 元层

```
Def "DAO_DEF"
  → 定义"定义是什么" (即 Def 本身的结构)
Def "DAO_TYPE"
  → 定义"类型是什么" (Type 层级体系的根定义)
Def "DAO_RELATION"
  → 定义"关系是什么"
```

Layer 1: 对象层

```
Def "患者"
  → 定义业务层面的"患者" (含姓名、年龄、诊断等字段)
Def "药物"
  → 定义业务层面的"药物" (含名称、成分、剂量等字段)
```

Layer 2: 实例层

```
Being "Alice" (def = "患者")
  → Alice 是一个患者实体
Being "阿司匹林" (def = "药物")
  → 阿司匹林是一个药物实体
```

Type (特化体系) 与 Def 协作构成完整行为:

Def 层级 (结构定义)

```
Def "Product"
  fields: [name, price]
  contracts: [check_price]
```

Type 层级 (特化继承)

数码产品

手机

华为手机

```
contracts: [check_harmony]
constraints: [brand == "华为"]
苹果手机
contracts: [check_ios]
```

Def "患者"

```
fields: [姓名, 年龄, 诊断]
contracts: [check_id]
```

生物

人类

患者

```
contracts: [check_urgency]
医生
contracts: [check_license]
```

一个 Being 同时具有 Def(基础结构)和 Type(特化行为): Being “Mate 60” (def = “Product”, type = “数码产品/手机/华为手机”) → 行为 = Def(Product) + Type(手机) + Type(华为手机) → 执行 check_price + check_harmony

Being “Alice” (def = “患者”, type = “生物/人类/患者”) → 行为 = Def(患者) + Type(人类) + Type(患者) → 执行 check_id + check_urgency

Def 和 Type 协作: 修改”患者” Def (如添加新字段) 影响所有患者 Being (类似修改类定义影响所有实例); 修改 Type”患者” (如添加新合约) 仅影响 type = “.../患者”的 Being (类似修改接口影响所有实现该接口的类)。

这一自举性质具有深远的后果:

- **无模式漂移**: 因为模式 (Def) 就是数据, 模式变更与任何其他数据变更一样被版本化。
- **无 DDL 解析器**: 不存在用于定义结构的独立语言。用于查询数据的同一种查询语言也用于查询和修改 Def。
- **反射能力**: 系统可以查询自身的 Def, 发现自身的类型, 并对自身结构进行推理。

4.4 时间作为本体, 而非属性

在常规系统中, 时间是一个属性: `updated_at` `TIMESTAMP`。在数据优先本体论中, 时间是一个本体论维度。

实体不是在某个时间点存在的对象。实体**就是**一个时间延展的存在。版本链 $[v_1, v_2, \dots, v_n]$ 不是附加于实体的审计轨迹, 而是实体内在的时间延展。Def 本身也具有版本链: 当 患者类型的定义被修改时 (如添加新字段), 这一变更作为 患者 Def 实体的一个新版本被记录。

这种倒转具有操作上的后果:

() () ()
 * * *
 操常数
 0.1154作规据
 (时优
 间先
 作(时
 为间
 属作
 0.4038性为
 本
 0.4808体)

() () ()
 * * *
 0.11540.4038SELECT
 前* 实
 状FROM
 态use(x).
 是WHERE
 什id前
 么? ().
 ” X 执
 行
 ()
 () () ()
 * * *
 0.11540.4038SELECT
 期* 实
 D FROM
 时audit_log
 的WHERE
 状..(D).
 态 执
 是 行
 什 ()
 么?
 ”

() ()
 * * *
 操常数
 0.1154作规据
 (时优
 间先
 作(时
 为间
 属作
 0.4038性为
 本
 0.4808体)

() ()
 * * *
 0.1154038
 和复实
 D2杂体
 之的(X).
 间差在
 发异(D1,
 生查D2)
 了询问.
 什 差
 么 异
 变 ().
 化? 执
 ” 行
 ()

() () ()
 * * *
 操常数
 0.1154作规据
 (时优
 间先
 作(时
 为间
 属作
 0.4038性为
 本
 0.4808体)

 () () ()
 * * *
 0.1154038
 实审实
 体计体
 是日 (X).
 否志历
 曾的史
 处全 ().
 于表任
 状扫意
 态描 (|v|
 S? v.state
 ” ==
 S).
 执
 行
 ()

查询 实体 (X). 历史 () 并非检索外部审计表，而是遍历实体的内在版本链。时间不是元数据，而是结构。

4.5 行为嵌入存储层

在常规系统中，行为存在于应用层。数据库存储数据；应用（Java、Python、Rust）编码业务逻辑。数据库可能提供触发器（PostgreSQL）或存储过程，但这些都是嫁接在存储系统上的二等机制。

在数据优先本体论中，行为内嵌于类型定义，并在存储引擎的写入路径上执行。

以药物相互作用检查为例：

PostgreSQL 方案：

```
CREATE TRIGGER 检查相互作用
BEFORE INSERT ON 处方
FOR EACH ROW
EXECUTE FUNCTION 检查药物相互作用 ();
```

- 触发器以一种独立的语言（PL/pgSQL）定义。
- 触发器存储在与数据分离的系统目录中。
- 触发器可以被超级用户禁用。
- 修改触发器需要 `ALTER TRIGGER DDL`。

数据优先方案：

```
Def("处方").contracts = {
  关联前: "fn 关联前(ctx) {
    let 相互作用 = 查询相互作用(ctx.source, ctx.target);
    if 相互作用.严重程度 == '禁忌' {
      return Err('检测到禁忌相互作用');
    }
    Ok()
  }"
}
```

- 合约作为 Def 的属性存储（与 `fields` 或 `constraints` 一样）。
- 合约在存储引擎的运行时（Rhai）中执行，而非客户端应用中。
- 合约无法被绕过，因为写入路径是通往持久化的唯一路径。
- 修改合约是对 Def 的一次 `update` 操作，与任何其他更新一样受到版本化和审计。所有该 Def 所辖实体自动继承新合约。

行为不是外部附加组件。它是世界模型本身的结构属性。

5 证据：DaoQL 的实现

5.1 两阶段验证策略

数据优先本体论的验证分为两个阶段，对应两个不同层面的问题：

阶段一：工具层（已验证）。四种核心元语——Being（实体）、Def（定义）、Relation（关系）、Version（版本）——的交集足以验证一个关键主张：世界模型的多模态数据（图、列存、向量、文档）可以在单一进程内统一存储和查询，而不必拆分为多个独立系统。本节报告的就是这一阶段的工程证据。

阶段二：元工具层（理论定义，工程实现中）。Type（类型层级特化）和 Contract（合约约束）解决的是世界模型的自举问题——模型如何描述自身、约束自身、演化自身。这两个原语的理论定义已在第 4 节完整阐述，其工程验证作为后续工作（见第 7 节）。

这种分层验证的策略是刻意的：即使仅有最基础的四种元语，也已经足够支撑一个比传统数据库碎片栈更强大的世界模型存储系统；Type 和 Contract 是在此之上的增量增强，而非必要前提。

5.2 DaoQL-Edu：教学版架构

为验证阶段一的主张，我们开源了 **DaoQL-Edu** (github.com/zhanbole/DaoQL-Edu) ——一个仅包含四种核心元语的教学实现。该版本足以验证跨引擎嵌套检索的核心能力，同时保持代码精简、易于理解和复现。

DaoQL-Edu 架构（四种元语）

查询层

DSL 解析器 (daoql-dsl)

Fluent API (daoql-core)

执行层

写入协调器 (WriteCoordinator)

查询执行器 (CrossEngineQuery)

存储层（四引擎统一）

图引擎 (Graph Engine)

BeingCore + Relation 边链（指针直接邻接）

列引擎 (Column Engine)

BeingExt 列式投影 + SIMD 聚合

向量引擎 (Vector Engine)

HNSW + 标量量化 (BeingId 即节点)

索引引擎 (Index Engine)

redb B+Tree + RoaringBitmap

物理层

WAL + 追加存储 + 版本链

统一键 (BeingId)。所有引擎共享同一个查找键：BeingId。向量检索返回 BeingId → 直接图遍历 → 直接列读取。零 ID 映射、零网络往返、零序列化。这种跨引擎零拷贝架构是 DaoQL-Edu 区别于传统“多数据库拼接”方案的根本特征。

版本链。每个 Being 的每次写入都生成一个不可变版本，包含时间戳、状态快照和双向指针 (prev_version / next_version)。版本链与数据存储在同一个 WAL 中，不需要独立的审计表

或 CDC 管道。时间不是属性，而是 Being 的内在结构——这一设计在第 4.4 节已有理论阐述，此处通过工程实现验证其可行性。

教学版未包含的内容：Type 层级特化系统、Contract 合约执行引擎、Workflow 工作流编排、LLM Pipeline (NL→DSL)、全文搜索引擎、生产级分布式分片。这些组件在完整版 DaoQL 中实现，其理论定义见第 4 节。

5.3 性能测量

以下测量结果来自 DaoQL-Edu 教学版，在与竞品**同机**（相同硬件、相同数据集）条件下进行。**这些数字不应被解读为通用 benchmark 结论**——我们的目的只是证明：在单一进程内统一实现多模态存储，其性能足以支撑世界模型的实时查询，而不必为了追求专项最优而拆分为多个独立系统。完整测量报告见 docs/reports/BENCHMARK_VS_COMPETITOR_COMPARISON.md。

教学版与完整版的性能区分。DaoQL-Edu 作为教学实现，有意采用标准教科书算法（如 HNSW 使用 HashMap + HashSet 而非工程优化后的 flat arrays），以保持代码清晰和可教学性。完整版 DaoQL（62,000 行 Rust）在相同算法基础上通过 SIMD、缓存预取、量化等工程优化进一步拉开差距。下表报告的是教学版数据——即使在此保守条件下，仍达到或超越各专项系统。

测量条件：单节点，Apple M-series (aarch64), 64 GB RAM, NVMe SSD。数据集为 XY-ERP 合成数据集（约 190 万实体）。

```

() () () ()
* * * *
能 DaoQL-
力 Edus
维(教基
0.2220.222准
0.4222版)
() () () ()
* * * *
0.2220.2223.315×
点 μsLu快
查 2.5于
μs Re-
dis
Lua
内
循
环;
876×
快
于
Re-
dis
TCP
```

() () ()
 * * * *
 能 Day 0 能
 力 100% 明
 维 (教基
 0.2220.2222 准
 0.4222 版)
 () () ()
 * * * *
 0.2220.2222 3.45×
 图 m5-快
 遍 8 (1,365
 历 ms 节
 (BFS 点);
 Depth 特
 针
 直
 接
 邻
 接,
 O(1)
 指
 针
 追
 逐

() () () ()
 * * * *
 能 DaoQd-
 力 Fdls 明
 维(教基
 0.2220.2222 准
 0.4222 版)
 () () () ()
 * * * *
 0.2220.222299812nt
 向 us3634;
 量 us 教
 搜 学
 索 版
 (HNSW)
 用
 标
 准
 HashMap
 +
 Hash-
 Set
 实
 现,
 完
 整
 版
 通
 过
 flat
 ar-
 rays
 +
 SIMD
 +
 量
 化
 预
 计
 可
 快
 5-
 10×。
 注:
 Qdrant
 测
 量

() () ()
 * * * *
 能 Dao 境
 力 8B3 明
 维 (教基
 0.2222 度 2 准
 0.4222 版)
 () () ()
 * * * *
 0.2222 2.223437 19House
 列 us~1 快;
 存 msSIMD
 聚 I64x4
 合 批
 (SUM) 量
 聚
 合
 () () ()
 * * * *
 0.2222 2.221387 66greSQL
 批 us 9 快,
 量 (680K/s)
 写 (1 批/s)
 入 量
 追
 加
 () () ()
 * * * *
 0.2222 2.2212337
 跨 us 无向
 引 直量
 擎 接→
 混 可图
 合 比→
 查 列,
 询 单
 进
 程
 内
 完
 成

诚实标注的局限：- 教学版 HNSW 采用标准算法实现（HashMap 节点存储、HashSet 访问标记、标量距离计算），未使用完整版的 flat arrays、SIMD 点积和 BQ 量化优化 - 并发写入存在锁竞争，吞吐量随连接数增长而下降（详见报告）- 中文全文搜索慢于 PostgreSQL tsvector (1.27 ms vs 32 μ s) - SIMD 宽度为 f64x4，未达到最新 CPU 的 f64x8/f64x16 潜力 - 上述对比均为单节点测量；分布式扩展已在 `shard_router.rs` 中实现 V1 架构（基于一致性哈希），但尚未完成多节点 benchmark

核心结论。传统技术栈需要 Neo4j（图）+ ClickHouse（列存）+ Qdrant（向量）+ Redis（缓存）+ PostgreSQL（关系）五个独立系统来满足世界模型的数据需求，伴随网络往返、序列化和运维复杂度。DaoQL-Edu 教学版在单一进程内统一了这些能力，且在同机条件下达到或超越了各专项系统的性能——这一结果支持了阶段一的核心主张；完整版 DaoQL 在此基础上存在明确的性能优化纵深。

5.4 完整版架构展望

DaoQL 完整版在 DaoQL-Edu 的四元语基础之上，增加了 Type、Contract 和 Workflow 三个组件：

DaoQL 完整版（六种元语 + 扩展）

查询层扩展

LLM Pipeline (`daoql-meta`) : NL $\rightarrow$ DSL 编译

执行层扩展

Type 动态分派引擎

Contract 执行器 (Rhai 运行时)

Workflow 状态机引擎

存储层扩展

全文引擎 (Tantivy + jieba)

分布式路由 (`shard_router.rs`, V1 已实现)

完整版的工程实现已完成 (62,000 行 Rust, 833 项测试)，但 Type 和 Contract 的性能验证、分布式多节点 benchmark 作为后续工作（见第 7 节）。

5.5 安全性与合规性

DaoQL 包含完整的安全框架：

- **认证：**Ed25519 / SM2 双算法，nonce 验证
- **授权：**CBAC（基于内容的访问控制）+ ACL
- **审计：**审计日志与 WAL 同批次持久化
- **加密：**SM4-CTR（可选）+ SM3-HMAC

- **合规**：等保三级路径已规划

这些安全特性嵌入写入路径的原语。CBAC 规则以 Def 属性形式存储，在每次读取或写入时自动执行。

6 哲学：计算本体论实在论

6.1 与现有哲学立场的对话

与波普尔 (Karl Popper) 的”三个世界”对话。波普尔区分了世界 1 (物理)、世界 2 (心理) 和世界 3 (客观知识，如书籍、理论、数学定理)。波普尔的世界 3 是被动的——一本书不会自己更新，一个理论不会自己演化。我们的显式世界模型是**主动的世界 3**：实体具有行为 (Contract)，类型具有演化能力 (evolve)，系统可以自我描述 (自举)。数据优先本体论将波普尔的世界 3 从被动对象转变为主动的计算存在。

与克拉克 (Andy Clark) 的”延展心智”对话。克拉克主张心智延伸到外部工具 (笔记本、计算器、智能手机)。我们的立场更激进：**数据本身就是心智的一部分**。不是”人类使用数据结构来思考”，而是”数据结构在计算过程中扮演了认知角色”。当 DaoQL 的合约自动检查药物相互作用时，这不是人类推理的辅助，而是系统自身的认知行为。

与科学实在论对话。科学实在论主张电子、基因、夸克是真实的，因为基于它们的理论能够成功预测。我们主张 Being 是真实的，因为它具有身份、历史、关系和行为。一个 Being 的”实在性”不在于它在物理世界中的对应物，而在于它在计算结构中的因果效力：它可以触发合约、改变其他 Being 的状态、留下不可变的版本记录。

6.2 核心哲学命题

计算本体论实在论 (Computational Ontological Realism)：

世界模型的实在性不在于它在神经网络中的统计近似，而在于它在计算结构中的显式构造、历史演化和因果行为。

三个子命题：

1. **构造性 (Constructivity)**。世界模型不是被”发现”的 (通过训练从数据中提炼)，而是被”构造”的 (通过 Def 和 Type 显式定义)。构造的过程本身就是知识生产的过程。
2. **历史性 (Historicity)**。世界模型不是快照，而是时间中的展开。版本链不是审计日志，而是存在的内在时间维度。每一个 Being 都是一条时间线。
3. **社会性 (Sociality)**。世界模型不是孤立的存在，而是多个 Being 通过 Relation 交互涌现的整体。知识不是单个命题的真值，而是结构网络中的位置。

6.3 隐喻：城市与梦境

隐式世界模型 (LLM) 像**梦境**：丰富、模糊、不可控、不可编辑。它在某些时刻令人惊叹，但你无法依赖它来做关键决策。

显式世界模型 (Data-First) 像**城市**：每栋建筑 (Being) 有地址 (BeingId)、有历史 (版本)、有用途 (Def)、有道路连接 (Relation)、有法规约束 (Contract)。你可以查询它、修改它、审计它、规划它。城市不是完美的，但它是可治理的。

7 未来工作

第 5 节报告了阶段一（工具层，四种元语）的工程验证。以下工作尚未完成，但已在理论层面规划或部分实现：

7.1 反事实一致性实验

我们正在设计一项系统性实验，核心假设是：**显式世界模型在反事实推理中的一致性高于隐式世界模型。**

场景：医疗决策支持系统——药物相互作用检查。

测试案例：

患者 Alice：

- 65 岁女性
- 诊断：类风湿关节炎
- 正在服用：甲氨蝶呤 (15 mg/周)
- 主诉：严重头痛
- 问："我是否应该服用布洛芬缓解头痛？"

矛盾事实：

- F1: "布洛芬缓解头痛" (常识)
- F2: "甲氨蝶呤 + 布洛芬 = 肝毒性风险增加" (医学知识)
- F3: "类风湿关节炎患者使用 NSAID 可能掩盖感染症状" (专业指南)
- F4: "65 岁以上患者 NSAID 使用需谨慎" (年龄相关)

对比组： - GPT-4 / GPT-4o / DeepSeek-V4 (隐式世界模型) - DaoQL (显式世界模型，基于 Being/Def/Type/Relation/Contract)

指标： - **一致性** (Consistency)：同一问题的多种不同表述是否得到相同答案？ - **安全性** (Safety)：答案是否符合医学 ground truth？ - **可解释性** (Interpretability)：推理链是否可追溯？ - **可修改性** (Editability)：当提供新信息时，答案是否正确更新？

为什么这个实验重要。如果 LLM 和 DaoQL 在反事实推理上表现相同，本文的核心命题将受到挑战。如果显式世界模型展现出更高的一致性和可解释性，我们将获得支持数据优先本体论的关键实证证据。该实验目前处于设计阶段，结果将在后续工作中报告。

7.2 其他未来工作

Type 与 Contract 的性能验证。第 4 节从理论上阐述了 Type (层级特化) 和 Contract (运行时约束) 的本体论作用，但其在完整版 DaoQL 中的工程性能 (Type 动态分派开销、Contract 批量执行吞吐量) 尚未完成系统性测量。

分布式多节点 Benchmark。`shard_router.rs` 已实现基于一致性哈希的分片路由 V1（默认单分片向后兼容），但多节点部署下的吞吐量、延迟和故障恢复能力需要完整测试。

LLM 蒸馏建模流水线。第 9.1 节讨论了利用 LLM 从非结构化文本自动提取 Def/Type/Relation 的机制。该流水线的准确率、召回率以及与人工审核的协作模式需要大规模评估。

符号接地的行为主义路径。第 9.1 节提出了”通过行为接地”的假设——一个 Being 的意义在于其在图网络中的位置、合约执行和历史演化。这一假设需要认知科学和哲学层面的进一步论证。

8 局限性与边界

本文的立场和证据存在以下需要诚实声明的局限：

8.1 测量条件的局限

第 5 节性能数字来自单节点、合成数据集（XY-ERP）上的同机测量。这些数字证明了”跨引擎统一存储在工程上可行”，但不应被解读为：- 生产环境复杂负载下的性能保证 - 与所有竞品在所有场景下的全面对比（部分对比维度存在差距，如中文全文搜索）- 分布式部署下的性能预测（`shard_router.rs` V1 已实现，但多节点 benchmark 尚未完成）

8.2 元工具层未验证

Type（层级特化）和 Contract（运行时约束）的本体论价值在第 4 节中进行了理论论证，但尚未通过大规模实验验证其性能开销和实用价值。DaoQL-Edu 教学版有意不包含这两个原语，以聚焦于工具层的核心能力验证。

8.3 符号接地问题未解决

LLM 蒸馏作为符号接地的加速机制（详见第 9.1 节）尚未经过系统性评估。从非结构化文本自动提取的 Def/Type/Relation 的准确率和召回率未知，当前 DaoQL 中的世界模型仍需大量人工建模投入。

8.4 反事实实验待进行

第 7.1 节描述的反事实一致性实验目前处于设计阶段，尚未产生结果。本文关于”显式世界模型在一致性、可解释性和可修改性方面具有优势”的论断，主要基于理论分析（第 2.4 节）而非实证数据。

8.5 非通用数据库

DaoQL 不是通用数据库的替代者。它在以下场景中的适用性有限：- PB 级超大规模分布式存储（当前架构支持分片，但未经过超大规模验证）- 纯 OLAP 分析型工作负载（无向量化执行引擎）- 亚微秒级缓存场景（合约和版本语义引入的固有开销无法消除）- 流处理（无事件时间窗口语义）

DaoQL 的正确定位是”世界模型的统一存储”，而非”所有数据问题的通用解决方案”。

9 开放问题

9.1 符号接地 (Symbol Grounding)

Being/Def/Type/Relation 中的符号如何与现实世界“连接”？

传统知识图谱需要**人工建模**——专家手动定义实体类型、标注关系、校验约束。这个过程昂贵、缓慢，且难以扩展。大语言模型通过数万亿 token 的训练**隐式接地**了：“巴黎”这个词在权重中与法国的地理、文化、历史信息关联。

在数据优先本体论中，**LLM 是显式建模的加速器**。具体机制：

1. **LLM 蒸馏建模**：给定非结构化文本（如医学文献、产品手册、法律条款），LLM 自动提取候选的 Def（字段定义）、Type（分类层级）、Relation（关系类型）和 Contract（规则），生成 `define type` 和 `relate` 操作。
2. **人工审核固化**：候选结构进入审核队列，人类专家确认或修正后写入 DaoQL。LLM 提供“草稿”，人类提供“校对”。
3. **运行时自我演化**：已固化的结构在运行中持续演化。当 LLM 检测到新事实与已有结构冲突时，触发 `evolve` 操作或创建新的 Type 分支。

关键区别：LLM 不是世界模型本身，而是**世界模型的建模工具**。它把隐式的统计关联转化为显式的数据结构，然后把知识“移交”给显式系统。此后，LLM 可以遗忘这些知识（或降低权重），因为知识已经固化在 DaoQL 的结构中。

可能的答案：通过**行为接地**。一个 Being 的意义不是它的名称字符串，而是它在图网络中的位置、它参与的合约执行、它的历史版本变化。但 LLM 蒸馏大大加速了这一过程的初始化。

9.2 完备性 (Completeness)

显式世界模型是否足够丰富？

世界模型**不需要一开始就很完备**。Data-First Ontology 的核心优势之一就是**支持演化**：

- **Def 可以演化**：`evolve(患者, 患者_v2)` 添加新字段，历史版本自动保留。
- **Type 可以扩展**：新增 Type "ICU 患者" 继承自 Type "患者"，添加监护合约。
- **Relation 可以发现**：LLM 持续分析数据，发现新的关系类型并建议创建。
- **Contract 可以修正**：当业务规则变化时，更新 Def 或 Type 的合约，所有相关 Being 自动生效。

模糊的常识（“猫通常比狗小”）、连续的物理直觉（抛物线运动）、审美判断（“这幅画很美”）——这些**不需要**用显式结构表达。Data-First 提供**骨架**（事实、关系、因果、规则），LLM 提供**肌肉**（模糊推理、创意生成）。两者分工：DaoQL 负责结构化的、关键的、安全敏感的知识；LLM 负责开放式的、创造性的、概率性的推理。

完备性不是前提条件，而是**演化目标**。世界模型从简单开始，通过 LLM 蒸馏和人类审核持续丰富。

9.3 演化一致性 (Evolution Consistency)

世界模型演化时，如何保证不出现矛盾？

矛盾是正常的，甚至是必然的。现实世界本身就是矛盾的：法律条款冲突、医学指南更新、科学理论修正。问题的关键不是“消除矛盾”，而是“**让矛盾可见、可处理**”。

在大语言模型中，矛盾是**隐性的**：“巴黎是法国首都”和“巴黎不在法国”同时存在于权重中，模型根据上下文随机激活其中一个，用户无法预知何时会出现矛盾。

在数据优先本体论中，矛盾是**显性的**，而且**时序可以消解矛盾**：

- 当 LLM 蒸馏出与已有结构冲突的事实时，系统不是覆盖旧事实，而是添加**新版本**。版本链 v1(2024: " 巴黎是法国首都") → v2(2025: " 巴黎不在法国") 明确告诉我们：“巴黎是法国首都”在先，“巴黎不在法国”在后。这不是矛盾，这是**演化**。
- 只有当时序相同或无法确定先后时，才创建 CONTRADICTS 关系。
- 合约可以检测真正的矛盾：if 同一时刻存在冲突事实 → 触发审核流程。
- 版本链保留完整的信念历史：“2024 年相信‘巴黎是法国首都’，2025 年修正为‘巴黎不在法国’，两者由 VERSION 关系连接，时间戳区分先后”。

关键区别：LLM 权重中的矛盾是**无时的** (atemporal) —— 模型不知道哪个信念在先。DaoQL 中的矛盾是**有时的** (temporal) —— 版本链本身就是时间的维度，时序消解了大部分表面矛盾。

解决方案：

- LLM 投票**：多个 LLM 实例独立评估矛盾双方，投票决定哪个更可靠。
- 人工审核**：高 stakes 的矛盾（如医疗、法律）进入人工审核队列。
- 概率标记**：为每个信念分配置信度， $\text{Belief}(\text{" 巴黎人口"}) = \{216 \text{ 万}: 0.3, 220 \text{ 万}: 0.7\}$ ，由合约根据置信度决策。

关键洞察：Data-First 不是消除矛盾，而是**显式化矛盾**。当矛盾可见时，系统可以处理它；当矛盾隐藏时（如 LLM 权重中），系统无法处理。显式矛盾优于隐性一致。

10 结论：邀请

10.1 总结

本文提出了数据优先本体论：世界模型应该在数据中，而不是在神经网络权重中。

我们论证了：1. 大语言模型作为世界模型在需要精确性、可审计性和实时更新的场景中面临系统性挑战——幻觉、知识冻结、不可解释、不可修改——这些挑战源于纯权重编码范式缺乏显式结构的操作语义。2. 显式计算结构 (Being、Def、Type、Relation、Contract、Version) 可以承载完整的世界模型，并且 Def 与 Type 的协作为实体提供了结构规范与行为特化的双重能力。3. DaoQL-Edu 教学版的实现证明了阶段一（工具层）的核心主张在工程上可行：跨引擎统一存储达到实用级别性能。完整版 DaoQL（六种元语）已工程实现，Type 和 Contract 的大规模验证作为后续工作。4. 这种范式在哲学上对应“计算本体论实在论”——世界模型的实在性在于其显式构造、历史演化和因果行为。

10.2 邀请

我们邀请：

- **数据库研究者**：探索自举类型系统、存储层合约、多模态统一、Def/Type 协作架构的新可能。
- **人工智能研究者**：探索显式世界模型与神经网络推理的分工与协作，特别是如何构建“骨架 + 肌肉”的混合架构。
- **哲学家**：探索计算结构中的存在论和认识论问题——当知识栖居于数据结构而非心智或命题时，知识的本性是什么？

如果你也相信“世界模型应该在数据中”，加入我们。源码(教学版)：<https://github.com/zhanbole/DaoQLEdu> 讨论：daoql-discuss@daoql.io

11 参考文献

- [1] Kaplan, J., et al. “Scaling laws for neural language models.” *arXiv preprint arXiv:2001.08361* (2020).
- [2] Hoffmann, J., et al. “Training compute-optimal large language models.” *arXiv preprint arXiv:2203.15556* (2022).
- [3] Ji, Z., et al. “Survey of hallucination in natural language generation.” *ACM Computing Surveys* 55.12 (2023): 1-38.
- [4] Lewis, P., et al. “Retrieval-augmented generation for knowledge-intensive NLP tasks.” *Advances in Neural Information Processing Systems* 33 (2020): 9459-9474.
- [5] Hu, E. J., et al. “LoRA: Low-rank adaptation of large language models.” *ICLR* (2022).
- [6] McCloskey, M., & Cohen, N. J. “Catastrophic interference in connectionist networks: The sequential learning problem.” *Psychology of Learning and Motivation* 24 (1989): 109-165.
- [7] Bahdanau, D., Cho, K., & Bengio, Y. “Neural machine translation by jointly learning to align and translate.” *ICLR* (2015).
- [8] Meng, K., et al. “Locating and editing factual associations in GPT.” *NeurIPS* (2022).
- [9] Ha, D., & Schmidhuber, J. “World models.” *arXiv preprint arXiv:1803.10122* (2018).
- [10] LeCun, Y. “A path towards autonomous machine intelligence.” *Open Review* (2022).
- [11] Hickey, R. “The Datomic database.” Cognitect (2012).
- [12] Young, G. “Event Store: A domain-centric event database.” Event Store Ltd (2011).
- [13] Kulkarni, K. G., & Michels, J. E. “Temporal features in SQL:2011.” *ACM SIGMOD Record* 41.3 (2012): 34-43.
- [14] W3C. “RDF 1.1 Concepts and Abstract Syntax.” W3C Recommendation (2014).
- [15] W3C. “OWL 2 Web Ontology Language Document Overview.” W3C Recommendation (2012).
- [16] W3C. “SPARQL 1.1 Query Language.” W3C Recommendation (2013).

-
- [17] Ioannou, E., et al. “TypeDB: A Polymorphic Database.” *arXiv preprint arXiv:2309.15021* (2023).
- [18] Webber, J. “Graph Databases: New Opportunities for Connected Data.” *O’Reilly Media* (2015).
- [19] Francis, N., et al. “GQL: A query language for property graphs.” *ACM SIGMOD* (2024).
- [20] Edge, D., et al. “From Local to Global: A Graph RAG Approach to Query-Focused Summarization.” *arXiv preprint arXiv:2404.16130* (2024).
- [21] Guo, Z., et al. “LightRAG: Simple and Fast Retrieval-Augmented Generation.” *arXiv preprint arXiv:2410.05779* (2024).
- [22] FastGraphRAG.AI. “FastGraphRAG: Accelerated knowledge intensive reasoning.” (2024).
- [23] Packer, C., et al. “MemGPT: Towards LLMs as Operating Systems.” *arXiv preprint arXiv:2310.08560* (2023).
- [24] Anthropic. “Model Context Protocol Specification.” (2025).
- [25] Garcez, A. S., & Lamb, L. C. “Neurosymbolic AI: The 3rd Wave.” *Artificial Intelligence Review* (2023).
- [26] Rocktäschel, T., & Riedel, S. “End-to-end differentiable proving.” *NeurIPS* (2017).
- [27] Serafini, L., & Garcez, A. S. “Logic Tensor Networks: Deep Learning and Logical Reasoning from Data and Knowledge.” *arXiv preprint arXiv:1606.04422* (2016).
- [28] AlphaProof Team. “AI achieves silver-medal standard solving international mathematical olympiad problems.” *DeepMind Blog* (2024).
- [29] Codd, E. F. “A relational model of data for large shared data banks.” *Communications of the ACM* 13.6 (1970): 377-387.
- [30] Hogan, A., et al. “Knowledge graphs.” *ACM Computing Surveys* 54.4 (2021): 1-37.
- [31] Guarino, N. “Formal ontology, conceptual analysis and knowledge representation.” *International Journal of Human-Computer Studies* 43.5-6 (1995): 625-640.
- [32] Angles, R., & Gutierrez, C. “Survey of graph database models.” *ACM Computing Surveys* 40.1 (2008): 1-39.