

好的，我们来一起分析这个JSON文件内容和你的需求，并提出知识图谱建模和提取的专业建议，最后修改 `config.yaml` 配置。

1. 分析JSON文件和需求

- **JSON结构：** 文件以 `chunk` 为单位组织，每个chunk包含文本内容、token数、ID、层级关系 (`parent/child_ids`)、文件路径、文档ID、章节标题 (`heading`) 和页码。这种结构非常好，因为它保留了文档的原始层级结构，这对于知识图谱构建非常有价值。
`parent_id` 和 `child_ids` 可以用来直接构建文档的章节层级关系。
- **内容特点：** 内容是具体的企业管理规定（应急预案），包含总则、组织管理、职责分工、事件分级、预警、处置流程、依据法规、附件等典型要素。文本中明确提到了组织部门（客运部、信息技术所）、角色（副总经理、主任）、特定术语（客票系统、电子客票）、事件类型（特别重大事件、售票故障）、依据文件（铁办客〔2021〕92号）、时间约束（4小时）、具体操作（汇报、启动应急预案、发售纸质车票）等。
- **用户问题/需求：**
 - 按主题/部门检索规定：需要 `主题`、`文档`、`组织` 实体及它们之间的关系。
 - 按单位/主题检索规定：需要 `组织`、`主题`、`文档` 实体及它们之间的关系。
 - 按主题比较不同文件/时期规定：需要 `主题`、`文档`、`时间` 实体，以及 `版本关系` 或 `时间` 关联，并且能够关联到具体的规定内容（章节或更细粒度的规则）。

2. 知识图谱建模建议

为了满足你的检索和问答需求，建议采用以下核心实体和关系类型：

核心实体类型 (Nodes):

1. **Document (文档):** 代表整个管理规定文件。
 - 属性: `title` (如 "广州局集团公司客票发售和预订系统（含互联网售票部分）应急预案"), `doc_id` (文号, 如 "广铁客发[2022]114号"), `issue_date` (发布日期), `effective_date` (生效日期), `source_file_path`, `full_doc_id` (唯一标识)。
2. **Section (章节):** 代表文档中的一个章节或子章节。
 - 属性: `heading` (章节标题, 如 "1 总则", "2.1 应急组织机构"), `section_number` (如 "1", "2.1"), `content` (该章节的文本内容), `chunk_id` (关联回源数据), `page_numbers`。

3. **Organization (组织)**: 代表公司、部门、单位、小组等。
 - 属性: `name` (规范化名称, 如 "中国铁路广州局集团有限公司", "集团公司客运部", "集团公司应急领导小组").
4. **Role (角色)**: 代表职位或职责身份。
 - 属性: `title` (如 "副总经理", "客运部主任", "售票员").
5. **Topic (主题)**: 代表文档或章节讨论的核心议题或关键词。
 - 属性: `name` (如 "应急组织机构", "事件分级", "售票业务处置").
6. **RegulationItem (规定项)**: (新增建议) 代表一个具体的规则、要求、定义、标准或程序步骤。这比 `章节` 更细粒度, 能更精确地回答关于“规定”的问题。
 - 属性: `description` (规则描述文本), `type` (如 '定义', '要求', '流程', '标准').
7. **EventDefinition (事件定义)**: (从原 `事件` 细化) 代表文档中明确定义的特定事件类型。
 - 属性: `name` (如 "特别重大事件", "售票故障"), `criteria` (定义标准/条件描述).
8. **Action (行动)**: (新增建议) 代表规定中提到的具体操作或行为。
 - 属性: `name` (如 "汇报", "启动应急预案", "发布预警", "办理退票").
9. **ReferenceDocument (引用文件)**: (从原 `法规` 细化) 代表被引用的其他法规、标准或文件。
 - 属性: `title`, `identifier` (如 "铁办客〔2021〕92号").
10. **TimeConstraint (时间约束)**: (从原 `时间` 细化) 代表时间点、时长、频率或期限。
 - 属性: `value` (如 "4小时", "10分钟", "180日内", "每年").

核心关系类型 (Edges):

1. **HAS_SECTION (包含章节)**: Document -> Section (结构关系, 可从JSON的 `parent_id` / `child_ids` 推断)。
2. **HAS_PARENT_SECTION (父章节)**: Section -> Section (结构关系, 可从JSON的 `parent_id` / `child_ids` 推断)。
3. **CONTAINS_ITEM (包含规定项)**: Section -> RegulationItem (将具体规则关联到章节)。
4. **ISSUED_BY (发布单位)**: Document -> Organization (标识文件的发布者)。
5. **APPLIES_TO (适用范围)**: Document/Section/RegulationItem -> Organization/Role (规定适用于哪些单位/角色)。
6. **MENTIONS (提及)**: Document/Section/RegulationItem ->

Organization/Role/Topic/EventDefinition/ReferenceDocument/TimeConstraint (通用提及关系)。

7. **HAS_TOPIC** (涉及主题): Document/Section/RegulationItem -> Topic (标识内容主题)。
8. **RESPONSIBLE_FOR** (负责): Organization/Role -> Action/Topic/RegulationItem/EventDefinition (明确职责)。
9. **REQUIRES_ACTION** (要求行动): RegulationItem/EventDefinition -> Action (规定或事件触发需要执行的动作)。
10. **TRIGGERS** (触发): EventDefinition/Condition -> Action/ResponseLevel (事件发生触发后续动作或响应级别)。
11. **DEFINES** (定义): Section/RegulationItem -> EventDefinition/Topic (某个章节/规定项定义了某个事件或主题)。
12. **BASED_ON** (依据): Document/RegulationItem -> ReferenceDocument (制定依据)。
13. **REFERENCES** (引用): Document/Section/RegulationItem -> ReferenceDocument/Section (内容引用)。
14. **HAS_CRITERIA** (具有标准): EventDefinition -> RegulationItem/TimeConstraint (事件分级的具体标准)。
15. **EFFECTIVE_FROM** (生效于): Document -> TimeConstraint/Date (文件生效时间)。
16. **HAS_VERSION** / **SUPERSEDES** (版本/取代): Document -> Document (处理文件版本关系)。
17. **PART_OF** (属于): Organization -> Organization / Role -> Organization (组织/角色层级关系, 如"办公室设在客运部")。

3. 提取策略建议

采用 混合提取策略:

1. 结构化信息提取 (基于JSON结构):

- 文档和章节节点 (**Document** , **Section**) 及层级关系 (**HAS_SECTION** , **HAS_PARENT_SECTION**): 直接利用JSON中的 **chunk** 信息。每个 **chunk** 可以映射为一个 **Section** 节点。 **chunk_id** , **parent_id** , **child_ids** 用于建立关系。顶层chunk (**parent_id** 为null) 可以关联到 **Document** 节点 (可根据 **file_path** 或 **full_doc_id** 创建/关联) 。 **heading** , **page_numbers** 等作为节点属性。
- 文档元数据: 从 **file_path** , **full_doc_id** 提取信息填充 **Document** 节点的属性。文号、发布日期等可能需要从文件名或文档开头/末尾的特定chunk中用规则或LLM提取。

2. 内容信息提取 (基于LLM):

- 其他实体提取: 对每个 **chunk** 的 **content** , 使用LLM根据更新后的 **config.yaml** 中的实体定义 (**Organization** , **Role** , **Topic** ,

- RegulationItem , EventDefinition , Action ,
ReferenceDocument , TimeConstraint) 进行提取。
- **关系提取:** 对每个 chunk 的 content , 使用LLM根据更新后的 config.yaml 中的关系定义 (APPLIES_TO , MENTIONS , HAS_TOPIC , RESPONSIBLE_FOR , REQUIRES_ACTION , TRIGGERS , DEFINES , BASED_ON , REFERENCES , HAS_CRITERIA , PART_OF 等) 提取文本内明确描述的关系。

3. 后处理与融合:

- **实体规范化/链接:** 使用 config.yaml 中的 normalization 部分将提取到的实体名称 (尤其是 Organization) 映射到规范形式, 确保知识图谱的一致性。
- **关系整合:** 将LLM提取的关系与结构化提取的节点关联起来。例如, 如果LLM从 chunk_id_A 的文本中提取了 (客运部 , 负责 , 协调工作) , 则将此关系链接到代表 chunk_id_A 的 Section 节点, 以及规范化的 集团公司客运部 (Organization)节点和 协调工作 (Action 或 Topic)节点。
- **跨Chunk关系:** 对于跨越多个文本块的关系, 可能需要在更高层级 (如整个文档) 进行关系提取或后处理逻辑来连接。
- **版本关系处理:** HAS_VERSION / SUPERSEDES 可能需要结合文件名、发布日期以及文本内容 (如“废止...”、“修订...”) 来判断。

4. 修改 config.yaml

基于以上建议, 对 config.yaml 进行修改:

```
# config.yaml
# Schema and Normalization Configuration for Knowledge Graph Extraction

schema:
  # Updated list of allowed entity types for extraction by LLM
  # Note: Document and Section nodes/relationships are primarily derived
  entity_types:
    - Organization # 组织: 公司、部门、小组等
    - Role          # 角色: 职位或职责身份
    - Topic         # 主题: 核心议题或关键词
    - RegulationItem # 规定项: 具体的规则、要求、定义、流程、标准
    - EventDefinition # 事件定义: 文件中明确定义的事件类型
    - Action        # 行动: 规定中提到的具体操作或行为
    - ReferenceDocument # 引用文件: 被引用的法规、标准、文件、附件
    - TimeConstraint # 时间约束: 时间点、时长、频率、期限
    # - Person      # 人员 (如果需要可以保留, 但此文档示例中较少)
    # - Keyword     # 关键词 (可以被Topic覆盖或作为Topic的补充)

# Mapping from Chinese entity types to Cypher labels
entity_type_map_cypher:
  Organization: Organization
```

```

Role: Role
Topic: Topic
RegulationItem: RegulationItem
EventDefinition: EventDefinition
Action: Action
ReferenceDocument: ReferenceDocument
TimeConstraint: TimeConstraint
# Person: Person
# Keyword: Keyword
# Document: Document # Primarily from structure
# Section: Section # Primarily from structure

# Updated list of allowed relation types for extraction by LLM
# Note: HAS_SECTION, HAS_PARENT_SECTION are primarily from JSON structure
relation_types:
- APPLIES_TO # 适用范围：规定适用于哪个组织/角色
- MENTIONS # 提及：通用提及关系
- HAS_TOPIC # 涉及主题：内容关于什么主题
- RESPONSIBLE_FOR # 负责：组织/角色负责什么行动/主题/规定
- REQUIRES_ACTION # 要求行动：规定/事件要求执行什么动作
- TRIGGERS # 触发：事件/条件触发什么动作/响应
- DEFINES # 定义：文本定义了什么事件/主题
- BASED_ON # 依据：制定依据是什么引用文件
- REFERENCES # 引用：内容引用了什么文件/章节/附件
- HAS_CRITERIA # 具有标准：事件定义有哪些具体标准/条件
# - EFFECTIVE_FROM # 生效于：(可能需要特定规则提取 Document -> TimeConstraint)
# - ISSUED_BY # 发布单位：(可能需要特定规则提取 Document -> Organization)
# - HAS_VERSION # 版本关系：(可能需要结合元数据和文本判断 Document -> Document)
# - SUPERSEDES # 取代：(可能需要结合元数据和文本判断 Document -> Document)
- PART_OF # 属于：组织/角色层级或归属关系

# Mapping from Chinese relation types to Cypher relationship types
relation_type_map_cypher:
APPLIES_TO: APPLIES_TO
MENTIONS: MENTIONS
HAS_TOPIC: HAS_TOPIC
RESPONSIBLE_FOR: RESPONSIBLE_FOR
REQUIRES_ACTION: REQUIRES_ACTION
TRIGGERS: TRIGGERS
DEFINES: DEFINES
BASED_ON: BASED_ON
REFERENCES: REFERENCES
HAS_CRITERIA: HAS_CRITERIA
# EFFECTIVE_FROM: EFFECTIVE_FROM
# ISSUED_BY: ISSUED_BY
# HAS_VERSION: HAS_VERSION
# SUPERSEDES: SUPERSEDES
PART_OF: PART_OF

```

```
# BELONGS_TO: BELONGS_TO # Can be replaced by PART_OF or RESPONSIBLE
```

```
normalization:
```

```
# Map of raw entity names to their canonical/normalized form (Expand as canonical_map:
```

```
    客运部：集团公司客运部
```

```
    信息技术所：集团公司信息技术所
```

```
    科信部：集团公司科信部
```

```
    财务部：集团公司财务部
```

```
    计统部：集团公司计统部
```

```
    电务部：集团公司电务部
```

```
    供电部：集团公司供电部
```

```
    宣传部：集团公司宣传部
```

```
    调度所：集团公司调度所
```

```
    集团公司应急领导小组办公室：集团公司应急领导小组办公室
```

```
    集团公司应急领导小组：集团公司应急领导小组
```

```
    国铁集团应急领导小组办公室：国铁集团应急领导小组办公室
```

```
    国铁集团应急领导小组：国铁集团应急领导小组
```

```
    国铁集团客运部：国铁集团客运部
```

```
    12306科创中心：12306科创中心
```

```
    广铁集团：中国铁路广州局集团有限公司
```

```
    广州局集团公司：中国铁路广州局集团有限公司
```

```
    集团公司：中国铁路广州局集团有限公司
```

```
    "中国铁路广州局集团有限公司"：中国铁路广州局集团有限公司 # Self-reference for  
    本预案：《广州局集团公司客票发售和预订系统（含互联网售票部分）应急预案》
```

```
    《铁路客票发售和预订系统（含互联网售票部分）应急预案》：《铁路客票发售和预订系统（含
```

```
    《广州局集团公司网络安全事件应急预案》：《广州局集团公司网络安全事件应急预案》（广
```

```
    《广州局集团公司信息系统故障应急处置和调查处理办法》：《广州局集团公司信息系统故障
```

```
    客票系统：客票发售和预订系统
```

```
    "客票发售和预订系统"：客票发售和预订系统
```

```
    "附件 1"："附件 1 集团公司客票系统安全事故和突发事件应急处置联系单位及联系方式"
```

```
    "附件 2"："附件 2 客票系统预警情况报告表"
```

```
    "附件 3"："附件 3 客票系统应急情况报告表"
```

```
    "附件 4"："附件 4 集团公司车站应急售票系统使用办法"
```

```
    "附件 5"："附件 5 地区性车站列表"
```

```
    "铁路局"："铁路局集团公司" # Normalization example
```

```
# Prompt 模板配置 (Updated)
```

```
prompts:
```

```
# 实体提取相关定义和模板
```

```
entity_extraction:
```

```
# 实体类型定义说明 (Updated)
```

```
definitions: |
```

```
    实体类型定义：
```

- Organization：涉及的单位、部门、公司、小组等，如 "中国铁路广州局集团有限公司"
- Role：指代具有特定职责的职位或岗位，如 "副总经理"、"客运部主任"、"售票员"。
- Topic：文档、章节或规定的核心议题或关键词，如 "应急组织机构"、"事件分级"、"
- RegulationItem：一个具体的规则、要求、定义、标准或程序步骤的描述性文本。例

- EventDefinition: 文件中明确定义的、有特定名称的事件类型, 通常有关键词如“事
- Action: 规定中提到的具体操作或行为动词/短语。例如: “汇报”、“启动应急预案”、
- ReferenceDocument: 被引用的其他法规、标准、文件名称或附件编号。例如: “《铁
- TimeConstraint: 具体的时间点、时长、频率或期限。例如: “4小时及以上”、“10分

完整的提示词模板 (Updated Example)

template: |

请从以下文本中提取定义的实体类型。

{definitions}

预定义的实体类型列表: {entity_types}

文本:

"""

{content}

"""

请以严格的 JSON 格式输出, 包含一个名为 "entities" 的列表, 其中每个对象包含 "name" 和 "type" 属性。
例如:

```
{{
  "entities": [
    {"name": "集团公司应急领导小组办公室", "type": "Organization"}},
    {"name": "客运部主任", "type": "Role"}},
    {"name": "制定、完善集团公司客票发售和预订系统应急预案", "type": "Regulation"}},
    {"name": "特别重大事件", "type": "EventDefinition"}},
    {"name": "汇报", "type": "Action"}},
    {"name": "《铁路客票发售和预订系统(含互联网售票部分)应急预案》(铁办客〔2021〕15号)", "type": "ReferenceDocument"}},
    {"name": "4小时及以上", "type": "TimeConstraint"}},
    {"name": "应急处置", "type": "Topic"}}
  ]
}}
```

关系提取相关定义和模板

relation_extraction:

关系类型定义说明 (Updated)

definitions: |

关系类型定义 (请仅提取文本段落内明确描述的关系, 实体名称使用文本中的原词):

- APPLIES_TO (适用范围): 指明规定/文件适用于哪个组织或角色。例如: (“本预案”适用于“集团公司”)
- MENTIONS (提及): 一个实体在文本中提到了另一个实体 (通用关系, 当其他关系不适用时)
- HAS_TOPIC (涉及主题): 指出文档/章节/规定项讨论的主题。例如: (“1.1 编制目的”涉及“应急预案”主题)
- RESPONSIBLE_FOR (负责): 指明某个组织或角色负责某项任务、行动或主题。例如: (“集团公司”负责“制定、完善应急预案”)
- REQUIRES_ACTION (要求行动): 规定或事件要求执行某个行动。例如: (“售票故障”要求“启动应急预案”)
- TRIGGERS (触发): 事件或条件触发了某个行动或响应级别。例如: (“特别重大事件”触发“启动应急预案”)
- DEFINES (定义): 文本内容定义了某个概念或事件。例如: (“客票系统突发事件”指“因客票系统故障导致无法正常售票”)
- BASED_ON (依据): 指出制定某文件或采取某行动所依据的文件。例如: (“本预案”, 依据“《铁路客票发售和预订系统(含互联网售票部分)应急预案》(铁办客〔2021〕15号)”)。
- REFERENCES (引用): 文本内容引用了其他文件、章节或附件。例如: (“重点的客运组织”参考“《铁路旅客运输规程》”)。
- HAS_CRITERIA (具有标准): 事件定义或分级具有特定的标准或条件 (通常是Regulation中定义的)

- PART_OF (属于): 描述组织、角色或概念的归属或组成关系。例如: ("集团公司应急

完整的提示词模板 (Updated Example)

template: |

请从以下文本中提取实体之间的关系。请专注于在文本段落中**直接陈述**的关系, 并使用

{definitions}

预定义的关系类型列表: {relation_types}

文本:

"""

{content}

"""

请以严格的 JSON 格式输出, 包含一个名为 "relations" 的列表, 其中每个对象包含例如:

{{

 "relations": [

 {{"source": "集团公司应急领导小组办公室", "target": "集团公司客运部",

 {{"source": "本预案", "target": "《铁路客票发售和预订系统(含互联网售票

 {{"source": "客运部", "target": "协调各相关部门和单位提供技术支持", "

 {{"source": "特别重大事件", "target": "一级应急响应", "type": "TRIG

 {{"source": "特别重大事件", "target": "一类业务全路性影响4小时及以上"

 {{"source": "车站", "target": "汇报", "type": "REQUIRES_ACTION"}}

]

}}

关键修改说明:

1. **实体类型调整:** 增加了 `RegulationItem`, `Action`, `EventDefinition`, `ReferenceDocument`, `TimeConstraint`, 使模型更精确。减少了可能泛泛的 `关键词` 和 `事件`。 `文档` 和 `章节` 主要靠结构化提取。
2. **关系类型调整:** 增加了 `APPLIES_TO`, `CONTAINS_ITEM`, `REQUIRES_ACTION`, `TRIGGERS`, `DEFINES`, `HAS_CRITERIA`, `PART_OF` 等, 以更好地捕捉规则逻辑和细节。 `ISSUED_BY`, `EFFECTIVE_FROM`, `HAS_VERSION`, `SUPERSEDES` 可能更适合规则提取或结合元数据处理, 暂时注释掉LLM提取部分, 但保留了概念。
3. **规范化:** 扩展了 `canonical_map`, 包含更多示例和类型 (如附件)。
4. **Prompt 定义:** 更新了实体和关系类型的定义, 使其更清晰, 并提供了更具体的例子。特别强调 `RegulationItem` 用于捕获具体规则。
5. **Prompt 示例:** 修改了JSON输出示例, 以反映新的实体和关系类型。

后续步骤:

1. **实施混合提取:** 开发代码逻辑, 首先处理JSON结构, 创建 `Document` 和 `Section` 节点及层级关系。然后, 对每个 `chunk` 内容调用LLM进行实体和关系提取 (使用更新后

的 `config.yaml`) 。

2. **迭代优化:** 在实际数据上运行提取流程, 评估结果。根据错误分析, 不断优化 `config.yaml` 中的定义、示例和 `canonical_map`, 调整Prompt模板, 甚至改进后处理逻辑。
3. **知识图谱构建:** 将提取和处理后的结果加载到图数据库 (如Neo4j) 中。
4. **查询与验证:** 使用目标问题 (例如, “客运部负责哪些应急处置任务?”或“查找所有关于‘售票故障’的规定项”) 在图谱上进行查询, 验证其是否能满足需求。

通过这种方式, 你可以构建一个结构清晰、内容丰富的知识图谱, 有效支持企业内部管理规定的检索和问答应用。

5W2H

好的, 采用 5W2H (What, Who, When, Where, Why, How, How much/many) 的思考模式来简化和通用化知识图谱的 Schema 和提取过程是一个非常好的主意。这种模式天然地契合了规定、流程、标准等文档的核心信息要素。

我们可以将 5W2H 映射到更通用的实体和关系类型上:

将 5W2H 映射到核心 KG 元素:

1. What (什么):

- **动作/活动 (Action):** 规定中描述的具体操作、行为、步骤。
 - 实体: `Action` (例如: "提交申请", "进行检查", "佩戴防护设备")
- **对象/资源 (Object/Resource):** 动作作用的对象, 或规定涉及的系统、工具、物品、数据。
 - 实体: `Resource` (例如: "ERP系统", "安全帽", "申请表", "风险等级")
- **主题/议题 (Topic):** 文档、章节或规定的核心内容或分类。
 - 实体: `Topic` (例如: "差旅报销", "安全生产", "数据备份")
- **规定/陈述 (Statement):** (核心泛化) 代表一个具体的规则、要求、定义、标准、目标或陈述。这是对原 `RegulationItem` 的泛化。
 - 实体: `Statement` (例如: "所有员工必须在月底前提交报销申请", "风险等级定义为...")
 - 属性: `statement_type`: 'Requirement', 'Prohibition', 'Permission', 'Definition', 'Standard', 'Goal', 'Condition'

2. Who (谁):

- **执行者/责任方 (Actor):** 执行动作或承担责任的组织、部门、角色或人员。
 - 实体: `Actor` (可以有子类型 `Organization` , `Role`) (例如: "财务部", "项目经理", "员工")
- **适用对象 (Applicable Actor):** 规定适用的对象。
 - 关系: `APPLIES_TO` (`Statement` -> `Actor`)
 - 关系: `PERFORMED_BY` (`Action` -> `Actor`)
 - 关系: `RESPONSIBLE_FOR` (`Actor` -> `Action/Statement/Topic`)

3. When (何时):

- **时间约束 (Time Constraint):** 动作发生的时间点、期限、频率、持续时长。
 - 实体: `TimeConstraint` (例如: "每月5号前", "24小时内", "每年一次", "持续30分钟")
- 关系: `OCCURS_AT` / `HAS_TIMING` (`Action/Statement` -> `TimeConstraint`)

4. Where (何地):

- **地点/位置 (Location/Context):** 动作发生的地点或规定适用的环境/系统。
 - 实体: `Location` (例如: "生产车间", "总部大楼"), 也可复用 `Resource` 代表系统 (例如: "OA系统")
- 关系: `OCCURS_IN` / `APPLIES_IN` (`Action/Statement` -> `Location/Resource`)

5. Why (为什么):

- **原因/依据 (Reason/Basis):** 制定规定或执行动作的原因、目的或依据的法规/文件。
 - 实体: `ReferenceSource` (法规、标准、上级文件)
 - 实体: `Condition` (触发条件、事件) - (从原`EventDefinition`泛化)
 - 实体: `Statement` (当类型为 'Goal' 或 'Purpose' 时)
- 关系: `BASED_ON` (`Statement/Action` -> `ReferenceSource`)
- 关系: `TRIGGERED_BY` / `HAS_CAUSE` (`Action/Statement` -> `Condition`)
- 关系: `HAS_PURPOSE` (`Action/Statement` -> `Statement[type=Goal]`)

6. How (如何):

- **方式/步骤 (Method/Steps):** 执行动作的方式、流程或步骤。
 - 属性: `Action` 实体的 `description` 或 `method` 属性。
 - 关系: `HAS_STEP` / `SEQUENCE` (`Action` -> `Action`) (表示流程顺序)

- **条件/标准 (Criteria/Standard):** 判断或执行的条件或标准。
 - 实体: `Statement` (当类型为 'Condition' 或 'Standard' 时)
 - 关系: `HAS_CRITERIA` (Action/Statement -> Statement[type=Condition/Standard])

7. How much/many (多少):

- **量化信息 (Quantity/Threshold):** 涉及的数量、阈值、限制。
 - 属性: `Resource` 的 `quantity` 属性, `Statement` 的 `threshold` 或 `limit` 属性, `TimeConstraint` 的 `value` 和 `unit` 属性。
 - 实体: (如果复杂可单独建) `Quantity`
- 关系: (如果需要) `HAS_QUANTITY` / `HAS_THRESHOLD`
(Statement/Action/Resource -> Quantity/Value)

基于 5W2H 的通用化 Schema (config.yaml 调整建议):

```
# config.yaml (Revised based on 5W2H)
# Schema and Normalization for General Corporate Regulations KG

schema:
  # Core, generalized entity types based on 5W2H
  entity_types:
    - Statement      # (What rule/req/def/goal? How is it defined?)
    - Actor          # (Who? - Org/Role subtypes possible)
    - Action         # (What activity? How is it done?)
    - Resource       # (What object/system/tool?)
    - Condition      # (Why triggered? Under what condition?)
    - Location       # (Where?)
    - TimeConstraint # (When? How often? How long?)
    - ReferenceSource # (Why based on what? Which document?)
    - Topic          # (What general subject?)
    # - Quantity      # (How much/many? - Optional, often attributes)

  # Mapping to Cypher labels
  entity_type_map_cypher:
    Statement: Statement
    Actor: Actor # Consider using labels like Actor:Organization, Actor:Person
    Action: Action
    Resource: Resource
    Condition: Condition
    Location: Location
    TimeConstraint: TimeConstraint
    ReferenceSource: ReferenceSource
    Topic: Topic
```

```

# Quantity: Quantity

# Core, generalized relation types based on 5W2H connections
relation_types:
    # Structural (Still leverage JSON structure where possible)
    - CONTAINS          # (Document->Section, Section->Statement)
    - HAS_PARENT        # (Section->Section)
    # Semantic (Extracted by LLM based on 5W2H prompts)
    - APPLIES_TO        # (Statement/Action -> Actor/Location/Resource) - What?
    - PERFORMED_BY      # (Action -> Actor) - Who does it?
    - RESPONSIBLE_FOR   # (Actor -> Action/Statement/Topic) - Who is responsible?
    - USES              # (Action -> Resource) - What tool/system used?
    - REQUIRES          # (Action/Statement -> Resource/Action) - What is required?
    - TRIGGERED_BY      # (Action/Statement -> Condition) - Why (cause)?
    - HAS_CONDITION     # (Statement/Action -> Condition) - How (under what condition)?
    - HAS_PURPOSE       # (Statement/Action -> Statement[type=Goal]) - Why (purpose)?
    - BASED_ON         # (Statement/Document -> ReferenceSource) - Why (justification)?
    - REFERENCES        # (Statement/Document -> ReferenceSource/Section) - What is referenced?
    - OCCURS_AT         # (Action/Condition -> TimeConstraint) - When?
    - OCCURS_IN         # (Action/Condition -> Location/Resource) - Where?
    - SEQUENCE          # (Action -> Action) - How (steps)?
    # - HAS_QUANTITY    # (Optional relation for How Much/Many)

# Mapping to Cypher relationship types
relation_type_map_cypher:
    CONTAINS: CONTAINS
    HAS_PARENT: HAS_PARENT
    APPLIES_TO: APPLIES_TO
    PERFORMED_BY: PERFORMED_BY
    RESPONSIBLE_FOR: RESPONSIBLE_FOR
    USES: USES
    REQUIRES: REQUIRES
    TRIGGERED_BY: TRIGGERED_BY
    HAS_CONDITION: HAS_CONDITION
    HAS_PURPOSE: HAS_PURPOSE
    BASED_ON: BASED_ON
    REFERENCES: REFERENCES
    OCCURS_AT: OCCURS_AT
    OCCURS_IN: OCCURS_IN
    SEQUENCE: SEQUENCE
    # HAS_QUANTITY: HAS_QUANTITY

normalization:
    # Maintain and expand canonical_map as before
    canonical_map:
        # ... (keep existing and add new ones)
        员工: 全体员工 # Example
        # ...

```

```

# Prompt 模板配置 (Crucially updated for 5W2H)
prompts:
  entity_extraction:
    definitions: |
      实体类型定义 (围绕5W2H识别):
      - Statement: 识别文本中的核心陈述句, 说明**是什么**规则、要求、禁止、许可、定义。
      - Actor: 识别执行动作、承担责任或规定适用的**是谁** (组织、部门、角色、岗位)。
      - Action: 识别具体的操作、行为、活动或步骤 (What is being done? How is it done?).
      - Resource: 识别动作涉及或规定提到的**是什么**对象、系统、工具、文档或数据。
      - Condition: 识别触发动作或规定适用的条件、情况或事件 (Why triggered? Under what circumstances?).
      - Location: 识别动作发生或规定适用的**在哪里** (物理地点或系统环境)。例如: "在服务器上"。
      - TimeConstraint: 识别**何时**发生、频率或持续时间 (When? How often? How long?).
      - ReferenceSource: 识别规定或动作**为什么**基于某个外部或内部的法规、标准、政策。
      - Topic: 识别文本段落讨论的**是什么**通用主题或类别。例如: "费用报销"、"安全政策"。
    template: |
      请依据5W2H框架从以下文本中提取实体。

      {definitions}

      预定义的实体类型列表: {entity_types}

      文本:
      """
      {content}
      """

      请以严格的 JSON 格式输出 "entities" 列表, 包含 "name" (文本原词) 和 "type" (实体类型)。
      例如:
      {{
        "entities": [
          {{ "name": "所有员工必须在月底前提提交报销申请", "type": "Statement" }},
          {{ "name": "员工", "type": "Actor" }},
          {{ "name": "提交报销申请", "type": "Action" }},
          {{ "name": "月底前", "type": "TimeConstraint" }},
          {{ "name": "费用报销", "type": "Topic" }}
        ]
      }}

  relation_extraction:
    definitions: |
      关系类型定义 (连接5W2H要素):
      - APPLIES_TO: 连接 Statement/Action 与其适用的 Actor/Location/Resource。
      - PERFORMED_BY: 连接 Action 与其执行者 Actor (谁执行动作?)。
      - RESPONSIBLE_FOR: 连接 Actor 与其负责的 Action/Statement/Topic (谁负责?)。
      - USES: 连接 Action 与其使用的 Resource (动作使用什么工具/系统?)。
      - REQUIRES: 连接 Action/Statement 与其必需的 Resource/Action (动作/规定需要什么?)。
      - TRIGGERED_BY: 连接 Action/Statement 与其触发条件 Condition (为什么被触发?)。

```

- HAS_CONDITION: 连接 Action/Statement 与其生效的条件 Condition (在什么条件下?)。
- HAS_PURPOSE: 连接 Action/Statement 与其目标 Statement[type=Goal] (做什么?)。
- BASED_ON: 连接 Statement/Document 与其依据 ReferenceSource (为什么?)。
- REFERENCES: 连接 Statement/Document 与其引用的 ReferenceSource/Section (在哪里?)。
- OCCURS_AT: 连接 Action/Condition 与其发生时间 TimeConstraint (何时发生?)。
- OCCURS_IN: 连接 Action/Condition 与其发生地点 Location/Resource (何地发生?)。
- SEQUENCE: 连接前后相继的 Action (如何按步骤?)。

template: |

请依据5W2H要素间的联系，从文本中提取关系。

{definitions}

预定义的关系类型列表: {relation_types}

文本:

"""

{content}

"""

请以严格的 JSON 格式输出 "relations" 列表, 包含 "source", "target" (均为字符串), 例如:

```
{{
  "relations": [
    {{ "source": "提交报销申请", "target": "员工", "type": "PERFORMED_BY" }},
    {{ "source": "所有员工必须在月底前提交报销申请", "target": "员工", "type": "OCCURS_IN" }},
    {{ "source": "提交报销申请", "target": "月底前", "type": "OCCURS_AT" }},
    {{ "source": "执行安全检查", "target": "安全检查表", "type": "USES" }},
    {{ "source": "发送警报", "target": "当系统发生故障时", "type": "TRIGGERS" }}
  ]
}}
```

优势:

- **通用性:** 5W2H 是分析性问题的通用框架, 适用于大多数描述性、规定性、流程性的文档。
- **简化 Schema:** 实体和关系类型更少、更核心, 易于理解和维护。
- **指导 LLM:** Prompt 更清晰地指导 LLM 关注文本中的关键信息要素, 有望提高提取准确性和一致性。
- **直观映射:** KG 中的节点和关系能更直观地回答 "谁在何时何地因何故如何做了什么" 这类问题, 非常适合 RAG 场景。

注意事项:

- **信息粒度:** Statement 实体可能包含较长的文本。需要考虑是否需要进一步将其分解或提取其核心动词/对象作为单独的 Action / Resource 节点。这可以在后处理中完成, 或者调整 Prompt 要求 LLM 同时提取核心 Action。
- **属性 vs. 节点:** "How much/many" 主要通过属性体现。需要决定哪些信息作为属性, 哪

些作为独立节点（例如，`TimeConstraint` 设为节点比设为属性更灵活）。

- **隐式信息:** LLM 主要提取文本中明确表述的关系。隐式的逻辑关系（例如，一个段落整体描述一个流程，但没有明确的“然后”、“下一步”字眼）可能需要更复杂的推理或后处理规则。
- **结构信息融合:** 别忘了将 LLM 提取的语义信息与 JSON 中的结构信息（文档、章节、层级）结合起来，形成完整的知识图谱。`chunk_id` 是关键链接。

通过这种 5W2H 驱动的方法，你可以构建一个既通用又强大的知识图谱基础，为 RAG 系统提供有效的知识支撑。

继续修改，增加文档、章节、组织、角色的类型

Okay, incorporating `Document`, `Section`, and `Organization` explicitly while leveraging the 5W2H framework is a good refinement. `Document` and `Section` provide essential structural context, and `Organization` is a critical and ubiquitous entity type in corporate regulations.

Here's how we can integrate them and refine the suggestions:

Integration Strategy:

1. **`Document` & `Section` as Structural Context:** These types primarily define the *container* and *location* of the information within the source material. They are best derived from the JSON structure (`chunk`, `heading`, `file_path`, `parent_id`, `child_ids`). While LLMs *could* extract mentions of document/section titles within the text, their primary role in the KG is structural. Semantic entities (Statements, Actions, etc.) will be *contained within* Sections.
2. **`Organization` as a Key Actor Subtype:** `Organization` directly answers the "Who" question. It's a specific, crucial type of `Actor`. We should keep it as a top-level, explicit entity type for extraction because of its importance and frequency.
3. **Retain `Role`:** Similarly, `Role` is another key subtype of `Actor` (answering "Who") and should remain an explicit extraction target.
4. **Maintain 5W2H Core:** The other 5W2H-derived entities (`Statement`, `Action`, `Resource`, `Condition`, `Location`, `TimeConstraint`, `ReferenceSource`, `Topic`) remain the core for capturing the *semantic content* within each Section.

Revised Schema and Configuration (`config.yaml`):

```
# config.yaml (Revised with Document, Section, Organization + 5W2H)
# Schema and Normalization for General Corporate Regulations KG
```

schema:

Entity types: Structural + Key Actors + 5W2H Semantic Core

entity_types:

- # Structural (Primarily from JSON structure, but LLM might identify more)
- Document
- Section
- # Key Actors (Explicit extraction targets answering "Who")
- Organization
- Role
- # 5W2H Semantic Core (Extracted by LLM from content)
- Statement # (What rule/req/def/goal? How defined?)
- Action # (What activity? How done?)
- Resource # (What object/system/tool?)
- Condition # (Why triggered? Under what condition?)
- Location # (Where?)
- TimeConstraint # (When? How often? How long?)
- ReferenceSource # (Why based on what? Which document/standard?)
- Topic # (What general subject?)

Mapping to Cypher labels

entity_type_map_cypher:

Document: Document
Section: Section
Organization: Organization # Explicitly Org
Role: Role # Explicitly Role
Statement: Statement
Action: Action
Resource: Resource
Condition: Condition
Location: Location
TimeConstraint: TimeConstraint
ReferenceSource: ReferenceSource
Topic: Topic
Note: Organization and Role nodes could also be labeled as :Actor

Relation types: Structural + Semantic (5W2H based)

relation_types:

- # Structural (Primarily from JSON structure)
- HAS_SECTION # (Document -> Section)
- HAS_PARENT_SECTION # (Section -> Section)
- CONTAINS # (Section -> Statement/Action/Org/Role/etc. extracted)
- # Semantic (Extracted by LLM based on 5W2H prompts, connecting semantic)
- APPLIES_TO # (Statement/Action -> Organization/Role/Location/TimeConstraint/ReferenceSource/Topic)
- PERFORMED_BY # (Action -> Organization/Role) # Connects to specific actor
- RESPONSIBLE_FOR # (Organization/Role -> Action/Statement/Topic) # Connects to specific action/statement
- USES # (Action -> Resource)
- REQUIRES # (Action/Statement -> Resource/Action)


```

- TRIGGERED_BY      # (Action/Statement -> Condition)
- HAS_CONDITION      # (Statement/Action -> Condition)
- HAS_PURPOSE        # (Statement/Action -> Statement[type=Goal])
- BASED_ON           # (Statement/Document -> ReferenceSource)
- REFERENCES          # (Statement/Document/Section -> ReferenceSource/Section)
- OCCURS_AT          # (Action/Condition -> TimeConstraint)
- OCCURS_IN           # (Action/Condition -> Location/Resource)
- SEQUENCE            # (Action -> Action)
- MENTIONS            # (Generic mention relation, e.g., Section -> Topic)

```

Mapping to Cypher relationship types

relation_type_map_cypher:

```

HAS_SECTION: HAS_SECTION
HAS_PARENT_SECTION: HAS_PARENT_SECTION
CONTAINS: CONTAINS # Links structure to content
APPLIES_TO: APPLIES_TO
PERFORMED_BY: PERFORMED_BY
RESPONSIBLE_FOR: RESPONSIBLE_FOR
USES: USES
REQUIRES: REQUIRES
TRIGGERED_BY: TRIGGERED_BY
HAS_CONDITION: HAS_CONDITION
HAS_PURPOSE: HAS_PURPOSE
BASED_ON: BASED_ON
REFERENCES: REFERENCES
OCCURS_AT: OCCURS_AT
OCCURS_IN: OCCURS_IN
SEQUENCE: SEQUENCE
MENTIONS: MENTIONS

```

normalization:

Crucial for Organization, Role, Resource names

canonical_map:

```

客运部: 集团公司客运部
信息技术所: 集团公司信息技术所
# ... (Expand significantly)
广铁集团: 中国铁路广州局集团有限公司
集团公司: 中国铁路广州局集团有限公司
# ...

```

Prompt 模板配置 (Updated to reflect explicit Org/Role and context)

prompts:

entity_extraction:

definitions: |

实体类型定义 (识别文本内容中的语义要素, 已知其所在文档/章节):

- Organization: 识别明确提及的公司、部门、单位、团队或小组名称 (**是谁**中的)
- Role: 识别明确提及的职位、岗位或职责身份 (**是谁**中的角色实体)。例如: "副"
- Statement: 识别核心陈述, 说明**是什么**规则、要求、定义、标准、目标或条件。

- Action: 识别具体操作、行为或步骤 (**做什么**? **如何做**?)。例如: "启动应急预案"
- Resource: 识别涉及的**是什么**对象、系统、工具、文档(非引用源)、数据。例如: "客票系统"
- Condition: 识别触发动作或规定适用的条件、情况或事件 (**为什么**触发? **在什么情况下**?)。例如: "当事故上升为红色、橙色预警时"
- Location: 识别动作发生或规定适用的**在哪里** (物理地点或系统环境)。例如: "集团公司应急领导小组办公室"
- TimeConstraint: 识别**何时**、频率或持续时间。例如: "4小时及以上"、"1个工作日内"
- ReferenceSource: 识别规定或动作**为什么**基于或引用了**哪个**法规、标准、文件等。例如: "附件2"
- Topic: 识别文本段落讨论的**是什么**通用主题。例如: "应急组织机构"、"事件分类与分级"
- # Document/Section: 这些主要来自结构, LLM一般无需提取, 除非文本明确引用*其他部分

template: |

请从以下文本(已知其所属文档和章节)中提取定义的语义实体类型。

```
{definitions}
```

预定义的实体类型列表 (请专注于提取这些): {entity_types}

文本:

```
"""
```

```
{content}
```

```
"""
```

请以严格的 JSON 格式输出 "entities" 列表, 包含 "name" (文本原词) 和 "type" (语义类型)。例如:

```
{{
  "entities": [
    {{ "name": "集团公司应急领导小组办公室", "type": "Organization" }},
    {{ "name": "客运部主任", "type": "Role" }},
    {{ "name": "制定、完善集团公司客票发售和预订系统应急预案", "type": "Statement" }},
    {{ "name": "启动应急预案", "type": "Action" }},
    {{ "name": "客票系统", "type": "Resource" }},
    {{ "name": "当事故上升为红色、橙色预警时", "type": "Condition" }},
    {{ "name": "1个工作日内", "type": "TimeConstraint" }},
    {{ "name": "附件2", "type": "ReferenceSource" }},
    {{ "name": "预警报告", "type": "Topic" }}
  ]
}}
```

relation_extraction:

definitions: |

关系类型定义 (连接文本中的语义要素, 包括组织和角色):

- APPLIES_TO: 连接 Statement/Action 与其适用的 Organization/Role/Location/Resource (适用于什么?)。
- PERFORMED_BY: 连接 Action 与其执行者 Organization/Role (谁执行动作?)。
- RESPONSIBLE_FOR: 连接 Organization/Role 与其负责的 Action/Statement (负责什么?)。
- USES: 连接 Action 与其使用的 Resource (动作使用什么?)。
- REQUIRES: 连接 Action/Statement 与其必需的 Resource/Action (需要什么?)。
- TRIGGERED_BY: 连接 Action/Statement 与其触发条件 Condition (为什么被触发?)。
- HAS_CONDITION: 连接 Action/Statement 与其生效的条件 Condition (在什么条件下生效?)。
- HAS_PURPOSE: 连接 Action/Statement 与其目标 Statement[type=Goal] (做什么?)。
- BASED_ON: 连接 Statement/Document 与其依据 ReferenceSource (依据什么?)。

```

- REFERENCES: 连接 Statement/Document/Section 与其引用的 ReferenceSou
- OCCURS_AT: 连接 Action/Condition 与其时间 TimeConstraint (何时?)。
- OCCURS_IN: 连接 Action/Condition 与其地点 Location/Resource (何地?)
- SEQUENCE: 连接前后相继的 Action (如何按步骤?)。
- MENTIONS: 连接实体与被提及的实体 (通用, 如 Section MENTIONS Topic) 。
# CONTAINS/HAS_SECTION/HAS_PARENT_SECTION: 主要由结构化处理负责。
template: |
    请从以下文本中提取语义实体(包括组织/角色)之间的关系。

    {definitions}

    预定义的关系类型列表: {relation_types}

    文本:
    """
    {content}
    """

    请以严格的 JSON 格式输出 "relations" 列表, 包含 "source", "target" (均为
    例如:
    {{
      "relations": [
        {{ "source": "集团公司应急领导小组办公室", "target": "制定、完善集团公
        {{ "source": "本预案", "target": "集团公司及所属各客运单位", "type":
        {{ "source": "站段客票应急办", "target": "上报", "type": "PERFORMED
        {{ "source": "上报", "target": "集团客票应急领导小组办公室", "type":
        {{ "source": "上报", "target": "第一时间", "type": "OCCURS_AT"}}
      ]
    }}

```

关键调整和说明:

1. **Schema 明确化:** Document, Section, Organization, Role 现在是 Schema 中明确的一等公民。
2. **Prompt 调整:**
 - 实体提取 Prompt 现在明确要求提取 Organization 和 Role, 并提供了相应的例子。它也澄清了 Document / Section 主要来自结构。
 - 关系提取 Prompt 明确了 PERFORMED_BY 和 RESPONSIBLE_FOR 等关系可以连接到 Organization 和 Role。
3. **CONTAINS 关系:** 增加了一个 CONTAINS 关系, 这个关系主要在后处理阶段添加, 用于连接结构化的 Section 节点和从该 Section 文本中提取出的所有语义节点 (Statement, Action, Organization mention 等)。这是连接结构和内容的关键。
4. **MENTIONS 关系:** 保留了通用的 MENTIONS 关系, 可用于捕捉文本中实体间的简单

共现或提及，特别是当其他更具体的关系不适用时（例如，一个章节 MENTIONS 一个 Topic）。

5. 实现流程微调:

- **步骤1 (Structure):** 不变，创建 :Document , :Section 节点和 :HAS_SECTION , :HAS_PARENT_SECTION 关系。
- **步骤2 (Semantics - LLM):** 不变，使用新 Prompts 提取语义实体和关系。
- **步骤3 (Integration):**
 - 创建语义节点 (:Organization , :Role , :Statement , etc.) 和它们之间的语义关系 (:RESPONSIBLE_FOR , :PERFORMED_BY , etc.)。
 - 对于从 chunk_x 提取的每个语义节点 s , 添加关系 (:Section {chunk_id: 'chunk_X'})-[:CONTAINS]->(s) 。

这个调整后的方法保留了 5W2H 的通用性和语义核心，同时给予了结构 (Document , Section) 和关键参与者 (Organization , Role) 应有的、明确的地位，更符合实际应用中对这些类型实体的查询需求。