

Agora Deliberation Schema Map (Updated)

Plain-English system designer's map of the deliberation platform

Below is a comprehensive, product-oriented explanation of how your deliberation schema works. This document groups models by cognitive function, explains what each record means in the product, highlights key relations and invariants, and traces typical data flows.

0) Mental Model: What a Deliberation "Is"

One deliberation = one bounded collaborative discourse with structure, memory, and governance.

People contribute **propositions** (lightweight assertions), promote them to **claims** (canonical propositions), construct **arguments** (reasoned lines of support), **link** them via edges (support/rebut/undercut/conflict/preference schemes), **diagram** internal inference structure, attach **evidence**, apply **argumentation schemes & critical questions**, form **clusters** and select **viewpoints**, participate in formal **dialogue moves** (including **ludics**-style proof games), moderate via **panels** and **receipts**, publish **briefs**, run **votes**, develop **theory works** (DN/IH/TC/OP frameworks), curate knowledge in **KB pages**, and connect across a **graph-of-graphs** via rooms, imports, and functors.

High-level structure:

Deliberation

- └─ Social substrate (Proposition → Claim promotion path)
- └─ Claims & Edges (Claim, ClaimEdge, ClaimLabel, ClaimEvidence/Citation, CanonicalClaim)
- └─ Arguments & Edges (Argument, ArgumentEdge, ArgumentPremise, ArgumentApproval, ArgumentSupport, AssumptionUse)
- └─ Conflict & Preference (ConflictScheme, ConflictApplication, PreferenceScheme, PreferenceApplication, DefaultRule)
- └─ Argument internals (ArgumentDiagram, Statement, Inference, InferencePremise)
- └─ Schemes & CQs (ArgumentScheme, SchemeVariant, SchemeInstance, CQStatus, CriticalQuestion)
- └─ Evidence (EvidenceNode, EvidenceLink)
- └─ Sheet-level maps (DebateSheet, DebateNode, DebateEdge, LocusStatus, SheetAcceptance, UnresolvedCQ, Outcome)
- └─ Clusters & Viewpoints (Cluster, UserCluster, ArgumentCluster, ViewpointSelection, ViewpointArgument, BridgeRequest/Assignment, AmplificationEvent)

- └ Governance & moderation (ContentStatus, DecisionReceipt, RoomLogbook, Panel, Panelist)
- └ Dialogue & burden (DialogueMove, ProofMode, ReplyTarget)
- └ Ludics (proof games: Locus, Design, Act, Chronicle, Trace, Commitment, Behaviour, DecisionReceipt)
- └ Voting (VoteSession, VoteBallot)
- └ Theory Works (TheoryWork, Work[DN|IH|TC|OP] structures, citations, practical justification, hermeneutic/pascal models)
- └ Knowledge graph (KnowledgeEdge, NLILink, EristicMark, Rule, ReasonPair)
- └ Briefs & Issues (Brief, BriefVersion, BriefLink; Issue, IssueLink)
- └ Inter-room graph (StackReference, ArgumentImport, SharedAuthorRoomEdge, RoomFunctor, XRef)
- └ KB & Wiki (KbSpace/Page/Block/Snapshot; WikiPage/Revision)
- └ Social discussion (Discussion, DiscussionMessage/Reaction/Participant, ForumComment/Vote/Save, DiscussionDeliberation upgrade)
- └ GitChat signals (ProposalSignal, AgreementLock, AgreementAck)

1) Social Substrate: Propositions → Claims

Proposition — lightweight assertion layer

Model: Proposition

Purpose: A pre-claim contribution that can be workshopped, voted on, tagged, and replied to before promotion.

Fields:

- deliberationId, authorId, text, mediaType/mediaUrl
- status: DRAFT | PUBLISHED | CLAIMED | ARCHIVED
- promotedClaimId (unique, nullable): links to the canonical Claim when promoted
- Workshop counters: voteUpCount, voteDownCount, endorseCount, replyCount
- legacyArgumentId: migration helper

Relations:

- promotedClaim: one-to-one with Claim (when promoted)
- votes: PropositionVote[] (user, value)
- endorsements: PropositionEndorsement[]
- replies: PropositionReply[]
- tags: PropositionTag[]

Cognitive role: The "workshop space" where ideas are floated, refined, and validated before becoming canonical claims. Allows social signals (votes, endorsements) to filter quality.

Typical flow:

1. User creates a Proposition (status: PUBLISHED)
2. Others vote, endorse, reply, tag

3. When ready, promote → creates Claim, sets promotedClaimId, changes status to CLAIMED

2) Claims and Their Graph

Claim — canonical atomic proposition

Model: Claim

Purpose: The fundamental unit of assertion; structured, persistent, and semantically labeled.

Fields:

- text, createdById, moid (unique ID), claimType (Agent, Assertion, Domain, etc.)
- deliberationId (nullable): can exist across deliberations
- canonicalClaimId: links to a CanonicalClaim family (cross-room equivalence)
- negatesClaimId: explicit negation link (for duality-based semantics)
- canonicalKey: unique stable identifier

Relations:

- cards: DeliberationCard[] (longform representations)
- warrant: ClaimWarrant (textual backing)
- arguments: Argument[] (arguments that link to this claim via claimId)
- asPremiseOf: ArgumentPremise[] (when this claim is a premise)
- asConclusion: Argument[] (when this claim is a conclusion)
- edgesFrom/edgesTo: ClaimEdge[] (claim-to-claim relations)
- citations: ClaimCitation[] (bibliographic anchors)
- ClaimEvidence[], ClaimLabel, claimValues: ClaimValue[] (value annotations)
- sourceProposition: nullable back-link to Proposition
- urns: Urn[] (stable URN identifiers)
- negates/negatedBy: negation relations
- debateNodes: DebateNode[] (when visualized in sheets)

Cognitive role: The "thing asserted"—truth-apt, citable, label-able, and wired into the argument graph.

ClaimEdge — claim-to-claim relation

Model: ClaimEdge

Purpose: Links claims with typed relations (supports, rebuts) and refined attack types.

Fields:

- fromClaimId, toClaimId
- type: ClaimEdgeType (supports | rebuts)

- **attackType:** ClaimAttackType? (SUPPORTS | REBUTS | UNDERCUTS | UNDERMINES)
- **targetScope:** premise | inference | conclusion
- **deliberationId** (nullable)

Cognitive role: How claims compose into argumentation at the claim level.

Undercuts target warrants/inferences, not just conclusions.

Unique constraint: [fromClaimId, toClaimId, type, attackType] prevents duplicates.

Evidence & Citations for Claims

- **ClaimEvidence:** lightweight (uri, title, citation, addedById)
- **ClaimCitation:** heavier (locatorStart/End, excerptHash, snapshotKey, cslJson, note)
- **ClaimWarrant:** textual warrant (1:1 with Claim)

Families & Labeling

- **CanonicalClaim:** cross-room family (slug, title, summary) — claims from different deliberations can map to the same canonical claim
- **Urn:** stable URN for claim | card | brief_version (entityType, entityId, urn)
- **ClaimStats:** per-deliberation counters (approvalsCount, supportsCount, rebutsCount, undercutsCount)
- **ClaimLabel:** semantic labeling (grounded/preferred/hybrid) with GroundLabel (IN | OUT | UNDEC) and explainJson

3) Arguments and Their Graph

Argument — authored line of reasoning

Model: Argument

Purpose: A narrative unit of reasoning, potentially linking multiple premises to a conclusion.

Fields:

- **deliberationId, authorId, text, sources (JSON), confidence**
- **isImplicit:** whether premises are assumed background knowledge
- **schemeld:** optional link to ArgumentScheme
- **conclusionClaimId:** the claim this argument concludes
- **claimId:** legacy/alternative claim link
- **quantifier:** Quantifier (SOME|MANY|MOST|ALL)
- **modality:** Modality (COULD|LIKELY|NECESSARY)
- **mediaType, mediaUrl**
- **implicitWarrant (JSON):** enthymeme storage

Relations:

- **deliberation:** parent

- premises: ArgumentPremise[] (join table to Claim)
- conclusion: Claim (via conclusionClaimId)
- outgoingEdges/incomingEdges: ArgumentEdge[]
- approvals: ArgumentApproval[]
- ViewpointArgument[], IssueLink[], debateNodes[]
- arglImportsSource/Target: ArgumentlImport[] (cross-deliberation imports)

Cognitive role: The authored, narrative container—can be structured or free-form, and linked to claims and other arguments.

ArgumentEdge — argument-to-argument relation

Model: ArgumentEdge

Purpose: Typed link between arguments (support, rebut, undercut, concede, CA).

Fields:

- deliberationId, fromArgumentId, toArgumentId
- type: EdgeType (support | rebut | undercut | concede | CA)
- attackSubtype: ArgumentAttackSubtype? (SUPPORT_ATTACK, CONSEQUENCE_ATTACK, JUSTIFICATION_ATTACK, UNDERMINE, REBUT, UNDERCUT, OVERCUT)
- targetScope: conclusion | premise | inference
- targetInferencelId: **fine-grain targetting** (specific inference inside toArgument)
- inferencelId, cqKey, targetPremiselId, targetClaimId, attackType

Cognitive role: Argument-to-argument relations, including **warrant attacks** (undercuts).

Conflict & Preference Schemes (NEW)

ConflictScheme & ConflictApplication

Purpose: Generalize and make explicit different types of conflicts beyond the basic attack types.

Fields (ConflictScheme):

- key, name, description
- legacyAttackType, legacyTargetScope (for AF compatibility)

Fields (ConflictApplication):

- deliberationId, schemelId, createdById
- Exactly one of: conflictingClaimId, conflictingArgumentId
- Exactly one of: conflictedClaimId, conflictedArgumentId
- legacyAttackType, legacyTargetScope

Cognitive role: A first-class, extensible way to represent conflicts (rebuttals, undercuts, undermines, etc.) as scheme applications rather than hard-coded enums.

PreferenceScheme & PreferenceApplication

Purpose: Model preferences between information, inferences, or schemes.

Fields (PreferenceScheme):

- key, name, description, scope (info | inference | scheme | mixed)

Fields (PreferenceApplication):

- deliberationId, schemeId, createdById
- Exactly one preferred element: preferredClaimId, preferredArgumentId, preferredSchemeId
- Exactly one dispreferred element: dispreferredClaimId, dispreferredArgumentId, dispreferredSchemeId

Cognitive role: Encodes orderings (e.g., "trust source A over source B", "prefer modus ponens over abductive inference") explicitly in the graph.

DefaultRule

Purpose: Compact defeasible rule (α , β , γ) bound to an argument.

Fields: worldId, argumentId, role (premise | claim), antecedent, justification, consequent, meta

ArgumentPremise — premise join table

Model: ArgumentPremise

Purpose: Links Argument to multiple Claim premises.

Fields: argumentId, claimId, isImplicit, groupKey

PK: [argumentId, claimId]

ArgumentApproval — user endorsements

Model: ArgumentApproval

Unique: [argumentId, userId]

Cognitive role: Signals community support.

ArgumentSupport — strength of support for a claim

Model: ArgumentSupport

Purpose: Tracks how strongly an argument supports a claim ($\text{hom}(I, \phi)$ in formal AF terms).

Fields:

- deliberationId, claimId, argumentId
- mode (product | min | custom), strength (0..1), composed (if chained)
- rationale, base, provenanceJson (if imported)

Unique: [claimId, argumentId, mode]

Cognitive role: Quantifies argument strength; supports multi-criteria aggregation and imported line filtering.

AssumptionUse — track implicit assumptions

Model: AssumptionUse

Purpose: Tag assumptions (explicit or freeform text) that an argument relies on.

Fields:

- deliberationId, argumentId
- assumptionClaimId (nullable), assumptionText (nullable)
- role (premise | warrant | value | ...), weight, confidence, metaJson

Cognitive role: Makes hidden premises explicit and trackable.

4) Argument Internals (Diagram-Level Detail)

ArgumentDiagram — micro-graph of statements & inferences

Model: ArgumentDiagram

Purpose: A two-level drill-down: the **argument** is the narrative unit; the **diagram** is the logical structure.

Fields: title, statements, inferences, cqStatus (JSON), evidence, createdById

Relations:

- statements: Statement[]
- inferences: Inference[]
- evidence: EvidenceLink[]
- DebateNode (back-relation)

Cognitive role: Lets you target **warrants** (inferences), represent logical shape, attach evidence at fine granularity.

Statement — atomic text unit in a diagram

Model: Statement

Fields: diagramId, text, role (premise | intermediate | conclusion | assumption | question | warrant), lang, tags

Relations:

- conclusionFor: Inference[] (back-relation)
- InferencePremise[]

Inference — inferential step

Model: Inference

Purpose: An inference rule application inside a diagram.

Fields:

- diagramId, kind (presumptive | deductive | inductive | abductive | defeasible | analogy)
- conclusionId (exactly one conclusion Statement)
- schemeKey (optional), cqKeys (critical questions)

Relations:

- premises: InferencePremise[] (many-to-many join to Statement)

Cognitive role: The "warrant" node; can be targeted by undercuts via `ArgumentEdge.targetInferenceld`.

InferencePremise — join table

PK: [`inferenceld`, `statementId`]

5) Schemes & Critical Questions

ArgumentScheme — library of schemes

Model: `ArgumentScheme`

Purpose: Templates for argument patterns (expert opinion, analogy, consequence, etc.).

Fields:

- `key`, `name`, `description`, `title`, `summary`, `cq` (JSON)
- **Macagno taxonomy fields:** `purpose`, `source`, `materialRelation`, `reasoningType`, `ruleForm`, `conclusionType`, `slotHints` (JSON), `validators` (JSON)

Relations:

- `SchemeInstance[]`, `variants: SchemeVariant[]`, `cqs: CriticalQuestion[]`, `Argument[]`

SchemeVariant

Purpose: Variants of a base scheme (e.g., positive vs. negative consequence).

Fields: `schemeld`, `key`, `name`, `notes`

SchemeInstance — applied scheme

Model: `SchemeInstance`

Purpose: Apply a scheme to a target (`card` | `claim`) with slot-filling data.

Fields: `targetType`, `targetId`, `schemeld`, `data` (JSON), `createdById`

Relations: `CriticalQuestion[]`

CQStatus — critical question tracking

Model: `CQStatus`

Purpose: Per-target per-scheme CQ satisfaction.

Fields:

- `targetType` (`argument` | `claim` | `card` | `warrant` | `rebuttal`), `targetId`, `argumentId`
- `schemeKey`, `cqKey`, `satisfied`, `status` (`open` | `answered`)
- `roomId` (denormalized for RLS)

Unique: [`targetType`, `targetId`, `schemeKey`, `cqKey`]

Cognitive role: CQs make **attack surfaces** explicit and machine-trackable.

CriticalQuestion — persisted open question

Model: CriticalQuestion

Fields:

- instanceId, schemaId, cqKey, cqId, text
- attackKind (UNDERMINES | UNDERCUTS | REBUTS), status (open | addressed | counter-posted)
- openedById, resolvedById
- attackType, targetScope (for automation)

Unique: [schemaId, cqKey]

Cognitive role: Turn CQs into trackable "open issues" that can trigger undercuts.

6) Evidence as First-Class

EvidenceNode — distinct resource

Model: EvidenceNode

Fields: url, title, citation, kind, reliability, addedById, addedAt

Relations: links: EvidenceLink[]

EvidenceLink — attach evidence to targets

Model: EvidenceLink

Purpose: Link evidence to argument | claim | sheet with locators.

Fields:

- evidenceId, targetKind, targetId
- selectors (JSON: annotation anchors), note, uri, snapshotKey
- argumentDiagramId (nullable)

Cognitive role: Uniform provenance objects; citable, filterable, reusable.

7) Sheet-Level Mapping (Debate Visualization)

DebateSheet — curated or auto-generated map

Model: DebateSheet

Purpose: A navigable, zoomable debate map (two-level: nodes ↔ internal diagrams).

Fields:

- title, scope, roles, rulesetJson
- deliberationId (nullable), roomId (nullable)

Relations:

- nodes: DebateNode[], edges: DebateEdge[]
- loci: LocusStatus[], acceptance: SheetAcceptance, unresolved: UnresolvedCQ[], outcomes: Outcome[]

DebateNode — sheet node

Fields:

- sheetId, title, summary

- diagramId (1:1 link to ArgumentDiagram)
- argumentId (nullable), claimId (nullable), authorsJson

Cognitive role: The visual "card" in the debate map; can drill down to a full diagram.

DebateEdge — typed relation between nodes

Fields:

- sheetId, fromId, toId
- kind: DebateEdgeKind (supports | rebuts | objects | undercuts | refines | restates | clarifies | depends_on)
- thread, ord, rationale

Unique: [sheetId, fromId, toId, kind, thread] (idempotent)

LocusStatus, SheetAcceptance, UnresolvedCQ, Outcome

- **LocusStatus:** which sub-addresses are open/closable (for ludics)
- **SheetAcceptance:** labels for nodes under chosen semantics (grounded | preferred | hybrid)
- **UnresolvedCQ:** unresolved CQs per node
- **Outcome:** structured outcomes with summary

8) Clusters, Viewpoints, and Bridges

Cluster — topic/affinity grouping

Model: Cluster

Fields: deliberationId, type (affinity | topic), label, createdAt

Relations:

- users: UserCluster[] (userId, score)
- arguments: ArgumentCluster[] (argumentId, score)
- bridgeRequestsTargeting: BridgeRequest[]

Cognitive role: Surface "camps" or themes; enable viewpoint extraction.

ViewpointSelection — k-viewpoint extraction

Model: ViewpointSelection

Purpose: Snapshot of k representatives under a rule (utilitarian | harmonic | maxcov).

Fields:

- deliberationId, rule, k, coverageAvg, coverageMin, jrSatisfied, explainJson, createdById

Relations:

- viewpointArgs: ViewpointArgument[] (argumentId, viewpoint index 0..k-1)
- amplificationEvents: AmplificationEvent[]

Cognitive role: Pick representatives; power "viewpoint diff" UIs.

Bridge Builder

- **BridgeRequest:** targets a cluster (status: open | assigned | completed | expired)
- **BridgeAssignment:** assignee, accepted/completed timestamps, summaryCardId, rewardCard

Cognitive role: Incentivize syntheses that connect opposing clusters.

AmplificationEvent

Purpose: Track viewpoint creation, bridge assignments, etc.

Fields: deliberationId, hostType, hostId, eventType, reason, payload, createdBy, viewpointSelectionId

9) Governance, Moderation, and Audit

Enums

- ModerationStatus (OK | NEEDS_SOURCES | WORKSHOP | OFF_TOPIC_REDIRECT | DUPLICATE_MERGE | DISPUTED | OUT_OF_BOUNDS)
- ContentTargetType (article | post | room_thread | deliberation | argument | card | claim | brief | brief_version)
- PanelistRole (member | chair | observer)
- PanelDecisionType (APPROVE | WORKSHOP | REDIRECT)
- LogEntryType (STATUS_CHANGE | PANEL_OPEN | PANEL_CLOSE | PANEL_DECISION | POLICY_CHANGE | NOTE)

Models

- **ContentStatus:** latest moderation status per (roomId?, targetType, targetId) with currentStatus, prevStatus, reason, decidedById, panelId
- **DecisionReceipt:** append-only receipts for actions (status change, panel decision), linking to panels/policies
- **RoomLogbook:** narrative log entries (entryType, summary, payload)
- **Panel & Panelist:** review panels with membership and lifecycle

Cognitive role: Traceable decisions and status across polymorphic content; suitable for scholarly audit.

10) Briefs & Issues

Brief — publishable synthesis

Model: Brief

Fields: roomId, title, slug, status (draft | published), visibility, createdBy, currentVersionId

Relations:

- currentVersion: 1:1 to BriefVersion

- versions: 1:N to BriefVersion

BriefVersion — versioned snapshot

Fields: briefId, number, compiledFromDeliberationId, sectionsJson, citations, createdById

Relations:

- links: BriefLink[] (sourceType: card | argument | post | claim; sourceId)

Cognitive role: From **debate** → **publication**; cites source atoms.

Issue — tracked problem/question

Model: Issue

Fields: deliberationId, label, description, state (open | pending | closed), createdById, closedById, closedAt, kind (general | cq | moderation | evidence | structural | governance), key, assigneeId

Relations:

- links: IssueLink[] (targetType, targetId, role: related | blocks | depends_on | warrant | evidence)

Cognitive role: Collect and route technical debt or open problems; cross-reference with arguments/claims.

11) Dialogue Moves and Burden Modes

DialogueMove — formal turn-taking

Model: DialogueMove

Purpose: Dialogue-game events (ASSERT | WHY | GROUNDS | RETRACT | CONCEDE | CLOSE, etc.).

Fields:

- authorId, type/kind, illocution (Assert | Question | Argue | Concede | Retract | Close)
- deliberationId, targetType, targetId, payload, actorId
- replyToMoveId, replyTarget (claim | argument | premise | link | presupposition)
- polarity (P | O), locusId, endsWithDaimon, argumentId, signature (unique per deliberation)

Unique: [deliberationId, signature] (prevents duplicate moves)

Cognitive role: Formal turn-taking & legality checking for "ask-why / give-grounds / concede / close" workflows.

ProofMode on Deliberation

Enum: symmetric | asymmetric

Purpose: Burden differences (stricter for Proponent in asymmetric mode).

12) Ludics (Deep Dialogical Traces)

Purpose: Proof-style interaction substrate for precise argumentative gameplay and commitment tracking.

Core Models

- **LudicLocus:** address tree (dialogueId, path, parentId); unique per [dialogueId, path]
- **LudicDesign:** participant's strategy tree (deliberationId, participantId, rootLocusId, semantics, hasDaimon, version)
- **LudicAct:** acts with polarity (designId, kind: PROPER | DAIMON, polarity: P | O | DAIMON, locusId, ramification, expression, orderInDesign)
- **LudicChronicle:** chronological record (designId, order, actId)
- **LudicTrace:** interaction trace between Proponent and Opponent designs (posDesignId, negDesignId, startLocusId, steps, status: ONGOING | CONVERGENT | DIVERGENT)
- **LudicCommitmentElement:** commitment store (ownerId, basePolarity, baseLocusId, label, entitled)
- **LudicCommitmentState:** snapshot of commitments (ownerId, elements, extJson)
- **LudicFaxMap:** address mappings (fromLocusId, toLocusId)
- **LudicBehaviour:** incarnation sets (deliberationId, base, polarity, regular, uniformBound)
- **LudicMaterialDesign:** membership (behaviourId, designId, asOfStepId, pathsJson)
- **LudicDecisionReceipt:** outcomes (deliberationId, kind: epistemic | procedural | allocative | editorial, subjectType, subjectId, issuedBy, rationale, inputsJson, version)

Cognitive role: Proof-style interaction; commitment tracking; enables rigorous "proof obligations" for claims.

13) Voting

VoteSession — approval or RCV voting

Fields: deliberationId, subjectType (option | view | claim), subjectId, method (approval | rcv), optionsJson, quorumMinCount, quorumMinPct, closesAt, closedAt, tallyJson, winnerId

Relations: ballots: VoteBallot[] (unique per [sessionId, voterId])

Cognitive role: Decision procedures layered over debates and viewpoints.

14) Works (Theory Frames) & Knowledge Links

TheoryWork — longform philosophical/scientific works

Model: TheoryWork

Purpose: Structured theory development (DN/IH/TC/OP frameworks).

Fields:

- slug, title, authorId, theoryType (DN | IH | TC | OP), summary, body, standardOutput
- deliberationId (nullable: provenance, not container), kbPageId (KB anchoring)
- integrityChecks (JSON), integrityValid, status (DRAFT | ACTIVE | PUBLISHED | ARCHIVED), visibility, publishedAt, lastExport

Relations:

- claims: TheoryWorkClaim[] (role: thesis | premise | exception | application | counter)
- ihProject, dnProject, tcProject, opProject: 1:1 to project structures
- dnStructure, ihTheses, tcTheses, opTheses: 1:1 to thesis structures
- provenances: WorkProvenance[], practicalJustification, hermeneuticProject, pascalModel
- rules: Rule[], citations: TheoryWorkCitation[]

Project Structures

- **WorkDnProject:** definitions, axioms, theorems, confirming, optimality (GDN scoring)
- **WorkIhProject:** practice description, ideal standard, subjective/objective reasons, corroboration, applications
- **WorkTcProject:** structure/function description, explanation, applications
- **WorkOpProject:** unrecognizability justification, alternatives

Thesis Structures

- **WorkDNStructure:** explanandum, nomological, ceterisParibus
- **WorkIHTheses:** structure, function, objectivity
- **WorkTCTheses:** instrumentFunction, explanation, applications
- **WorkOPTheses:** unrecognizability, alternatives

Practical Justification (MCDA)

WorkPracticalJustification: purpose, criteria, options, scores, result, adequacy (completeness, dominance, robustness)

Hermeneutic & Pascal Models

- **WorkHermeneuticProject:** corpusUrl, facts, hypotheses, plausibility, selectedIds
- **WorkPascalModel:** propositions (worlds), actions, utilities, assumption, exhaustiveExclusive, noTheoreticalEvidence, orientationCriteria, decision

Citations Bridge

TheoryWorkCitation: workId, targetType (claim | argument | proposition | work),

targetId, section, role (premise | evidence | example | counterpoint | reference | thesis), note, kbPageId

Additional Source Deliberations

WorkSourceDeliberation: workId, deliberationId, role (source | context)

Knowledge Graph Edges

- **KnowledgeEdge:** deliberationId, kind (SUPPLIES_PREMISE | REVISES | CHALLENGES | SUPPORTS | REBUTS | UNDERCUTS | UNDERMINES | ALTERNATIVE_TO | EVALUATES | REASON_FOR | REASON_AGAINST | NOT_REASON_FOR | NOT_REASON_AGAINST | CONCLUSIVE_ABOUT | INCONCLUSIVE_ABOUT | ENTAILS), fromWorkId, toWorkId, fromClaimId, toClaimId, meta
- **NLIlink:** fromId, toId, relation, score, createdById (external NLI relationships)
- **EristicMark:** deliberationId, targetType, targetId, tactic, detector, strength (detected rhetorical/eristic tactics)
- **Rule:** workId, kind (STRICT | DEFEASIBLE), head, body, meta
- **ReasonPair:** deliberationId, claimId, reasonId, stance (FOR | AGAINST | NOT_FOR | NOT_AGAINST), strong (LA strong link), meta

Cognitive role: Explicit **theory scaffolds** and **knowledge edges**; debates connect to scholarship and methods.

15) Social Discussion Substrate & Upgrades

Discussion — forum thread with upgrade path

Model: Discussion

Purpose: Light-weight discourse that can **promote** into structured deliberations.

Fields: slug, title, description, createdById, visibility, conversationId (link to existing chat), attachedToType/Id (polymorphic), replyCount, viewCount, lastActiveAt

Relations:

- conversation: 1:1 with chat Conversation
- upgradedToDeliberation: nullable back-link to Deliberation
- messages: DiscussionMessage[], participants: DiscussionParticipant[]
- subscriptions: DiscussionSubscription[], forumComments: ForumComment[], deliberations: DiscussionDeliberation[]

DiscussionMessage — threaded messages

Fields: discussionId, parentId, authorId, kind (text | media | system | poll | note), text, richJson, isPinned, isEphemeral, expiresAt

Relations: replies, reactions: DiscussionReaction[]

ForumComment — promoted comments

Purpose: Cross-post from chat → forum.

Fields: discussionId, parentId, authorId, body (tiptap JSON), bodyText, sourceMessageId, sourceConversationId, score, isDeleted, isRemoved

Unique: [discussionId, sourceMessageId] (prevents duplicate promotion)

DiscussionDeliberation — upgrade bridge

Unique: [discussionId, deliberationId]

ForumVote & ForumSave

- ForumVote: userId, commentId, dir (-1/0/1)
- ForumSave: userId, commentId (bookmarks)

16) Inter-Room / Cross-Debate Graph

AgoraRoom — organizational container

Fields: id, slug, title, summary, visibility, createdAt, updatedAt

Relations: sheets: DebateSheet[], deliberations: Deliberation[]

StackReference — when a stack references another room

Fields: fromDeliberationId, toDeliberationId, stackId, relation (attached | cites | embeds | ...)

Unique: [fromDeliberationId, toDeliberationId, stackId, relation]

ArgumentImport — import/restatement edges

Fields: fromDeliberationId, toDeliberationId, fromArgumentId, toArgumentId, kind (import | restatement | quote | ...), fromClaimId, toClaimId, baseAtImport, fingerprint (unique SHA1), metaJson

Unique: [fromArgumentId, toArgumentId, kind]

Cognitive role: The **graph-of-graphs**; work travels and composes across rooms while preserving provenance.

SharedAuthorRoomEdge — weak ties

Fields: fromId, told, strength (count or weighted score)

Unique: [fromId, told] (canonicalize fromId < told in app code)

RoomFunctor — claimed mapping

Fields: fromRoomId, toRoomId, claimMapJson (JSON), notes, createdById

Unique: [fromRoomId, toRoomId]

AgoraFollow — user following

PK: [userId, kind, targetId] (kind: room | tag)

AgoraOutbox — event outbox

Fields: ts, topic, roomId, deliberationId, targetType, targetId, payload, delivered

XRef — cross-references

Purpose: Polymorphic link (fromType, fromId, toType, toId, relation: cites | evidence-for | discusses | originates-from | replies-to | cross-claim | ...)

Unique: [fromType, fromId, toType, toId, relation]

17) Knowledge Base & Wiki

KB (Curated Pages with Live Transclusions)

- **KbSpace:** slug, title, summary, visibility (public | org | followers | private), kind (personal | team | org | project), createdById
- **KbSpaceMember:** spaceId, userId, role (owner | editor | commenter | reader)
- **KbPage:** spaceId, slug, title, summary, visibility, tags, frontmatter (JSON), createdById, updatedById
- **KbBlock:** pageId, ord, type (text | image | link | claim | claim_set | argument | sheet | room_summary | transport | evidence_list | cq_tracker | plexus_tile | **theory_work** | theory_section), live (vs pinned), dataJson, pinnedJson, citations, createdById
- **KbSnapshot:** pageId, label, atTime, createdById, manifest (frozen block manifest)

Cognitive role: Narrative, citable **knowledge pages** that embed live debate artifacts or pinned snapshots.

Wiki

- **WikiPage:** slug, title, entityUrn, createdById, currentId (1:1 to current WikiRevision)
- **WikiRevision:** pageId, body, citations, createdById; back-relations: page (1:N), currentOf (1:1)

18) Brief Governance & Economy

- **Bounty:** roomId, type (synthesis | source | replication | bridge), title, brief, rewardCare, status, opensAt, closesAt
- **BountySubmission:** bountyId, submitterId, submissionType, submissionId, status
- **CareLedger:** userId, delta, reason, refType, refId
- **AudiencePreference:** ownerType, ownerId, order (e.g., "Fairness>Autonomy>Efficiency")
- **Value & ClaimValue:** annotate claims with values and weights (normative

axis)

- **Commitment:** deliberationId, participantId, proposition, isRetracted (per-participant commitments)

19) GitChat: Proposal Signals (Approve/Block)

ProposalSignal

Purpose: Lightweight approve/block signals on proposal facets.

Fields: conversation_id, message_id, facet_id, user_id, kind (APPROVE | BLOCK), created_at

Unique: [facet_id, user_id]

AgreementLock & AgreementAck

- **AgreementLock:** message_id (unique), locked_by, locked_at
- **AgreementAck:** PK [message_id, user_id], ack_at

20) Enums (Quick Glossary)

Argument/Claim Relations

- EdgeType: support | rebut | undercut | concede | CA
- ArgumentAttackSubtype: SUPPORT_ATTACK | CONSEQUENCE_ATTACK | JUSTIFICATION_ATTACK | UNDERMINE | REBUT | UNDERCUT | OVERCUT
- ClaimEdgeType: supports | rebuts
- ClaimAttackType: SUPPORTS | REBUTS | UNDERCUTS | UNDERMINES
- TargetScope: conclusion | premise | inference
- AttackType: REBUTS | UNDERCUTS | UNDERMINES

Logic & Modality

- StatementRole: premise | intermediate | conclusion | assumption | question | warrant
- InferenceKind: presumptive | deductive | inductive | abductive | defeasible | analogy
- Quantifier: SOME | MANY | MOST | ALL
- Modality: COULD | LIKELY | NECESSARY

Media & Visibility

- MediaType: text | image | video | audio
- Visibility: public | unlisted | room_only
- KbVisibility: public | org | followers | private

Representation & Selection

- RepresentationRule: utilitarian | harmonic | maxcov

Delib Hosts

- DeliberationHostType: article | post | room_thread | library_stack | site | inbox_thread | work

Proposition & Brief

- PropositionStatus: DRAFT | PUBLISHED | CLAIMED | ARCHIVED
- BriefStatus: draft | published
- BriefSourceType: card | argument | post | claim

Issue

- IssueState: open | pending | closed
- IssueLinkTargetType: argument | claim | card | inference
- IssueLinkRole: related | blocks | depends_on | warrant | evidence
- IssueKind: general | cq | moderation | evidence | structural | governance

KB

- KbRole: owner | editor | commenter | reader
- KbSpaceKind: personal | team | org | project
- KbBlockType: text | image | link | claim | claim_set | argument | sheet | room_summary | transport | evidence_list | cq_tracker | plexus_tile | theory_work | theory_section

Ludics

- LudicPolarity: P | O | DAIMON
- LudicActKind: PROPER | DAIMON
- LudicTraceStatus: ONGOING | CONVERGENT | DIVERGENT

Dialogue

- Illocution: Assert | Question | Argue | Concede | Retract | Close
- ReplyTarget: claim | argument | premise | link | presupposition
- ProofMode: symmetric | asymmetric

Knowledge Edges

- KnowledgeEdgeKind: SUPPLIES_PREMISE | REVISES | CHALLENGES | SUPPORTS | REBUTS | UNDERCUTS | UNDERMINES | ALTERNATIVE_TO | EVALUATES | REASON_FOR | REASON_AGAINST | NOT_REASON_FOR | NOT_REASON_AGAINST | CONCLUSIVE_ABOUT | INCONCLUSIVE_ABOUT | ENTAILS

Theory Works

- PhilosophyTheoryType: DN | IH | TC | OP
- WorkStatus: DRAFT | ACTIVE | PUBLISHED | ARCHIVED

DebateSheet

- DebateEdgeKind: supports | rebuts | objects | undercuts | refines | restates | clarifies | depends_on

Bridge & Cluster

- ClusterType: topic | affinity
- BridgeRequestStatus: open | assigned | completed | expired

Evidence

- EvidenceKind: primary | secondary | dataset | code
- UrnEntityType: claim | card | brief_version
- GroundLabel: IN | OUT | UNDEC
- TargetType: argument | claim | card | warrant | rebuttal

21) Typical Flows (Cognitive & Data Paths)

1. **Start a debate** → create Deliberation; add DeliberationAnchors, set proofMode
2. **Workshop ideas** → users create Propositions; vote, endorse, reply, tag
3. **Promote to claims** → convert Proposition to Claim (sets promotedClaimId, status CLAIMED)
4. **Add claims** (with citations/evidence) → Claim, ClaimCitation/ClaimEvidence
5. **Write arguments** → Argument with ArgumentSupport to Claim; compose ArgumentDiagram with Statement + Inference
6. **Link arguments** → ArgumentEdge with type=undercut and set targetInferenceld (fine-grain warrant attacks)
7. **Apply conflict/preference schemes** → ConflictApplication, PreferenceApplication (more expressive than hard-coded enums)
8. **Link claims** → ClaimEdge (supports/rebuts); run labeling → ClaimLabel (IN/OUT/UNDEC)
9. **Apply schemes & answer CQs** → SchemeInstance, CQStatus, CriticalQuestion; address CQs with WHY/GROUNDS DialogueMoves
10. **Map the debate** → auto/saved DebateSheet with DebateNode/Edge, SheetAcceptance, unresolved CQs
11. **Cluster viewpoints** → Cluster, ViewpointSelection & ViewpointArgument; launch BridgeRequest to connect clusters
12. **Moderate** → status via ContentStatus; panels via Panel/DecisionReceipt

13. **Develop theory** → create TheoryWork (DN/IH/TC/OP); fill project/thesis structures; link claims via TheoryWorkCitation
14. **Publish** → compile to Brief/BriefVersion with BriefLinks
15. **Decide** → optionally run VoteSession (approval/RCV)
16. **Propagate** → cross-room StackReference, ArgumentImport, or define RoomFunctor mappings
17. **Curate in KB** → KbPage with KbBlocks, live or pinned; snapshot with KbSnapshot
18. **Ludics proof game** → participants create LudicDesigns with LudicActs; system generates LudicTraces; track commitments via LudicCommitmentState

22) Integrity & Idempotency Highlights

Unique Constraints (Prevent Duplicates)

- ArgumentApproval: @@unique([argumentId, userId])
- ClaimEdge: @@unique([fromClaimId, toClaimId, type, attackType])
- DebateEdge: @@unique([sheetId, fromId, toId, kind, thread])
- DialogueMove: @@unique([deliberationId, signature])
- ArgumentSupport: @@unique([claimId, argumentId, mode])
- DiscussionDeliberation: @@unique([discussionId, deliberationId])
- ArgumentImport.fingerprint: @unique (and composite unique across from/to/kind)
- ProposalSignal: @@unique([facet_id, user_id])
- LudicLocus: @@unique([dialogueId, path])

On-Delete Policies

- Core graphs: **Cascade**
- Cross-references: often **SetNull** (preserve provenance without breaking graphs)

Time-Sorted Indexes

Most content models have @@index([...createdAt]) for efficient feeds and audits.

23) Where to Be Especially Mindful (Design Notes)

- **Undercuts**: prefer targeting **inferences** (targetInferenceId) to keep rebut vs undercut analytically clean
- **Conflict/Preference schemes**: use ConflictApplication and PreferenceApplication for extensible, first-class representation instead of hard-coded enums
- **Labeling semantics**: ClaimLabel.semantics allows multiple semantics; don't assume grounded only

- **Imports & provenance:** always record `ArgumentSupport.provenanceJson` and `ArgumentImport.metaJson` so rooms can include/exclude imported lines
- **CQ coverage:** treat `CQStatus` as ground truth for "open questions" dashboards; ensure `DebateSheet.unresolved` mirrors it
- **KB blocks:** use **pinned** snapshots for briefs/journal-grade outputs; **live** for internal wikis
- **Values:** `Value/ClaimValue` let you project debates into a value space (for value-aware views or audience preferences)
- **Ludics scoping:** `LudicLocus` now scoped by `dialogueId` to avoid global address clashes
- **Proposition → Claim:** use `Proposition` as workshop space; promote to `Claim` when ready (sets `promotedClaimId`, status `CLAIMED`)
- **Theory Works:** decoupled from deliberation (now optional provenance); can cite across claims/arguments/propositions/works via `TheoryWorkCitation`

24) Quick Cross-Reference (What to Use When)

- **Rigorous, navigable map** → `DebateSheet` (+ nodes/edges; expand nodes to `ArgumentDiagram`)
- **Which claims stand** → `ClaimLabel` with chosen semantics; surface `ClaimStats`
- **Trace how/why** → `Argument` + `ArgumentDiagram` + `EvidenceLinks`; use `AssumptionUse` to expose hidden premises
- **Organize camps** → `Cluster`, `ViewpointSelection`; use `BridgeRequest` to spur syntheses
- **Publishing** → `Brief/BriefVersion`; cite sources via `BriefLink`
- **Formal moves/audit** → `DialogueMove`, `DecisionReceipt`, `RoomLogbook`, `Panel`
- **Proof-style play** → `Ludics` (`Ludic*`) with traces and commitments
- **Share across rooms** → `StackReference`, `ArgumentImport`, `RoomFunctor`, `XRef`
- **Develop theory** → `TheoryWork` with `DN/IH/TC/OP` structures; cite via `TheoryWorkCitation`
- **Workshop ideas** → `Proposition` (vote, endorse, reply) → promote to `Claim`
- **Conflict & preference** → `ConflictApplication`, `PreferenceApplication` (extensible, first-class)

25) Major Changes from Previous Schema

1. **Propositions introduced:** New social layer before claims (`Proposition` →

Claim promotion path)

2. **Conflict & Preference Schemes:** ConflictScheme, ConflictApplication, PreferenceScheme, PreferenceApplication (extensible conflict/preference representation)
3. **DefaultRule:** Compact defeasible rule model
4. **ArgumentPremise.groupKey:** Support for grouped premises
5. **TheoryWork decoupled:** deliberationId now optional (provenance, not container)
6. **TheoryWorkCitation:** Polymorphic citation bridge (cite claims, arguments, propositions, works)
7. **WorkSourceDeliberation:** Track additional source deliberations beyond home one
8. **KB blocks extended:** New types theory_work, theory_section
9. **LudicLocus scoping:** Now scoped by dialogueId to avoid global clashes
10. **GraphEdge:** Simple edge model for /api/attacks route (optional)
11. **XRef:** Polymorphic cross-reference model (cites | evidence-for | discusses | originates-from | replies-to | cross-claim)
12. **GitChat models:** ProposalSignal, AgreementLock, AgreementAck
13. **Issue.kind:** New field (general | cq | moderation | evidence | structural | governance)
14. **ReplyTarget enum:** For DialogueMove.replyTarget
15. **KbPage.updatedById:** Track who last updated a page

Final Note

This map is intentionally **product-oriented**: it explains what each object means to the user and how it composes cognitively. The schema supports a rich, multi-layered deliberation platform with:

- **Social → Structured** path (Propositions → Claims)
- **Extensible conflict/preference** modeling
- **Fine-grain targeting** (inferences, CQs)
- **Theory development** (DN/IH/TC/OP frameworks)
- **Cross-room provenance**
- **Formal dialogue & ludics**
- **KB integration**
- **Governance & audit trails**

This is the **canonical snapshot** of how discourse objects hang together.