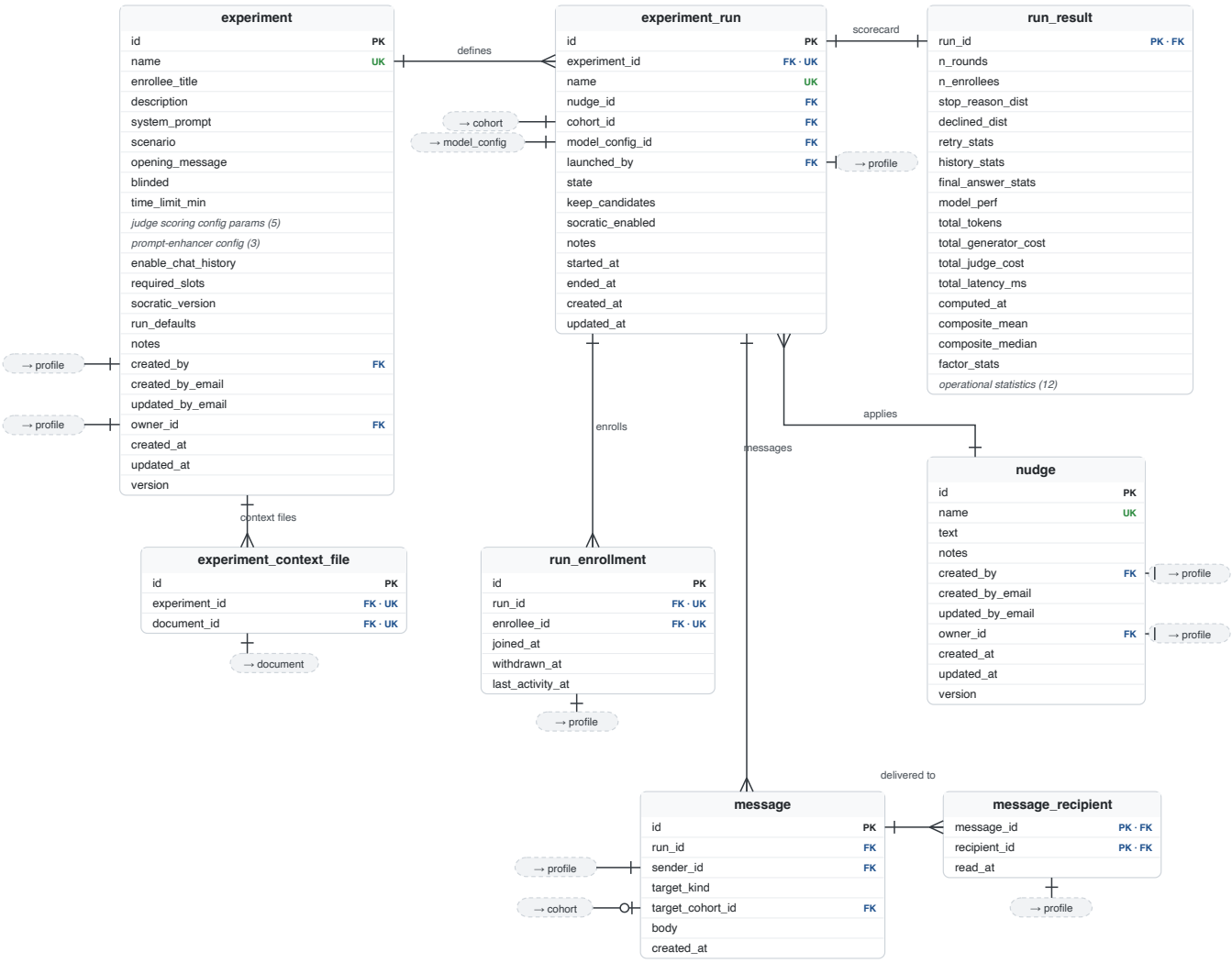
ChatMaestro — Data Model: `core-experiment`

Revision: 2026-09-20

This module comprises the reusable experiment, each run of it, who is enrolled, the run's context files, and the operator↔enrollee messaging.

Entity-Relationship Diagram



Narrative

What This Module Does

This module is the spine of the platform. An experiment is the reusable study. It is a fixed frame — the topic, the prompts, the context files, the enrollee interface, and the scoring configuration — and it is the comparison container its runs sit in. Each run is one cell of that comparison: a combination of a nudge, a model configuration, and a cohort. A nudge is an enrollee-safe steering message, and a cohort is a named group of enrollees. A run enrolls enrollees and carries the out-of-band operator↔enrollee messaging. The definitions a run points at — the experiment, the nudge, the model configuration, the cohort, and the Socratic switch — are held immutable while the run references them, so the captured data stays faithful without copying anything into a snapshot.

experiment — The Invariant Frame + Comparison Container

This holds everything constant across the run series, while the per-run variables — the nudge, the model configuration, and the cohort — live on the run. Every column is described below. The diagram shows the five `score_*` analytics-config columns as one grouped row, and the three prompt-enhancer columns as another; both are broken out here:

Column	Meaning · values · when populated
`id`	The surrogate primary key for the row, marked **PK**.
`name`	The experiment name, marked **UK**, shown to staff. Enrollees see `enrollee_title` instead, so the name can name the study's intent without unblinding them.
`enrollee_title`	The enrollee-safe short title shown to enrollees, for example "Open-topic chat," kept distinct from `name` so it never reveals the study's intent. The app prefills a head-started title from the experiment and scenario at creation, and the composer can edit it; when blank, the interface falls back to a generic title.
`description`	A free-text description of the study, shown to staff only. Like `name`, it can reveal the study's intent, so it is hidden from enrollees, who see the enrollee-safe `enrollee_title`, `scenario`, and `opening_message` instead.
`system_prompt`	The model's standing instruction, hidden from the enrollee. It is the base of a two-layer system-prompt cascade: within a run, each model in the configuration may override it. The override says, through a mode, whether to keep, replace, or append. This base carries no mode of its own, because nothing sits above it. The fully resolved prompt is what the model receives. It is not copied anywhere: because the experiment and the model configuration are held immutable while a run references them, re-resolving the prompt later reproduces exactly what the run used.
`scenario`	The enrollee-safe framing, which is the session topic.
`opening_message`	This is the enrollee-safe first assistant message that opens the chat.
`blinded`	The blinding gate for the enrollee interface, a boolean defaulting to `true` so it fails safe. When true, the enrollee sees only the single chosen response: the "See all" viewer of every model and iteration is hidden, and file upload and download are disabled so nothing about the condition can leave or enter the session. When false, those are enabled. Chat conveniences unrelated to blinding (regenerate, clear-session, example prompts) are always available and are not configured here.
`time_limit_min`	An optional per-session time limit for enrollees, in minutes. When set, the countdown is always shown to the enrollee. Operator↔enrollee messaging is always available, so it carries no per-experiment switch.

Column	Meaning · values · when populated
judge scoring config params — the 5 score_* columns	The configuration for the experiment's one trusted judge — the single model that does **all** scoring, grading each candidate to pick a winner and drive the self-improvement loop during a round and producing the per-round scores after. It is seeded from an install-level global default when the experiment is composed and can be overridden here. Because the experiment is held immutable while any run references it, the scoring a run used cannot be changed after launch, so a later edit cannot silently re-score past runs. `score_judge_catalog_id` is the foreign key to the curated `model_catalog` entry, so identity lives in one place. The provider and model literals beside it are the durable historical record of what actually scored the rounds, and they survive catalog retirement the way `audit_log.actor_label` survives a deleted profile. The columns are: • `score_judge_catalog_id`, `score_judge_provider`, and `score_judge_model` — naming which judge model runs the uniform scoring pass. • `score_rubric_version` — naming which rubric it uses. The rubric is a versioned, immutable artifact in the middle tier that defines **which** factors are scored, their definitions, and their anchored scales. The middle tier's registry bundles every available rubric and publishes their list, so the experimenter picks one from a drop-down rather than typing a version. The built-in default is a four-factor answer-quality rubric (groundedness, relevance, coherence, instruction-following), but an experiment may pick a different rubric whose criteria suit its study — say funniness, originality, and timeliness for a joke experiment. • `score_weights` — the per-factor weights, keyed by the chosen rubric's factors, where null means equal. Together they populate `chat_round`'s scores, applied identically to every run so the experiment's runs are comparable within it.
prompt-enhancer config — enable_prompt_enhancer · prompt_enhancer_version · human_review_enabled	Whether, and how, the middle tier rewrites the enrollee's raw prompt into a better-engineered one before the model set-up answers (see `ALGORITHMS §18`). `enable_prompt_enhancer` is a boolean defaulting to `false`, so the enrollee's exact words are used unless enhancement is opted into; `prompt_enhancer_version` names a versioned enhancer-instructions artifact in the middle-tier registry, picked from a drop-down exactly as the rubric is. Like the scoring config, both are frozen once the experiment has runs, so the enhancement policy is uniform across a comparison and never confounds the swept variable. The rewrite runs on the experiment's judge model (`score_judge_*`) and stores both the original and the enhanced text on `chat_round`. `human_review_enabled` reserves the later approve-or-edit step; it is always `false` in this version and shown read-only, because a per-enrollee hand-edit is not a uniform condition and so belongs to ad-hoc use rather than to a controlled study.
enable_chat_history	Whether the enrollee's earlier exchanges are replayed to the models on later rounds — whether the assistant remembers the conversation. It defaults to `true` and is frozen once the experiment has runs, like the scoring and enhancer configuration, so memory is a uniform condition rather than something that varies by enrollee. When `false`, every round is self-contained: the models receive the system prompt, the retrieved context and this prompt alone, which removes conversation length as a variable and is the honest setting for comparing single answers. There is deliberately no per-experiment turn count beside this switch — a second limit invites confusion about which one is in force, and an experiment-level depth would let memory vary with the model configuration, which is itself a swept variable. Depth is a platform constant bounded by the configuration's smallest `model_catalog.context_window`, so it is identical across every arm (see `ALGORITHMS §23`).
socratic_version	The versioned instructions the trusted judge composes the sufficiency gate's replies under, and null only on an experiment whose gate can never fire. It names an artifact in the middle-tier registry, chosen from a drop-down as the rubric and enhancer versions are, and frozen once the experiment has runs. One artifact holds all four phrasings the gate can need — asking for a missing piece, or redirecting a request the material does not cover, each worded once as a question and once as a demand to rewrite — because the questioning and rewrite wordings are the two arms of the comparison, and freezing one while the other could change between releases would move the control arm without recording it. A version is required whenever the experiment attaches context documents or declares `required_slots`, and a run cannot set `socratic_enabled` without one (see `ALGORITHMS §21`, `§30`).

Column	Meaning · values · when populated
`required_slots`	The list of things a request must supply before this experiment will answer it — the second half of the sufficiency gate (`ALGORITHMS §21`). Each entry declares a slot in prose, written for a reader rather than as a grammar, because whether a message fills it is decided by the trusted judge reading the declaration rather than by a parser. When an enrollee's message leaves one out, the gate replies naming what is missing and no `chat_round` is written; answers accumulate across attempts, so somebody supplying the pieces one at a time is never made to restate the whole request. An empty or absent list disables the test, which is the default, so an experiment that wants no gate does not get one. Like the scoring and enhancer config it is frozen once the experiment has runs, so every enrollee meets the same standard. No screen writes it in this version — it is configured directly against the database until the authoring screen arrives.
`run_defaults`	A jsonb seed bag that pre-fills the launch form for a run's per-run settings. It is not authoritative, because the run stores its own typed, foreign-key-enforced, launch-frozen values. The keys are `model_config_id` and `model_config_name` for the default model configuration, where the name is kept alongside for readable display without a join; `nudge_id` for a baseline; and `keep_candidates`. A stale default, such as a deleted config, is simply ignored at launch, because it was never a foreign key. The cohort is deliberately not defaulted, because concurrent runs must target disjoint cohorts.
`notes`	Free-text notes a user writes about this experiment. They are searchable by SQL keyword, and in the semantic index by meaning.
`created_by`	A foreign key to `profile` naming the composer who first authored it. It never changes.
`owner_id`	A foreign key to `profile` naming the current owner, which defaults to the composer. The owner edits the experiment, and peers read it and launch their own runs from it. The admin can transfer ownership, and only while the experiment has no running or paused runs. The value goes null if the owner is deleted, which leaves the experiment admin-managed.
`created_at / updated_at`	These record when the row was created and when it was last updated.
`version`	A counter that supports optimistic concurrency control (OCC), the check that stops two users from overwriting each other's edits. The counter increases on every save. A save built on a stale copy is rejected, and the edit screen shows the message "record changed — reload."

experiment_run — One Cell of the Comparison Grid

A run is identified by the tuple (`experiment_id`, `nudge_id`, `model_config_id`, `cohort_id`) — its container plus its three independent variables. The row is append-frozen: once launched, only `state` changes. It keeps no snapshot of its condition, because the definitions it points at are held immutable while it references them, so analytics resolve every label and setting by joining straight to them.

Column	Meaning · values · when populated
`id`	The surrogate primary key for the row, marked **PK**.
`experiment_id`	A foreign key to the parent `experiment` — the comparison container that fixes the prompts, context files, enrollee interface, and scoring configuration. It is `ON DELETE RESTRICT`: the experiment cannot be hard-deleted while any of its runs exist.
`nudge_id`	A foreign key to the `nudge` applied in this run — the first of the three per-run independent variables, and an enrollee-safe steering message. It is `ON DELETE RESTRICT`, and the nudge is held immutable while any run references it, so a finished run always shows the exact steering message it used, with no per-run copy to keep. It is held constant across a model-configuration comparison, and varied across a nudge comparison.
`cohort_id`	A foreign key to the `cohort` this run targets — the second independent variable, and a named group of enrollees. It is `ON DELETE RESTRICT`, and the cohort and its membership are held immutable while referenced, so the group at run time is preserved without a copy; the actual enrollees are pinned in `run_enrollment`.
`model_config_id`	A foreign key to the one `model_config` applied uniformly to every enrollee — the third independent variable. It is `ON DELETE RESTRICT`, and the configuration and its constituent models are held immutable while referenced, so the run always resolves to the exact models, temperatures, and per-model prompts it ran with. A comparison holds two of these three variables constant and varies the third.

Column	Meaning · values · when populated
`name`	A human label for the run, unique within the experiment (marked **UK** together with `experiment_id`). If it is left blank at launch, a default is created and frozen, combining the experiment, cohort, model configuration, and datetime.
`launched_by`	A foreign key to `profile` naming who launched this run, which can differ from who composed the experiment. It drives run-purge authorization: an experimenter may purge runs they composed (`experiment.created_by`) or launched, and only when the run is not in progress. It is `ON DELETE SET NULL`, so the run survives the launcher's deletion.
`state`	The run's lifecycle state: `running`, `paused`, `done`, or `aborted`. Only `done` or `aborted` runs may be purged, and a paused run still counts as live for the non-overlapping-cohort rule. Reaching `done` is what triggers the `run_result` scorecard to be built.
`socratic_enabled`	How the sufficiency gate replies when it stops an enrollee's message: with the switch on the trusted judge asks for one missing piece at a time and the request is built through dialogue, and with it off the judge names everything missing and asks for the whole request to be rewritten and resent. It is pre-filled from `experiment.run_defaults` at launch, which makes it the **fourth per-run variable** beside the nudge, the model configuration and the cohort. The switch changes the reply and not the detection, so the same missing piece stops the same message either way; an experiment declaring no `required_slots` is the exception, where the judge falls back to reading the request itself and does so only when this switch is on. Both phrasings live in `experiment.socratic_version`, and every stopped attempt counts in `chat_round.retry_count` under either setting (see `ALGORITHMS §30`).
`keep_candidates`	Whether to retain this run's `chat_round_candidate` rows, the per-model and per-iteration detail behind each round, after the round is finalized. It is a boolean: `true` keeps them for a later side-by-side; `false` (the default) clears them as each round completes, which is leanest for high-volume runs and still lets the "See all" viewer work live during the round. It is pre-filled from `experiment.run_defaults` at launch and can be overridden here, and candidate rows are deleted with their round or run regardless of this setting.
`notes`	Free-text notes a user writes about this run. They are searchable by SQL keyword, and in the semantic index by meaning. They are staff-only and frozen at run end.
`started_at · ended_at`	The run window: when execution began, and when it reached `done` or `aborted`.
`created_at / updated_at`	When the run row was composed and last edited. The composition time may precede `started_at` if the run is configured as a draft before launch.

run_result — The Per-Run Scorecard (One Flat Row per Run)

The run's durable scorecard, one-to-one with `experiment_run` and computed once the run reaches `done`. It is what the comparison queries read. It survives a **Cleanup** (which reclaims the heavy transcripts but keeps this row); it is removed only when the finished run is **Purged**, which cascades it away, so it never dangles. Its scores are the run's `chat_round` rows — the experiment's one trusted judge's scores — aggregated.

Column	Meaning
`run_id`	The **PK · FK** to `experiment_run`, one-to-one, with `ON DELETE CASCADE`. Since the scorecard is built only at `done`, it never exists for a running or paused run, and the cascade only fires on a Delete of a finished run.
`n_rounds · n_enrollees`	The sample size: the scored rounds aggregated, and the number of distinct contributing enrollees.
`stop_reason_dist`	A jsonb tally of why loops stopped, over {`score_target, min_gain, max_iterations, single_pass`}.
`final_answer_stats`	The score distribution over each session's **last round** rather than over every round, as jsonb, and null when no session contributed one — a run with no rounds, or one whose every round was rewound away. It carries the same five-number summary and composite mean and median that `factor_stats` and `composite_mean` hold, computed on that one round per session. It exists because the last answer is what the enrollee walked away with, and a long session can drift a long way from where it started. Read it beside `composite_mean`, never instead of it: the all-rounds figure still says what the whole session cost and scored, and this one says where it ended up (see `ALGORITHMS §6`).

Column	Meaning
`history_stats`	What conversation memory did across this run, as jsonb, and null when `enable_chat_history` is off. It records `context_window_min` (the smallest window among the configuration's members, which bounded every round), the rounds truncated, the rounds where the enrollee was alerted, the rewinds, and the mean and maximum turns actually replayed. It is the roll-up that makes a between-arm difference visible: `model_config` is swept, so one arm can have a smaller smallest-window than another and therefore systematically shallower memory. Read alone that looks like a quality difference; read beside these figures it is a condition difference the experimenter can account for. Like the rest of this row it outlives a Cleanup of the rounds it came from (see `ALGORITHMS §23`).
`declined_dist`	A jsonb tally of how the run's rounds answered the request, over `{appropriate, unwarranted, attempted}` — the first two counting the rounds whose `chat_round.declined` carries that verdict, the third the rounds that attempted rather than refused. The three sum to `n_rounds`. It exists for the same reason `model_perf` does: this row outlives a Cleanup of the raw rounds, so a refusal rate derivable only from `chat_round` would disappear exactly when the run becomes a historical record. It counts refusals by the models under test only — requests the sufficiency gate turned back never became rounds, so they are in neither this tally nor `n_rounds` (see `chat_round.clarification`).
`retry_stats`	What the sufficiency gate cost this run in enrollee effort, as jsonb, and null when no gate could fire. It records `rounds_with_retries`, the mean and maximum `chat_round.retry_count` across the run's rounds, and a tally of why attempts were turned back — for example `{"rounds_with_retries":41,"retries_mean":0.7,"retries_max":4,"reasons":{"missing_slots":38,"out_of_scope":9}}`. It is the outcome measure of the Socratic switch, and the reason the switch has something to report at all: answer quality between the two arms is already in `composite_mean`, while this says how many attempts each arm needed before an answer existed. Like the rest of the scorecard it outlives a Cleanup of the rounds it was computed from (see `ALGORITHMS §6`, produced by `§21`).
`model_perf`	A per-model summary, `{catalog_model: {wins, losses, avg_score}}`, the per-run dud input.
`total_tokens`	The run-wide token sum. It is distinct from the per-round `tokens` distribution in the operational statistics below.
`total_generator_cost · total_judge_cost`	The run's US-dollar spend, split so a report can isolate the variable's effect. `total_generator_cost` sums `chat_round.token_cost` (the generator models under test); `total_judge_cost` sums `chat_round.judge_cost` (the judge's scoring overhead). The total spend is their sum, frozen at run end.
`total_latency_ms`	The per-round wall-clock summed across the run — cumulative latency, not compute time, and distinct from the calendar duration `started_at → ended_at`.
`computed_at`	When the scorecard was frozen, at run end.
`operational statistics` — the 12 `<metric>_<statistic>` columns	A fixed 3 × 4 matrix: for each of `iterations`, `latency_ms`, and `tokens`, the four statistics `_mean`, `_median`, `_min`, `_max` (for example `iterations_mean`, `latency_ms_max`), describing the distribution across rounds.
`score statistics` — `composite_mean`, `composite_median`, `factor_stats`	The composite headline is two fixed columns, `composite_mean` and `composite_median` — the two keys the comparisons sort by. Every score dimension, each rubric factor plus the composite, also carries a nonparametric five-number summary `{min, q1, median, q3, max}` in the `factor_stats` jsonb, keyed by factor. All of it is computed with the **enrollee as the unit** (over per-enrollee means), so a chatty enrollee cannot dominate and there is no separate round-weighted figure. Standard deviation is deliberately omitted: the interquartile range (q3 minus q1) is the distribution-free spread, assuming no Gaussian shape. The per-factor stats live in a jsonb bag because the factor set is configurable per experiment. Quantiles cannot be re-pooled across runs, so any drill-down reads the raw rounds.

nudge — A Reusable, Named Steering Message

An enrollee-safe instruction a study varies across its runs to see how it changes results. A run references one nudge through `experiment_run.nudge_id`. It is authored independently and reused, like a cohort or a model configuration. It is held immutable while any run references it: its substantive fields are read-only and it cannot be deleted (`RESTRICT`). To change a nudge, you clone it, which yields a new, distinctly-labeled condition. Only `notes` stay editable while referenced.

Column	Meaning · values · when populated
`id`	The surrogate primary key for the row, marked **PK**.

Column	Meaning · values · when populated
`name`	A short, unique display label, for example “concise” or “thorough,” marked **UK**. It is the stable key that analytics comparisons group and label by, distinct from the message text. Renames are blocked while any run references the nudge, so the label stays stable across a study.
`text`	The enrollee-safe steering message shown to enrollees — the nudge itself, the thing a study deliberately changes between runs. It is authored once and reused across runs and experiments.
`notes`	Free-text notes about this nudge. They are searchable by SQL keyword, and in the semantic index by meaning. They stay editable even while the nudge is referenced.
`created_by`	A foreign key to `profile` recording who first authored the nudge. It does not change on ownership transfer, and it is `ON DELETE SET NULL`; its email is preserved in `created_by_email`.
`created_by_email`	The author’s email captured at creation (**DENORM**, like `audit_log.actor_label`), so the origin survives after the profile is deleted and `created_by` goes null.
`updated_by_email`	The email of whoever last edited this nudge, captured on each update (**DENORM**).
`owner_id`	A foreign key to `profile` naming the current owner, which defaults to the author and can be transferred by the admin. Under row-level security (RLS), the owner may edit and delete the nudge, while peers read it and use it in their own runs. It goes null if the owner is deleted, leaving the nudge admin-managed.
`created_at / updated_at`	These record when the row was created and when it was last updated.
`version`	A counter that supports optimistic concurrency control (OCC), the check that stops two users from overwriting each other’s edits. It increases on every save.

run_enrollment — One Enrollee (A Faceless "Worker") in One Run

Column	Meaning · values · when populated
`id`	The surrogate primary key for the row, marked **PK**.
`run_id + enrollee_id`	The two columns are unique together. `run_id` is `ON DELETE CASCADE`, so an enrollee is purged with the run. Every enrollee uses the run’s one `model_config`, and there is no per-enrollee model.
`joined_at`	When the enrollee was enrolled in, or first joined, the run.
`withdrawn_at`	A soft remove. The enrollee stops receiving the condition, but their captured data is retained.
`last_activity_at`	The enrollee’s presence, refreshed as they act. It is what separates an **attendee** — an enrollee actively taking part right now, meaning not withdrawn and recently active — from one who is enrolled but idle. This attending status is derived, not stored, because presence drifts live; the recency window is a configurable threshold. An enrollee who leaves and returns continues the same session, a come-and-go pattern.

experiment_context_file — A Document Attached to an Experiment (A Pure Link)

Column	Meaning · values · when populated
`id`	The surrogate primary key for the row, marked **PK**.
`experiment_id`	A foreign key to `experiment`, unique together with `document_id` (marked **FK · UK**). Context is a property of the experiment, shared by all its runs, so it is not a per-run variable. It is `ON DELETE CASCADE`: deleting the experiment removes its links.
`document_id`	A foreign key to `document`, in the search module, marked **FK · UK**. It is `ON DELETE RESTRICT`: a document cannot be hard-deleted (decommissioned) while any experiment still attaches it. Removing the link never deletes the shared document.
This row is just the link. It scopes retrieval-augmented generation (RAG), the technique of feeding retrieved text to the model, to the experiment’s documents, so every run of the experiment draws on the same context. There is no ingest status here. A document is embedded once and shared across experiments, and its readiness is derived from the search sidecar. The set of context files is held immutable while the experiment has runs, so the grounding a finished run used stays fixed.	

message — One Out-of-Band Operator↔Enrollee Note

Column	Meaning · values · when populated
`id`	The surrogate primary key for the row, marked **PK**.
`run_id`	A foreign key to `experiment_run`. Messaging is scoped to a run, with `ON DELETE CASCADE`.
`sender_id`	A foreign key to `profile`. Messaging is bidirectional, so an enrollee can send. The sender's side is not stored: the enrollee client shows a non-self sender as "from Experimenter," through blinding and RLS.
`target_kind`	The message target: `direct`, `cohort`, or `everyone` for operator-to-enrollees, or `operator` for enrollee-to-run-staff. It carries both the direction and the broadcast distinction, so there is no separate `kind` column.
`target_cohort_id`	A foreign key to `cohort`, set exactly when `target_kind = cohort` and null otherwise; a CHECK constraint enforces that pairing.
`body`	The free-text of the message. This is the out-of-band operator↔enrollee note, not a turn in the chat with the large language model (LLM).
`created_at`	This records when the message was sent.

message_recipient — One Delivery/Read-Receipt Row per Reader

Column	Meaning · values · when populated
`message_id`	A foreign key to `message`, naming which message this receipt is for. It is half of the composite primary key, marked **PK · FK**, with `ON DELETE CASCADE`.
`recipient_id`	The fan-out, with one row per message and reader. The reader is an enrollee, for an operator-to-enrollee message, or a staff member, for an enrollee-to-operator message, and in both cases it is a generic profile. Every message creates one row per intended reader, so read receipts work both ways. It is the other half of the composite primary key, marked **PK · FK**.
`read_at`	The per-recipient read receipt: when this reader opened the message, or null when unread.

How They Relate (Reading the Lines)

- defines: one experiment has many runs.
- scorecard: each run has one `run_result` (one-to-one), built when it finishes.
- applies: many runs apply one nudge.
- enrolls: one run has many enrollees.
- context files: one experiment has many attached documents.
- messages: one run has many messages.
- delivered to: one message has many recipients.
- dashed tags: cross-module foreign keys to `profile` for the experiment and nudge authors and owners, the launcher, enrollees, senders, and recipients; to `cohort` for run targeting and the broadcast target; to the run's one `model_config`; and to the `document` a context file points at.

Two Rules That Live in Application Logic (Not Columns)

Targeting is always per cohort, never per individual, and concurrent runs must target non-overlapping cohorts, so no enrollee is live in two conditions at once.

Clearing a run is allowed only when it is done or aborted. Both operations archive the transcripts and scorecard to a file first, and if the export fails, nothing is deleted. There are two ways of clearing a run, with distinct intents:

- **Cleanup**, run by an admin or by the owning experimenter who composed or launched it, reclaims space. It deletes the heavy produced data — enrollees, chat rounds and candidates, these messages and recipients, and the run's audit rows — but keeps the `experiment_run` row and its `run_result` scorecard as the durable record, so the analytics stay queryable.
- **Purge**, admin only, removes the run entirely: it drops the `experiment_run`, which cascades the `run_result` away too, so it never dangles. A definition becomes deletable only after its last referencing run is purged.

Neither Cleanup nor Purge touches the reusable definitions — the experiment, the nudge, the model configuration and its models, or the cohort — nor the experiment's context files, which belong to the experiment rather than the run.