Govern a Saved Question

Admin Turn an experimenter's private draft into a shared one-click button — or take one back: unapprove it, retire it (undoably) and restore it, or delete it.

Screens Saved questions `saved-questions-admin.html` · Edit saved question `edit-saved-question.html` · Confirm `confirm-delete.html`

Writes the trust level `nl_query.status` · hidden, but kept `nl_query.retired_at` · the approval note `nl_query.notes` · who did what, and when `audit_log`

Key fields draft / approved / system `status` · retired when set `retired_at` · how often it was reused `hit_count` · the button caption `label` · the full question `canonical_prompt` · the SQL it runs `derived_sqls` · where it is offered `scope_key` · concurrent-edit guard `version`

1 · Given / When / Then

Human wording first; the exact table/field in `mono`.

GIVEN

The signed-in person is an **admin**, whose copy of the **Saved-questions Screen** shows every row: experimenters' drafts (`status = draft`), some reused often enough (`hit_count`) to be flagged as candidates, the approved questions, the retired ones, and the built-ins. An experimenter's copy of the same screen shows only their own drafts, drafts shared with them, and the approved and built-in questions that are not retired.

WHEN

Reviewing a draft. The admin filters to drafts and opens a candidate. **View** shows the button caption, the full question and the SQL it runs, read-only. If it is good as it is, they click **Approve**, adding a note if they want one. If the wording needs work, **Edit** opens the label, the question text and the notes; the SQL is not hand-edited. If the question should not exist, **Delete** asks for confirmation and removes the draft.
Taking an approved question back. Three controls. **Unapprove** returns it to its owner as a draft. **Retire** hides it from every Ask surface and from matching while keeping the row, its SQL, its use count and its notes; a retired question shows a *retired* badge and offers **Restore**, which makes it a button again. **Delete** is permanent and asks for confirmation; an admin may delete an approved question whether or not it was retired first.

THEN

Approval flips `status` from `draft` to `approved` , appends the stamped note to `notes` , and writes one `audit_log` row. From then on everyone in that context sees the question as a one-click button on their Ask surface, the engine may reuse it as a match for a typed question, and only an admin may edit it. Unapproval flips `status` back to `draft` ; retirement sets `retired_at` and restoration clears it; each writes its own `audit_log` row. Nothing is pushed to anyone; a button appears or disappears the next time an Ask surface loads.

writes `nl_query.status` , `nl_query.retired_at` , `nl_query.notes` , `audit_log`

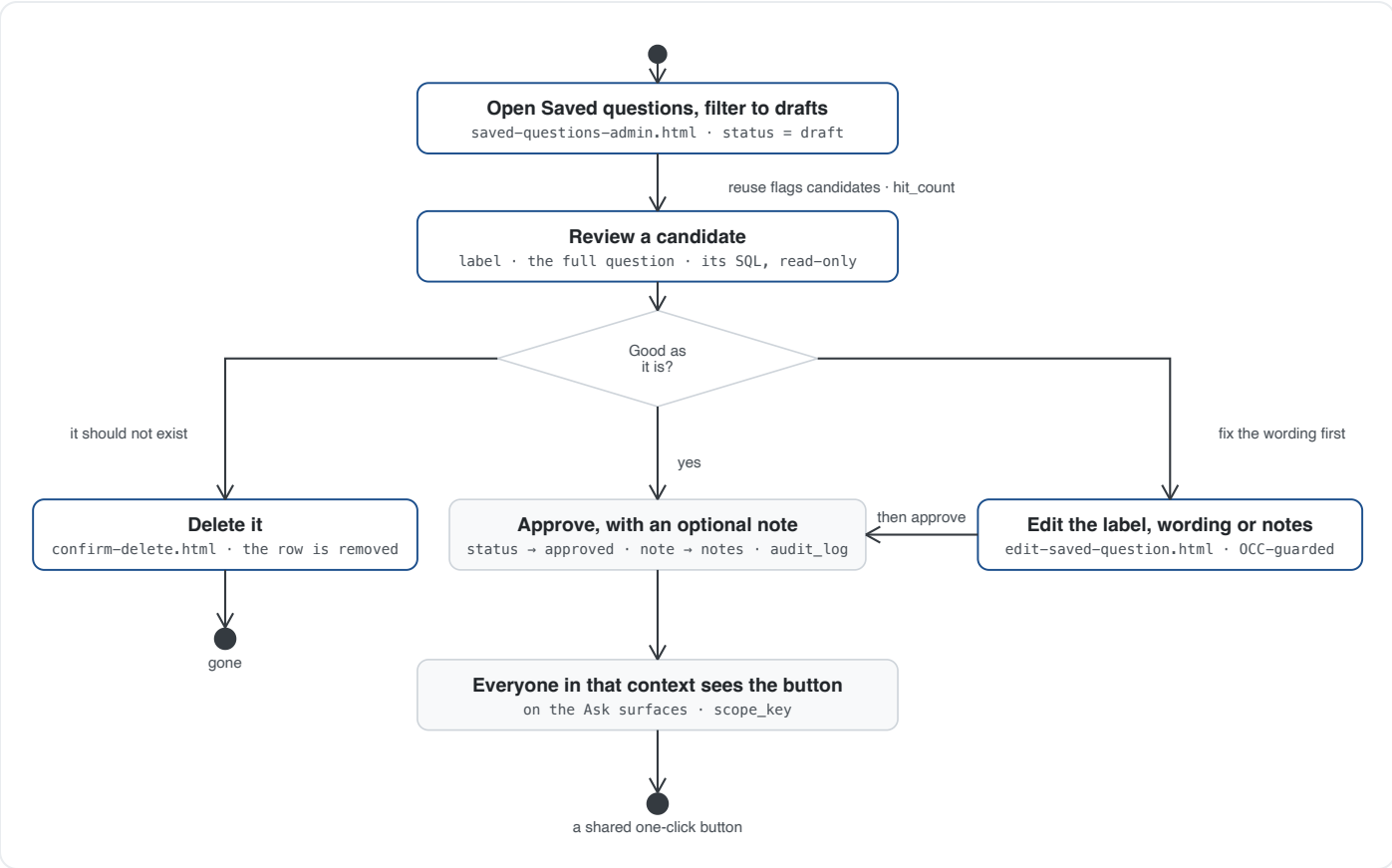
Variations and exceptions

IF…	THEN…
The row is already approved	Approve is replaced by Unapprove; there is nothing further to grant.
The row is a built-in (`status = system`)	Admins may edit it; it is never approved, unapproved, retired or deleted.
They try to unapprove a retired question	Refused until it is restored: `retired_at` may be set only while `status = approved` (a database CHECK).
`schema_fingerprint` no longer matches the tables the question reads	The stored SQL is regenerated from `canonical_prompt` before it runs again; review the regenerated SQL before approving such a draft.
Someone edited the row since it was loaded	The action is refused: every action carries the row's `version` , the screen says the record changed and asks for a reload, and nothing is overwritten.

Acceptance test — *Given* an admin and a draft saved by an experimenter, *When* the admin approves it with a note, *Then* `status = approved` , the note is appended to `notes` , one `audit_log` row carries it, and an experimenter's next Ask load shows the button; *and When* the admin retires that question, *Then* `retired_at` is set, the button is gone from every Ask surface at the next load, and the row is still there; *and When* the admin restores it, *Then* `retired_at` is null and the button is back.

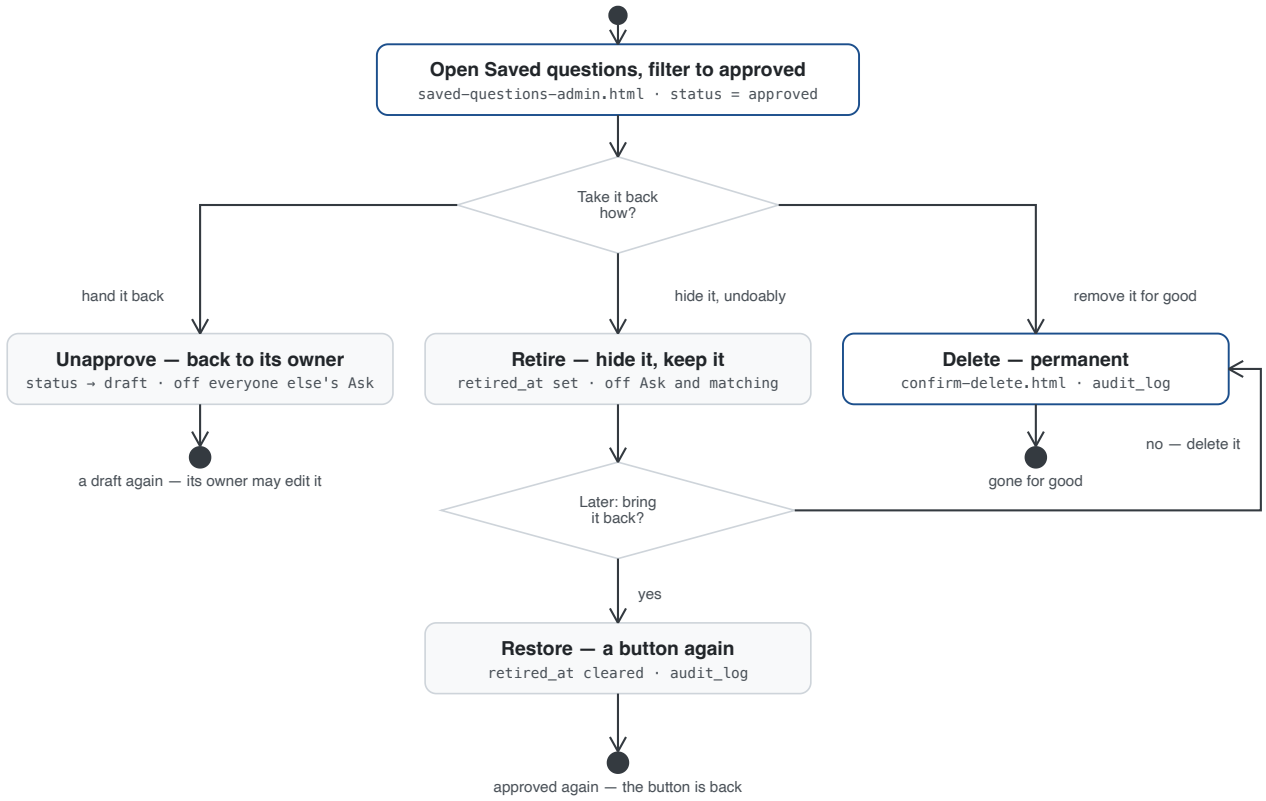
2 · The Journey — Review a Draft

Read it, then one of three outcomes.



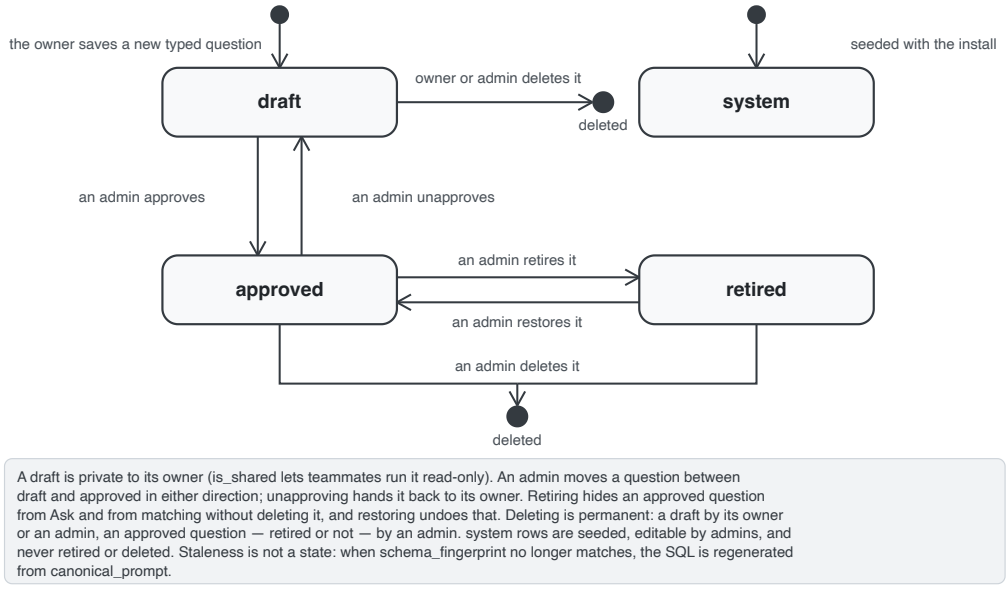
3 · The Journey — Take an Approved Question Back

Unapprove returns it to its owner; Retire parks it, undoably; Delete is permanent, from either state.



4 · The Life of a Saved Question

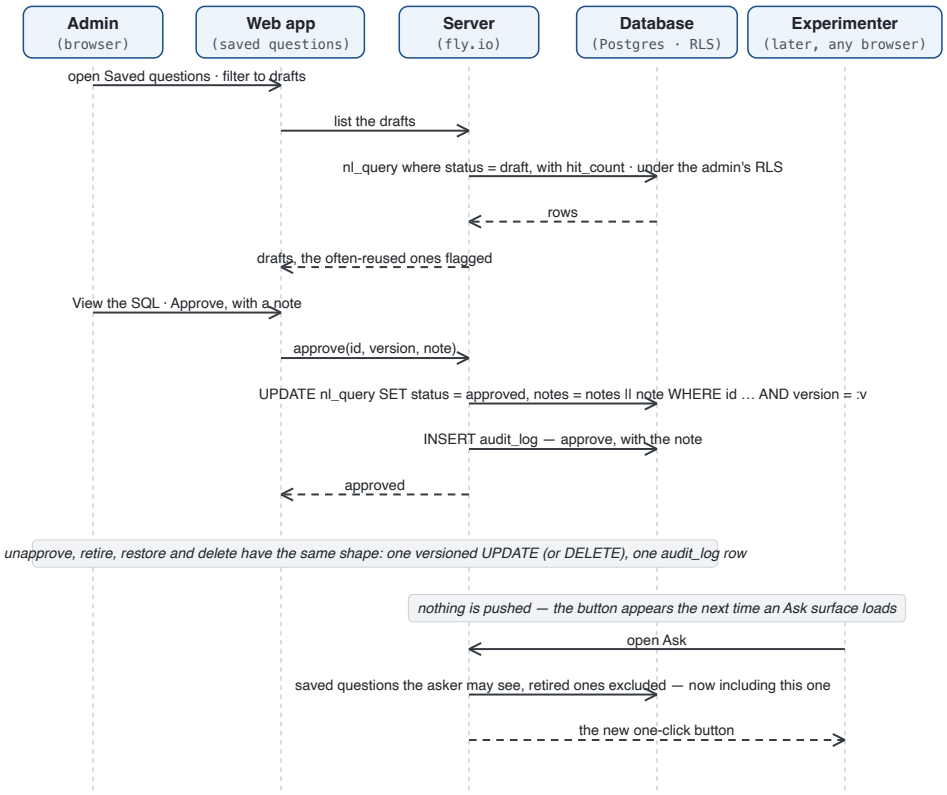
`nl_query.status` and `retired_at` together.



Shown as each row's status badge on the **Saved-questions Screen**; a retired question keeps `status = approved` and gains a *retired* badge. Deletion is permanent; retirement is the undoable alternative for an approved question.

5 · Behind the Scenes

One field flips, one audit row is written, and the next Ask load picks it up.



The experimenter is a separate browser at a later time; nothing is pushed to them. The read at the end is the ordinary visibility-filtered match set, which now includes the approved row and never includes a retired one.

6 · In Plain English

When someone types a good question into Ask and saves it, it is theirs alone. Making it a button for the whole team is an admin's call, and this is where that happens.

The screen lists every saved question. For an admin that means everything; for an experimenter it means their own drafts, whatever a teammate has shared, and the team's approved buttons. The admin's copy points out the drafts people keep reusing, since a question that has been asked two hundred times is probably worth sharing, and shows exactly what each one does — the caption, the full question, and the database query behind it — before any decision.

Three outcomes for a draft. Approve it, and it becomes a shared button on the spot, with a note saying why if the admin wants one. Tidy the wording first if the caption is unclear, then approve. Or, if it should never have been saved, delete it. Approval is recorded in the activity log, note included.

An approved button can also be taken back, three ways. Unapprove hands it back to its owner as a private draft. Retire hides it from everyone without deleting anything, so a question that stops earning its place can be parked and restored later. Delete is the permanent step, and it is the admin's call whether to park a question first or remove it outright. Built-in questions sit above all of this: they ship with the install and stay.

Fits the schema cleanly. Approval and unapproval are `nl_query.status` moving between `draft` and `approved`; retirement and restoration are `retired_at` set and cleared, with a CHECK that ties it to `status = approved`; the approval note lands in `notes`; every step writes an `audit_log` row and is guarded by `version`. Each changes one row and leaves `owner_id` alone; transferring a question is a separate admin action. *Retire* means the same soft `retired_at` here as it does for documents.