SumizAI .com Your Second Brain, Powered by Intelligence.
https://sumizai.com/blog/links-between-notes-and-backlinks.html

UNDER THE HOOD

How do notes link to each other — and what links back?

Linked notes are the oldest promise in personal knowledge management and the one most often quietly broken: the feature exists, the section renders, and no link is ever actually created. **SumizAI's version had exactly that bug once**, and the fix was to change who owns a link.

Updated: 2026-08-14 **8** questions 4 min read

Q. Where does a link between two notes live?

In the text of the note, as an ordinary Markdown link: `[Chunking strategies](chunking-strategies.md)`.

Not in a database table, and not in a *Related* section appended to the bottom. The link is a phrase in a sentence that happens to point somewhere, which is how links work in prose and how they work in every Markdown tool you might open the file with.

This was a correction to an earlier design. Links used to be rows in a table with a *Related* section rendered from them — and in real vaults there were none, because no write path ever filled the table in. The section rendered an empty list forever. Moving ownership into the text meant the link either exists in the file or does not exist at all.

Q. Who decides what gets linked?

The model that writes the note, working from a list it is not allowed to leave.

It is handed a map of up to eight candidates — title to filename — taken from the search results for the material it is writing about. It is asked to work a link into a sentence where the connection is genuine, rather than to produce a list.

Then a filter runs on the way in: **any `.md` target that is not in the map is removed**. External `http` links are left alone.

Q. Why strip links instead of trusting the model?

Because a model asked for a filename will produce a plausible one. Ask for a link to your note about retrieval costs and you may well get `retrieval-costs.md` — a perfectly sensible name for a file that does not exist.

A dead link is worse than no link, and it is worse in a specific way: it looks like a lead. You click it, nothing is there, and you now distrust every other link on the page.

So the constraint is deliberately strict. The model chooses where a connection belongs in the prose, and the application decides which targets are real.

Q. What are backlinks and where do they come from?

Under each note there are two lists: the links going out of this note, and the notes that link *into* it.

The outgoing list is read from the file, because the file is where links live. The incoming list comes from a derived column in the index — a question the content itself cannot answer without reading every other file in the vault on every page load.

That column is derived, not authoritative. It can be rebuilt from the files at any time by reindexing, which keeps the rule intact: the file is the source of truth, and the index only answers questions faster.

Q. What happens to links when I rename a note?

They are rewritten. The filename is the slug of the title, so a rename moves the file — and then every `](name.md)` reference to it across the rest of the vault is updated.

The order is chosen so a failure cannot leave you in a half-state: write the new file, rewrite the references, delete the old file. If anything fails, the rewritten references are rolled back and the new file is removed.

There is a trade-off here that was made on purpose. Names used to be permanent, which made renames trivial and folders unreadable. Now the folder reads properly from the outside, at the cost of a rename being a real operation — and archives you exported before the rename keep the old names.

Q. Can I rebuild links across a whole vault?

Yes — there is a vault-level *rebuild links* action that walks the whole thing.

It is the right tool after importing a batch of material, because notes written before their neighbours existed could not have linked to them. It is also how you recover from a period of heavy renaming.

The same guard applies: the model proposes a phrase from the existing text and a target file, and the application performs the substitution. The model does not get to rewrite your prose in the process.

Q. Can I write links myself?

Yes, and they behave identically, because there is nothing special about a generated link. It is a Markdown link in a Markdown file.

Point them at the real filename in `notes/` and everything works: the outgoing list picks them up, and the note on the other end will show yours in its backlinks after reindexing.

This is the practical benefit of not owning links in a database. The format is one everyone already knows, and your edits and the application's edits are the same kind of thing.

Q. Do the links survive an export?

That is the point of using relative filenames. A vault exports as a ZIP with `base.md` and a `notes/` folder, and the links between the files are relative paths.

Unzip it and open it in Obsidian, VS Code or a GitHub repository and the graph works — no plugin, no import step, no conversion.

It is the difference between owning your notes and owning a copy of your notes. The exported vault is not a snapshot of a structure that lives somewhere else; it *is* the structure.