Nexus: A Relationship Ledger for Research Computing

A knowledge graph and interaction ledger that started as a database script in September 2025 and is now the record my staff and our AI agents work from

Charles E. Forsyth III
Research Computing
University of California, Riverside
Riverside, California, USA
charles.forsyth@ucr.edu

Abstract

Nexus is the system of record for the relationships of Research Computing at the University of California, Riverside (UCR): a PostgreSQL knowledge graph of researchers, labs, departments, Google Cloud projects, assets and engagements, plus a ledger of the emails, meetings and decisions that touched them. I run it through the `nexus` command-line interface (CLI) and a hosted Model Context Protocol (MCP) server that lets AI assistants read it under a campus sign-in. On 28 October 2026 it held 972 researcher records, 408 organizational units, 176 cloud projects, 911 assets and 3,983 interactions.

The code dates from 5 January 2026: 541 commits and 212 tags, the latest v0.1.233, with the code itself at version 0.1.235. Its direct ancestor, a database and helper script for people and interactions, started on 5 September 2025, so the continuous line is about 14 months. The main lesson is that a ledger is worth what its write rules are worth: the rules that every write goes through the CLI, and that AI output is staged before it becomes a record, mattered more than any feature.

CCS Concepts

• **Information systems** → **Information integration**; **Enterprise information systems**; • **Human-centered computing** → *Human computer interaction (HCI)*; • **Software and its engineering** → *Software creation and management*.

Keywords

research computing, knowledge graph, interaction ledger, customer relationship management, PostgreSQL, pgvector, Model Context Protocol, AI agents, entity resolution

Author Note

The work was directed by me and carried out with AI agents under written briefs, independent checks and explicit approval steps. Figures come from the Nexus specification (SPEC.md, version 1.6), git history and tags, read-only `nexus stats` and `nexus interactions list` queries on 28 October 2026, and GitHub repository metadata. People, labs and some identifiers are replaced with role names; aggregate figures are real. This note was drafted with AI assistance from those records and reviewed by me. All numbers are real unless marked as representative.

Cite this note. C. E. Forsyth III. 2026. Nexus: A Relationship Ledger for Research Computing. Field Notes.
doi:10.5281/zenodo.23752279

1 Context

Research Computing supports roughly a thousand researchers across 22 colleges and schools, on the campus cluster, our CephRDS storage (built on Ceph [1]), 176 Google Cloud projects (knowledge graph, 5 October 2026), secure enclaves and a growing AI stack. Requests arrive by email, ServiceNow, Slack, meetings and hallway conversations. Before Nexus, the answer to "what is our history with this principal investigator (PI)?" lived in an inbox, a legacy customer relationship management (CRM) database, Jira and my memory. The questions I needed most were the hardest: who is this person and who works with them, what did we promise, which cloud projects do they use and who pays, and which faculty have we not spoken to in months.

The goal in the spec is to reduce "Time to Science": every feature has to make it faster to understand a researcher's situation and act on it. Nexus is a CRM in the literal sense, with the research enterprise as the customer. It is not a ticket system, a scheduler, a mail client or a cloud auditor. Other tools do those jobs, and Nexus records the result and the connections.

Governance and privacy. Nexus holds professional service records only: who asked Research Computing for what, what we did, and which labs, projects and services were involved. It holds no student academic records, personnel records or health information, and those are not to be logged. Access is limited to Research Computing staff by role, re-checked on every call. UCR is a public university, so I write each entry assuming it may one day be read under a public-records request: facts and commitments, no opinions about people. Retention, and whether a researcher can see their own record, still need a written policy; Section 8 returns to that.

2 Related Work

The research computing and data (RCD) profession has a shared vocabulary for what a support organization should do. The CaRCC RCD Capabilities Model [2] scores institutions on facets that include researcher engagement and outreach, which is the work a relationship ledger records. A name collision is worth stating up front: the NSF-funded RCD Nexus project [3], run with CaRCC, built a portal and community resources for RCD professionals. It is unrelated to the system described here, which took its name before I knew of the overlap.

The closest open tools in the field model allocations and usage. ColdFront [4] manages PIs, projects, allocations, grants and publications for HPC centers, and Open XDMoD [5] reports job and resource usage. Both answer "who has what and how much do they

use it" better than Nexus does; neither is built to hold the conversation history, commitments and people around a lab. Commercial platforms do hold that history: ServiceNow Customer Service Management [6], which UCR already runs for tickets, Salesforce Education Cloud [7] and Microsoft Dynamics 365 Sales [8]. I did not adopt one because the record had to be scriptable by agents from the first week, had to join to our own cloud audit data by project ID, and had to cost nothing to change. The price of that choice is that Nexus has one maintainer.

Nexus is a small knowledge graph in the survey sense of Hogan et al. [9]: typed entities and typed edges with a schema. It is stored relationally rather than in a property graph database such as Neo4j [10], and it differs from personal linked-note tools such as Obsidian [11] by enforcing types and identity at write time. Those tools descend from personal information management, which Jones [12] surveys as the keeping and re-finding of one person's information; Nexus applies the same habits to a shared, typed record for a team. Search uses pgvector [13] for embeddings and pg_trgm [14] for trigram similarity inside the same database, so there is no second store to keep in sync.

Linking the right person is a record-linkage problem. The resolver's order (exact identifiers before names, a fixed alias list before any fuzzy match) is the deterministic end of the spectrum that Fellegi and Sunter [15] formalized and Christen [16] surveys; it accepts lower recall for fewer false matches. Large language models can match entities with little or no task-specific training [17], but Nexus keeps the model out of the final match decision and uses it only to propose candidates for a person to confirm.

On the AI side, language models that call tools [18] are now ordinary, and MCP [19] standardizes how a client discovers and calls a server's tools. Keeping a person between model output and the record follows the human-AI interaction guidance of Amershi et al. [20] and the oversight functions of the NIST AI Risk Management Framework [21]. Excessive agency and prompt injection, both in the OWASP Top 10 for LLM applications [22], are the reasons destructive verbs are kept off MCP. The guard tests borrow the idea of mutation analysis [23].

3 Lineage

It feels to me like I have used Nexus for years. The repositories say something more modest, shown in Table 1: about 14 months of continuous lineage, of which Nexus proper is 10.

The ledger reaches further back than the code. Its oldest interaction is dated 2003 and several are from 2022, because imports kept the original event dates. Those are history brought in, and I don't count them as years of use.

The pace was uneven. Of the 541 commits, 246 landed in January 2026, when the core went in: the scaffold, the dossier, identity resolution and AI ingest on day one, Cloud SQL and hybrid search on day two. March, April and July saw 9 commits between them. Then 58 in September and 137 in October, when agents became the main users and the CLI had to grow a machine contract.

Table 1: The lineage of Nexus, from repository metadata and commit history.

Dates	System and what it was
Before Sep 2025	**Legacy research CRM.** A PostgreSQL database for people and contacts, imported once by `nexus sync legacy` (6 Jan 2026).
5 Sep–23 Dec 2025	**Research Data Hub workspace.** A unified PostgreSQL database for people and interaction events, a Python helper as the required interface, and a separate SQLite inventory of cloud projects joined by project ID; 107 commits. Its rule "verify the campus ID before logging" survives in Nexus.
19 Nov 2025	**A CRM prototype.** One commit, made in Google AI Studio.
5 Jan 2026–now	**Nexus.** The CLI, the graph and the ledger; 541 commits, 212 tags.
Feb 2026	**Offshoots.** A web front end and a multi-agent experiment on Google's Agent Development Kit; neither became the main path.

4 The Graph and the Ledger

The schema has 8 entity tables, one edge table and two support tables. Researchers, labs (which hold every unit from a lab up to a college), cloud projects, research projects, grants, assets, tasks and interactions each get a strictly typed table with a JSONB `details` column for loose facts and a 768-dimension embedding for semantic search. One polymorphic `connections` table links any two entities with a typed edge: `PI_OF`, `MEMBER_OF`, `OPERATES`, `PARTICIPATED_IN`, `REFERENCED_IN` and 14 others in use. There are no per-pair join tables. The trade is that the database cannot cascade deletes, so the delete function removes a node's edges itself.

Interactions are the ledger. Each one keeps its date, a one-line summary, the full original text, and edges to every person, unit and project involved. On 5 October 2026, 6,944 of 12,681 edges were `PARTICIPATED_IN`: more than half the graph is "who was in this conversation". That is the part I query most. Writes follow a few rules from the spec:

- **One transaction per command.** A decorator commits on success and rolls back on any error; command bodies never commit.
- **Identity before linking.** A resolver tries UUID, email, campus ID, exact name and a fixed list of nickname pairs before any fuzzy match, and fuzzy matches need 0.8 similarity. A wrong link is worse than a missing one, so a failed name gets an alias pair, never a looser threshold.
- **The "Glass Wall".** Gemini may classify free text and extract a summary and named entities, but a person confirms each link before it is written. If the model fails, the interaction still saves with the raw text.
- **Correct, don't erase.** `interactions edit` rewrites a summary in place and keeps the prior one under `details.superseded`.
- **Provenance on every write.** Since v0.1.213 one database hook stamps who wrote each row, through which client, and when.

Tasks deserve a note. Nexus had them from the first week and logged 237. On 7 October 2026 tasks moved to a Jira project, with Hermes, my agent assistant, as the single Jira writer. Nexus tasks

are now a read-and-update archive, and the MCP tool to add one was removed in v0.1.233.

5 Architecture

Figure 1 shows the parts. Everything goes through the real CLI code, whether the caller is me at a terminal, an agent on my laptop, or a colleague's assistant over MCP.

The CLI is Python 3.12 with Typer and Rich, about 14,000 lines in `src/nexus` today (12,518 when the spec was last measured), packaged as a uv tool that installs pinned to its lockfile. The database is PostgreSQL 16 on Cloud SQL with pgvector [13] and pg_trgm [14] for hybrid exact, fuzzy and semantic search. Clients reach it only through the Cloud SQL connector with their own Google identity and Identity and Access Management (IAM) database login [24]. Deletion protection and encrypted-only connections went on in September 2026, with daily backups and 7 days of point-in-time recovery.

Slow tools stay off the read path. Cloud audits from Skywalker, my Google Cloud audit tool, are cached in a table (155 rows) and refreshed on demand. Mail goes through astropost, my Gmail CLI. For dashboards, `nexus serve` keeps one warm engine open locally so a call takes milliseconds instead of a 3 to 8 second cold start; its policy refuses resets and bulk commands and makes every delete wait for a confirmation.

The hosted MCP server is the part other people touch. It follows the model of our cluster MCP server: sign-in with a campus Google account checked against an allowlist of users with roles, the role re-checked on every call, and one audit line per sign-in and tool call. It exposes 40 tools: 23 reads, 16 staff writes that run the real CLI command as the caller, and one admin action that needs a preview and a confirmation. Every delete, merge, reset, bulk import and email stays off MCP. A mutation check [23] disables each of the server's security guards in turn and requires a test to fail. At v0.1.219 there were 38 guards, all 38 were caught, and a 39th, no-op mutation, the control, correctly survived.

Around it sits a small family of servers. Atrium, the researcher front door we are building, is designed to show researchers a narrow projection of Nexus while staff keep the full tool set. The AI gateway's admin server (aigw) is planned to read Nexus for lab and member suggestions only, labelled as hints, because the campus directory is the authority on people. people-mcp, a read-only directory server, is what corrects Nexus when they disagree. The daily agent work that ties them together, and the approval gates on mail, are in the companion Field Note "A Seneschal for the Research Computing Desk" [25].

6 Growth in Numbers

Table 2 compares three read-only snapshots. The drop between 2 and 5 October is a cleanup, described in Section 8; changes since then are from use and continued cleanup.

The ledger grew unevenly (Figure 2). By event date and calendar month, the read-only query gave 228 interactions in January 2026, 263 in February, 238 in March, 119 in April, 214 in May, 98 in June and 37 in July, then 398 in August, 402 in September and 358 in October through the 28th. On 28 October there were 90 in the last 7 days, 390 in the last 30 and 1,158 in the last 90. The thin months match the commit log: I was busy elsewhere and logging slipped.

Table 2: Graph size at three points, from SPEC.md section 12 (2 and 5 October 2026) and `nexus stats --json` (28 October 2026). A dash means not recorded.

Records	2 Oct	5 Oct	28 Oct
Researchers	992	943	972
Units (labs to colleges)	494	431	408
Cloud projects	171	176	176
Research projects, services	38	28	47
Assets	–	916	911
Interactions	–	3,706	3,983
Tasks (archive since 7 Oct)	–	225	237
Edges	13,514	12,681	–

The scoreboard shows what the ledger is for. Coverage of 48 tagged senior contacts over 90 days rose from 22.9% (11 of 48) on 5 October to 56.2% (27 of 48) on 28 October. Some of that is better logging, not more contact, and I can't separate the two from these numbers. Who writes is changing too: of the 447 rows stamped with provenance from v0.1.213 through 5 October (cumulative), 280 came through MCP, 121 from Hermes and 46 from the CLI.

The repository has 46 test modules and 331 test functions run against a real Postgres with pgvector in continuous integration, with lint, format and type checks on every pull request.

7 How It Gets Used

The operating loop in the spec is plain. Something happens. I look before acting: `search`, `dossier`, `tree` and `interactions list --contains` give the history and the people around it. I act. Then I record it with `nexus log`, linking each person, unit and project explicitly, with `--strict-links` so a missing link fails instead of passing quietly.

A typical case uses stand-ins. Researcher A, a PI in a biology lab, writes about a cloud bill. Before anyone replies, the dossier shows the lab's two cloud projects, the cached audit of each, three earlier conversations about storage, and that a staff member already promised a cost review. The reply cites the history instead of rebuilding it, and the outcome goes back into the ledger linked to the PI, the lab and the project. The next person to ask, human or agent, starts there.

Nexus also writes. The daily briefing, meeting dossiers (terminal, JSON or PDF), lists of researchers we have not followed up with, and the dated work files in the repository root come from it: 35 briefings and 4 captain's logs among 201 dated Markdown files. Since 5 October 2026 it holds the Research Computing service catalog as tagged research projects, mirrored from the public website.

Mostly it is driven by agents now. Claude Code works in the repository, Hermes runs the same procedures for me through the MCP tools, and a colleague reads through a chat assistant. Gemini-based assistants may read and never write. That rule came from experience: two writers on one system once produced duplicate tickets and a wrongly closed issue.

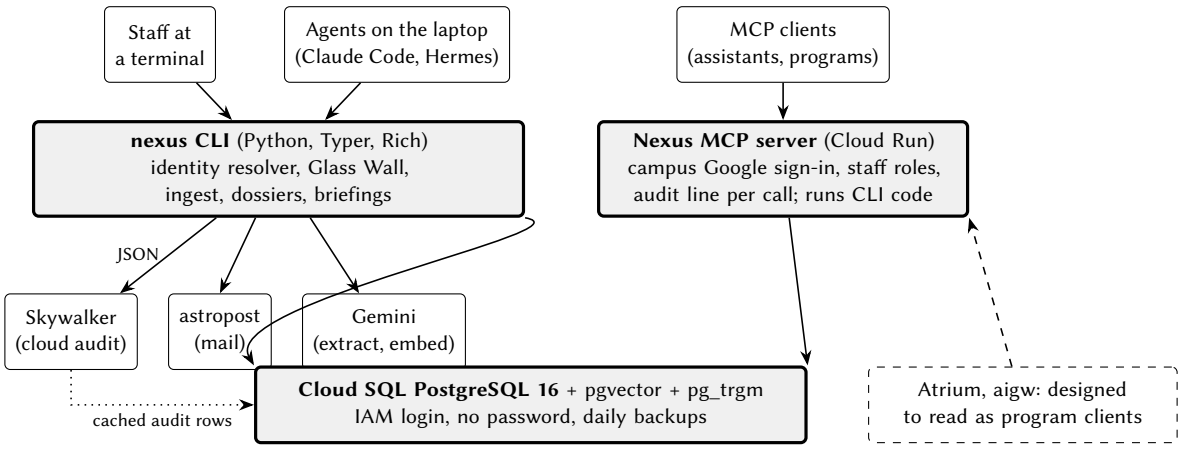


Figure 1: Nexus architecture, drawn from SPEC.md section 3. Every write path ends in the same CLI command code, so the transaction, identity and provenance rules apply no matter who calls. Dashed elements are designed, not built.

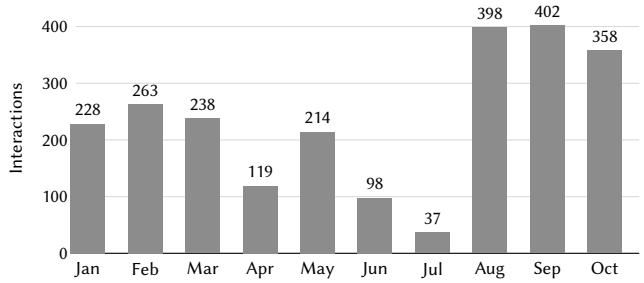


Figure 2: Interactions logged per calendar month of 2026, by event date (October through the 28th), from read-only `nexus interactions list` queries.

8 What Went Wrong

The graph drifted. A read on 2 October 2026 found 1,024 dangling edges, 79 placeholder researchers named things like "Auto-Resolved" plus an ID, duplicates nobody could prove, and departments that existed twice, once filed as a lab. The cleanup's own audit counted 994 people (the table count that day was 992; the two reads differ by two records and I report each where its source does); 272 had no department and 369 no email. Most dangling edges (749) came from tasks deleted before the delete function cleaned up edges. Fixing it took a cleanup on 4 and 5 October: a read-only enrichment from the campus directory, then merges and renames, each rehearsed on a restored copy of production, backed up and diffed first. People went from 994 to 943 and units from 494 to 431; dangling edges and placeholders went to zero.

Matching by email trusted the email. One stored address belonged to a different person with a similar name, so enrichment attached the wrong directory record. I caught it on a re-read, and v0.1.225 now skips an email match whose given name disagrees. A re-check found only that one case, but it is the failure the identity rules exist to prevent.

A library broke every command. On 25 September 2026 a plain tool upgrade pulled a new `sqlmodel` and every command failed on a datetime mapping. Installs now go through a script that pins to the lockfile.

Tasks were the wrong fit. Nexus tasks and Jira tickets drifted: a one-time crosswalk on 7 October found 41 open Nexus tasks, 20 with no Jira ticket, and 11 open Jira tickets with no Nexus task. Moving tasks to Jira was the answer, and it means 237 Nexus tasks are now history.

One maintainer. Nexus has one author, much of the code was written by AI agents under my direction (git metadata records this only for some commits), the main CLI file is 5,464 lines today, and there is no schema migration tool, so a column change would need manual SQL the rules forbid. If I left, UCR would have a well-documented database and a spec, not a supported product. A runbook and a second staff maintainer are the honest fix.

Gaps remain. Grants hold 3 records, so the "why" third of the model is mostly empty. 520 interactions have no links. "Project" still means three things across Nexus and Skywalker. The senior-contact score mixes behavior and bookkeeping. Retention and self-access policies are not yet written.

9 Lessons

(1) Make every write go through one command path. Then transactions, identity checks and provenance hold for humans, scripts and agents alike.
(2) Keep a person between the model and the record. Extraction errors are cheap when staged and expensive once linked.
(3) Prefer a missing link to a wrong one. A wrong link spreads into every dossier and briefing that reads it.
(4) Give agents JSON and refuse them the dangerous verbs. Scraped Rich tables can wrap and truncate, so a grep returns false negatives.
(5) Check the graph against an outside authority. Drift was invisible until the campus directory was read against it.

(6) Rehearse bulk changes on a restored copy and diff the result. Every cleanup change in October went this way.

10 What's Next

- A read-only reconcile between Nexus cloud project records and Skywalker's live list.
- A schema migration tool, a runbook and a second maintainer.
- Written retention and self-access policies, reviewed with campus records staff.
- Importing active grants, so the "why" in the model has data.
- Atrium's narrow researcher-facing view of Nexus, once staging is reviewed.
- An outcome measure: time to first reply on requests where the dossier was used, against requests where it was not, which would show whether the ledger saves time.

11 Sources

Nexus and its precursors are private because they hold researcher records and interaction history. Private records used: the Nexus repository (SPEC.md version 1.6; README; git history from 5 January to 28 October 2026; 212 tags); read-only `nexus stats --json` and `nexus interactions list` queries on 28 October 2026; the Nexus and Jira crosswalk of 7 October 2026; the Atrium tools document of 20 October 2026; the Research Data Hub repository README and commit history; GitHub metadata for the prototype and offshoot repositories. Public companion notes: "A Seneschal for the Research Computing Desk" and "One Day on the Research Stack" [26].

References

[1] Sage A. Weil, Scott A. Brandt, Ethan L. Miller, Darrell D. E. Long, and Carlos Maltzahn. 2006. Ceph: A Scalable, High-Performance Distributed File System. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI '06)*. USENIX Association, 307–320. https://www.usenix.org/conference/osdi-06/ceph-scalable-high-performance-distributed-file-system

[2] Patrick Schmitz, Claire Mizumoto, John Hicks, Dana Brunson, Gail Krovitz, James Bottum, Joel Cutcher-Gershenfeld, Karen Wetzel, and Thomas Cheatham. 2020. A Research Computing and Data Capabilities Model for Strategic Decision-Making. In *Practice and Experience in Advanced Research Computing (PEARC '20)*. ACM, New York, NY, USA, 77–84. doi:10.1145/3311790.3396643

[3] U.S. National Science Foundation. 2021. Award 2100003: CI CoE: Demo Pilot: Advancing Research Computing and Data: Strategic Tools, Practices, and Professional Development (RCD Nexus). NSF Award Search. https://www.nsf.gov/awardsearch/showAward?AWD_ID=2100003 Accessed 28 October 2026.

[4] Andrew Bruno and Dori Sajdak. 2021. ColdFront: Resource Allocation Management System. In *Practice and Experience in Advanced Research Computing (PEARC '21)*. ACM, New York, NY, USA, 1–5. doi:10.1145/3437359.3465585

[5] Jeffrey T. Palmer, Steven M. Gallo, Thomas R. Furlani, Matthew D. Jones, Robert L. DeLeon, Joseph P. White, Nikolay Simakov, Abani K. Patra, Jeanette Sperhac, Thomas Yearke, Ryan Rathsam, Martins Innus, Cynthia D. Cornelius, James C. Browne, William L. Barth, and Richard T. Evans. 2015. Open XDMoD: A Tool for the Comprehensive Management of High-Performance Computing Resources. *Computing in Science & Engineering* 17, 4 (2015), 52–62. doi:10.1109/MCSE.2015.68

[6] ServiceNow. 2026. Customer Service Management (CSM). Product documentation. https://www.servicenow.com/docs/bundle/zurich-customer-service-management/page/product/customer-service-management/concept/c_CustomerServiceManagement.html Accessed 28 October 2026.

[7] Salesforce. 2026. Education Cloud: Education CRM Software. Product documentation. https://www.salesforce.com/education/ Accessed 28 October 2026.

[8] Microsoft. 2026. Welcome to Dynamics 365 Sales. Microsoft Learn. https://learn.microsoft.com/en-us/dynamics365/sales/overview Accessed 28 October 2026.

[9] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia D'amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. 2021. Knowledge Graphs. *Comput. Surveys* 54, 4 (2021), 1–37. doi:10.1145/3447772

[10] Neo4j. 2026. What Is a Graph Database. Neo4j Getting Started documentation. https://neo4j.com/docs/getting-started/graph-database/ Accessed 28 October 2026.

[11] Obsidian. 2026. Obsidian. Product website. https://obsidian.md/ Accessed 28 October 2026.

[12] William Jones. 2007. Personal Information Management. *Annual Review of Information Science and Technology* 41, 1 (2007), 453–504. doi:10.1002/aris.2007.1440410117

[13] pgvector contributors. 2026. pgvector: Open-Source Vector Similarity Search for Postgres. GitHub repository. https://github.com/pgvector/pgvector Accessed 28 October 2026.

[14] PostgreSQL Global Development Group. 2026. PostgreSQL 16 Documentation, F.35: pg_trgm, Support for Similarity of Text Using Trigram Matching. https://www.postgresql.org/docs/16/pgtrgm.html Accessed 28 October 2026.

[15] Ivan P. Fellegi and Alan B. Sunter. 1969. A Theory for Record Linkage. *J. Amer. Statist. Assoc.* 64, 328 (1969), 1183–1210. doi:10.1080/01621459.1969.10501049

[16] Peter Christen. 2012. *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection.* Springer, Berlin, Heidelberg. doi:10.1007/978-3-642-31164-2

[17] Avanika Narayan, Ines Chami, Laurel Orr, and Christopher Ré. 2022. Can Foundation Models Wrangle Your Data? *Proceedings of the VLDB Endowment* 16, 4 (2022), 738–746. doi:10.14778/3574245.3574258

[18] Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*. Curran Associates, 68539–68551. doi:10.52202/075280-2997

[19] Model Context Protocol. 2025. Model Context Protocol Specification, Version 2025-06-18. https://modelcontextprotocol.io/specification/2025-06-18 Accessed 28 October 2026.

[20] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N. Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz. 2019. Guidelines for Human-AI Interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. ACM, New York, NY, USA, 1–13. doi:10.1145/3290605.3300233

[21] Elham Tabassi. 2023. *Artificial Intelligence Risk Management Framework (AI RMF 1.0).* Technical Report NIST AI 100-1. National Institute of Standards and Technology, Gaithersburg, MD. doi:10.6028/NIST.AI.100-1

[22] OWASP Gen AI Security Project. 2025. OWASP Top 10 for Large Language Model Applications. https://genai.owasp.org/llm-top-10/ Accessed 28 October 2026.

[23] Yue Jia and Mark Harman. 2011. An Analysis and Survey of the Development of Mutation Testing. *IEEE Transactions on Software Engineering* 37, 5 (2011), 649–678. doi:10.1109/TSE.2010.62

[24] Google Cloud. 2026. IAM Authentication, Cloud SQL for PostgreSQL. Google Cloud documentation. https://cloud.google.com/sql/docs/postgres/iam-authentication Accessed 28 October 2026.

[25] Charles E. Forsyth III. 2026. A Seneschal for the Research Computing Desk. Field Note. https://charles-forsyth.github.io/papers/2026-10-29-a-seneschal-for-the-research-computing-desk.html

[26] Charles E. Forsyth III. 2026. One Day on the Research Stack. Field Note. https://charles-forsyth.github.io/papers/2026-10-03-one-day-on-the-research-stack.html