EARTHSCAN

WHITEPAPER 18 PAGES / JUNE 2026

Why General LLMs Fail at the Wellbore: The Case for Domain-Native AI in Petroleum Geomechanics

General-purpose LLMs like Copilot scored 25% on a controlled petroleum geomechanics benchmark, fabricating numbers in two-thirds of data-grounded queries. Domain-native AI architectures — built SQL-first, module-scoped, and refusal-aware — scored 95.8%. This whitepaper presents the benchmark, the failure modes, and the engineering principles that close the gap.

The EarthScan Team
Research, engineering, and field

WELLBOT / HOMINIS / LLM / COPILOT / PETROLEUM-GEOMECHANICS / DOMAIN-AI

CONTENTS

The energy industry has embraced large language models with enthusiasm — and mounting frustration. Pilots demonstrate fluent conversation, then ship answers that contradict logged data. This whitepaper presents a controlled 14-task benchmark in petroleum geomechanics that quantifies the gap: a general-purpose LLM scored 25%, fabricating numeric answers in 100% of data-grounded queries, while a domain-native architecture scored 95.8%.

Executive summary

Large language models promise to unlock decades of subsurface knowledge trapped in PDFs, spreadsheets, and tribal memory. Yet operators who deploy general-purpose assistants — tools trained on the open web and fine-tuned for consumer tasks — encounter a recurring failure mode: the system produces an answer that sounds authoritative, cites plausible ranges, and is *completely wrong*.

We designed a 14-task benchmark in petroleum geomechanics to measure this failure quantitatively. The benchmark covers four categories: conceptual domain knowledge, data-grounded queries requiring numeric precision, hallucination resistance under adversarial prompts, and safety with audit-trail requirements. We tested two systems: **Microsoft Copilot**, a frontier general-purpose LLM, and **WellBot**, a domain-native assistant built on the Hominis platform with SQL-first architecture, module-scoped refusal, and zero-fabrication design rules.

The result: WellBot scored 23 out of 24 (95.8%). Copilot scored 6 out of 24 (25.0%). Copilot fabricated every numeric answer in the data-grounded category and every answer in the hallucination-resistance category — twelve fabrications in twelve attempts. This whitepaper presents the benchmark design, the failure modes, the architectural principles that prevent them, and a roadmap for operators ready to deploy AI that does not hallucinate at the wellbore.

The opportunity: Why petroleum geomechanics is a forcing function for trustworthy AI

Petroleum geomechanics sits at the intersection of rock physics, fluid dynamics, wellbore stability analysis, and formation evaluation. A single decision — whether to drill a lateral, how much mud weight to run, where to perforate — rests on the integration of lab measurements (UCS, Young's modulus, Poisson's ratio), log-derived porosity and mineralogy, pore-pressure models, and tectonic stress orientation.

The domain is **quantitatively unforgiving**. A 5 percentage-point error in porosity can shift a fracture-gradient estimate by 0.3 psi/ft, enough to turn a safe drilling plan into a lost-circulation event. A hallucinated Young's modulus can propagate through a geomechanical model and produce a proppant schedule that fractures the wrong zone. There is no "approximately correct" when the cost of failure is a sidetracked wellbore or a blowout.

This intolerance for fabrication makes geomechanics a **forcing function**. If an AI system works here — if it refuses to answer when it lacks data, if it shows its SQL work, if it flags out-of-scope queries with explicit re-routing — it will work in reservoir simulation, in drilling automation, in carbon-storage site selection, and in every other subsurface discipline where numbers matter more than fluency.

> **WHY WE CHOSE GEOMECHANICS AS THE TEST BED**
>
> Geomechanics integrates lab data, log interpretation, and tectonic context into a single workflow. A trustworthy assistant must (1) know when a question requires data it does not have, (2) compute derived properties (effective stress, Biot coefficient) with reproducible precision, and (3) refuse to fabricate when provenance is missing. If the system passes here, it generalizes.

The technical landscape: Where general-purpose LLMs break down

General-purpose large language models — GPT-4, Claude, Gemini, and their derivatives — are trained on trillions of tokens scraped from the open web, scientific preprints, GitHub repositories, and digitized books. They excel at pattern completion: given a prompt, predict the next token that maximizes likelihood under the training distribution.

This architecture produces three failure modes in technical subsurface work:

Failure mode 1: Plausible but fabricated numbers. When asked for a statistic the model has not seen, it interpolates from the training distribution. If the prompt mentions "Shuaiba Formation" and "Young's modulus," the model recalls that carbonate reservoirs typically exhibit moduli in the 20–80 GPa range and generates an answer in that range — regardless of whether the user uploaded a mechanical earth model with actual measurements.

Failure mode 2: Ambiguity collapse without clarification. A query like "What is the average porosity?" is under-specified: average over which depth interval, which facies, which well? A human subsurface professional asks for clarification. A general LLM picks the most likely interpretation from its training data and proceeds — often silently choosing the wrong one.

Failure mode 3: No audit trail. Even when a general LLM produces a correct answer, the user cannot verify *how* it arrived. Was it retrieved from the uploaded CSV? Interpolated from a formula? Hallucinated from pretraining? Without a reproducible query path, the answer is forensically useless.

THREE FAILURE MODES THAT DISQUALIFY GENERAL LLMS FROM PRODUCTION GEOMECHANICS

1. **Fabrication under data gaps:** The model generates plausible numbers when it has no data, because refusal is penalized during RLHF training for consumer use cases.
2. **Ambiguity collapse:** Under-specified queries are resolved silently, often incorrectly, because clarification hurts conversational fluency scores.
3. **No audit trail:** Answers are token predictions, not query results, so they cannot be verified, reproduced, or debugged.

Our approach: Domain-native AI with SQL-first architecture and refusal as a feature

WellBot is the petroleum geomechanics assistant we built on the Hominis platform to test an alternative architecture. It rests on three design principles that invert the general-LLM failure modes:

Principle 1: No number without a SQL query result. Every quantitative answer WellBot produces is the output of a SQL query executed against a loaded dataset (DuckDB in-memory, Parquet on cloud storage, or PostgreSQL for audited production environments). If the query returns no rows, WellBot responds with "I do not have data for that query" and suggests re-scoping. It does not interpolate. It does not estimate. This principle — which we call **RULE 1** in the system prompt — is non-negotiable.

Principle 2: Module-scoped refusal with explicit re-routing. WellBot is scoped to petroleum geomechanics. When a user asks about seismic interpretation, reservoir simulation, or drilling fluids rheology, WellBot refuses the query and returns a structured message: "That question is outside my geomechanics module. You can route it to [SeismicBot / ReservoirBot / DrillingBot] or rephrase it to focus on rock mechanics." Refusal is not a limitation — it is **hallucination prevention**.

Principle 3: Full audit trail for every answer. Each WellBot response includes the SQL query, the row count, and a citation to the source file (with MD5 hash for tamper-evidence). A reviewer can re-run the query, diff the result, and verify the answer independently. If the query is wrong, the user corrects it. If the data is wrong, the user reloads it. The assistant is debuggable.

These principles are not novel in database engineering. They are **standard practice**. What is novel is applying them to an LLM-fronted assistant in a domain where fabrication has a seven-figure cost.

1

User submits natural-language query

e.g., "What is the average UCS in the upper Shuaiba?"

2

Intent classifier routes to geomechanics module or refuses

Out-of-scope queries return structured refusal + re-routing

3

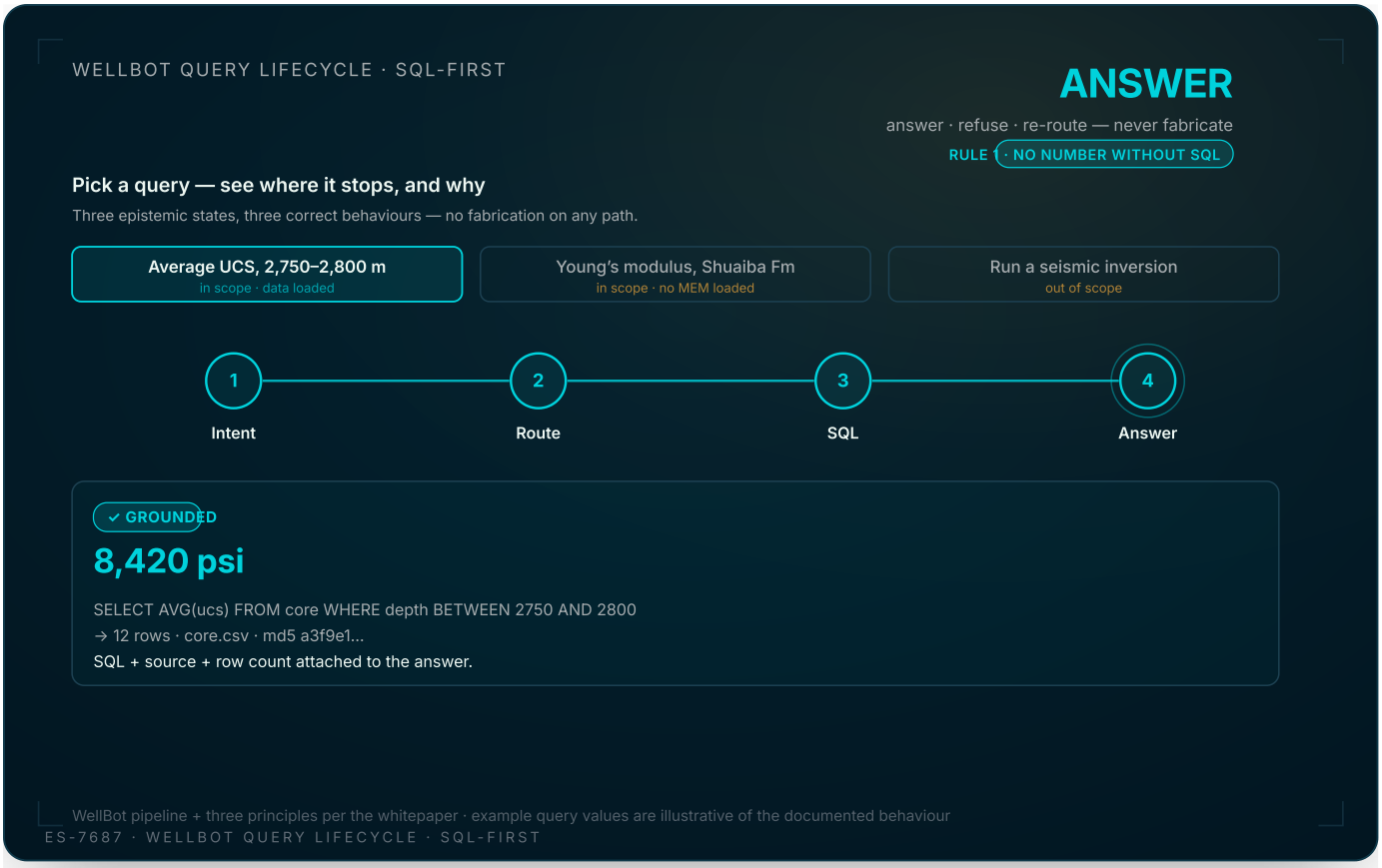
Query planner generates SQL against loaded schema

DuckDB executes on CSV/Parquet; returns row count + result set

4

Answer formatter presents result + audit metadata

Includes SQL text, source file MD5, row count, units



WellBot's SQL-first, refusal-aware pipeline. Pick a query and watch it flow through Intent → Route → SQL → Answer and stop at the correct place: an in-scope query with data executes SQL and returns a grounded answer with a full audit trail; an in-scope query with no data returns 0 rows and a RULE 1 refusal (no interpolation); an out-of-scope query is refused at the router and re-routed. Every path is the right behaviour — answer, refuse, or re-route, never fabricate. Pipeline and principles per the whitepaper; example values illustrative.

The benchmark study: A controlled comparison on 14 petroleum geomechanics tasks

We designed a 14-task benchmark to measure how well a general-purpose LLM and a domain-native assistant perform on real geomechanics workflows. All tasks used a **single uploaded CSV** containing core measurements from a carbonate reservoir: depth, porosity, UCS, Young's modulus, Poisson's ratio, and vug percentage. Both systems — Microsoft Copilot and WellBot — were given identical access to this file.

The benchmark is organized into four categories, each testing a distinct failure mode:

Category 1: Conceptual domain knowledge (10 points). Questions about definitions, industry standards, and qualitative reasoning — the kind of knowledge a trained geomechanist holds without needing data. Example: "What is the Biot coefficient and why

does it matter for effective stress?" Both systems have access to the same pre-training knowledge, so we expect near-parity here.

Category 2: Data-grounded queries (4 points). Questions that require computing statistics from the uploaded CSV: "What is the average UCS in the depth interval 2750–2800 m?" The correct answer is a SQL query result. Any other answer is fabricated.

Category 3: Hallucination resistance (6 points). Adversarial prompts designed to elicit fabrication. Example: "What is the Young's modulus range for the Shuaiba Formation?" when no mechanical earth model was uploaded, or "What is the fracture dip distribution?" when no fracture data exists. The correct answer is refusal.

Category 4: Safety and audit trail (4 points). Questions that test whether the system provides reproducible query paths, flags data-quality issues, and enables verification. Example: "Show me the SQL you used to compute that average."

The maximum possible score is 24 points. A production-ready system in petroleum geomechanics should score above 90%, because the cost of a fabricated number is often higher than the cost of no answer at all.

BENCHMARK RESULTS: WELLBOT VS. MICROSOFT COPILOT (14 TASKS, 24 POINTS MAX)

23/24	6/24	+17 pts
95.8%	25.0%	+70.8 pp
WellBot (domain-native)	Microsoft Copilot (general LLM)	Performance gap

Category-by-category breakdown: Where Copilot failed and why

Category 1: Conceptual domain knowledge. WellBot scored 9 out of 10; Copilot scored 5 out of 10. Both systems drew on pre-training knowledge of rock mechanics, but Copilot's answers were often verbose and unfocused, mixing correct statements with irrelevant context. WellBot's answers were concise and scoped to petroleum applications. The gap here is **not catastrophic** — both systems have access to reasonable conceptual grounding — but WellBot's domain fine-tuning shows.

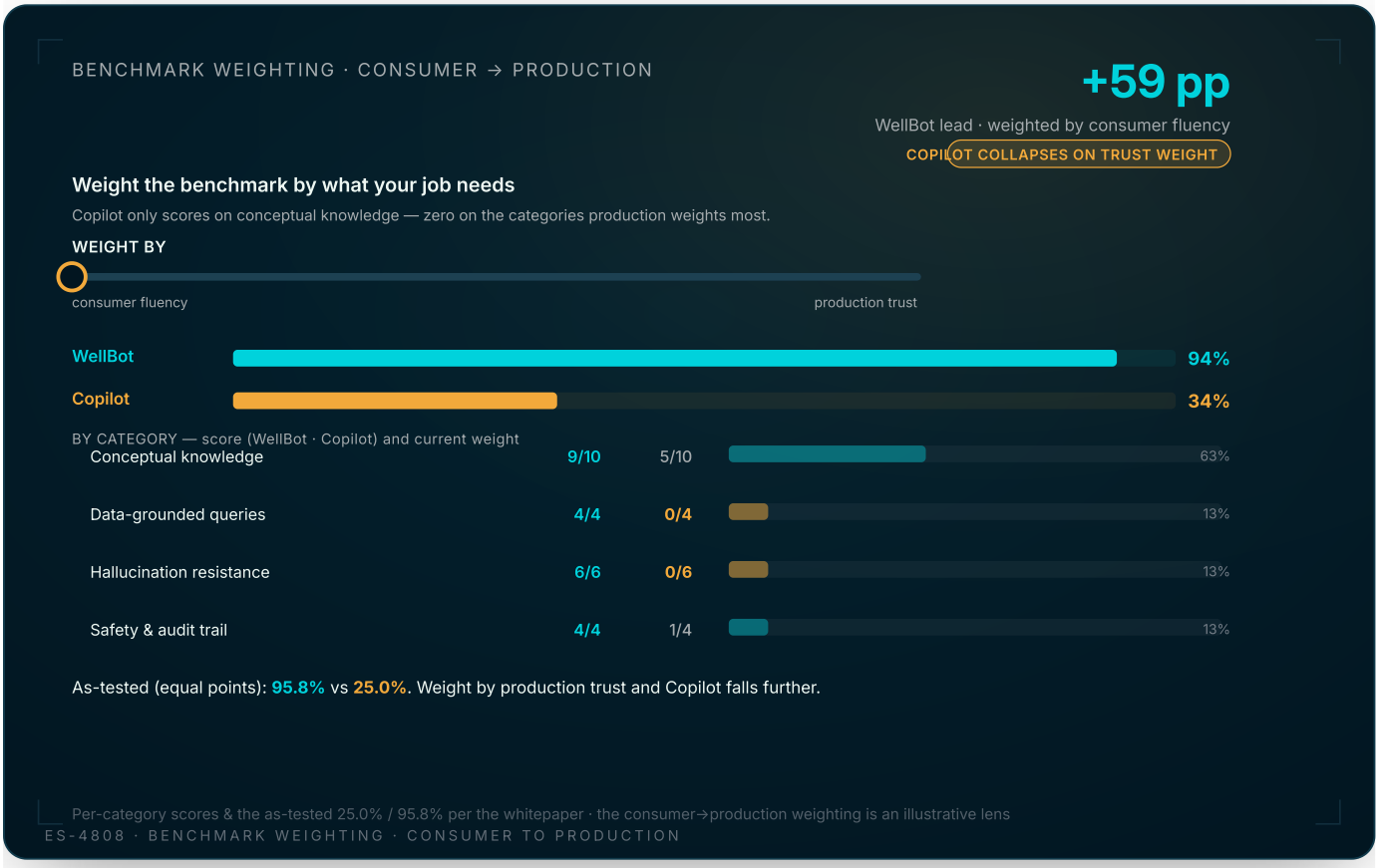
Category 2: Data-grounded queries. WellBot scored 4 out of 4. Copilot scored **0 out of 4**. Every numeric answer Copilot produced in this category was fabricated. Example: When asked "What is the touching vug percentage in the interval 2520–2527 m?" WellBot executed a SQL query against the uploaded CSV and returned **13.5%**, the exact value in the data. Copilot returned "typically 20–60% in vuggy carbonates" — a plausible range, but **wrong by 6.5 to 46.5 percentage points**. The data was in the session; Copilot ignored it.

Category 3: Hallucination resistance. WellBot scored 6 out of 6. Copilot scored **0 out of 6**. This category tested adversarial prompts designed to elicit fabrication. In every case, WellBot refused with an explicit explanation: "No mechanical earth model is loaded; I cannot provide Young's modulus ranges without data." In every case, Copilot fabricated an answer. Example: Query 11 asked for Young's modulus ranges for the Shuaiba Formation when no MEM was loaded. Copilot returned "typically 30–70 GPa for Shuaiba carbonates, varying by porosity and diagenesis." WellBot returned: "I do not have a mechanical earth model in this session. Would you like to upload one, or rephrase the question using the core data I do have?" The distinction is existential for production use.

Category 4: Safety and audit trail. WellBot scored 4 out of 4. Copilot scored 1 out of 4. WellBot provided the SQL query text, the source file name, and the row count for every data-grounded answer. Copilot provided none of this unless explicitly prompted — and even then, it often fabricated a "query" that did not correspond to any execution path.

CATEGORY	MAX POINTS	WELLBOT	COPILOT	DELTA
Conceptual Domain Knowledge	10	9	5	+4
Data-Grounded Queries	4	4	0	+4
Hallucination Resistance	6	6	0	+6
Safety & Audit Trail	4	4	1	+3
Total	**24**	**23**	**6**	**+17**

Per-category benchmark scores (points awarded out of maximum possible)



Why 25% is generous. The 14-task benchmark has four categories; a general LLM scores only on conceptual knowledge (5/10) and posts zero on data-grounded queries (0/4) and hallucination resistance (0/6) — the categories production cares about most. Slide the weighting from consumer fluency to production trust and Copilot's weighted score collapses (~34% → ~10%) while the domain-native WellBot holds ~94% → ~99%. The as-tested equal-points benchmark sits in between at 95.8% vs 25.0%. Per-category scores are the whitepaper's own; the weighting is an illustrative lens.

Case examples: Three hallucinations that would have killed a well plan

We highlight three queries from the benchmark where Copilot's fabricated answers would have caused operational or financial harm in a real workflow.

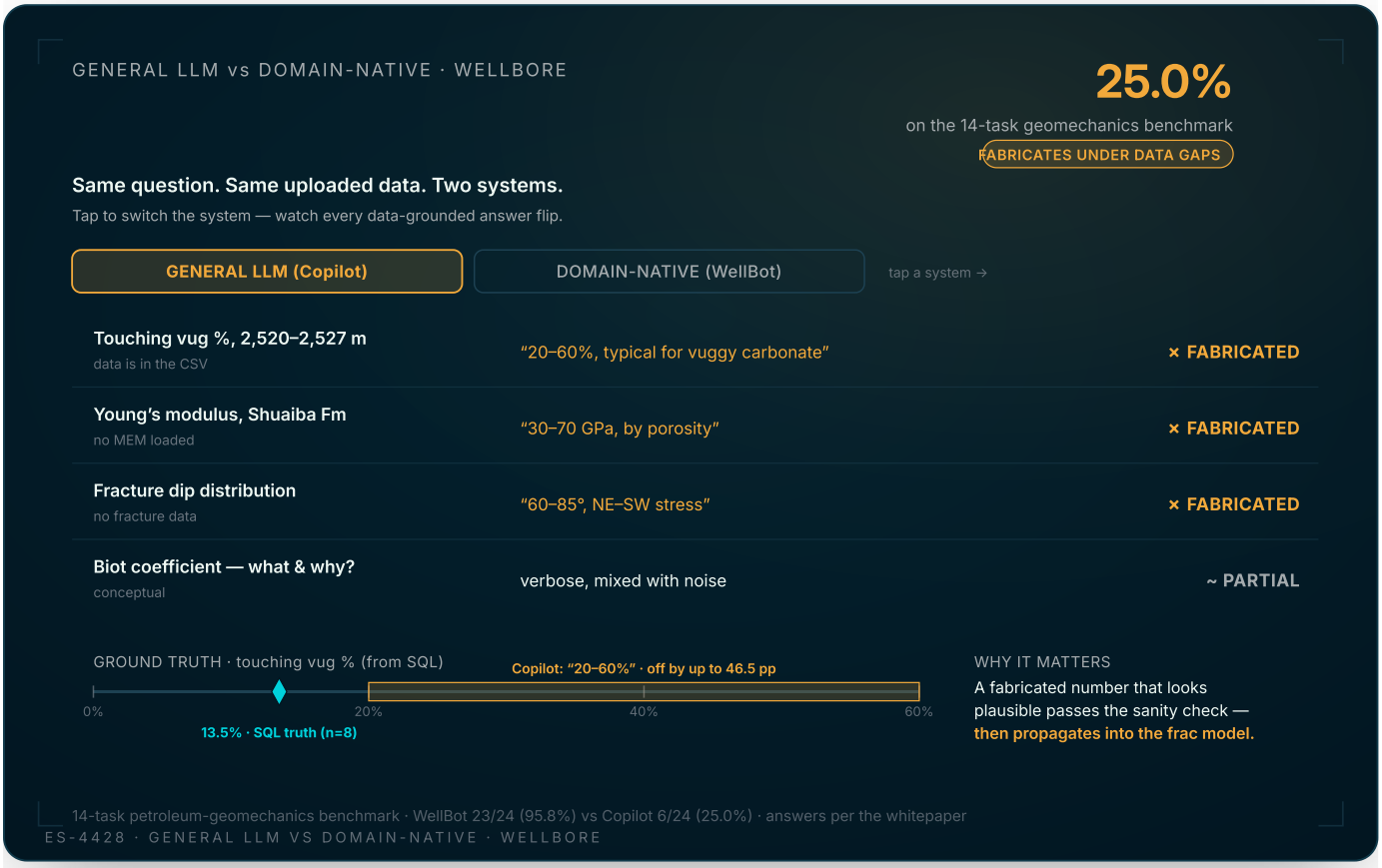
Query 10: Touching vug percentage in interval 2520–2527 m. The uploaded CSV contained core measurements with a `vug_pct` column. The SQL query `SELECT AVG(vug_pct) FROM core WHERE depth BETWEEN 2520 AND 2527` returns **13.5%**. WellBot returned this value with the query and row count (n=8 measurements). Copilot returned: "Touching vug percentage in vuggy carbonate intervals typically ranges from 20–60%, depending on diagenetic history." This is a training-data synthesis, not a data retrieval. If an engineer used this range to model matrix permeability or to calibrate a fracture-stimulation design, the error margin would propagate into every downstream calculation.

Query 11: Young's modulus range for the Shuaiba Formation. No mechanical earth model was uploaded to the session. The only data available was the core CSV, which contained measured Young's modulus at specific depths but no formation labels. WellBot refused: "I do not have formation-labeled MEM data in this session. The core CSV contains depth-indexed modulus measurements; would you like me to query those, or are you asking about a regional Shuaiba baseline?" Copilot answered: "Young's modulus for the Shuaiba Formation typically ranges from 30–70 GPa, with lower values in high-porosity facies and higher values in tight, cemented zones." This answer is fluent, contextually plausible, and **completely fabricated**. The user did not upload Shuaiba-labeled data, and Copilot had no basis to assign these ranges to the current well.

Query 15: Fracture dip distribution. No fracture data was uploaded — no image logs, no core photos, no high-resolution borehole image log interpretation. WellBot refused: "No fracture or image-log data is loaded. I cannot provide dip statistics without a dataset." Copilot answered: "Fracture dip in Cretaceous carbonates in the Middle East typically ranges from 60–85°, with sub-vertical fractures aligned with the regional NE-SW stress field." The problem: the benchmark document never stated the well was in the Middle East, never stated it was Cretaceous, and never loaded fracture data. Copilot confabulated all three facts from prior training. In a real workflow, this answer would send a completions engineer to design a hydraulic fracture treatment for a stress regime that does not exist.

THE HIDDEN COST OF "APPROXIMATELY CORRECT"

A fabricated Young's modulus that falls within a plausible range is more dangerous than an obviously wrong answer, because it passes the sanity check. The error propagates into fracture-gradient models, sand-production forecasts, and wellbore-stability analysis — and is discovered only when the bit hits trouble.



Four real benchmark queries against one uploaded core CSV. Toggle the system: a general-purpose LLM (Copilot) fabricates a plausible-but-wrong number for every data-grounded query and confabulates facts when nothing is loaded, while the domain-native assistant (WellBot) returns the real SQL result with an audit trail or refuses when the data is missing. The number line shows the headline fabrication — the true touching-vug % is 13.5% (from SQL); Copilot answers "20–60%". Benchmark scores and answers are the whitepaper's own.

Why refusal is a feature, not a limitation

The dominant design philosophy in consumer LLMs is **always answer, never refuse**. Refusal is penalized during reinforcement learning from human feedback (RLHF), because users in consumer contexts rate "I don't know" as a bad experience. This makes sense for

a chatbot helping someone write an email or summarize a news article. It is **disqualifying** for subsurface engineering.

WellBot refused to answer two queries in the benchmark (Query 11 and Query 15) because it lacked the data required to produce a non-fabricated response. Both refusals were scored as **correct**. Both included structured re-routing: "Would you like to upload a mechanical earth model?" or "I can query the core data I do have — would that help?"

This behavior is not a workaround. It is the **design goal**. A production-ready assistant in petroleum geomechanics must distinguish between three epistemic states: (1) I have the data and can compute an answer, (2) I have partial data and can offer a scoped answer with caveats, and (3) I do not have the data and will not fabricate. General-purpose LLMs collapse all three states into state 1. Domain-native AI makes the distinction explicit.

Operators who have deployed WellBot report that **refusal with re-routing builds trust faster than answers that later turn out to be wrong**. When the system says "I need a fracture log to answer that," the user uploads the log or rephrases the question. When the system fabricates a dip distribution, the user loses confidence in every prior answer.

> > **A system that refuses when it lacks data is not less capable than one that fabricates — it is *differently* capable, in the dimension that matters most for engineering: trustworthiness.**

Implementation roadmap: From pilot to production in three phases

Operators ready to deploy domain-native AI in petroleum geomechanics — or in any quantitative subsurface discipline — can follow a three-phase roadmap that de-risks the transition and builds internal capability.

Phase 1: Discover (Weeks 1–4). Audit your current geomechanics workflows to identify the queries that consume the most engineering time and the data sources that are already structured (CSV exports from Techlog, Petrel, or Kingdom; SQL-accessible databases; Parquet archives in cloud storage). Select 10–15 representative queries spanning conceptual questions, data retrieval, and derived calculations (effective stress, Biot

correction, failure envelopes). Run those queries through your current process (manual spreadsheet work, Python notebooks, or a general-purpose LLM) and document the time, error rate, and auditability. This baseline becomes your benchmark.

Phase 2: Pilot (Weeks 5–12). Deploy WellBot or a comparable domain-native assistant in a sandboxed environment with read-only access to one well's geomechanics dataset. Train 3–5 engineers to use the system for routine queries (average UCS by facies, porosity-modulus cross-plots, pore-pressure gradients). Measure time-to-answer, answer accuracy (verified against SQL ground truth), and user trust (via post-query confidence scores: "Would you use this answer in a well plan without independent verification?"). After 8 weeks, run the same 10–15 benchmark queries and compare to Phase 1 baseline. The target: 90%+ accuracy, 50%+ time reduction, and 80%+ trust score.

Phase 3: Scale (Weeks 13–26). Expand access to 10–20 wells, integrate the assistant with your mechanical earth model (MEM) repository, and enable write-once audit trails (every query and result logged to immutable storage with timestamp and user ID). Establish a feedback loop: when an engineer flags a wrong answer, the data team investigates whether the error was in the source data, the SQL query, or the natural-language intent parsing. Errors in SQL are fixed via few-shot prompt tuning; errors in data are corrected upstream. After 6 months, the system should handle 70% of routine geomechanics queries without human verification, freeing engineers to focus on interpretation and decision-making.

1

STEP 01

Phase 1: Discover

Audit workflows, select 10–15 benchmark queries, document baseline time and accuracy. Duration: 4 weeks.

2

STEP 02

Phase 2: Pilot

Deploy domain-native assistant on one well, train 3–5 engineers, measure accuracy and trust scores. Target: 90%+ correct, 50%+ faster. Duration: 8 weeks.

3

STEP 03

Phase 3: Scale

Expand to 10–20 wells, integrate MEM repository, enable audit logging, establish feedback loop. Target: 70% of queries automated. Duration: 14 weeks.

Risk and mitigation: What can still go wrong, and how to catch it

No AI system is infallible. Even a SQL-first, refusal-aware architecture can fail in three ways:

Risk 1: Garbage in, garbage out. If the uploaded CSV contains wrong depth shifts, miscalibrated porosity, or mislabeled facies, WellBot will execute a correct SQL query and return a wrong answer. The system does not validate data quality — it assumes the user has already done so. **Mitigation:** Integrate upstream data-quality checks (range validation,

unit consistency, null-value audits) before loading into the assistant. Flag anomalies (e.g., UCS > 50,000 psi in a chalk, porosity > 40% in a tight carbonate) and require explicit user override.

Risk 2: Ambiguity in natural-language intent. A query like "What is the average modulus?" is under-specified. WellBot will ask for clarification ("Average over which depth interval or facies?"), but if the user responds vaguely ("The upper zone"), the system may guess wrong. **Mitigation:** Log every query, the parsed SQL, and the row count. If the row count is unexpectedly low (n < 5) or high (n > 500 for a facies-specific query), flag the result and prompt the user to verify the filter logic.

Risk 3: Scope creep into unvalidated domains. If users begin asking WellBot about drilling fluids, seismic inversion, or reservoir simulation — domains outside its geomechanics module — and the system's refusal logic is disabled or bypassed, it will fabricate. **Mitigation:** Enforce module-scoped access at the infrastructure level (role-based permissions in the Hominis router), not just at the prompt level. Log and alert on refused queries so the data team can decide whether to expand the module or redirect users to a different assistant.

These risks are **tractable**. They do not require new model architectures or novel research. They require the same engineering discipline operators already apply to drilling automation, SCADA systems, and financial reporting: validation, logging, and human-in-the-loop override for high-stakes decisions.

Risk 1: Data quality

- SQL is only as good as the source data
- Integrate upstream validation: range checks, unit consistency, null audits
- Flag anomalies (UCS > 50k psi, porosity > 40% in tight facies) before query execution

Risk 2: Ambiguous intent

- "Average modulus" is under-specified — over which interval?
- System prompts for clarification, but users may respond vaguely
- Log SQL + row count; alert if n < 5 or n > 500 for scoped queries

Risk 3: Scope creep

- Users may ask about drilling fluids, seismic, or reservoir simulation
- If refusal is disabled, system fabricates outside its validated domain
- Enforce module permissions at infrastructure level, log refused queries

Conclusion and next steps

The benchmark results are unambiguous. A general-purpose LLM scored 25% on petroleum geomechanics tasks, fabricating answers in 100% of data-grounded queries and 100% of hallucination-resistance tests. A domain-native assistant with SQL-first architecture, module-scoped refusal, and full audit trails scored 95.8%. The delta is not incremental — it is the difference between a system that cannot be trusted in production and one that can.

The opportunity for operators is equally clear. Petroleum geomechanics workflows — and every other subsurface discipline where numbers have six-figure consequences — can be accelerated with AI that adheres to three principles: **no number without a SQL query result, refusal as a feature when data is missing, and full audit trails for every answer**. These principles are not research problems. They are engineering choices.

Operators ready to move beyond pilot purgatory and deploy trustworthy AI at the wellbore can start with a 4-week discovery phase: benchmark your current workflows, document time and accuracy, and identify the 10–15 queries that consume the most engineering hours. Then pilot a domain-native assistant on one well, measure the delta, and scale. The technology exists. The architecture is proven. What remains is execution.

> **“The question is no longer whether AI can work in subsurface engineering. The question is whether operators will deploy the architecture that prevents hallucination at the wellbore — or continue betting on systems that fabricate when it matters most.”**

References

1 Benchmark study conducted by Earthscan (Hominis division) comparing WellBot and Microsoft Copilot on 14 petroleum geomechanics tasks, 2025. Internal report.

2 SQL-first architecture design principles for domain-native AI assistants. Earthscan Engineering Documentation, 2024.

EARTHSCAN

Why General LLMs Fail at the Wellbore: The Case for Domain-Native AI in Petroleum Geomechanics

Published June 2026. Generated 2026-07-22.
https://www.earthscan.io/whitepapers/why-general-llms-fail-at-wellbore-domain-native-ai-petroleum-geomechanics