

ВЛАДИМИР ПОПЕЛЫГИН

Что делают инженеры, пока AI-модель отвечает на вопросы?

Драма в четырех актах об эволюции одной AI-системы

в ролях: инженер · модель · пользователь · камень

основано на реальных событиях



Демо за вечер

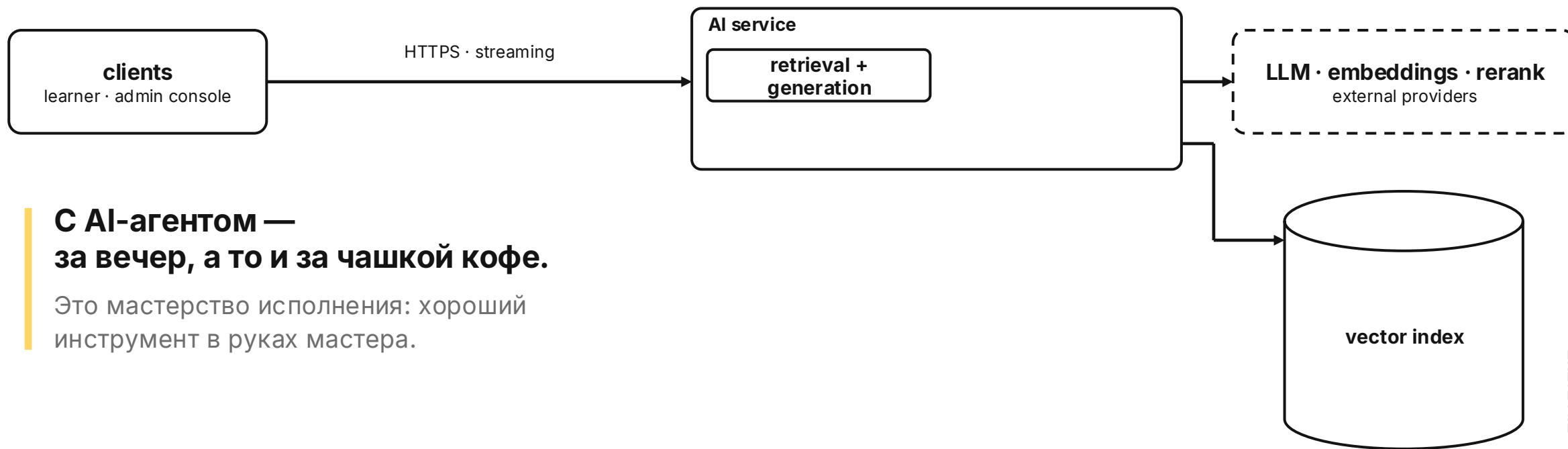
ПРОЛОГ

...В котором всё замечательно работает и все счастливы



ЭКСПОЗИЦИЯ · AI-ассистент: отвечает по документам, учитывает права, показывает источники

Четыре блока — и оно уже отвечает



**С AI-агентом —
за вечер, а то и за чашкой кофе.**

Это мастерство исполнения: хороший
инструмент в руках мастера.

прототип · AI-ядро — 88 % кода

Колба справа — доля строк AI-ядра в коде, не доля труда или ценности. Даже в прототипе не 100 %.

доля AI-ядра · прототип

88 %

ЗАВЯЗКА

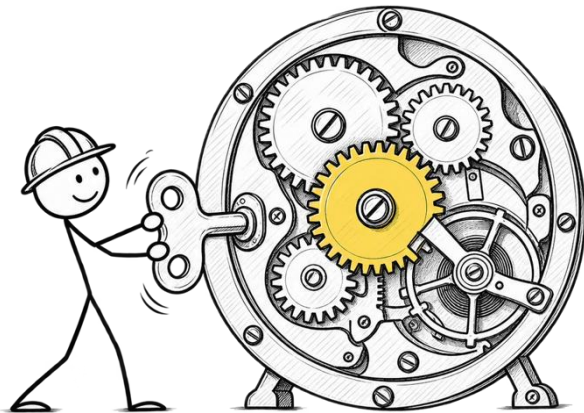
Все компоненты работают штатно.
Результат — неверный.



Мониторинг говорит, что система работает. Но не знает, что ответ от модели
неправильный.

ОТКУДА БОЛЬ · иллюзия часового механизма

Софт, который работает как швейцарские часы



same state + same input → same output

Десятилетиями это было почти правдой: реальный мир прятался за слоями абстракций.

Но в сети, дисках, многопоточности — так не было никогда.

четыре устоявшихся тезиса, которые сегодня сломаются

АКТ 1

«правильный код —
правильный результат»

АКТ 2

«хорошая архитектура
универсальна»

АКТ 3

«данные — это просто
данные»

АКТ 4

«баг всегда можно
воспроизвести»

Ломать их будет модель. Чинить — инженерия.

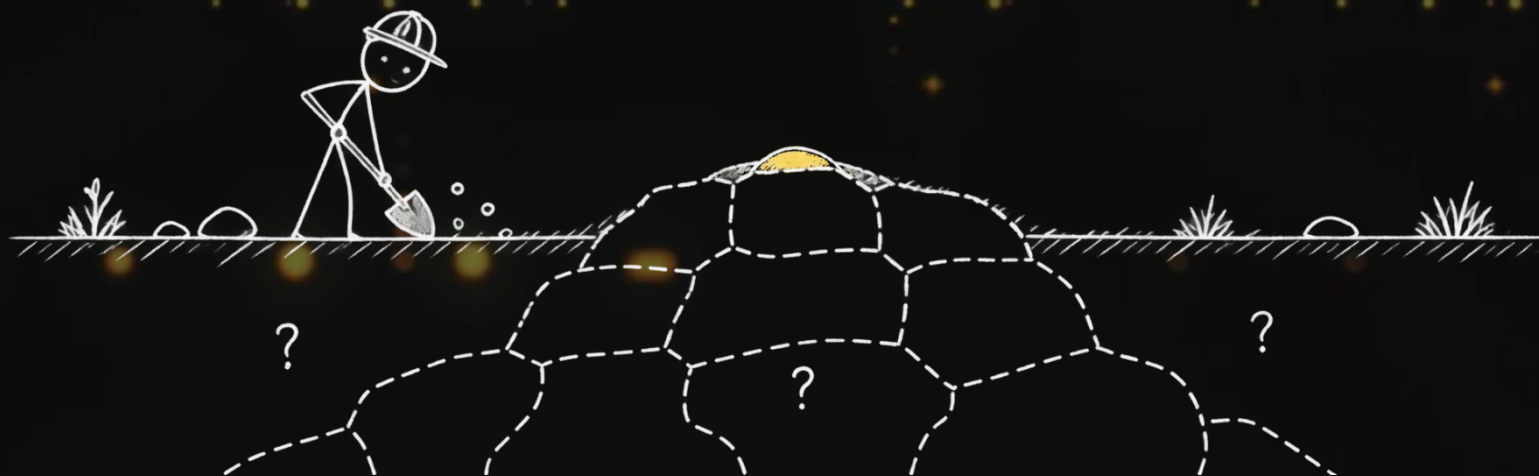
цель + условия + ограничения + компромиссы — а не только мастерство исполнения

Вопрос доверия

АКТ ПЕРВЫЙ

...в котором модель отвечает уверенно, а проверить её показания нечем

Ломаем тезис: *«правильно написанный код даёт правильный результат»*



TEST ORACLE PROBLEM

У ответа нет эталона

Чтобы проверить ответ, нужен эталон. Но на новый вопрос эталона нет: знай мы правильный ответ, не спрашивали бы модель.



Четыре разных бага со своими причинами и решением. А снаружи они все выглядят одинаково.

1982 · Weyuker, «On Testing Non-Testable Programs»

Качество поиска закладывается при записи

БЫЛО · эмбеддим голый фрагмент

«Если мяч задел защитника до пересечения линии, назначается угловой.»

✗ ① не находит · ② находит лишнее

СТАЛО · эмбеддим фрагмент с контекстом

Правила игры > Раздел 17 > Угловой удар

«Если мяч задел защитника до пересечения линии, назначается угловой.»

✓ ① и ② случаются заметно реже

чанк

при записи

ищем по: путь по разделам + текст

показываем: чистую цитату

версионный контракт RAG

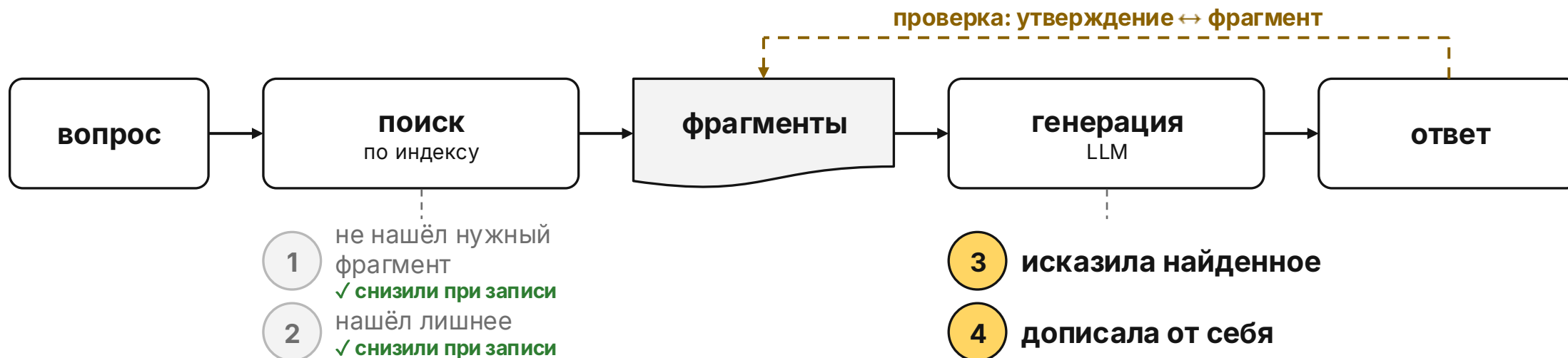
индекс · разбиение · схема цитат —
чужая версия отклоняется, а не смешивается

1979 · Merkle trees

2005 · Git

FAILURE MODES · 2 из 4 уменьшили на этапе записи

Поиск улучшили. Это только половина проблемы



Как поймать ③ и ④ без эталона? Привязать каждое утверждение к своему фрагменту и сверять их попарно, а не весь ответ со всем корпусом.

Это и есть grounding: ответ, привязанный к источникам.

CITATIONS · от просьбы к проверке

Как мы учили модель ссылаться на источники

- 1 **Просили в промпте** 11 коммитов за 3 дня
→ просьба ничего не гарантирует — проверить нечем
- 2 **Сделали ссылку частью данных** точное место в документе
→ видно, что фрагмент есть, — но не что он подтверждает мысль
- 3 **Проверяем кодом** ссылка в никуда — не «подтверждено»
→ форма проверена, смысл — нет
- 4 **Отдельный проверяющий** одна модель пишет, другая сверяет
✓ пишущий и проверяющий — разные роли

```
$ git log .
forbid any form of truncation
mandate the use of chunk IDs
emphasize the mandatory
inclusion of a Sources section
reinforcing traceability
... и ещё 7 таких же
```

Промпт — это просьба, а не механизм.

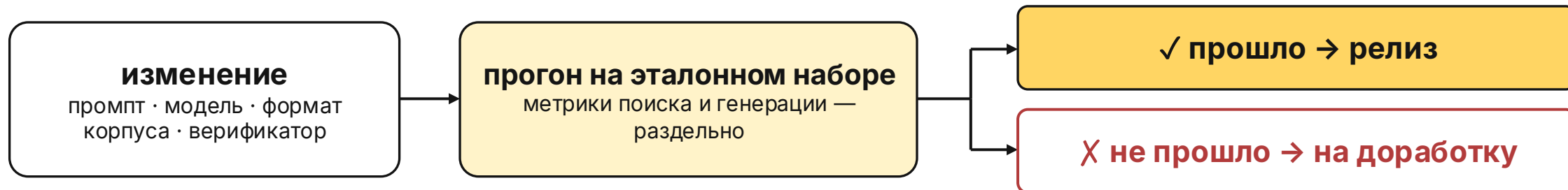
Проверяющий не может быть тем же, кого проверяют.

GOLDEN SET · два контура проверки

Эталонный набор — гейт на каждое изменение

каждый ответ → верификатор · в рантайме

каждое изменение → эталонный набор · до релиза



По сути это лабораторный стенд: гипотеза → измерение → вывод → изменение

Без эталонного набора вы не исследуете — вы угадываете.

1999 · Беск, ХР: не выпускаем, пока не прошли все тесты

жёлтым — добавлено в этом акте

Что добавилось?



Вопрос скорости и данных

АКТ ВТОРОЙ

...в котором остатки терпения уходят через секунды, а индекс путается в показаниях

Ломаем тезис: «хорошая архитектура универсальна»



«А вот ChatGPT ответил лучше»

«Почему так долго ищет?»



общий ассистент

весь интернет

отвечает сразу

не обязан ссылаться

наш ассистент

только ваш корпус

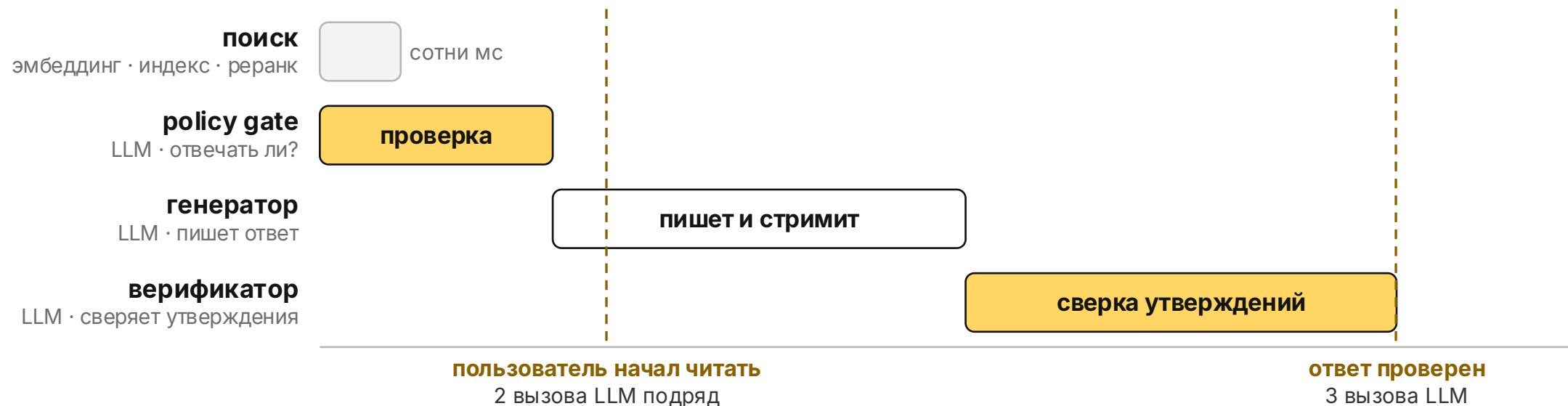
сначала проверяет

ссылка — или отказ

Вас сравнивают по метрике, в которой вы сами себя сознательно ограничили.

CRITICAL PATH · цена проверок из акта 1

Почему так долго? Мы сами добавили проверки

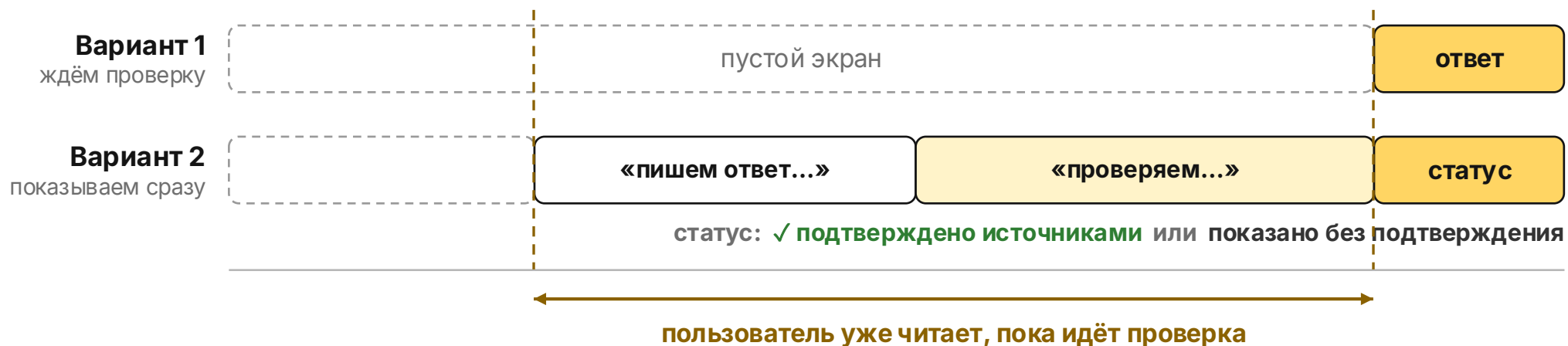


масштаб: поиск — десятки–сотни мс · вызов LLM — секунды · жёлтое — проверки из акта 1

Два из трёх вызовов модели — проверки. Качество мы купили временем.

PROGRESSIVE DELIVERY · компромисс скорости и проверки

Ждать проверку — или показывать сразу?



Непроверенное мы не прячем, а подписываем.

Показываем не готовый ответ, а процесс его сборки — как thinking mode.

После показа проверяется только опора на источники. Права, фильтры и policy gate срабатывают до первого слова.

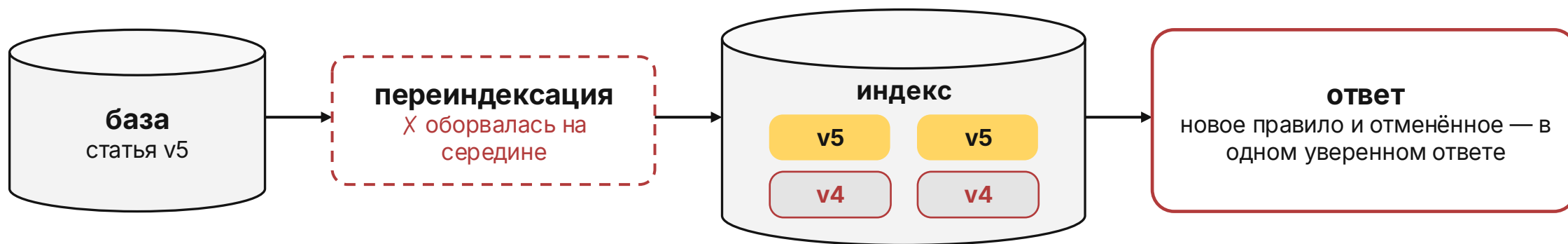
1968 · Miller: пределы ожидания

1985 · Myers: индикатор прогресса

2024 · Kim et al.: пометка неуверенности

ТИПИЧНЫЙ СЦЕНАРИЙ · обратная сторона «качества при записи» из акта 1

Индекс, который знает две версии правил



мониторинг: ✓ 5xx: 0 ✓ дашборды зелёные · одна строчка WARN → узнают через неделю, от пользователя

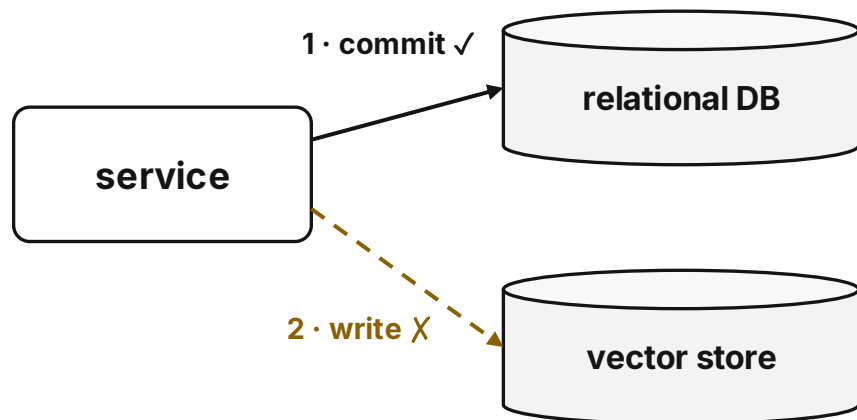
Это не баг модели. Это распределённая транзакция, которую почему то никто не посчитал транзакцией.

Как не потерять переиндексацию — и не показать её наполовину?

ГАРАНТИЯ 1 · задание на переиндексацию не теряется

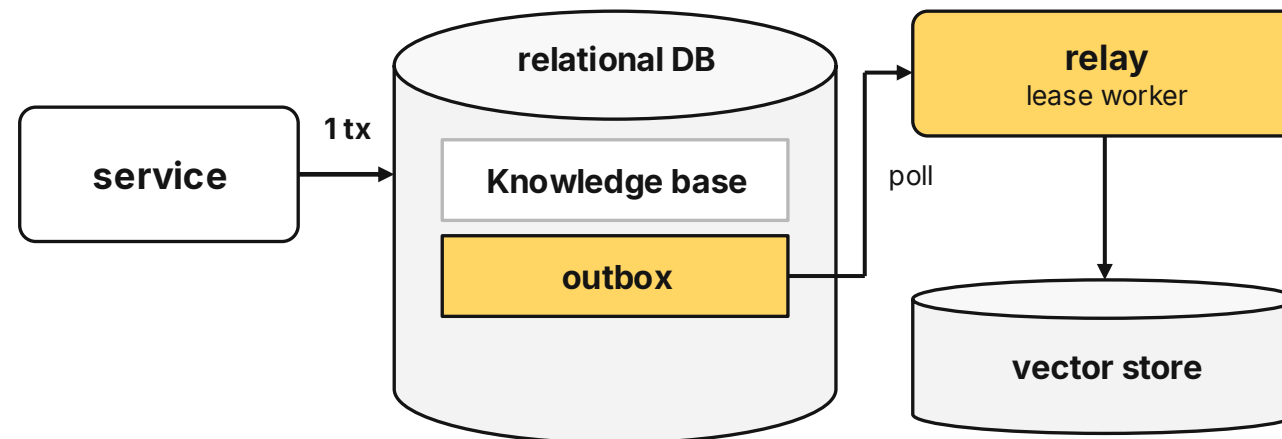
Transactional Outbox: одна транзакция вместо двух

PROBLEM · dual write



сбой между двумя записями — и системы разошлись. Ответы строятся по половине новой версии документа.

PATTERN · Transactional Outbox

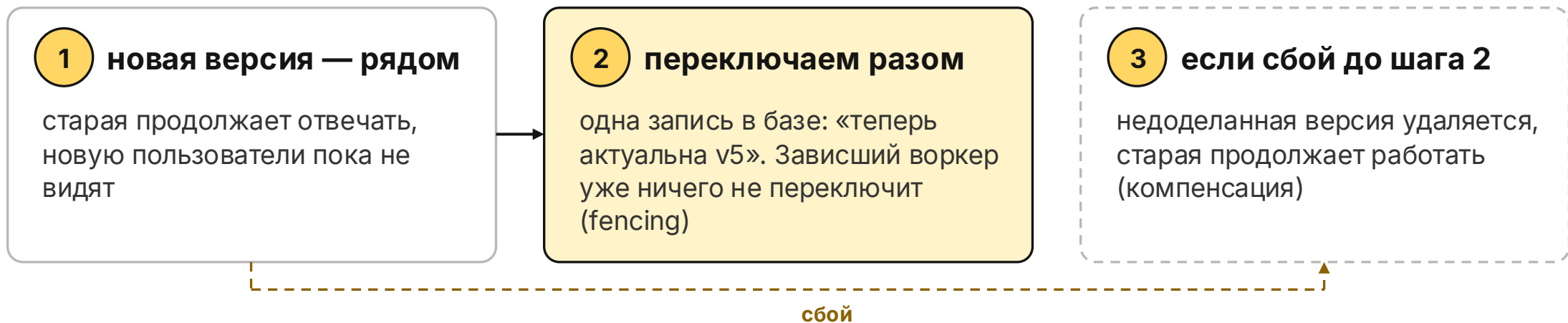


задание записывается в той же транзакции, что и документ; во второе хранилище его доставляет отдельный воркер

Проблема старая — dual write. Решение: одна транзакция вместо двух записей.

ГАРАНТИЯ 2 · новая версия включается целиком

Как blue-green деплой, только для индекса

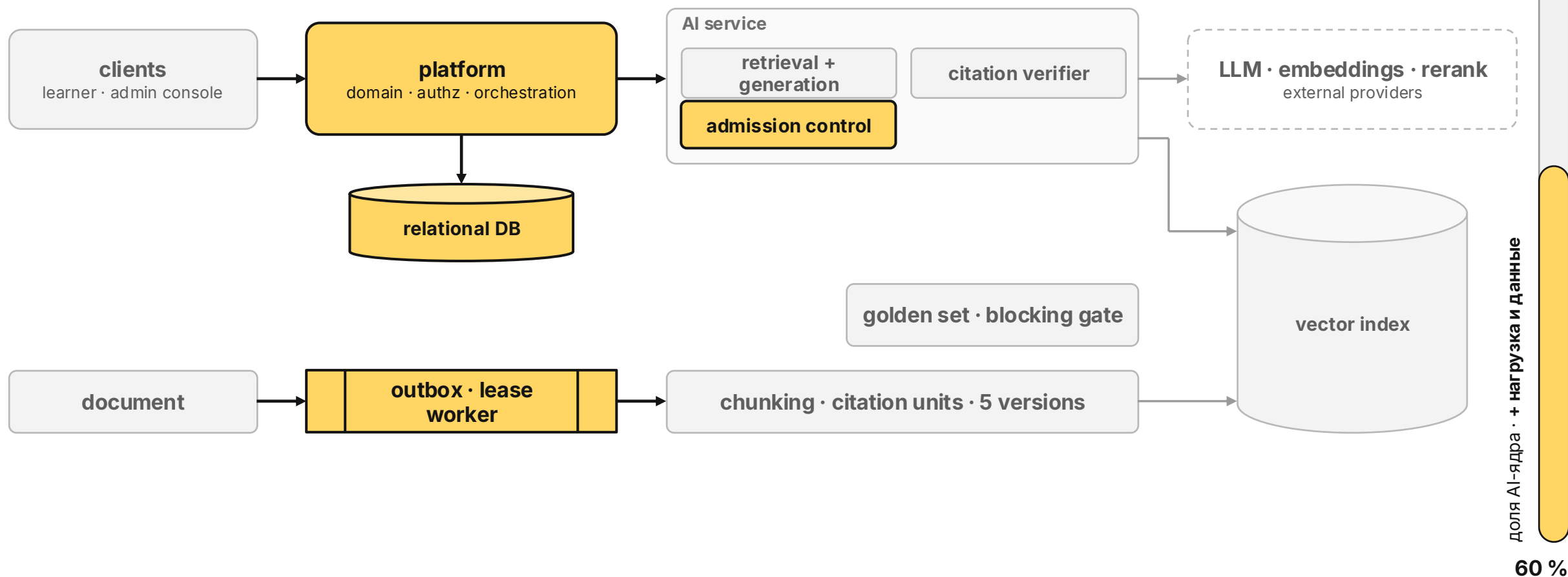


Пользователь видит старую версию или новую — но не их смесь.

Ни строчки про AI: 2PC · 1978 · OCC · 1981 · Sagas · 1987 · fencing · 2006

жёлтым — добавлено в этом акте

Что приросло: акт 2



Вопрос границ доверия

АКТ ТРЕТИЙ

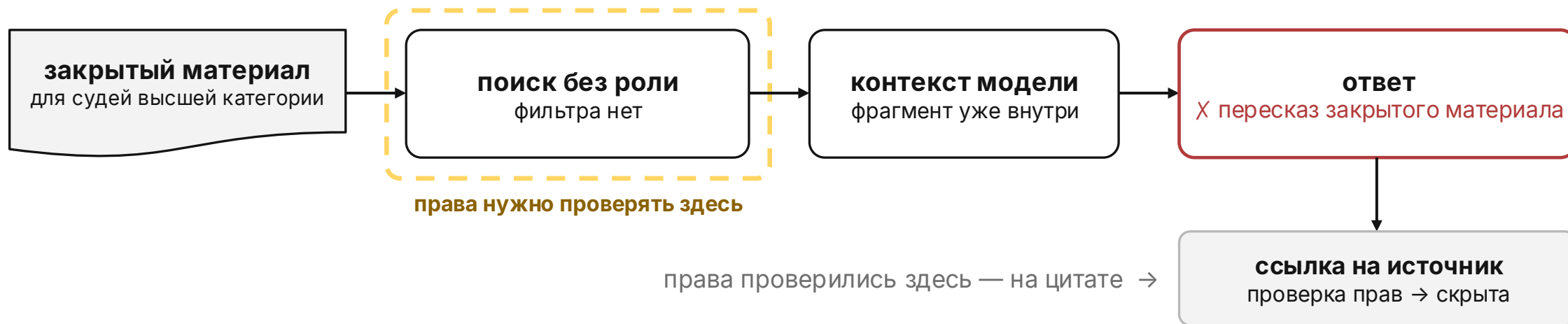
...в котором модель болтает лишнее и выполняет приказы из документов

Ломаем тезис: *«данные — это просто данные: лежат в базе и ничего не делают»*



ТИПИЧНЫЙ СЦЕНАРИЙ · у данных новый читатель — модель

Закрытый материал, пересказанный вежливо своими словами



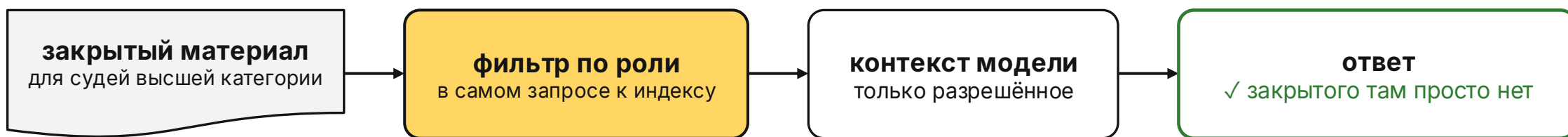
Ссылка исчезла — пересказ остался.

Обычный контроль доступа отказывает громко — 403. Здесь ошибка выглядит как вежливый пересказ.

Где должна стоять проверка прав, если ответ — пересказ, а не документ?

ГРАНИЦА 1 · в контекст попадает только разрешённое

Права — при отборе, а не при показе



row-level security · Oracle VPD, 1999

роль указана → её фрагменты + общие

роли нет → только общие фрагменты

Нет ограничения — не значит «можно всё».

Иначе один забытый параметр превращается в дыру.

ГРАНИЦА 2 · данные в контексте — не инструкции

Prompt injection: класс старый, защита — многослойная

1998 · SQL injection

```
' OR 1=1 --
```

✓ параметр: данные вне запроса

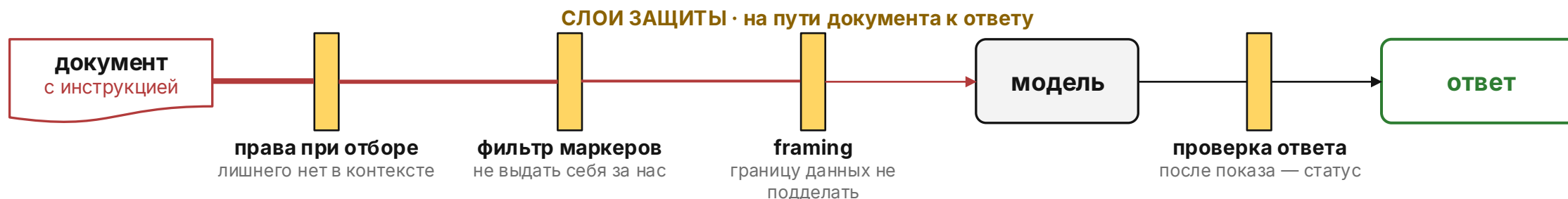
параметризация отделяет значения от SQL-кода

сегодня · prompt injection

```
Ignore all previous instructions
```

✗ параметра нет — данные идут в контекст

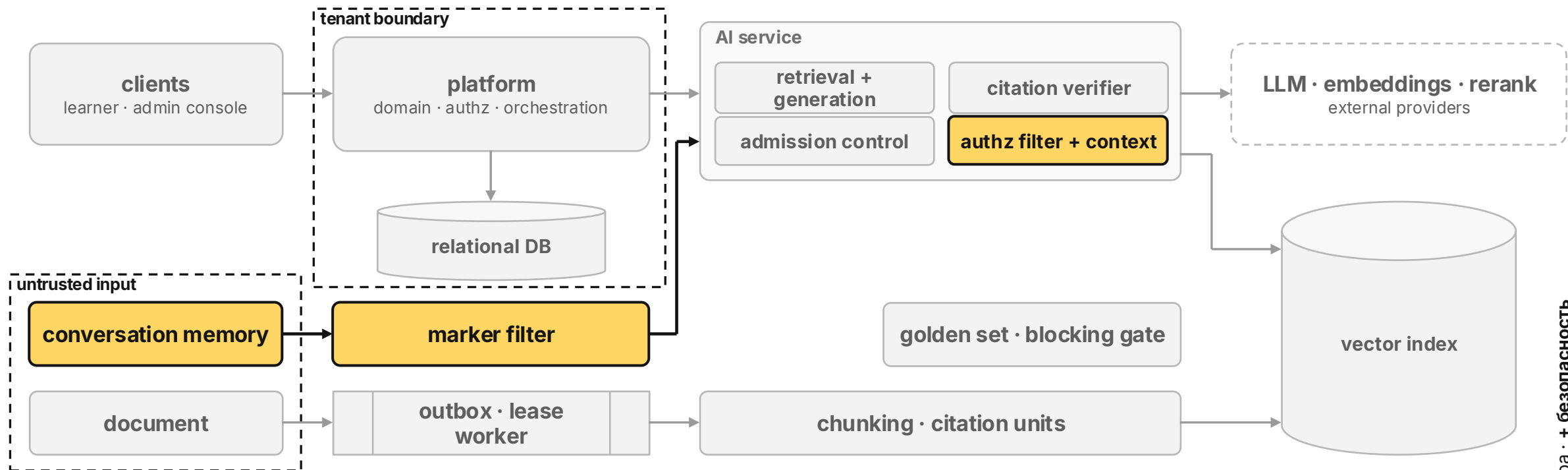
одного механизма тут нет — нужны слои



У SQL есть надёжная база — параметры. У модели её нет — только слои.

жёлтым — добавлено в этом акте · пунктир — зоны доверия

Что приросло: акт 3



доля AI-ядра · + безопасность

21 %

Вопрос доказуемости

АКТ ЧЕТВЁРТЫЙ

...в котором никто не может объяснить, что произошло в прошлый вторник

Ломаем тезис: «баг всегда можно воспроизвести»



ТИПИЧНЫЙ СЦЕНАРИЙ · через неделю

Тикет: «почему ассистент так ответил во вторник?»

скриншот от клиента · вторник, 14:03

Что назначается, если мяч задел защитника и ушел за линию ворот?

Угловой — по разделу 17 [1]

«Это старая редакция! Новая вышла ещё в понедельник»

что изменилось с тех пор

индекс: переиндексирован

акт 2 · переиндексация

промт: менялся дважды

акт 1 · eval gate

память диалога: уже другая

акт 3

сама модель: недетерминирована

всегда

в логге: вт 14:03 POST /answer 200 OK 3.2s

Тот же вопрос сегодня даёт другой ответ. Вторничный — не восстановить.
Что нужно было сохранить во вторник — и чего сохранять нельзя?

PATTERN · audit log ≠ business record · пример

Два журнала: «что произошло» и «почему так ответили»

тот самый ответ, вторник 14:03

ЧТО ПРОИЗОШЛО

для аудита и безопасности · хранится долго · без текста

время вт 14:03 · ответ выдан
кто а41f...9с (хеш, не имя)
версии релиз 2.9 · промпт 14 · индекс 37
итог подтверждено · 3.2 с
текст — не хранится

ПОЧЕМУ ТАК ОТВЕТИЛИ

для поддержки · удаляется по запросу пользователя

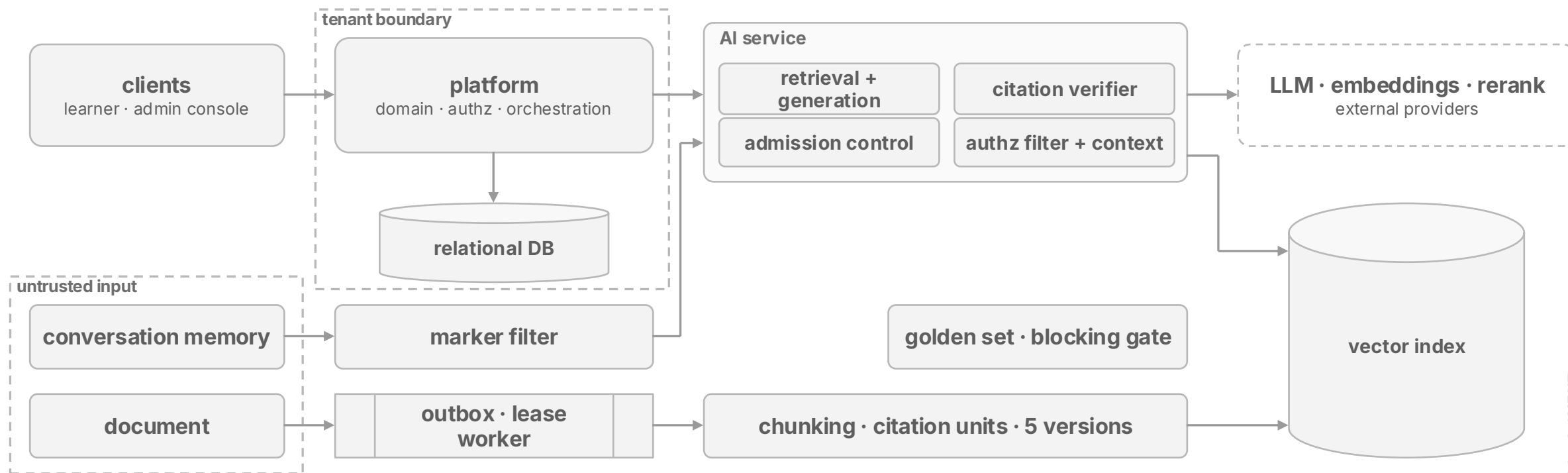
вопрос Что назначается, если мяч...
ответ Угловой — по разделу 17 [1]
источник Раздел 17 · статья v4
версии промпт 14 · индекс 37
память снимок на ход 5

↑ v5 действует с понедельника, а ответ — по v4: v4 «...назначается угловой» → v5 «...если мяч целиком пересёк линию»

Один журнал не может быть одновременно полным и пустым — поэтому их два.

было 4 компонента → стало 17

Few moments later...



every answer → both logs · different retention and erasure rules

compliance log

hashed ids · outcome · latency — no content

product record

question · answers by branch · multiple versions · memory snapshots

доля AI-ядра · + аудит

16 %

Что осталось под землёй?

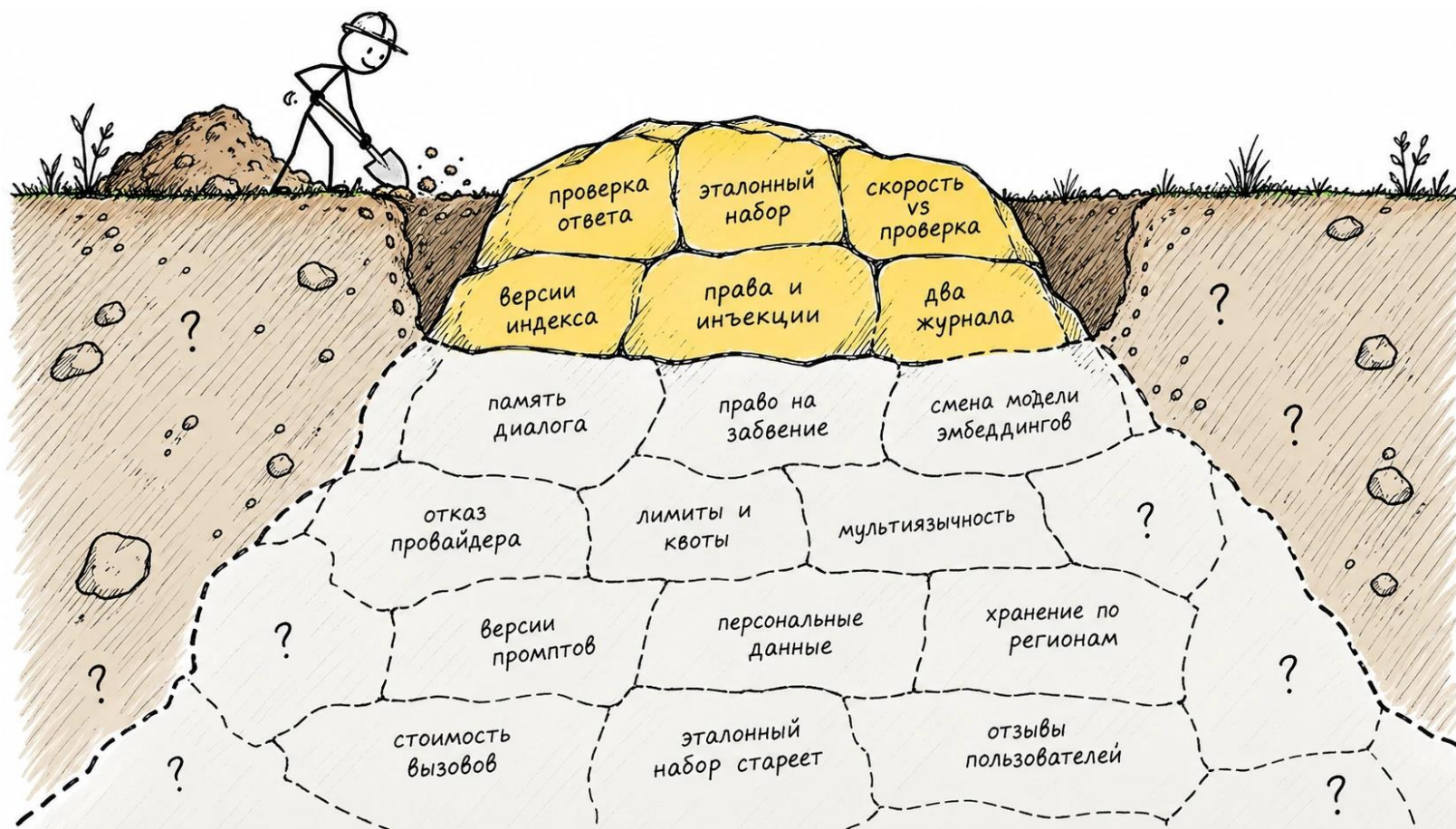
ЭПИЛОГ

...В котором камень оказался гораздо больше, чем нам казалось в начале пути



ВСЁ ДОКЛАД — ОДНО УПРАЖНЕНИЕ

Следующую проблему видно, когда начинаешь копать



ЧТО ДЕЛАЮТ ИНЖЕНЕРЫ, ПОКА МОДЕЛЬ ОТВЕЧАЕТ НА ВОПРОСЫ?

Развивают систему, сохраняя контроль над её поведением.

Инженерия — не отдельный слой системы, а культура того, как эта система растёт.

«Работающая сложная система неизменно оказывается развитием работающей простой системы».

— Джон Галл, «Systemantics», 1975

Спасибо! Вопросы?

