Inżynieria danych w epoce agentycznej AI: od potoków ETL do architektury sprawczości i ładu poznawczego w oparciu o Data Engineering with Generative n Agentic AI Justina J Leto.

AUTOR

Fundacja Dobre Państwo

STRESZCZENIE / ABSTRACT

Artykuł analizuje fundamentalną zmianę roli inżyniera danych, który z wykonawcy procesów ET staje się architektem poznawczym organizacji. W dobie agentycznej AI dane przestają być jedynie zapleczem raportowym, a stają się operacyjnym fundamentem dla autonomicznego rozumowania i systemów RAG. Autor wprowadza pojęcie AI-Augmented Data Prac tice oraz ostrzega przed zjawiskiem 'woodpushera' – osoby korzystającej z narzędzi AI bez głębokiego zrozumienia architektury. Kluczowym przesłaniem jest teza, że wiarygodność systemów AI nie wynika z zaawansowania modeli, lecz z rygoru danych, poprawnej warstwy semantycznej i ścisłego nadzoru nad cyklem życia rozwoju AI (AI-DLC), co pozwala uniknąć kosztownych halucynacji w procesach decyzyjnych.

The article analyzes the fundamental shift in the role of the data engineer, who evolves from an executor of ETL processes into a cognitive architect of the organization. In the era of agentic AI, data ceases to be merely a reporting backend and becomes the operational foundation for autonomous reasoning and RAG systems. The author introduces the concept of AI-Augmented Data Practice and warns against the 'woodpusher' phenomenon—someone using AI tools without a deep understanding of architecture. The key message is the thesis that the reliability of AI systems does not stem from the sophistication of models, but from data rigor, a correct semantic

layer, and strict oversight of the AI Development Life Cycle (AI-DLC), which allows for the avoidance of costly hallucinations in decision-making processes.

Współczesna inżynieria danych przechodzi fundamentalną transformację: z rzemieślniczego budowania potoków ETL w stronę projektowania architektury sprawczości. W dobie agentowej AI dane przestają być jedynie zapleczem raportowym, a stają się środowiskiem decyzyjnym i pamięcią instytucjonalną organizacji. Główną tezą artykułu jest uznanie, że wartość i wiarygodność systemów AI nie wynikają z efektowności samych modeli, lecz są bezpośrednią pochodną rygoru praktyk danych, ładu semantycznego i odpowiedzialności domenowej. Autor argumentuje, że w nowym paradygmacie inżynier danych musi ewoluować z operatora narzędzi w architekta poznawczego ustroju organizacji. Musi on zastąpić podejście 'woodpushera' – bezrefleksyjnego użytkownika technologii – głębokim projektowaniem warunków prawdziwości, gdzie decentralizacja odpowiedzialności (Data Mesh) i rygorystyczna kontrola jakości są jedynymi gwarantami, że autonomiczni agenci AI nie będą jedynie szybciej i bardziej przekonująco generować organizacyjne błędy.

SŁOWA KLUCZOWE

inżynieria danych w epoce agentycznej AI | architektura sprawczości | ład poznawczy | AI-Augmented Data Practice | Retrieval-Augmented Generation (RAG) | cykl życia rozwoju AI (AI-DLC) | ciemne dane | warstwa semantyczna | non-linear revenue | Model Context Protocol (MCP) | enterprise data mesh | obserwowalność systemów AI | data lineage | agentic AI

Przejście od rzemiosła ETL do architektury sprawczości

Inżynier danych po epoce potoków. Od rzemiosła ETL do architektury sprawczości.

Współczesna inżynieria danych znajduje się w punkcie przesilenia, którego nie da się uczciwie opisać językiem zwykłej modernizacji narzędzi. Nie chodzi o szybsze hurtownie, bardziej elastyczne klastry czy kolejną generację automatyzacji.

Chodzi o zmianę miejsca danych w organizacji. Przestają one być zapleczem raportowym, ukrytym w piwnicy korporacyjnej infrastruktury, gdzie specjaliści cierpliwie karmią tabele i dostarczają zarządom miesięczne podsumowania. Dane stają się środowiskiem decyzyjnym, surowcem operacyjnym i pamięcią instytucji – fundamentem automatycznego rozumowania oraz warunkiem działania agentów AI. Kto kontroluje dane, ten panuje nie tylko nad raportowaniem, ale i nad tempem uczenia się organizacji, jakością jej decyzji oraz granicami automatyzacji.

Dlatego punkt ciężkości przesuwa się z budowy potoków ku projektowaniu systemów wspierających autonomiczne lub półautonomiczne rozumowanie. Dawna inżynieria danych pytała: skąd pobrać dane, jak je przekształcić i jak obniżyć koszt zapytania. Nowa inżynieria pyta dodatkowo: jakie mają one znaczenie, kto za nie odpowiada, czy model może im ufać, a czy odpowiedź wygenerowana na ich podstawie jest audytowalna. Pyta o to, czy system nie podejmuje decyzji pozornie inteligentnej, lecz w rzeczywistości pozbawionej kontekstu.

To jest różnica między hydraulikiem danych a architektem poznawczego ustroju organizacji. Pojęcie AI-Augmented Data Practice jest więc czymś więcej niż modnym określeniem; to praktyka, w której tradycyjne fundamenty inżynierii zostają wzmocnione przez generatywną AI, systemy agentowe, RAG, inteligentne metadane i multimodalne embeddingi.

Nie jest to jednak prosta opowieść o zastąpieniu człowieka maszyną. Przeciwnie: im więcej autonomii otrzymują systemy AI, tym większe znaczenie mają ludzkie decyzje architektoniczne. Model wygeneruje kod, ale nie ustanowi hierarchii odpowiedzialności. Agent wykona zapytanie, lecz nie rozstrzygnie, czy w danym kontekście miał do tego prawo. RAG ograniczy halucynacje tylko wtedy, gdy korpus wiedzy jest poprawnie dobrany i zabezpieczony. Figura inżyniera danych ulega więc przemianie: albo awansuje, albo zostaje zdegradowana przez własne lenistwo.

Inżynier w epoce AI nie może być już wyłącznie wykonawcą transformacji, lecz musi stać się projektantem warunków prawdziwości. Musi rozumieć, że odpowiedź modelu nie jest magicznym aktem wiedzy, lecz wynikiem konkretnej architektury: od danych źródłowych i polityk dostępu, przez warstwę semantyczną i bazy wektorowe, aż po guardrails i obserwowalność.

Każda z tych warstw może wnieść porządek albo chaos. Każda może chronić organizację lub otworzyć jej drzwi włamywaczowi w eleganckim garniturze "asystenta produktywności".

Zmiana ta wiąże się z pojęciem AI-DLC, czyli cyklem życia rozwoju AI. Wiele organizacji wciąż traktuje sztuczną inteligencję jak nakładkę na istniejące procesy – trochę czatu, trochę automatycznego streszczenia. Tymczasem AI-DLC wymaga myślenia pełnym cyklem: od jakości danych i zgód na ich użycie, przez ewaluację, aż po audyt i ciągłe doskonalenie. Agent AI nie jest zwykłą "funkcją" w systemie; jest uczestnikiem procesu. Jeśli otrzyma narzędzia i dostęp do wiedzy,

staje się elementem organizacyjnej sprawczości. A sprawczości nie wdraża się jak wtyczki do przeglądarki.

Tu pojawia się pojęcie woodpushera. W szachach to ktoś, kto przesuwa figury bez strategii. W technologii jest to człowiek, który używa narzędzi, nie rozumiejąc gry. Potrafi wygenerować kod, ale nie wie, dlaczego architektura działa. Poprosi model o DAG dla Airflow, lecz nie zrozumie idempotencji czy konsekwencji błędnego harmonogramu. Stworzy zapytanie Text-to-SQL, nie wiedząc, czy jest ono bezpieczne i semantycznie poprawne. Zbuduje wektorową bazę wiedzy, nie rozumiejąc różnicy między podobieństwem kosinusowym a relacją znaczeniową istotną dla biznesu. Taki człowiek nie jest już inżynierem.

Wiarygodność systemów AI zależy od rygoru danych, a nie efektowności modeli

AI jest operatorem iluzji kompetencji. Niebezpieczeństwo woodpushera polega na tym, że AI dostarcza natychmiastowej nagrody poznawczej: model odpowiada szybko, składnie i często przekonująco, stwarzając wrażenie pełnego panowania nad złożonością. Tymczasem w systemach danych najgroźniejsze błędy nie zawsze krzyczą – często wyglądają rozsądnie. Błędny join może sztucznie zawyżyć przychód na wykresie. Źle dobrany chunking może wyciąć z dokumentu kluczowy warunek prawny. Brak maskowania PII przed wektoryzacją zamieni bazę wiedzy w kopalnię danych osobowych, a niepoprawny zakres uprawnień pozwoli agentowi udzielić odpowiedzi użytkownikowi, który nigdy nie powinien mieć dostępu do źródeł.

Halucynacja w chatbotcie bywa zabawna. Halucynacja w systemie decyzyjnym staje się już błędną rekomendacją kredytową, fałszywą oceną ryzyka lub automatycznie uruchomioną akcją biznesową, która trafia do wniosków zarządu. Dlatego nowy inżynier danych musi być sceptykiem wyposażonym w narzędzia. Nie ma być hamulcem rozwoju, lecz osobą rozumiejącą, że wiarygodność systemu nie wynika z efektowności odpowiedzi, a z jakości całego łańcucha dowodowego. Gdy model odpowiada, trzeba zapytać: na podstawie czego? Gdy agent wykonuje działanie: w czyim imieniu i w jakim zakresie uprawnień? Jeśli dashboard generuje automatyczną narrację: czy warstwa semantyczna zna język organizacji, czy tylko udaje, że go zna? A gdy system wykrywa anomalię: czy jest to zjawisko biznesowe, statystyczne, techniczne, sezonowe, czy może artefakt źle przygotowanego strumienia danych?

W tym miejscu należy odróżnić automatyzację od autonomii. Klasyczna automatyzacja wykonuje z góry zaprojektowany proces. Agent AI działa inaczej: interpretuje cel, dobiera narzędzia, wykonuje

kroki, ocenia rezultat i kontynuuje pętlę działania. AWS opisuje Amazon Bedrock AgentCore jako usługę do bezpiecznego budowania, wdrażania i operowania agentami w skali, z możliwością wykorzystania różnych frameworków. To przesuwa problem z poziomu pytania „jak napisać skrypt" na poziom projektowania środowiska, w którym skrypt, model, narzędzie, użytkownik i polityka bezpieczeństwa tworzą kontrolowaną całość. W świecie agentowym samo pytanie o technologię jest zbyt wąskie; trzeba pytać o konstytucję systemu. Ta konstytucja zaczyna się od danych. Dane są aktywem strategicznym nie dlatego, że brzmi to efektownie w prezentacji zarządczej, lecz ponieważ stanowią unikalny zapis działania konkretnej organizacji.

Każda firma może kupić podobne serwery, modele, narzędzia chmurowe i dashboardy. Nie może jednak kupić identycznej historii własnych klientów, procesów, błędów, reklamacji, transakcji, logów, rozmów, dokumentów, decyzji, wzorców sezonowych i relacji operacyjnych. Dane firmy są zlokalizowanym monopolem poznawczym. Są jak nieruchomość położona w miejscu, którego nie da się skopiować – można sprzedać podobny budynek, ale nie tę samą lokalizację.

Z tego powodu demokratyzacja danych nie oznacza anarchii dostępu. To częsty błąd. Demokracja danych nie polega na tym, że każdy widzi wszystko i eksportuje dane do arkusza, który krąży po organizacji jak elektroniczny samizdat. Oznacza ona, że właściwi ludzie i systemy mają dostęp do właściwych danych, z odpowiednim kontekstem, w odpowiednim czasie i pod kontrolą ładu organizacyjnego. Bez tego demokracja danych staje się populizmem danych: dużo głosów, mało odpowiedzialności, każdy ma własną prawdę w CSV. Amazon DataZone wpisuje się właśnie w tę logikę, służąc katalogowaniu, odkrywaniu i zarządzaniu dostępem do danych w AWS, on-premises oraz źródłach zewnętrznych. W oficjalnych materiałach AWS DataZone jest przedstawiany jako element budowy enterprise data mesh – architektury rozpraszającej własność danych między domeny biznesowe przy zachowaniu wspólnych mechanizmów ładu i kontroli. To kluczowe rozróżnienie: decentralizacja bez ładu tworzy chaos, a centralizacja bez odpowiedzialności domenowej staje się wąskim gardłem. Data Mesh próbuje uniknąć obu tych patologii.

Dawne scentralizowane jeziora danych niosły obietnicę, że zgromadzenie wszystkiego w jednym miejscu uczyni organizację mądrzejszą. W praktyce wiele z nich zamieniło się w bagna danych. Gromadzono pliki i logi, ale brakowało właścicieli, opisów, standardów jakości czy semantyki. Dane były „gdzieś", lecz nie były gotowe do użycia. To jak biblioteka, do której zwieziono milion książek, ale usunięto katalog, nazwiska autorów i światło. Teoretycznie: bogactwo wiedzy. Praktycznie: magazyn celulozy.

Jakość danych jako jedyny warunek sensu AI

W epoce AI taka sytuacja staje się jeszcze groźniejsza. Człowiek, który nie rozumie danych, może czasem zauważyć własną niewiedzę; model językowy jest mniej wstydliwy. Jeśli dostanie źle opisane, sprzeczne lub niepełne dane, często wygeneruje odpowiedź niepewną, lecz językowo elegancką. Właśnie dlatego RAG nie jest magicznym lekarstwem na halucynacje, lecz architekturą całkowicie zależną od jakości korpusu.

Retrieval-Augmented Generation działa tylko wtedy, gdy system potrafi wyszukać właściwy kontekst i przekazać go modelowi w sposób użyteczny. Jeżeli korpus jest zanieczyszczony, źle pocięty, pozbawiony metadanych albo pełen danych wrażliwych, RAG może nie tyle ograniczać halucynacje, co produkować je z większą pewnością siebie. Wniosek jest prosty: jakość AI jest pochodną jakości praktyki danych. Modele są widowiskowe, lecz dane są rozstrzygające. Czipy i energia tworzą warunki skali, ale to dane tworzą warunki sensu. Organizacja może kupić dostęp do potężnych modeli, korzystać z chmury i wdrażać narzędzia generatywne, ale bez ładu danych, warstwy semantycznej, polityk bezpieczeństwa, obserwowalności i odpowiedzialności domenowej, zbuduje jedynie technologiczną fasadę bez instytucjonalnego kręgosłupa.

A fasada pięknie wygląda do pierwszego wiatru. Rynek zmierza w stronę agentowości, jednak z istotnym zastrzeżeniem. Gartner prognozował, że do 2028 roku 33% aplikacji korporacyjnych będzie zawierało agentic AI, a 15% codziennych decyzji w pracy będą podejmować autonomicznie agenci; jednocześnie ta sama firma ostrzegała, że ponad 40% projektów agentic AI może zostać anulowanych do końca 2027 roku z powodu kosztów, niejasnej wartości biznesowej i zjawiska "agent washing", czyli marketingowego przemalowywania zwykłej automatyzacji na rzekomą agentowość. To trzeźwiące zestawienie. Mamy przed sobą realny kierunek rozwoju, ale też wielką strefę udawania. Rynek technologiczny bywa pełen obietnic bez pokrycia – często sprzedają mosiądz w chmurze.

Wartość AI zależy od ładu poznawczego i semantyki danych

Dlatego w centrum tego wywodu nie może pozostać zachwyt nad agentami, lecz pytanie o warunki ich odpowiedzialnego działania. Agent AI w organizacji potrzebuje danych, narzędzi i pamięci, ale także tożsamości, polityk dostępu, monitoringu oraz jasno określonych granic. MCP (Model Context Protocol), opisywany jako „uniwersalny port USB-C dla AI”, ma znaczenie właśnie dlatego, że porządkuje relację między agentem a zewnętrznym światem narzędzi i danych. Jednak sam

standard komunikacji nie wystarczy. Można mieć doskonały port i podłączyć do niego katastrofę. Prawdziwe pytanie brzmi: kto kontroluje zasoby widoczne dla agenta? Kto weryfikuje jego działania, rozlicza błędy i decyduje, które narzędzia może wywołać? I wreszcie: kto bada, czy odpowiedź została oparta na poprawnym kontekście?

Podobnie rzecz ma się z GenBI. Amazon Quick, zgodnie z dokumentacją AWS, jest generatywną platformą Business Intelligence, która umożliwia analizę danych, tworzenie wizualizacji i automatyzację budowania historii danych; funkcje generative BI pozwalają z kolei zadawać pytania o dane, generować podsumowania menedżerskie oraz tworzyć data stories. To fundamentalnie zmienia relację między użytkownikiem biznesowym a analityką. Dawniej menedżer prosił analityka o raport, czekał na statyczny wynik, a każde kolejne pytanie wydłużało ten cykl. W GenBI interfejsem staje się samo pytanie – użytkownik może po prostu rozmawiać z danymi. Taka rozmowa ma jednak sens tylko wtedy, gdy dane mówią językiem organizacji, a nie przypadkowym dialektem nazw kolumn. Warstwa semantyczna staje się więc jednym z kluczowych narzędzi ładu poznawczego. To ona sprawia, że pojęcia takie jak „klient aktywny", „sprzedaż netto", „retencja" czy „ryzyko churnu" nie są luźnymi słowami, lecz terminami powiązanymi z konkretnymi definicjami, regułami i źródłami. Bez warstwy semantycznej GenBI może produkować odpowiedzi szybkie, ale niekoniecznie prawdziwe. A w biznesie nie ma nic bardziej zdradliwego niż szybka bzdura podana w ładnym wykresie.

Na tym polega zasadnicza różnica między organizacją dojrzałą danowo a organizacją jedynie narzędziową. Ta druga kupuje systemy; ta pierwsza buduje praktyki. Pierwsza pyta: jaki produkt wdrożyć? Druga zaś zastanawia się, jaki problem poznawczy, decyzyjny i operacyjny chce rozwiązać, jakie dane są do tego niezbędne, kto jest ich właścicielem, jakie ryzyka niesie automatyzacja oraz jak zmierzyć skuteczność systemu i zatrzymać go w momencie awarii. Podczas gdy jedna organizacja kolekcjonuje technologie, druga tworzy instytucjonalną inteligencję.

Jeżeli pierwszym błędem organizacji w epoce sztucznej inteligencji jest traktowanie AI jak magicznej funkcji doklejonej do starego systemu, to drugim jest postrzeganie danych jako technicznego odpadu ubocznego działalności. Przedsiębiorstwa od dziesięcioleci produkują dane niemal bez przerwy: transakcje, logi, nagrania rozmów, korespondencję, skany, obrazy, zapisy sensorów czy ślady zachowań klientów. Wszystkie te sygnały razem tworzą ukrytą autobiografię organizacji.

Problem polega na tym, że większość tej autobiografii nigdy nie została przeczytana. Leży w archiwach, repozytoriach i zimnych magazynach chmurowych niczym pamiętnik instytucji, którego nikt nie potrafił otworzyć. W klasycznym modelu analitycznym wartość danych była ograniczana przez ich formę. Najłatwiej pracowało się z tym, co uporządkowane: tabelami w CRM, ERP czy

hurtowniach danych. To one stanowiły podstawę raportowania i dashboardów. Jednak taki porządek był jednocześnie filtrem poznawczym – organizacja widziała przede wszystkim to, co wcześniej zdołała ustrukturyzować. Wszystko inne pozostawało na obrzeżach.

Dane jako unikalny i strategiczny kapitał organizacji

Nie dlatego, że było bezwartościowe, lecz dlatego, że trudniej było je wydobyć, opisać, sklasyfikować i powiązać z konkretną decyzją biznesową.

Literatura trafnie nazywa ten obszar ciemnymi danymi – są to zasoby gromadzone w toku regularnej działalności, lecz faktycznie niewykorzystywane przez nieustrukturyzowaną formę, rozproszenie, brak dokumentacji lub niedojrzałość narzędzi analitycznych. Mowa tu między innymi o nagraniach z centrów obsługi, archiwach e-maili, dokumentach tekstowych, logach aplikacji, materiałach wideo i danych z czujników IoT. W epoce przed generatywną AI ciemne dane stanowiły kosztowny problem; dziś stają się strategiczną szansą, ale i ryzykiem. Szansą, ponieważ modele potrafią transkrybować, streszczać, klasyfikować, wyszukiwać semantycznie i wydobywać znaczenie z materiałów, które dawniej były niemal martwe analitycznie. Ryzykiem zaś, gdyż to samo wydobywanie może ujawnić dane osobowe, tajemnice przedsiębiorstwa, informacje prawnie chronione lub wzorce, których organizacja sama nie była świadoma.

Stara analityka przypominała patrzenie na firmę przez małe, starannie umyte okno. Widać było jedynie wycinek: sprzedaż, koszty, marże, rotację klientów, czas realizacji zamówień czy efektywność kampanii. Nowa analityka, wspierana przez AI, dąży do rozbierania ścian, by zobaczyć cały budynek: rozmowy, emocje, opóźnienia, wyjątki, skargi, dokumenty, decyzje, kontekst procesów i milczące zależności między zdarzeniami. Właśnie dlatego potrzebuje ona architekta. Bez niego organizacja nie zyskuje pełniejszego poznania, lecz większy bałagan w wysokiej rozdzielczości.

Dane są strategicznym aktywem, ponieważ stanowią zapis unikalnego doświadczenia organizacji. Można skopiować proces technologiczny, kupić podobny model AI, zatrudnić tych samych konsultantów czy wdrożyć analogiczne narzędzia AWS, ale nie da się skopiować historii działania konkretnej firmy. Dane są zatem aktywem niepowtarzalnym – posiadają lokalizację, pamięć i kontekst. Można je porównać do zlokalizowanego monopolu: żadna inna organizacja nie dysponuje dokładnie takim samym zestawem informacji. To porównanie jest niezwykle silne, gdyż pozwala wyrwać dane z abstrakcyjnego języka IT i przenieść je w sferę ekonomii politycznej.

Dane są własnością strategiczną, lecz samo ich posiadanie nie wystarcza. Trzeba umieć z nich korzystać, chronić je, interpretować i przekształcać w decyzje. Tu pojawia się pierwsza zasadnicza kategoria wartości: podejmowanie decyzji oparte na danych. Nie jest to banalne hasło z prezentacji zarządczej; w rzeczywistości chodzi o zmianę epistemologii organizacji, czyli sposobu, w jaki firma definiuje wiedzę. W organizacji niedojrzałej decyzje bywają podejmowane na podstawie hierarchii, przyzwyczajenia, intuicji lidera, presji politycznej, autorytetu działu lub ostatniego głośnego incydentu. W organizacji dojrzałej dane nie eliminują intuicji, lecz ją dyscyplinują. Nie zastępują doświadczenia, ale zmuszają je do konfrontacji z rzeczywistością. Decyzja oparta na danych nie jest zatem decyzją „bezludzką". Przeciwnie: jest to moment, w którym człowiek uznaje, że jego pamięć, emocje i status nie są wystarczającym miernikiem prawdy.

Druga kategoria wartości to generowanie przychodów z danych. W tym ujęciu dane nie służą już wyłącznie do opisu działania firmy, lecz stają się elementem produktu, usługi, modelu sprzedaży lub przewagi rynkowej. Mogą optymalizować kampanie reklamowe, personalizować ofertę, skracać czas reakcji na potrzeby klienta, wspierać rekomendacje, wykrywać ryzyko odejścia, umożliwiać tworzenie nowych usług informacyjnych lub zasilać modele AI, które same stają się produktem. Dane przestają być wówczas kosztem zaplecza – stają się kapitałem produkcyjnym. Kto ich nie używa, nie tylko traci wiedzę.

Nieliniowy wzrost przychodu wymaga zmiany roli człowieka

Traci możliwy przychód. Na tym tle zrozumiała staje się kategoria non-linear revenue, czyli przychodu nieliniowego. W tradycyjnym przedsiębiorstwie wzrost przychodów często wymaga proporcjonalnego zwiększenia zatrudnienia, liczby procesów, obsługi, administracji oraz kosztów koordynacji. W modelu agentowym ambicją jest częściowe rozerwanie tej zależności: większa skala działania nie musi oznaczać analogicznego wzrostu liczby pracowników wykonujących powtarzalne czynności. Agenci AI, GenBI, automatyczna orkiestracja i inteligentne systemy danych mają umożliwić asymetryczny wzrost produktywności.

Nie jest to jednak wizja firmy bez ludzi. To raczej opowieść o organizacji, w której ludzka praca przesuwa się w stronę projektowania, nadzoru, oceny, relacji, strategii i rozwiązywania wyjątków. Jeżeli ten ruch nie nastąpi, organizacja zyska jedynie szybszy sposób produkowania przeciętności. Wskaźnik revenue per employee staje się tu symbolem głębszego procesu. Nie chodzi wyłącznie o księgowy iloraz przychodu i liczby pracowników, lecz o pytanie, ile wartości firma potrafi wygenerować z jednostki ludzkiej uwagi, kompetencji i odpowiedzialności. W erze agentowej

człowiek nie powinien być mierzony liczbą wykonanych ręcznych operacji, lecz jakością zaprojektowanego systemu, przewidzianymi ryzykami, umożliwionymi decyzjami, usuniętymi kosztami i nowo otwartą przestrzenią działania. Jeśli AI przejmuje coraz więcej pracy operacyjnej, człowiek musi przejąć większą odpowiedzialność za sens, strukturę i ocenę tej pracy. W przeciwnym razie nie będzie liderem systemu, lecz dodatkiem biologicznym do automatu.

Ta przemiana ma także wymiar kulturowy. W wielu firmach raportowanie przez lata było rytuałem instytucjonalnym. Raport powstawał, krążył, był prezentowany i archiwizowany, a następnie zastępowany kolejnym dokumentem. W tym procesie często ważniejszy od realnego wpływu na decyzje był sam fakt istnienia raportu. Dashboardy stawały się ikonostasem zarządzania: kolorowe, regularnie odświeżane, czczone na spotkaniach, lecz nie zawsze prowadzące do konkretnych działań. GenBI uderza w ten rytuał, zamieniając raport z przedmiotu w rozmowę. Użytkownik nie musi czekać na statyczne opracowanie; może pytać, drążyć i żądać innego przekroju danych lub wyjaśnienia anomalii. Rodzi to jednak nowe niebezpieczeństwo: pozorną demokratyzację bez semantycznej odpowiedzialności. Jeśli każdy może zapytać dane o wszystko, a system odpowiada natychmiast, organizacja może pomylić szybkość z prawdą.

Mechanizmy nieufności i ładu jako warunek bezpieczeństwa AI

Natural language interface obniża próg wejścia, lecz nie znosi wymogu poprawności. Text-to-SQL jest potężnym narzędziem, ale model językowy nie „rozumie” bazy danych tak jak doświadczony analityk rozumie historię jej błędów, wyjątków i dziwnych kompromisów. Może wygenerować zapytanie formalnie poprawne, a biznesowo fałszywe; połączyć tabele, które technicznie dają się zestawić, lecz semantycznie nie powinny być łączone bez dodatkowego warunku. Może policzyć „klienta”, nie wiedząc, że w jednej domenie oznacza to płatnika, w drugiej użytkownika, w trzeciej konto, a w czwartej osobę prawną. Właśnie dlatego kluczową rolę odgrywa SQL Evaluation Agent jako strażnik bezpieczeństwa i poprawności zapytań.

Sam Text-to-SQL jest jedynie generatorem propozycji. Dopiero agent ewaluacyjny wprowadza element kontroli: weryfikuje składnię, bezpieczeństwo, sens wykonania, ryzyko halucynowanego kodu oraz niepożądane skutki zapytania. W dojrzałej architekturze nie wystarczy zatem mieć agenta, który mówi – trzeba mieć także agenta, który wątpi. Nie wystarczy system produkujący odpowiedź; niezbędny jest system potrafiący zapytać: czy ta odpowiedź powinna w ogóle powstać? Ten sceptyczny komponent staje się fundamentem nowej inżynierii danych. Inżynier nie projektuje już wyłącznie przepływu informacji, lecz projektuje mechanizmy nieufności. Nieufność nie jest tu

chorobą organizacji, lecz warunkiem jej zdrowia. System ufający wszystkiemu, co sam wygeneruje, przypomina urzędnika podpisującego każdy dokument tylko dlatego, że wyszedł z drukarki ministerstwa. Elegancki papier nie jest dowodem prawdy, podobnie jak elegancki output modelu nie jest dowodem poprawności.

Właściwie rozumiana demokracja danych wymaga trzech elementów. Po pierwsze: dostępności – użytkownicy biznesowi muszą móc znaleźć dane i użyć ich bez wielotygodniowego oczekiwania na centralny zespół analityczny. Po drugie: odpowiedzialności – dane muszą mieć właścicieli domenowych, zdefiniowaną jakość, cykl życia i reguły użycia. Po trzecie: kontroli – dostęp musi być zgodny z politykami bezpieczeństwa, klasyfikacją informacji, zasadą najmniejszych uprawnień i wymogami audytu. Brak pierwszego elementu tworzy oligarchię danych, drugiego – chaos, a trzeciego – anarchię. Żaden z tych ustrojów nie nadaje się do agentowej AI.

Dane jako aktywo różnią się od zasobów materialnych. Maszyna zużywa się w pracy; dane mogą zwiększać wartość poprzez użycie, o ile są poprawnie wzbogacane i łączone. Budynek ma jedno położenie, natomiast dane mogą być replikowane, indeksowane, wersjonowane, agregowane, anonimizowane, wektoryzowane i wykorzystywane w wielu kontekstach jednocześnie. Ta płynność czyni je jednak trudnymi do kontroli. Źle skopiowany plik może uciec z organizacji szybciej niż ciężarówka z magazynu, a niepoprawnie udostępniony zbiór może naruszyć prywatność tysięcy osób bez jednego wyłamanego zamka. Agent z nadmiernymi uprawnieniami staje się doskonałym narzędziem ataku, gdyż wygląda jak legalny użytkownik systemu. Dlatego bezpieczeństwo nie jest dodatkiem do strategii danych, lecz warunkiem jej możliwości. Notatka źródłowa nazywa to „Job Zero” – zadaniem zero. Jest to trafne, ponieważ zadanie zero poprzedza wszystkie inne; nie jest punktem na liście, lecz zasadą decydującą o tym, czy lista w ogóle powinna zostać zrealizowana.

W epoce AI bezpieczeństwo danych wykracza poza klasyczne szyfrowanie, IAM, KMS czy separację środowisk. Obejmuje ochronę przed prompt injection, wyciekiem danych wrażliwych, zatruwaniem korpusu RAG, atakami supply chain, zjawiskiem confused deputy i niekontrolowanym wydobywaniem informacji przez modele. Im inteligentniejszy system, tym bardziej wyrafinowane mogą być jego błędy. Stary system po prostu odmawiał dostępu; nowy może uprzejmie streścić cudzą tajemnicę.

Confused deputy to nie tylko techniczny atak eskalacji uprawnień, ale i trafna metafora organizacyjna. Zdezorientowany zastępca to system posiadający władzę, lecz nie rozumiejący kontekstu jej użycia. W klasycznej administracji bywa nim urzędnik mechanicznie wykonujący procedurę bez zrozumienia celu. W systemie AI takim zastępcą jest agent z dostępem do danych i narzędzi, który nie odróżnia uprawnionej prośby od manipulacji. Jeśli użytkownik skłoni go do

działania poza intencją projektanta, agent staje się posłańcem atakującego – nie dlatego, że jest zły, lecz dlatego, że jest posłuszny bez mądrości.

Ukazuje to głęboki związek między technologią a antropologią instytucji. Systemy AI dziedziczą nie tylko dane organizacji, ale i jej bałagan, skróty, niejawne praktyki oraz słabości ładu. Jeśli w firmie nikt nie wie, kto jest właścicielem danych, agent również tego nie będzie wiedział. Jeśli definicje KPI są negocjowane politycznie w kuluarach, GenBI jedynie przyspieszy konflikt interpretacyjny. Jeśli dostęp do danych przyznawano historycznie „na wszelki wypadek", agent odziedziczy nadmiarowe uprawnienia. Jeżeli dokumenty są pełne sprzecznych wersji, RAG będzie wybierał kontekst z archiwalnego cmentarza.

Od centralizacji infrastruktury do zdecentralizowanej odpowiedzialności domenowej

Nowoczesna inżynieria danych musi być zatem praktyką porządkowania organizacji. Nie wystarczy stworzyć warstwę techniczną; należy ustalić, co w danej domenie biznesowej oznacza „prawda". Trzeba odróżnić dane surowe od certyfikowanych produktów danych i precyzyjnie określić, które zbiory służą do eksploracji, które do raportowania zarządczego, które do trenowania modeli, a które nigdy nie powinny trafić do wektoryzacji. Systemy należy projektować tak, aby różnica między „mogę technicznie" a „wolno mi organizacyjnie" była wpisana w samą architekturę.

W tym miejscu ujawnia się sens Data Mesh i roli właścicieli domen danych. Centralna jednostka rzadko rozumie subtelności danych powstających w sprzedaży, logistyce, obsłudze klienta, ryzyku, HR, produkcji czy finansach. Zespół centralny może dostarczyć platformę, standardy, bezpieczeństwo i katalog, ale nie zastąpi lokalnej wiedzy. Dlatego odpowiedzialność musi być zdecentralizowana. Podejście „dane jako produkt" oznacza, że domena nie może po prostu wrzucić pliku do jeziora danych i umyć rąk. Domena publikuje zasób wraz z opisem, gwarancją jakości, definicją, właścicielem, SLA oraz politykami dostępu, mając pełną świadomość potrzeb użytkownika końcowego. Produkt danych, podobnie jak produkt rynkowy, musi mieć odbiorcę, sens i przypisaną odpowiedzialność. Prowadzi to do kluczowego rozróżnienia: centralizacja infrastruktury nie musi oznaczać centralizacji wiedzy. Wiele organizacji popełnia błąd, próbując rozwiązać problemy z danymi poprzez stworzenie jednego, ogromnego zespołu centralnego.

Taki zespół szybko staje się wąskim gardłem. Inni idą w przeciwnym kierunku: pozwalają każdej domenie działać po swojemu, co tworzy federację chaosu. Data Mesh proponuje kompromis: domenową własność danych przy zfederowanym ładzie. To model polityczny, nie tylko techniczny.

Można rzec: republika danych zamiast cesarstwa hurtowni albo plemiennej anarchii arkuszy kalkulacyjnych. Amazon DataZone nie jest więc jedynie narzędziem katalogowym, lecz elementem instytucjonalizacji danych. Pozwala tworzyć przestrzeń, w której zasoby są publikowane, odkrywane i konsumowane w sposób kontrolowany. Oświetlanie dark data nie polega na wrzuceniu wszystkiego do wyszukiwarki, lecz na nadaniu zasobom statusu, opisu i reguł użycia. Ciemne dane stają się aktywem dopiero po przejściu procesu rozpoznania; do tego momentu są jedynie nieoszacowanym złożem – mogą zawierać złoto, ale mogą też skrywać toksyczne odpady. Największe organizacje przyszłości nie będą tymi, które po prostu „mają dużo danych". Dużo danych można mieć jak dużo gratów w piwnicy: imponująca objętość, znikoma użyteczność i rosnące ryzyko pożaru.

Przewagę uzyskają organizacje potrafiące dane klasyfikować, semantyzować, zabezpieczać, udostępniać, wersjonować i łączyć je z procesami decyzyjnymi, by zasilać nimi systemy agentowe. W epoce AI nie wygrywa ten, kto posiada największy magazyn informacji, lecz ten, kto potrafi przekuć informację w sprawdzalne działanie. Inżynier danych przestaje być więc tylko specjalistą od infrastruktury, a staje się tłumaczem między światem danych a światem decyzji. Musi rozumieć, że biznes nie potrzebuje danych dla samych danych, lecz redukcji niepewności, skrócenia czasu odpowiedzi, obniżenia kosztów, wzrostu przychodów oraz wzmocnienia zdolności adaptacyjnych.

Jeżeli projekt danych nie łączy się z którymś z tych celów, może być elegancki technicznie, ale pozostaje politycznie kruchy. W organizacji budżetowej infrastruktura, której sensu nie da się przełożyć na wartość, nie przetrwa długo. Należy jednak bronić się przed prymitywnym redukowaniem wartości danych wyłącznie do natychmiastowego zysku. Nie każda korzyść jest widoczna w kwartalnym rachunku wyników. Czasem wartością jest odporność: zdolność wykrycia oszustwa, odtworzenia przebiegu decyzji, udowodnienia zgodności regulacyjnej czy uniknięcia błędnej automatyzacji. Najlepszy system danych nie zawsze zarabia spektakularnie, ale często chroni organizację przed kosztowną katastrofą. W języku zarządczym jest to trudniejsze do sprzedania, ponieważ uniknięta klęska rzadko ma własny dashboard triumfu. A jednak dojrzałość polega na docenieniu tych niewidzialnych zwycięstw.

Tu powraca figura woodpushera. Woodpusher w świecie danych powie: „model wygenerował", „agent policzył", „dashboard pokazał" czy „RAG znalazł". Dojrzały inżynier zapyta: „na jakich danych?", „z jaką definicją?", „z jakim zakresem uprawnień?", „z jakim błędem?", „z jaką możliwością audytu?" i „z jakim ryzykiem decyzji?". Różnica między nimi to różnica między popychaniem figur a grą strategiczną. Jeden wykonuje ruch, drugi rozumie planszę.

Właśnie dlatego przyszłość danych nie może być opowiadana wyłącznie językiem narzędzi. Amazon S3, Redshift, Glue, Bedrock, DataZone, QuickSight, OpenSearch, pgvector, MCP, AgentCore czy

Strands SDK mogą tworzyć potężny ekosystem, ale same w sobie nie ustanowią sensu. Sens ustanawia architektura odpowiedzialności. Technologia jest dźwignią, lecz dźwignia bez punktu podparcia służy co najwyżej do efektownego machania metalem.

Najważniejsza teza brzmi zatem następująco: dane stają się strategicznym aktywem dopiero wtedy, gdy organizacja potrafi przejść od gromadzenia do rozumienia, od raportowania do działania, od silosów do odpowiedzialnej demokracji danych, od statycznych dashboardów do kontrolowanej rozmowy z informacją oraz od ciemnych danych do certyfikowanych produktów wiedzy.

Architektura danych jako fundament epistemologii organizacji

Generatywna i agentowa AI nie znoszą braku rygoru; one go zaostrzają. Im potężniejszy model, tym mniej wolno tolerować bylejakość danych. Im bardziej autonomiczny agent, tym precyzyjniej trzeba projektować granice jego świata. Im łatwiej zapytać system o wszystko, tym pilniej musimy wiedzieć, co ten system naprawdę wie.

Architektura danych: od jeziora do mesh. Dlaczego aktywo bez ładu staje się bagnem

Jeżeli dane są strategicznym aktywem organizacji, to architektura danych jest ustrojem, który decyduje, czy zasób ten będzie pracować, gnić i przeciekać, czy mutować w źródło ryzyka. Sama obecność danych nie tworzy przewagi. Można dysponować ogromnym majątkiem informacyjnym, a jednocześnie nie posiadać żadnej realnej zdolności poznawczej. Można przechowywać petabajty plików i wciąż nie potrafić odpowiedzieć na proste pytanie zarządu: co naprawdę dzieje się z klientem, produktem, kosztem, reklamacją, ryzykiem czy procesem?

W epoce AI problem ten nie znika – on staje się ostrzejszy. Modele i agenci nie potrzebują już człowieka jako pośrednika przy każdym zapytaniu. Jeśli architektura jest wadliwa, błędna odpowiedź może powstać szybciej, ładniej i z większym autorytetem językowym niż kiedykolwiek wcześniej. Nowoczesna architektura danych nie jest więc tylko kwestią wydajności; jest kwestią epistemologii organizacyjnej: mówi, skąd firma wie to, co twierdzi, że wie.

W dawnym modelu technicznym pytano głównie o miejsce przechowywania i sposób przetwarzania danych. W modelu agentowym trzeba zapytać głębiej: które dane są źródłowe, a które przekształcone? Które są certyfikowane, a które eksperymentalne? Co wolno udostępnić agentowi, co nadaje się do RAG, a co wymaga maskowania PII? Gdzie jest właściciel domenowy, a gdzie jedynie porzucony artefakt dawnego procesu? Bez tych rozróżnień organizacja buduje nie

inteligencję, lecz informacyjny labirynt, w którym nawet Minotaur poprosiłby o katalog metadanych.

Pierwszym wielkim ruchem nowoczesnej architektury było oddzielenie danych od mocy obliczeniowej, czyli separation of storage and compute. W tradycyjnych systemach dane i obliczenia były silnie związane z konkretną bazą, silnikiem czy infrastrukturą. Taki model był wygodny, dopóki skala i zmienność obciążeń pozostawały umiarkowane. Jednak wraz ze wzrostem wolumenu danych, różnorodnością źródeł i potrzebą elastycznej analityki, stał się on ograniczeniem. Organizacje zaczęły płacić za sztywność: nadmiarową infrastrukturę, trudne migracje, zależność od jednego dostawcy (vendor lock-in) oraz brak możliwości swobodnego dopasowania silników obliczeniowych do konkretnych zadań.

Oddzielenie przechowywania od obliczeń zmienia logikę systemu. Dane stają się trwalszym aktywem niż technologia ich przetwarzania. Można je przechowywać w otwartych formatach, takich jak Parquet, organizować w strukturach lakehouse, przetwarzać przez Spark, analizować przez Redshift, indeksować semantycznie, zasilać RAG i udostępniać przez warstwy katalogowe. Jest to istotne nie tylko technicznie, ale i ekonomicznie – dane przestają być zakładnikiem jednego silnika i stają się zasobem, do którego można podłączać różne formy mocy obliczeniowej.

W świecie, w którym AI wymaga ogromnej elastyczności i okresowych, gigantycznych obciążeń, taka separacja nie jest luksusem, lecz warunkiem manewru. Przejście ku architekturze zapewniającej skalowalność i niezależność od dostawcy jest kluczowe, ponieważ sztuczna inteligencja w przedsiębiorstwie to nie jednorazowa aplikacja, lecz dynamiczny ekosystem. Dzisiaj organizacja korzysta z jednego modelu, jutro z innego; dziś buduje hurtownię, jutro bazę wektorową; dziś potrzebuje raportu, a jutro agenta wykonującego zadania. Architektura, która zamyka dane w zbyt ciasnej technologicznej klatce, odbiera organizacji przyszłą zdolność adaptacji.

Drugim ruchem było przejście od klasycznego data warehouse do data lake. Hurtownia danych była wielkim osiągnięciem epoki raportowania. Jej siłą była struktura: dane były czyszczone, transformowane i modelowane pod określone scenariusze analityczne, co zapewniało spójność i zaufanie. Ceną była jednak sztywność. Hurtownia świetnie obsługiwała dane ustrukturyzowane i przewidywalne pytania, lecz zawodziła przy ogromnej różnorodności formatów, eksploracji czy multimodalności (logi, dokumenty, obrazy). Była doskonałym narzędziem dla świata, w którym najpierw wiemy, co chcemy mierzyć; gorzej sprawdzała się tam, gdzie chcemy odkrywać to, czego jeszcze nie umiemy nazwać.

Data lake obiecywał rozwiązanie tego problemu. Miał przyjmować dane w różnych formatach, często w stanie surowym, bez natychmiastowego narzucania schematu. Zamiast schema-on-write,

właściwego hurtowniom, wprowadzono schema-on-read: dane przechowuje się elastycznie, a strukturę nadaje im dopiero podczas odczytu i analizy. Model ten lepiej odpowiada potrzebom eksploracji, uczenia maszynowego oraz pracy AI.

Jednak data lake miał swoją ciemną stronę. Jeśli wszystko można wrzucić do jeziora, łatwo wrzucić wszystko źle. Bez metadanych, właścicieli, kontroli jakości i zdefiniowanego cyklu życia, jezioro zmienia się w bagno. To „bagno danych" jest jedną z najważniejszych porażek organizacyjnych ostatniej dekady. Nie wynikało ono z braku technologii, lecz z błędnego przekonania, że centralne zgromadzenie danych samoistnie wytworzy wiedzę. To tak, jakby zsypanie książek, paragonów, akt sądowych i starych gazet do jednego magazynu tworzyło uniwersytet. Nie tworzy – tworzy jedynie magazyn. Wiedza zaczyna się tam, gdzie istnieje porządek, opis, kryteria jakości i wspólnota interpretacyjna. Organizacje często zapominały o tej prawdzie, ponieważ łatwiej było sfinansować infrastrukturę niż odpowiedzialność.

Lakehouse łączy elastyczność jeziora z rygorem hurtowni

Odpowiedzią na ograniczenia data lake stał się model lakehouse, który łączy elastyczność jeziora danych z rygorem transakcyjnym i analitycznym hurtowni. Kluczową rolę odgrywają tutaj Apache Iceberg oraz S3 Tables, umożliwiające pracę z transakcjami ACID, ewolucją schematu, podróżą w czasie oraz ewolucją partycji.

Standardy spójności i certyfikacji jako fundament zaufania AI

Sens tego rozwiązania wykracza poza techniczną wygodę. Iceberg pozwala traktować dane w jeziorze nie jako przypadkowe pliki, lecz jak zarządzane tabele z historią, spójnością i możliwością kontrolowanych zmian, co przybliża data lake do standardów korporacyjnego zaufania. Kluczową rolę odgrywają tu transakcje ACID. Atomicity gwarantuje, że operacja zapisu nie może zakończyć się połowicznie, zostawiając dane w stanie rozkroku. Consistency zapewnia przejście między poprawnymi stanami systemu. Isolation chroni współbieżne operacje przed wzajemnym psuciem wyników, a Durability gwarantuje trwałość zatwierdzonego zapisu.

Choć pojęcia te brzmią jak klasyczna informatyczna gramatyka, w epoce agentów AI stają się warunkiem odpowiedzialnej automatyzacji. Agent nie może wnioskować na podstawie tabeli

aktualizowanej w sposób niespójny. System GenBI nie powinien budować narracji z danych, których wersja zmienia się w połowie zapytania. RAG natomiast nie może korzystać z korpusu, w którym część dokumentów pochodzi z jednej wersji procesu, a część z innej, bez śladu tej różnicy. Bezcenna dla audytu i odpowiedzialności jest funkcja time travel, umożliwiająca odpytywanie historycznych stanów danych. W klasycznej organizacji pytanie „dlaczego podjęliśmy wtedy taką decyzję?" prowadzi do archeologii maili, zrzutów ekranu i starych eksportów. Dojrzała architektura pozwala natomiast odtworzyć stan danych z konkretnego momentu, co ma wymiar prawny, regulacyjny, zarządczy i poznawczy. Jeśli agent AI zarekomendował działanie na podstawie określonego korpusu, organizacja musi móc ustalić, co dokładnie widział w chwili generowania wyniku. Bez tego odpowiedzialność rozpływa się w chmurze, a chmura – mimo nazwy – nie jest miejscem, w którym powinno znikać rozliczanie decyzji.

Architektura typu medallion, obejmująca etapy landing zone, audit, raw, optimized, conform, unified analytics i published stage, wprowadza do tego procesu porządek dojrzewania danych. Dane nie rodzą się gotowe do użycia; przechodzą drogę od surowego zapisu do produktu informacyjnego. W landing zone trafiają jako materiał wejściowy, w audit stage są sprawdzane, a w raw stage zachowane w formie pierwotnej, co jest kluczowe dla odtwarzalności. Następnie w optimized stage przechodzą do formatów wydajnych analitycznie, w conform stage uzyskują wspólne typy i schematy, by w unified analytics zostać połączonymi w spójny model. Dopiero w published stage stają się certyfikowanym zasobem dla użytkowników biznesowych, modeli i agentów.

Tę strukturę można porównać do procesu awansu poznawczego. Nie każda informacja zasługuje od razu na status wiedzy, nie każdy plik powinien zasilać decyzje, a nie każdy dokument może być bezpiecznie wektoryzowany. Sam fakt, że system wyprodukował rekord, nie czyni go poprawnym. Architektura medallion przypomina, że dane muszą przejść przez instytucjonalne sito: surowość jest cenna dla audytu, ale niewystarczająca dla decyzji; certyfikacja zaś jest niezbędna dla zarządu i agentów, lecz nie może usuwać śladu pochodzenia. Dobra architektura nie niszczy tych warstw, lecz je zachowuje, gdyż każda pełni inną funkcję.

W epoce AI szczególne znaczenie zyskuje published stage – etap publikacji certyfikowanych produktów danych. To z niego powinny korzystać systemy generujące odpowiedzi dla szerokiej grupy użytkowników. Jeśli agent operuje na danych niecertyfikowanych, powinno być to jawne i ograniczone do eksploracji. Dashboard zarządczy korzystający z danych eksperymentalnych musi wyraźnie ostrzegać o ich statusie. GenBI tworzący automatyczne historie danych musi wiedzieć, czy opowiada na podstawie zatwierdzonej wersji prawdy organizacyjnej, czy jedynie roboczego szkicu.

W przeciwnym razie organizacja wprowadza literacką fikcję do obiegu decyzyjnego - a potem dziwi się, że rzeczywistość nie chce jej recenzować łagodnie.

Tu pojawia się jednak kolejny problem: centralna architektura lakehouse nie rozwiązuje sama z siebie kwestii znaczenia domenowego. Może zapewnić formaty, spójność, transakcje i wydajność, ale nie odpowie na pytanie, co dokładnie oznaczają dane w konkretnym dziale. Dane sprzedażowe, logistyczne, finansowe czy HR mają własne języki; ich sens bywa lokalny i zależny od specyficznych praktyk oraz kompromisów.

Data Mesh jako rozwiązanie konfliktu między skalą a znaczeniem danych

Centralny zespół platformowy często nie ma świadomości, że określone pole w tabeli przez lata było interpretowane różnie w dwóch regionach, że status „aktywny" zmienił definicję po migracji systemu, czy że „data zamknięcia sprawy" oznacza co innego dla działu prawnego, a co innego dla obsługi klienta. Właśnie dlatego pojawia się Data Mesh. To nie tylko kolejny wzorzec technologiczny, lecz próba rozwiązania konfliktu między skalą a znaczeniem. Centralizacja zapewnia kontrolę, ale gubi lokalną wiedzę; decentralizacja dostarcza wiedzy domenowej, lecz grozi chaosem. Data Mesh proponuje rozwiązanie: przekażmy odpowiedzialność za dane domenom biznesowym, zachowując jednocześnie zfederowany ład, wspólną platformę i standardy bezpieczeństwa. Domena staje się właścicielem produktu danych, ponieważ to ona najlepiej rozumie jego kontekst. Platforma centralna ma natomiast dostarczyć narzędzia, katalog, polityki oraz zapewnić interoperacyjność i egzekwowanie reguł. Kluczowe są tu cztery filary: decentralizacja i własność domenowa, dane jako produkt, samoobsługowa platforma danych oraz zfederowany ład.

Każdy z tych filarów adresuje konkretną patologię. Decentralizacja eliminuje niewydolność centralnego zespołu jako jedynego pośrednika. Koncepcja danych jako produktu kończy erę bylejakości i wrzucania plików bez odpowiedzialności. Samoobsługowa platforma usuwa zależność biznesu od technicznych kolejek oczekiwania. Zfederowany ład natomiast niweluje ryzyko plemiennej anarchii danych, w której każda domena buduje własne definicje, zabezpieczenia i narzędzia. Traktowanie danych jako produktu jest jednym z najbardziej płodnych pojęć tej architektury. Produkt danych posiada odbiorcę, opis, gwarantowaną jakość, właściciela, SLA, reguły użycia oraz określony cykl życia.

Nie jest on zwykłym „zrzutem z systemu", lecz zasobem zaprojektowanym do konsumpcji. Wymaga to zmiany kultury pracy. Działy biznesowe nie mogą już twierdzić: „my tylko generujemy dane,

niech IT coś z nimi zrobi". Z kolei IT nie może odpowiadać: „my tylko utrzymujemy platformę, niech biznes sam wyjaśni sens". W Data Mesh odpowiedzialność jest rozdzielona, ale nie rozmyta. Domena odpowiada za znaczenie, platforma za środowisko, ład za granice, a użytkownik za świadome wykorzystanie.

Amazon DataZone służy tutaj jako narzędzie umożliwiające praktyczne zarządzanie modelem rozproszonym. Pozwala rozwiązywać problem silosów poprzez decentralizację odpowiedzialności, angażując właścicieli domen danych oraz centrum doskonałości (CoE), które ustanawia standardy bezpieczeństwa i operacyjne. To połączenie jest istotne: domena nie działa w izolacji, a CoE nie rządzi despotycznie. Efektywne CoE nie powinno być ministerstwem pieczątek, lecz instytucją wyznaczającą standardy, oferującą wsparcie, wzorce, kontrolę i edukację. Gdy CoE staje się biurokratycznym smokiem, biznes zaczyna obchodzić zasady. Gdy jest zbyt słabe, domeny tworzą własne księstwa danych. DataZone rozwiązuje także problem „odkrywalności" danych. W wielu organizacjach dane istnieją, ale nikt nie wie, gdzie są, kto je posiada, czy są aktualne i czy wolno ich użyć. Prowadzi to do zjawiska równoległego wysiłku: wiele zespołów buduje podobne zestawy, powiela transformacje i tworzy sprzeczne definicje. Katalog danych nie jest więc luksusową wyszukiwarką, lecz mapą instytucjonalnej pamięci. Bez niej organizacja nie eksploruje danych, lecz błądzi po nich z latarką i nadzieją.

W tym kontekście demokracja danych nabiera dojrzałego sensu. Nie chodzi o to, by każdy miał dostęp do wszystkiego, ale by dane były możliwe do odnalezienia, zrozumienia i legalnego wykorzystania przez osoby z uzasadnioną potrzebą. Demokracja ta wiąże się zatem z katalogiem, metadanymi, workflow dostępu, subskrypcją, klasyfikacją, maskowaniem i audytem. Jest to demokracja konstytucyjna, a nie wiec na placu. Bez „konstytucji danych" zwycięża ten, kto dysponuje największą liczbą eksportów do Excela.

W epoce agentów AI katalog danych i Data Mesh zyskują nową funkcję: stają się środowiskiem orientacji dla maszyn. Agent wykonujący zadanie analityczne musi wiedzieć, jakie zasoby istnieją, które są certyfikowane, jakie mają definicje i ograniczenia dostępu. Metadane przestają być jedynie dokumentacją dla ludzi, a stają się częścią kontekstu operacyjnego AI. Jeśli metadane są ubogie, agent działa jak stażysta pierwszego dnia pracy: jest pewny siebie i uprzejmy, ale nie wie, gdzie są akta i kto podejmuje decyzje. Tu widać, że AI-Augmented Data Practice nie oznacza zastąpienia ładu danych przez AI, lecz jego wzmocnienie. AI może automatycznie generować opisy, wykrywać wzorce, profilować zbiory czy sugerować klasyfikację PII, ale bez reguł odpowiedzialności automatyzacja jedynie przyspieszy rozprzestrzenianie nieporządku. Automatyczne metadane są cenne, lecz wymagają weryfikacji; automatyczne profilowanie jest pomocne, ale nie zastąpi wiedzy domenowej.

Architekt danych jako urbanista informacji

Agent utrzymania tabel Iceberg może planować kompaktowanie i optymalizację, jednak to człowiek musi określić, które tabele są krytyczne, jakie SLA obowiązuje i jakie będą konsekwencje błędu. Koncepcja Iceberg Table Maintenance Agent, który wykonuje discovery, inventory, metrics collection, analysis, planning oraz maintenance actions, stanowi świetny przykład przejścia od ręcznego administrowania do autonomicznego utrzymania.

Tabele w lakehouse nie są statyczne. Pliki rosną i fragmentują się, partycje wymagają optymalizacji, a statystyki muszą pozostać aktualne. Małe pliki mogą degradować wydajność, podczas gdy koszt zapytań często rośnie niezauważalnie. Agent utrzymania potrafi wykrywać te problemy i planować konserwację, ale nawet tutaj autonomia musi być ograniczona przez politykę. Nie każda optymalizacja techniczna jest zasadna w każdym momencie; nie każdą tabelę można przebudowywać podczas krytycznego okna raportowego i nie każdy zysk kosztowy jest ważniejszy od stabilności.

To prowadzi do pojęcia obserwowalności jako trzeciego oka architektury. Jeśli dane mają zasilać decyzje i agentów, organizacja musi widzieć wszystko: co dzieje się z pipeline'ami, zapytaniami, tabelami, indeksami, modelami i samymi agentami. Logi, metryki i traces nie są jedynie dodatkiem dla administratorów – są materiałem dowodowym. Pozwalają ustalić, gdzie proces się zatrzymał, dlaczego wynik uległ zmianie, kto uzyskał dostęp, który agent wywołał narzędzie, jaki prompt został użyty, jakie dokumenty pobrano i jaki był stan systemu w chwili podjęcia decyzji. Bez obserwowalności AI w przedsiębiorstwie staje się czarną skrzynką z melodyjnym głosem.

Architektura danych jest zawsze kompromisem; nie istnieje jedna idealna forma dla wszystkich organizacji. Data warehouse może nadal być najlepszy dla stabilnego raportowania finansowego, data lake niezbędny do eksploracji surowych danych, a lakehouse zdolny połączyć elastyczność z rygorem. Z kolei Data Mesh pozwala skalować odpowiedzialność domenową.

Bazy wektorowe wspierają wyszukiwanie semantyczne, Redshift obsługuje wydajne zapytania analityczne, a S3 stanowi trwały rdzeń przechowywania. Problem nie polega na wyborze jednego zwycięzcy, lecz na zaprojektowaniu spójnej architektury, w której każdy komponent ma właściwą funkcję i granice. Najbardziej niebezpieczne są architektury przypadkowe, powstałe przez akumulację mód: trochę hurtowni, trochę jeziora, trochę katalogu, RAG-a, agentów i dashboardów, a do tego kopie danych w prywatnych arkuszach, eksperymentalne modele i wyjątkowe ustawienia bezpieczeństwa „na chwilę”, które zostają na lata. Taki krajobraz może wyglądać nowocześnie w prezentacji, ale pod spodem przypomina miasto budowane bez planu zagospodarowania: drogi

prowadzą w ściany, rury krzyżują się z kablami, a każdy burmistrz dzielnicy ma własne przepisy przeciwpożarowe.

Architekt danych musi więc być urbanistą informacji. Musi rozumieć przepływy, własność, skalę, ryzyko, przyszłą rozbudowę, koszty utrzymania i scenariusze ewakuacji przy awarii. Jego zadaniem jest projektowanie nie tylko tego, „gdzie dane leżą", ale także jak dojrzewają, kto je opisuje i używa, jak są zabezpieczane, wersjonowane, w jaki sposób trafiają do modeli i jak można odtworzyć ich historię. W epoce AI architektura danych nie jest zapleczem – jest sceną, na której działają modele, agenci i ludzie. Jeśli scena jest źle zbudowana, nawet najlepszy aktor spadnie do orkiestronu.

Różnica między Lake a Mesh nie powinna być zatem opisywana jako prosta alternatywa. Data Lake odpowiada na pytanie o skalowalne przechowywanie różnorodnych danych, natomiast Data Mesh dotyczy skalowalnej odpowiedzialności za znaczenie tych danych. Lakehouse łączy elastyczność jeziora z transakcyjnością i wydajnością, DataZone pozwala odnajdywać, publikować i kontrolować produkty danych w środowisku domenowym, a Apache Iceberg umożliwia zarządzanie tabelami w jeziorze z dojrzałością hurtowni.

Architektura danych jako normatywny ustrój wiedzy organizacji

Każde z tych rozwiązań dotyka innej warstwy problemu. Dopiero razem mogą tworzyć spójną odpowiedź. W organizacji agentowej te warstwy nabierają dodatkowego znaczenia. Agent nie powinien być wypuszczony na surowe bagno danych. Powinien działać w środowisku, w którym wie, które produkty danych są zatwierdzone, jakie mają definicje, jakie są ograniczenia dostępu, jakie zapytania wolno wykonać, jakie wyniki wymagają walidacji i kiedy należy odmówić odpowiedzi. Demokracja danych nie oznacza więc "otwórzmy wszystko agentom". Oznacza: zbudujmy taki ład, aby agenci mogli działać produktywnie, a jednocześnie nie stali się szybkim kanałem nadużyć, błędów i wycieków. Dlatego architektura danych w epoce AI jest jednocześnie architekturą techniczną, ekonomiczną i normatywną. Techniczną, bo dotyczy formatów, tabel, silników, indeksów, potoków, orkiestracji i metadanych. Ekonomiczną, bo decyduje o koszcie przetwarzania, czasie odpowiedzi, produktywności, przychodach i wykorzystaniu aktywów. Normatywną, bo określa, kto ma prawo wiedzieć, kto ma prawo pytać, kto ma prawo automatyzować i kto odpowiada za wynik. każdy model danych jest małą konstytucją organizacji. Niektóre konstytucje są dojrzałe. Inne są napisane na serwetce podczas wdrożenia.

Wracamy więc do głównej figury eseju: inżynier danych nie może być woodpusherem. W architekturze danych woodpusher pyta: "jakiego narzędzia użyć?". Architekt pyta: "jaki ustrój wiedzy budujemy?". Woodpusher wrzuca dane do jeziora. Architekt projektuje drogę od źródła do produktu. Woodpusher ufa, że katalog sam rozwiąże problem silosów. Architekt wie, że katalog bez właścicieli i standardów jest tylko eleganckim spisem bałaganu. Woodpusher mówi: "mamy Data Mesh". Architekt pyta: "czy domeny naprawdę odpowiadają za jakość, definicje i cykl życia swoich produktów danych?".

Woodpusher instaluje narzędzia AI. Architekt buduje warunki, w których AI może być użyte bez utraty kontroli nad prawdą. Najważniejszy wniosek jest prosty, choć wymagający: organizacja nie stanie się agentowa dzięki samemu wdrożeniu agentów. Stanie się agentowa dopiero wtedy, gdy jej architektura danych pozwoli agentom działać na zasobach uporządkowanych, opisanych, zabezpieczonych, wersjonowanych i powiązanych z odpowiedzialnością domenową. Bez tego agent AI jest tylko szybkim turystą po cudzym bałaganie. Zadaje pytania, znajduje coś, odpowiada pewnie, a potem wszyscy odkrywają, że korzystał z mapy sprzed trzech reorganizacji. Przetwarzanie i orkiestracja.

Dane nie płyną same, a pipeline bez obserwowalności jest tylko nadzieją w przebraniu systemu

Architektura danych wyznacza porządek przestrzeni informacyjnej organizacji, ale sama architektura nie wystarcza. Dane nie są marmurowymi posągami ustawionymi w muzeum. Są ruchem. Napływają, zmieniają się, psują, dublują, starzeją, tracą kontekst, wymagają walidacji, transformacji, wzbogacania, kontroli jakości i publikacji. Jeśli poprzednia część dotyczyła ustroju danych, ta musi dotyczyć ich pracy: tego, jak surowy materiał staje się zasobem analitycznym, produktem danych, kontekstem dla RAG, paliwem dla GenBI i środowiskiem działania agentów AI. W tradycyjnej inżynierii danych centralne miejsce zajmowały procesy ETL, czyli extract, transform, load: pobrać dane ze źródła, przekształcić je, załadować do systemu docelowego. Ten model dobrze opisuje wiele klasycznych zadań, ale w epoce AI okazuje się zbyt wąski. Dziś dane mogą wymagać nie tylko transformacji tabelarycznej, lecz także klasyfikacji dokumentów, ekstrakcji tekstu ze skanów, transkrypcji audio, analizy obrazów, wykrywania danych osobowych, maskowania, segmentacji, generowania embeddingów, zapisu do bazy wektorowej, budowy warstwy metadanych, aktualizacji indeksów semantycznych i kontroli, czy agent korzystający z tych danych nie przekracza przyznanych mu uprawnień. To nie jest już prosty rurociąg.

Elastyczność chmury wymaga dyscypliny projektowej

To sieć procesów poznawczych, technicznych i bezpieczeństwowych.

Elastyczność pozwala rozwiązywać dwa klasyczne problemy infrastruktury: niewydajność systemów o stałej pojemności oraz brak skalowalności przy nagłym wzroście obciążeń. W środowisku on-premise organizacja często musiała wybierać między przewymiarowaniem zasobów a ryzykiem niedostatecznej mocy w krytycznym momencie. Chmura przesuwa ten problem: zasoby można powoływać na czas zadania, skalować poziomo i płacić za faktyczne użycie, dobierając moc do charakteru obciążenia. Jest to szczególnie istotne w świecie AI, gdzie jednego dnia potrzeba zwykłej transformacji wsadowej, drugiego masowego generowania embeddingów, trzeciego przetwarzania multimodalnego, a czwartego intensywnego profilowania danych przed wdrożeniem nowego agenta.

Jednak elastyczność nie jest synonimem dojrzałości. Można dysponować elastyczną chmurą i nieelastycznym myśleniem. Można dynamicznie powoływać klastry, które wykonują źle zaprojektowane transformacje, skalując koszt szybciej niż wartość. Można zbudować pipeline, który działa, dopóki nikt nie zapyta, dlaczego wynik jest taki, a nie inny. Dlatego elastyczność techniczna musi iść w parze z dyscypliną projektową. Chmura pozwala uniknąć ograniczeń fizycznych, ale nie zwalnia z obowiązku architektury. Przeciwnie: im łatwiej uruchomić zasoby, tym łatwiej także uruchomić chaos z fakturą miesięczną.

Szczególne miejsce zajmują AWS Glue i Apache Spark jako narzędzia przetwarzania wsadowego i rozproszonego. Spark pozostaje jednym z najważniejszych fundamentów nowoczesnej inżynierii danych, umożliwiając pracę na dużych zbiorach w modelu rozproszonym. Jego znaczenie nie sprowadza się jednak wyłącznie do szybkości. Spark uczy myślenia w kategoriach partycji, równoległości, kosztu shufflingu, transformacji leniwych, operacji szerokich i wąskich, pamięci, serializacji oraz fizycznej ceny abstrakcyjnie wyglądającego kodu. To właśnie tu odróżnia się inżynier od woodpushera.

Woodpusher poprosi model o kod PySpark. Inżynier zapyta natomiast, czy ten kod nie wywoła katastrofalnego joinu, czy nie przeniesie danych niepotrzebnie między węzłami, czy nie stworzy tysięcy małych plików, czy nie złamie założeń partycjonowania i czy będzie możliwy do utrzymania za pół roku. W epoce generatywnej AI tworzenie kodu ETL rzeczywiście staje się szybsze. Asystent może wygenerować szkic transformacji, zaproponować strukturę tabeli, napisać testy jakości, przygotować fragment PySpark, pomóc w konwersji do Apache Iceberg albo zasugerować strategie partycjonowania. To ogromny skok produktywności, ale właśnie dlatego rośnie znaczenie recenzji i głębokiego rozumienia. Kod wygenerowany przez AI nie jest mniej kodem.

Kontrola jakości i orkiestracja jako gwaranty wiarygodności produktów wiedzy

Ma to swoje skutki. Zużywa zasoby i tworzy dane, którym ktoś zaufa. Jeśli wygeneruje błąd, może on zostać opublikowany jako prawda organizacyjna. W inżynierii danych „prawie dobrze" bywa gorsze niż jawnie źle, ponieważ potrafi przejść przez procesy, raporty i decyzje bez żadnego alarmu. Dlatego kontrola jakości danych staje się centralnym elementem nowej praktyki.

Wspomniany w notatce framework Deequ służy do programowej weryfikacji poprawności milionów rekordów pod kątem anomalii i odchyleń. To istotny trop, gdyż testowanie danych różni się od tradycyjnego testowania kodu. Kod może być logicznie poprawny, ale dane wejściowe mogą zmienić się w sposób, który całkowicie unieważnia wynik. System źródłowy może zacząć wysyłać nowe wartości, zmienić format daty, wprowadzić puste pola, zdublować identyfikatory lub nagle przesłać dane z opóźnieniem. Pipeline nadal „działa", ale produkuje wynik, który przestaje znaczyć to, co dotychczas. Test jakości danych jest więc strażnikiem semantycznej ciągłości. Sprawdza nie tylko, czy proces się wykonał, ale czy dane zachowały sensowne właściwości: kompletność, unikalność, zakresy wartości, rozkłady, relacje między polami, spójność referencyjną, oczekiwane proporcje i brak nienaturalnych skoków. W świecie AI testy te nabierają dodatkowego znaczenia, ponieważ dane zasilają modele, agentów i systemy konwersacyjne.

Jeśli do korpusu RAG trafi wadliwa partia dokumentów, model nie powie: „przepraszam, moja baza została zanieczyszczona". Raczej odpowie pewnym tonem, używając tego, co otrzymał. A pewny ton bywa najtańszym kostiumem błędu.

Kontrola jakości musi obejmować także etap profilowania. Narzędzia takie jak AWS Glue DataBrew, wskazane w notatce jako sposób wizualnego badania struktury danych, integralności i relacji między zmiennymi, mają kluczowe znaczenie diagnostyczne. Profilowanie pozwala poznać dane przed ich użyciem: rozpoznać typy, braki, rozkłady, wartości odstające, możliwe PII, niestandardowe kody i nietypowe korelacje. To odpowiednik badania pacjenta przed operacją. Chirurg, który pomija diagnostykę, może być szybki, ale trudno nazwać go odpowiedzialnym. Inżynier danych, który przetwarza niepoznane dane, również działa na ślepo – nawet jeśli robi to w bardzo eleganckim notebooku.

W praktyce przetwarzanie danych w epoce AI powinno być projektowane jako ciąg etapów dojrzewania, a nie pojedynczy skrypt. Najpierw dane są pobierane i rejestrowane, potem profilowane, następnie klasyfikowane pod względem wrażliwości, walidowane, transformowane i standaryzowane, by na koniec zostać przetestowane jakościowo. Dopiero wtedy są publikowane

jako produkt danych lub kierowane do dalszej obróbki: wektoryzacji, indeksowania, wzbogacania metadanych, agregacji lub zasilania warstwy semantycznej.

Każdy etap powinien zostawiać ślad, mieć właściciela i być obserwowalny. Jeśli coś zawiedzie, system musi wiedzieć co, gdzie, dlaczego i czy można bezpiecznie ponowić zadanie. To prowadzi nas do orkiestracji. Gdy system danych składa się z wielu zależnych procesów, potrzebuje dyrygenta. Orkiestracja nie polega jednak na tym, że narzędzie wykonuje całą pracę. Notatka słusznie podkreśla zasadę separation of concerns: orkiestrator nie powinien służyć do przetwarzania danych; logika transformacji powinna znajdować się w usługach takich jak AWS Glue, a orkiestrator ma uruchamiać zadania, kontrolować zależności, monitorować statusy oraz obsługiwać błędy i ponowienia. To zasada higieny architektonicznej. Gdy orkiestrator staje się jednocześnie miejscem logiki biznesowej, przetwarzania, wyjątków i integracji, powstaje potwór. A potwory w danych nie ryczą. One po prostu działają coraz trudniej.

Dwa narzędzia wymienione w notatce – AWS Step Functions oraz Amazon MWAA (zarządzany Apache Airflow) – reprezentują dwa style orkiestracji. Step Functions opiera się na maszynach stanów, definiowanych deklaratywnie, z wbudowaną obsługą błędów i ponowień. Airflow wykorzystuje DAG-i, czyli skierowane grafy acykliczne pisane w Pythonie, oferując bogaty ekosystem operatorów i szerokie uznanie w świecie inżynierii danych.

Orkiestracja i obserwowalność jako gwaranty wiarygodności systemu

Wybór narzędzi nie powinien wynikać z mody, lecz z charakteru procesów, kompetencji zespołu, wymagań kontroli i integracji, a także sposobu utrzymania i oczekiwanej przejrzystości. Orkiestracja to w istocie sztuka zarządzania zależnościami: dane z systemu A muszą dotrzeć przed transformacją B; ta z kolei musi zakończyć się sukcesem, zanim ruszy walidacja C. Dopiero gdy walidacja potwierdzi jakość danych i trafią one do published stage, można odświeżyć warstwę semantyczną, zaktualizować indeks wektorowy, uruchomić alert anomalii lub pozwolić agentowi odpowiedzieć na pytanie użytkownika. Jeśli jeden element zawiedzie, system musi wiedzieć, co zatrzymać, co ponowić, co oznaczyć jako nieaktualne i kogo powiadomić. Bez orkiestracji organizacja nie posiada pipeline'u. Ma serię życzeń wykonywanych w przypadkowej kolejności.

Kluczowa jest tutaj enkapsulacja, rozumiana jako podział długich procesów na mniejsze, logiczne części. Dobrze zaprojektowany pipeline nie jest jednym gigantycznym zadaniem, którego awaria wymusza powtórzenie całości, lecz strukturą modułową. Jeśli zawiedzie ekstrakcja z jednego źródła,

nie trzeba przetwarzać od nowa całego świata. Błąd w walidacji konkretnej tabeli powinien zatrzymać publikację tej jednej tabeli, a nie paraliżować cały ekosystem. Enkapsulacja ogranicza koszt awarii, który w systemach danych nie mierzy się tylko w pieniądzach. To także utrata czasu i zaufania oraz ryzyko błędnych decyzji – sytuacje, w których ludzie muszą wyjaśniać, dlaczego zarząd przez trzy dni analizował fałszywy wykres.

Ważnym pojęciem jest idempotencja. Proces idempotentny można uruchomić ponownie bez niepożądanych skutków ubocznych, co jest fundamentalne dla retry logic. Jeśli zadanie ładowania danych zostanie powtórzone po błędzie, nie powinno ono zdublować rekordów, nadpisać danych w sposób nieprawidłowy ani zmienić wyniku w zależności od liczby uruchomień. W epoce orkiestracji i automatycznych ponowień idempotencja jest jak dobre maniery systemu: nie robi bałaganu, gdy musi wrócić do stołu. Jej brak to proszenie się o subtelne katastrofy, które ujawniają się dopiero podczas audytu lub – co gorsza – przy kluczowej decyzji biznesowej.

Kolejnym zagadnieniem jest Time to Answer, wiążący rygorystyczne ograniczenia czasowe analiz z biznesowym SLA. Czas odpowiedzi nie jest parametrem kosmetycznym; w wielu procesach informacja spóźniona staje się informacją martwą. Alert o oszustwie po tygodniu to kronika, a nie ochrona. Analiza churnu po odejściu klienta jest epitafium, nie interwencją, a rekomendacja cenowa po zakończeniu kampanii jedynie ćwiczeniem historycznym.

Dlatego architektura przetwarzania musi rozróżniać obciążenia wsadowe, strumieniowe, quasi-real-time i interaktywne. Nie każda analiza musi być natychmiastowa, ale każda powinna mieć określony sens czasowy. Pytanie nie brzmi: „jak szybko możemy?”, lecz „jak szybko musimy, aby wynik nadal miał wartość?”. W epoce AI Time to Answer nabiera nowego wymiaru – użytkownicy przyzwyczajeni do modeli konwersacyjnych oczekują natychmiastowości. Rodzi to presję na systemy danych, które historycznie operowały w rytmach nocnych odświeżeń i miesięcznych zamknięć. GenBI, Text-to-SQL i agenci analityczni przesuwają oczekiwania ku dialogowi z danymi, jednak nie wszystkie dane są do niego gotowe w czasie rzeczywistym. Jeśli organizacja nie odróżni świeżości danych od szybkości interfejsu, użytkownik otrzyma błyskawiczną odpowiedź na podstawie wczorajszego świata. Nie jest to błędem, o ile system jasno komunikuje aktualność źródła. Szybka odpowiedź bez znacznika czasu jest jak gazeta bez daty – może być wiadomością albo eksponatem muzealnym.

Automatyzacja orkiestracji przez AI jest jednym z najbardziej kuszących obszarów nowej praktyki. Modele generatywne potrafią tworzyć definicje DAG, pliki JSON dla Step Functions, szkice retry logic czy dokumentację procesu. Znacząco obniża to koszt wejścia, jednak im łatwiej wygenerować DAG, tym głębiej należy rozumieć jego istotę. Skierowany graf acykliczny to nie ozdobny diagram przepływu, lecz formalny opis zależności o konkretnych konsekwencjach operacyjnych. Błędna

zależność może uruchomić zadanie zbyt wcześnie, jej brak pozwolić na publikację niepełnego wyniku, a nadmiarowość spowolnić cały system. Cykliczne myślenie w acyklicznej strukturze kończy się zwykle rachunkiem wystawionym przez rzeczywistość.

Tutaj powraca problem woodpushera. Woodpusher powie: „AI wygenerowała mi pipeline". Inżynier zapyta o graf zależności, punkty awarii, strategię ponowień, warunki zatrzymania, zakres danych, idempotencję, politykę alertów, koszt wykonania, czas odpowiedzi i sposób odtworzenia wyniku. To nie pedanteria, lecz różnica między prototypem a systemem produkcyjnym. Prototyp ma zachwycać; system produkcyjny ma działać, gdy nikt nie patrzy – gdy źródło spóźnia się o dwie godziny, API zwraca nietypowy błąd, schemat zmienia kolumnę, indeks wektorowy wymaga aktualizacji, a użytkownik biznesowy pyta o wynik, który powinien być już dostępny.

Właśnie dlatego kluczowa jest obserwowalność, oparta na trzech filarach: logach, metrykach i śladach (traces). Logi rejestrują to, co system zapisał o swoim działaniu. Metryki pozwalają mierzyć stan i zachowanie: czas wykonania, liczbę rekordów, błędy, opóźnienia, koszt czy odsetek niepoprawnych danych. Traces umożliwiają śledzenie przepływu żądania przez wiele komponentów. Razem tworzą pamięć operacyjną systemu. Bez nich każda awaria staje się zagadką, a śledztwo sprowadza się do zbiorowego szeptania: „u mnie działało".

Obserwowalność i ekonomia jako fundamenty zaufania do AI

W nowej inżynierii danych obserwowalność musi obejmować nie tylko pipeline'y, lecz także produkty danych, modele, zapytania, indeksy i agentów. Należy widzieć, kiedy dane zostały ostatnio odświeżone, jakie testy przeszły, które reguły jakości zawiodły, jakie dokumenty trafiły do bazy wiedzy i kiedy wygenerowano embeddingi. Trzeba wiedzieć, jakie źródła pobrał RAG oraz który agent wywołał konkretne narzędzie, w jakiej sesji, z jakim użytkownikiem i jaki był wynik. Jeśli agent AI staje się uczestnikiem procesu, jego działania muszą być obserwowalne tak jak działania człowieka – a czasem bardziej, bo człowiek przynajmniej potrafi zostać wezwany na spotkanie i rumienić się w sposób dowodowy.

Współczesne rozwiązania oferują wyspecjalizowanych asystentów, takich jak AWS DevOps Agent czy Log Analysis Agent, którzy mogą analizować awarie pipeline'ów i wskazywać konkretne linie kodu lub poprawki składniowe. To pokazuje istotny kierunek: AI nie tylko korzysta z danych, ale pomaga utrzymywać infrastrukturę danych.

Agent może analizować logi, wykrywać wzorce błędów, sugerować poprawki i porównywać anomalie z historią, co znacząco skraca czas diagnozy. Jest to funkcja niezwykle cenna, gdyż nowoczesne systemy są zbyt złożone, by człowiek mógł ręcznie przeglądać terabajty logów przy każdej awarii. Jednak agent diagnostyczny nie znosi potrzeby rozumienia; jego diagnoza powinna być traktowana jak rekomendacja zdolnego analityka, a nie jak dekret monarchy. W tym miejscu należy rozróżnić monitoring od observability. Monitoring odpowiada na pytanie, czy znane elementy działają zgodnie z oczekiwaniem. Obserwowalność pozwala natomiast badać nieznane stany systemu na podstawie jego zewnętrznych sygnałów.

Monitoring mówi: „ten job trwa zbyt długo". Obserwowalność pozwala ustalić dlaczego tak się dzieje – przez które źródło, transformację, zasób, zmianę danych lub błąd zależności. W systemach AI obserwowalność musi iść jeszcze dalej: powinna umożliwiać rekonstrukcję nie tylko przebiegu technicznego, ale także kontekstu decyzyjnego. Jakie dane widział model? Jakie dokumenty przywołał retriever? Jaką wersję promptu zastosowano? Czy guardrails zadziałały? Czy odpowiedź została oznaczona jako niepewna i czy użytkownik otrzymał źródła?

To wszystko prowadzi do pojęcia data lineage, czyli pochodzenia i ścieżki przekształceń danych. Bez lineage organizacja nie wie, skąd wziął się wynik, a bez tej wiedzy nie może mu naprawdę ufać. W klasycznej analityce brak lineage utrudniał audyt; w AI może on uniemożliwić odpowiedzialność. Jeśli model zasugerował decyzję na podstawie danych, które przeszły przez pięć transformacji, trzy agregacje, anonimizację i aktualizację indeksu, organizacja musi umieć prześledzić tę drogę. W przeciwnym razie każde wyjaśnienie jest tylko opowieścią po fakcie, a takie narracje mają skłonność do wybielania architektury.

Przetwarzanie i orkiestracja muszą być projektowane również z myślą o kosztach. Chmura daje elastyczność, ale nie darmowość. Rozproszone przetwarzanie skraca czas, lecz źle napisane zadanie może wygenerować nadmierne wydatki. Wektoryzacja dużych korpusów, obsługa dokumentów multimodalnych, odświeżanie indeksów czy zapytania ad hoc mogą stać się kosztowne przy braku kontroli. W epoce AI pojawia się nowe zjawisko: koszt ciekawości. Użytkownicy zadają więcej pytań, agenci wykonują więcej kroków, a modele generują więcej operacji. Bez polityk limitów, cache'owania i obserwowalności wydatków, demokracja danych może stać się fiskalnym festynem. Nie oznacza to jednak, że należy ograniczać AI z lęku przed kosztami, lecz że trzeba projektować ekonomię systemu. Nie każde zapytanie wymaga najdroższego modelu, nie każda aktualizacja pełnej regeneracji embeddingów i nie każda analiza musi działać w czasie rzeczywistym. Dobra architektura wie, kiedy użyć mocy, a kiedy pamięci; kiedy przeliczyć, a kiedy odczytać; kiedy zapytać model, a kiedy wystarczy reguła. To jest ekonomia inteligencji.

Inżynier danych jako mediator między technologią a biznesem

Przetwarzanie danych w epoce AI coraz częściej obejmuje zasoby nieustrukturyzowane i multimodalne. Klasyczny pipeline tabelaryczny musi dziś współistnieć z potokami dokumentowymi, obrazowymi, audio i wideo. Dokument PDF wymaga ekstrakcji tekstu, rozpoznania tabel, klasyfikacji typu, wykrycia podpisów, usunięcia danych osobowych, podziału na fragmenty semantyczne i zapisania embeddingów. Nagranie rozmowy z kolei wymusza transkrypcję, diarizację mówców, analizę sentymentu, detekcję tematów, anonimizację i powiązanie z rekordem klienta. Obraz może wymagać detekcji obiektów, opisu sceny, wektoryzacji oraz powiązania z metadanymi technicznymi. Są to wciąż procesy danych, choć nie przypominają dawnych systemów ETL. Orkiestracja musi zatem obsługiwać zależności nie tylko między tabelami, ale i pomiędzy różnymi modalnościami.

Jeśli dokument zostanie najpierw zwektoryzowany, a dopiero potem wykryje się w nim PII, jest za późno – dane wrażliwe mogły już trafić do indeksu. Błędy w transkrypcji audio mogą sprawić, że analiza semantyczna błędnie sklasyfikuje intencję klienta, a opis obrazu wykonany bez uwzględnienia kontekstu branżowego pozostanie powierzchowny. Dlatego pipeline multimodalny musi być projektowany jako sekwencja kontroli i wzbogacania, a nie jako wrzucenie pliku do "magicznego AI". Z punktu widzenia organizacji kluczowy jest jednak wynik: przetwarzanie i orkiestracja mają przekształcać dane w wiarygodną, terminową i użyteczną wartość. Wymaga to połączenia technologii z zarządzaniem. Zespół danych musi uzgodnić z biznesem SLA, priorytety, definicje, kryteria jakości, świeżość danych, dopuszczalne opóźnienia i poziomy krytyczności.

Nie wszystkie pipeline'y są równie ważne. Potok zasilający regulatorowy raport ryzyka ma inną wagę niż eksperymentalna analiza marketingowa. Pipeline aktualizujący indeks RAG dla agenta obsługi klienta bezpośrednio wpływa na jakość odpowiedzi udzielanych ludziom, natomiast system wykrywający fraud wymaga minimalnego opóźnienia.

Architektura bez priorytetów traktuje wszystko jednakowo, czyli w praktyce nie rozumie niczego. Nowy inżynier danych pracuje zatem na styku systemów technicznych i umów organizacyjnych. Musi potrafić rozmawiać z biznesem o wartości, z prawnikami o ryzyku, z security o uprawnieniach, z data scientistami o jakości cech, z analitykami o definicjach, z inżynierami platformy o skalowalności, a z zarządem o konsekwencjach opóźnień. To nie jest rola zamknięta w kodzie; to rola mediacyjna. Inżynier danych jest tłumaczem między maszynową operacyjnością a ludzką odpowiedzialnością.

Odpowiedzialność architektoniczna jako fundament wiarygodności AI

Warto przywołać tutaj bon mot: „Dobrzy ludzie na słabych statkach są lepsi niż słabi ludzie na dobrych statkach”. W kontekście przetwarzania i orkiestracji oznacza to, że nawet najlepsze narzędzia nie zastąpią zespołu rozumiejącego fundamenty. Można dysponować AWS Glue, Airflow, Step Functions, Deequ, Iceberg, DataZone, Bedrock i agentami diagnostycznymi, a mimo to zbudować kruchy system. Można natomiast operować skromniejszym stosem technologicznym, ale opierać się na dobrych standardach, testach, obserwowalności i kulturze jakości. Narzędzia są wzmacniaczem – potęgują kompetencje albo pogłębiają bałagan. AI jest pod tym względem bezlitosna: przyspiesza wszystko, także głupotę.

Nie należy jednak popadać w konserwatywny lęk przed automatyzacją. Generatywna AI w obszarze przetwarzania i orkiestracji stanowi ogromną szansę. Może skracać czas tworzenia pipeline'ów, wspierać dokumentację, analizować błędy, sugerować testy jakości, wykrywać dane wrażliwe czy proponować optymalizacje. Ułatwia onboarding nowych inżynierów i zmniejsza dystans między intencją biznesową a implementacją techniczną.

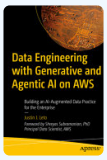
Warunkiem jest jednak dojrzałe podejście. AI powinna być kopilotem architekta, a nie kierowcą autobusu bez prawa jazdy. Siedząc obok, może podpowiadać trasę, ale jeśli oddamy jej kierownicę bez mapy, hamulców i przeglądu technicznego, nie miejmy potem pretensji do asfaltu. Najbardziej praktyczna zasada brzmi: automatyzuj wykonanie, nie automatyzuj odpowiedzialności. Można automatycznie generować kod, lecz odpowiedzialność za jego skutki pozostaje ludzka i organizacyjna. Można automatycznie ponawiać zadania, ale to człowiek musi określić, kiedy jest to bezpieczne. Można automatycznie analizować logi, jednak ktoś musi zdecydować, czy rekomendowana poprawka jest właściwa. Można automatycznie publikować produkty danych, lecz tylko w ramach jasnych progów jakości. Można automatycznie aktualizować indeks RAG, ale wyłącznie po klasyfikacji i ochronie danych wrażliwych.

Automatyzacja bez odpowiedzialności to technologiczna wersja zrzucania winy na drukarkę. W epoce agentów AI przetwarzanie i orkiestracja zyskują nową funkcję: tworzą rytm poznawczy organizacji. To one decydują, jak szybko organizacja uczy się z nowych danych, wykrywa zmiany, aktualizuje swoją pamięć i odpowiada na pytania ludzi oraz systemów. Jeśli pipeline'y są powolne, kruche i nieobserwowalne, organizacja myśli wolno i niepewnie. Jeśli są szybkie, kontrolowane i dobrze opisane, zyskuje zdolność adaptacji – nie poprzez slogan o innowacyjności, lecz dzięki faktycznej infrastrukturze uczenia się. To wyjaśnia, dlaczego inżynier danych staje się kluczową postacią transformacji AI.

Nie dlatego, że pisze najwięcej kodu, lecz dlatego, że projektuje warunki, w których kod, dane, modele, agenci i ludzie tworzą sensowną całość. W klasycznej firmie pipeline był zapleczem raportu; w firmie agentowej jest tętnicą decyzji. Jeśli tętnica jest zatkana, organizacja nie myśli. Jeśli przecieka – traci dane. Jeśli pompuje zanieczyszczenia – zatruwa własną inteligencję. Wniosek jest jednoznaczny: przetwarzanie i orkiestracja to nie „techniczna kuchnia", w której coś się gotuje, zanim trafi na elegancki stół zarządu. To miejsce, gdzie rozstrzyga się wiarygodność całego systemu. Tam dane są badane, czyszczone, transformowane, testowane, publikowane i monitorowane. AI może tam pomóc, ale nie zastąpi architektonicznej dyscypliny. To tutaj woodpusher najczęściej zdradza się tym, że cieszy go działający skrypt, zanim zapyta o jakość wyniku. Dojrzały inżynier wie bowiem, że w danych nie wystarczy status „success". Trzeba jeszcze wiedzieć, czy sukces nie jest tylko dobrze sformatowaną porażką.

> W świecie zdominowanym przez agentową AI, gdzie model potrafi wygenerować kod w sekundy, największym ryzykiem staje się lęk przed brakiem automatyzacji. Prawdziwym wyzwaniem nie jest już bowiem to, by systemy działały szybko, lecz by nie stały się one jedynie elegancką fasadą dla instytucjonalnego chaosu. Ostatecznie to nie technologia, a architektoniczna dyscyplina rozstrzyga, czy organizacja buduje realną inteligencję, czy jedynie szybki i kosztowny sposób na produkowanie przeciętności.

Inspiracje

[1] 

"Data Engineering with Generative n Agentic AI" Justin J Leto

2026 | Apress | ISBN: 9798868821981

Justin J. Leto, PE, MBA, PMP, jest globalnym liderem technologicznym, autorem i mówcą z ponad dwudziestoletnim doświadczeniem w inżynierii danych i sztucznej inteligencji. Obecnie, pełniąc funkcję głównego architekta rozwiązań w Amazon Web Services (AWS), doradza globalnym firmom private equity w zakresie tworzenia wartości w chmurze i sztucznej inteligencji. Jego specjalizacja zawodowa koncentruje się na inżynierii danych na dużą skalę, transformacji sztucznej inteligencji oraz rozwoju nowoczesnych architektur danych, w tym jezior danych i siatki danych. Leto jest doceniany za swój wkład w tę dziedzinę dzięki swojemu technicznemu przywództwu i pracy nad przygotowaniem organizacji na przyszłość praktyk danych wspomaganych przez sztuczną inteligencję. Posiada certyfikaty zawodowe Professional Engineer (PE), Project Management Professional (PMP) oraz MBA, co odzwierciedla jego multidyscyplinarne podejście do technologii i strategii biznesowej.

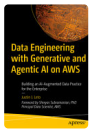
Justin J. Leto, PE, MBA, PMP, is a global technology leader, author, and speaker with over two decades of experience in data engineering and artificial intelligence. Currently serving as a Principal Solutions Architect at Amazon Web Services (AWS), he advises global private equity firms on cloud and AI value creation. His professional expertise focuses on large-scale data engineering, AI transformation, and the development of modern data architectures, including data lakes and data mesh. Leto is recognized for his contributions to the field through his technical leadership and his work in preparing organizations for the future of AI-augmented data practices. He holds professional certifications as a Professional Engineer (PE), a Project Management Professional (PMP), and an MBA, reflecting his multidisciplinary approach to technology and business strategy.

Amazon Web Services (AWS)

Literatura uzupełniająca

Książki

Autorzy przywoływani w artykule:



[1] **"Data Engineering with Generative and Agentic AI on AWS"**

Justin J. Leto

2026 | Apress | ISBN: 9798868821981

Literatura pokrewna (Google Scholar):

[2] **"Agentic Mesh The GenAI-Powered Agent"**

Eric Broda

[3] **"Nonlinear Big Data and AI-Enabled Problem"**

Scott M Shemwell

[4] **"Microsoft Fabric Analytics Engineer Associate"**

Brian Bonk

[5] **"Navigating Data Science Babita Singla"**

[6] **"97 Things Every Engineering Manager Should Camille Fournier"**

Artykuły naukowe

Artykuły pokrewne (Google Scholar):

[1] **"Data Warehousing with Generative AI and Text-to-SQL Reporting with Amazon Redshift"**

Justin J. Leto

2026 | Apress eBooks | DOI: 10.1007/979-8-8688-2199-8_10

Google Scholar

[2] **"Data Engineering with Generative and Agentic AI on AWS"**

Justin J. Leto

2026 | Apress eBooks | DOI: 10.1007/979-8-8688-2199-8

Google Scholar

[3] **"Retrieval-Augmented Generation (RAG) with S3 Vectors and Vector Databases"**

Justin J. Leto

2026 | Apress eBooks | DOI: 10.1007/979-8-8688-2199-8_8

Google Scholar

[4] **"Multimodal Data Extraction and Enrichment with Amazon Bedrock and AWS ML Services"**

Justin J. Leto

2026 | Apress eBooks | DOI: 10.1007/979-8-8688-2199-8_7

Google Scholar

[5] **"Data Security and Governance"**

Justin J. Leto

2026 | Apress eBooks | DOI: 10.1007/979-8-8688-2199-8_2

Google Scholar

[6] **"Streaming and Real-Time Data Processing with Generative AI Enrichment"**

Justin J. Leto

2026 | Apress eBooks | DOI: 10.1007/979-8-8688-2199-8_9

Google Scholar

[7] **"Introduction to Data Engineering with Generative and Agentic AI on AWS"**

Justin J. Leto

2026 | Apress eBooks | DOI: 10.1007/979-8-8688-2199-8_1

Google Scholar

[8] **"Data Lake Design with Apache Iceberg and S3 Tables"**

Justin J. Leto

2026 | Apress eBooks | DOI: 10.1007/979-8-8688-2199-8_3

Google Scholar

[9] **"Big Data Processing and Transformation with AWS Glue and AI Agents"**

Justin J. Leto

2026 | Apress eBooks | DOI: 10.1007/979-8-8688-2199-8_5

Google Scholar

[10] **"Data Mesh Design with Amazon DataZone"**

Justin J. Leto

2026 | Apress eBooks | DOI: 10.1007/979-8-8688-2199-8_4

Google Scholar

Tekst opracowany z udziałem sztucznej inteligencji.
Redakcja i kontrola merytoryczna: Fundacja Dobre Państwo.
APA ONE Publishing System

Article ID: 3101978c-3a33-43fe-8cea-d2f9515821b3