beyond chaotic analytics

Ihre AI antwortet flüssig. Sie haften.

DER KONTEXTLAYER FÜR REGULIERTE ORGANISATIONEN

Eine produktneutrale Architektur für Glossar, Semantic Layer, Ontologie und Agentenkontext.

beyond chaotic analytics GmbH | Version 3.0 | Stand 20.07.2026
Produktangaben an diesem Datum verifiziert

Contents

1 Management Summary

Eine Bank, die in eine Datenplattform investiert hat, ist damit noch nicht AI-fähig. Der Grund ist kein technischer, sondern ein fachlicher: AI beantwortet Fragen nach Ähnlichkeit und Häufigkeit. Was «Deckungsbeitrag» in Ihrem Haus verbindlich bedeutet, wie er gerechnet wird und wer ihn sehen darf, weiss sie nicht. **Sie antwortet selbstsicher, auch wenn sie fachlich falsch liegt.** In einer regulierten Umgebung bleibt ein solches Fehlurteil nicht bei seinen eigenen Kosten: Nach dem ersten sichtbaren Fehler wird jede Zahl doppelt geprüft. **Diese Misstrauensprämie zahlt jede folgende Entscheidung mit.** Die zentrale These dieses Papiers lautet deshalb:

AI-ready ist kein Datenproblem. Es ist ein Kontextproblem.

Datenqualität, Sicherheit und Modellrisikosteuerung bleiben Pflicht; dieses Papier ersetzt keine davon. Aber wo bereits in eine Datenplattform investiert wurde, fehlt typischerweise genau eine Schicht: der Kontext. Das ist die Schicht, die drei Fragen verbindlich beantwortet, bevor die AI spricht: **Was bedeutet der Begriff? Wie wird die Zahl gerechnet? Wer darf sie sehen?** Diese Schicht heisst in diesem Papier der Kontextlayer.

Die gute Nachricht für jeden, der die Investition verantwortet: **Der Kontextlayer beginnt nicht bei null. Er macht verbindlich, was Ihr Haus schon besitzt.** Die Kennzahlendefinitionen des Controllings, die Glossare der Fachbereiche, die Berechnungslogik in Ihren Berichten, die bestehenden Berechtigungen, die Datenplattform: Aus diesen vorhandenen Artefakten entsteht die Schicht. Neu ist nicht der Inhalt. Neu ist, dass jeder Begriff einen benannten Eigentümer bekommt, an genau einem Ort gilt und von der AI nicht umgangen werden kann.

Dahinter steht eine Idee, die Ihr Haus seit dreissig Jahren verfolgt: konsistente Zahlen und verbindliche Begriffe, für alle zugänglich. Warehouse, Glossar und Lakehouse haben je einen Teil davon gelöst. **Was keine dieser Generationen gelöst hat, ist der Zugang.** AI löst ihn jetzt: Jeder im Unternehmen kann direkt fragen, ohne Werkzeugausbildung. Damit wird Konsistenz von der Fachaufgabe zur Führungsfrage, denn breiter Zugriff auf verbindliche Bedeutung ist ein Gewinn, breiter Zugriff auf zerklüftete Bedeutung skaliert die Fehlurteile.

Dass das funktioniert, zeigt Kapitel 6 an einer lauffähigen Demo für das Management Accounting einer Bank. Dort trägt **jede Antwort ihre Belege** bis zur Definition. Bei mehrdeutigen Begriffen fragt das System zurück, statt zu raten. Ohne Berechtigung verweigert es begründet, und was nicht verbindlich definiert ist, meldet es ehrlich als nicht gefunden. Verifiziert mit 478 bestandenen Tests, Stand 15.07.2026. **Die Verantwortung für Entscheidungen bleibt beim Menschen; das System bereitet sie belegt vor.**

Die Technik folgt danach, in der Tiefe, die Ihr Architekturteam braucht: das Schichtenmodell in Kapitel 3, die produktneutrale Zuordnung mit einer Referenzimplementierung auf Databricks und AWS in Kapitel 4 und 5, Varianten, Risiken und Grenzen ab Kapitel 7. Wenn Sie nur eine Sache

mitnehmen: In Kapitel 10 steht die eine Frage, die Sie vor jeder Unterschrift unter eine AI-gestützte Vorlage stellen können.

2 Ausgangslage: Der Zugang fällt, die Konsistenzfrage bleibt

Vorab eine Begriffsklärung, weil an ihr Verantwortlichkeiten hängen. Agent meint in diesem Papier die Kombination aus Sprachmodell, Orchestrierung und kontrolliertem Werkzeugzugriff. Sie beschafft unter definierten Policies Informationen und bereitet Antworten vor. Kein Bestandteil davon entscheidet; jede Komponente hat einen benannten Verantwortlichen.

Der typische Fehlerfall beginnt harmlos. Jemand fragt das neue System nach dem CIO. Die Antwort kommt flüssig und beschreibt den IT-Chef. In einer Bank ist mit CIO aber häufig der Chief Investment Officer gemeint. Wer auf Basis der falschen Rolle weiterarbeitet, baut die nächste Folie bereits auf einem Missverständnis. Dasselbe Muster bei «Security»: Das Modell versteht Zugriffsberechtigung, die Bank meint das Wertpapier. **Ein Securitytransfer bewegt Positionen, keine Zugriffsrechte.**

Das Problem dahinter ist älter als jedes Sprachmodell. Konsistente Zahlen und verbindliche Begriffe für das ganze Unternehmen zugänglich zu machen, war bereits die Idee hinter dem Data Warehouse, dem Enterprise Warehouse, dem Unternehmens-Wiki, dem Glossar und zuletzt dem Lakehouse. Jede Generation hat einen Teil gut gelöst: das Warehouse die konsistente Zahl, das Glossar den verbindlichen Begriff, das Lakehouse die Skalierung über Datenarten hinweg. **Der Zugang blieb ungelöst.** Die Lösungen blieben zerklüftet, jede mit eigener Oberfläche, eigener Pflege und eigener Nutzergruppe. Wer weder SQL noch das BI-Werkzeug beherrschte, blieb auf Vermittler angewiesen.

Genau diese Zugangsgrenze fällt jetzt. Mit natürlicher Sprache als Schnittstelle kann erstmals jeder im Unternehmen direkt fragen, ohne Werkzeugausbildung und ohne Umweg über einen Report. Das ist der eigentliche Hebel, und er schneidet in beide Richtungen. Verbreiterter Zugriff auf konsistente Bedeutung ist ein enormer Nutzen. Verbreiterter Zugriff auf zerklüftete Bedeutung skaliert die Fehlurteile mit derselben Geschwindigkeit, hinein in Rollen, die sie früher nie erreicht hätten. **Der Kontextlayer entscheidet, welche der beiden Richtungen eintritt.**

Denn das Modell findet Fakten über Ähnlichkeit. Es kennt keine Geschäftsregeln, keine Beziehungshierarchien und keine Definition dessen, was in dieser Domäne gilt. Ein grösseres Modell löst das nicht zuverlässig. Fehlender oder widersprüchlicher Kontext ist die zentrale, wenn auch nicht die einzige Ursache fachlich falscher Antworten, und er ist **die Ursache, die eine Organisation selbst beheben kann**. Retrieval liefert Nähe, die Ontologie liefert Bedeutung, der Semantic Layer liefert die korrekte Berechnung, und die Governance liefert das Vertrauen. Fehlt eine dieser Schichten, entstehen genau die belastbar aussehenden Fehlurteile, die in einer Bank nicht tragbar sind.

Der regulatorische Rahmen verschärft die Anforderung. Für eine FINMA-beaufsichtigte Bank gelten das Bankkundengeheimnis und die institutsspezifischen Residenz- und Transferanforderungen; sie ergeben sich aus dem revidierten Datenschutzgesetz, insbesondere

der grenzüberschreitenden Bekanntgabe, sowie aus Auslagerungs- und Risikovorgaben. Für Risikodaten kommen die Grundsätze von BCBS 239 hinzu, dem Standard des Basler Ausschusses für Bankenaufsicht zur Aggregation von Risikodaten und zur Risikoberichterstattung: Risikozahlen müssen genau, vollständig, aktuell und bis zur Quelle nachvollziehbar sein. Formal bindet der Standard die grossen, systemrelevanten Institute; welche Domänen im eigenen Haus nach denselben Grundsätzen behandelt werden, legt die Risikoklassifikation des Instituts fest. Daraus folgt eine harte Bedingung: **Jede Zahl, die ein Agent in eine Entscheidungsvorbereitung einspeist, muss als vollständiger Nachweis rekonstruierbar sein.** Dazu gehören Definition, Quelle, Berechnungsversion, Abfragezeitpunkt, Zugriffsentscheid und gegebenenfalls die Freigabe. Und jeder Zugriff muss prüfbar und regionskonform ablaufen. Zur Klarheit des Anspruchs: Diese Architektur liefert Information, Empfehlung und Entscheidungsvorbereitung. Autonome Aktionen liegen ausserhalb ihres Geltungsbereichs.

Der wirtschaftliche Hebel lässt sich modellieren. Ein strategischer Entscheid über einen zweistelligen Millionenbetrag hängt nach einem sichtbaren Zahlenfehler drei Monate in der Nachprüfung; die Wirkung des Kapitals verschiebt sich um ein Quartal. Je nach Vorhaben sind das mehrere hunderttausend bis einige Millionen Franken pro Monat, als Bandbreite gerechnet, nicht als Punktwert. Ein Kontextlayer, der Nachvollziehbarkeit nicht von Beginn an mitträgt, ist deshalb kein Beschleuniger. **Er ist ein Risiko mit Preisschild.**

3 Vier Schichten, vier verschiedene Fragen

Die häufigste Verwechslung im Markt ist die Gleichsetzung von Semantic Layer, Ontologie und Knowledge Graph. Das ist kein Schönheitsfehler. Es führt zu Architekturen, in denen eine Schicht die Aufgaben einer anderen übernehmen soll und an beiden scheitert. Das folgende Modell hält die Schichtung präzise.

Grafik 1: Die vier Schichten in sauberer Trennung, jede mit eigener Frage und eigenem Adressaten.

Rohdaten sind Tabellen, Spalten und Events: maschinenlesbar, aber ohne Geschäftsbedeutung. **Das Glossar** ist die Schicht der fachlichen Definition: Es definiert Kennzahlen und Dimensionen verbindlich, mit einem Owner je Begriff, und bei Kennzahlen gehört die fachliche Formel dazu. **Der Semantic Layer** ist die Schicht der Messung: Er setzt die Formel als konkrete, geprüfte Berechnung um und bringt Kennzahlen und Dimensionen zusammen, konsistent für alle Konsumenten, etwa Deckungsbeitrag II pro Kunde und Produkt. **Die Ontologie** ist die Schicht der Bedeutung; sie sagt nicht, wie man eine Kennzahl rechnet, sondern was ein Kunde, ein Vertrag und eine Gegenpartei sind und wie sie zusammenhängen. **Der Knowledge Graph** ist die mit realen Daten gefüllte Instanz dieser Ontologie. Und **der Kontextlayer** legt dem Agenten je Frage den richtigen Ausschnitt aus Bedeutung und operativem Zustand vor.

Ein Beispiel aus dem Management Accounting macht die Arbeitsteilung greifbar. «Deckungsbeitrag» ist im Glossar ein mehrdeutiger Oberbegriff mit einer Familie konkreter Kennzahlen dahinter; «Deckungsbeitrag III» ist dort eine definierte Kennzahl mit fachlicher Formel, Abhängigkeiten und einem Owner. Der Semantic Layer setzt diese Formel als Berechnung um und verbindet sie mit den Dimensionen: Deckungsbeitrag III pro Produkt und Geschäftseinheit

liefert **dieselbe Zahl, egal wer fragt** und mit welchem Werkzeug. Die Ontologie kennt die Beziehung zwischen Kennzahl, Produkt, Geschäftseinheit und Policy. Und der Kontextlayer entscheidet, welcher Ausschnitt davon für Ihre konkrete Frage relevant ist, in Ihrem Zugriffskontext.

Ähnlichkeit ist nicht Checking

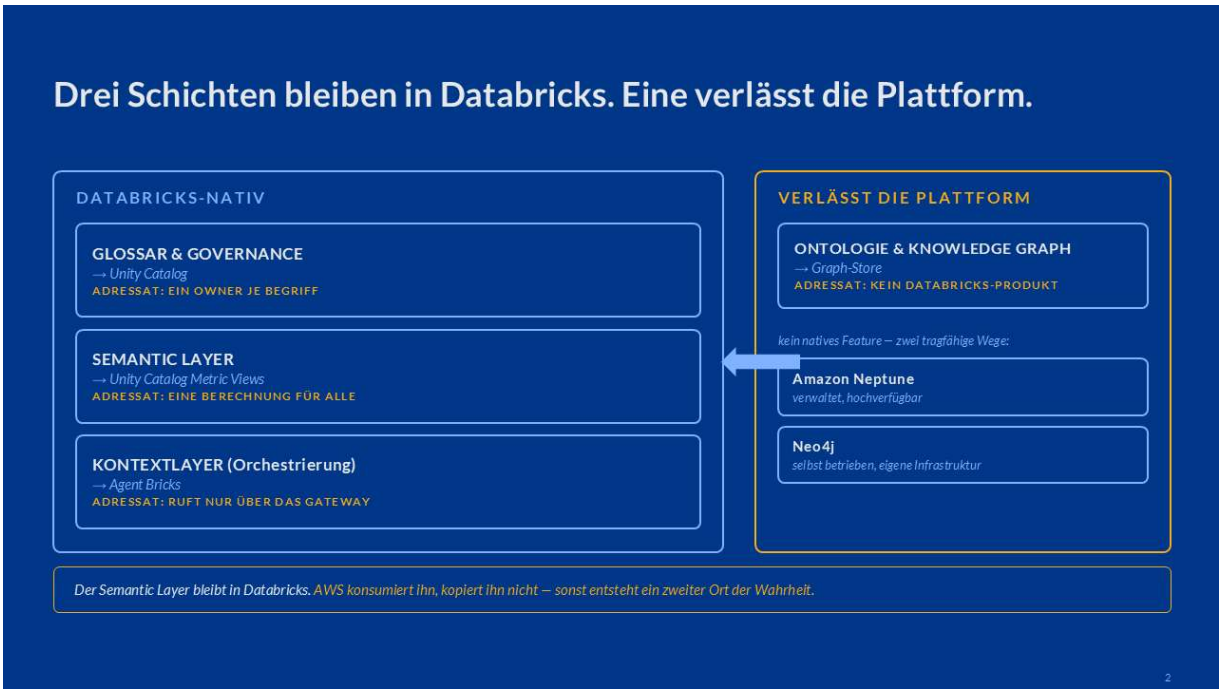
Diese Unterscheidung trägt das ganze Papier. Eine agentische, selbstlernende Ontologieschicht ordnet Bedeutung nach Ähnlichkeit und Häufigkeit: Sie identifiziert die verbreitetste Definition und rankt sie nach Popularität. **Ähnlichkeit ist nicht Checking.** Für eine regulierte Bank ist das an den risikorelevanten Stellen zu wenig, denn **Häufigkeit ist nicht Korrektheit.**

Die Prüfung selbst besteht aus zwei getrennten Schritten, die nicht austauschbar sind. Ein OWL-Reasoner zieht logische Schlüsse und prüft das Klassenmodell auf Konsistenz. SHACL validiert die Instanzdaten gegen Constraints, also gegen definierte Pflichtregeln. Der Unterschied ist grundsätzlicher, als er klingt. Der Reasoner arbeitet unter der Annahme der offenen Welt: Was nicht bekannt ist, gilt nicht als falsch, sondern als unbekannt. SHACL prüft geschlossen, was vorhanden ist, gegen das, was vorhanden sein muss. Wer nur den Reasoner einsetzt, wundert sich, warum fehlende Pflichtangaben durchgewinkt werden. Wer nur SHACL einsetzt, entdeckt Widersprüche im Klassenmodell nicht. *Erst beide zusammen ergeben Prüfung.* Beide sind eigene, bewusst zu verantwortende Komponenten und kein Nebenprodukt eines Kontextgraphen. Wer das Ordnen nach Ähnlichkeit und Häufigkeit mit diesen Prüfschritten verwechselt, kauft sich Komfort ein und bezahlt an den risikorelevanten Stellen mit der Prüffähigkeit.

4 Die Referenzimplementierung: Databricks und AWS

Zuerst das Prinzip: **Die Architektur ist herstellerneutral.** Glossare, Semantic Layer, Graph-Stores und Gateways gibt es von mehreren Anbietern, und die Regeln aus Kapitel 3 und 5 gelten unabhängig vom Logo. Wer andere Produkte einsetzt, ersetzt die Bausteine und behält die Regeln. Dieses Kapitel zeigt eine Referenzimplementierung für die verbreitete Kombination aus Databricks und AWS, weil sich Autorität, Grenzen und Residenzfragen am konkreten Fall schärfer zeigen als am Diagramm.

Drei der vier Schichten stellt Databricks in dieser Kombination erstklassig, die vierte ist nicht seine native Stärke: Es gibt keine native, spezialisierte Graphdatenbank mit eigener Abfragesprache und Traversal-Dienst, also einem Dienst für das schnelle Durchwandern von Beziehungsketten. Die folgende Zuordnung zeigt, wer welchen Begriff besitzt und wo die Grenze verläuft.



Grafik 2: Zuordnung der Schichten zu Databricks-nativen und AWS-nativen Diensten.

Das Glossar und die Governance gehören in den Unity Catalog, weil Begriffe, Herkunftsnachweise und Zugriffspolitik dort mit den Daten an einem Ort sitzen. Der Semantic Layer ist auf der angenommenen Databricks-Zielplattform mit den Unity Catalog Metric Views am besten aufgehoben, den zentral definierten und versionierten Kennzahl-Sichten. Im betrachteten AWS-nativen Stack wurde kein funktional gleichwertiges, zentral gesteuertes Gegenstück identifiziert. Diese Schicht wandert nicht auf die AWS-Seite; sie wird von dort nur konsumiert. Wer sie

verschiebt, kauft sich **einen zweiten Ort der Wahrheit** ein, und damit genau den Zahlenstreit, den die Architektur beenden soll.

Warum der Graph Databricks verlässt

Databricks hat keinen nativen Graph-Store und keine native Graph-Abfragesprache. In der Praxis behilft man sich mit der Bibliothek GraphFrames auf klassischen Clustern. Auf Serverless-Compute ist dieser Weg wegen technischer Abhängigkeiten der Bibliothek nur eingeschränkt nutzbar und keine verlässliche Zielarchitektur. Eine Mehrschritt-Traversierung über Millionen Kanten liesse sich zwar iterativ über SQL nachbilden. Für interaktive Abfragen mit vorhersehbar niedriger Latenz ist das **operativ ungeeignet**.

Für den eigenen Graph-Store gibt es zwei tragfähige Varianten, und sie sind kein Widerspruch, sondern ein Anforderungsentscheid. **Amazon Neptune** ist der verwaltete Cloud-Weg: hochverfügbar ohne eigenen Betriebsaufwand, mit beiden Graph-Welten an Bord (Property Graph über openCypher, RDF über SPARQL). Für eine an FIBO ausgerichtete Ontologie, also an der offenen Finanzbranchen-Ontologie, ist das der direktere Pfad. **Neo4j** ist der selbst betriebene Weg: in der eigenen Infrastruktur lauffähig, mit Cypher als ausgereifter Abfragesprache und breitem Werkzeug-Ökosystem; die RDF-Welt wird über Erweiterungen angebunden statt nativ. Entschieden wird über fünf Kriterien: Residenz, Betriebsmodell, Ontologie-Pfad, Fähigkeiten im Haus, Gesamtkosten über den Lebenszyklus. Wer den Graph-Store vollständig in der eigenen Infrastruktur halten muss oder die passende Cloud-Region nicht bekommt, landet eher bei Neo4j. Wer verwalteten Betrieb in der richtigen Region bekommt, eher bei Neptune.

Bei einer FIBO-Ausrichtung ist das kanonische Modell damit festgelegt: RDF/OWL als führende Repräsentation, SPARQL als Abfragepfad. Ein Property Graph existiert nur als abgeleitete Projektion für spezifische Auswertungen, nie als zweite gepflegte Wahrheit; sonst entsteht Doppelpflege mit Drift. FIBO selbst wird nie als Ganzes importiert. Die Ontologie ist modular und in voller Breite weder ladbar noch wartbar. Tragfähig ist ein versioniertes Subset, zugeschnitten über Kompetenzfragen: Welche Fragen muss der Graph beantworten können? Jede Klasse, die keine dieser Fragen stützt, bleibt draussen. **Das Subset wächst mit den Fragen, nicht mit dem Ehrgeiz.** Ein Punkt gilt für beide Stores: Keiner bringt einen eingebauten OWL-Reasoner mit. Reasoning und SHACL-Validierung sind zwei bewusst hinzuzufügende Bausteine, kein Automatismus.

Gemanagtes GraphRAG gegen handmodellierten Graphen

AWS bietet für die oberste Schicht zwei Wege, die man nicht verwechseln darf. Amazon Bedrock Knowledge Bases stellt ein vollständig gemanagtes GraphRAG bereit: Es extrahiert aus eingelesenen Dokumenten automatisch Entitäten und Beziehungen und legt den Graphen in Neptune Analytics ab. Das ist bequem, hat aber genau die Eigenschaft, die hier stört. Der Graphaufbau bietet laut Dokumentation keine Konfigurationsoptionen; wählbar ist nur das Konstruktionsmodell. Zielschema, Extraktionslogik und fachliche Freigabe bleiben ausserhalb der

Kontrolle, die Entitäten und Beziehungen entstehen modellbasiert. **Modellbasierte Extraktion ersetzt weder die formale Konsistenzprüfung noch die fachliche Freigabe.** Genau diese beiden braucht die regulierte Domäne.

Der auditierbare Weg ist der handmodellierte Graph in Neptune Database, gespeist aus den verbindlichen Definitionen des Unity Catalog und geprüft gegen die formale FIBO-OWL-Schicht. Der Aufbaupfad ist bewusst einfach gehalten, damit er prüffähig bleibt. Für den laufenden Betrieb kommen Änderungsdaten, Idempotenz, Identitätsauflösung und Versionierung hinzu, siehe Kapitel 9.

Grafik 3: Der Antwortpfad.

Die Residenzfrage in der Region Zürich

Für eine Bank mit Bankkundengeheimnis ist die Region kein Detail, sondern ein Gate. Und hier sind Produkt- und Featureverfügbarkeit sauber zu trennen; alle Angaben sind am 20.07.2026 verifiziert. Neptune Database und Neptune Analytics sind in der Region Europe/Zürich verfügbar, Neptune Analytics seit Februar 2026. **Ein Schweizer Graph-Store auf AWS ist damit technisch möglich.** Das gemanagte GraphRAG-Feature ist dagegen laut Feature-Regionsliste **in Zürich nicht freigeschaltet**; verfügbar ist es unter anderem in Frankfurt, London und Irland. Wer den gemanagten Pfad für schützenswerte Bestände nutzen wollte, würde kundenidentifizierende Daten aus der Schweiz herausführen.

Auch die Inferenzseite verlangt Präzision. Ein Bedrock-Endpoint in Zürich bedeutet nicht, dass jedes Modell und jedes Feature ausschliesslich in Zürich verarbeitet. Geografische Cross-Region-Inference kann Prompts und Ergebnisse innerhalb einer Geografie weiterleiten. Massgeblich sind das konkrete Modell, der Inferenzmodus und die Zielregion, einzeln geprüft, nicht die Serviceebene. Diese Punkte bewegen sich monatlich; ein einmaliger Stand genügt nicht. Zur Architektur gehört ein wiederkehrendes Verifikationsverfahren mit definierten Prüfpunkten und festem Intervall, verankert beim Plattform-Owner. Und es gibt eine zweite Antwort auf die Residenzfrage, die keine Regionsliste braucht: Ein selbst betriebener Neo4j in der eigenen Schweizer Infrastruktur erfüllt die Residenz per Konstruktion. Er bezahlt dafür mit eigenem Betriebs- und Verfügbarkeitsaufwand. Auch das ist ein Anforderungsentscheid, kein Glaubenssatz.

5 Governance: eine Tür, ein Protokoll, getrennte Autoritäten

Die Ontologie von Hand zu modellieren löst das Bedeutungsproblem. Die nächste Frage ist, wie der Databricks-Agent diesen Graphen nutzt, ohne die Governance in eine zweite Kontrollebene zu zersplittern. Der Weg führt nicht über einen Import des Graphen in Databricks, sondern über einen kontrollierten Werkzeugzugriff.

Das Prinzip ist wieder produktneutral: eine Tür, ein Protokoll. Vor dem Graph-Store steht ein MCP-Server, eine standardisierte Werkzeug-Schnittstelle; in der Referenzimplementierung wird er im Unity Catalog als geschütztes Objekt registriert, und die Tür heisst Unity AI Gateway. **Der Agent erreicht den Graphen ausschliesslich über dieses Gateway.** Jeder Aufruf läuft durch denselben geprüften Pfad: Das Gateway prüft das Ausführungsrecht, blendet nur freigegebene Werkzeuge ein, wertet eine Service-Policy aus (erlauben, verweigern, Freigabevorbehalt) und protokolliert jeden Aufruf in den System-Tabellen. Die Zugangsdaten hält ein verwalteter Proxy; der Agent sieht sie nie. Zwei Identitätsgrenzen sind dabei getrennt zu führen: die vom Agenten zum Gateway auf Databricks-Seite, und die von der MCP-Laufzeit zu AWS, die eine eigene Identität mit minimalen Berechtigungen braucht. Netzwerk, private Anbindung und Secrets sind Teil der Detailplanung, siehe Kapitel 9. Und der Pfad hat einen Preis, der benannt gehört: Jeder Werkzeugaufruf überquert eine Plattformgrenze. Ein Latenzbudget je Antworttyp, als Bandbreite definiert und gemessen, ist deshalb Freigabebedingung der Architektur und nicht Feinschliff.

Grafik 4: Der Laufzeitpfad. Der Agent erreicht den Graphen nur über das Gateway, das jeden Werkzeugaufruf prüft und protokolliert.

Ein ehrlicher Einwand gehört hierher. Wer eine agentische Ontologie als Black Box ablehnt und dann eine stark automatisierte Agentenschicht einsetzt, verschiebt die Black Box, statt sie aufzulösen. Der Unterschied, der die Entscheidung trotzdem trägt, liegt in der Frage, was automatisiert wird. Hier ist es die Orchestrierung, nicht die Bedeutung. Solange der gelieferte Kontext deterministisch, versioniert und regelgeprüft ist und jeder Aufruf über das Gateway läuft, ist die Automatik tragbar: Sie kann die Grammatik nicht mehr anfassen.

Praktisch heisst das zweierlei. Erstens wird für risikorelevante Domänen verbindlich festgelegt, welche Abfragen als Werkzeuge freigegeben sind und nach welchen Regeln sie laufen. Das ist mehr als eine Toolauswahl, es ist das Regelwerk je Werkzeug. Der Graph wird nicht als offenes Abfragefeld freigegeben, sondern als benannte, parametrisierte, versionierte Abfragen, erzwungen über die Service-Policy. Zum Regelwerk gehören Parametergrenzen, Traversaltiefen, Timeouts, Result Limits und ein definiertes Fehlerverhalten. Zweitens bleibt die Autorität getrennt, mit einer wichtigen Präzisierung: **Der Graph ist nicht das führende System für Geschäftsfakten.** Operative Fakten bleiben in den Quellsystemen und im kuratierten Lakehouse autoritativ. Der Graph hält die versionierte, abgeleitete Beziehungssicht, deren Aktualität messbar sein muss. Für den Laufzeitkontext braucht es eine eigene Autorität: eine Registry für den Werkzeugkatalog, seine Regeln und die Policy-Versionen, mit benanntem Owner. Denn das Gateway ist ein Kontrollpunkt, keine fachliche Autorität.

Artefakt	**Autoritative Instanz**	**Laufzeit / Konsument**
Operative Geschäftsfakten	Quellsysteme, kuratiertes Lakehouse	Graphprojektion, Agenten
Berechnungsimplementierung (Kennzahl mal Dimensionen)	Unity Catalog Metric Views	Agenten, BI, Anwendungen
Glossar: Kennzahlen mit fachlicher Formel, Dimensionen	Unity Catalog, ein Owner je Begriff	Ontologie-Mapping, Agenten
Ontologie (FIBO-Subset)	Versioniertes Ontologie-Repository	Deployment in den Graph-Store
Constraints	Versioniertes SHACL-Repository	Validierungspipeline
Graphprojektion	Graph-Store (Neptune oder Neo4j), abgeleitet und versioniert	Parametrisierte Werkzeugabfragen

Artefakt	Autoritative Instanz	Laufzeit / Konsument
Werkzeugkatalog, Regeln und Policies	Tool- und Policy-Registry mit benanntem Owner	AI Gateway, MCP-Dienst
Audit-Evidenz	Unveränderbare Logging-Schicht	Audit, Modellrisiko, Betrieb

Tabelle 1: Autoritätsmodell in der Referenzimplementierung. Die Produktnamen sind austauschbar, die Trennung von Repository und Laufzeit je Artefakt ist es nicht.

Versionierung gilt in diesem Modell doppelt. Das Ontologie-Repository versioniert das Schema; der Graph-Store hält Instanzstände getrennt referenzierbar, in RDF über benannte Graphen. Erst diese Doppelung macht den entscheidenden Prüffall möglich: **Eine Antwort von gestern wird gegen den Wissensstand von gestern geprüft**, nicht gegen den von heute.

6 Der Prüfstein: eine lauffähige Demo

Architekturen auf Folien halten jeder Diskussion stand. Deshalb wurde das Schichtenmodell als lauffähige Demo für das Management Accounting einer Bank umgesetzt; sie ist der Prüfstein dieses Papiers, an ihr wird geprüft, ob die Regeln tragen: eine Antwortpipeline mit React-Frontend und FastAPI-Backend über einer Databricks-Wertschicht und einem Kontextgraphen. Als Graph-Store verwendet die Demo Neo4j und belegt damit die selbst betriebene Variante aus Kapitel 4. Für diese Prüfung zählt die Architekturregel, nicht das Produkt: Der Graph-Store ist eine austauschbare Komponente. Dieselbe Pipeline lässt sich mit überschaubarem Aufwand auf Neptune stellen. Genau diese Variantenfähigkeit ist Teil des Beweises. **Wer Architekturvarianten nur auf Folien vergleicht, diskutiert. Wer sie als lauffähige Demos vergleicht, ermöglicht Entscheide.**

Die Demo trennt die Datenhoheit strikt: je Inhalt genau eine Quelle. Der Graph hält Begriffe, Definitionen, Synonyme, Abhängigkeiten, Policies und Mehrdeutigkeit. Databricks hält die Werte: Metric Views rechnen die Kennzahlen, Unity Catalog erzwingt den Zugriff über profilgebundene Service Principals und Dynamic Views, also zeilenweise gefilterte Sichten je Rolle. Kein System übernimmt die Rolle des anderen. **Der Graph enthält keine einzige Finanzzahl.**

Der entscheidende Mechanismus ist die **Evidenzpflicht**. Jede Antwort entsteht aus einem Evidence Pack: Es bündelt alle Belege einer Antwort in definierten, versionierten und schemageprüften Feldern (in der Demo Version 1.0.0): die aufgelöste Kennzahl, ihre Definition, das Mapping auf die Berechnungsquelle, die Policy-Lage, den beobachteten Zugriffsumfang, die Herkunft und die festgestellten Lücken. Ein Critic prüft jede Aussage gegen diese Belege; eine einzige finale Entscheidung steuert die Ausgabe. Der Composer formuliert ausschliesslich aus dem validierten Evidence Pack. **Er erfindet keine Fakten, denn er bekommt keine Gelegenheit dazu.**

Das sichtbarste Ergebnis sind die Antworttypen. Sie folgen keiner Stimmung, sondern einer formalen Entscheidungslogik:

Signal	**Regel**	**Antworttyp**
Eindeutiger Treffer im verbindlichen Katalog	Definition und Belege ausliefern	Belegte Antwort
Schreibvariante («Deckungsbeitrag 3»)	Normalisierung auf die kanonische Kennzahl	Belegte Antwort, identisch
Mehrere gleichwertige Kandidaten	Kein Wert-Abruf, Bedeutungen vorlegen	Kontrollierte Rückfrage

Signal	Regel	Antworttyp
Policy oder Zugriff nicht gedeckt	Kein Wert, Begründung aus der Policy-Lage	Begründete Verweigerung
Kein Treffer im verbindlichen Katalog	Kein Näherungsversuch	Ehrliches Nichtgefunden

Tabelle 2: Entscheidungslogik der Antworttypen in der Demo. Schwellen und Regeln sind versioniert, nicht im Anwendungscode.

Ein System, das an den richtigen Stellen zurückfragt, verweigert und schweigt, ist im regulierten Umfeld mehr wert als eines, das immer antwortet.

Der Stand ist verifiziert, nicht behauptet, und die Zahlen haben einen definierten Messkontext. Die 478 bestandenen Backend-Tests umfassen Unit- und Integrationstests sowie Architektur-Gates, die Fachlogik im Anwendungscode verbieten. 24 Frontend-Tests prüfen die Evidenz-Darstellung. Die 18 Live-Fragen bilden ein **Golden-Question-Set**: Je Frage ist der erwartete Antworttyp definiert, und bestanden heisst, dass Typ und Belege übereinstimmen. Deploys für Graph-Seeds und SQL sind idempotent, also beliebig wiederholbar mit gleichem Ergebnis. Änderungen am Graphen laufen als versionierte Seeds durch Review und dieselben Test-Gates wie Code. Geprüft am 15.07.2026.

Ebenso dokumentiert sind die Grenzen. Starke Tippfehler ohne Entsprechung im Graphen führen zu Nichtgefunden statt zu einer unscharfen Vermutung. Es gibt kein sitzungsübergreifendes Gedächtnis. Die Demo nutzt Demo-Profile statt Produktions-SSO. Diese Grenzen sind ausgewiesen, weil eine Demo, die ihre eigenen Ränder kennt, mehr beweist als eine, die alles kann.

7 Drei Varianten, eine Empfehlung, eine ehrliche Gegenrichtung

Die Architektur lässt sich in drei Ausprägungen führen. Sie unterscheiden sich im Verhältnis von Kontrolle und Geschwindigkeit und werden je Institutsprofil gewählt, nicht pauschal.

Variante	Aufbau	Stärke	Aufwand
Auditierbarer Zwilling	Regelgeprüfter Graph auf Neptune oder selbst betriebenem Neo4j, gegen FIBO-OWL und SHACL geprüft, kein gemanagtes GraphRAG	Maximale Prüffähigkeit und Residenz	Mehr Modellierungsarbeit
Hybrid	Regelgeprüfter Graph für kritische Domänen, GraphRAG als Komfortschicht für unkritische Bestände	Bester Kompromiss aus Tempo und Kontrolle	Harte Trennlinie nötig
Schlanker Einstieg	Glossar, Semantic Layer, Begriffsnetz und Antworttyp-Logik über einen bewusst kleinen Ausschnitt, voller Prüfstandard	Schnell startklar, günstig	Kleiner Umfang, öfter Nichtgefunden

Tabelle 3: Die drei Architekturvarianten. Der auditierbare Zwilling ist ein regelgeprüfter Fachgraph, kein Digital Twin im Simulationssinn.

Empfehlung. Beginnen Sie schlank, aber nie unter Standard. **Der schlanke Einstieg begrenzt den Umfang, nicht die Qualität:** Glossar, Semantic Layer, ein kleines Begriffsnetz für Synonyme und Mehrdeutigkeit sowie die Antworttyp-Logik decken einen bewusst kleinen Ausschnitt ab, etwa eine einzige Kennzahlenfamilie. Innerhalb dieses Ausschnitts gilt der volle Prüfstandard, jede Antwort trägt ihre Belege. Ausserhalb antwortet das System ehrlich mit «Nichts gefunden», und diese Antwort wird mit jedem Ausbauschritt seltener. Für die Domänen, deren Zahlen Ihre Bank nach ihrer eigenen Risikoklassifikation als risikorelevant führt, kommt der auditierbare Zwilling dazu: der von Hand modellierte, gegen formale Regeln geprüfte Graph, mit festgelegten Werkzeugen samt Grenzen und genau einem führenden System je Inhalt (Kapitel 5, Tabelle 3). Das automatische GraphRAG bleibt eine optionale Ergänzung für Bestände ohne Kundenbezug,

sobald es in der Region Zürich verfügbar ist. **So wächst der Umfang, während der Standard konstant bleibt.**

Zur Ehrlichkeit der Empfehlung gehört eine Präzisierung: **Der Graph ist keine spätere Ausbaustufe, er ist vom ersten Tag an dabei, nur in anderer Tiefe.** Überall dort, wo Daten im Kontext gelesen werden müssen, entscheidet er mit: «Bank» kann das Institut oder die Gegenpartei meinen, hinter «Konto» steht oft das Portfolio, und wer Kunden zählt, ohne die Haushalte zu kennen, zählt falsch. Synonyme, Mehrdeutigkeit und die Zuordnung von Begriff zu Kennzahl sind Graphwissen; ohne dieses Begriffsnetz gibt es keine kontrollierte Rückfrage, sondern nur Raten. In der Demo trägt genau dieses Netz die Rückfrage-Logik. Was dagegen entlang echter Kompetenzfragen wachsen darf, ist die zweite Ausbaustufe: die formale Ontologie mit gefüllten Instanzen, Regelprüfung und Schlussfolgerung, unverzichtbar bei Policy-Vererbung, Abhängigkeiten zwischen Kennzahlen und Herkunft über Systemgrenzen. **Klein anfangen heisst beim Graphen: wenige Begriffe, nicht keine.**

8 Fünf Punkte vor der Festlegung

Erstens der Reifegrad, je Funktion verschieden. Externe MCP-Dienste, das Unity AI Gateway und die Service-Policies laufen als Beta, die verwalteten MCP-Server als Public Preview; Agent-Bricks-Komponenten tragen eigene Status (Stand gemäss Review-Verifikation vom 20.07.2026, vor Freigabe je Funktion gegenzuprüfen). Für einen Prototyp ist das tragbar. Für den produktiven Einsatz braucht es je Funktion Status, Support-Zusage und einen ausgearbeiteten Rückfallpfad, etwa über eine gekapselte Unity-Catalog-Funktion, dessen Gleichwertigkeit bei Zugriff, Policy-Durchsetzung und Protokollierung nachgewiesen ist.

Zweitens die Residenz. Der Werkzeugausgang des Graphen fliesst durch das Agentenmodell. Damit ist die Region des Serving-Endpoints Teil der Residenzfläche, nicht nur der Store. Modell-Serving und Graph-Store gehören in dieselbe Region.

Drittens die Drift zwischen Definition und gelebter Datenrealität. Sie bleibt nur durch benannte Ownership und laufende Pflege beherrschbar.

Viertens der Betrieb. Monitoring, Alerting, Runbooks und der Pflegeaufwand je neuer Kennzahl gehören in die Gesamtrechnung. Die Demo zeigt: Jede neue Kennzahl läuft durch Graph-Seed, Berechnungslogik und Tests. Das ist gewollte Disziplin, und sie kostet Kapazität, die eingeplant gehört.

Fünftens das Modellrisiko. Technische Auditierbarkeit ersetzt keine fachliche Modellvalidierung und keine menschliche Aufsicht. Je Use-Case-Klasse (Information, Empfehlung, Entscheidungsvorbereitung) braucht es ein Freigabeverfahren, definierte Eskalation und periodische Revalidierung. Die Verantwortung für die Entscheidung bleibt in jeder Klasse beim

Menschen. **Eine grüne Prüfkette verschiebt diese Verantwortung nicht, sie macht die Freigabe belegbar.**

Die Kontrollsicht des Prüfers

Ein Prüfer liest keine Architekturdiagramme, er liest Kontrollen. Die Matrix zeigt den Zuschnitt; vollständig ausgearbeitet und durch Risk und Compliance bestätigt wird sie im Architekturkapitel je Institut:

Anforderung	**Architekturkontrolle**	**Evidenz**	**Owner**
Nachvollziehbarkeit jeder Zahl	Evidence Pack mit Definition, Quelle, Version, Zeitpunkt	Versionierter Nachweis je Antwort (in der Demo Contract v1.0.0)	Fachbereich
Zugriffsschutz je Rolle	Profilgebundene Principals, Dynamic Views, Gateway-Policies	Zugriffs- und Aufrufprotokolle (System-Tabellen)	Informationssicherheit
Menschliche Freigabe	Antworttyp-Klassen, Freigabevorbehalt in der Service-Policy	Freigabeprotokoll mit benannter Person	Prozess-Owner
Bestandssicherung	Unveränderbare Logging-Schicht, Aufbewahrung nach handelsrechtlichen Fristen	Revisionssicheres Audit-Log	Compliance
Auslagerungssteuerung	Cloud-Nutzung als Auslagerung mit Prüf- und Auditrechten gemäss FINMA-Vorgaben	Vertrags- und Kontrollnachweise je Anbieter	Auslagerungsverantwortlicher

Tabelle 4: Kontrollmatrix im Zuschnitt. Anforderung, Kontrolle, Evidenz und Owner gehören in jeder Zeile zusammen.

9 Was dieses Papier leistet und was danach kommt

Ehrlichkeit über den Zuschnitt gehört in ein Architekturpapier. Dieses Papier begründet das Schichtenmodell, die Plattformzuordnung, das Autoritätsmodell und den kontrollierten Werkzeugzugriff. Und es belegt die Mechanik an der lauffähigen Demo. Eine freigabefähige Zielarchitektur braucht darüber hinaus Bausteine, die hier bewusst nur benannt werden. Ausgearbeitet werden sie im Architekturkapitel im BCDA-Format.

Drei Bausteine gehören vor jeden Produktionsstart geklärt. Die laufende Synchronisation zwischen Quellsystemen und Graph, mit Änderungsdaten, Idempotenz und messbarer Aktualität. Die Identitätsauflösung für Kunden, Verträge und Gegenparteien; an ihr scheitern Knowledge Graphs in der Praxis häufiger als an der Traversierung. Und das vollständige Threat Model samt Modellrisiko-Verfahren. Die wichtigsten Angriffsflächen adressiert die Architektur bereits: parametrisierte Abfragen gegen Injektion über Werkzeuge, Result Limits gegen überbreite Resultate, getrennte Identitätsgrenzen gegen ausgeweitete Berechtigungen, der Werkzeugkatalog im Gateway gegen nicht freigegebene Zugriffe. Offen bleiben unter anderem Prompt Injection über Eingaben und Datenabfluss über zulässige Antworten.

Die übrigen Bausteine tragen vor der Skalierung: messbare Abnahmekriterien statt des Anspruchs «weniger Halluzination» (Korrektheitsquote, Belegtreue, Verweigerungsrate auf einem Golden-Question-Set); nichtfunktionale Anforderungen mit Latenz-, Verfügbarkeits- und Kostenzielen; der Lebenszyklus der Ontologie; das Betriebsmodell mit Rollen; die Netzwerk- und Secrets-Architektur; ein Exit-Design, das die Semantik exportierbar hält. Jede dieser Positionen taucht in der Gesamtrechnung auf: **als geplanter Baustein, oder später als ungeplanter Vorfall.**

10 Die Frage vor der Unterschrift

Wie diese Prüfung im Alltag aussieht, beginnt mit einer einzigen Frage. Sie können sie in jedem Sitzungszimmer stellen, bevor Ihr Name unter einer AI-gestützten Vorlage steht:

«Führt dieselbe Frage nächste Woche zu derselben Zahl, und wer im Raum kann belegen, aus welcher Definition sie sich herleitet?»

Die erste Hälfte prüft, ob die Herleitung über Zeit und Personen hält. Die zweite prüft, ob die Belegkette existiert, bevor sie gebraucht wird. **Der Kontextlayer in diesem Papier ist die Architektur, die beide Hälften mit Ja beantwortbar macht.**

Ihr nächster Schritt. Sie brauchen dafür kein Projekt und kein Budget, sondern eine Stunde. Stellen Sie die Frage aus diesem Kapitel in der nächsten Sitzung, in der eine AI-gestützte Vorlage auf dem Tisch liegt. Und machen Sie den Test im eigenen Haus: Fragen Sie Ihr System nach Ihrer wichtigsten Kennzahl, einmal präzise, einmal bewusst unpräzise, einmal ohne die nötige Berechtigung. Kommt dreimal eine flüssige Zahl, kennen Sie jetzt Ihr Risiko. Kommt eine Definition, eine Rückfrage und eine begründete Verweigerung, sind Sie weiter als die meisten. Wenn Sie Ihr Ergebnis besprechen wollen, schreiben Sie uns. Wir zeigen Ihnen in einer Stunde an der lauffähigen Demo, wie eine belegte Antwort aussieht und wie eine begründete Verweigerung, und Sie nehmen die Prüfpunkte für Ihre eigene Architektur mit.

11 Quellen und Stand

Dieses Papier stützt sich auf vier Quellengruppen. Die Tabelle zeigt je Gruppe, was aus ihr stammt, auf welchem Stand sie ist und wie sie geprüft wurde.

Quelle	Was daraus stammt	Stand	Prüfstatus
Fachliteratur zum Enterprise Context Layer	Die Begriffsarchitektur: Semantic Layer, Ontologie, Knowledge Graph, Kontextlayer	Juli 2026	Konzeptionell, nicht zeitkritisch
Databricks: Ankündigungen der Data + AI Summit 2026 und Produktdokumentation	Funktionsumfang von Unity Catalog, Metric Views, Agent Bricks, verwalteten MCP-Servern und AI Gateway	Juli 2026	Die Status-Angaben (Beta, Public Preview) stammen aus einer externen Prüfung vom 20.07.2026 und sind vor der Freigabe je Funktion gegen die

Quelle	Was daraus stammt	Stand	Prüfstatus
			Dokumentation zu prüfen
AWS-Produktdokumentation	Regionsverfügbarkeit und Inferenzverhalten	20.07.2026	Selbst verifiziert: GraphRAG-Regionsliste ohne Zürich; Neptune Database (seit Dezember 2025) und Neptune Analytics (seit Februar 2026) in Zürich verfügbar; Cross-Region-Inference dokumentiert
Interne Verifikationsdokumentation der Demo	Testzahlen, Antworttypen, Deploy-Verhalten (Kapitel 6)	15.07.2026	Vollständiger Abnahmelauf, Ergebnisse einzeln protokolliert

Tabelle 5: Quellengruppen mit Stand und Prüfstatus.

Für alle zeitgebundenen Angaben gilt: Sie veralten monatlich. Vor einer Veröffentlichung oder produktiven Festlegung wird jede von ihnen neu geprüft; zuständig ist der Plattform-Owner mit dem wiederkehrenden Verifikationsverfahren aus Kapitel 4.

12 Glossar

Agent | In diesem Papier die Kombination aus Sprachmodell, Orchestrierung und kontrolliertem Werkzeugzugriff. Bereitet Antworten vor, entscheidet nicht.

Agent Bricks | Databricks-Baukasten zum Erstellen und Betreiben von Agenten auf der Plattform.

BCBS 239 | Standard Nr. 239 des Basler Ausschusses für Bankenaufsicht (2013): Grundsätze für die Aggregation von Risikodaten und die Risikoberichterstattung. Verlangt genaue, vollständige, aktuelle und bis zur Quelle nachvollziehbare Risikozahlen.

Bedrock (Amazon) | Verwalteter AWS-Dienst für den Zugriff auf AI-Modelle. Die Knowledge Bases sind sein Baustein für dokumentbasierte Wissensabfragen, inklusive gemanagtem GraphRAG.

Constraints | Formale Pflichtregeln für Daten, etwa: jede Position braucht eine Gegenpartei. Werden über SHACL geprüft.

Cross-Region-Inference | Bedrock-Betriebsmodus, der Anfragen zur Lastverteilung innerhalb einer Geografie in andere Regionen weiterleiten kann. Relevant für die Residenzfrage.

Dynamic Views | Unity-Catalog-Sichten, die Zeilen abhängig von der Rolle des Abfragenden filtern. Setzen Zugriffsschutz in der Plattform durch, nicht in der Anwendung.

Evidence Pack | Versioniertes, schemageprüftes Bündel aller Belege einer Antwort: Kennzahl, Definition, Berechnungsquelle, Policy-Lage, Zugriffsumfang, Herkunft, Lücken.

FIBO | Financial Industry Business Ontology, die offene Fachontologie der Finanzbranche. Wird als versioniertes Subset genutzt, nie als Ganzes.

FINMA | Eidgenössische Finanzmarktaufsicht, die Schweizer Aufsichtsbehörde über Banken, Versicherungen und weitere Finanzinstitute.

Glossar (als Schicht) | Die Schicht der fachlichen Definition: Kennzahlen und Dimensionen, verbindlich, mit Owner je Begriff; bei Kennzahlen inklusive fachlicher Formel.

Golden-Question-Set | Feste Prüffragen mit definiertem Soll-Ergebnis je Frage. Bestanden heisst: Antworttyp und Belege stimmen überein.

GraphFrames | Spark-Bibliothek für Graphberechnungen auf Databricks-Clustern. Behelf, kein Ersatz für eine Graphdatenbank.

GraphRAG | Verfahren, das Dokumente per Sprachmodell automatisch in einen Graphen aus Entitäten und Beziehungen übersetzt und für Antworten nutzt.

Idempotent | Beliebig wiederholbar mit identischem Ergebnis. Eigenschaft sauberer Deployments.

Knowledge Graph | Die mit realen Daten gefüllte Instanz einer Ontologie: konkrete Kunden, Verträge, Beziehungen.

Kontextlayer | Die Schicht, die dem Agenten je Frage den richtigen Ausschnitt aus Bedeutung, Regeln und operativem Zustand vorlegt.

Lakehouse | Datenarchitektur, die die Skalierung des Data Lake mit den Verwaltungsfähigkeiten des Warehouse verbindet.

MCP / MCP-Server | Model Context Protocol, ein offener Standard, über den Agenten Werkzeuge aufrufen. Der MCP-Server stellt die Werkzeuge bereit.

Metric Views (Unity Catalog) | Zentral definierte, versionierte Kennzahl-Sichten in Databricks: eine Berechnung je Kennzahl, identisch für alle Konsumenten.

Neptune (Amazon) | Verwaltete Graphdatenbank auf AWS. Neptune Database für den produktiven Betrieb, Neptune Analytics für analytische Graphauswertungen.

Neo4j | Verbreitete Graphdatenbank, auch in eigener Infrastruktur betreibbar. Abfragesprache Cypher.

Ontologie | Die Schicht der Bedeutung: formales Modell der Klassen, Beziehungen und Regeln einer Domäne. Sagt, was ein Kunde ist, nicht, wie man eine Kennzahl rechnet.

OWL / OWL-Reasoner | Web Ontology Language, Standard für formale Ontologien. Ein Reasoner zieht daraus logische Schlüsse und prüft das Klassenmodell auf Konsistenz, unter der Annahme der offenen Welt.

Property Graph / RDF | Die zwei Graph-Datenmodelle: Property Graph (Knoten und Kanten mit Eigenschaften, Abfrage per Cypher / openCypher) und RDF (Tripel-Modell des W3C, Abfrage per SPARQL). RDF ist der Pfad für OWL-Ontologien.

revDSG | Das revidierte Schweizer Datenschutzgesetz (in Kraft seit 2023). Regelt unter anderem die grenzüberschreitende Bekanntgabe von Personendaten.

Semantic Layer | Die Schicht der Messung: setzt die fachliche Formel als geprüfte Berechnung um und verbindet Kennzahlen mit Dimensionen.

Service Principal | Technisches Konto für Systeme statt Menschen, mit eigenen, minimalen Berechtigungen.

Service-Policy | Regelwerk am AI Gateway, das je Werkzeugaufruf entscheidet: erlauben, verweigern oder unter Freigabevorbehalt stellen.

SHACL | Shapes Constraint Language, W3C-Standard zur Validierung von Instanzdaten gegen Constraints. Prüft geschlossen: was da ist, gegen das, was da sein muss.

Traversierung | Das Durchwandern von Beziehungsketten in einem Graphen über mehrere Schritte.

Unity Catalog | Die Governance-Schicht von Databricks: verwaltet Begriffe, Definitionen, Zugriffsrechte, Herkunftsnachweise und Protokolle an einem Ort.

beyond chaotic analytics GmbH | Die Plattform ist Rohmaterial. Der Entscheid ist der Unterschied.