NQAD — Neural Quantum Artificial Database

Nov 5, 2026 · @Dr. Ing. Luigi Forino Von Thyssen

NQAD is the native cognitive memory of the NQAi ecosystem: a dynamic database that lives inside the AI and records, as generative JSON code, every algorithm state, weighted internal state and trigger event that changes the outcome of the AI's computing.

1. Executive summary

NQAD makes an AI's reasoning storable, replayable and auditable. It does not store data *about* the world, as Oracle and other databases do. It stores the AI's own cognitive state, from inside the AI.

Every event that changes a status or a result of NQAi's computing writes a **synapse**: a JSON record holding the *generative recipe* of that state, not a snapshot of it. AutoCAD works the same way: it stores the vector instructions that draw an image, not the pixels.

Synapses are stored physically in the AI's own environment, linked by pointers to an index, and classified by an audit system. Together they form the **Synapses matrix**. The linked synapses behind any decision make up the **brain logic of that action**, a chain of reasons that can be reconstructed step by step.

Key properties:

- **Native:** NQAD exists only inside an NQAi system. It is not a standalone database service.
- **Dynamic:** the structure updates itself continuously as the AI's environment changes.
- **Generative:** records hold the code and numbers to rebuild a state, not the state itself.
- **Quantum-inspired:** each record can hold several weighted candidate states that collapse into one decision.
- **Auditable:** every state points back to its predecessor, giving full lineage for regulators and clients.
- **Open hardware:** it runs on RISC-V processors, inside the NQAi.cloud platform.

2. The problem

Today's AI systems decide, but they do not remember *why* in a form anyone can inspect. Their reasoning lives in transient memory and opaque model weights, and it is lost once a computation ends.

Conventional databases were never designed to hold this:

- **Relational databases** (the dBase-to-Oracle lineage) store facts in fixed tables. They describe the world, not the thinking about it.
- **Vector databases** store embeddings for similarity search. They capture meaning, not cause and effect.
- **Graph databases** store relationships, but they have no native concept of probabilistic states or of decisions collapsing out of alternatives.
- **Logs and event stores** record that something happened, not the generative logic that made it happen.

All of them sit *outside* the AI as separate services. Every interaction crosses a boundary, and the AI's inner state is translated into someone else's data model.

The gap matters most where decisions carry legal or financial weight. Algorithmic trading, banking compliance and risk management increasingly require AI decisions to be explained and reproduced on demand. NQAD closes that gap by making cognition itself the stored object.

3. Core concept

NQAD is built on three ideas: the synapse, the trigger and the brain logic of an action.

Synapse. A synapse is the smallest unit of stored cognition: one JSON file describing how to regenerate a specific state of the AI. It holds algorithm parameters, candidate outcomes with their weights, the weighted internal states that influenced them, and a pointer to the synapse it evolved from. Like a vector drawing, it is compact, exact and can be rebuilt at any time.

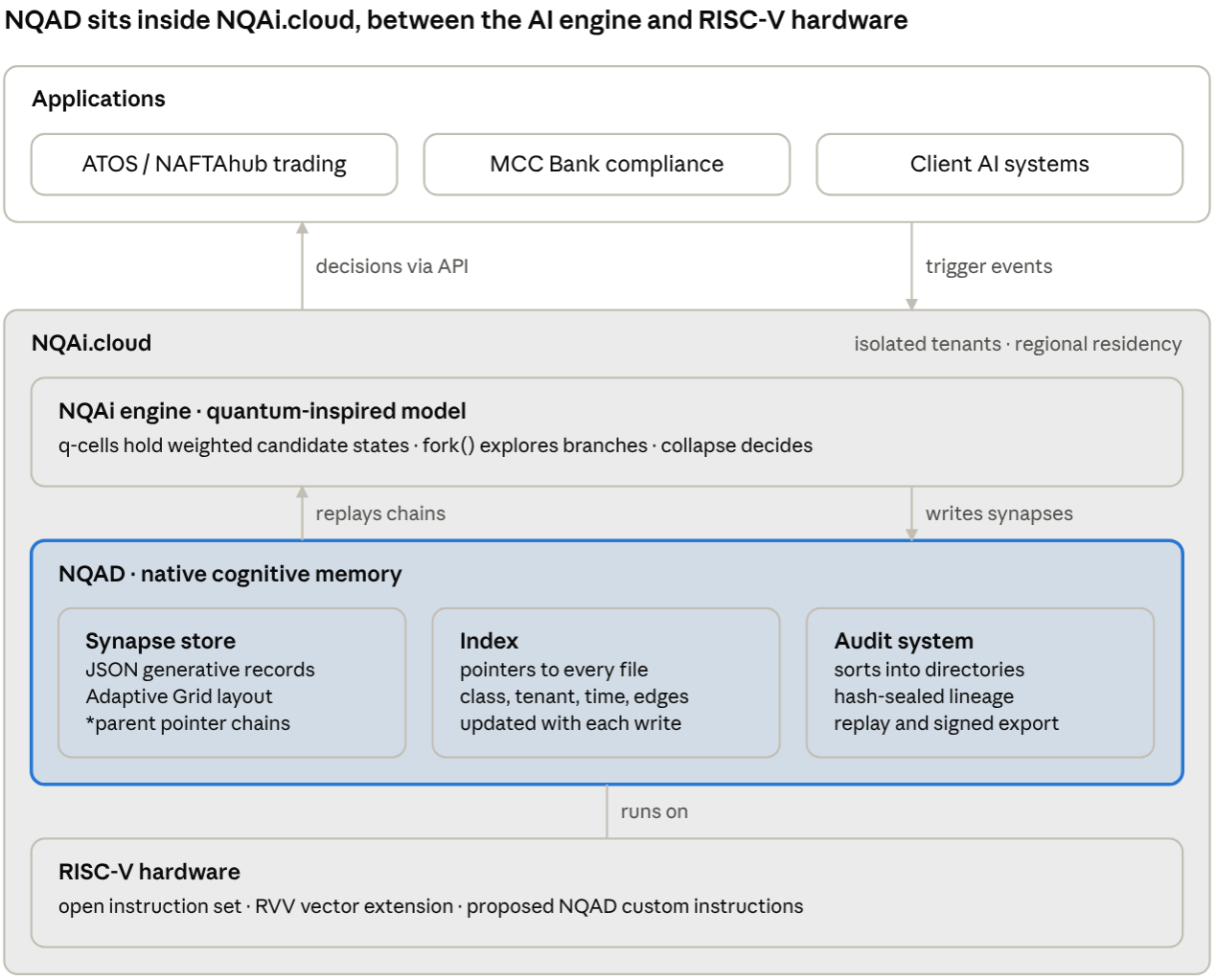
Trigger. A trigger is any event that changes a status or a result of the AI's computing: a new input, a market tick, a rule firing, a change of confidence or priority. Triggers are what keep NQAD alive. Nothing is written on a schedule; everything is written because something changed.

Brain logic of an action. No decision comes from a single synapse. It comes from a chain of linked synapses, each pointing to its predecessor. Together, that chain is the brain logic of the action: the complete, replayable path from the first trigger to the final decision.

Weighted internal states ("emotions"). NQAi models internal states that work like emotions in a human brain: confidence, urgency, risk aversion, priority, caution. In NQAD these are numeric weights stored in each synapse, so they can be measured, compared and audited like any other parameter.

The result is a database that is active, not static. Its contents and its structure both change as the AI thinks.

4. Architecture at a glance



NQAD architecture · four layers

Applications send trigger events into NQAi.cloud. The engine forks and collapses candidate states, NQAD records every synapse and replays chains on demand, and the whole stack runs on RISC-V.

5. Hardware foundation: RISC-V

NQAi and NQAD run on RISC-V, the open, royalty-free processor instruction-set architecture. RISC-V is classical computing hardware; the "quantum" in NQAD describes the cognitive model (Section 6), not the processor.

Why RISC-V:

- **Independence:** there are no ARM or x86 licences, so the full stack from silicon to software stays under NQAi's control.

- **Custom instructions:** RISC-V allows dedicated instructions to be added. NQAD can define hardware primitives for its most frequent operations: writing a synapse, firing a trigger, forking candidate states and collapsing them.
- **Vector processing:** the standard RISC-V Vector extension (RVV) provides the parallel arithmetic needed for neural workloads and for updating many q-cells at once.
- **Quantum-ready:** RISC-V cores are already used as classical control processors beside quantum chips, so a physical quantum co-processor can be added later without changing architecture.

Proposed NQAD instruction extension (concept):

Instruction	Function
`syn.write`	Commit a synapse record and its parent pointer
`syn.fork`	Branch a q-cell into parallel candidate states
`syn.collapse`	Resolve candidate states into one outcome by weight
`syn.prune`	Discard branches below a weight threshold
`syn.link`	Update index pointers and graph edges

These names are illustrative. Defining the final extension is part of the hardware roadmap (Section 13).

6. Quantum-inspired cognitive model

NQAi's model borrows three concepts from quantum physics and implements them as computation on classical RISC-V hardware. This is what "Quantum" means in NQAi and NQAD: a quantum-inspired model, not execution on physical qubits.

Quantum concept	NQAD implementation
Superposition	A **q-cell** holds several candidate states at once, each with a probability weight
Entanglement	Linked synapses influence each other: a change in one re-weights the states of those connected to it
Measurement and collapse	A trigger forces the q-cell to resolve into a single outcome, which is recorded

q-cells (simulated qubits). The basic storage unit of the Adaptive Grid. A q-cell is a classical data structure holding a set of weighted candidate states. Calling it a q-cell rather than a qubit makes clear that it is simulated, not physical.

Continuous fork(). When a trigger arrives, the engine branches the affected q-cells into parallel candidate paths and explores several possible outcomes at the same time. Each branch carries its own weights and its own internal-state values.

Pointer chain (`*parent`). Every new state stores a pointer to the state it came from. The result is an unbroken lineage, comparable to Git commits or a blockchain, which makes every decision traceable back to its origin.

Collapse. When the evidence is sufficient, or a decision is required, the branches resolve into one outcome. The winning branch and the rejected alternatives with their weights are recorded, so an auditor can see what the AI chose and also what it considered and set aside.

7. Data model: the Adaptive Grid

NQAD uses a hybrid data model called the **Adaptive Grid**. It does not choose between relational, vector and graph; it combines all three. Every synapse can be read through three views, and the grid reorganises its physical layout as usage patterns change.

View	What it gives	Typical use
Relational	Fields, filters, joins	Reporting, compliance queries
Vector	Embeddings for similarity	"Find past situations like this one"
Graph	Edges and pointer chains	Lineage, replay, brain logic of an action

Physically, each synapse is a JSON file stored in the AI's own environment, in a directory chosen by the audit classifier. The index holds pointers to the files; the files hold the content.

Synapse schema (draft v0.1):

```
{
  "id": "syn_8f21",
  "parent": "*syn_7c04",
  "timestamp": "2026-11-05T10:37:12.418Z",
  "tenant": "atos-prod",
  "trigger": {
    "type": "market_tick",
    "source": "NAFTAhub",
```

```
    "payload": {"symbol": "XAUUSD", "price": 2374.10}
  },
  "generator": {
    "algorithm": "confluence_v3",
    "params": {"adx": 27.4, "ema_fast": 2371.8, "ema_slow": 2366.2}
  },
  "internal_state": {
    "confidence": 0.71,
    "urgency": 0.40,
    "risk_aversion": 0.55
  },
  "states": [
    {"branch": "fork_a", "action": "BUY",  "weight": 0.62},
    {"branch": "fork_b", "action": "HOLD", "weight": 0.38}
  ],
  "collapsed": "fork_a",
  "views": {
    "vector": [0.12, -0.44, 0.08],
    "edges": ["syn_7c04", "syn_8a99"]
  },
  "audit": {
    "class": "trading.decision",
    "path": "/nqad/atos-prod/trading/decision/2026/11/05/",
    "hash": "sha256:…"
  }
}
```

The `generator` block is what makes NQAD generative: it holds the algorithm and parameters needed to recompute the state, so the state can be reproduced exactly rather than merely described.

8. Synapse lifecycle

Every synapse passes through the same seven stages, from the event that causes it to its long-term archive.

1. **Trigger:** an event changes a status or result in the AI's environment (a new input, a rule, a market tick, a change of internal state).
2. **Fork:** the engine branches the affected q-cells into parallel candidate states.
3. **Weight:** each branch is scored by the algorithm and adjusted by the weighted internal states.
4. **Collapse:** when a decision is needed or one branch dominates, the branches resolve into a single outcome.

5. **Record:** the synapse is written as a JSON file with its generator, branches, outcome and `*parent` pointer.
6. **Index and audit:** the classifier assigns the synapse to a class and directory, updates index pointers and graph edges, and seals it with a hash.
7. **Prune and archive:** branches below a weight threshold are discarded, and older chains are compressed. Pointers stay intact so lineage is never broken.

Entanglement adds a feedback loop. Writing one synapse can re-weight linked synapses, which may cross a threshold and fire new triggers. This continuous propagation is what makes the Synapses matrix behave as a living structure rather than a stored archive. Section 13 covers the controls that keep this propagation bounded.

9. Index and audit system

The index and the audit system are what turn a directory of JSON files into a database. The index tells NQAD where everything is; the audit system decides how it is classified and makes it tamper-evident.

Index. A pointer structure maps synapse IDs, classes, tenants, time ranges and graph edges to the physical files. Queries resolve through the index first and open only the files they need. Because the index is updated by the same trigger that writes the synapse, it is never out of date.

Classification. Every new synapse is assigned a class (for example `trading.decision`, `risk.alert`, `compliance.check`) and a directory path derived from tenant, class and date. The directory tree is therefore itself a readable map of the AI's activity.

Integrity. Each synapse is sealed with a hash that covers its content and its parent's hash. Altering any record breaks the chain from that point on, so tampering is detectable.

Replay. Given any decision, NQAD walks the `*parent` chain backwards and re-runs each `generator` block to reproduce the exact sequence of states. That gives the brain logic of the action as evidence: what the AI saw, what it considered, how it weighted the options and why it chose what it did.

Audit queries (examples):

- Show the full brain logic of trade #48211.
- List every decision in October where confidence was below 0.5 and the action was still executed.
- Find past situations most similar to today's market and how the AI decided then.

10. Deployment in NQAi.cloud

NQAD is the native memory layer of **NQAi.cloud**. It comes with the platform rather than being sold or deployed as a separate product. Applications such as ATOS/NAFTAhub and MCC Bank use NQAi through the cloud, and NQAD records the cognition behind every result they receive.

Four design decisions define how NQAD runs in the cloud:

Area	Design principle
Tenancy	Each client or AI instance gets an isolated synapse space with its own index and keys. Cross-tenant links are not allowed.
Residency	Synapses stay in the jurisdiction of the RISC-V nodes that hold them. Region choice (Switzerland, EU, Hong Kong) follows client and regulatory needs.
Access	Applications receive decisions through the NQAi.cloud API. Raw synapses stay sealed inside the AI and are reachable only through controlled audit interfaces.
Audit export	A regulator or client can receive the brain logic of a specific decision as a signed, self-contained package, without access to the rest of the system.

Synapses can contain client data, trading positions and internal reasoning. Deployment therefore has to meet the applicable data-protection and confidentiality rules in each region, including GDPR in the EU and the Swiss Federal Act on Data Protection.

11. Positioning

NQAD draws on established ideas: event sourcing, knowledge graphs, vector search and ML lineage tracking. What is new is the combination: generative records of an AI's own cognition, stored natively inside the AI and driven by triggers.

Capability	Relational (Oracle-class)	Vector databases	Graph databases	Event sourcing	NQAD
Stores	Facts about the world	Embeddings	Entities and relations	Events that occurred	The AI's cognitive states

Capability	Relational (Oracle-class)	Vector databases	Graph databases	Event sourcing	NQAD
Location	External service	External service	External service	External service	Inside the AI
Record type	Values	Vectors	Nodes and edges	Event payloads	Generative recipe (code and numbers)
Probabilistic states	No	No	No	No	Yes (q-cells)
Writes driven by	Applications	Applications	Applications	Applications	AI triggers, automatically
Reproduces a decision	No	No	Partly	Partly	Yes, by replaying generators
Data model	Relational	Vector	Graph	Log	Hybrid Adaptive Grid

NQAD is not designed to replace Oracle or other operational databases for business records. Customer accounts, ledgers and transactions still belong in conventional systems. NQAD is the database for the reasoning about them.

12. Use cases

ATOS: one gold trade, stored as brain logic

ATOS receives an XAUUSD price tick from NAFTAhub. Here is how NQAD records the decision that follows (illustrative values):

1. **syn_7c01, trigger:** the price breaks above a pivot level. The trend q-cell forks into *continuation* (0.55) and *false breakout* (0.45).
2. **syn_7c04, trigger:** ADX rises to 27.4 and the fast EMA crosses above the slow EMA. Entanglement re-weights the trend cell: continuation 0.68.
3. **syn_8a99, trigger:** the risk module reports exposure within limits. Risk aversion drops from 0.70 to 0.55.
4. **syn_8f21, trigger and collapse:** the decision q-cell forks into BUY (0.62) and HOLD (0.38) and collapses to BUY. The order is sent.

If the trade is questioned a month later, an auditor asks for the brain logic of syn_8f21. NQAD walks back through the four synapses, re-runs each generator, and shows what the AI saw, which alternatives it weighed, and why BUY won.

MCC Bank: explainable compliance decisions

A payment is screened by NQAi before release. The AI forks *release*, *hold for review* and *reject*, weighted by sanctions matches, transaction patterns and counterparty risk. The collapsed decision and its full chain are sealed in NQAD.

When a compliance officer, auditor or regulator asks why a payment was held or released, the bank exports that decision's brain logic as a signed package. It does not need to reconstruct the reasoning from logs, and it does not need to open the rest of the system.

13. Engineering challenges and roadmap

The main risk is growth. A system that forks on every trigger and propagates through linked synapses can grow exponentially unless it is bounded by design.

Challenge	Mitigation
Branch explosion from continuous fork()	Maximum branches per q-cell; prune below a weight threshold; collapse on a deadline
Runaway propagation through entanglement	Propagation depth limits and damping factors on re-weighting
Storage growth	Tiered storage; compress old chains while keeping pointers and hashes
Concurrency	Copy-on-write synapses: a new state is always a new record, never an overwrite
Index consistency	Synapse and index updated in one atomic commit
Replay determinism	Version and pin every algorithm referenced by a generator; record random seeds
Privacy	Per-tenant encryption keys; field-level redaction in audit exports

Roadmap (phases):

1. **Specification:** finalise the synapse schema, the trigger taxonomy and the internal-state model.
2. **Reference engine:** software implementation on off-the-shelf RISC-V boards, with Adaptive Grid storage, index and audit.
3. **ATOS pilot:** record live trading decisions in a sandbox and validate replay and audit export.

4. **Hardware acceleration:** define and prototype the NQAD custom instruction extension.
5. **NQAi.cloud release:** multi-tenant deployment with regional residency and client audit portals.
6. **Quantum co-processor (optional):** offload selected optimisation and sampling tasks to physical quantum hardware.

14. Glossary and open questions

Term	Meaning
NQAi	Neural Quantum Artificial Intelligence: the AI ecosystem NQAD belongs to
NQAD	Neural Quantum Artificial Database: NQAi's native cognitive memory
Synapse	One JSON record holding the generative recipe of an AI state
Synapses matrix	The full set of linked synapses in an NQAi system
Brain logic of an action	The chain of synapses that produced a decision
Trigger	An event that changes a status or result of the AI's computing
q-cell	A simulated qubit: a cell holding weighted candidate states
fork()	Branching a q-cell into parallel candidate paths
Collapse	Resolving candidate paths into one recorded outcome
`*parent`	Pointer from a synapse to the state it evolved from
Adaptive Grid	NQAD's hybrid relational, vector and graph storage model
Generator	The algorithm and parameters that recompute a synapse's state
Internal states	Numeric weights such as confidence, urgency and risk aversion

Open questions:

- [ ] Is NQAi.cloud for MCC Group companies only, or also for external clients? This determines how strict tenant isolation must be.
- [ ] Will NQAi run on off-the-shelf RISC-V boards, on custom silicon, or on both in phases?
- [ ] Which regions will host the first NQAi.cloud nodes?
- [ ] What is the default retention period for synapses, and does it differ for trading and banking?
- [ ] Which internal states form the standard set for v1?