Towards large scale openEHR Clinical Data Repositories: results and lessons learned from Catalonia.

Jordi Piera Jiménez[4,5], Vinzenz Müller[1], Holger Reise[1], Alexandru Vidrean[1], Pablo Beirán Amigo[2], Juan Bescós Grillo[2], Josep Antoni Mira Palacios[3], Jordi Gabaldà Azofra[3]

[1]vitagroup health intelligence GmbH, Mannheim, Germany
[2]IBM Consulting, Barcelona, Spain
[3]Government of Catalonia – CTTI, Barcelona, Spain
[4]Digitalization for the Sustainability of the Healthcare System (DS3) research group, Barcelona, Spain
[5] Information Systems Directorate, Catalan Health Service, Barcelona, Spain

Introduction

As in most Western countries, the digital transformation of Catalonia is now essential to sustain and improve the Catalan health system [1,2]. This need is driven by demographic ageing, the rise in chronic conditions, sustainability challenges, and increasing expectations from citizens [1]. Health information systems can be a key driver of this digital transformation. However, it is widely recognized that they can also become barriers if they remain fragmented, slow, and poorly interoperable [3,4]. A major criticism of current health information systems is that many are based on an episodic, activity-centered, and siloed care model. This results in information islands, duplicated documentation, and a poor experience for both healthcare professionals and citizens [1–4]. To address these technical and social challenges, the Catalonia Department of Health has designed Plataforma Oberta (Open Platform). This platform aims to support the redesign of healthcare workflows around longitudinal, integrated, and person-centered care [5]. A key requirement for developing Plataforma Oberta is a semantically rich and expressive information model that accurately represents the patient journey throughout the healthcare system. This is essential to support person-centered care. To achieve this, the open standard openEHR was selected. openEHR enables the unified representation of clinical information in a Clinical Data Repository (CDR), which will serve as the backbone of Plataforma Oberta. The openEHR CDR used for Plataforma Oberta is the Health Intelligence Platform (HIP) (Vitagroup Health Intelligence GmbH, Mannheim, Germany) which operates over a distributed relational database (Yugabyte v2025.2).

However, while openEHR repositories have existed for almost 30 years, no previous implementation has managed health data at the population scale of Catalonia. In addition, the patient-centered model enforced by openEHR has direct implications for the design of Online Transaction Processing (OLTP) systems. openEHR CDRs are built for highly concurrent CRUD operations, focusing on one patient at a time, while maintaining strict ACID compliance and versioning. This requires extensive joins across archetypes

referenced in the contains section, access to elements in deep paths, and traversing complex relationships inherent to the openEHR Reference Model. These characteristics present significant challenges for achieving high performance when processing Archetype Query Language (AQL) queries at the scale required by Plataforma Oberta. During the CDR implementation, these challenges led to AQL performance limitations that required extensive optimization efforts.

Objective

During the last 2 years we have worked in the implementation of several optimization mechanisms on the CDR for patient-bound AQL queries. The objective of this paper is to present the deployment of the HIP CDR for Plataforma Oberta, the challenges encountered, and the improvements made to support a regional-scale implementation (13 million patients). First, we describe the methods used to optimize the AQL processor of the HIP. Second, we outline the evaluation methods. Third, we present the results of the optimization strategies. Finally, we discuss which optimization approaches are most suitable in different scenarios.

Methods

Several optimization strategies were evaluated for increasing the performance of the CDR. The most relevant ones are: in-memory AQL processing, index definition, and SQL enrichment.

In-memory AQL

AQL engines translate AQL queries into the underlying database query language (e.g., SQL for relational databases) and delegate query execution to the underlying databases engine. Once the database retrieves the results, the AQL engine parses and reconstructs them into an openEHR-compliant format. However, this approach is inefficient for long paths that require multiple joins to preserve the intended semantics—such as deep tree structures traversal. To address this limitation, we developed an in-memory AQL technique, that executes part of the AQL query logic in application memory — after the data has been fetched from the database — rather than relying entirely on the database engine for filtering and projection. In-memory AQL performs two main steps: (i) fetching a broader or simpler result set from the underlying database, and (ii) performing AQL projection, filtering, aggregation and ordering on openEHR structures (e.g. COMPOSITIONs) in server memory. We evaluated the in-memory AQL execution mode ,a.k.a. In Memory Projection (IMP), as an alternative query-processing path for EHR-/Composition-bound queries, with the goal of improving response times without requiring schema changes on the database side. The evaluation methodology included: (i) selecting representative AQL queries for targeted testing, (ii) executing these queries using the IMP engine and comparing results against the baseline execution path, and (iii) documenting operational constraints and applicability boundaries. Performance was measured using a 10% randomly selected complete EHRs from the production dataset.

All data used, both in production and tests, had been previously anonymized. Tests were conducted with one and ten concurrent requests to assess the impact of query concurrency on performance.

Indexing strategy and evaluation

A second strategy for improving AQL execution involved the development of purpose-built index structures that materialize frequently queried openEHR paths into relational structures. Indexing focused on two complementary mechanisms:

Conventional indexes

Conventional indexes were defined on high-utility temporal and path fields, including planned indexes on time-related attributes such as ACTION.time, POINT_EVENT.time, INTERVAL_EVENT.time, and EVENT_CONTEXT.start_time. Additional indexes on composition data were also created to improve query-path performance and support join-removal optimizations.

Integrated approach (index table)

Another indexing strategy proposed was the implementation of an index table designed to support specific use cases. This strategy considered operational impacts, including the risk of requiring a dedicated index table for each use case.

Where applicable, the team also examined index mechanics and query plan behavior to ensure performance gains. Index creation was planned and executed with explicit operational constraints.

The impact of indexing was evaluated through repeated performance testing cycles, including:

- Automatic performance test runs after introducing new indexes and adjusting AQL queries, with explicit tracking of best and worst-case latency under different time-range constraints.
- Environment simulation by restoring PREPROD/PROD statistics into a cloud environment (using an anonymized subset containing a 10% of the production anonymized dataset with full EHRs) to reproduce query planner behavior and enable comparative benchmarking of SQL and AQL workloads.
- Operational monitoring during staged migrations and deployments, including observation of disk usage and size changes associated with indexing.

Manual AQL

Manual AQL is a feature designed to provide a custom SQL instructions instead of relying on the AQL engine to translate the AQL query. Leveraging in-depth knowledge of the database model and properties of the data, it is often possible to design SQL that

outperforms the AQL engine. These custom SQL queries can be provided as part of a stored AQL query definition.

This approach allows for fine tuning of AQL performance, while maintaining the AQL-level interface and parameterization used by the application.

Results

Overview

Across the evaluation period, the deployment demonstrated functional feasibility for large-scale openEHR data ingestion and operational continuity when following certain patterns for leveraging AQL design, in-memory AQL and use of indexes. The most salient findings are described in the sections below.

Effects of applying in-memory AQL

The benchmark of IMP focused only on single patient queries (one openEHR EHR). Across the benchmarked AQL queries, enabling in-memory AQL yielded a substantial reduction in median latency compared with the regular execution path. Under non-concurrent load, the median latency across queries decreased from 153 ms to 26 ms, corresponding to a 5.7× median speedup, with most queries showing multi-fold improvements (up to 12×). Under concurrent load (10 concurrent requests), median latency decreased from 157 ms to 38 ms, a 3.9× median speedup, indicating that gains largely persisted under load but were partially attenuated by contention. Performance improvements were observed across nearly all queries. Overall, the results indicate that the in-memory approach markedly improves responsiveness for COMPOSITION- and EHR-scoped queries, while a small subset of query patterns may require additional tuning or fallback to the conventional path. In one benchmarked lab query, there was a reduction from approximately 1.5–2.0 seconds to ~600 ms (on a 10 % subsample of anonymized production data).

Overall, in-memory AQL has a high-impact performance lever for specific queries, with early evidence of improved latency, but with acknowledged limitations described in the following. Our results evidence that in-memory AQL yields the highest performance improvements when the standard AQL engine / DB queries are not performant enough, or some operations on COMPOSITIONs are hard to express efficiently in SQL queries. Or the combination of many projections and filters on COMPOSITION data are required. In these cases, the underlying DB is often too slow due to the complexity of openEHR structures. Performance measurements showed incremental improvements in throughput and tail latency. However, these improvements did not translate into consistent compliance with all production targets for part of the queries. Results for patient-centric queries were stable across different system sizes, suggesting that performance depended more on per-EHR data volume than total database size. It is relevant to note that the use of IMP

changed the load distribution from the DB to the HIP AQL processor, thus requiring less DB resources and increasing the HIP AQL engine ones, including RAM memory, CPUs, and network bandwidth.

Effect of applying indexing

Overall, the index strategy demonstrated clear performance and scalability benefits, particularly for population-level queries (queries over all patients), while remaining feasible to operate from a maintainability and storage perspective.

In mini-performance tests, the query execution trend without index tables increased sharply with dataset size, whereas queries using index tables showed an approximately logarithmic trend, indicating improved scalability; in contrast, the write-path overhead from maintaining index tables during inserts remained near-constant with increasing data volume.

In performance tests against the large test dataset (10% anonymized production data), the largest relative improvements were consistently observed for the cross-patient use cases, and index tables enabled execution of at least one query that otherwise timed out (notably cross-patient queries over a 1-year period without index support).

For patient-centric queries, the improvement factor was smaller but still meaningful because the baseline query was already comparatively fast. As with IMP, results for patient-centric queries were stable across different system sizes. Again, this suggests that performance depends more on per-EHR data volume than total database size.

Maintainability experiments showed that creating and filling new indexes is operationally achievable but resource intensive on production-scale data: successive iterations reported filling on the order of tens of minutes to hours depending on volume and parallelization.

Storage analyses indicated that index tables are substantially smaller than the generic composition data structure used by standard AQL, supporting the premise that materializing targeted fields reduces both data scanned and query complexity. On the large test dataset (117,196,656 compositions; storage size ~1.38 TB), the index table for laboratory results (for single patient queries) covered 8,227,707 compositions and produced 188,840,434 rows, while vaccination actions (for cross-patient queries) covered 13,708,414 compositions and produced 15,256,535 rows.

The report further quantified substantial size reductions compared with composition data (e.g., composition data vs. laboratory results ~28.5× larger by table size; composition data vs. vaccination actions ~203× larger), reinforcing that index tables can provide performance benefits while keeping storage overhead tractable for targeted use cases.

Effects of Manual AQL

Manual SQl has been used in very specific scenarios where other optimization strategies had limited success. Specifically, it was used in cross-patient queries which had problematic behaviors timing out. While Manual SQL can be useful for cross-patient queries, our experience shows that cross-patient queries oriented towards analytical uses (research, epidemiology, patient stratification) are better managed in separate OLAP repositories which can be synchronized by means of the event trigger functionality of the HIP.

Discussion

Running an openEHR Clinical Data Repository (CDR) at regional scale requires treating AQL performance as a multi-layer optimization problem with multiple more specialized solutions. In the Catalonia deployment, the dominant performance constraints emerged from the inherent mismatch between (i) the patient-centric, highly normalized, versioned OLTP characteristics of openEHR data and (ii) the need to serve both interactive, per-patient retrieval and population-level, cross-patient queries. As a result, different optimization strategies were effective for different query classes, and the most reliable improvements were achieved by selecting the best strategy among: optimal AQL design patterns, execution-path changes (in-memory AQL), database-side indexing, and targeted SQL-level tuning for stable stored queries.

In-memory AQL proved to be a high-impact lever for selected AQL patterns, particularly those where the standard AQL to SQL translation yields complex plans or where operations on openEHR structures (e.g. multiple complex projections and filters) are difficult to express efficiently in SQL. In our benchmarks on a production-like dataset, the in-memory execution mode reduced median latency from 153 ms to 26 ms under single-thread load (5.7× median speedup) and from 157 ms to 38 ms under 10-request concurrency (3.9× median speedup), indicating that benefits largely persist under load but are attenuated by contention. Indexing delivered measurable improvements but remained heterogeneous across workloads, reinforcing that adding indexes is not a one size fits all remedy for AQL performance. For cross-patient queries (population-based queries), tests combining new indexes and AQL adjustments ranged from best-case sub-second behavior to worst-case multi-second responses, and performance was strongly sensitive to time-window width, with larger intervals consistently degrading response times. While the benefits of in-memory AQL are significant, it still needs to benefit from optimizations which currently are in progress. These include the use of caches to minimize large data retrievals from the underlying database and better traceability.

The results also highlighted that query design patterns—especially result limiting and paging—can dominate performance in some scenarios: in one example, broad timeframes without limits produced ~60 s responses, while applying limits reduced runtime to <1 s.

Operationally, index creation itself can become a bottleneck. For this reason their implementation requires alignment with real data distribution, partitioning, and robust deployment workflows with validation, rollback, and production-like rehearsal. Also, the underlying DB data model should be optimized for the performance of AQL.

The most scalable database-side approach for cross-patient workloads was the index table (Integrated Approach) strategy, which materializes frequently queried paths into relational structures and rewrites SQL generation to target those structures. In our evaluation on production-like data (>117M compositions), index tables produced stable trends as dataset size increased: non-indexed execution showed sharply worsening trends, whereas indexed queries remained approximately logarithmic with scale. This benefit was particularly pronounced for cross-patient population-level queries, and index tables enabled execution of at least one query that otherwise timed out.

Importantly, storage overhead was not prohibitive: the evaluation reported substantial size reductions versus the generic composition data representation (e.g., composition data ~28.5× larger than laboratory results index, and ~203× larger than vaccination actions index by table size), supporting the premise that targeted materialization reduces scanned data and planning complexity. However, index tables introduce governance and maintainability implications, since they are most suitable for stable, parameterized query families and can drift toward one index table per use case if not managed systematically.

Where neither in-memory execution nor indexing delivers acceptable behavior, manual stored queries provide a pragmatic escape hatch for high-value, stable workloads: by explicitly specifying the SQL and optional planner hints for a known AQL query, teams can leverage database expertise to stabilize plans and reduce planning overhead for complex queries. This approach should be treated as a last resort, since it requires engineering effort and in-depth knowledge to keep AQL semantics and SQL behavior aligned, also as the system evolves.

Finally, the combined evidence suggests that even with an optimal strategy per query type, some scenarios—especially demanding cross-patient aggregations—may remain intrinsically difficult to serve with OLTP-oriented guarantees (strict ACID, versioning, high-concurrency writes) at sub-second latencies. For these, a hybrid architecture is warranted: keep the openEHR CDR as the system of record for write-heavy clinical workflows, while offloading analytics-style workloads to an OLAP-optimized store via controlled exports. In this direction, the repository's ability to trigger data exports provides a pathway to synchronize external read-optimized systems without compromising the CDR's transactional role. While the strategies presented are oriented towards relational database management systems, other optimization strategies may be possible using other database paradigms such as documental ones [6]. Future research may consider a a comparable benchmark between the two approaches.

Practical implication

The operational recipe that emerged from the Catalonia work is: (i) modeling based on clinical scenarios identifying most queried datapoints and align those with template design to facilitate indexing; (ii) start with query design guidance (limits/paging and known-good patterns), (iii) apply in-memory AQL selectively for structurally complex, per-patient retrievals, (iv) use conventional indexes for broadly reusable RM-level fields , and (v) reserve manual SQL for stable, business-critical stored queries.

References

[1] Piera-Jiménez J, Carot-Sans G, Ramiro-Pareta M, Nogueras MM, Folguera-Profitós J, Ródenas P, Jiménez-Rueda A, de Pando Navarro T, Mira Palacios JA, Fajardo JC, Ustrell Campillo J, Vela E, Monterde D, Valero-Bover D, Bonet T, Tarrasó-Urios G, Cantenys-Sabà R, Fabregat-Fabregat P, Gómez Oliveros B, Berdún J, Michelena X, Cano I, González-Colom R, Roca J, Solans O, Pontes C, Pérez-Sust P. A 25-Year Retrospective of Health IT Infrastructure Building: The Example of the Catalonia Region J Med Internet Res 2024;26:e58933 doi: 10.2196/58933

[2] Piera Jiménez J, Michelena Vegas X, Climent Fageda A, Valle Martín Ll, Maymó Costa A, Dodas Perpinyà A, et al. Visió col·lectiva per al futur de la salut: bases per a un nou model assistencial i de sistemes d'informació. Barcelona: Departament de Salut; 2025. https://salutweb.gencat.cat/ca/detalls/Articles/2025-07-10-llibre-blanc

[3] Hertzum M, Ellingsen G, Cajander Å. Implementing Large-Scale Electronic Health Records: Experiences from implementations of Epic in Denmark and Finland. Int J Med Inform. 2022 Nov;167:104868. doi: 10.1016/j.ijmedinf.2022.104868. Epub 2022 Sep 14. PMID: 36194994.

[4] Torab-Miandoab A, Samad-Soltani T, Jodati A, Rezaei-Hachesu P. Interoperability of heterogeneous health information systems: a systematic literature review. BMC Medical Informatics and Decision Making. 2023;23:18. doi:10.1186/s12911-023-02115-5.

[5] Publication data - Plataforma de Serveis de Contractació Pública, Centre de Telecomunicacions i Tecnologies de la Informació de la Generalitat de Catalunya, 2023, https://contractaciopublica.cat/en/detall-publicacio/944a111e-7bec-d408-0756-42a2549aa849/200041523.

[6] Mateu Amengual F, Cox G, Mira i Palacios JA, Crossley J, Underwood J, Rodríguez G, Alonso C, Sanz de Acedo J, Pachot F. A document-first openEHR persistence layer for operational single-patient and cross-patient queries. BMC Proc. 2026. In press.