**Case Study: Modernizing Data Infrastructure at HCSC through Azure Cosmos DB Adoption**

**The Problem Statement:**

Health Care Service Corporation (HCSC) faced an urgent need to replace its legacy in-memory data grid technology, Gemfire, due to increasing licensing costs, lack of enterprise support, and strategic misalignment with the broader digital transformation goals of the company. This urgency was compounded by the growing complexity and data volume of its flagship app handling over a terabyte of claims data, along with various other critical applications dependent on real-time data access and processing.

**Industry:** Healthcare/Health Insurance.

**Domain/Department:** Digital Department – focused on supporting web and mobile experiences for policyholders.

**The Person Interviewed:** Michael LaBrot, Senior Software Developer / Architect   
Key role: Lead contributor to the Center for Enablement (C4E) for Cosmos DB at HCSC.

**Background:**

HCSC’s digital ecosystem included a suite of applications with varying sizes and complexities. The flagship application, central to member experience, served claims information history by presenting the most effective (i.e., most recent) adjustment. The database exceeded over a terabyte in size.

A smaller application referred to as the “hello-cosmos” app was also used. It was a few megabytes in size and served 800 (toll-free) numbers.

Several other mid-sized applications (in gigabytes) played a critical role in delivering information related to:

* Primary Care Providers (PCPs)
* Coverage Benefit Codes
* Coverage Code Customization

While most applications used Azure Databricks to write documents to the database, the flagship app utilized Talend to feed documents via Kafka into Spring Boot apps. These Spring Boot apps then transformed the data and loaded it into Cosmos DB (which is the targeted replacement for the aging Gemfire system). This migration effort was essential not just to reduce costs and improve support, but also to ensure scalability and alignment with HCSC’s long-term cloud and data modernization strategies.

**Business Objectives:**

* **Eliminate licensing costs** associated with Gemfire.
* **Increase system resilience and availability** by moving to cloud-native solutions.
* **Enable developer agility** with better access to cloud-native databases.
* **Standardize tech stack** to align with enterprise-supported tools.

**The Challenges:**

* Avoid steep licensing fees from Broadcom post-acquisition of Gemfire.
* Migrate legacy systems without disrupting mission-critical web and mobile services.
* Operationalize a new cloud-native NoSQL database (Azure Cosmos DB) for enterprise-scale use.
* Gemfire did not support data replication which made it hard to recover quickly if something went wrong (hindered recovery and resiliency).
* Gemfire would at times lock-up, the team in some cases would need to re-import the full data causing delays, disruptions, and increased maintenance overhead.
* HCSC’s onshore developers use Macs (AVMs for offshore), but Microsoft’s Cosmos DB Emulator is only available in GA for the Windows version. Attempts to run the Linux version via Docker on MacOS (using Rancher) were unsuccessful, making it difficult for developers to test locally.
* To support the development workflow, the team provisioned “sandbox” accounts in Cosmos DB as an alternative, enabling developers to build and test effectively without a local emulator.
* Overcome process bottlenecks in the development-to-testing pipeline.

**The Approach:**

Michael was brought in to operationalize the use of Cosmos DB by:

* Establishing best practices, documentation, and governance frameworks.
* The core focus of this initiative was not just getting a single application into production, but using that process to shape repeatable, scalable operational patterns for broader enterprise adoption of Cosmos DB i.e., determining how to live with and support it effectively in a large-scale enterprise environment.
* Best practices refer to the creation and refinement of proven, repeatable methods that ensure consistent, secure, and efficient implementation. These were derived directly from the experience of getting the first application into production and included:
* Implementing role-based access control (RBAC)
* Securing access using firewall policies
* Establishing the correct way to connect Spring Boot applications to Cosmos DB,
* Developing methods to monitor both Cosmos DB health and application health, ensuring observability and responsiveness to potential issues.
* These practical insights fed into a governance framework which is a structured approach to overseeing the use of Cosmos DB across the enterprise. This framework ensured that all future applications would follow the same high standards for security, connectivity, observability, and maintainability.
* Collaborating across security, support, and architecture teams to make Cosmos DB compliant and usable.
* Michael authored a simple and easy to understand reference app known as “good code” that future developers could run for themselves, and copy/paste. This was a working example of Cosmos DB integration for developers for future projects.
* Prototyping environments for developers to enable experimentation and local development.
* Innovating around emulator issues by proposing sandbox environments with elevated privileges.
* As application teams migrated from Gemfire to Cosmos DB, certain recurring architectural patterns and operational needs began to surface particularly around data lifecycle management. One significant theme was data retention.
* In Gemfire, the team previously followed a “Fill & Kill” strategy, where a new container was filled annually, and the old one was discarded. However, with Cosmos DB, this approach evolved. The team began exploring Time-to-Live (TTL) policies native to Cosmos DB, which offered a more granular and automated mechanism for managing data expiration, therefore, highlighting how platform-native capabilities (like Cosmos DB’s TTL support) could lead to leaner, more efficient data architectures when fully leveraged.

**The Implementation:**

* Created a developer-friendly sandbox environment for each application team to provision and test Cosmos DB usage.
* Collaborated with security and infrastructure teams to define firewall and access protocols.
* Conducted resiliency testing to ensure the new infrastructure met availability and reliability standards.
* Supported phased migration of applications from Gemfire to Cosmos DB.
* Instrumented applications with additional metrics for better observability of integration with Cosmos DB.
* The reference application developed during the flagship migration effort effectively served as an archetype for future Cosmos DB integrations. It provided reusable patterns for establishing connections and implementing custom metrics, with much of the code intentionally written to be copy-pasted by other application teams. This approach streamlined early adoption by offering a tested, developer-friendly foundation.
* In terms of automation:
* No generic scripts were created to automate the transformation of applications from Gemfire to Cosmos DB, as the migration often required tailored changes based on application specific requirements.
* However, automation did emerge within the Databricks ETL pipelines, where scripts were developed to handle data ingestion and transformation tasks feeding into Cosmos DB. These scripts contributed to consistency and efficiency across data engineering workflows.

**The Tech Stack:**

* **Old Tech Stack:** Gemfire (in-memory data grid), Java (older versions), Spring Boot (legacy versions)
* **New Tech Stack:** Azure Cosmos DB, Java (upgraded versions), Spring Boot (upgraded), Azure Infrastructure

**The Impact:**

* **Cost Reduction & Licensing Elimination**: Successfully replaced Gemfire, avoiding steep licensing fees and long-term vendor lock-in, generating substantial cost savings for the enterprise.
* **Increased System Resilience & Availability**: By moving to a cloud-native, geo-distributed NoSQL platform, HCSC improved fault tolerance, eliminated single points of failure, and enhanced disaster recovery capabilities addressing Gemfire’s replication and recovery limitations.
* **Standardized and Future-Proofed Data Architecture**: Established repeatable best practices and governance frameworks that enabled consistent, secure, and scalable Cosmos DB adoption across multiple applications. These included role-based access controls, firewall policy definitions, and observability strategies.
* **Accelerated Developer Productivity**: Created developer-friendly sandbox environments and a reusable reference application that significantly reduced onboarding time and improved code quality across teams. This hands-on approach enabled faster iteration and experimentation, overcoming emulator limitations on MacOS.
* **Improved Operational Efficiency**: Transitioned from manual data container management (“Fill & Kill”) to automated, policy-driven retention using Cosmos DB’s native TTL features, streamlining data lifecycle management.
* **Enhanced Collaboration & Enterprise Readiness**: Fostered cross-functional collaboration across architecture, security, and infrastructure teams, ensuring Cosmos DB integration aligned with enterprise standards and compliance requirements.
* **Informed Future Migrations**: The flagship application served as a critical proof of concept, informing smoother and faster migrations for other enterprise applications through reusable code patterns, lessons learned, and improved data modeling strategies.

**The Project Timeline (From Prototype to Full Adoption of Cosmos DB):**

1. **Prototype Phase (Pre-Engagement)**

* A prototype was delivered by Microsoft before Michael LaBrot joined the project.
* This prototype failed to meet enterprise requirements, especially around data modeling, API access patterns, and performance tuning.
* Management assumed this work would provide a sufficient foundation, but it became evident that key architectural and operational gaps remained.

1. **Integration Phase (6 Months)**

* A dedicated six-month effort was launched to integrate Cosmos DB into the flagship application.
* This phase exposed a range of “known unknowns”, including:
* How to structure the data model.
* How to optimize REST API access based on expected access patterns.
* Gaps in the previous assumptions around data architecture.
* The team gathered critical insights but did not reach full readiness, especially regarding performance and consistency during writes.

1. **Summer Planning & Refactoring (Phase 2)**

* The official final implementation phase was delayed by a summer-long debate over:
* The best data modeling strategy.
* How to avoid race conditions during writes.
* The solution involved “materializing views” by creating additional containers to better align with different application access patterns.
* This period was pivotal, reinforcing the importance of getting data strategy right early: modeling, performance, API integration, and cost optimization.

1. **Initial Planning & Design Activities (Concurrent with Phases 2 & 3)**

* Developed an understanding of application access patterns to inform data modeling.
* Enterprise architects defined the integration points and flow between systems.
* The Solution Design & Delivery (SDD) team created comprehensive documentation:
* Spreadsheets outlining RBAC roles, firewall rules, integration governance (IG), and MuleSoft interfaces.

1. **Migration of Application**

* Migration followed a standard software development lifecycle (SDLC).
* Application teams worked within a structured process for developing, testing, and deploying changes.

1. **Operationalization & Full Adoption**

* Applied only to the flagship app, which acted as the test bed for figuring out how to operationalize Cosmos DB for the enterprise.
* Informed best practices, governance decisions, and future-proofed the strategy for other app migrations.
* Subsequent migrations were smoother, thanks to lessons learned and patterns established during this flagship effort.

1. **Developer Sandbox**

* Parallel to early testing, developers initially experimented using the Cosmos DB Emulator, but this proved limiting.
* Shifted to using a sandbox account with a special “SBX” role, offering:
* Controlled but elevated developer access.
* DBA created databases per developer, with freedom to create containers.
* A hands-on, secure environment to explore Cosmos DB features and test integration approaches.

**Team Composition:**

* Michael LaBrot, Senior Software Developer / Architect
* Sriram Krishna (Michael LaBrot reported to Sriram Krishna)
* Sean Griffin (left the company)
* Seema Malhotra
* Lakeisha (handled claims related issues)

**Others:**

* Cosmos DBAs
* Senior Java Developers (onshore)
* Associate Java Developers (offshore)
* Databricks Developers
* Enterprise Architects
* Infrastructure and Security Teams
* QA/Test Teams (mostly offshore)
* DevOps Engineers
* Scrum Masters
* Application Owners

**Key Stakeholders:**

* **Policy Members (Policyholders):** As the end users holding active insurance policies, their interests were indirectly represented through policyholder-facing business units. These units prioritized system reliability, fast response times, and secure access to policy information to ensure seamless customer experiences.
* **Policyholder-Facing Business Units:** These teams were focused on maintaining service quality and operational continuity during the transition. Their primary concern was that the migration should not disrupt existing customer interactions or data access flows.
* **HCSC Digital Leadership:** The digital leadership team sought to modernize the data infrastructure by adopting a scalable, cloud native solution. Their interests included cost optimization, long-term maintainability, and support for future digital initiatives.
* **Center for Enablement (C4E) & Development Teams:** These groups were instrumental in adopting best practices, building reusable components, and ensuring the reference application could be easily extended. They emphasized developer experience, repeatable patterns, and robust documentation.
* **Enterprise Architecture & Cybersecurity Teams:** These stakeholders were involved in designing secure, compliant, and scalable data architectures, including firewall rules, role-based access control (RBAC), and integration with MuleSoft.
* **Quality Assurance/Testing Teams:** Their priority was to ensure the data migration did not introduce regressions or performance issues. They worked closely with the development teams to validate functionality and reliability through rigorous testing.

**Other Vendors:**Accenture and Infosys

**Methodology:** SCRUM mostly. QA/Test Teams practiced waterfall methodology as they were offshore.

**Conclusion:**

The modernization journey at Health Care Service Corporation (HCSC), led by Michael LaBrot and supported by a cross-functional team, showcased how strategic adoption of Azure Cosmos DB successfully replaced a costly and unreliable legacy system - Gemfire, while aligning with broader enterprise goals of digital transformation. By developing best practices, enabling developer agility through sandbox environments, and solving real-world challenges such as emulator limitations and data modeling complexities, HCSC built a repeatable, resilient foundation for future application migrations. The flagship application not only served as a critical test case but also catalyzed the operational frameworks and architectural patterns now guiding enterprise-wide Cosmos DB integration. As a result, HCSC achieved significant cost savings, increased infrastructure reliability, and empowered its development teams. marking a key milestone in its evolution toward a more scalable, cloud native data ecosystem.

**Case Study Prepared By:** Donna Susan Mathew