St10263683

Thando Motha

CloudPOE

Difference between Azure SQL and Cosmos DB

Azure SQL Database and Azure Cosmos DB are both cloud-based database services offered by Microsoft Azure, but they are different in their own respective ways. **Azure SQL Database** is a relational database service enhanced for structured data and SQL-based queries. **Azure Cosmos DB** is a globally distributed, multi-model database service made for flexible data models, massive scalability, and low-latency global access. (Arun Sengottaiyan, 2024)

Key Differences:

1. **Data Model:**

- **Azure SQL Database:** Azure SQL Database is a relational database service based on the SQL Server engine. It uses a tabular data model with structured schemas, supporting traditional SQL queries and transactions. It is suitable for applications with structured data and relational dependencies. (Arun Sengottaiyan, 2024)

- **Azure Cosmos DB:** Azure Cosmos DB is a globally spread, multi-model database service. It supports various data models including document, key-value, graph, and column-family, offering elasticity to store and query data in different setups. Cosmos DB is particularly well-suited for applications with semi-structured or unstructured data and requires high accessibility and scalability. (Arun Sengottaiyan, 2024)

2. Scalability:

-Azure SQL Database: provides scalability options like scaling up or down resources in one database or scaling out with the use of elastic pools. Scaling is primarily vertical rather than horizontal. (Arun Sengottaiyan, 2024)

-Azure Cosmos DB: Gives horizontal scalability with global distribution built in it. It automatically clones data across a lot of regions globally, providing low-latency access to data for users all over the world. Cosmos DB is designed for big scalability and can handle workloads ranging from small apps to larger enterprise solutions. (Arun Sengottaiyan, 2024)

3. Consistency Models:

- Azure SQL Database: It offers strong consistency by default, making sure that transactions are ACID-compliant (Atomicity, Consistency, Isolation, Durability). It follows a traditional consistency model. (Arun Sengottaiyan, 2024)

- **Azure Cosmos DB:** Azure Cosmos DB offers many consistency models including strong, bounded staleness, session, and eventual consistency. It allows developers to pick the suitable consistency level based on their application requirements, balancing between consistency, availability, and latency. (Arun Sengottaiyan, 2024)

4. **Global Distribution:**

- **Azure SQL Database:** While Azure SQL Database supports geo-replication for disaster recovery purposes, it does not offer native global distribution of data with low-latency access across regions. (Arun Sengottaiyan, 2024)

- **Azure Cosmos DB:** Azure Cosmos DB is designed for global distribution from the ground up. It automatically copies data across Azure regions worldwide, allowing low-latency access and high obtainability across geographically discrete locations. (Arun Sengottaiyan, 2024)

5. **Pricing Model:**

- **Azure SQL Database:** Azure SQL Database offers pricing based on compute resources (e.g., vCore-based purchasing model) and storage usage. (Arun Sengottaiyan, 2024)

- **Azure Cosmos DB:** Azure Cosmos DB pricing is based on amount (Request Units or RU/s) and storage usage. Customers pay for the provisioned amount and used storage. (Arun Sengottaiyan, 2024)

Key considerations when designing logic apps:

1. Secure Secret Management:

- Azure Key Vault: Instead of hardcoding delicate information such as passwords or API keys straight into your Logic App, keep them safely in Azure Key Vault.(Learn Microsoft,2024)

- Managed Identities: Use managed identities for confirmation to Azure resources. This removes the need to manage IDs directly and improves security. (Learn Microsoft,2024)

- Secure Parameters: Use safe limits in your logic app workflows to protect delicate information from being evident in the workflow definition. (Learn Microsoft,2024)

2. Secure Communication:

- HTTPS: Always use HTTPS for all communication between your Logic App and external services or applications to guarantee encrypted data transmission. (Learn Microsoft,2024)

- Virtual Network Integration: Integrate your Logic App with a virtual network to control incoming and outgoing traffic, improving network security. (Learn Microsoft,2024)

- Private Endpoints: Consider using private endpoints to access backend services within your virtual network securely. (Learn Microsoft,2024)

- Network Security Groups (NSGs): Implement NSGs to restrict network traffic to your Logic App based on rules for ports, protocols, and IP addresses. (Learn Microsoft,2024)

3. Minimize Data Exposure:

- Secure Inputs/Outputs: Protect the inputs and outputs of your Logic App activities to avoid delicate data from being logged or displayed unnecessarily.(AzureTechInsider,2024)

- Data Encryption: Encrypt delicate data both in transit and at rest to keep it from unauthorized access.(DEV Community,2024)

- Least Privilege: Grant your Logic App the smallest amount of access necessary to perform its tasks, lowering the potential impact of a security break. (Learn Microsoft,2024)

4. Security Best Practices:

- Regular Audits: Conduct consistent security checks of your Logic Apps to find and report any potential weaknesses. (DEV Community,2024)

- Monitoring and Alerting: Implement strong checking and notifying instruments to sense and respond to security incidents punctually. (Learn Microsoft,2024)

- Least Privilege: Apply the principle of least privilege to all resources your Logic App relates with, guaranteeing only necessary approvals are decided. (Learn Microsoft,2024)

- Regular Updates: Keep your Logic App and its dependencies up to date with the latest security patches and updates. (DEV Community,2024)

- **Compliance**: Confirm your Logic Apps comply with relevant data privacy regulations and security standards. (DEV Community,2024)

Explain how combining event grid with other services can create robust workflows

Event-Driven Architecture with Azure Event Grid and Service Bus:

Combining Azure Event Grid and Azure Service Bus makes a strong foundation for building scalable, event-driven architectures. This integration influences the strengths of both services to create robust, elastic systems capable of managing complex event processing scenarios.(Multishoring,2024)

The Azure Event Grid architecture shines at real-time event directing and delivery. It can quickly fan out events to many subscribers, making it ideal for scenarios where instant notification is crucial. On the other hand, the Azure Service Bus event driven architecture delivers consistent message queuing and pub/sub capabilities, which are important for scenarios requiring guaranteed message delivery and processing. (Multishoring,2024)

When used together, these services match each other well. Event Grid can act as the original entry point for events, quickly directing them to numerous endpoints, including Service Bus queues or topics. Service Bus then takes over, delivering advanced message handling capabilities such as ordered delivery, transaction support, and message sessions. (Multishoring,2024)

Key components that make up the azure integration services architecture:

Azure Event Grid:

Event Grid is the support for event routing in Azure. It efficiently handles the delivery of events from many sources to multiple ends, enabling real-time event-driven architectures. (Multishoring,2024)

Azure Service Bus:

Service Bus provides enterprise-grade message queuing and publish-subscribe messaging capabilities. It's important for scenarios requiring guaranteed message delivery, transaction support, and complex message processing. (Multishoring,2024)

Azure Logic Apps:

Logic Apps allow you to systematize workflows and business processes. They can be activated by events from Event Grid or messages from Service Bus, making them a critical factor in event-driven architecture. Logic Apps give a visual designer for creating complex workflows without writing code. (Multishoring,2024)

Azure Functions:

These serverless compute services are perfect for event-driven scenarios. Purposes can be activated by events from Event Grid or messages from Service Bus, allowing you to perform code in reply to specific events without managing infrastructure. (Multishoring,2024)

Azure API Management:

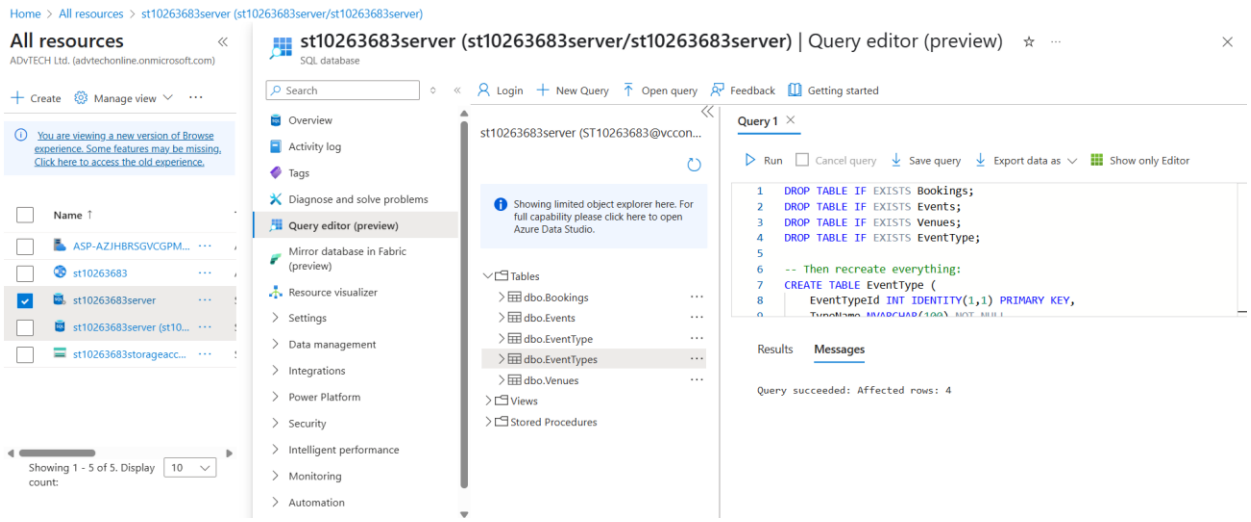
API Management is a front for your backend services, offering a combined interface for outside consumers. It can mix with Event Grid to publish events when APIs are called, allowing event-driven patterns in API-centric architectures. (Multishoring,2024)

Azure Storage:

While not firmly an integration service, Azure Storage plays a vital role in many integration scenarios. Blob Storage, for instance, can activate events when files are added or improved, which can then be processed using Event Grid and other services. (Multishoring,2024)

Azure Monitor:

This service offers complete monitoring capabilities across your Azure resources. It's vital for keeping visibility into your integration solutions, allowing you to track performance, set up alerts, and gain insights into your system's behavior. (Multishoring,2024)



Reflective Technical Report: EventEasePOE

1. System Features and Requirements

The EventEasePOE web application is intended to make event and venue management for users simpler. It offers the following key features:

Core Features:

- **User-friendly Interface:** Clean and available web interface using ASP.NET Core MVC. (Microsoft, 2023)
- **Venue Management:** Add, edit, view, and delete venues with Azure Blob Storage for image uploads. (Microsoft, 2023)
- **Event Management:** CRUD actions for events linked to venues.
- **Booking Management:** Capability to book events at venues, with following of customer name, date, event, and venue. (Microsoft, 2023)
- **Event Type Classification:** EventType lookup table with pre-defined groups to categorize events.
- **Advanced Filtering:** Search functionality improved with filters for Event Type, Venue Availability, and Booking Date Choices.
- **Azure Blob Storage Integration:** Safe image storage for venue images.
- **Azure SQL Database:** Unified, cloud-hosted database for data storage. (Microsoft, 2023)
- **Azure Web App Deployment:** The application is deployed to Azure for availability and scalability.

2. Components and Azure Services Discussion

Azure Services Used:

Component	Azure Service	Justification
Web Application	Azure App Service (Web App)	Provides simplified deployment and automatic scaling of the ASP.NET Core MVC app. (Microsoft, 2023)

Database	Azure SQL Database	A managed, relational database ideal for complex relationships and filtering. (Microsoft, 2023)
Image Storage	Azure Blob Storage	Secure, cost-effective cloud storage for venue images. (Microsoft, 2023)
Search & Filtering	Server-side filtering via EF Core and SQL	Allows advanced filtering with efficient query processing. (Microsoft, 2023)

Alternatives Considered:

Service	Alternative	Reason for Rejection
Azure Blob Storage	Amazon S3, Google Cloud Storage	Azure integration provides seamless development experience and consistent security models.
Azure SQL Database	Cosmos DB (NoSQL)	While Cosmos DB offers global distribution, relational SQL was better suited for this system's joins and filters. (Microsoft, 2023)
Azure Web App	Virtual Machines (IaaS)	Web App abstracts infrastructure management, reducing complexity and improving scalability.

3. Technologies Used and Justification

- **ASP.NET Core MVC:** Selected for its strong structure, separation of concerns, and fast development abilities. (Microsoft, 2023)
- **Entity Framework Core:** Allows effective database operations with LINQ and migrations. (Microsoft, 2023)
- **Azure Blob Storage:** Handles venue image uploads strongly. (Microsoft, 2023)
- **Bootstrap:** Delivers receptive, mobile-friendly frontend design.

- **Azure SQL Database:** Dependable and scalable cloud-hosted database service. (Microsoft, 2023)

4. Reflection on Project Journey

Throughout the development of the EventEasePOE project, I learned and gained valued experience in building, deploying, and maintaining a cloud-based application. Key reflections include:

Challenges Faced:

- **Azure Resource Management:** struggled with navigating Azure, setting up storage containers, and configuring SQL Database authorizations. (Microsoft, 2023)
- **Data Relationships:** Designing relationships between Events, Venues, Event Types, and Bookings required careful planning to evade data redundancy.
- **Deployment Hurdles:** Overcoming issues related to Azure deployment, such as correct connection strings and resource limitations. (Microsoft, 2023)

Lessons Learned:

- **Cloud Application Design:** Knowing the importance of cloud scalability, security, and cost-efficiency.
- **Azure Ecosystem:** experience with multiple Azure services, showing how integrated cloud services improve output. (Microsoft, 2023)
- **Security Considerations:** Learned to restrict image upload functionality to the Venue controller as per project requirements, refining system integrity.
- **Filtering and Search:** Applying progressive filtering taught me how to structure queries and enhance user experience.

Understanding of Cloud-Based Systems:

I now understand the complexity of building scalable, reliable, and secure cloud-based applications. I understand how many Azure services work together, from storage to computing to databases, and how to design a system that is both maintainable and efficient. (Microsoft, 2023)

Reference List

1.Microsoft (2023a) 'Introduction to Azure App Service', Microsoft Learn. Available at: https://learn.microsoft.com/en-us/azure/app-service/ (Accessed: 30 May 2025).

2.Microsoft (2023b) 'What is Azure SQL Database?', Microsoft Learn. Available at: https://learn.microsoft.com/en-us/azure/azure-sql/database/sql-database-overview (Accessed: 30 May 2025).

3.Microsoft (2023c) 'Introduction to Azure Blob Storage', Microsoft Learn. Available at: https://learn.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction (Accessed: 30 May 2025).

4.Microsoft (2023d) 'Introduction to Azure Cosmos DB', Microsoft Learn. Available at: https://learn.microsoft.com/en-us/azure/cosmos-db/introduction (Accessed: 30 May 2025).

5.Microsoft (2023e) 'Introduction to ASP.NET Core MVC', Microsoft Learn. Available at: https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-7.0 (Accessed: 30 May 2025).

6.Sengottaiyan,A. (2024, April 23). “Difference between Azure SQL and Cosmos DB”Check it here: [https://www.linkedin.com/pulse/difference-between-azure-sql-cosmos-db-arun-sengottaiyan-dhwrc] (Accessed 22 June 2025)

7.Learn Microsoft.(2025) SQL vs Cosmos database.[Online] Available at:https://learn.microsoft.com/en-us/answers/questions/2084372/sql-vs-cosmos-database [accessed 23 June 2025]

8.Multishoring.(2024) Building Scalable Event-Driven Architectures with Azure EventGrid and Service Bus.[online] Available at: https://multishoring.com/blog/building-scalable-event-driven-architectures-with-azure-event-grid-and-service-bus/ [Accessed 22 June 2025]

9.DEV Community.(2023) Securing Azure Logic Apps Best Practices and Considerations.[Online] Available: https://dev.to/sardarmudassaralikhan/securing-azure-logic-apps-best-practices-and-considerations-3h2o [Accessed 22 June 2025]