



Assessment (Week 6) Fall 2023

DATA WAREHOUSING AND ANALYTICS IN THE CLOUD

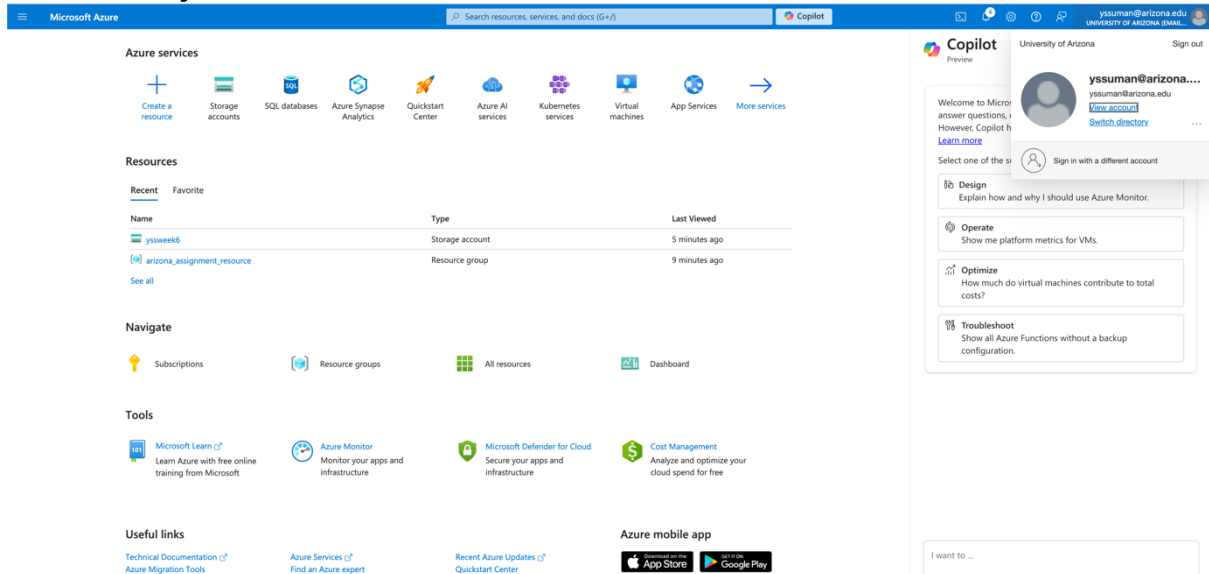
SUMAN, YATENDER SINGH

Submitted To: Dr. Nayeem Rehman

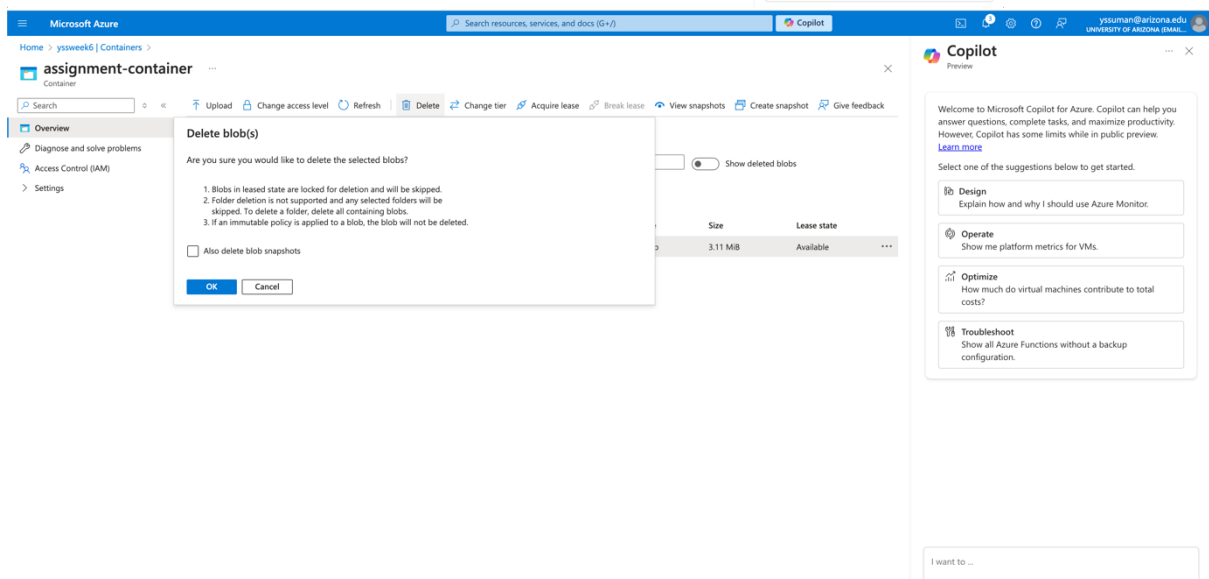
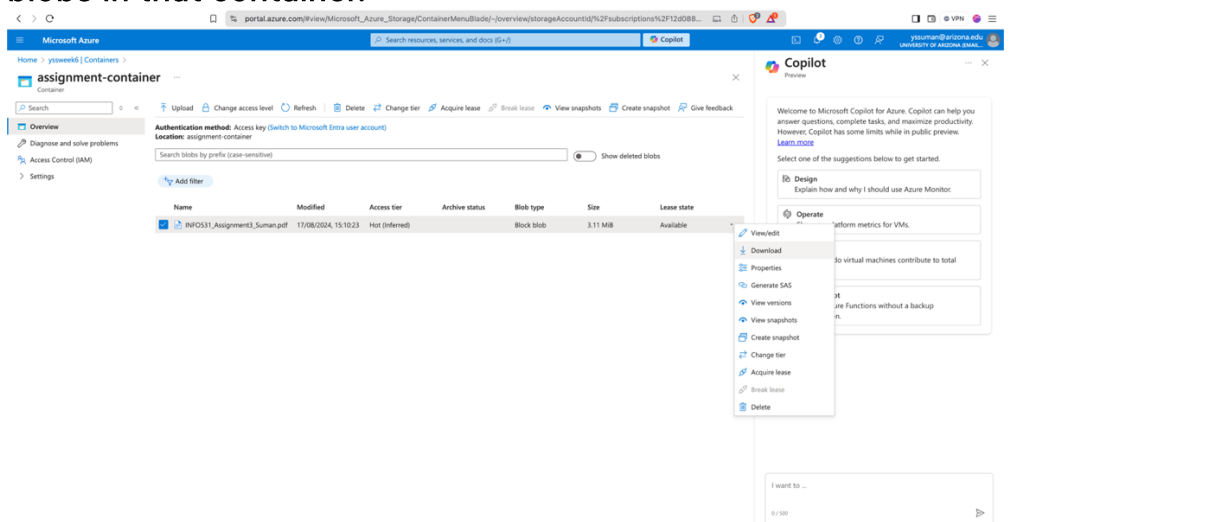
On: 18 Aug 2024



Q1. Create your Azure free account



Q2. Create a container in Azure Storage, and to upload and download block blobs in that container.



Q3. Create a SQL database on Azure.

Home > SQL databases > Create SQL Database >

Create SQL Database Server

Microsoft

Server details

Enter required settings for this server, including providing a name and location. This server will be created in the same subscription and resource group as your database.

Server name *

assignment-db-dev

.database.windows.net

Location *

(Europe) Germany West Central

Authentication

Azure Active Directory (Azure AD) is now Microsoft Entra ID. [Learn more](#)

Select your preferred authentication method for accessing this server. Create a server admin login and password to access your server with SQL authentication, select only Microsoft Entra authentication [Learn more](#) or using an existing Microsoft Entra user, group, or application as Microsoft Entra admin [Learn more](#) or, or select both SQL and Microsoft Entra authentication.

Authentication method

☐ Use Microsoft Entra-only authentication

☐ Use both SQL and Microsoft Entra authentication

☒ Use SQL authentication

Server admin login *

assignment_admin

Password *

Confirm password *

✔ Password and confirm password must match.

OK

Home >

Microsoft.SQLDatabase.newDatabaseNewServer_388c46c5e7bc4906bc5ba | Overview

Deployment

Search

Delete Cancel Redeploy Download Refresh

Overview

Inputs

Outputs

Template

Deployment is in progress

Deployment name : Microsoft.SQLDatabase.newDatabaseNewServer_388c46c5e7bc4906bc5ba

Subscription : Azure for Students

Resource group : anizona_assignment_resource

Start time : 17/08/2024, 15:59:27

Correlation ID : f584907d-3180-4862-9a16-967b142145ae

Deployment details

Resource	Type	Status	Operation details
There are no resources to display.			

Give feedback

Tell us about your experience with deployment

Microsoft Defender for Cloud

Secure your apps and infrastructure

Go to Microsoft Defender for Cloud >

Free Microsoft tutorials

Start learning today >

Work with an expert

Azure experts are service provider partners who can help manage your assets on Azure and be your first line of support.

Find an Azure expert >

Home >

Microsoft.SQLDatabase.newDatabaseNewServer_388c46c5e7bc4906bc5ba | Overview

Deployment

Search

DeleteCancelRedeployDownloadRefresh

OverviewInputsOutputsTemplate

Your deployment is complete

Deployment name : Microsoft.SQLDatabase.newDatabaseNewServer_388c46c5e7bc4906bc5ba
Subscription : Azure for Students
Resource group : arizona_assignment_resource

Start time : 17/08/2024, 15:59:27
Correlation ID : f584907d-3180-4862-9a16-967b142145ae

Deployment details

Resource	Type	Status	Operation details
assignment-db-dev/Clientip-2024-8-17_15-59-17	Microsoft.Sql/servers/firewallrul	OK	Operation details
assignment-db-dev/AllowAllWindowsAzureIps	Microsoft.Sql/servers/firewallrul	OK	Operation details
assignment-db-dev/assignment_db_dev	Microsoft.Sql/servers/databases	Created	Operation details
assignment-db-dev/Default	Microsoft.Sql/servers/connectio	OK	Operation details
assignment-db-dev	Microsoft.Sql/servers	OK	Operation details
assignment-db-dev	Microsoft.Sql/servers	Created	Operation details

Next steps

Go to resource

Give feedback

Tell us about your experience with deployment

portal.azure.com/#view/HubsExtension/DeploymentDetailsBlade~/overview/id/%2Fsubscriptions%2F12d0880d-2b11-44a8-af5...

Microsoft Azure

Search resources, services, and docs (G+I)

Copilot

ysunamuniv

Home >

Microsoft.SQLDatabase.newDatabaseNewServer_388c46c5e7bc4906bc5ba | Overview

Deployment

Search

DeleteCancelRedeployDownloadRefresh

OverviewInputsOutputsTemplate

Deployment is in progress

Deployment name : Microsoft.SQLDatabase.newDatabaseNewServer_388c46c5e7bc4906bc5ba
Subscription : Azure for Students
Resource group : arizona_assignment_resource

Start time : 17/08/2024, 15:59:27
Correlation ID : f584907d-3180-4862-9a16-967b142145ae

Deployment details

Resource	Type	Status	Operation details
assignment-db-dev	Microsoft.Sql/servers	Accepted	Operation details

Give feedback

Tell us about your experience with deployment

Microsoft Defender for Cloud

Secure your apps and infrast

Go to Microsoft Defender fo

Free Microsoft tutorials

Start learning today >

Work with an expert

Azure experts are service prc who can help manage your s and be your first line of supp

Find an Azure expert >

assignment_db_dev (assignment-db-dev/assignment_db_dev) | Compute + storage

SQL database

Search

Feedback

OverviewActivity logTagsDiagnose and solve problemsQuery editor (preview)Mirror database in Fabric (preview)SettingsCompute + storageConnection stringsPropertiesLocksData managementIntegrationsPower PlatformSecurityIntelligent performanceMonitoringAutomationHelp

Hardware Configuration

Standard-series (Gen5)
up to 80 vCores, up to 240 GB memory
Change configuration

General Purpose (GP_5_Gen5_1)
Cost per GB (in USD)
0.14
Max storage selected (in GB)
x 41.6
ESTIMATED STORAGE COST / MONTH
5.70 USD
COMPUTE COST / VCORE SECOND
0.000159 USD

Max vCores

Min vCores

2.02 GB MIN MEMORY 3 GB MAX MEMORY

0.5 vCores

Auto-pause delay

The database automatically pauses if it is inactive for the time period specified here, and automatically resumes when database activity recurs. Alternatively, auto-pausing can be disabled.

Enable auto-pause

DaysHoursMinutes

010

Data max size (GB)

32

9.6 GB LOG SPACE ALLOCATED

Would you like to make this database zone redundant?

YesNo

Backup storage redundancy

Locally-redundant backup storage

Zone-redundant backup storage

Geo-redundant backup storage

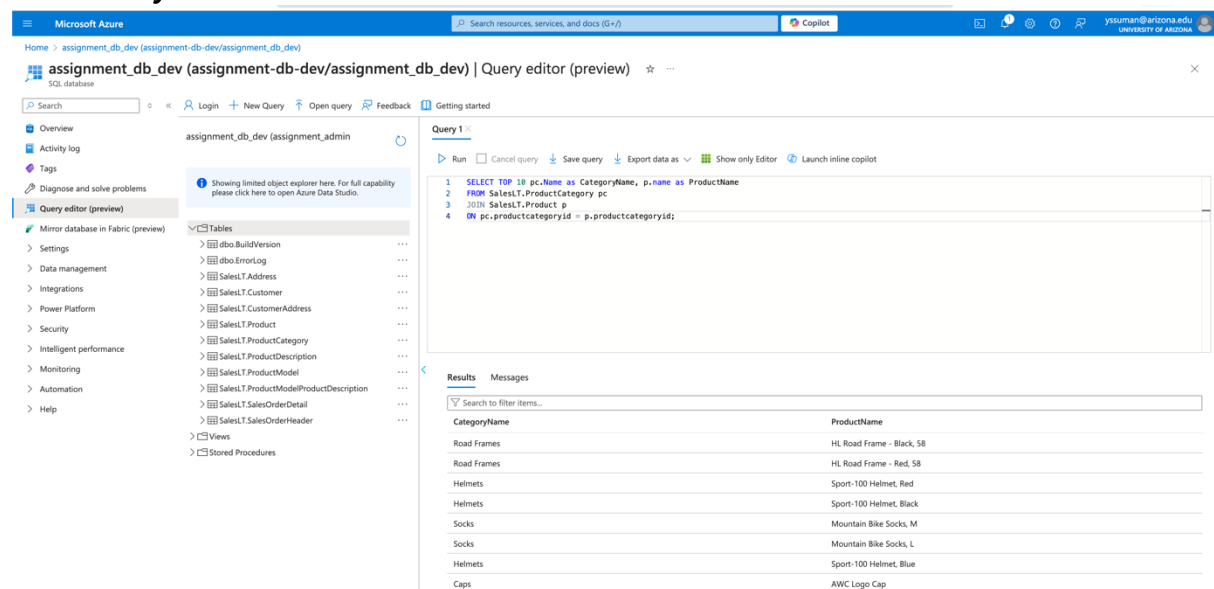
Apply

Q4. Summarize SQL database creation in Azure.

I've just created my first Azure SQL Database through the Azure Portal, and it's been quite an enlightening experience. Navigating through the portal, I learned how to provision a cloud-based relational database from scratch. I had to make decisions about the performance tier, storage capacity, and server location - opting for the Basic tier with 5 DTUs and 2 GB of storage to keep costs low for my development needs. It was interesting to see how Azure's pricing model works for SQL databases, especially understanding concepts like DTUs and their impact on performance.

This hands-on experience with cloud resource management has shown me how flexible and scalable cloud services can be. I'm impressed by how easy it was to set up a fully managed database service without worrying about the underlying infrastructure. It's clear that this approach could significantly simplify database management tasks for development and testing. Overall, I feel I've gained practical insights into cloud services and how they can provide a cost-effective platform for database-driven application development.

Q5. Query the SQL database in Azure



The screenshot displays the Microsoft Azure portal interface for a SQL database named 'assignment_db_dev'. The 'Query editor (preview)' is open, showing a SQL query that selects the top 10 products from the 'SalesLT' schema, joining 'ProductCategory' and 'Product' tables. The results are displayed in a table with two columns: 'CategoryName' and 'ProductName'.

CategoryName	ProductName
Road Frames	HL Road Frame - Black, 58
Road Frames	HL Road Frame - Red, 58
Helmets	Sport-100 Helmet, Red
Helmets	Sport-100 Helmet, Black
Socks	Mountain Bike Socks, M
Socks	Mountain Bike Socks, L
Helmets	Sport-100 Helmet, Blue
Caps	AWC Logo Cap

Q6. Describe the SQL database "SalesLT" on Azure.

1. Database Structure:

- Contains 10 tables in the **SalesLT** schema
 - Address
 - Customer
 - CustomerAddress
 - Product
 - ProductCategory
 - ProductDescription
 - ProductModel
 - ProductModelProductDescription
 - SalesOrderDetail

- SalesOrderHeader

2. Primary Keys:

- Each table has at least one primary key column
- Most use single-column integer primary keys (e.g., **CustomerID**, **ProductID**)
- Some tables have composite primary keys (e.g., **CustomerAddress**,

ProductModelProductDescription)

3. Common Columns:

- Most tables include '**rowguid**' (uniqueidentifier) and '**ModifiedDate**' (datetime)
- Likely used for tracking and synchronization purposes

4. Data Types:

- int: Used for ID columns
- nvarchar: Common for text fields
- datetime: Used for date and time information
- money: Used for financial values
- uniqueidentifier: Used for rowguid columns
- Custom types: Name, Phone, NameStyle, Flag, OrderNumber, AccountNumber

5. Normalization:

- Database appears to follow normalization principles
- Customer data split between Customer and Address tables
- Product information distributed across multiple tables

6. Relationships:

- **CustomerAddress** acts as a junction table between Customer and Address
- **SalesOrderHeader** and **SalesOrderDetail** form the core of the sales process
- Foreign key relationships evident (e.g., **ProductCategoryID** in **Product** table)

7. Nullable Fields:

- Most columns are non-nullable (is_nullable = False)
- Some optional fields allow NULL values (e.g., MiddleName, Suffix, Color)

8. Sales Process:

- **SalesOrderHeader** contains overall order information
- **SalesOrderDetail** stores individual line items for each order

9. Product Management:

- Hierarchical structure with **ProductCategory** table
- Separate tables for **ProductDescription** and **ProductModel**

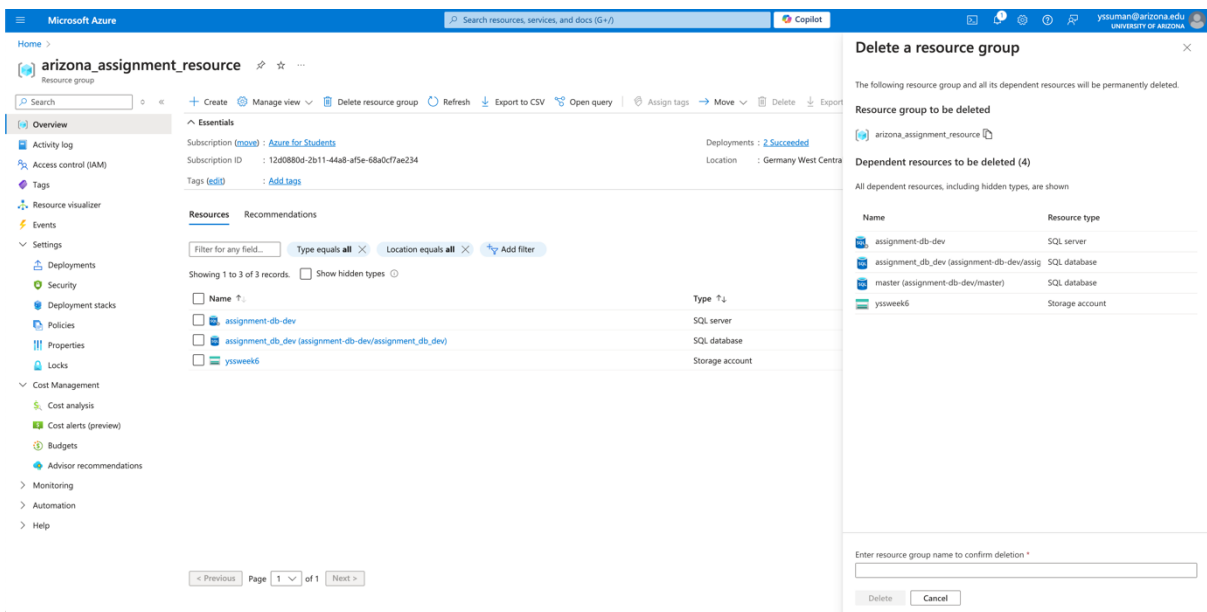
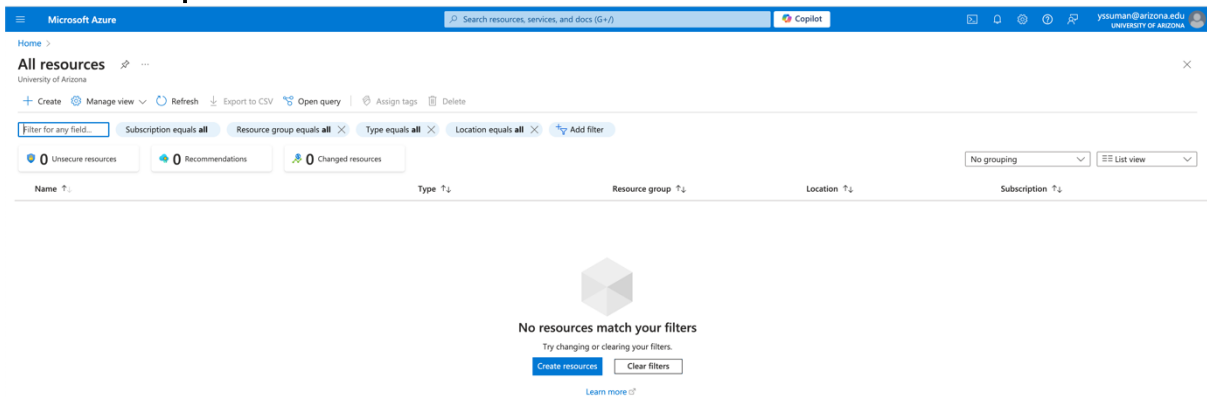
10. Data Integrity:

- Use of primary keys and non-nullable columns ensures data integrity
- Appropriate data types chosen for different kinds of information

11. Tracking and Auditing:

- Consistent use of **ModifiedDate** suggests capability for change tracking
- **rowguid** could be used for replication or unique identification across systems

Q7. Clean up resources in Azure



Q8. Summarize the article on "Understanding Data Store Models"

Relational Database Management Systems (RDBMS): These are great for structured data and complex queries. They use SQL and provide strong consistency guarantees.

Key/Value Stores: Optimized for simple lookups using unique keys. They're highly scalable but less suitable for complex queries.

Document Databases: These store data in flexible, JSON-like documents. They're good for applications where data structures might change over time.

Graph Databases: Designed for managing and querying highly connected data. They excel at analyzing relationships between entities.

Data Analytics Stores: These are built for processing and analyzing large volumes of data, often using parallel processing techniques.

Column-family Databases: Good for managing structured, volatile data and handling high write throughput.

Search Engine Databases: Specialized for full-text search and complex querying across large volumes of text data.

Time Series Databases: Optimized for time-ordered data, often used for IoT and monitoring applications.

Object Storage: Designed for storing and retrieving large binary objects like images, videos, and backup files.

Shared Files: Simple file-based storage accessible across a network, useful for legacy applications or shared content.

This article indicates that choosing the right data store depends on factors like the type of data being stored, the required query patterns, scalability needs, and consistency requirements. It's also worth noting that many cloud providers, including Azure, offer services that cater to these different data store models.

In conclusion, understanding these different data store models is crucial for designing effective data management solutions in modern applications. It's clear that there's no one-size-fits-all solution, and the choice of data store should be carefully considered based on the specific requirements of each application or use case.

Q9. Summarize the article on "Understanding the differences between NoSQL and relational databases".

Challenges with Traditional Databases:

- Many database engines use locks and latches to enforce strict ACID semantics.
- This approach provides high consistency but compromises concurrency, latency, and availability.
- High transactional workloads often require manual workarounds like sharding data across multiple databases or nodes.

NoSQL Databases:

- Designed to simplify horizontal scaling by adjusting consistency levels.
- Offer configurable consistency to balance trade-offs between speed, availability, and data consistency.
- Allow for better scalability across multiple nodes.

Distributed Databases:

- Scale across multiple instances or locations.

- Not all NoSQL databases are distributed by default.
- Implementing a globally distributed database can be complex and time-consuming.

Azure Cosmos DB:

- A database platform offering both NoSQL and relational distributed data APIs.
- Provides various consistency options for fine-tuning based on application requirements.
- Automatically distributes data across local instances or globally.
- Offers ACID guarantees and scalable throughput.
- Simplifies the process of building and managing high transactional workloads.

Benefits of Distributed NoSQL Databases:

- Enable easier management of high transactional workloads.
- Allow for flexible consistency configurations to meet specific application needs.
- Provide options for balancing consistency, performance, and availability.
- The article emphasizes that Azure Cosmos DB combines the benefits of NoSQL and distributed databases, offering a flexible solution for various data storage and processing needs. It allows developers to choose the appropriate consistency level and scale their databases efficiently without the complexities typically associated with distributed systems.

Q10. Summarize the article on "Understanding Azure Cosmos DB".

Here are the key-points we observed from the article.

- Azure Cosmos DB is a fully managed NoSQL, relational, and vector database that simplifies development of responsive and always-online applications. It provides single-digit millisecond response times, automatic scalability, and guaranteed speed at any scale.

- It accommodates multiple data models including document, vector, key-value, graph, and table to serve as a unified database for application needs. This is especially helpful for AI-powered apps that often integrate multiple data stores.

- Key capabilities include turnkey global data distribution, open source APIs and SDKs, and AI database functionalities like vector indexing and integration with Azure AI services.

- Various free options are available, including a lifetime free tier providing 1000 RU/s throughput and 25 GB storage, and an Azure AI Advantage offer of 40,000 RU/s for 90 days for Azure AI/GitHub Copilot customers.

- It is a good fit for apps requiring flexible schema, low latency, high elasticity, high throughput, and mission-critical availability. It is less suited for analytical workloads and highly relational apps.

- Benefits include guaranteed speed at scale, simplified development with many APIs/SDKs, mission-critical reliability, and fully managed cost-effective operation.

- Azure Synapse Link enables near real-time analytics over operational data in Cosmos DB without ETL.

In summary, Azure Cosmos DB provides a versatile, performant and highly scalable database platform that simplifies development of modern, globally distributed applications, especially those powered by AI. Its multi-model support and tight integration with Azure AI services makes it attractive as a unified data store.

Q11. Summarize the article on "Azure Cosmos DB API for MongoDB".

Summary of the key points from the article on Azure Cosmos DB API for MongoDB:

- Azure Cosmos DB for MongoDB makes it easy to use Azure Cosmos DB as if it were a MongoDB database. You can continue using your existing MongoDB skills, drivers, SDKs, and tools.

- It offers two architectures:

1. vCore architecture (recommended): Provides dedicated instances, instantaneous scaling, and native integration with Azure services. Includes features like native vector search, flat pricing, text indexes, vertical scaling without shard key, and free 35-day backups with point-in-time restore.

2. Request Unit (RU) architecture: Flexible scaling using Request Units, designed for cloud-native apps. Offers instantaneous scalability with autoscale, automatic sharding, 99.999% availability, active-active multi-region support, granular unlimited scaling, real-time analytics with Azure Synapse Link, and serverless deployments.

- Azure Cosmos DB for MongoDB implements the MongoDB's wire protocol for transparent compatibility with MongoDB client SDKs, drivers and tools.

- It doesn't host the MongoDB database engine itself. Rather, it provides a MongoDB-compatible API layer over Azure Cosmos DB.

- Microsoft doesn't run actual MongoDB databases to provide this service. Azure Cosmos DB is not affiliated with MongoDB, Inc.

In summary, Azure Cosmos DB for MongoDB offers a fully managed, highly scalable MongoDB-compatible database service on Azure. It provides the familiarity of MongoDB with the enterprise capabilities, global distribution, and integration with Azure services of Azure Cosmos DB. The service offers compelling features over other MongoDB offerings like MongoDB Atlas.

Q12. Summarize the article on "Nodes and tables in Azure Database for PostgreSQL – Hyperscale".

Summary of the key points from the article on nodes and tables in Azure Cosmos DB for PostgreSQL:

- Azure Cosmos DB for PostgreSQL uses a "shared nothing" architecture where PostgreSQL servers (called nodes) coordinate with each other. This allows the cluster to collectively hold more data and CPU cores than a single server.

- Every cluster has a coordinator node and multiple worker nodes. Applications send queries to the coordinator, which relays them to relevant workers and accumulates results.

- The database administrator can distribute tables and/or schemas across worker nodes. Distributed tables/schemas are key to performance by enabling cross-machine parallelism.

- There are five types of tables:

1. Distributed tables: Horizontally partitioned across worker nodes into shards based on a distribution column.

2. Reference tables: Distributed tables whose entire contents are concentrated into a single shard replicated on every worker for local access.

3. Local tables: Regular non-sharded tables on the coordinator node, good for small admin tables not used in joins.

4. Local managed tables: Created automatically due to foreign key references with reference tables, or manually using a function. Can be queried from any node.

5. Schema tables: With schema-based sharding, tables in distributed schemas are automatically converted to colocated distributed tables without a shard key.

- Shards are identified by an ID and hash range stored in metadata tables. The coordinator uses the hash of the distribution column value to determine which shard contains a row.

- The mapping of shards to worker nodes is called shard placement, also tracked in metadata tables. The coordinator rewrites queries with shard-specific table names to execute on the appropriate workers.

In summary, Azure Cosmos DB for PostgreSQL scales by distributing data across multiple coordinated PostgreSQL servers using sharding. The choice of table types and distribution columns allows performance optimization through parallelism.

Q13. Summarize the article on "Overview - Azure Database for PostgreSQL - Flexible Server".

Summary of the key points from the article "Overview - Azure Database for PostgreSQL - Flexible Server":

- Azure Database for PostgreSQL Flexible Server is a fully managed database service that provides more granular control and flexibility over database management and configuration compared to the Single Server deployment option.

- Key features include:

- Support for major community versions of PostgreSQL
- Ability to colocate database engine with client tier for lower latency
- High availability options within a single availability zone or across multiple zones
- Better cost optimization controls with stop/start capability and burstable compute tier
- Automated patching with managed maintenance windows
- Automatic backups with configurable retention period
- Adjustable performance and scaling
- Enterprise-grade security with data encryption at rest and in motion
- Monitoring and alerting capabilities

- Flexible Server uses a decoupled architecture that separates compute and storage. The database engine runs in a container on a Linux VM, while data files reside on Azure storage with 3 locally redundant copies.

- Zone redundant high availability is supported by provisioning a warm standby across availability zones with synchronous replication.

- The service performs automated patching of the underlying infrastructure and database engine. Minor version upgrades are included.

- Backups are automatically created and stored in ZRS. Retention period is configurable up to 35 days.

- Compute tiers include Burstable for low-cost dev/test, General Purpose, and Memory Optimized for production workloads. Scaling can be adjusted as needed.

- Security features include FIPS 140-2 validated encryption, TLS 1.2 enforcement, and private access via VNet integration.
- Built-in monitoring includes metrics with 1-minute frequency and 30-day history, configurable alerts, and host server metrics.
- A built-in **PgBouncer** connection pooler is available.
- The service is available in many Azure regions worldwide, with varying support for compute generations, high availability, and backup options per region.
- Migration options include an Azure DB for PostgreSQL migration tool, dump/restore, and Azure Database Migration Service.

In summary, Azure Database for PostgreSQL Flexible Server provides a more customizable, full-featured managed PostgreSQL offering in Azure compared to Single Server, with key benefits around performance, scalability, availability, security and cost optimization.

Q14. Summarize the article on "Azure Database for PostgreSQL - Single Server".

Summary of the key points from the article "Servers - Azure Database for PostgreSQL - Single Server":

- An Azure Database for PostgreSQL server in the Single Server deployment option acts as a central administrative point for multiple databases. It provides the same PostgreSQL server construct found in on-premises environments.
- A server is created within an Azure subscription, is the parent resource for databases, provides a namespace for databases, and is a container with strong lifetime semantics.
- The server collocates resources in a region and provides a connection endpoint for server and database access.
- Management policies that apply to databases, such as login, firewall, users, roles, and configurations, are scoped at the server level.
- Servers are available in multiple versions and can be extended by users through PostgreSQL extensions.
- Within a server, you can create one or multiple databases. Pricing is structured per-server based on the pricing tier, vCores, and storage.
- To connect and authenticate to a server, PostgreSQL's native authentication is supported. Connections use a message-based protocol over TCP/IP.

- Firewall rules prevent access to the server and its databases until permitted computers are specified.
- Server management can be done through the Azure portal or Azure CLI.
- The admin user set up during server creation has the highest privileges (azure_pg_admin role), but does not have full superuser permissions. The PostgreSQL superuser attribute is assigned to the azure_superuser role, which is not accessible.
- Default databases include postgres, azure_maintenance, and azure_sys (for Query Store).
- PostgreSQL server parameters determine the server configuration. A subset of parameters compared to on-premises PostgreSQL can be viewed and edited through the Azure portal or CLI.

In summary, in the Azure Database for PostgreSQL Single Server option, the server is the central administrative unit that provides management, security, and configuration scope over the databases it contains, similar to on-premises PostgreSQL servers but with some differences due to the managed nature of the service.

Q15. Summarize the article on "What is Azure Database for PostgreSQL?".

Azure Database for PostgreSQL is a relational database service in the Microsoft cloud based on the open-source PostgreSQL database engine. It has two deployment modes:

1. Flexible Server
2. Single Server

Flexible Server:

- Provides more granular control and flexibility over database management and configuration
- Allows high availability within a single availability zone or across multiple zones
- Offers cost optimization with ability to stop/start server and burstable compute tier
- Supports PostgreSQL versions 16, 15, 14, 13, 12, 11
- Best suited for apps requiring more control and customization, cost optimization, zone redundant high availability

Single Server:

- Fully managed service with minimal requirements for customizations
- Architecture optimized for built-in high availability (99.99% in single zone)
- Automated patching, backups, security with minimal user configuration
- Supports PostgreSQL versions 9.5, 9.6, 10, 11
- Three pricing tiers: Basic, General Purpose, Memory Optimized with dynamic scalability
- Best for cloud native apps designed to handle automated patching without granular control.

Both offer capabilities like:

- Built-in high availability
- Data protection with automatic backups and point-in-time restore
- Automated maintenance
- Predictable performance with pay-as-you-go pricing
- Enterprise grade security and compliance
- Monitoring and automation to simplify management

In summary, Azure provides two managed PostgreSQL options - Flexible Server for more control and Single Server for ease of use, allowing customers to focus on app development rather than database administration. The choice depends on the level of customization, control, and management required.

Q16. Summarize the article on "Azure Database for MySQL - Flexible Server".

The article "**Flexible server deployment model overview - Azure Database for MySQL - Flexible Server**" provides an overview of Azure Database for MySQL Flexible Server, a fully managed relational database service based on the MySQL Community Edition. Key points include:

1. Flexible Server offers zone redundant and same zone high availability options, data protection with automatic backups, automated maintenance, elastic scaling, and cost optimization controls.
2. It supports MySQL versions 5.7 and 8.0, and provides three compute tiers: Burstable, General Purpose, and Business Critical.
3. High availability options include Zone Redundant HA for infrastructure redundancy across availability zones, and Same-Zone HA within a single zone.
4. Automated patching and maintenance occur during configurable maintenance windows. Automatic backups are stored in redundant storage with retention up to 35 days.
5. Network isolation is provided through private access (VNet integration) or public access (allowed IP addresses).
6. Performance can be adjusted within seconds. Storage scaling is online and supports autogrowth. Additional IOPS can be provisioned up to 80K.
7. Read workloads can be scaled out using up to 10 read replicas. Hybrid or multicloud data synchronization is enabled through data-in replication.
8. Servers can be stopped/started to optimize costs. Enterprise-grade security includes encryption at-rest and in-transit, virtual network support, monitoring and alerting.

9. Offline and online migration options are available from on-premises or other cloud databases to Flexible Server.

10. The service is available in many Azure regions globally with varying levels of high availability and backup redundancy options.

In summary, Azure Database for MySQL Flexible Server provides a scalable, secure, highly available managed MySQL database service with granular control and flexibility over configuration and management.

Q17. Summarize the article on "Azure Database for MySQL Single Server".

The article "Overview - Azure Database for MySQL single server" provides an overview of the Single Server deployment model for Azure Database for MySQL, a fully managed database service based on the MySQL Community Edition. Key points include:

1. Single Server is on the retirement path, with Azure recommending upgrading to the Flexible Server deployment model for new developments and migrations.
2. The Single Server architecture is optimized for built-in high availability with 99.99% availability in a single availability zone. It separates compute and storage for elasticity at a reduced cost.
3. Automated patching handles security and software updates, including minor MySQL engine version upgrades. Users can subscribe to planned maintenance notifications.
4. Automatic backups are created and stored in user-configured redundant storage, with retention up to 35 days.
5. Performance can be adjusted within seconds across three SKU tiers: Basic, General Purpose, and Memory Optimized. Storage scaling is online and supports autogrowth.
6. Enterprise-grade security features include encryption at-rest and in-transit, private link access, threat protection, Microsoft Entra ID authentication, and audit logging.
7. Built-in performance monitoring and alerting is provided, including slow query logs and Query Store for performance troubleshooting.
8. Migration options include offline methods like dump and restore, Azure Database Migration Service, and minimal downtime approach using data-in replication.
9. The service is fully compatible with MySQL Community Edition, allowing easy migration of existing applications with minimal refactoring.

In summary, while Azure recommends the Flexible Server deployment model for new developments, the Single Server model provides a fully managed MySQL database

service with built-in high availability, automated management, comprehensive security, and easy migration options for existing MySQL applications.