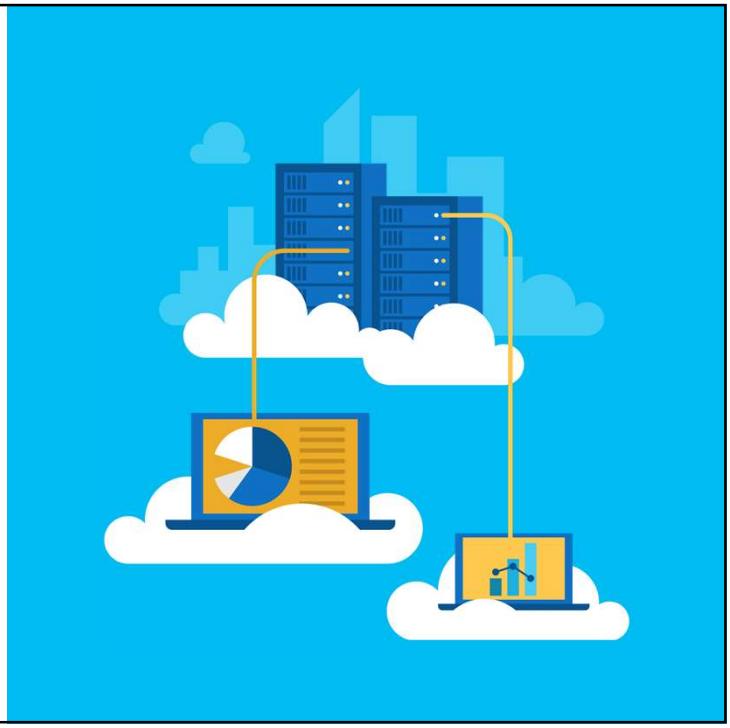
Microsoft

AZ-300T03 Module 03: Measure Throughput and Structure of Data Access

Ahmad Majeed Zahoory



1

Module 03: Measure Throughput and Structure of Data Access

Lesson 01: Address Durability of Data and Caching

2

ACID (Atomic, Consistent, Isolated, Durable)

Predictable transactions must possess these basic properties:

Atomic
- A transaction must execute exactly once and must be atomic—either all of the work is done or none of it is. Operations within a transaction usually share a common intent and are interdependent. By performing only a subset of these operations, the system could compromise the overall intent of the transaction. Atomicity eliminates the chance of processing only a subset of operations

Consistent
- A transaction must preserve the consistency of data, transforming one consistent state of data into another consistent state of data. Much of the responsibility for maintaining consistency falls to the application developer

Isolated
- A transaction must be a unit of isolation, which means that concurrent transactions should behave as if each were the only transaction running in the system. Because a high degree of isolation can limit the number of concurrent transactions, some applications reduce the isolation level in exchange for better throughput

Durable
- A transaction must be recoverable and therefore must have durability. If a transaction commits, the system guarantees that its updates can persist even if the computer crashes immediately after the commit. Specialized logging allows the system's restart procedure to complete unfinished operations required by the transaction, making the transaction durable

3

Caching in distributed applications

Common technique that aims to improve the performance and scalability of a system

Temporarily copies frequently accessed data to fast storage that's located close to the application
- If the fast storage is closer to the application than the original source, then caching can significantly improve response times for client applications by serving data more quickly

Most effective when a client instance repeatedly reads the same data, especially if all the following conditions apply:
- It remains relatively static
- It's slow compared to the speed of the cache
- It's subject to a high level of contention
- It's far away when network latency can cause access to be slow

4

Caching considerations

When to cache data

- Caching is exponentially most effective when you have a large amount of data and users attempting to access the data
- Caching can alleviate common database congestion issues
  - Limited number of concurrent connections
  - Network issues
  - Allocated compute and memory
  - Allocated storage

How to cache data effectively

- Determine early the most appropriate data to cache
  - Cache most access data that infrequently changes
  - If you cache constantly changing data, your cache effectively becomes the database
- Determine the appropriate time to cache
  - Cache when data is first accessed
  - Pre-load cache with data as part of application startup

5

Redis Cache

Redis

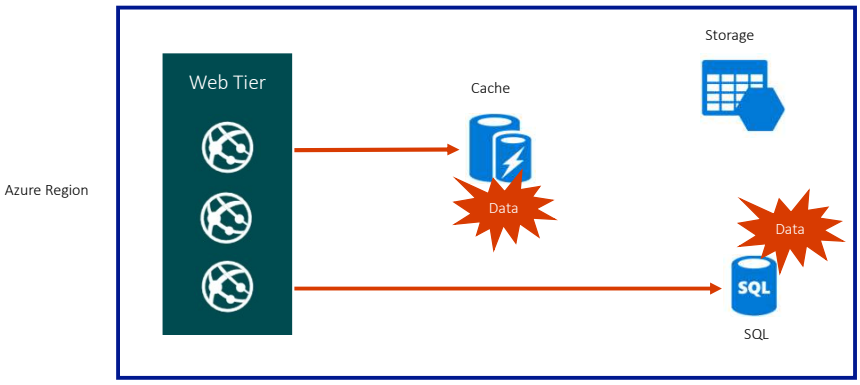
- https://redis.io
- Open-source NoSQL storage mechanism that is implemented in the key-value pair pattern common among other NoSQL stores
- Most commonly used as a cache mechanism

Azure Redis Cache

- Managed service based on Redis that provides secure nodes as a service
- Two service tiers:
  - Basic
    - Single node
  - Standard
    - Two nodes in the Primary/Replica configuration. Replication support and Service Level Agreement (SLA) is included
- High degree of compatibility with existing tools and applications that already integrate with Redis

6

Azure Redis Cache



7

Module 03: Measure Throughput and Structure of Data Access

Lesson 02: Measure Throughput and Structure of Data Access

8

Normalized units

In the context of hyperscale cloud database services:

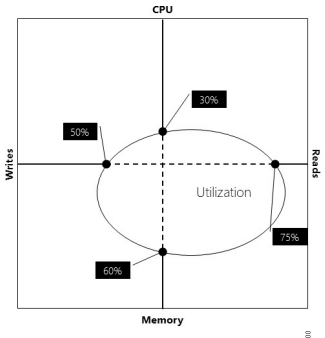
- Assist with comparing database tiers
- Sometimes represent direct relationship to on-premises equivalents
- Can serve as relative performance guarantees

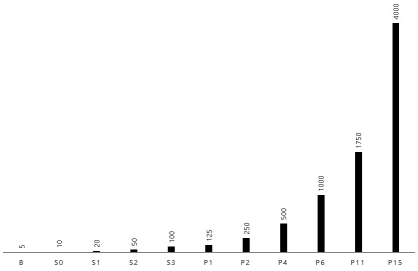
9

DTUs –Azure SQL Database

The Database Transaction Unit (DTU) model:

- **Represents a combination of:**
  - CPU
  - Memory
  - Read-write performance
- **Comprises of:**
  - DTUs – single databases
  - eDTUs – elastic database pools
- **Describes the capacity of:**
  - 3 service tiers:
    - Basic, Standard, Premium
  - Multiple performance levels:
    - within each service tiers





10

RUs – Azure Cosmos DB

The Request Unit Processing per second (RU/s) model:

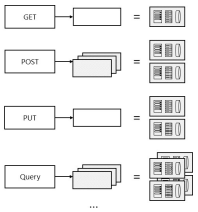
- Represents a combination of:
  - CPU
  - Memory
  - IOPS
- Provides normalized measure of request processing costs:
  - 1 UR is the processing capacity required to read

Normalized across various access methods

1 RU = 1 read of 1 KB document

Each request consumes fixed RUs

Applies to reads, writes, queries, and stored procedure execution



11

Using Structured Data Stores

Relational databases:

- Provide data organization:
  - Based on a well-defined schema
  - As a series of two-dimensional tables with rows and columns
  - Supporting Structured Query Language (SQL)
  - Transactionally-consistent and conforming to ACID model
- Are well suited for scenarios that require strong consistency
- Are not well suited for scenario that require scaling out
- Include Azure managed structured data stores:
  - Azure SQL Database
  - Azure Database for MySQL
  - Azure Database for PostgreSQL

12

Using Unstructured or semi-structured data stores

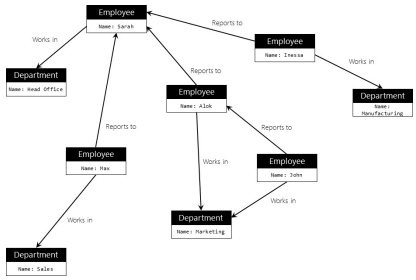
Non-relational databases utilize an optimized storage model based on specific requirements. For example, a non-relational database might store date as simple key/value pairs, as JSON documents, or as a graph consisting of edges and vertices.

DOCUMENT DATABASES

- A document database stores a collection of named fields and data (known as documents), each of which could be simple scalar items or compound elements such as lists and child collections.

GRAPH DATABASES

- The purpose of a graph database is to allow an application to efficiently perform queries that traverse the network of nodes and edges, and to analyze the relationships between entities.



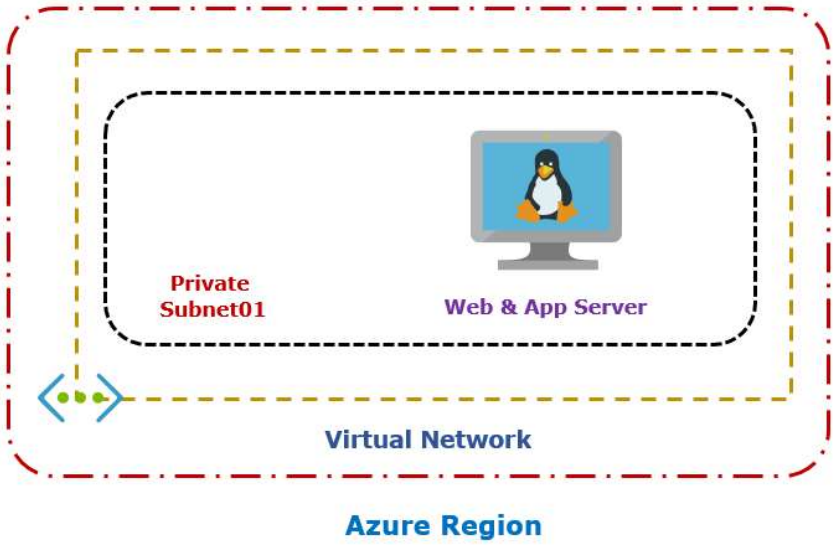
13

LAB [300TO03-M03-01]

1. Create 3 Tier Architecture.

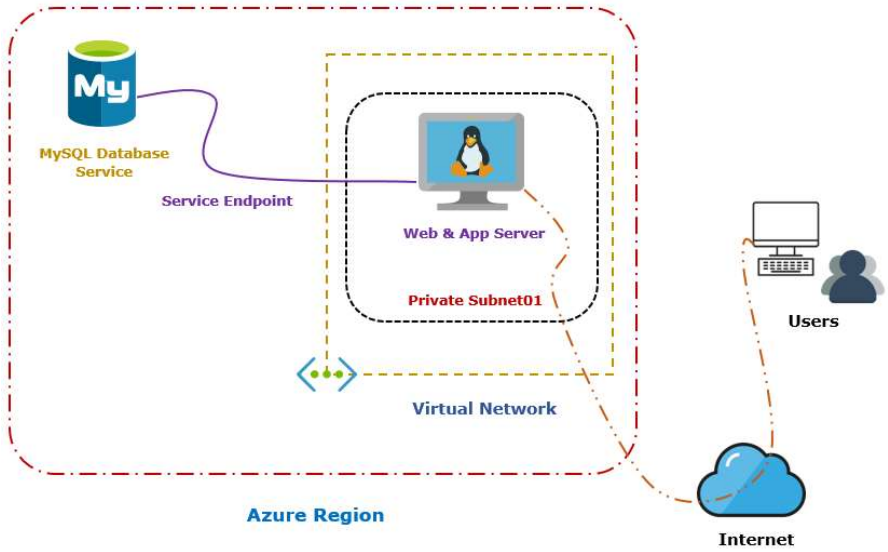
14

Part A



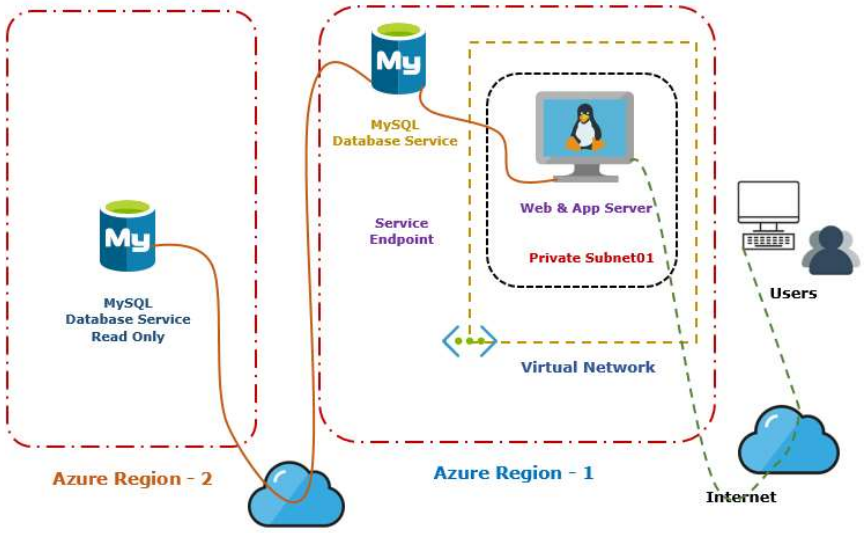
15

Part B



16

Part C



17

LAB [300TO03-M03-01]

1. Create 3 Tier Architecture.
   a. **Services, Tools & Code used**
      i. Azure Virtual Machine
      ii. HTML & PHP Code
      iii. Azure MySQL Database
      iv. Azure Service Endpoint

Duration: 30 mnts.

18


Microsoft
© Copyright Microsoft Corporation. All rights reserved.

19