PassQueen
a.k.a dumpsqueen.com

SOA Design & Architecture Lab with Services & Microservices

SOA S90.08B

Version Demo

Total Demo Questions: 3

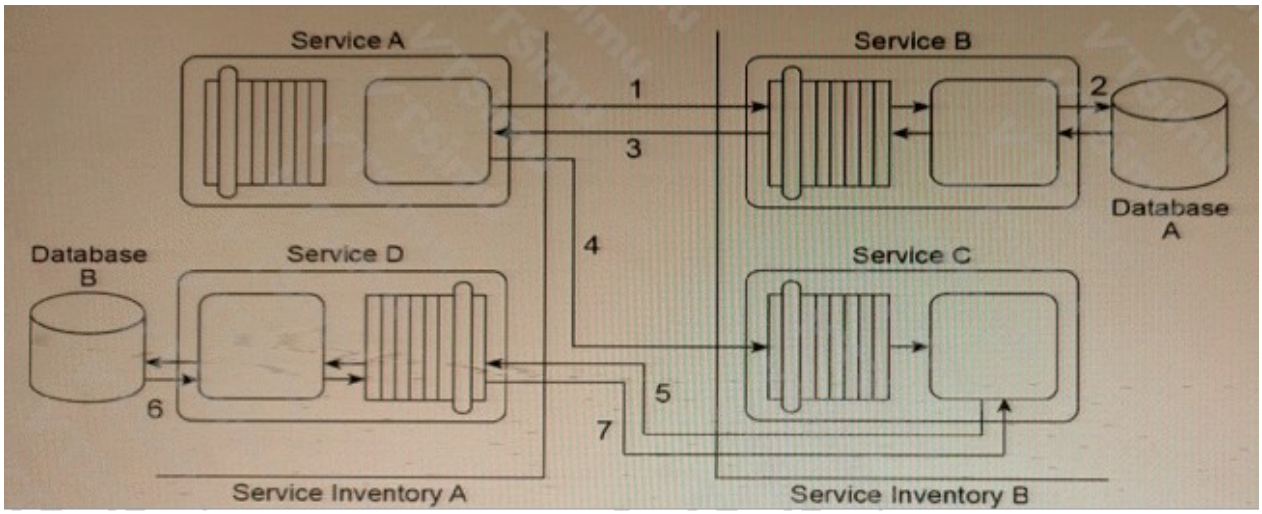
Total Premium Questions: 38

Buy Premium PDF

https://passqueen.com

support@passqueen.com

QUESTION NO: 1



Service A sends a message to Service B (1). After Service B writes the message contents to Database A (2), it issues a response message back to Service A (3). Service A then sends a message to Service C (4). Upon receiving this message, Service C sends a message to Service D (5), which then writes the message contents to Database B (6) and issues a response message back to Service C (7).

Service A and Service D are located in Service Inventory A. Service B and Service C are located in Service Inventory B.

You are told that In this service composition architecture, all four services are exchanging invoice-related data in an XML format. However, the services in Service Inventory A are standardized to use a different XML schema for invoice data than the services in Service Inventory B. Also, Database A can only accept data in the Comma Separated Value (CSV) format and therefore cannot accept XML-formatted data. Database B only accepts XML-formatted data. However, it is a legacy database that uses a proprietary XML schema to represent invoice data that is different from the XML schema used by services in Service Inventory A or Service Inventory B.

What steps can be taken to enable the planned data exchange between these four services?

A. The Data Model Transformation pattern can be applied so that data model transformation logic is positioned between Service A and Service B, between Service C and Service D, and between the Service D logic and Database B. The Data Format Transformation pattern can be applied so that data format transformation logic is positioned between Service A and Service C, and between the Service B logic and Database A.

B. The Protocol Bridging pattern can be applied so that protocol conversion logic is positioned between the Service B logic and Database A. The Data Format Transformation pattern can be applied so that data format transformation logic is positioned between Service A and Service B, between Service A and Service C, between Service C and Service D, and between the Service D logic and Database B.

C. The Data Model Transformation pattern can be applied so that data model transformation logic is positioned between Service A and Service B, between Service A and Service C, between Service C and Service D, and between the Service D logic and Database B. The Data Format Transformation pattern can be applied so that data format transformation logic is positioned between the Service B logic and Database A.

D. The Protocol Bridging pattern can be applied so that protocol conversion logic is positioned between Service A and Service B, between Service A and Service C, and between Service C and Service The Data Format Transformation pattern can be applied so that data format transformation logic is positioned between the Service B logic and Database A and between the Service D logic and Database B.

ANSWER: C

Explanation:

The Data Model Transformation pattern can be applied so that data model transformation logic is positioned between Service A and Service B, between Service A and Service C, between Service C and Service D, and between the Service D logic and Database B. The Data Format Transformation pattern can be applied so that data format transformation logic is

positioned between the Service B logic and Database A. This placement addresses each boundary at which the meaning or representation of the invoice must change. Messages exchanged across the two service inventories remain XML, but their invoice elements, structures, and schema-defined semantics differ. They therefore require data-model mappings at every cross-inventory interaction: the exchanges from Service A to Service B, from Service A to Service C, and from Service C to Service D. The proprietary XML vocabulary required by Database B is likewise an XML data-model difference, so Service D must map its inventory-standard invoice model to the database's legacy schema before persistence. XML Schema provides the formal vocabulary for defining XML document structures, as described by the W3C XML Schema specification. Database A presents a separate format-level concern: it accepts CSV rather than XML. A data-format transformation at the persistence boundary serializes the invoice data into CSV while preserving the required business content. CSV's conventional record-and-field representation is specified in RFC 4180.

QUESTION NO: 2

An organization is gradually decomposing a monolithic application. A new service is implemented for customer-address management, but all existing consumers must continue to use the established application endpoint during the migration.

Which migration approach best enables incremental replacement while preserving the existing consumer interface?

A. Strangler pattern

B. Shared library extraction

C. Database replication

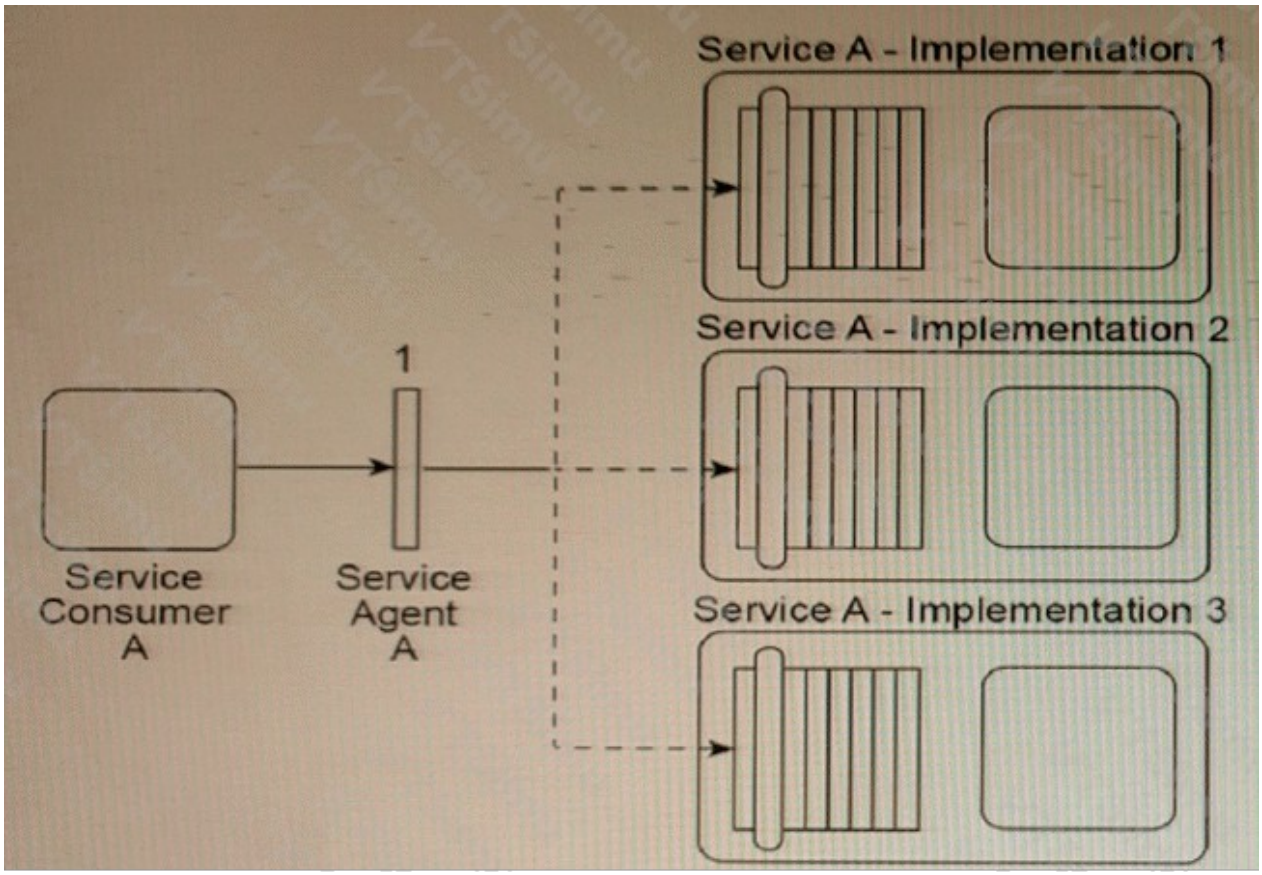
D. Two-phase commit

ANSWER: A

Explanation:

The strangler pattern places a routing layer in front of the existing application and incrementally directs selected functionality to replacement services. Consumers can continue using the established endpoint while capabilities are migrated over time.

QUESTION NO: 3



Service Consumer A sends a message to Service A. There are currently three duplicate implementations of Service A (Implementation 1, Implementation 2 and Implementation 3). The message sent by Service Consumer A is intercepted by Service Agent A (1), which determines at runtime which implementation of Service A to forward the message to. All three implementations of Service A reside on the same physical server.

You are told that despite the fact that duplicate implementations of Service A exist, performance is still poor at times. You are also informed that a new service capability will soon need to be added to Service A to introduce functionality that will require access to a shared database being used by many other clients and applications in the IT enterprise. This is expected to add further performance demands on Service A.

How can this service architecture be changed to improve performance in preparation for the addition of the new service capability?

A. The Standardized Service Contract principle can be applied to ensure that the new service capability extends the existing service contract in a manner that is compliant with current design standards. The Redundant Implementation pattern can be applied to establish separate implementations of Service A that include duplicate databases with copies of the data that Service A requires from the shared database.

B. The Service Autonomy principle can be applied to further isolate the individual implementations of Service A by separating them onto different physical servers. When the new service capability is added, the Service Data Replication pattern can be applied to give each implementation of Service A its own copy of the data it requires from the shared database.

C. The Service Loose Coupling principle can be applied together with the Standardized Service Contract principle to ensure that Service Consumer A is not indirectly coupled to the shared database after the new service capability is added to the service contract. The Legacy Wrapper pattern can be applied to establish a new utility service that will provide standardized data access service capabilities for the shared database.

D. The Service Autonomy principle can be applied to further isolate the individual implementations of Service A by separating them onto different physical servers. When the new service capability is added, the State Repository pattern can be applied to give each implementation of Service A its own copy of the data it requires from the shared database.

ANSWER: B

Explanation:

The Service Autonomy principle can be applied to further isolate the individual implementations of Service A by separating them onto different physical servers. When the new service capability is added, the Service Data Replication pattern can be applied to give each implementation of Service A its own copy of the data it requires from the shared database.

This approach addresses both identified sources of contention. Although duplicate service implementations already exist, colocating all of them on one physical server means they still compete for the same host-level resources, such as processor capacity, memory, network interfaces, storage access, and operating-system resources. Deploying the implementations separately creates stronger runtime isolation and allows the service agent to distribute requests among independently scalable service instances. This is a practical application of service autonomy because each implementation is less dependent on the availability and workload of a shared hosting environment.

Service Data Replication further protects performance when the new capability requires frequently accessed enterprise data. Each implementation can access a local replicated data store rather than repeatedly competing with other enterprise clients for the shared database. This reduces central database load and request latency while enabling each service implementation to scale with its own data-access workload. Replication requires an appropriate synchronization and consistency strategy, but it is well suited when read performance and reduced dependency on a shared data source are primary concerns. See Arcitura SOA Patterns and Arcitura SOA Principles.