Microsoft

Implementing a SQL Data Warehouse

Exam Ref 70-767

Jose Chinchilla
Raj Uchhana

Microsoft

Exam Ref 70-767 Implementing a SQL Data Warehouse

Jose Chinchilla
Raj Uchhana

lationship constraints to enforce referential integrity. A partition strategy that involves splitting dimension and fact tables on different databases does not work in this case because referential integrity enforcement across databases is not supported. A better partition approach is to group related tables together that only relate to each other and don't have relationships to the rest of the data model. This group of related tables that are isolated from the rest are often referred to as data marts.

There are many options to scale-out the data, but the six more frequently implemented scale-out solutions involve the following mechanisms:

1. Allowing multiple database engines to access a single copy of the database.
2. Replicating the database to multiple database servers.
3. Implementing linked servers and executing distributed queries.
4. Implementing distributed partitioned views.
5. Routing queries to the correct database using some middleware service.
6. Implementing a stretch database with Azure.

One of the easiest scale-out solutions is to share a single copy of a database on a SAN storage device. This scale-out solution is often referred to as a shared disk scale-out solution, where processing power is scaled-out to many servers using a single disk image of the data. In this case, the database is attached as a read-only database to each of the database servers. This scale-out solution is only suitable for scenarios where read-only operations are required, such as reporting solutions. This scale-out solution does not work in scenarios where data is constantly updated and reports need to reflect the most recent data loaded in the database.

A shared database scale-out solution works great for data warehouses that are only updated after business hours or during scheduled maintenance windows. In this case, the database is placed in read-write mode in one of the servers and data is loaded. After the data loads complete, the database is placed in read-only mode again. If reporting is expected to continue after business hours or during the scheduled maintenance window, a separate copy of the database is used for data loads and the read-only database image is replaced with the newly loaded database image. Figure 1-38 shows an illustration of a shared database scale-out solution.

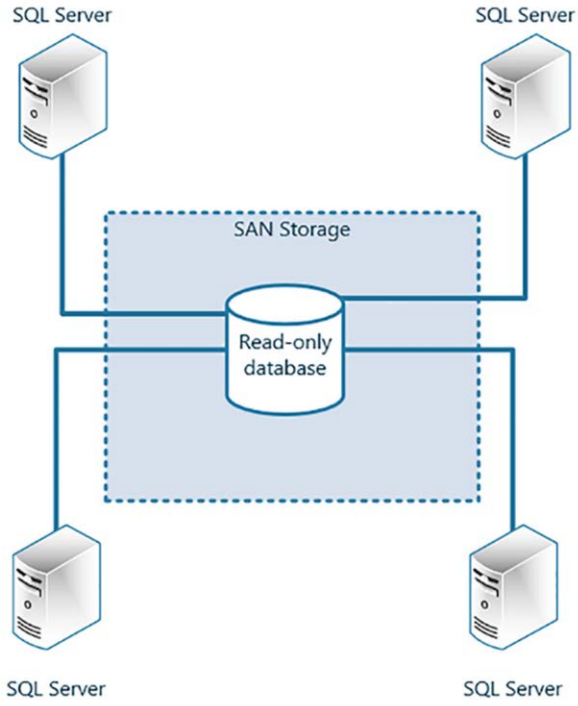

FIGURE 1-38 Illustration of a shared database scale-out solution

A second approach to scale-out is to replicate the data to multiple servers, each containing its own copy of the database. This scale-out solution has additional storage implications because multiple copies of the database are needed. Data updates are replicated by using peer-to-peer transactional replication. The main advantage of this scale-out solution is the ability to replicate data updated in any SQL Server to all other copies of the database. A peer-to-peer replication scale-out solution works great for environments where data is updated with moderate frequency and where no data update conflicts are expected. If data update conflicts are expected, merge replication might be more appropriate. The drawback of merge replication is that it imposes more overhead than peer-to-peer replication. Figure 1-38 shows an illustration of peer-to-peer replication.

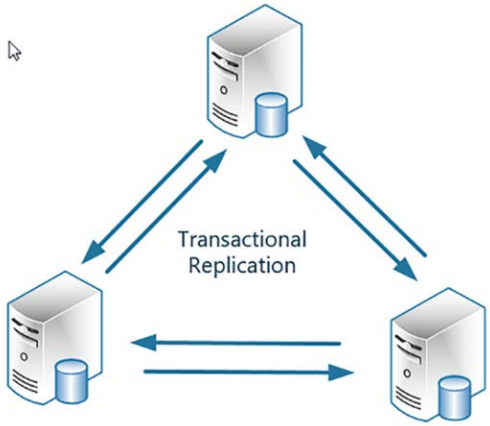

FIGURE 1-39 Illustration of peer-to-peer replication scale-out solution

A third scale-out solution involves the configuration of linked servers. Databases from linked-servers can be queried as if they were local databases. A scaled-out database across several linked servers can be logically accessed as a single large database. This scale-out solution requires servers to be configured as linked servers and for queries to be changed slightly by including a four-part name that includes the server name. For example, a query against an HR database located on a linked server would look as follows:

```
SELECT * FROM HRServer.HRDB.dbo.Employees
```

A linked server scale-out approach provides the ability to segregate databases by subject area, but still provide the ability to report across all subject areas if required. For example, a dedicated database for an Inventory data warehouse and a dedicated database for a Sales data warehouse might exist on separate SQL servers. By linking these servers, a query can bring data from both databases together into a consolidated Inventory and Sales report. Figure 1-40 shows a logical representation of linked servers.

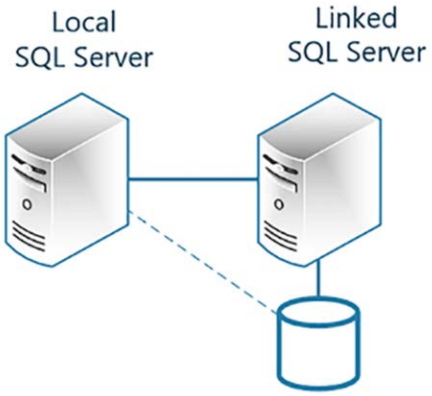

FIGURE 1-40 Illustration of linked servers

A fourth approach is to implement distributed partitioned views. Just like partitioned views covered previously, data is partitioned among tables, but in this case, the tables are distributed across several databases. The check constraints in each table tell SQL Server which table, and therefore which database, to get the data from. Distributed partitioned views rely on queries using the partitioning key as part of the filter criteria. Queries that don't use the partitioning key result in each table from all databases being read.

A distributed partition view scale-out approach works well in data warehouses where historical data is archived in one or more historical databases. For example, some organizations might choose to archive data in yearly databases. Queries can take advantage of partition elimination by using this approach if they commonly filter data for particular years as part of the `WHERE` clause predicate.

A fifth scale-out solution involves the use of a middleware service to route queries to the correct database. In this approach, a software application acts as the middleman between the user or the reporting application and the database. This type of scale-out solution is referred to as data-dependent routing. Queries are then routed by the middleware to the appropriate database containing the data. This scale-out approach requires a more complex management of the data.

For example, a data warehouse database can be broken down into separate databases that contain all information about customer sales by region. When a query is executed for sales data about a customer that belongs to a country, the middleware service routes the query to the corresponding database. Figure 1-41 shows a data dependent routing scale-out solution.

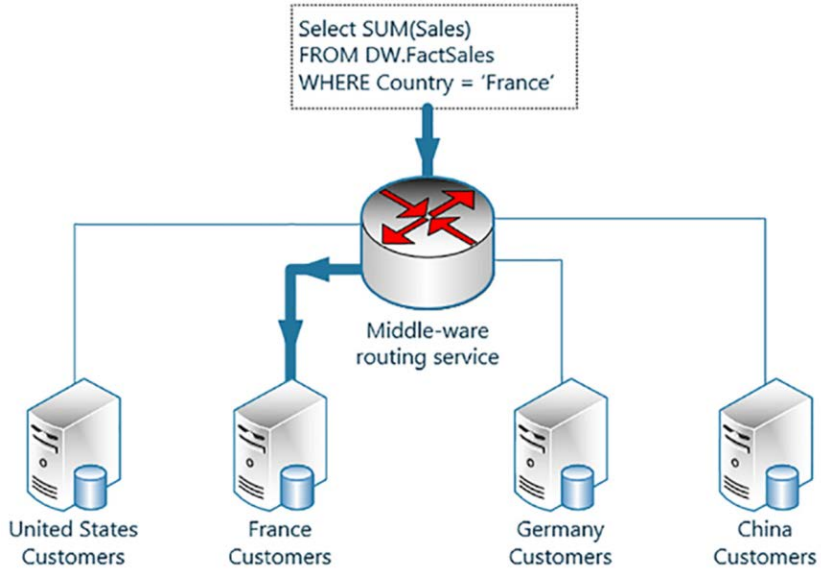

FIGURE 1-41 Illustration of data-dependent routing scale-out solution

Notice in Figure 1-41 that the query that sums the sales for customers in France is executed by routing it to the France Customers database.

The sixth scale-out solution commonly implemented is through

. SQL Server stretch databases allow you to dynamically place transactional data into an Azure database. In this approach, you end up with a hybrid. This scale-out solution allows you to store cold or warm data in the Azure cloud while maintaining data that is accessed more frequently in on-premise servers. Stretch databases allow you to move data out of large tables based on a simple filter function specified on one or more tables. After a table is defined as a stretch table and a filter function is defined, you can gradually start to migrate data to the Azure database.

A stretch database is seamless and transparent to the user. Queries are executed normally against the local table. SQL Server can eliminate reading the rows stored in Azure if they are not required in the query results. The filter function specified in the table acts, in this case, as a partition key. Figure 1-42 shows an illustration of a stretch database.

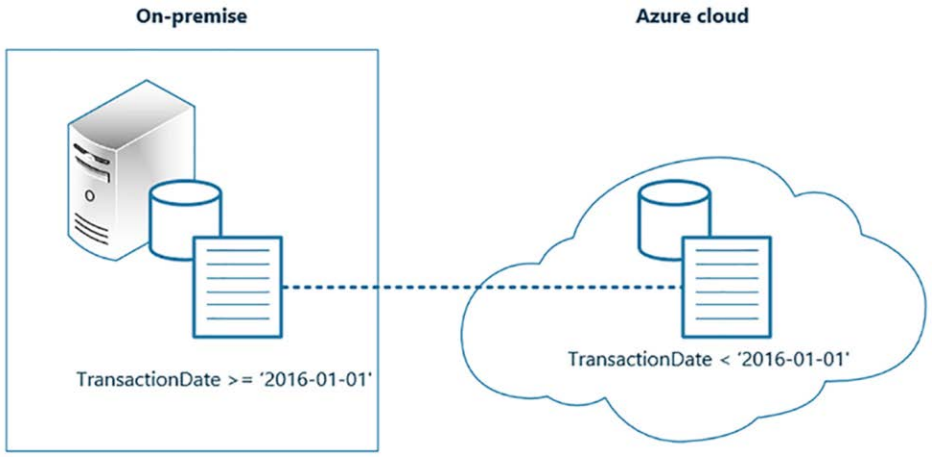

FIGURE 1-42 Illustration of a stretch database

Notice that in Figure 1-42 a table has been defined as a stretch table in the Azure cloud. The rows that are physically stored on-premise include data with a transaction date greater than or equal to '2016-01-01'. Data with a transaction date older than this is moved to the Azure cloud.

Thought experiment

In this thought exercise, demonstrate your skills and knowledge of the topics covered in this chapter. You can find the answer to this thought experiment in the next section.

You are tasked to create a logical star schema diagram that models the Wide World Importers purchasing business process. The business is interested in creating an analysis to compare the quantity of stock items ordered vs. the quantity of stock items received for all their purchase orders and corresponding suppliers over time.

The Wide World Importers sample database contains a dedicated schema for the Purchasing related tables. An entity relationship diagram of the source tables needed for your requirements is shown in Figure 1-43.

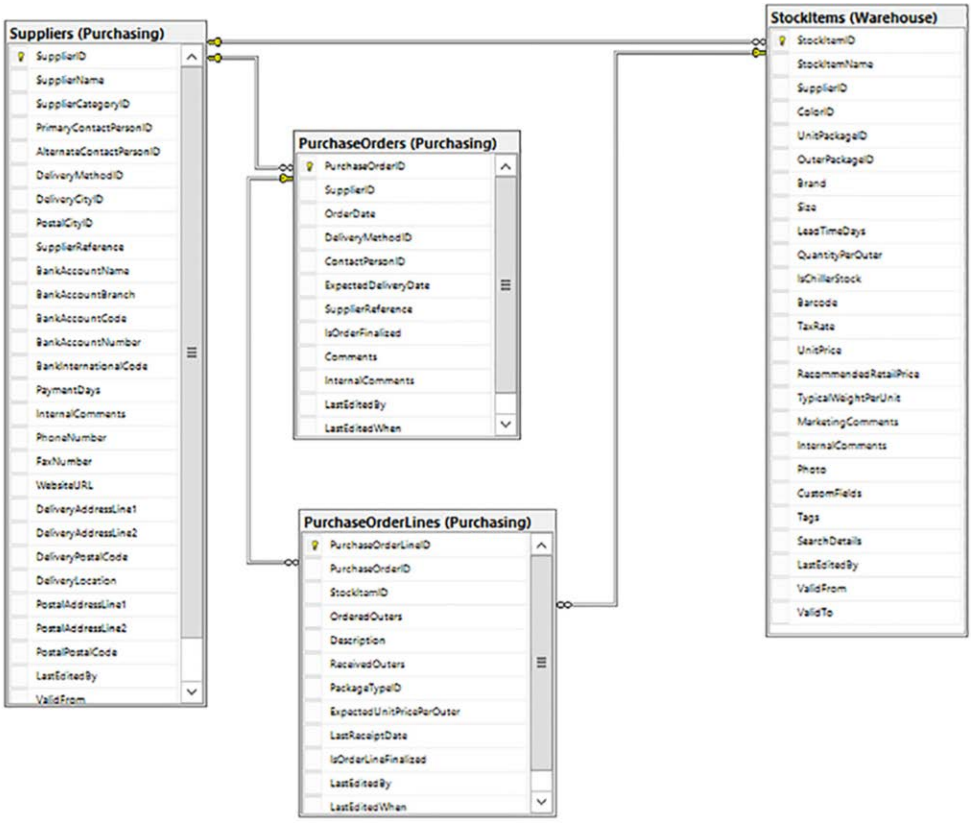

FIGURE 1-43 Entity relationship diagram for the Purchasing related tables

Notice that there is a PurchaseOrders parent table and a PurchaseOrderLines child table. The PurchaseOrderLines child table tracks the individual stock items that were ordered in the purchase order. The `[OrderedOuters]` column tracks the stock item quantity that was ordered.