

Azure SQL Database – Point-in-Time Restore

Level-300 Demonstration

Script

Version 1.0

9/3/2015

Prepared for: Darmadi Komo

Prepared by: Prowess Consulting

Overview

This demonstration explores point-in-time database restoration in Microsoft Azure SQL Database in the context of a software-as-a-service (SaaS) provider, Wingtip Tickets, which provides ticketing software to artists and groups. This demonstration centers on the tenant Julie and the Plantes (a fictitious pop-music tenant.

Other Tenants that will be discussed in future labs, will include the following:

* The Archie Boyle Band (a fictitious rock-music tenant)
* Walla Walla Symphony (a fictitious classical-music tenant)

## Demo Architecture

## 

**Figure 1** Overall architecture of demo components

## Dependencies

This demonstration requires running the deployment and configuration PowerShell scripts from the Level-200 demonstration in order to work. Because this demonstration also involves restoring a deleted record from the database, you will need to first purchase a ticket (Section 0 of the level-200 Azure SQL Database demo) before you can present this demonstration.

|  |  |  |
| --- | --- | --- |
|  | Section 0: Prerequisite for Point-in-time Restore – Deleting an Event |  |
|  | 1. Switch back to the Tenant website. 2. Click the **My Events** tab. 3. Notice the one event that you previously purchased. 4. Click **Sign Off**. |  |
|  | * 1. Click **Sign In**. |  |
|  | 1. Sign in using the administrator credentials:  * **admin@admin.com** * **P@ssword1** |  |
|  | 1. Click the **Admin** tab. |  |
|  | 1. In the **City**, **Venue**, **Event**, and **Artist** drop-down menus, select the values that match those for the event to which you bought tickets in Task 1. 2. Verify that this event is the same event for which you purchased tickets in the Task 1. 3. Click **Delete Event**. |  |
|  | 1. At this point you should receive a notification that the event has been deleted. Make a note of the event ID, in this example #1. Click **OK**. |  |
|  | 1. Click **Sign Off**. 2. Click **Sign In**. |  |
|  | 1. Sign in using the account you previously created. |  |
|  | 1. Click the **My Events** tab.   Notice that the event that you previously purchased is now missing. |  |

|  |  |  |
| --- | --- | --- |
|  | **Section 1: Point-in-Time Restore** |  |
|  | In this demonstration, an event was inadvertently deleted from the database. To resolve this issue, you will use point-in-time restore to restore the related database Customer1.  In addition to seeing how easy it is to restore a database to a specific point in time, from the Tenant website you will also see the deleted concert event is restored, along with previously purchased tickets. | |
|  | 1. If you did *not* complete the audit section, complete steps 1–15 of Task 5 before proceeding.   **Important:** Unless you delete a concert event in Task 5, steps 1–15, you will have nothing to restore in this section. |  |
|  | 1. Switch to the Azure portal. 2. Browse to the Customer1 SQL database. 3. Click the **Customer1** database on the *Primary Database* *Server* (in this example *mbjulieandtheplantesprimary.database.windows.net*). | In order to restore the event, we need to go to the Customer1 database. |
|  | 1. Click **Restore**. 2. Change the restore point time to two minutes before the delete event occurred according to the audit log or if you didn’t complete the Auditing section, roughly two minutes before you deleted the event at the start of this section. 3. Click **Create**.   The Customer1 database is now being restored.  Once restored, the Customer1 database status changes to Online. | We are going to restore the database to two minutes before the deletion took place. (We don’t want to go back too far before the deletion in case we accidentally undo some legitimate changes to the database in the process.) |
|  | 1. Switch to SQL Server Management Studio (SSMS). 2. Click the ***Primary Database Server*** (in this example *mbjulieandtheplantesprimary.database.windows.net*). 3. Click **Refresh**. 4. Notice the newly restored Customer1 database. | Let’s verify that the restoration was successful. We’ll switch to SQL Server Management Studio.  We can see the newly restored Customer1 database on the database server. |
|  | 1. Within the *Primary Database Server* database server (in this example, *mbjulieandtheplantesprimary.database.windows.net*), expand **Databases**, then expand the newly restored **Customer1\_[date] Database**, and then expand **Tables**. 2. Right-click on the **dbo.Concerts** table, and then select **Script Table As** > **Select To** > **New Query Editor Window**. 3. Execute the script. | And drilling into the Customer1 database … |
|  | Notice the previously deleted event, in this example ConcertID 1, is now restored. | … we see that ConcertID 1 is there. |
|  | 1. Right-click on the **dbo.Tickets** table, and then select **Script Table As** > **Select To** > **New Query Editor Window**. | Let’s also run another SQL query… |
|  | 1. Modify the T-SQL to filter only Tickets for the event that you deleted. In the **SQLQuery** field, delete the auto-generated SQL query and type the following, replacing the 1 with the correct ConcertID from your environment:  select \* from [dbo].[Tickets] where ConcertID = 1 2. Notice the ticket we had previously purchased. | …this one to search specifically for ConcertID 1. We have also restored the tickets previously deleted.  Now that we have successfully restored the Customer1 database, we now need to replace the current live version of the database. |
|  | 1. Switch back to the Azure portal. 2. Browse to the Customer1 database. 3. Click the **Customer1** database on the *Primary Database Server* (in this example *mbjulieandtheplantesprimary.database.windows.net*). 4. To the right of the row, click **…** (the three-dot ellipses). 5. Click **Delete**. | We will delete the version of the Customer1 database that has the deletion in it… |
|  | 1. Verify the Customer1 database has now been deleted. | …and then verify that the old version of the database has been deleted. |
|  | 1. Switch back to SQL Server Management Studio. 2. Click ***Primary Database Server*** (in this example *mbjulieandtheplantesprimary.database.windows.net*). 3. Click **Refresh**. 4. Notice that the Customer1 database has been deleted. | Let’s switch to SQL Server Management Studio to verify this.  And we see that the Customer1 database has been deleted. |
|  | 1. Right-click on **Customer1\_*[date]***. 2. Click **Rename**. 3. Delete everything after Customer1. 4. Press Enter. | Now we need to change the name of the restored Customer1 database to make it our production Customer1 database. |
|  | 1. Verify Customer1\_*[date]* has been renamed to Customer1. | Now that it is renamed, we are good to go. |
|  | 1. Switch back to the Traffic Manager website. 2. Refresh Traffic Manager Website. | Let’s go back to the Traffic Manager website to verify the restoration. |
|  | 1. If you’re not still signed in, sign in using the credentials you previously created. |  |
|  | 1. Click **My Events**. 2. Verify the deleted ticket has now been restored. | And there are the tickets that had been deleted. |

|  |  |  |
| --- | --- | --- |
|  | **Section 2: Conclusion** |  |
|  | Point-in-time restore is great for localized, one-off mistakes, but it is not well suited for broader server or datacenter failures? To see the Azure tool designed for these larger types of failures, ask your Microsoft sales representative about seeing the Azure SQL Database geographic disaster recovery demonstration to see how Azure can help protect vital database workloads from regional disasters and outages. | |