# Solutions

The coupon management system storage can be implemented as a sql storage using Azure Sql or as a document-orientated storage using CosmosDb.

If it was a real project, I would implement prototypes for both versions of storage and tested performance and performance degradation as data grows.

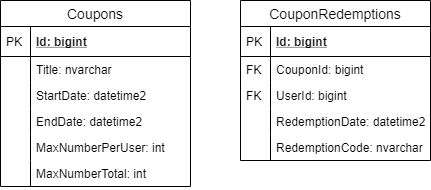
Also, when I make a decision on which storage solution to use, I would take into consideration overall system architecture, geographic distribution requirements, and storage cost.

Below are descriptions of both solutions.

## SQL Solution

### Tables

The coupon management system storage has two tables: Coupons and CouponRedemptions. Coupons table stores coupons and their attributes. CouponRedemptions table stores redemption events.



### Optimisation

Index for Coupons table

CREATE NONCLUSTERED INDEX [IX\_Dates] ON [dbo].[Coupons] ([StartDate], [EndDate])

IX\_Dates index helps to retrieve list of active coupons

Indexes for CouponRedemption table

CREATE NONCLUSTERED INDEX [IX\_CouponId] ON [dbo].[CouponRedemptions] ([CouponId])

CREATE NONCLUSTERED INDEX [IX\_UserId] ON [dbo].[CouponRedemptions] ([UserId])

CREATE NONCLUSTERED INDEX [IX\_CouponId\_UserId] ON [dbo].[CouponRedemptions] ([CouponId], [UserId])

* IX\_CouponId index helps to get redemptions for a specific offer
* IX\_UserId index helps to get redemptions for a user. Not required but it is a good idea to have indexes on all foreign keys.
* IX\_CouponId\_UserId helps to determine if a consumer can redeem a coupon

Probably, after performance tests, IX\_UserId and IX\_CouponId\_UserId might be replaced with just one index

CREATE NONCLUSTERED INDEX [IX\_UserId2] ON [dbo].[CouponRedemptions] ([UserId]) INCLUDE([CouponId])

IX\_UserId2 might replace both indexes. It reduces database size and increases writing performance.

## CosmosDb solution

The coupon management system storage has containers: Coupons and CouponRedemptions. Coupons container stores coupons and their attributes. CouponRedemptions container stores redemption events.

#### Coupons container

Coupon container uses CouponId as the partition key.

Example of data in Coupons container  
{

   "couponId":1,  
   "title":"Coupon Title",  
   "startDate":"2020-06-01",  
   "endDate":"2020-07-01",  
   "maxNumberPerUser":2,  
   "maxNumberTotal":100000

}

#### CouponRedemptions container

CouponRedemptions container uses CouponId as the partition key.

Example of data in CouponRedemptions container

{

   "redemptionId":1,  
   "couponId":12345,  
   "userId":54321,  
   "redemptionDate":"2020-06-01T07:22Z",  
   "redemptionCode":"qwerty12345"

}

# Exercise 2:

A coupon management system stores a definition of a coupon and also every time a user redeems a coupon. A coupon consists of an id, a title, a start date, end date, maximum number of coupons per user and also a maximum number of coupons across all users. When a user redeems a coupon the system keeps track of the users identifier, the datetime when the redemption occurred and a unique code that is generated by the application.

The coupon management system is required to provide the following capabilities:

* Provide an active list of coupons.
* Determine if a consumer can redeem a coupons.
* Store redemptions as they occur.
* Provide reporting on the redemptions for a specific offer.

The coupon management system is expected to have the following capacity:

* Coupons up to 50000 unique coupons. Generally 100-500 active at any point in time.
* Redemptions – Upwards of 1 billion rows. Expect at least 1 million redemptions per day.

Requirements:

* Create a data model to represent the data in the above scenario. You are free to choose any storage mechanism you feel appropriate. The model should take into account the volume and expected operations into account.
* Define additional optimizations that you would apply to the basic data model. (For example if you were to choose SQL you might want indexes on certain columns) Note: Implementation/Code is not required.