**AMAZON RDS VS DYNAMO DB**

**1.DATABASE**

**AMAZON RDS SQL, DYNAODB NoSQL veri tipi kullanır**

SQL veritabanları, verileri depolamak ve almak için yapılandırılmış sorgulama dili kullanan ilişkisel veritabanlarıdır. NoSQL veritabanları, verileri depolamak için çeşitli yöntemler kullanan ilişkisel olmayan veritabanlarıdır.

SQL veritabanları, verileri tablo biçiminde depolayan ilişkisel veritabanlarıdır. NoSQL veritabanları, verileri JSON, XML ve Binaries dahil olmak üzere çeşitli biçimlerde depolayan ilişkisel olmayan veritabanlarıdır.

SQL veritabanları, verileri satırlar ve sütunlar içeren tablolar halinde düzenleyen ilişkisel modeli temel alır. NoSQL veritabanları, verileri belgelerde anahtarlar ve değerlerle depolayan ilişkisel olmayan veya "NoSQL" modelini temel alır.

**2. FEATURES**

**Amazon RDS Özellikleri**

* Amazon RDS, Amazon Aurora, MySQL, MariaDB, Oracle, Microsoft SQL Server ve PostgreSQL dahil birden çok veritabanı motorunu destekler.
* Amazon RDS, veritabanı bulut sunucularınızın depolama boyutunu ve performansını ölçeklendirmenize olanak tanır.
* Amazon RDS, kurulumu kolaylaştırır, bulutta bir ilişkisel veritabanını çalıştırın ve ölçeklendirin.
* Amazon RDS, bulutta ilişkisel veritabanlarını yönetmek için uygun maliyetli bir yol sağlar.

**DynamoDB Özellikleri**

* DynamoDB, öncelikle esneklik, ölçeklenebilirlik ve performans sunar.
* Kurulum veya yapılandırma gerektirmeden kutudan çıkar çıkmaz yüksek kullanılabilirlik sunar.
* DynamoDB, size hata toleransı ve yüksek kullanılabilirlik sağlamak için verilerinizi bir bölge içindeki birden çok AZ otomatik olarak çoğaltır.

**3. STORAGE**

* Amazon RDS, MySQL, MariaDB, Oracle ve PostgreSQL motorları için 128 TB'a kadar ve 64 TB'a kadar depolama alanıyla Amazon Aurora motorunu kullanabilir. Öte yandan, SQL Server motoru 16 TB depolama gerektirir.
* DynamoDB Neredeyse her boyuttaki tabloları destekler. Genel Amaçlı SSD DB bulut sunucuları için depolama boyutu aralıkları aşağıdaki gibidir:

MariaDB, MySQL, Oracle ve PostgreSQL için veritabanı bulut sunucuları: 20 GiB-64 TiB

20 GiB-16 TiB SQL Server Enterprise, Standard, Web ve Ekspres Editions

**4. PERFORMANCES**

İki hizmet arasındaki temel fark,

Amazon RDS'nin ilişkisel veritabanlarıyla kullanılmak üzere tasarlanmış olmasıdır.

Bunun aksine, DynamoDB ilişkisel olmayan veritabanlarıyla kullanım için tasarlanmıştır.

RDS, DynamoDB'den daha pahalıdır ancak daha fazla özellik ve esneklik sunar.

Amazon RDS, uygulamanızın ihtiyaçlarını karşılamak için otomatik olarak ölçeklenir. RDS, uygulamanızın yükü arttığında performansı korumak için otomatik olarak daha fazla kaynak ekler. Yük azaldığında, RDS kaynakları buna göre kaldıracaktır. Bu, altyapıyı yönetmek yerine uygulamanızı geliştirmeye odaklanmanızı sağlar.

DynamoDB, uygulamalarınızın taleplerini karşılamak için üretim kapasitesini otomatik olarak ölçeklendirir. İstek sayısı arttıkça, DynamoDB tablonuza tahsis edilen kapasite birimlerinin sayısını artırır. Bu, verilerinizin boyutu büyüdükçe DynamoDB'nin tutarlı performansı sürdürmesini sağlar.

**5. AVAILABILITY AND DURABILITY**

**Amazon RDS**, MySQL, MariaDB, PostgreSQL, Microsoft SQL Server, Oracle ve Amazon Aurora dahil olmak üzere çeşitli veritabanı motorlarını destekleyen bulut tabanlı bir ilişkisel veritabanı hizmetidir. Amazon RDS, bulutta bir ilişkisel veritabanı kurmayı, çalıştırmayı ve ölçeklendirmeyi kolaylaştırır. Amazon Elastic Compute Cloud'da (Amazon EC2) bulunan Amazon RDS, bulutta veritabanı yönetimi için eksiksiz bir çözüm sağlayan Amazon Simple Storage Service (Amazon S3), Amazon Virtual Private Cloud (Amazon VPC) ve Amazon CloudWatch ile entegre edilmiştir.

**DynamoDB**, oldukça dayanıklı ve kullanılabilir bir veritabanıdır. Tüm veriler, yerleşik yedeklilik ve yüksek kullanılabilirlik sağlamak için AWS Regions birden çok AZ otomatik olarak çoğaltılır. Veriler, bir altyapı kesintisi veya arızası durumunda yüksek kullanılabilirliği ve dayanıklılığı korumak için otomatik olarak birden çok AZ çoğaltılır. DynamoDB, size düşük gecikmeli bir okuma ve yazma deneyimi sunmak için tüm AZ eşzamanlı çoğaltma kullanır.

**6. SCABILITY**

**Amazon RDS**, bulutta ilişkisel bir veritabanı kurmayı, çalıştırmayı ve ölçeklendirmeyi kolaylaştırmak için tasarlanmış bulut tabanlı bir ilişkisel veritabanı hizmetidir. Amazon RDS, veritabanı iş yüklerinizin büyümesini yönetmenize yardımcı olmak için iki ölçeklenebilirlik seçeneği sunar: dikey ölçeklendirme ve replikaları okur. Ölçek büyütme olarak da adlandırılan dikey ölçeklendirme, Amazon RDS DB bulut sunucunuzun bilgi işlem ve depolama kapasitesini artırmak anlamına gelir. Ölçeklendirme yaptığınızda, veritabanı bulut sunucunuzun CPU, bellek ve depolama kaynaklarını iyileştirebilirsiniz. Öte yandan, okuma replikaları, veritabanı bulut sunucunuzun birincil veritabanı bulut sunucunuzdan okuma trafiğini boşaltmak için kullanılabilen kopyalarıdır.

**DynamoDB**, yatay olarak ölçeklenecek şekilde tasarlanmıştır. Bu, kapasiteyi ve iş hacmini artırmak için sisteme ek sunucuların eklenebileceği anlamına gelir. DynamoDB dikey olarak da ölçeklenebilir, bu da tek bir sunucunun yeteneğinin geliştirilebileceği anlamına gelir.

**7. GÜVENLİK**

**Amazon RDS,** security groupları biz belirleriz.

**DynamoDB**, default olarak belirler

**8. ENCRYPION**

**Amazon RDS,** Veritabanlarınızı şifrelemek için AWS KMS ile kontrol ettiğiniz anahtarları kullanın. Şifreleme etkinleştirildiğinde, beklemede depolanan verilerin tüm otomatik yedeklemeleri, okuma kopyaları ve anlık görüntüleri de şifrelenir. Oracle ve SQL Server'da şeffaf veri şifrelemeyi etkinleştirir. Verilerin SSL aracılığıyla güvenli iletimini sağlar.

**Dynamo DB**, default olarak AWS KMS'de saklanan şifreleme anahtarlarını kullanarak aktarılan verileri şifreler.

**9. BACKUPS**

**Amazon RDS,** Veritabanı eşgörünümünüz için belirli bir noktada kurtarma, otomatikleştirilmiş yedekleme özelliği aracılığıyla mümkün olur. Veritabanı anlık görüntüleri, özel olarak silinene kadar Amazon S3'te tutulan bulut sunucunuzun kullanıcı tarafından başlatılan kopyalarıdır.

**Dynamo DB**, belirli bir noktadan kurtarma (PITR), DynamoDB tablolarınızdaki verilerin sürekli yedeklenmesini sağlayarak, tabloyu önceki 35 gün içinde herhangi bir saniyeye geri yüklemenize olanak tanır. İsteğe bağlı yedeklemeleri ve geri yüklemeyi kullanarak veri arşivleme için DynamoDB tablolarınızdaki verilerin eksiksiz yedeklerini oluşturabilirsiniz.

**10. MAINTENANCE**

**Amazon RDS,** en son güncellemeler veritabanlarına uygulayacaktır. Veritabanı anınızın yama uygulanıp uygulanmayacağını ve ne zaman uygulanacağını seçebilirsiniz.

**DynamoDB,** sunucusuz olduğundan bakım gerektirmez.

**11. PRICING**

**Amazon RDS,** ücretlendirmeyi planlamanıza göre şekillendirir.

**DynamoDB,** tablolarınızda okuma ve yazma işlemlerini işlemek için özel faturalandırma seçenekleriyle birlikte gelen iki kapasite modu vardır: isteğe bağlı ve provizyonlu.

**12. USE CASES**

**Amazon RDS,** çoğunlukla geleneksel uygulamalarda, ERP'de, CRM'de ve e-ticarette kullanılır. Bu tür uygulamalar, verilerin gerçek zamanlı olarak alınmasını gerektirmez ve nispeten büyük ölçekte kullanılır.

**Amazon DymanoDB**, genellikle internet ölçekli uygulamalarda, gerçek zamanlı teklif vermede, alışveriş sepetlerinde, müşteri Tercihlerinde, içerik yönetiminde, Kişiselleştirmede ve mobil uygulamalarda kullanılır.

**AMAZON RDS VS AMAZON AURORA**

**1- Database Type:**

"RDS, MySQL, PostgreSQL, MariaDB, Oracle ve Microsoft SQL Server gibi çeşitli ilişkisel veritabanı motorlarını destekleyen yönetilen bir ilişkisel veritabanı hizmetidir.

"Aurora, standart MySQL ve PostgreSQL veritabanlarına kıyasla daha yüksek performans ve güvenilirlik sunan ilişkisel veritabanı motorudur."

**2- Use Cases:**

"RDS, yapılandırılmış veri ve karmaşık sorgular, işlemler ve ilişkiler için destek gerektiren uygulamalar için uygundur. Geleneksel ilişkisel uygulamalar için uygundur."

"Aurora, yüksek performans, güvenilirlik ve MySQL veya PostgreSQL ile uyumluluk gerektiren uygulamalar için tasarlanmıştır, bu da onu OLTP (Çevrimiçi İşlem İşleme) iş yükleri için iyi bir seçenek yapar."

**3-High Availability:**

"RDS, yüksek kullanılabilirlik için Multi-AZ dağıtımlar sunar. Burada yedek bir örnek, birincil örneğin başarısız olması durumunda otomatik bir geçiş için farklı bir Kullanılabilirlik Bölgesinde (AZ) korunur."

"Aurora aynı zamanda daha hızlı bir geçiş ve replikasyon ile Multi-AZ dağıtımları sunar. Aurora'nın mimarisi, daha yüksek kullanılabilirlik ve azalmış kesinti süresine katkıda bulunur.

**4-Depolama:**

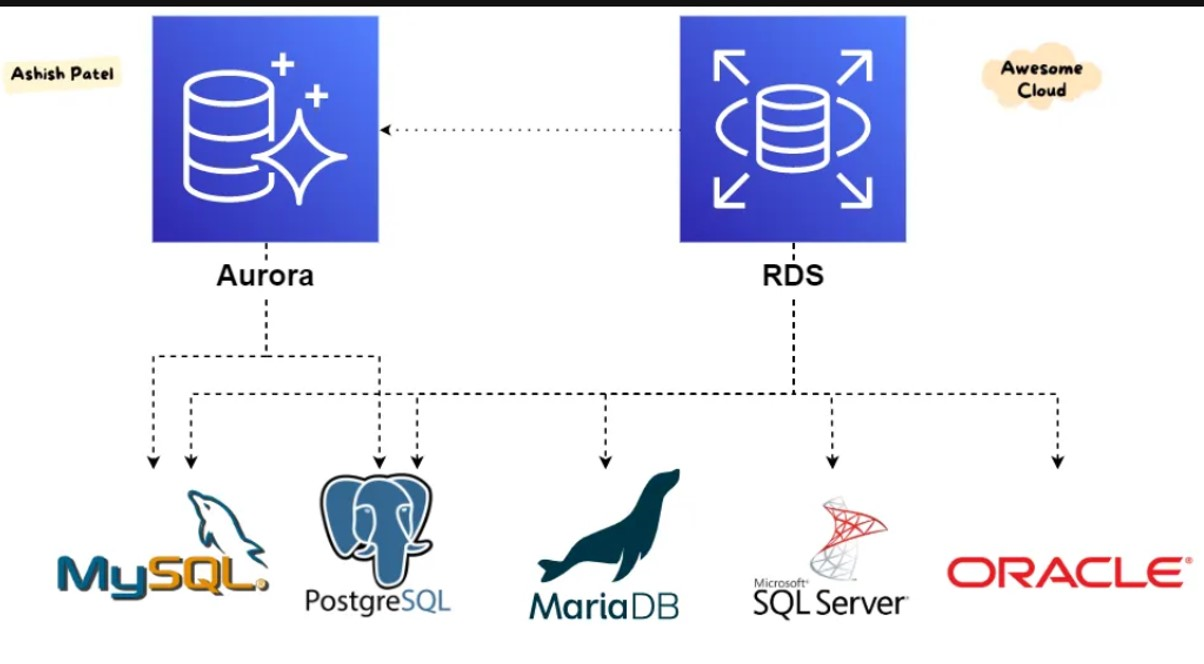
"RDS, Amazon EBS (Elastic Block Store) destekli depolama sağlar, bu da bazı I/O değişkenliklerini beraberinde getirebilir."

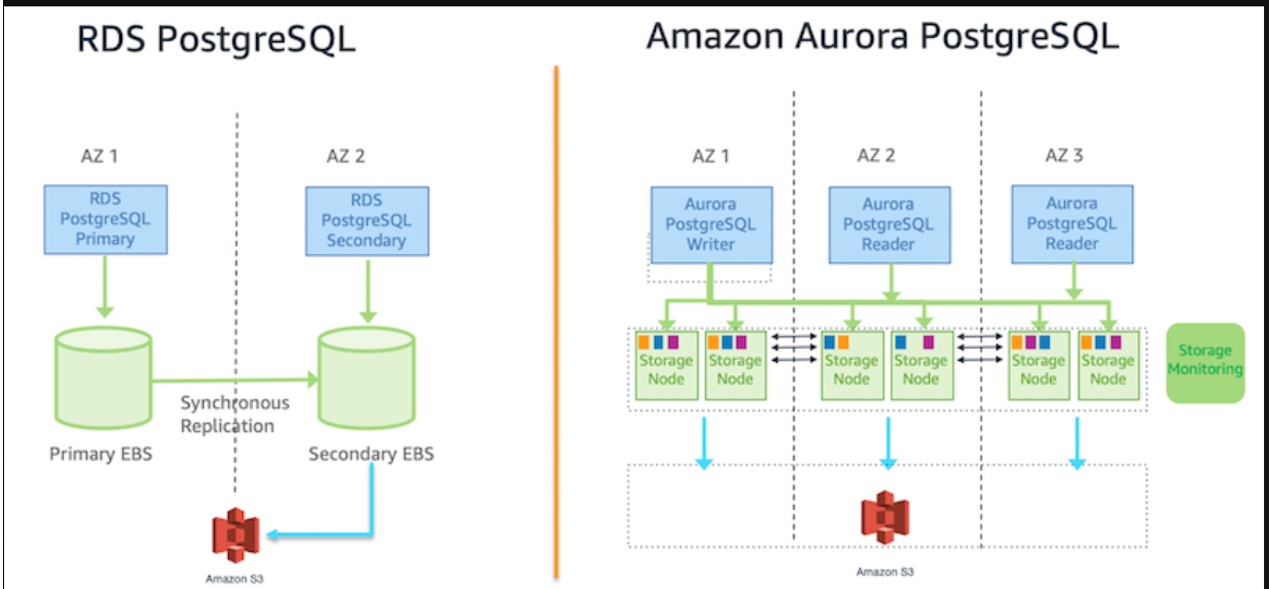
"Aurora, daha yüksek performans ve daha iyi güvenilirlik sunan dağıtılmış ve replike edilmiş bir depolama katmanı kullanır. Veri boyutu artsa bile tutarlı performans sağlamak üzere tasarlanmıştır."

**5-Performance:**

"RDS, geleneksel ilişkisel iş yükleri için iyi performans sunar, ancak performans seçilen veritabanı motoruna (MySQL, PostgreSQL, vb.) bağlı olarak değişebilir."

"Aurora, standart RDS veritabanlarından daha yüksek performans sağlamak üzere tasarlanmıştır. Paylaşılan depolama ile dağıtılmış bir mimari kullanır, bu da daha iyi okuma ve yazma performansına olanak tanır. Aurora yoğun iş yüklerini yönetebilir ve daha hızlı replikasyon ve kurtarma süreleri sağlar."





**RDS VS DYNAMO DB (NECİP BEY TARAFINDAN DERLENEN)**

Eventually consıstent reads : dynamodb’ye applıcatıon’larımız aynı anda sorgu gönderiyor bir şeyler okumaya çalişiyor, bir yandan da bir şeyler yazilmaya çalişiliyor. Bir sorgu veri yazarken, bir sorgu veri okuyor. Bir read operasyonu olduğunda dynamodb tablosuna gittiğinizde bir yandan da yazma işlemi de gerçekleşiyor. Tam read esnasinda write işlemini dikkate almiyorsunuz. Bu şuna sebebiyet verir, tam yazilma esnasindaki bilgiyi kaçirabilirsiniz. Ama daha hizli sonuç alirsiniz. Stale data olur.

Strongly consıstent reads : ben read sırasında wrıte olan sorguyu da öğrenmek en güncel datayi almak istiyorum dersen bunu kullan. Bu da biraz latency’e sebep oluyor. Ayrica daha çok kapasite kullaniliyor ve daha pahali.

Transactıonal reads: daha çok banka işlemlerinde olur. Eş zamanlı olarak bazı bilgilerin girilmesi ve okunması gerekebilir. Müşterinin bir parasi, ya da oyunda oyuncunun puani var. Bunu bir yerden başka bir yere transfer edecek. Bunu eş zamanli yapmazsa olmayan parayi ya da puani transfer yapabilir. Bu tarz hatalarin önüne geçmek için yapilir.

Read/wrıte capacıty’nin 2 modeli var.

On demand: hiçbir kisit yok. Ne kadar talep gelirse hepsini karşiliyor. Ne kadar talebi arttirirsan o kadar maliyetli oluyor.

Provisioned: ücreti optımıze (en uygun) etmek için önden siz ayariyorsunuz. Önden ne kadar kapasite tüketiyor bilmemiz gerekiyor.

Burada read/wrıte capacıty unıt’i sabitte yapabiliriz. Ya da auto scalıng’te yapabiliriz.

DynamoDB AWS’YE ÖZGÜ BİR ŞEYDİR.

DynamoDB SERVERLESS’TIR. YANİ ARKADA ÇALIŞAN INFRASTRUCTURE’A (SERVER, YÖNETİM VB.) SİZ HİÇ KARIŞMAZSINIZ. BUNU TAMAMEN AWS KARŞILAR. SİZ SADECE TABLOLARI OLUŞTURUR VERİLERİ GİRERSİNİZ.

AYNI ZAMANDA FULLY MANAGED’TIR.

DynamoDB NoSQL DATABASE’DİR.

DynamoDB’DE FLEXIBLE TABLES KULLNILABİLİR.

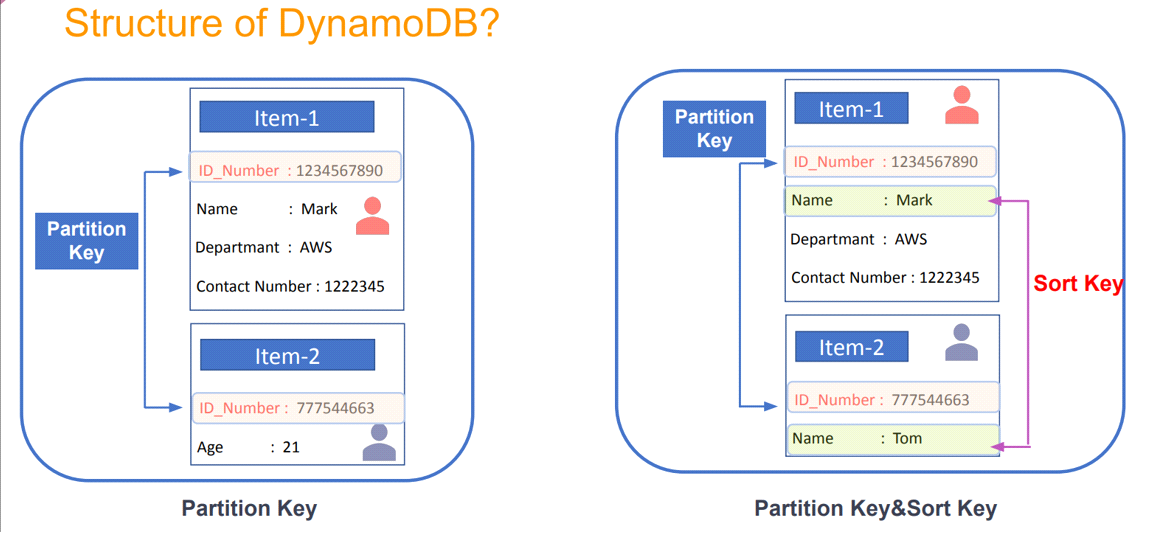
DynamoDB’DE JOIN FUNCTION YAPAMAZSIN.

NoSQL GEREĞİ HER ITEM’IN FARKLI ATTRIBUTE’LARI OLABİLİYOR.

HER ITEM’DE BİR TANE PRIMARY KEY OLMASI GEREKİYOR.

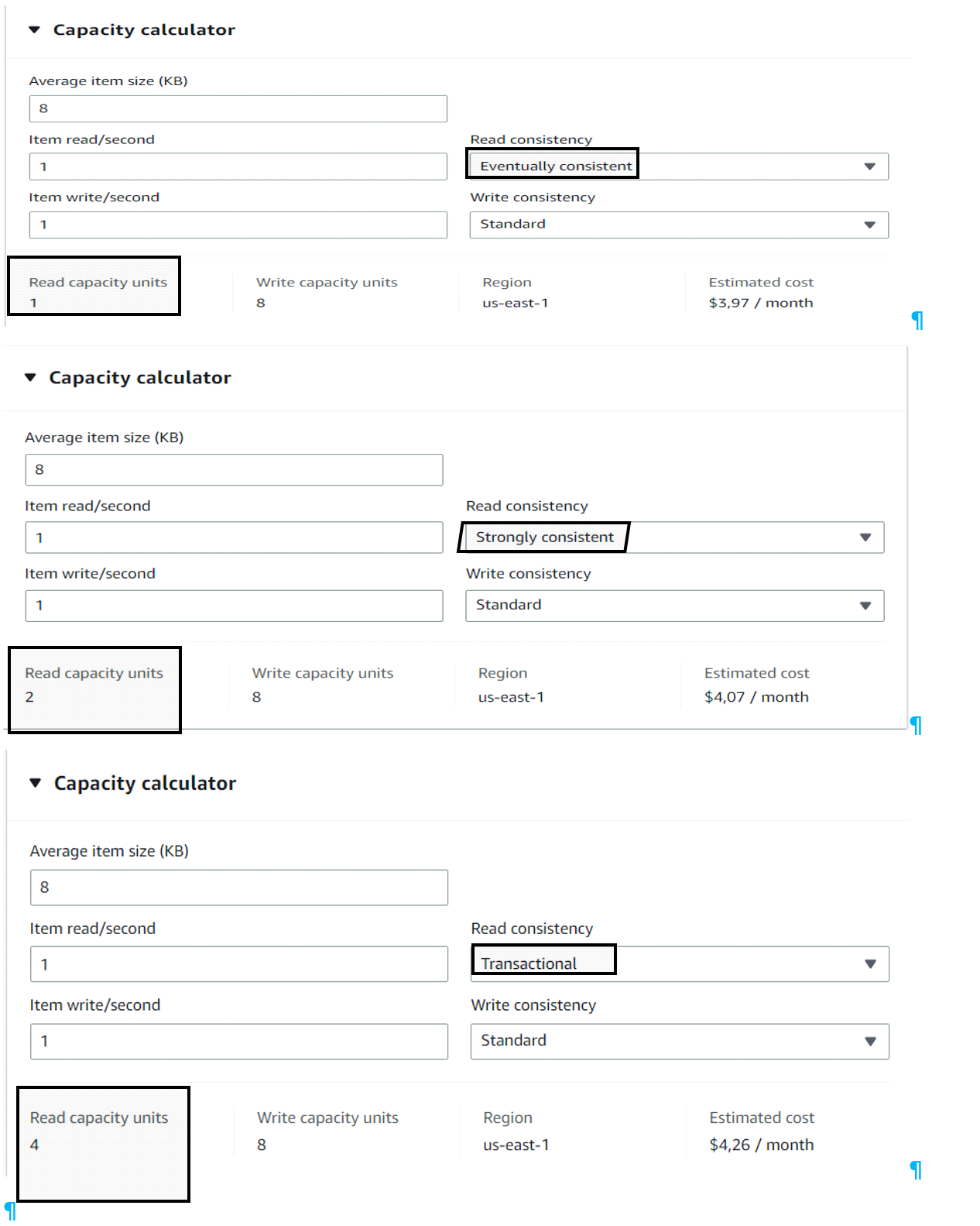
ÖNCE TABLOYU DAHA SONRA PRIMARY KEY’İ OLUŞTURMAMIZ GEREKİYOR.

PRIMARY KEY, PARTITION KEY VE PARTITION KEY + SORTY KEY OLMAK ÜZERE 2’YE AYRILIYOR.



SOLDAKİNDE PARTITION KEY HER ITEM’DA FARKLI OLMAK ZORUNDA. (BU SATIR MUTLAKA DOLDURULMASI GEREKİYOR)

SAĞDAKİNDE PARTITION KEY AYNI OLABİLİR AMA SORT KEY FARKLI OLMASI GEREKİR. (BU İKİ SATIR DA MUTLAKA DOLMASI GEREKİYOR)



EVENTUALLY CONSISTENT READS : DynamoDB’YE APPLICATION’LARIMIZ AYNI ANDA SORGU GÖNDERİYOR BİR ŞEYLER OKUMAYA ÇALIŞIYOR, BİR YANDAN DA BİR ŞEYLER YAZILMAYA ÇALIŞILIYOR. BİR SORGU VERİ YAZARKEN, BİR SORGU VERİ OKUYOR. BİR READ OPERASYONU OLDUĞUNDA DynamoDB TABLOSUNA GİTTİĞİNİZDE BİR YANDAN DA YAZMA İŞLEMİ DE GERÇEKLEŞİYOR. TAM READ ESNASINDA WRITE İŞLEMİNİ DİKKATE ALMIYORSUNUZ. BU ŞUNA SEBEBİYET VERİR, TAM YAZILMA ESNASINDAKİ BİLGİYİ KAÇIRABİLİRSİNİZ. AMA DAHA HIZLI SONUÇ ALIRSINIZ. STALE DATA OLUR.

STRONGLY CONSISTENT READS : BEN READ SIRASINDA WRITE OLAN SORGUYU DA ÖĞRENMEK EN GÜNCEL DATAYI ALMAK İSTİYORUM DERSEN BUNU KULLAN. BU DA BİRAZ LATENCY’E SEBEP OLUYOR. AYRICA DAHA ÇOK KAPASİTE KULLANILIYOR VE DAHA PAHALI.

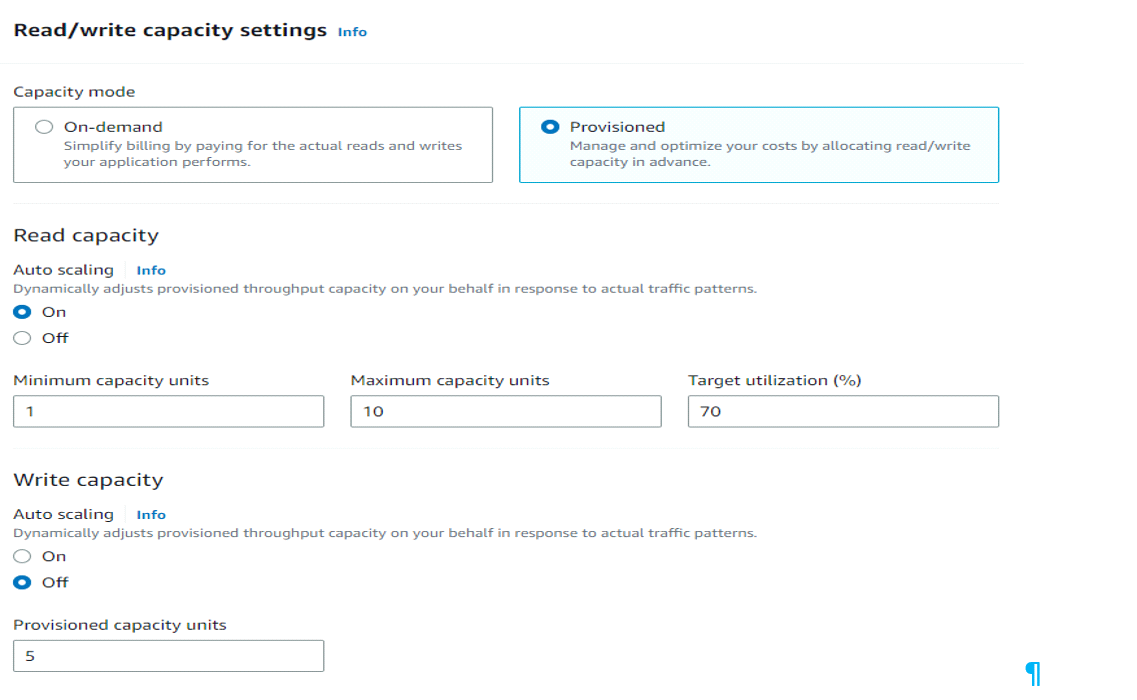
TRANSACTIONAL READS: DAHA ÇOK BANKA İŞLEMLERİNDE OLUR. EŞ ZAMANLI OLARAK BAZI BİLGİLERİN GİRİLMESİ VE OKUNMASI GEREKEBİLİR. MÜŞTERİNİN BİR PARASI, YA DA OYUNDA OYUNCUNUN PUANI VAR. BUNU BİR YERDEN BAŞKA BİR YERE TRANSFER EDECEK. BUNU EŞ ZAMANLI YAPMAZSA OLMAYAN PARAYI YA DA PUANI TRANSFER YAPABİLİR. BU TARZ HATALARIN ÖNÜNE GEÇMEK İÇİN YAPILIR.

READ/WRITE CAPACITY’NİN 2 MODELİ VAR.

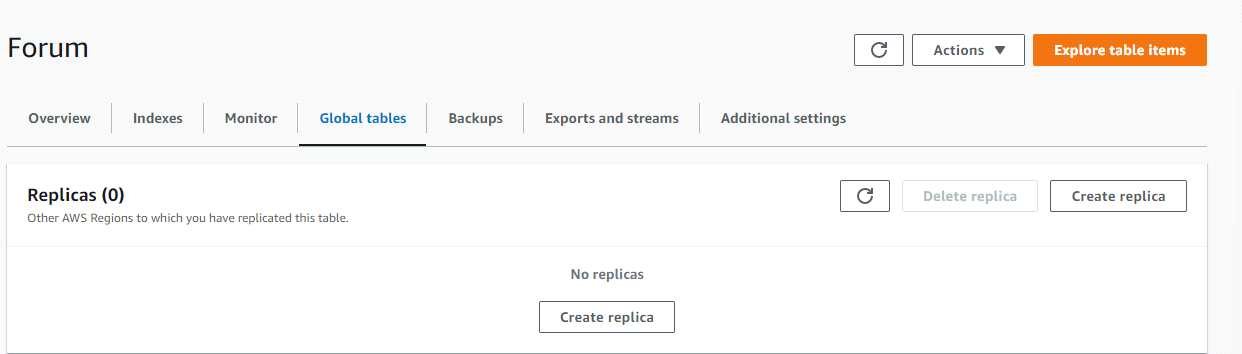
ON DEMAND: HİÇBİR KISIT YOK. NE KADAR TALEP GELİRSE HEPSİNİ KARŞILIYOR. NE KADAR TALEBİ ARTTIRIRSAN O KADAR MALİYETLİ OLUYOR.

PROVISIONED: ÜCRETİ OPTIMIZE (EN UYGUN) ETMEK İÇİN ÖNDEN SİZ AYARIYORSUNUZ. ÖNDEN NE KADAR KAPASİTE TÜKETİYOR BİLMEMİZ GEREKİYOR.

BURADA READ/WRITE CAPACITY UNIT’İ SABİTTE YAPABİLİRİZ. YA DA AUTO SCALING’TE YAPABİLİRİZ.

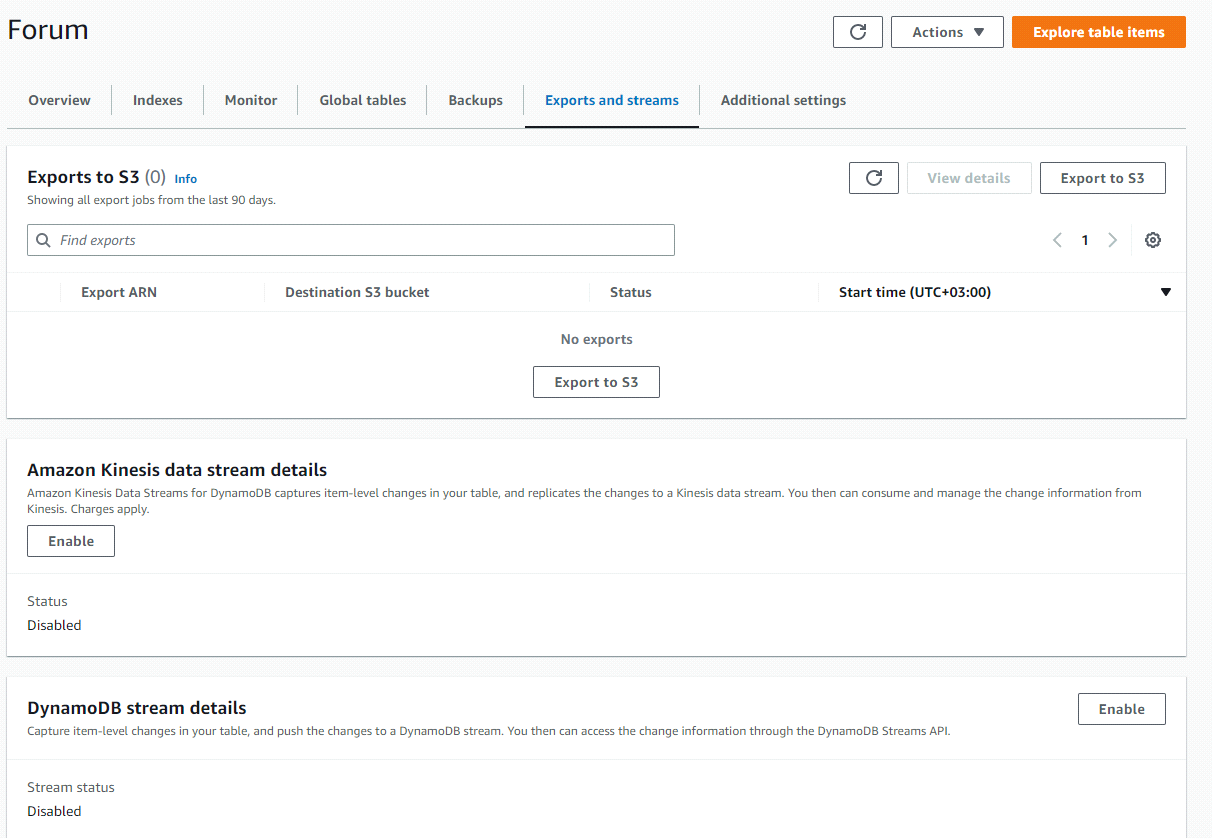


GLOBAL TABLES : SİZ BAŞKA REGIONLAR’DA DA OLUŞTURDUĞUNUZ TABLE’LARI REPLICA YAPABİLİRİZSİNİZ.

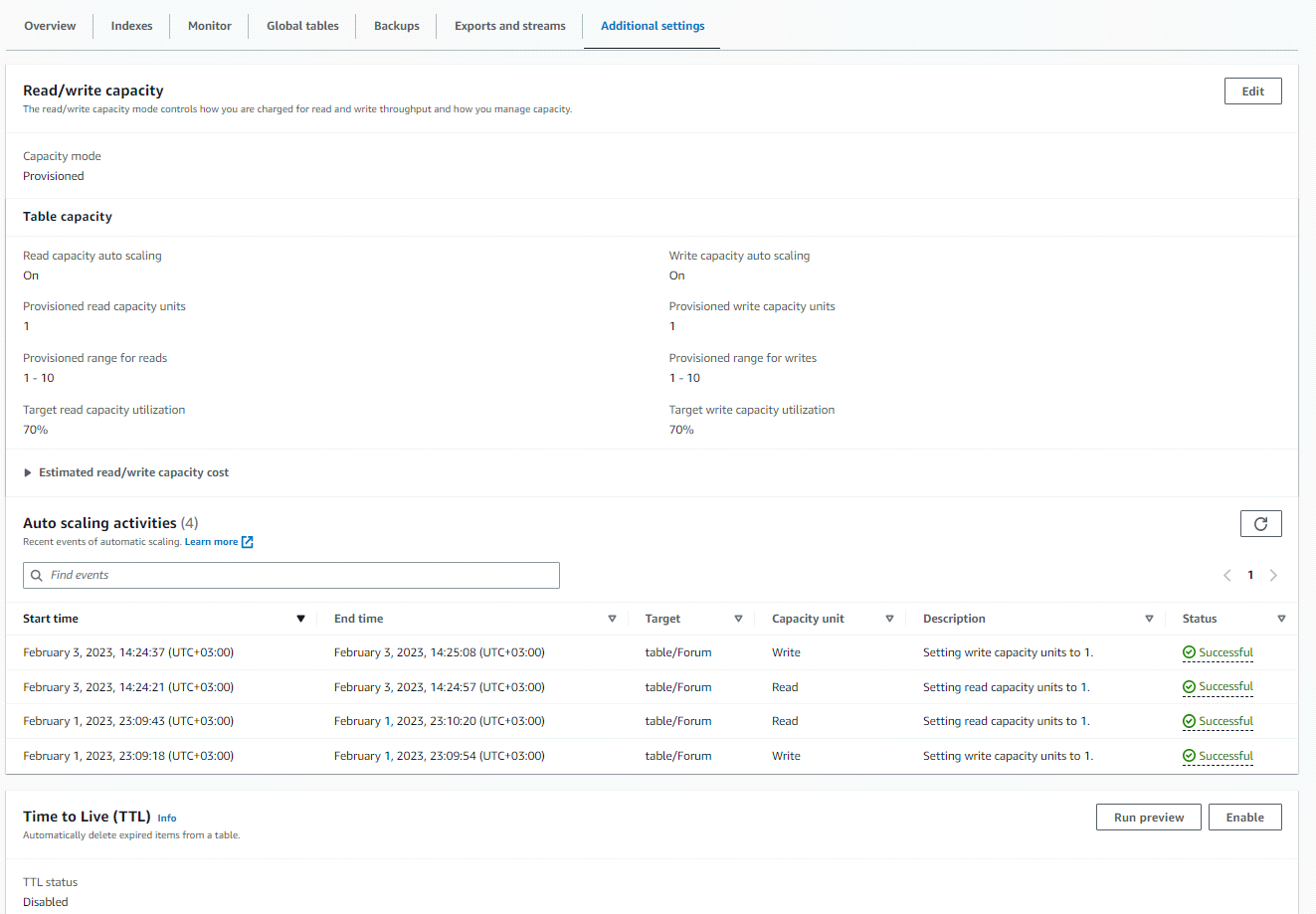


BACKUP ALABİLİYORSUNUZ.

EXPORT AND STREAMS >> DynamoDB STREAM DETAILS İLE DYNAMODB’DE DEĞİŞİKLİK OLDUYSA BURADAN GÖREBİLİYORUZ.



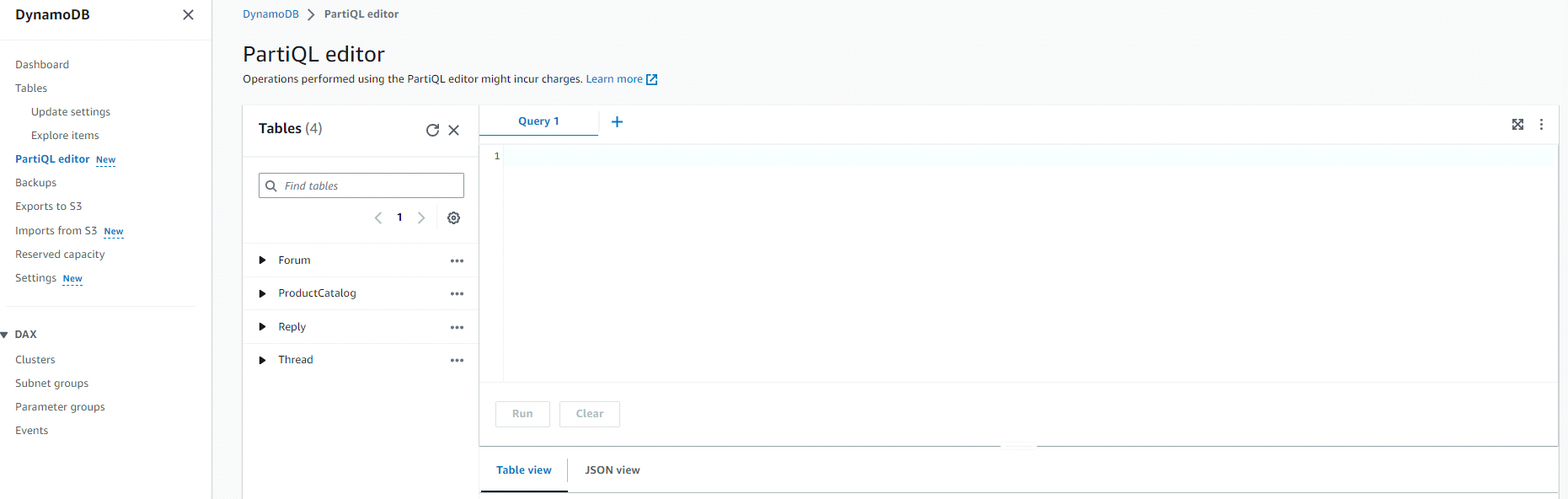
ADDITIONAL SETTINGS >> TIME TO LIVE TABLODAKİ BİR ITEM’A 2 SAAT İHTİYACIM VAR DİYORSUNUZ. 2 SAAT SONRA TABLODAKİ O ITEM KAYBOLUYOR.



DAX >> CLUSTER RDS’TEKİ ELASTICACHE GİBİ BUNU DYNAMODB’NİN ÖNÜNE KOYUP CACHE’LİYORSUNUZ. BÖYLECE DAHA HIZLI CEVAP ALABİLİYORSUNUZ.

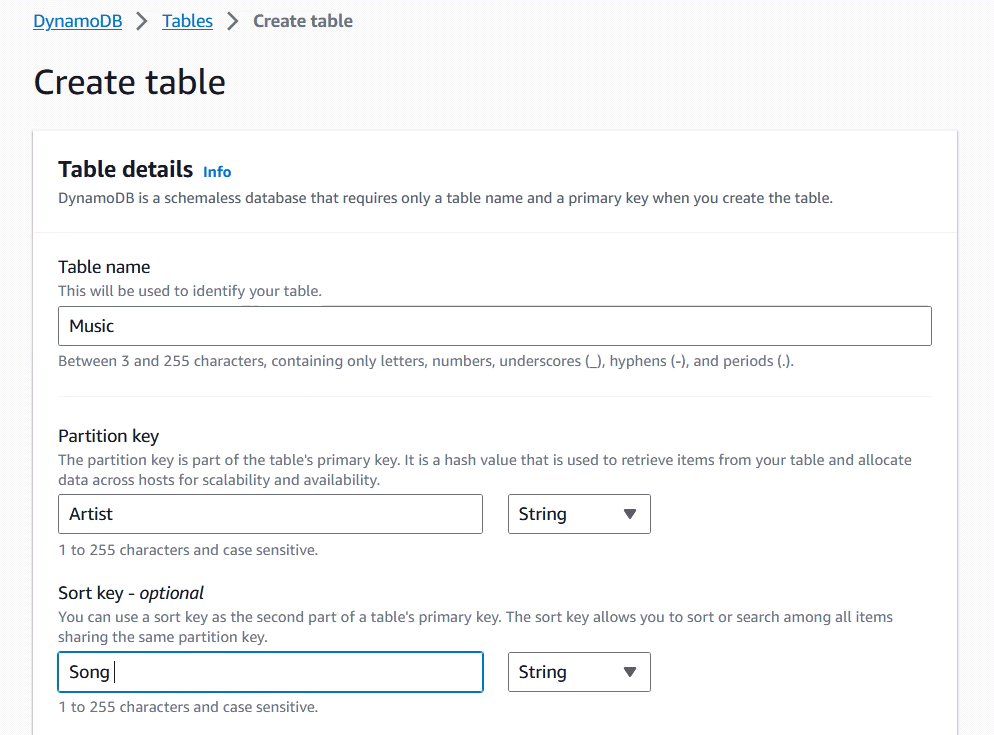
DAX: DYNAMODB ACCELERATOR

PartiQL editor >> SQL KOMUTLARI İLE SORGULAMA YAPABİLİYORSUNUZ.



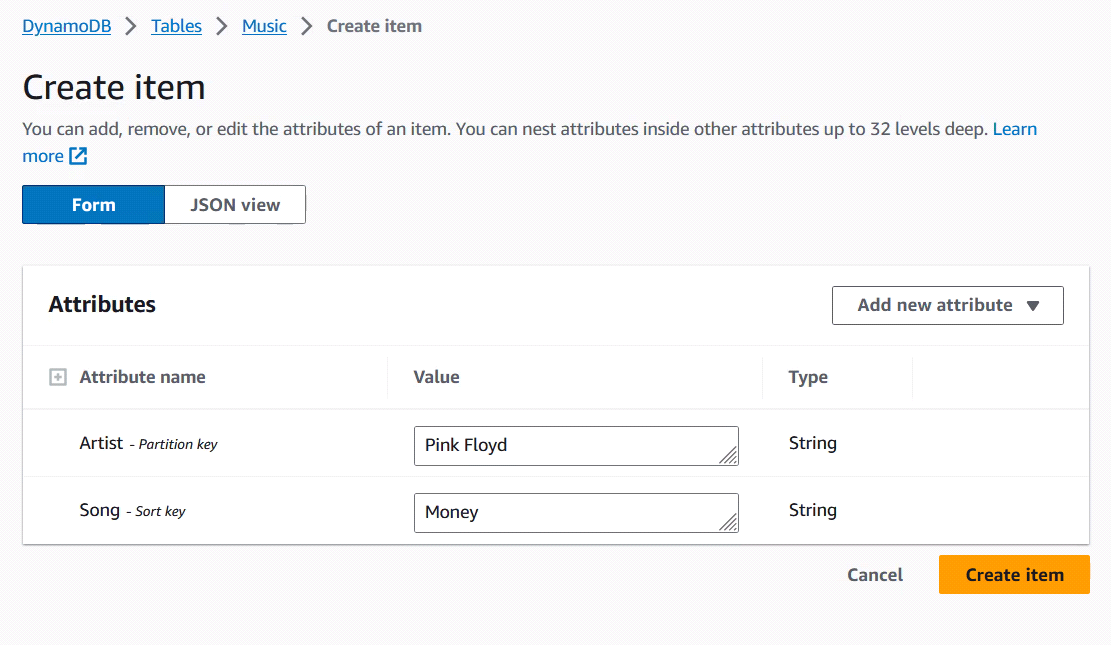
NECİP BEY’İN LAB SS’LERİ:

RDS İLE FARKLARI:

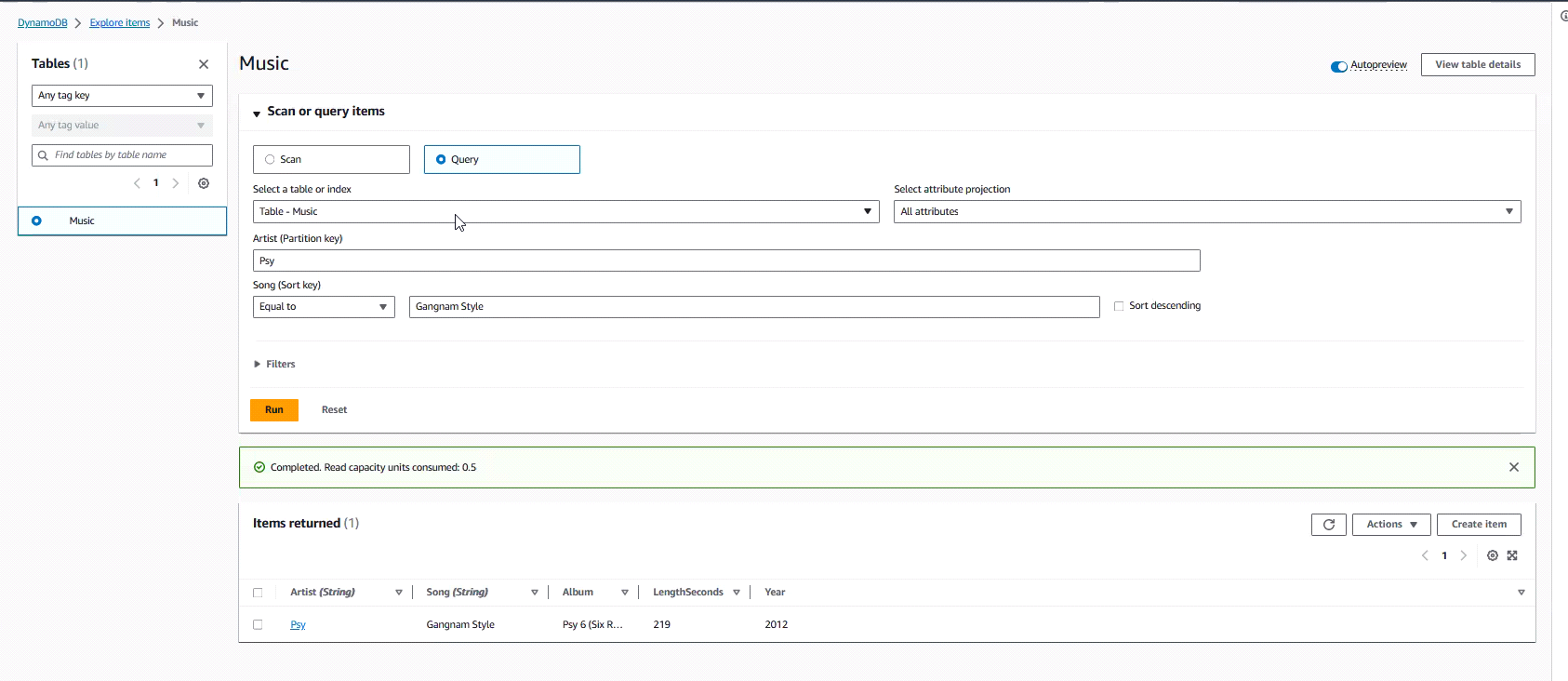


AWS CONSOLE ÜZERİNDEN BİR TABLE OLUŞTURBİLİYORUZ.

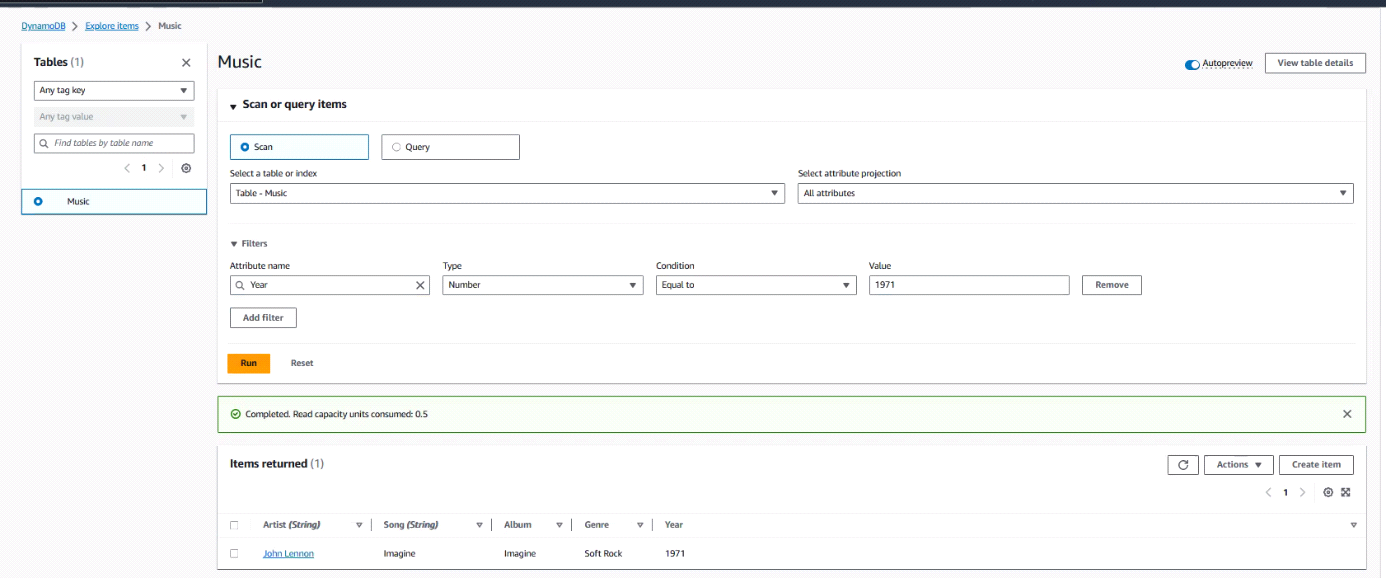
AWS CONSOLE ÜZERİNDEN OLUŞTURDUĞUMUZ TABLOYA BİLGİLER GİREBİLİYORUZ.



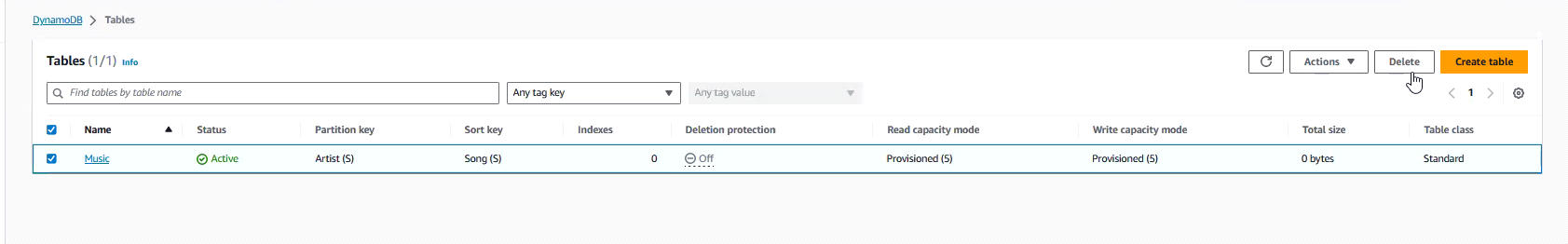
AWS CONSOLE ÜZERİNDEN OLUŞTURDUĞUMUZ TABLODA “QUERY” YAPABİLİYORUZ



AWS CONSOLE ÜZERİNDEN OLUŞTURDUĞUMUZ TABLODA “SCAN” YAPABİLİYORUZ



AWS CONSOLE ÜZERİNDEN OLUŞTURDUĞUMUZ TABLOYU SİLEBİLİYORUZ.



**Emine Hanım’ın tespitleri:**

\* DynamoDB, saklanan bilgilere erişmek için iki yol sunar: Sorgulama ve Tarama. Bir Sorgu, bilgi bulmak için primary güvenir. Sorgu, doğrudan belirli bir öğeye (veya diğer öğeler grubuna) işaret edebilir ve bunları hızlı ve verimli bir şekilde alabilir. Scan, adından da anlaşılacağı gibi, tablo öğelerini baştan sona tarar.

**Fatih Bey’in illetiği bilgiler:**

\* Aralarındaki en önemli fark, DynamoDB bir NoSQL veritabanı, Aurora ise ilişkisel bir veritabanı olmasıdır.

\* Ayrıca, çok büyük miktarda veriyi göz önünde bulundurursak, petabaytlarca veriyi destekleyen ve sorunsuz bir şekilde yukarı veya aşağı ölçekleme yapan DynamoDB kazanır. Bununla birlikte, Aurora küme hacmi de 128 Terabayta kadar büyüyebilir, ancak boyutunu değiştirirken 15 dakika kadar sürebilir.

\* Amazon DynamoDB her türlü ölçekleme için hızlı ve esnek bir NoSQL servisidir.

DynamoDB yönetimi AWS tarafından yapılıyor ve Key-value, doküman tipi veri depolaması yapılıyor. Bunu yaparken her türlü ölçeklemede milisaniye bazında performans sunar.

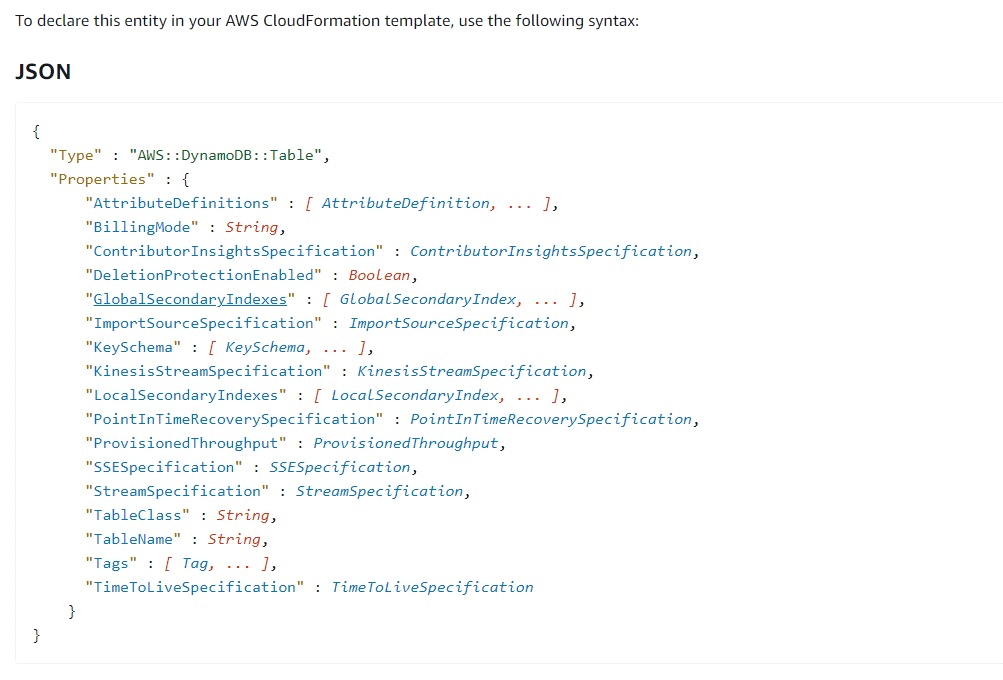
Create table diyerek doğrudan tablo oluşrabiliyoruz.

\* Partition Key tablonun primary key’nin bir parçasıdır. binary, string ve number olarak değer verebiliyoruz. Ölçeklenebilirlik ve kullanılabilirlik için tablonuzdan öğeler almak ve ana bilgisayarlar arasında veri tahsis etmek için kullanılan bir karma değerdir.

\* Sort-key ise opsiyonel bir değerdir. Sort key, primary key’in 2.parçasıdır.

Sort Key, aynı bölümleme anahtarını paylaşan tüm öğeler arasında sıralama yapmanızı veya arama yapmanızı sağlar.

\* JSON ile tablo özellikleri ni ayarlayabiliyoruz.

JSON dosya örneği:

**Security:**

**Encryption at Rest**

Both DynamoDB and Aurora support encryption at rest using encryption keys stored in AWS KMS (Key Management Service). We can use AWS KMS to create, store, and manage their encryption keys.

On the other hand, Amazon DynamoDB users have three options to encrypt the table with the use of AWS KMS.

The key of DynamoDB is encrypted by default, and it is at zero cost (AWS-owned key).

KMS will secure the keys stored in your account, and a fee will apply (AWS-managed key).

KMS Customer Management Keys (CMK) gives users complete control over KMS keys and charges a price (Customer-managed key).

Amazon Aurora has AWS-managed keys and customer-managed keys. Customer managed or customer master keys will be known as the CMK feature in AWS KMS.

DynamoDB

DynamoDB can create database tables that can store data and serve any level of on-demand traffic. In addition, the high IO performance of SSDs provides less latency and response time for storing and reading data at any scale.

This server instance supports the computational resources needed for database operations. Regardless of the size of your table, it maintains low latency (millisecond latency) behind this practice. In addition, maintaining a better table layout, using the keys and indexes effectively for queries, and streamlining the database workload, optimizes DynamoDB’s performance.

DynamoDB handles up to 10 trillion requests per day and more than 20 million requests per second. I hope you can simply imagine how scalable it is by looking at these numbers.

Aurora

Amazon Aurora is a MySQL and PostgreSQL-compatible relational database solution. It has a five-time throughput rate as standard MySQL output and a three-time throughput rate as standard PostgreSQL output. According to AWS, the Storage service design for Aurora makes this possible.

One of the basic design principles for this storage service is to minimize the latency in writing in the foreground. This design takes the majority of storage processing to the background. As a result, it improves throughput and makes it a high-performing cloud-native relational database in AWS.

bir de anladığım kadarıyla DynamoDB de ec2 olmadan tablo oluşturabiliyoruz ama aurora da ec2 ya ihtiyaç duyuyoruz.

FİYATLANDIRMA

Amazon Aurora

AWS, otomatikleştirilmiş kümeleme, yeniden tasarım ve depolamayı denetleyecektir. Depolama tüketimi ve GÇ tüketimi için faturalandırılır. AWS, veritabanı depolama tüketimi için GB-aylık artışlarla ücretlendirilir ve tüketilen G/Ç'ler, milyon istek artışı başına faturalandırılır. Bu bağlantı, nasıl hesaplandığına ilişkin basitleştirilmiş örnekler sağlayacaktır.

Global Database , Backtrack, Snapshot Export gibi özellikler ve Aurora'dan veri aktarımı ayrıca ücretlendirilir. Ayrıca Amazon Aurora'nın MySQL uyumlu ve PostgreSQL uyumlu sürümleri için iki farklı fiyatlandırma tablosu bulunmaktadır.

Amazon DynamoDB

DynamoDB, veri okuma, yazma ve depolama dışında birçok isteğe bağlı özellik içerir. Bu hizmetler ücrete tabidir. Vurgulanması gereken en önemli şey, isteğe göre isteğe bağlı veya tedarik kapasitesi seçerek tercihlerinize göre doğru fiyatlandırma modelini seçmektir.

İsteğe bağlı kapasite modu  — DynamoDB, iş yükünüzü gerektiğinde yukarı veya aşağı uyarlar, böylece uygulamanızın çalıştırmasını beklediğiniz okuma ve yazma verimi miktarını belirtmeniz gerekmez. Okuma ve yazma eylemlerinin fiyatı dinamiktir ve uygulamaya alınan trafiğe bağlıdır.

Tedarik edilen kapasite modu  — Burada, uygulamanız için gereken okuma ve yazma isteklerinin sayısını belirtmeniz gerekir. Daha sonra okuma yazma fiyatları bunlara göre hesaplanacaktır. İlk 25 GB depolama kullanımı,isteğe bağlı kapasite ve tedarik edilen kapasite modları için AWS ücretsiz kullanımı kapsamına girer.