Migrieren Ihrer Datenbanken zu Amazon Aurora

Juni 2016

Hinweise

Dieses Dokument wird nur zu Informationszwecken zur Verfügung gestellt. Es stellt das aktuelle Produktangebot und die Verfahren von AWS zum Ausstellungsdatum dieses Dokuments dar. Änderungen vorbehalten. Kunden sind für ihre eigene unabhängige Einschätzung der Informationen in diesem Dokument und jedwede Nutzung der AWS-Services verantwortlich. Jeder Service wird „wie besehen“ ohne Gewähr und ohne Garantie jeglicher Art, weder ausdrücklich noch impliziert, bereitgestellt. Dieses Dokument gibt keine Garantien, Gewährleistungen, vertraglichen Verpflichtungen, Bedingungen oder Zusicherungen von AWS, seinen Partnern, Zulieferern oder Lizenzgebern. Die Verantwortung und Haftung von AWS gegenüber seinen Kunden werden durch AWS-Vereinbarungen geregelt. Dieses Dokument ist weder ganz noch teilweise Teil der Vereinbarungen von AWS mit seinen Kunden und ändert diese Vereinbarungen auch nicht.

Inhalt

Kurzbeschreibung

Amazon Aurora ist eine MySQL-kompatible, relationale Datenbank-Engine der Enterprise-Klasse. Da es sich bei Amazon Aurora um eine Cloud-gestützte Datenbank handelt, bietet sie zahlreiche Vorteile gegenüber herkömmlichen, relationalen Datenbank-Engines. Dieses Whitepaper zeigt bewährte Methoden für die Migration bestehender Datenbanken zu Amazon Aurora auf. Abgesehen von Überlegungen zur Migration wird darin der Prozess für die Migration von Open-Source- und kommerziellen Datenbanken zu Amazon Aurora bei minimaler Unterbrechung der Anwendungen Schritt für Schritt beschrieben.

Einführung in Amazon Aurora

Seit Jahrzehnten sind herkömmliche relationale Datenbanken die erste Wahl für Datenspeicherung und Persistenz. Solche Datenbanksysteme greifen nach wie vor auf monolithische Architekturen zurück und sind auch nicht dafür ausgelegt, die Vorteile einer Cloud-Infrastruktur zu nutzen. Diese monolithischen Architekturen bringen viele Herausforderungen mit sich, insbesondere in Bezug auf Kosten, Flexibilität und Verfügbarkeit. Um diese Herausforderungen anzugehen, hat AWS die relationale Datenbank für die Cloud-Infrastruktur neu gestaltet und Amazon Aurora eingeführt.

Amazon Aurora ist eine MySQL-kompatible relationale Datenbank-Engine, welche die Geschwindigkeit, Verfügbarkeit und Sicherheit einer hochwertigen kommerziellen Datenbank mit der Einfachheit und Wirtschaftlichkeit von Open-Source-Datenbanken vereint. Aurora bietet eine bis zu fünfmal höhere Leistung als MySQL-Datenbanken und hochwertige kommerzielle Datenbanken mit vergleichbarer Leistung. Dabei schlägt Amazon Aurora mit nur einem Zehntel der Kosten kommerzieller Engines zu Buche.

Amazon Aurora ist über die Amazon Relational Database Service (Amazon RDS)-Plattform verfügbar. Ebenso wie andere Amazon RDS-Datenbanken ist Aurora ein vollständig verwalteter Datenbankservice. Mit der Amazon RDS-Plattform werden die meisten Datenbankverwaltungsaufgaben wie Hardwarebereitstellung, Software-Patches, Einrichtung, Konfiguration, Überwachung und Sicherung vollständig automatisiert.

Amazon Aurora ist für geschäftskritische Workloads und hohe Verfügbarkeit ausgelegt. Ein Aurora-Datenbank-Cluster erstreckt sich über mehrere Availability Zones (Verfügbarkeitszonen) in einer Region. Damit wird eine sofortige Beständigkeit und Fehlertoleranz für Ihre Daten über mehrere physische Rechenzentren hinweg erreicht. Eine Availability Zone (AZ) besteht aus einem oder mehreren hochverfügbaren, von Amazon betriebenen Rechenzentren. Availability Zones sind voneinander isoliert und über Verbindungen mit niedriger Latenz miteinander verbunden. Jedes Segment eines Datenbank-Volumes wird sechsmal über diese Availability Zones hinweg repliziert.

Die Volumes der Aurora-Cluster werden automatisch mit der wachsenden Datenmenge in Ihrer Datenbank vergrößert, und zwar gänzlich ohne Leistungs- oder Verfügbarkeitseinbußen. Sie müssen daher weder im Voraus die Menge des später benötigten Datenspeichers schätzen, noch große Mengen an Datenbankspeicher vorab bereitstellen. Ein Aurora-Cluster-Volume kann bis auf eine maximale Größe von 64 Terabyte (TB) wachsen. Ihnen wird lediglich der Speicherplatz in Rechnung gestellt, den Sie in einem Aurora-Cluster-Volume belegen.

Die automatische Sicherungsfunktion von Aurora unterstützt die zeitpunktbezogene Wiederherstellung Ihrer Daten. Damit können Sie Ihre Datenbank sekundengenau für jeden Zeitpunkt innerhalb Ihres Aufbewahrungszeitraums von maximal fünf Minuten wiederherstellen. Automatisierte Sicherungen werden in Amazon Simple Storage Service (Amazon S3) gespeichert, einem Service, der für eine Beständigkeit von 99,999999999 % ausgelegt ist. Sicherungen erfolgen in Amazon Aurora automatisch, inkrementell und fortlaufend. Die Leistung der Datenbank wird dadurch nicht beeinträchtigt.

Für Anwendungen, die Read Replicas (schreibgeschützte Replikate) benötigen, können Sie bis zu 15 Aurora-Replicas pro Aurora-Datenbank mit sehr niedriger Replica-Verzögerung erstellen. Da diese Replicas denselben Speicher wie die

Quell-Instance nutzen, ergeben sich niedrigere Kosten und es müssen auch keine Daten auf die Replica-Knoten geschrieben werden.

Amazon Aurora ist überaus sicher und ermöglicht die Verschlüsselung Ihrer Datenbanken. Die hierzu benötigten Schlüssel erstellen und verwalten Sie mit dem AWS Key Management Service (AWS KMS). Bei einer mit Amazon Aurora ausgeführten Datenbank-Instance werden ruhende Daten im zugrunde liegenden Speicher verschlüsselt. Gleiches gilt für die automatisierten Sicherungen, Snapshots und Replicas im gleichen Cluster. Amazon Aurora schützt Daten während der Übertragung mit SSL (AES-256).

Eine vollständige Liste der Aurora-Funktionen finden Sie auf der Produktseite von Amazon Aurora.[1] Angesichts der umfassenden Funktionen und Kosteneffektivität wird Amazon Aurora verstärkt als Datenbank für geschäftskritische Anwendungen wahrgenommen.

Überlegungen zur Datenbankmigration

Bei den meisten Anwendungen bildet die Datenbank eine wichtige Architekturkomponente. Die Migration der Datenbank auf eine neue Plattform ist ein bedeutendes Ereignis im Lebenszyklus einer Anwendung, das Auswirkungen auf die Funktionalität, Leistung und Zuverlässigkeit haben kann. Bevor Sie Ihr erstes Projekt zur Migration auf Amazon Aurora in Angriff nehmen, sollten Sie einige wichtige Aspekte berücksichtigen.

Migrationsphasen

Da Datenbankmigrationen häufig komplex sind, befürworten wir einen stufenweisen, iterativen Ansatz.

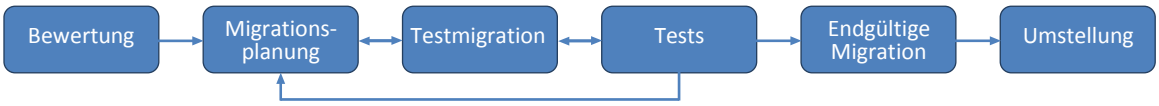

Abbildung 1: Migrationsphasen

Überlegungen zu Anwendungen

Bewertung der Aurora-Funktionen

Zwar können die meisten Anwendungen mit vielen relationalen Datenbank-Engines zusammen eingesetzt werden, aber Sie sollten sicherstellen, dass Ihre Anwendung mit Amazon Aurora kompatibel ist. Amazon Aurora ist Wire-kompatibel mit MySQL 5.6. Daher können die heute für MySQL-Datenbanken verwendeten Programme, Anwendungen, Treiber und Tools größtenteils mit geringfügigen oder ganz ohne Änderungen in Aurora eingesetzt werden.

Bestimmte MySQL-Funktionen, etwa die MyISAM-Speicher-Engine, sind jedoch bei Amazon Aurora nicht verfügbar. Da es sich bei Aurora um einen verwalteten Service handelt, ist der SSH-Zugriff auf Datenbankknoten beschränkt. Unter Umständen können Sie daher nicht alle Drittanbieter-Tools oder -Plugins auf dem Datenbank-Host installieren.

Überlegungen zur Leistung

Die Datenbankleistung ist ein wichtiger Aspekt bei der Migration einer Datenbank auf eine neue Plattform. Viele erfolgreiche Datenbank-Migrationsprojekte beginnen daher mit einer Bewertung der Leistung der neuen Datenbankplattform. Die Ausführung von Sysbench- und TPC-C-Benchmarks mag zwar einen brauchbaren Eindruck der Gesamtleistung vermitteln, doch diese Benchmarks können die Zugriffsmuster Ihrer Anwendungen nicht emulieren. Brauchbarere Ergebnisse erhalten Sie, wenn Sie die Datenbankleistung in Bezug auf zeitkritische Workloads testen. Führen Sie dazu Ihre Abfragen (oder eine Teilmenge davon) direkt auf der neuen Plattform aus.

Ziehen Sie die folgenden Strategien in Erwägung:

- Handelt es sich bei Ihrer aktuellen Datenbank um eine MySQL-Datenbank, planen Sie bei der Migration zu Amazon Aurora Ausfallzeiten ein und testen Sie die Leistung Ihrer Datenbank mit einer Test- oder Staging-Version Ihrer Anwendung oder durch die Simulation von Produktions-Workloads.

- Handelt es sich um eine nicht mit MySQL konforme Engine, können Sie die Tabellen mit dem größten Zugriffsvolumen in Amazon Aurora kopieren und Ihre Abfragen für diese Tabellen testen. Dies bietet Ihnen einen guten Ausgangspunkt. Ein echtes Gesamtbild der realen Leistung Ihrer

Anwendung auf der neuen Plattform liefern natürlich erst Tests nach Abschluss der Datenmigration.

Amazon Aurora bietet eine vergleichbare Leistung wie kommerzielle Engines und eine erhebliche Verbesserung der Leistung im Vergleich zu MySQL. Erreicht wird dies durch die enge Integration der Datenbank-Engine in einen SSD-basierten, virtualisierten Speicher-Layer, der speziell für Datenbank-Workloads entwickelt wurde. Dies reduziert die Zahl der Schreibvorgänge in das Speichersystem, minimiert Sperrkonflikte und eliminiert Verzögerungen aufgrund von Datenbankprozess-Threads. Unsere Tests mit SysBench auf r3.8xlarge-Instances haben ergeben, dass Amazon Aurora über 585 000 Lesevorgänge und 107 000 Schreibvorgänge pro Sekunde erreicht. Das entspricht dem Fünffachen des Ergebnisses, das MySQL bei Verwendung desselben Benchmarks auf derselben Hardware liefert.

In einem Bereich ist Amazon Aurora deutlich besser als herkömmliche MySQL-Datenbanken: wenn eine große Zahl nicht sequenzieller Workloads vorliegt. Um den Durchsatz Ihrer Workloads in Amazon Aurora zu optimieren, empfiehlt es sich, Anwendungen so zu gestalten, dass vorzugsweise eine große Anzahl von Abfragen gleichzeitig ausgeführt wird.

Überlegungen zu Sharding und Read Replica

Wenn Ihre aktuelle Datenbank durch Sharding über mehrere Knoten hinweg aufgeteilt ist, haben Sie die Möglichkeit, diese Shards bei der Migration in einer einzigen Aurora-Datenbank zusammenzuführen. Eine einzelne Amazon Aurora-Instance ist auf bis zu 64 TB skalierbar und bietet Unterstützung für Tausende von Tabellen und für eine wesentlich höhere Anzahl von Lese- und Schreibvorgängen als eine Standard-MySQL-Datenbank.

Bei Anwendungen mit einer großen Anzahl von Lese- und Schreibzugriffen empfiehlt sich die Verwendung von Aurora Read Replicas zur Auslagerung der Lese-Workloads aus dem Master-Datenbankknoten. Auf diese Weise kann die Anzahl der gleichzeitig ausgeführten Schreibvorgänge bei Ihrer primären Datenbank und damit die Lese- und Schreibleistung insgesamt verbessert werden. Read Replicas bieten außerdem die Möglichkeit, Kosten bei einer Multi-AZ-Konfiguration einzusparen, da Sie unter Umständen kleinere Instances für Ihre primäre Instance verwenden und zugleich Ihrem Datenbank-Cluster Failover-Funktionen hinzufügen können. Aurora Read Replicas bieten eine nahezu verzögerungsfreie Replikation und Sie können bis zu 15 Read Replicas erstellen.

Überlegungen zur Zuverlässigkeit

Ein wichtiger Aspekt bei Datenbanken ist eine hohe Verfügbarkeit und die Notfallwiederherstellung. Bestimmen Sie die Anforderungen Ihrer Anwendung in Bezug auf RTO (Recovery Time Objective, Wiederherstellungsdauer) und RPO (Recovery Point Objective, Wiederherstellungszeitpunkt). Mit Amazon Aurora können Sie erhebliche Verbesserungen bei diesen beiden Faktoren erreichen.

In den meisten Datenbank-Absturzszenarien reduziert Amazon Aurora die Dauer des Neustarts auf unter 60 Sekunden. Aurora lagert außerdem den Puffercache aus dem Datenbankprozess aus und macht ihn zum Zeitpunkt des Neustarts sofort verfügbar. In den seltenen Szenarien, in denen es zu Ausfällen der Hardware und der Availability Zone kommt, wird die Wiederherstellung automatisch von der Datenbankplattform abgewickelt.

Aurora bietet innerhalb einer AWS-Region Wiederherstellungszeitpunkte von null Minuten (Zero RPO), eine wesentliche Verbesserung gegenüber lokalen Datenbanksystemen. Aurora unterhält sechs Kopien der Daten, die auf drei Availability Zones verteilt sind, und versucht automatisch, Ihre Datenbank in einer fehlerfreien AZ ohne Datenverlust wiederherzustellen. Im unwahrscheinlichen Fall, dass Ihre Daten im Amazon Aurora-Speicher nicht verfügbar sind, können Sie sie aus einem DB-Snapshot wiederherstellen oder eine zeitpunktbezogene Wiederherstellung auf eine neue Instance vornehmen.

Überlegungen in Bezug auf Kosten und Lizenzierung

Der Besitz und der Betrieb von Datenbanken ist mit Kosten verbunden. Bevor Sie eine Datenbankmigration planen, sollten Sie unbedingt die Gesamtbetriebskosten (TCO, Total Cost of Ownership) der neuen Datenbankplattform analysieren. Die Migration auf eine neue Datenbankplattform sollte idealerweise die Gesamtbetriebskosten senken und zugleich ähnliche oder bessere Funktionen für Ihre Anwendungen bereitstellen. Wenn Sie eine Open Source-Datenbank-Engine (MySQL, Postgres) ausführen, stehen Ihre Kosten vorwiegend im Zusammenhang mit Hardware-, Server- und Datenbankverwaltung. Wenn Sie jedoch eine kommerzielle Datenbank-Engine (Oracle, SQL Server, DB2 usw.) betreiben, machen die Datenbanklizenzen einen erheblichen Teil Ihrer Kosten aus.

Da Aurora zu einem Zehntel der Kosten von kommerziellen Engines verfügbar ist, lassen sich die Gesamtbetriebskosten bei vielen Anwendungen durch die

Verlagerung auf Aurora beträchtlich senken. Aufgrund der hochperformanten Read Replicas mit ihrer Doppelfunktion können Sie durch einen Umstieg auf Amazon Aurora erhebliche Einsparungen erzielen, selbst wenn Sie eine Open Source Engine wie MySQL oder Postgres betreiben. Weitere Informationen finden Sie auf der Seite Amazon RDS for Aurora – Preise.[2]

Sonstige Überlegungen zur Migration

Nachdem Sie die Eignung der Anwendung, deren Leistung, die Gesamtbetriebskosten und die Zuverlässigkeit in Betracht gezogen haben, sollten Sie sich Gedanken darüber machen, welchen Aufwand eine Migration auf die neue Plattform mit sich bringt.

Einschätzung des Aufwands für Codeänderungen

Es ist wichtig, den Umfang der Code- und Schemaänderungen abzuschätzen, die Sie im Rahmen der Migration Ihrer Datenbanken zu Amazon Aurora vornehmen müssen. Bei der Migration von MySQL-kompatiblen Datenbanken sind die erforderlichen Codeänderungen zu vernachlässigen. Wenn die Migration jedoch von einer Nicht-MySQL-Engine erfolgt, können Schema- und Codeänderungen erforderlich werden. Bei der Einschätzung dieses Aufwands kann sich das weiter unten in diesem Whitepaper vorgestellte AWS Schema Conversion Tool als hilfreich erweisen.

Anwendungsverfügbarkeit während der Migration

Bei der Migration zu Amazon Aurora können Sie sich für einen Ansatz mit vorhersehbaren Ausfallzeiten oder für einen Ansatz mit nahezu keiner Ausfallzeit entscheiden. Für welchen Ansatz Sie sich entscheiden, hängt von der Größe Ihrer Datenbank und den Anforderungen an die Verfügbarkeit Ihrer Anwendungen ab. Unabhängig davon sollten Sie die Auswirkungen der Migration auf Ihre Anwendungen und Ihr Unternehmen berücksichtigen, bevor Sie eine Datenbankmigration in Angriff nehmen. In den folgenden Abschnitten werden beide Ansätze im Einzelnen erläutert.

Planen der Datenbankmigration

Im vorherigen Abschnitt wurden einige der wichtigsten Überlegungen erläutert, die während der Migration von Datenbanken zu Amazon Aurora zu berücksichtigen sind. Wenn Sie sich für Aurora als die optimale Lösung für Ihre Anwendung entschieden haben, müssen Sie im nächsten Schritt einen

vorläufigen Migrationsansatz wählen und einen Plan für die Datenbankmigration entwickeln.

Homogene Migration

Wenn es sich bei Ihrer Quelldatenbank um eine mit MySQL 5.6 kompatible Datenbank handelt (MySQL, MariaDB, Percona usw.), dann ist die Migration zu Aurora denkbar einfach.

Homogene Migration mit Ausfallzeit

Wenn Ihre Anwendung eine vorhersagbare Ausfallzeit außerhalb der Spitzenzeiten zulässt, ist eine Migration mit Ausfallzeit die einfachste und deshalb auch dringend empfohlene Vorgehensweise. Die meisten Datenbank-Migrationsprojekte fallen in diese Kategorie, da die meisten Anwendungen bereits ein eindeutig definiertes Wartungsfenster besitzen. Bei der Migration Ihrer Datenbank mit Ausfallzeit haben Sie folgende Möglichkeiten:

- **RDS-Snapshot-Migration** – Wenn Ihre Quelldatenbank auf Amazon RDS MySQL 5.6 ausgeführt wird, können Sie einfach einen Snapshot dieser Datenbank zu Amazon Aurora migrieren. Bei Migrationen mit Ausfallzeiten müssen Sie Ihre Anwendung während der Snapshot-Erstellung und Migration anhalten oder Schreibvorgänge in die Datenbank stoppen. Die für die Migration benötigte Zeit hängt in erster Linie von der Größe der Datenbank ab. Sie können sie vor der Produktionsmigration bestimmen, indem Sie eine Testmigration durchführen. Die Option der Snapshot-Migration wird im Abschnitt RDS-Snapshot-Migration erläutert. Eine Migration mit nahezu keiner Ausfallzeit ist möglich, wenn Sie eine Snapshot-Migration und eine Binlog-Replikation durchführen, wie dies im nächsten Abschnitt beschrieben wird.
- **Migration mit MySQL-eigenen Tools.** Sie können die Migration Ihrer Daten und Ihres Schemas zu Aurora mit MySQL-eigenen Tools vornehmen. Dies bietet sich an, wenn Sie mehr Kontrolle über die Datenbankmigration haben wollen, sich mit der Verwendung von MySQL-Tools auskennen und andere Migrationsmethoden für Ihren Anwendungsfall nicht so gut geeignet sind. Bewährte Methoden für diese Option finden Sie im Whitepaper Amazon RDS for Aurora Export/Import Performance Best Practices.[3]
- **Migration mit AWS Database Migration Service (AWS DMS).** Die einmalige Migration mit AWS DMS ist ein weiteres Tool, mit dem Sie Ihre

Quelldatenbank zu Amazon Aurora verschieben können. Bevor Sie die Daten mit AWS DMS verschieben können, müssen Sie das Datenbankschema unter Verwendung von MySQL-eigenen Tools von der Quelle zum Ziel kopieren. Eine schrittweise Beschreibung dieses Vorgangs finden Sie im Abschnitt Migrieren von Daten. Wenn Sie noch keine Erfahrung mit MySQL-eigenen Tools haben, ist AWS-DMS eine gute Wahl für Sie.

Homogene Migration mit nahezu keiner Ausfallzeit

Manche Szenarien erfordern bei Ihnen unter Umständen eine Datenbankmigration zu Aurora mit minimaler Ausfallzeit. Hierzu zwei Beispiele:

- Ihre Datenbank ist relativ umfangreich und die Ausfallzeit für die Migration überschreitet das Wartungsfenster für die Anwendung.
- Sie möchten Quell- und Zieldatenbanken für Testzwecke parallel ausführen.

In diesen Fällen können Sie Änderungen aus Ihrer MySQL-Quelldatenbank in Echtzeit in Aurora replizieren. Hierbei haben Sie die Wahl zwischen mehreren Optionen:

- **Migration mit nahezu keiner Ausfallzeit unter Verwendung von MySQL-Binlog-Replikation.** Amazon Aurora unterstützt die herkömmliche MySQL-Binlog-Replikation. Wenn Sie eine MySQL-Datenbank verwenden, dürften Sie bereits mit dem klassischen Binlog-Replikations-Setup vertraut sein. Wenn dies der Fall ist und Sie mehr Kontrolle über den Migrationsprozess wünschen, dann bietet Ihnen der Einsatz systemeigener Tools in Kombination mit der Binlog-Replikation zum einmaligen Laden der Datenbank einen vertrauten Migrationspfad zu Aurora.
- **Migration mit nahezu keiner Ausfallzeit unter Verwendung von AWS Database Migration Service (AWS DMS).** Neben der einmaligen Migration unterstützt AWS DMS auch Echtzeit-Datenreplikation mit Change Data Capture (CDC, Erfassung von Datenänderungen) von der Quelle bis zum Ziel. AWS DMS nimmt Ihnen die komplexen Aufgaben im Zusammenhang mit der Erstkopie der Daten, der Einrichtung von Replikations-Instances und der Überwachung der Replikation ab. Nach Abschluss der ersten Datenbankmigration bleibt die Zieldatenbank so lange mit der Quelldatenbank synchronisiert, wie Sie es wünschen. Sollten

Sie nicht mit der Binlog-Replikation vertraut sein, ist AWS DMS die zweitbeste Option für eine homogene Migration zu Amazon Aurora mit nahezu keiner Ausfallzeit. Weitere Informationen finden Sie im Abschnitt Einführung und Allgemeines zu AWS DMS.

- **Migration mit nahezu keiner Ausfallzeit unter Verwendung von RDS-Snapshot-Migration und MySQL-Binlog-Replikation.** Wenn Ihre Quelldatenbank auf Amazon RDS MySQL 5.6 ausgeführt wird, können Sie einen Snapshot dieser Datenbank zu Amazon Aurora migrieren. Im Anschluss an die Snapshot-Migration starten Sie dann die Binlog-Replikation zwischen Quelle und Ziel. Weitere Informationen zu dieser Migrationsoption finden Sie unter Replication with Amazon Aurora im *Amazon RDS User Guide*.[4]

Heterogene Migration

Wenn Sie vorhaben, eine nicht mit MySQL kompatible Datenbank (Oracle, SQL Server, PostgreSQL usw.) zu Amazon Aurora zu migrieren, stehen Ihnen mehrere Optionen zur Verfügung, mit denen Sie dies schnell und einfach erreichen können.

Schemamigration

Die Schemamigration von einer nicht mit MySQL kompatiblen Datenbank zu Amazon Aurora kann mit dem AWS Schema Conversion Tool erfolgen. Dieses Tool ist eine Desktop-Anwendung, mit der Sie Ihr Datenbankschema von einer Oracle-, Microsoft SQL Server- oder PostgreSQL-Datenbank in eine Amazon RDS MySQL-DB-Instance oder einen Amazon Aurora-DB-Cluster konvertieren können. In Fällen, in denen das Schema aus Ihrer Quelldatenbank nicht automatisch und vollständig konvertiert werden kann, bietet das AWS Schema Conversion Tool Anleitungen zur Erstellung des entsprechenden Schemas in Ihrer Amazon RDS-Zieldatenbank. Weitere Informationen finden Sie im Abschnitt Migrieren des Datenbankschemas.

Datenmigration

AWS Database Migration Service (AWS DMS) unterstützt neben homogenen Migrationen mit nahezu keiner Ausfallzeit auch eine kontinuierliche Datenreplikation über heterogene Datenbanken hinweg. Dieser Service ist daher eine bevorzugte Option zum Verschieben der Quelldatenbank in die Zieldatenbank, sowohl für Migrationen mit Ausfallzeit und Migrationen mit

nahezu keiner Ausfallzeit. Sobald die Migration gestartet wurde, verwaltet AWS DMS alle Aspekte der Migration wie beispielsweise die Datentyptransformation, Komprimierung und (schnellere) parallele Datenübertragung. Dabei wird gleichzeitig sichergestellt, dass Datenänderungen in der Quelldatenbank, die während der Migration erfolgen, automatisch in der Zieldatenbank repliziert werden.

Neben der Verwendung von AWS DMS können Sie auch verschiedene Drittanbieter-Tools wie Attunity Replicate, Tungsten Replicator, Oracle Golden Gate usw. für die Migration Ihrer Daten zu Amazon Aurora verwenden. Unabhängig davon, für welches Tool Sie sich entscheiden, sollten Sie in jedem Fall die Leistung und Lizenzierungskosten sorgfältig prüfen, bevor Sie eine endgültige Entscheidung hinsichtlich des Toolsets für die Migration treffen.

Migrieren umfangreicher Datenbanken zu Amazon Aurora

Die Migration von großen Datensätzen stellt bei jeder Datenbankmigration eine ganz einzigartige Herausforderung dar. Bei vielen erfolgreichen Migrationsprojekten für große Datenbanken kommt eine Kombination aus den folgenden Strategien zum Einsatz:

- **Migration mit kontinuierlicher Replikation.** Bei großen Datenbanken sind für die Datenverschiebung von der Quelle zum Ziel in der Regel längere Ausfallzeiten erforderlich. Zur Reduzierung der Ausfallzeit können Sie zunächst Basisdaten von der Quelle auf das Ziel laden und dann die Replikation (mit MySQL-eigenen Tools, AWS DMS oder Drittanbieter-Tools) aktivieren, um die nachträglichen Änderungen zu übertragen.
- **Kopieren Sie statische Tabellen zuerst.** Wenn Ihre Datenbank auf große statische Tabellen mit Referenzdaten angewiesen ist, können Sie diese großen Tabellen in die Zieldatenbank kopieren, bevor Sie Ihr aktives Dataset migrieren. Sie können mit AWS DMS Tabellen selektiv kopieren oder diese Tabellen manuell exportieren und importieren.
- **Migration in mehreren Phasen.** Die Migration großer Datenbanken mit Tausenden von Tabellen kann in mehrere Phasen unterteilt werden. Sie können beispielsweise jedes Wochenende eine Reihe von Tabellen ohne Cross Joins (Kreuzverknüpfungen) verschieben, bis die Quelldatenbank vollständig zur Zieldatenbank migriert ist. Allerdings müssen Sie hierzu Änderungen in Ihrer Anwendung vornehmen, damit Sie eine gleichzeitige

Verbindung zu zwei Datenbanken herstellen können, während Ihr Dataset auf zwei verschiedenen Knoten verteilt ist. Dies ist zwar kein gängiges Migrationsmuster, aber immerhin eine Option.

- **Datenbankbereinigung.** Viele große Datenbanken enthalten Daten und Tabellen, die nicht genutzt werden. In vielen Fällen belassen Entwickler und Datenbankadministratoren Sicherungskopien von Tabellen in derselben Datenbank oder sie vergessen einfach, nicht verwendete Tabellen zu löschen. Ein Datenbank-Migrationsprojekt bietet ungeachtet von der Motivation eine Chance, die vorhandene Datenbank vor der Migration zu bereinigen. Wenn einige Tabellen nicht genutzt werden, können Sie diese entweder löschen oder in einer anderen Datenbank archivieren. Sie können auch alte Daten aus großen Tabellen löschen oder diese Daten in Flat Files archivieren.

Partitions- und Shard-Konsolidierung auf Amazon Aurora

Wenn Sie mehrere Shards oder funktionale Partitionen Ihrer Datenbank ausführen, um eine hohe Leistung zu erreichen, haben Sie die Möglichkeit, diese Partitionen oder Shards in einer einzelnen Aurora-Datenbank zu konsolidieren. Eine einzelne Amazon Aurora-Instance ist auf bis zu 64 TB skalierbar und bietet Unterstützung für Tausende von Tabellen und für eine wesentlich höhere Anzahl von Lese- und Schreibvorgängen als eine Standard-MySQL-Datenbank. Die Konsolidierung dieser Partitionen auf einer einzelnen Aurora-Instance sorgt nicht nur für eine Reduzierung der Gesamtbetriebskosten und eine Vereinfachung der Datenbankverwaltung, sondern verbessert auch die Leistung von partitionsübergreifenden Abfragen erheblich.

- **Funktionale Partitionen.** Bei einer funktionalen Partitionierung werden unterschiedlichen Aufgaben unterschiedliche Knoten zugewiesen. So können Sie beispielsweise in einer E-Commerce-Anwendung einen Datenbankknoten für die Bereitstellung von Produktkatalogdaten und einen anderen Datenbankknoten für die Erfassung und Verarbeitung von Bestellungen definieren. Infolgedessen haben diese Partitionen in der Regel getrennte, nicht überlappende Schemas.

  Konsolidierungsstrategie. Migrieren Sie jede funktionale Partition als eigenständiges Schema zu Ihrer Aurora-Ziel-Instance. Wenn Ihre Quelldatenbank MySQL-kompatibel ist, nutzen Sie MySQL-eigene Tools

für die Schemamigration und dann AWS DMS für die Datenmigration, entweder in einem einmaligen Vorgang oder kontinuierlich mit Replikation. Wenn Ihre Quelldatenbank nicht mit MySQL kompatibel ist, verwenden Sie das AWS Schema Conversion Tool für die Migration der Schemas zu Aurora und dann AWS DMS für einen einmaligen Ladevorgang oder für die kontinuierliche Datenreplikation.

- **Daten-Shards.** Wenn Sie dasselbe Schema mit unterschiedlichen Datensätzen über mehrere Knoten hinweg verwenden, nutzen Sie das sogenannte Datenbank-Sharding. Beispielsweise können Sie bei einem Blog-Service mit hohem Daten-Traffic Benutzeraktivitäten und Daten über mehrere Datenbank-Shards verteilen und dabei dasselbe Tabellenschema verwenden.

  Konsolidierungsstrategie. Da alle Shards dasselbe Datenbankschema verwenden, müssen Sie das Zielschema nur einmal erstellen. Wenn Sie mit einer MySQL-kompatiblen Datenbank arbeiten, verwenden Sie systemeigene Tools, um das Datenbankschema zu Aurora zu migrieren. Wenn Sie eine Nicht-MySQL-Datenbank verwenden, migrieren Sie das Datenbankschema mit dem AWS Schema Conversion Tool zu Aurora. Sobald das Datenbankschema migriert wurde, sollten Sie keine Schreibvorgänge mehr auf den Datenbank-Shards ausführen und systemeigene Tools einsetzen oder einen einmaligen Datenladevorgang mit AWS DMS durchführen, um die einzelnen Shards zu Aurora zu migrieren. Wenn es nicht möglich ist, Schreibvorgänge in die Anwendung für einen längeren Zeitraum auszusetzen, können Sie AWS DMS mit Replikation unter Umständen weiter zu nutzen, aber nur nach ordnungsgemäßer Planung und Prüfung.

Migrationsoptionen im Überblick

Typ der Quelldatenbank	Migration mit Ausfallzeit	Migration mit nahezu keiner Ausfallzeit
Amazon RDS MySQL	Option 1: RDS-Snapshot-Migration	Option 1: Migration mit systemeigenen Tools + Binlog-Replikation
	Option 2: Manuelle Migration mit systemeigenen Tools*	Option 2: RDS-Snapshot-Migration + Binlog-Replikation
	Option 3: Schemamigration mit systemeigenen Tools + AWS DMS zum Laden von Daten	Option 3: Schemamigration mit systemeigenen Tools + AWS DMS zum Verschieben von Daten

Typ der Quelldatenbank	Migration mit Ausfallzeit	Migration mit nahezu keiner Ausfallzeit
MySQL Amazon EC2 oder lokal	Option 1: Migration mit systemeigenen Tools Option 2: Schemamigration mit systemeigenen Tools + AWS DMS für das Laden von Daten	Option 1: Migration mit systemeigenen Tools + Binlog-Replikation Option 2: Schemamigration mit integrierten Tools + AWS DMS zum Verschieben von Daten
Oracle/SQL Server	Option 1: AWS Schema Conversion Tool + AWS DMS (empfohlen) Option 2: Schemakonvertierung manuell oder mit Drittanbieter-Tool + Laden von Daten im Ziel manuell oder mit Drittanbieter-Tool	Option 1: AWS Schema Conversion Tool + AWS DMS (empfohlen) Option 2: Schemakonvertierung manuell oder mit Drittanbieter-Tool + Laden von Daten im Ziel manuell oder mit Drittanbieter-Tool für die Replikation.
Andere Nicht-MySQL-Datenbanken	Option: Schemakonvertierung manuell oder mit Drittanbieter-Tool + Laden von Daten im Ziel manuell oder mit Drittanbieter-Tool	Option: Schemakonvertierung manuell oder mit Drittanbieter-Tool + Laden von Daten im Ziel manuell oder mit Drittanbieter-Tool + Drittanbieter-Tool für die Replikation (GoldenGate usw.)

* MySQL-eigene Tools: mysqldump, SELECT INTO OUTFILE, Drittanbieter-Tools wie mydumper/myloader

RDS-Snapshot-Migration

Damit Sie die RDS-Snapshot-Migration für den Umstieg auf Aurora verwenden können, muss Ihre MySQL-Datenbank unter Amazon RDS MySQL 5.6 ausgeführt werden und Sie müssen einen RDS-Snapshot der Datenbank erstellen. Diese Migrationsmethode eignet sich nicht für lokale Datenbanken oder Datenbanken, die auf Amazon Elastic Compute Cloud (Amazon EC2) ausgeführt werden. Wenn Sie Ihre Amazon RDS MySQL-Datenbank auf einer älteren Version als 5.6 ausführen, müssen Sie zunächst ein Upgrade auf 5.6 durchführen.

Der größte Vorteil dieser Migrationsmethode besteht darin, dass dies die einfachste Methode mit der geringsten Anzahl von Schritten ist. Vor allem deshalb, weil sie die Migration über alle Schemaobjekte, sekundären Indizes und gespeicherten Prozeduren sowie alle Datenbankdaten hinweg ermöglicht.

Während der Snaphot-Migration ohne Binlog-Replikation muss sich Ihre Quelldatenbank entweder im Offline- oder im schreibgeschützten Modus befinden (damit während der Migration keine Änderungen an der Quelldatenbank vorgenommen werden). Zur Schätzung der Ausfallzeiten können Sie einfach einen vorhandenen Snapshot Ihrer Datenbank für eine Testmigration

verwenden. Wenn die Migrationszeit Ihre Anforderungen an die Ausfallzeit erfüllt, ist dies möglicherweise die beste Methode für Sie. In manchen Fällen kann die Migration mit AWS DMS oder mit systemeigenen Migrations-Tools schneller als die Snapshot-Migration sein.

Wenn längere Ausfallzeiten für Sie nicht akzeptabel sind, können Sie dennoch eine Migration mit nahezu keiner Ausfallzeit erreichen, indem Sie zunächst einen Snapshot der RDS-Datenbank zu Amazon Aurora migrieren und gleichzeitig weitere Aktualisierungen der Quelldatenbank zulassen. Anschließend bringen Sie die Datenbank mit einer MySQL-Binlog-Replikation von MySQL zu Aurora auf den neuesten Stand.

Sie können die Migration entweder manuell oder per automatisiertem DB-Snapshot durchführen. Hierzu führen Sie die folgenden allgemeinen Schritte aus:

1. Bestimmen Sie den für die Migration Ihrer Amazon RDS MySQL 5.6-Instance zu einem Aurora-DB-Cluster benötigten Speicherplatz. Weitere Informationen hierzu finden Sie im nächsten Abschnitt.
2. Verwenden Sie die Amazon RDS-Konsole, um den Snapshot in der Region zu erstellen, in der sich die Amazon RDS MySQL 5.6-Instance befindet.
3. Verwenden Sie die Konsolenfunktion **Migrate Database**, um einen Amazon Aurora-DB-Cluster zu erstellen, der mit dem DB-Snapshot der ursprünglichen DB-Instance von MySQL 5.6 aufgefüllt wird.

Hinweis: Einige MyISAM-Tabellen lassen sich möglicherweise nicht fehlerfrei konvertieren und erfordern manuelle Änderungen. Der InnoDB-Engine lässt beispielsweise in einem zusammengesetzten Schlüssel kein AutoIncrement-Feld zu. Außerdem werden räumliche Indizes derzeit nicht unterstützt.

Schätzen des Speicherplatzbedarfs für die Snapshot-Migration

Wenn Sie einen Snapshot einer MySQL-DB-Instance zu einem Aurora-DB-Cluster migrieren möchten, verwendet Aurora ein Amazon Elastic Block Store (Amazon EBS)-Volume, um die im Snaphot enthaltenen Daten vor der Migration zu formatieren. In manchen Fällen wird zusätzlicher Speicherplatz zum Formatieren der Daten für die Migration benötigt. Während der Migration können zwei Funktionen zu Platzproblemen führen: MyISAM-Tabellen und die Verwendung der Option `ROW_FORMAT=COMPRESSED`. Wenn Sie diese Funktionen in Ihrer Quelldatenbank nicht verwenden, können Sie diesen Abschnitt überspringen, da Sie keine Speicherplatzprobleme haben sollten. Bei der

Migration werden MyISAM-Tabellen in InnoDB-Tabellen konvertiert und alle komprimierten Tabellen werden dekomprimiert. Daher muss genügend Speicherplatz für die zusätzlichen Kopien dieser Tabellen vorhanden sein.

Die Größe des Migrations-Volumes richtet sich nach dem Speicherplatz, den Sie der Quelldatenbank zugewiesen haben, die als Basis für den Snapshot diente. Falls also die vorliegenden MyISAM- oder komprimierten Tabellen nur einen kleinen Prozentsatz der Gesamtgröße der Datenbank ausmachen und in der ursprünglichen Datenbank noch Platz verfügbar ist, sollte die Migration erfolgreich und ohne Speicherplatzprobleme verlaufen. Sollte die ursprüngliche Datenbank jedoch nicht genügend Platz zum Speichern einer Kopie der konvertierten MyISAM-Tabellen und einer weiteren (nicht komprimierten) Kopie der komprimierten Tabellen aufweisen, dann reicht der Speicherplatz des Migrations-Volumes nicht aus. In diesem Fall müssen Sie den Speicherplatz, den Sie der Amazon RDS MySQL-Quelldatenbank zugewiesen haben, so weit vergrößern, dass genügend Platz für die zusätzlichen Kopien dieser Tabellen vorhanden ist. Anschließend erstellen Sie einen neuen Snapshot der Datenbank und migrieren diesen.

Beachten Sie bei der Migration von Daten in Ihren DB-Cluster die folgenden Richtlinien und Einschränkungen:

- Amazon Aurora unterstützt zwar bis zu 64 TB Speicher, aber die Migration eines Snapshots in einen Aurora-DB-Cluster ist durch die Größe des Amazon EBS-Volumes des Snapshots und daher auf eine Größe von maximal 6 TB beschränkt.
- Nicht-MyISAM-Tabellen in der Quelldatenbank können eine Größe von bis zu 6 TB aufweisen. Aufgrund zusätzlicher Speicherplatzanforderungen während der Konvertierung müssen Sie sicherstellen, dass keine der MyISAM-Tabellen und der komprimierten Tabellen, die Sie von Ihrer MySQL-DB-Instance migrieren, eine Größe von mehr als 3 TB aufweist.

Gegebenenfalls sollten Sie Ihr Datenbankschema ändern (MyISAM-Tabellen in InnoDB-Tabellen konvertieren und `ROW_FORMAT=COMPRESSED` entfernen), bevor Sie zu Amazon Aurora migrieren. Dies kann sich in folgenden Fällen als hilfreich erweisen:

- Sie möchten die Migration beschleunigen.
- Sie wissen nicht genau, wie viel Speicherplatz Sie bereitstellen müssen.

- Sie haben versucht, Ihre Daten zu migrieren und die Migration ist fehlgeschlagen, weil nicht genügend Speicherplatz bereitgestellt wurde.

Stellen Sie sicher, dass Sie diese Änderungen nicht in Ihrer Amazon RDS MySQL-Produktionsdatenbank vornehmen, sondern in einer Datenbank-Instance, die Sie von Ihrem Produktions-Snapshot wiederhergestellt haben. Weitere Informationen hierzu finden Sie unter Reducing the Amount of Space Required to Migrate Data into Amazon Aurora im *Amazon RDS User Guide*.[5]

Migrieren eines DB-Snapshots mit der Konsole

Sie können einen DB-Snapshot einer Amazon RDS MySQL-DB-Instance migrieren, um einen Amazon Aurora DB-Cluster zu erstellen. Der neue DB-Cluster wird mit den Daten aus der ursprünglichen Amazon RDS MySQL-DB-Instance aufgefüllt. Der DB-Snapshot muss von einer RDS DB-Instance mit MySQL 5.6 erstellt werden und darf nicht verschlüsselt sein. Weitere Informationen zum Erstellen eines DB-Snapshots finden Sie unter Creating a DB-Snapshot im *Amazon RDS User Guide*.[6]

Wenn sich der DB-Snapshot nicht in der Region befindet, die Sie für Ihren Aurora-DB-Cluster vorgesehen haben, kopieren Sie den DB-Snaphot über die Amazon RDS-Konsole in diese Region. Weitere Informationen zum Kopieren eines DB-Snapshots finden Sie unter Copying a DB Snapshot im *Amazon RDS User Guide*.[7]

Migrieren Sie einen MySQL 5.6-DB-Snapshot wie folgt mit der Konsole:

1. Melden Sie sich bei AWS Management Console an und öffnen Sie die Amazon RDS-Konsole unter https://console.aws.amazon.com/rds/.
2. Wählen Sie **Snapshots** aus.
3. Wählen Sie auf der Seite **Snapshots** den Amazon RDS MySQL 5.6-Snapshot aus, den Sie in einen Aurora-DB-Cluster migrieren möchten.
4. Wählen Sie **Migrate Database** aus.
5. Geben Sie auf der Seite **Migrate Database** die Werte für Ihre Umgebung und Ihre Vorgaben für die Verarbeitung wie in der folgenden Abbildung dargestellt an. Eine Beschreibung dieser Optionen finden Sie unter Migrating a DB Snapshot by Using the Console im *Amazon RDS User Guide*.[8]

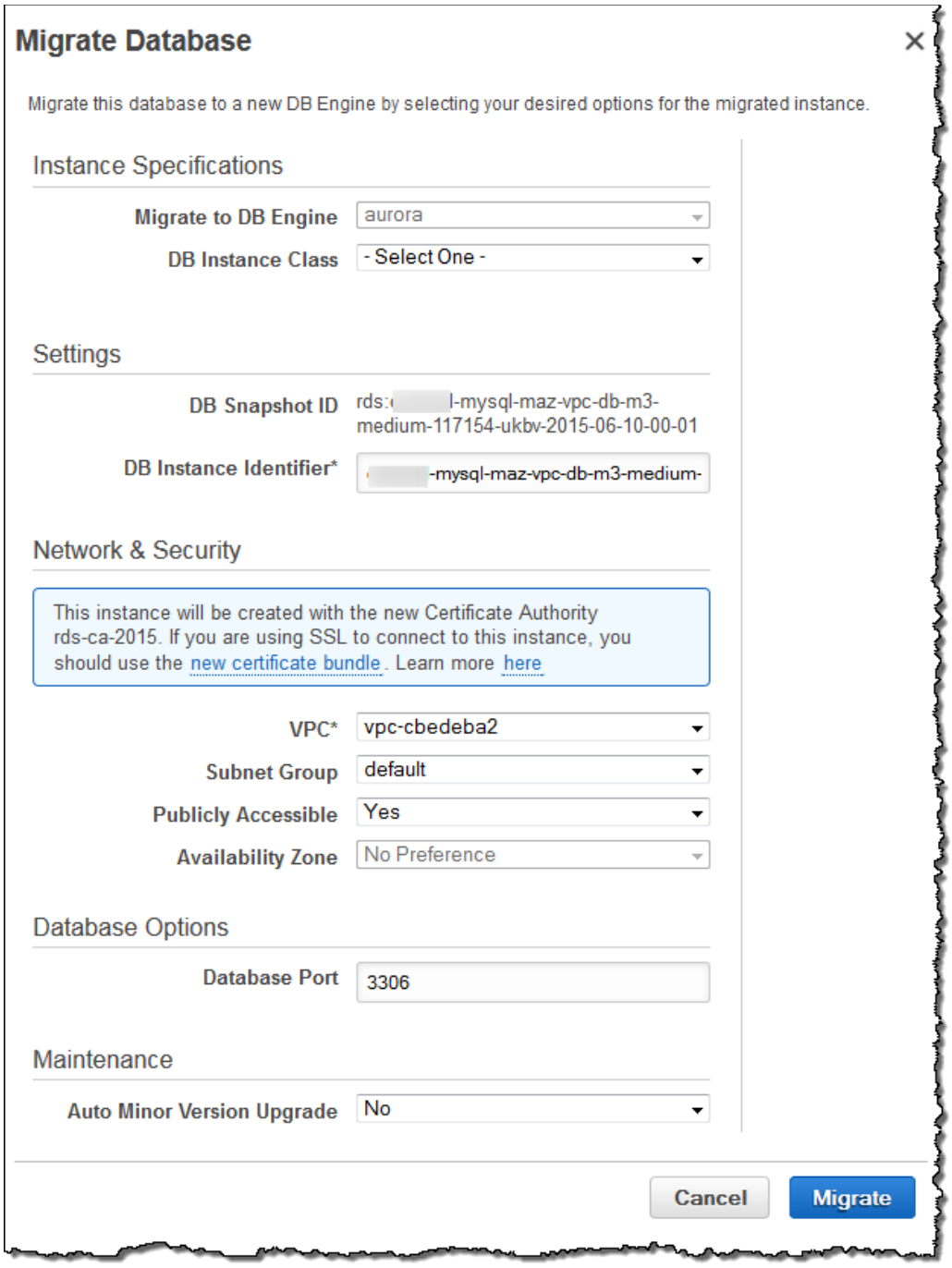

Abbildung 2: Snapshot-Migration

6. Klicken Sie auf **Migrate**, um Ihren DB-Snapshot zu migrieren.

Klicken Sie in der Liste der Instances auf das entsprechende Pfeilsymbol, um die DB-Cluster-Details anzuzeigen und den Fortschritt der Migration zu überwachen. Dieser Detailbereich zeigt den Cluster-Endpunkt an, der zum Herstellen einer Verbindung zur primären Instance des DB-Clusters verwendet wird. Weitere

Informationen zum Herstellen einer Verbindung zu einem Amazon Aurora-DB-Cluster finden Sie unter Connecting to an Amazon Aurora DB Cluster im *Amazon RDS User Guide*.[9]

Migrieren des Datenbankschemas

Bei der RDS DB Snapshot-Migration werden sowohl das vollständige Schema als auch die Daten auf die neue Aurora-Instance migriert. Sollte der Speicherort Ihrer Quelldatenbank oder die Anforderungen an die Anwendungsverfügbarkeit den Einsatz einer RDS Snapshot-Migration nicht zulassen, müssen Sie zunächst das Datenbankschema von der Quelldatenbank zur Zieldatenbank migrieren. Erst danach können Sie die eigentlichen Daten verschieben. Ein Datenbankschema bildet das Grundgerüst, das die logische Ansicht der gesamten Datenbank widerspiegelt. Es umfasst in der Regel Folgendes:

- Objekte des Datenbankspeichers: Tabellen, Spalten, Einschränkungen, Indizes, Sequenzen, benutzerdefinierte Typen und Datentypen
- Objekte des Datenbankcodes: Funktionen, Verfahren, Pakete, Auslöser, Ansichten, materialisierte Ansichten, Ereignisse, SQL-Skalarfunktionen, SQL-Inlinefunktionen, SQL-Tabellenfunktionen, Attribute, Variablen, Konstanten, Tabellentypen, öffentliche Typen, private Typen, Cursor, Ausnahmen, Parameter und andere Objekte

In den meisten Fällen bleibt das Datenbankschema relativ statisch, daher können Sie die Migration des Datenbankschemas ohne Ausfallzeit durchführen. Das Schema kann während des laufenden Datenbankbetriebs aus Ihrer Quelldatenbank extrahiert werden, ohne dass sich dies auf die Leistung auswirkt. Wenn Ihre Anwendung oder Ihre Entwickler häufig Änderungen am Datenbankschema vornehmen, stellen Sie sicher, dass Änderungen während des Migrationsvorgangs unterbleiben oder aber bei der Schemamigration berücksichtigt werden.

In den folgenden Abschnitten werden Methoden für die Migration des Datenbankschemas erörtert, die Sie je nach Art Ihrer Quelldatenbank verwenden können. Als Voraussetzung für die Schemamigration muss eine Aurora-Zieldatenbank erstellt und verfügbar sein.

Homogene Schemamigration

Wenn Ihre Quelldatenbank MySQL 5.6-kompatibel ist und unter Amazon RDS, Amazon EC2 oder außerhalb von AWS ausgeführt wird, können Sie MySQL-eigene Tools für den Export und Import des Schemas verwenden.

- **Exportieren des Datenbankschemas.** Sie können das Client-Dienstprogramm mysqldump für den Export des Datenbankschemas verwenden. Um dieses Dienstprogramm ausführen zu können, müssen Sie eine Verbindung zu Ihrer Quelldatenbank herstellen und die Ausgabe des Befehls `mysqldump` in eine Flat File umleiten. Mit der Option `–no-data` stellen Sie sicher, dass nur das Datenbankschema exportiert wird, aber nicht die eigentlichen Tabellendaten. Eine vollständige Referenz zum Befehl `mysqldump` finden Sie unter mysqldump—A Database Backup Program.[10]

```
mysqldump –u source_db_username –p --no-data --routines --
triggers –databases source_db_name > DBSchema.sql
```

- **Importieren des Datenbankschemas in Aurora.** Zum Importieren des Schemas in Ihre Aurora-Instance stellen Sie von einem MySQL-Befehlszeilen-Client (oder einem entsprechenden Windows-Client) aus eine Verbindung zu Ihrer Aurora-Datenbank her und leiten den Inhalt der Exportdatei in MySQL.

```
mysql –h aurora-cluster-endpoint -u username -p <
DBSchema.sql
```

Beachten Sie Folgendes:

- Wenn Ihre Quelldatenbank gespeicherte Prozeduren, Auslöser und Ansichten enthält, müssen Sie die Syntax `DEFINER` aus Ihrer Dumpdatei entfernen. Einen einfachen Perl-Befehl, den Sie hierfür verwenden können, finden Sie nachstehend. Dadurch werden alle Auslöser, Ansichten und gespeicherten Prozeduren für den aktuell verbundenen Benutzer als `DEFINER` erstellt. Bewerten Sie in jedem Fall die Auswirkungen, die dies auf die Sicherheit haben könnte.

```
$perl -pe 's/\sDEFINER=`[^`]+`@`[^`]+`//' < DBSchema.sql >
DBSchemaWithoutDEFINER.sql
```

- Amazon Aurora unterstützt nur InnoDB-Tabellen. Wenn Ihre Quelldatenbank MyISAM-Tabellen aufweist, ändert Aurora bei der Ausführung des Befehls `CREATE TABLE` die Engine automatisch in InnoDB.
- Amazon Aurora bietet keine Unterstützung für komprimierte Tabellen (d. h. Tabellen, die mit `ROW_FORMAT=COMPRESSED` erstellt wurden). Wenn Ihre Quelldatenbank komprimierte Tabellen aufweist, ändert Aurora bei der Ausführung des Befehls `CREATE TABLE` die Einstellung für `ROW_FORMAT` automatisch in `COMPACT`.

Sobald Sie das Schema erfolgreich aus Ihrer MySQL 5.6-konformen Quelldatenbank in Amazon Aurora importiert haben, kopieren Sie als nächsten Schritt die eigentlichen Daten aus der Quell- in die Zieldatenbank. Weitere Informationen finden Sie weiter unten in diesem Whitepaper unter Einführung und Allgemeines zu AWS DMS.

Heterogene Schemamigration

Wenn Ihre Quelldatenbank nicht MySQL-kompatibel ist, müssen Sie Ihr Schema in ein Format konvertieren, das mit Amazon Aurora kompatibel ist. Die Schemakonvertierung von einer Datenbank-Engine zu einer anderen ist keine einfache Aufgabe und kann bedeuten, dass Sie bestimmte Teile Ihres Datenbank- und Anwendungscodes umschreiben müssen. Sie haben zwei Optionen für die Umwandlung und die Migration Ihres Schemas zu Amazon Aurora:

- **AWS Schema Conversion Tool.** Das AWS Schema Conversion Tool vereinfacht heterogene Datenbankmigrationen durch eine automatische Konvertierung des gesamten Datenbankschemas und eines Großteils des in Ansichten, gespeicherten Prozeduren und Funktionen verwendeten benutzerdefinierten Codes der Quelldatenbank in ein Format, das kompatibel mit der Zieldatenbank ist.[11] Code, der nicht automatisch konvertiert werden kann, wird eindeutig gekennzeichnet, damit Sie ihn konvertieren können. Sie können dieses Tool verwenden, um Ihre auf Oracle oder Microsoft SQL Server basierenden Quelldatenbanken in eine Amazon Aurora-, MySQL- oder PostgreSQL-Zieldatenbank in Amazon RDS oder Amazon EC2 zu konvertieren. Zur Konvertierung Ihres Oracle-,

SQL Server- oder PostgreSQL-Schemas in ein Aurora-kompatibles Format sollten Sie vorzugsweise das AWS Schema Conversion Tool verwenden.

- **Manuelle Schemamigration und Drittanbieter-Tools.** Wenn es sich bei Ihrer Quelldatenbank nicht um eine Oracle-, SQL Server- oder PostgreSQL-Datenbank handelt, können Sie Ihr Quelldatenbankschema entweder manuell zu Aurora migrieren oder Drittanbieter-Tools für die Migration des Schemas in ein Format verwenden, das mit MySQL 5.6 kompatibel ist. Die manuelle Schemamigration kann sich je nach Größe und Komplexität Ihres Quellschemas als ziemlich komplexer Vorgang erweisen. Angesichts der Kosteneinsparungen, Leistungsvorteile und anderen Verbesserungen, die Amazon Aurora mit sich bringt, lohnt sich jedoch die Schemakonvertierung in den meisten Fällen.

Schemamigration mit AWS Schema Conversion Tool

AWS Schema Conversion Tool bietet eine projektbasierte Benutzeroberfläche für die automatische Konvertierung des Datenbankschemas Ihrer Quelldatenbank in ein Format, das mit Amazon Aurora kompatibel ist. Bevor Sie die Produktionsmigration in Angriff nehmen, sollten Sie unbedingt mit dem AWS Schema Conversion Tool den Aufwand bewerten, der mit der Datenbankmigration und der Pilotmigration verbunden ist.

Die folgende Beschreibung führt Sie schrittweise durch die generelle Verwendung von AWS Schema Conversion Tool. Detaillierte Anweisungen finden Sie im Abschnitt Getting Started des *AWS Schema Conversion Tool User Guide*.[12]

1. Installieren Sie zunächst das Tool. AWS Schema Conversion Tool ist für Microsoft Windows, Mac OS X, Ubuntu Linux und Fedora Linux verfügbar.

   Detaillierte Anweisungen zum Herunterladen und Installieren finden Sie im Abschnitt Installing and Updating des Benutzerhandbuchs.[13] Wo Sie AWS Schema Conversion Tool installieren, ist wichtig. Das Tool muss für die Konvertierung und Anwendung eines Schemas eine direkte Verbindung zur Quell- und zur Zieldatenbank herstellen. Stellen Sie sicher, dass der Desktop, auf dem Sie das AWS Schema Conversion Tool installieren, über das Netzwerk mit der Quell- und Zieldatenbank verbunden ist.

2. Installieren Sie die JDBC-Treiber. Das AWS Schema Conversion Tool verwendet JDBC-Treiber für die Verbindung mit der Quell- und Zieldatenbank. Damit Sie dieses Tool verwenden können, müssen Sie diese JDBC-Treiber auf Ihren lokalen Desktop herunterladen. Anweisungen zum

Herunterladen von Treibern finden Sie unter Required Database Drivers im *AWS Schema Conversion Tool User Guide*.[14] Anweisungen zum Einrichten des JDBC-Treibers für verschiedene Datenbank-Engines finden Sie außerdem im AWS-Forum für AWS Schema Conversion Tool.[15]

3. Erstellen Sie eine Zieldatenbank. Erstellen Sie eine Amazon Aurora-Zieldatenbank. Weitere Anweisungen zum Erstellen einer Amazon Aurora-Datenbank finden Sie unter Creating an Amazon Aurora DB Cluster im *Amazon RDS User Guide*.[16]
4. Öffnen Sie das AWS Schema Conversion Tool und starten Sie den **New Project Wizard**.

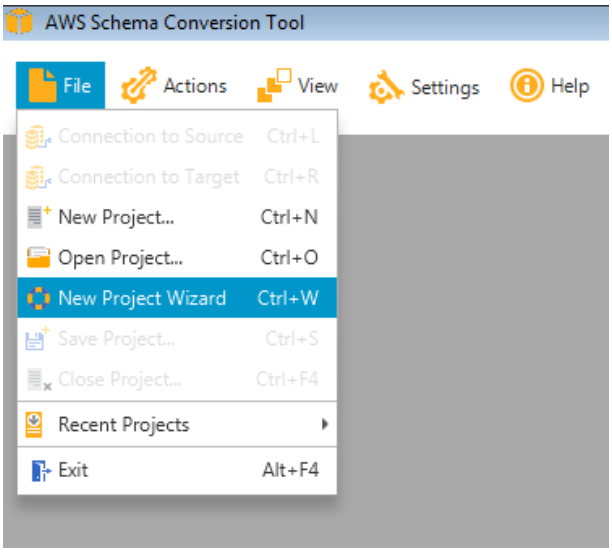

Abbildung 3: Erstellen eines neuen AWS Schema Conversion Tool-Projekts

5. Konfigurieren Sie die Quelldatenbank und testen Sie die Verbindung zwischen AWS Schema Conversion Tool und der Quelldatenbank. Damit dies funktioniert, muss Ihre Quelldatenbank von Ihrem Desktop aus erreichbar sein. Stellen Sie daher sicher, dass Ihre Netzwerk- und Firewall-Einstellungen entsprechend konfiguriert sind.

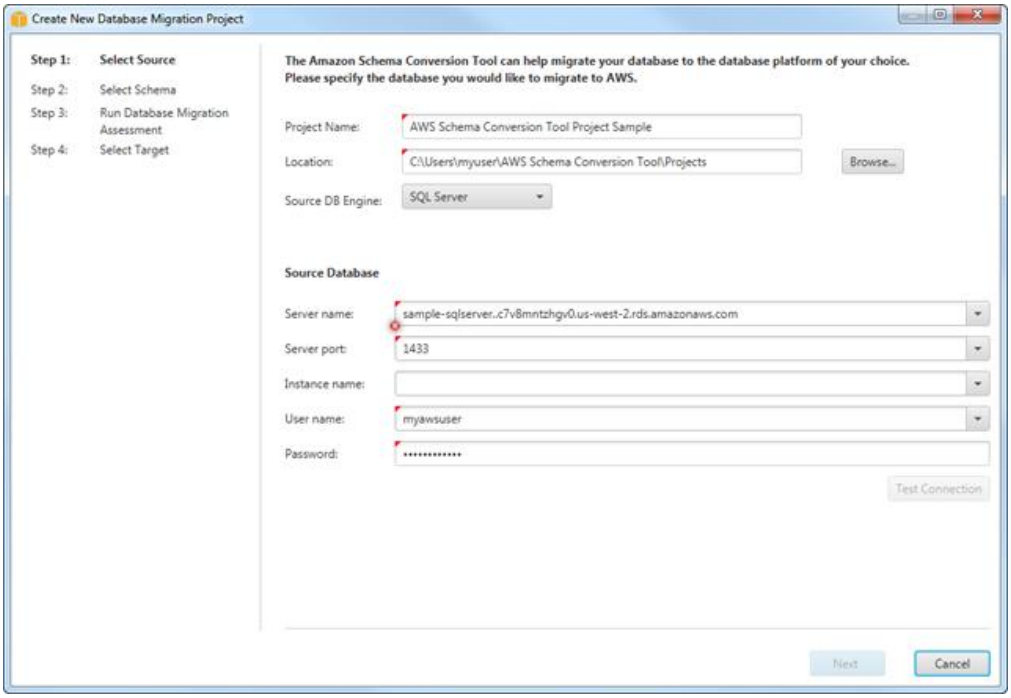

Abbildung 4: Assistent „Create New Database Migration Project“

6. Wählen Sie im nächsten Bildschirm das Schema der Quelldatenbank aus, die Sie in Amazon Aurora konvertieren möchten.

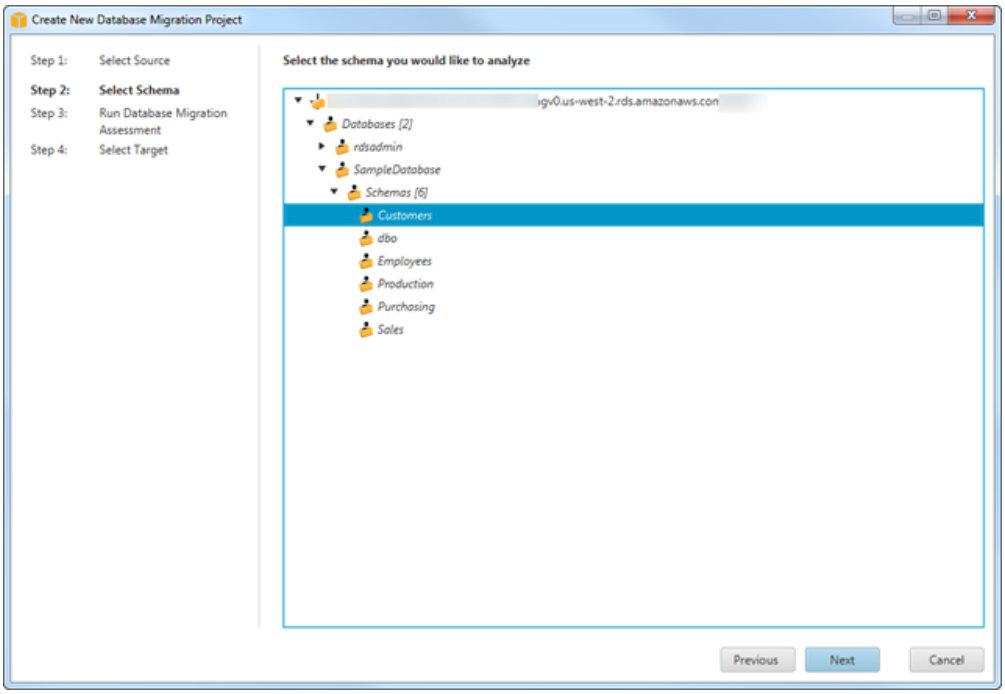

Abbildung 5: Auswählen des Schemas im Migrationsassistenten

7. Führen Sie den Bericht zur Bewertung der Datenbankmigration aus. Dieser Bericht liefert wichtige Informationen zur Konvertierung des Schemas aus Ihrer Quelldatenbank in Ihre Amazon Aurora-Ziel-Instance. Er fasst sämtliche Schemakonvertierungsaufgaben zusammen und bietet detaillierte Aktionselemente für jene Teile des Schemas, die nicht automatisch in Aurora konvertiert werden können. Der Bericht enthält außerdem eine Schätzung des Aufwands, der damit verbunden ist, dass für nicht automatisch konvertierte Teile entsprechender Code für die Zieldatenbank geschrieben werden muss.

   Klicken Sie auf **Next**, um die Zieldatenbank zu konfigurieren. Sie können diesen Migrationsbericht später erneut anzeigen.

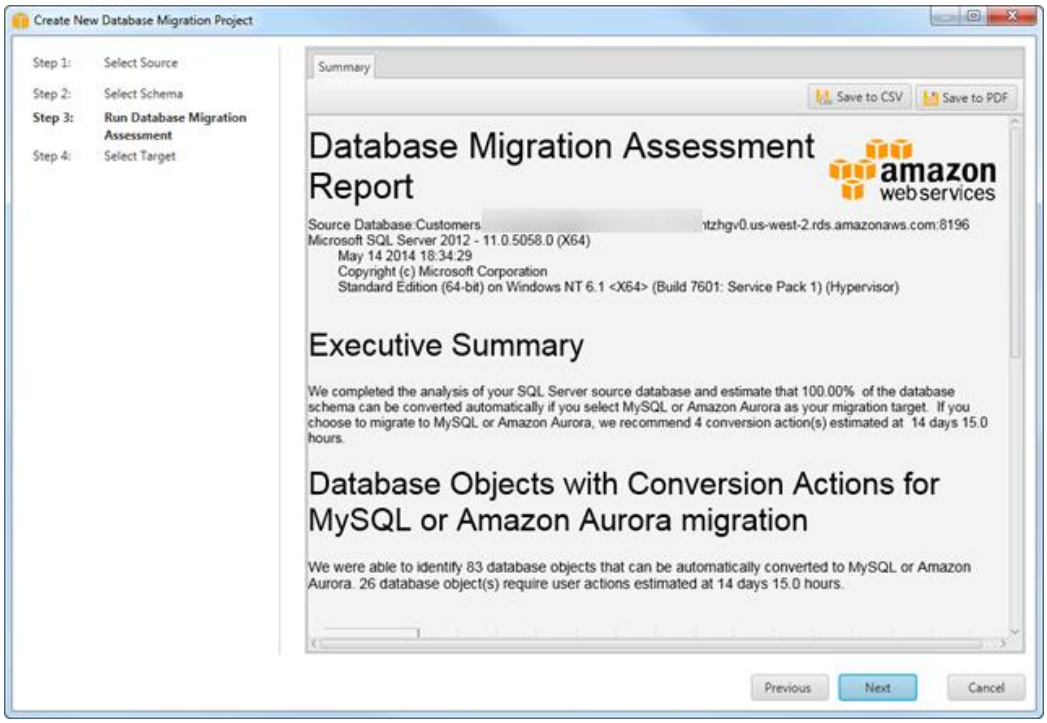

Abbildung 6: Migrationsbericht

8. Konfigurieren Sie die Amazon Aurora-Zieldatenbank und testen Sie die Konnektivität zwischen dem AWS Schema Conversion Tool und der Quelldatenbank. Damit dies funktioniert, muss Ihre Zieldatenbank von Ihrem Desktop aus erreichbar sein. Stellen Sie daher sicher, dass Ihre Netzwerk- und Firewall-Einstellungen entsprechend konfiguriert sind. Klicken Sie auf **Finish**, um zum Projektfenster zu wechseln.

9. Wenn das Projektfenster angezeigt wird, haben Sie bereits eine Verbindung zur Quell- und Zieldatenbank hergestellt und können jetzt den detaillierten Bewertungsbericht prüfen und das Schema migrieren.

10. Wählen Sie im linken Bereich mit dem Schema Ihrer Quelldatenbank ein Schemaobjekt aus, für das ein Bewertungsbericht erstellt werden soll. Klicken Sie mit der rechten Maustaste auf das Objekt, und wählen Sie dann **Create Report** aus.

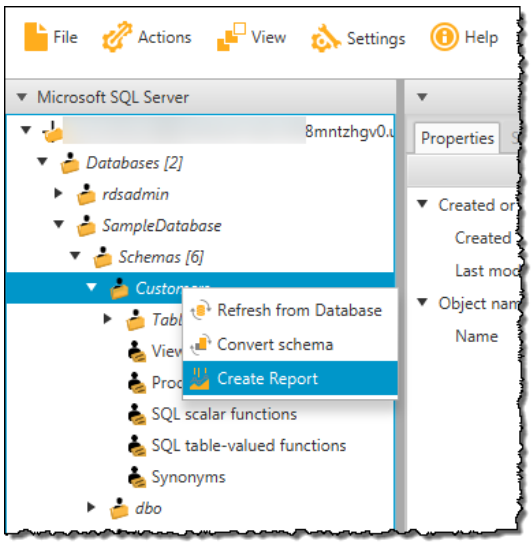

Abbildung 7: Erstellen des Migrationsberichts

Die Registerkarte **Summary** zeigt die zusammenfassenden Informationen aus dem Bewertungsbericht zur Datenbankmigration an. Sowohl automatisch konvertierte Elemente als auch Elemente, die nicht automatisch konvertiert werden konnten, sind darin aufgeführt.

Für Schemaelemente, die nicht automatisch in die Zieldatenbank-Engine konvertiert werden konnten, bietet die Zusammenfassung eine Schätzung des Aufwands, der zur Erstellung eines Schemas in Ihrer Ziel-DB-Instance erforderlich wäre, das Ihrer Quelldatenbank entspricht. Der geschätzte Zeitaufwand für die Konvertierung dieser Schemaelemente wird im Bericht in folgende Kategorien unterteilt:

- **Einfach** – Aktionen, die in unter einer Stunde abgeschlossen werden können
- **Medium** – Aktionen, die komplexer sind und in ein bis vier Stunden abgeschlossen werden können

- **Erheblich** – Aktionen, die sehr komplex sind und mehr als vier Stunden in Anspruch nehmen werden

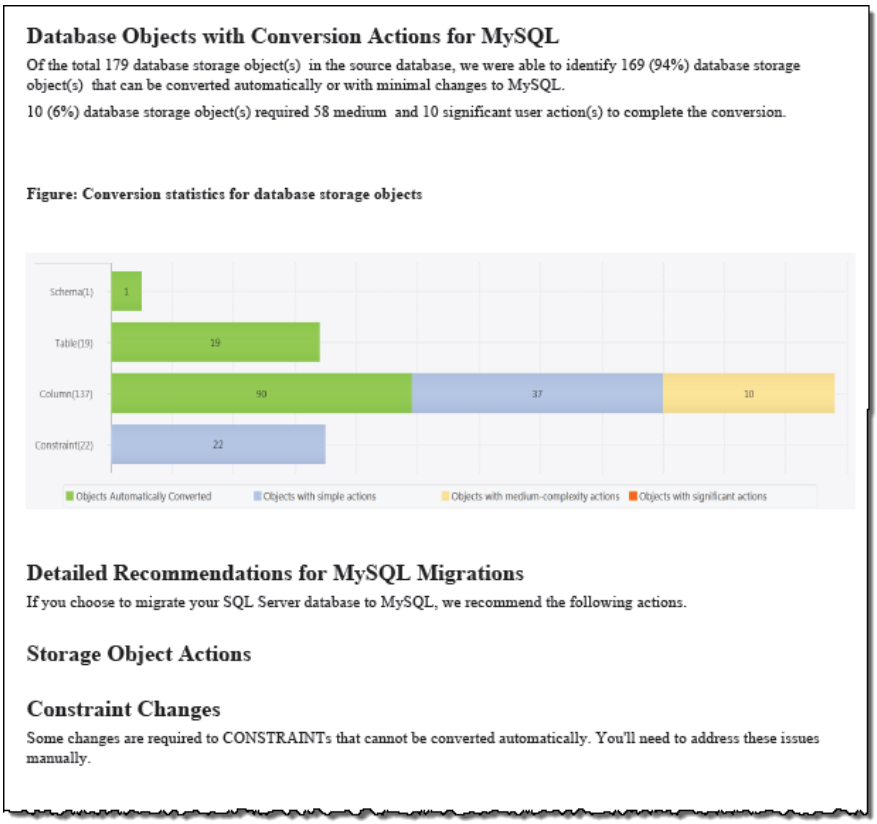

Abbildung 8: Migrationsbericht

Wichtig: Bei der Bewertung des Aufwands für Ihr Datenbank-Migrationsprojekt sollten Sie diesen Bewertungsbericht unbedingt berücksichtigen. Sehen Sie sich den Bewertungsbericht sehr genau an, um zu bestimmen, welche Codeänderungen beim Datenbankschema erforderlich sind und welche Auswirkungen diese Änderungen auf die Funktionalität und das Design Ihrer Anwendung haben könnten.

11. Der nächste Schritt besteht darin, das Schema zu konvertieren. Das konvertierte Schema wird nicht sofort für die Zieldatenbank übernommen. Stattdessen wird es lokal gespeichert, bis Sie das konvertierte Schema explizit für die Zieldatenbank übernehmen. Um das Schema aus Ihrer Quelldatenbank zu konvertieren, wählen Sie ein Schemaobjekt aus dem linken Bereich Ihres Projekts. Klicken Sie mit der rechten Maustaste auf das Objekt, und wählen Sie **Convert Schema**, wie in der folgenden Abbildung gezeigt.

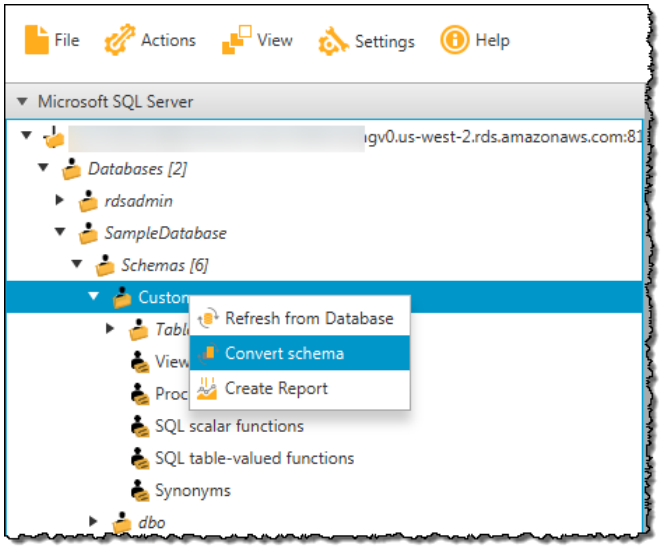

Abbildung 9: Konvertieren des Schemas

Diese Aktion fügt das konvertierte Schema zum rechten Bereich des Projektfensters hinzu und zeigt Objekte, die vom AWS Schema Conversion Tool automatisch konvertiert wurden.

12. Sie können die Aktionselemente im Bericht auf unterschiedliche Weise in Angriff nehmen:

- Sie können manuell ein vergleichbares Schema hinzufügen. Sie können den Teil des Schemas, der automatisch konvertiert werden kann, in die Ziel-DB-Instance schreiben, indem Sie im rechten Bereich Ihres Projekts **Apply to database** wählen. Das Schema, das in die Ziel-DB-Instance geschrieben wird, enthält nur Elemente, die automatisch konvertiert werden konnten. Diese Elemente sind in Ihrem Bewertungsbericht zur Datenbankmigration aufgeführt.

  Nachdem Sie das Schema in Ihre Ziel-DB-Instance übernommen haben, können Sie das Schema für die Elemente, die nicht automatisch konvertiert werden konnten, manuell in der Ziel-DB-Instance erstellen. In manchen Fällen können Sie unter Umständen kein vergleichbares Schema in Ihrer Ziel-DB-Instance erstellen. Möglicherweise müssen Sie Teile Ihrer Anwendung und Datenbank dahingehend überarbeiten, dass Sie die Funktionalität der DB-Engine für die Ziel-DB-Instance nutzen können. In anderen Fällen können Sie das nicht automatisch konvertierbare Schema einfach ignorieren.

Achtung: Wenn Sie das Schema manuell in der Ziel-DB-Instance erstellen, wählen Sie **Apply to database** erst aus, wenn Sie eine Kopie all Ihrer manuellen Bemühungen gespeichert haben. Wenn Sie das Schema aus Ihrem Projekt in die Ziel-DB-Instance übernehmen, wird das Schema mit demselben Namen in der Ziel-DB-Instance überschrieben und Sie verlieren sämtliche Aktualisierungen, die Sie manuell hinzugefügt haben.

- Ändern Sie Ihr Quelldatenbankschema und aktualisieren Sie das Schema in Ihrem Projekt. Für einige Elemente sollten Sie vorzugsweise das Datenbankschema in Ihrer Quelldatenbank in das Schema ändern, das mit Ihrer Anwendungsarchitektur kompatibel ist und auch automatisch in die DB-Engine Ihrer Ziel-DB-Instance konvertiert werden kann. Nachdem Sie das Schema in der Quelldatenbank aktualisiert und verifiziert haben, dass die Aktualisierungen mit Ihrer Anwendung kompatibel sind, wählen Sie im linken Bereich Ihres Projekts **Refresh from Database** aus, um das Schema aus Ihrer Quelldatenbank zu aktualisieren. Anschließend können Sie Ihr aktualisiertes Schema konvertieren und den Bewertungsbericht zur Datenbankmigration erneut generieren. Das Aktionselement für das aktualisierte Schema wird nicht mehr angezeigt.

13. Wenn Sie bereit sind, Ihr konvertiertes Schema auf Ihre Aurora-Ziel-Instance zu übertragen, wählen Sie das Schemaelement im rechten Bereich Ihres Projekts aus. Klicken Sie mit der rechten Maustaste auf das Schemaelement, und wählen Sie **Apply to database** aus, wie in der folgenden Abbildung dargestellt.

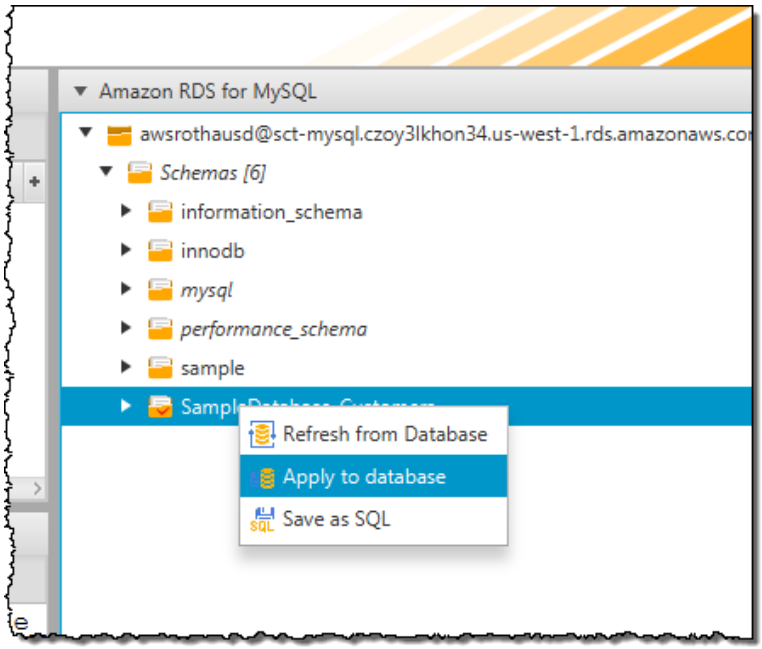

Abbildung 10: Übernehmen des Schemas für die Datenbank

Hinweis: Wenn Sie Ihr konvertiertes Schema erstmals für Ihre Ziel-DB-Instance übernehmen, fügt das AWS Schema Conversion Tool ein zusätzliches Schema (`AWS_ORACLE_EXT` bzw. `AWS_SQLSERVER_EXT`) zu Ihrer Ziel-DB-Instance hinzu. Dieses Schema implementiert Systemfunktionen der Quelldatenbank, die benötigt werden, wenn Ihr konvertiertes Schema in die Ziel-DB-Instance geschrieben wird. Ändern Sie dieses Schema keinesfalls, denn dies könnte zu unerwarteten Ergebnissen im konvertierten Schema führen, das in die Ziel-DB-Instance geschrieben wird. Wenn Ihr Schema vollständig in die Ziel-DB-Instance migriert ist und Sie das AWS Schema Conversion Tool nicht mehr benötigen, können Sie das Schema `AWS_ORACLE_EXT` bzw. `AWS_SQLSERVER_EXT` löschen.

Das AWS Schema Conversion Tool ist eine benutzerfreundliche Ergänzung für Ihr Migrations-Toolkit. Weitere bewährte Methoden für AWS Schema Conversion Tool finden Sie im Thema Best Practices im *AWS Schema Conversion Tool User Guide*.[17]

Migrieren von Daten

Nachdem das Datenbankschema von der Quelldatenbank in die Aurora-Zieldatenbank kopiert wurde, besteht der folgende Schritt in der Migration der

Daten von der Quelle zum Ziel. Zwar kann die Datenmigration auch mit anderen Tools durchgeführt werden, aber es empfiehlt sich, hierfür AWS Database Migration Service (AWS DMS) zu verwenden, denn dieser Service bietet die Einfachheit und die Funktionen, die Sie für die jeweilige Aufgabe benötigen.

Einführung und Allgemeines zu AWS DMS

Mit AWS Database Migration Service (DMS) können Kunden ihre Produktionsdatenbanken mit minimaler Ausfallzeit zu AWS migrieren. Sie können Ihre Anwendungen während der Migration Ihrer Datenbank weiter ausführen. Außerdem ist mit AWS Database Migration Service sichergestellt, dass Datenänderungen in der Quelldatenbank, die während und nach der Migration erfolgen, kontinuierlich in der Zieldatenbank repliziert werden. Migrationsaufgaben können in wenigen Minuten in der AWS Management Console eingerichtet werden. AWS Database Migration Service kann Ihre Daten zu und von gängigen Datenbankplattformen wie Oracle, SQL Server, MySQL, PostgreSQL, Amazon Aurora, MariaDB und Amazon Redshift migrieren.

Der Service unterstützt sowohl homogene Migrationen, etwa Oracle zu Oracle, als auch heterogene Migrationen zwischen unterschiedlichen Datenbankplattformen, etwa Oracle zu Amazon Aurora oder SQL Server zu MySQL. Sie können einmalige Migrationen durchführen oder eine kontinuierliche Datenreplikation zwischen Datenbanken vornehmen, ohne dass irgendwelche komplizierte Software installiert oder konfiguriert werden muss.

AWS DMS eignet sich für Datenbanken, die lokal, auf Amazon EC2 oder auf Amazon RDS ausgeführt werden. AWS DMS bietet sich jedoch nicht für Situationen an, in denen sich sowohl die Quell- als auch die Zieldatenbank lokal befinden; ein Endpunkt muss in AWS angesiedelt sein.

AWS DMS unterstützt bestimmte Versionen von Oracle, SQL Server, Amazon Aurora, MySQL und PostgreSQL. Welche Versionen derzeit unterstützt werden, können Sie im AWS Database Migration Service User Guide nachlesen.[18] Im Mittelpunkt dieses Whitepapers steht jedoch Amazon Aurora als Ziel für die Migration.

Migrationsmethoden

AWS DMS bietet drei Methoden zur Datenmigration:

Migrieren von vorhandene Daten. Bei dieser Methode werden die Tabellen in der Zieldatenbank erstellt und die im Ziel erforderlichen Metadaten automatisch definiert. Schließlich werden die Tabellen automatisch mit Daten aus der Quelldatenbank aufgefüllt („vollständiges Laden“). Zur Verbesserung der Effizienz werden die Daten aus den Tabellen parallel geladen. Tabellen werden nur bei homogenen Migrationen erstellt. Sekundäre Indizes werden nicht automatisch von AWS DMS erstellt. Weitere Einzelheiten folgen.

Migrieren von vorhandenen Daten und Replizieren von laufenden Änderungen. Bei dieser Methode wird ein vollständiger Ladevorgang durchgeführt, wie oben beschrieben. Zusätzlich werden alle laufenden Änderungen an der Quelldatenbank während des vollständigen Ladens erfasst und auf der Replikations-Instance gespeichert. Sobald der vollständige Ladenvorgang abgeschlossen ist, werden die gespeicherten Änderungen in die Zieldatenbank übertragen, bis diese mit der Quelldatenbank übereinstimmt. Zusätzlich werden alle laufenden Änderungen an der Quelldatenbank weiterhin in die Zieldatenbank repliziert, damit beide synchronisiert bleiben. Diese Migrationsmethode ist sehr nützlich, wenn Sie eine Datenbankmigration mit möglichst geringer Ausfallzeit durchführen möchten.

Reines Replizieren von Datenänderungen. Diese Methode liest nur Änderungen aus der Recovery-Protokolldatei der Quelldatenbank und übernimmt diese Änderungen fortlaufend in die Zieldatenbank. Wenn die Zieldatenbank nicht verfügbar ist, werden diese Änderungen in der Replikations-Instance gepuffert, bis das Ziel verfügbar ist.

Wenn AWS DMS eine Migration mit vollständigem Laden durchführt, verursacht dies eine Belastung der Tabellen in der Quelldatenbank, die sich auf die Leistung von Anwendungen auswirken kann, die zeitgleich auf diese Datenbank zugreifen. Wenn dies ein Problem darstellt und Sie Ihre eigenen Anwendungen während der Migration nicht herunterfahren können, sollten Sie Folgendes in Erwägung ziehen:

- Ausführen der Migration zu einer Zeit, zu der die Belastung der Datenbank möglichst gering ist
- Erstellen einer Read Replica der Quelldatenbank und anschließende AWS DMS-Migration auf der Basis der Read Replica

Migrationsverfahren

Die allgemeine Vorgehensweise bei der Nutzung von AWS DMS ist folgende:

1. Erstellen Sie eine Zieldatenbank.
2. Kopieren Sie das Schema.
3. Erstellen Sie eine AWS DMS-Replikations-Instance.
4. Definieren Sie die Endpunkte für die Quell- und Zieldatenbank.
5. Erstellen Sie eine Migrationsaufgabe und führen Sie sie aus.

Erstellen der Zieldatenbank

Erstellen Sie Ihren Amazon Aurora-Datenbank-Cluster mittels des Verfahrens, das unter Creating an Amazon Aurora DB Cluster im *Amazon RDS User Guide* beschrieben ist.[19] Sie sollten die Region und den Instance-Typ bei der Erstellung der Zieldatenbank so wählen, dass beide Ihren geschäftlichen Anforderungen entsprechen. Zur Verbesserung der Migrationsleistung stellen Sie sicher, dass die Multi-AZ-Bereitstellung bei der Zieldatenbank deaktiviert ist. Sie können sie aktivieren, sobald der Ladevorgang abgeschlossen ist.

Kopieren des Schemas

Außerdem sollten Sie das Schema in dieser Zieldatenbank erstellen. AWS DMS unterstützt eine grundlegende Schemamigration, einschließlich der Erstellung von Tabellen und Primärschlüsseln. AWS DMS erstellt jedoch nicht automatisch sekundäre Indizes, Fremdschlüssel, gespeicherte Prozeduren, Benutzerkonten usw. in der Zieldatenbank. Ausführliche Informationen zur Schemamigration finden Sie im Abschnitt Migrieren des Datenbankschemas.

Erstellen einer AWS DMS-Replikations-Instance

Um AWS DMS nutzen zu können, müssen Sie eine AWS DMS-Replikations-Instance erstellen, die auf Ihrem VPC ausgeführt wird. Diese Instanz liest die Daten aus der Quelldatenbank, führt die angegebenen Tabellenzuweisungen durch und schreibt die Daten in die Zieldatenbank. In der Regel lässt sich die Datenbankmigration durch Verwendung einer größeren Replikations-Instance beschleunigen (wenngleich die Migration auch durch andere Faktoren wie die Kapazität der Quell- und Zieldatenbank, Verbindungslatenz usw. verlangsamt werden kann). Ferner kann die Replikations-Instance nach Abschluss der Datenmigration auch gestoppt werden.

Abbildung 11: AWS Database Migration Service

AWS DMS unterstützt derzeit die Instance-Klassen T2 und C4 als Replikations-Instances. Die T2-Instance-Klassen sind kostengünstige Standard-Instances mit einer CPU-Basisleistung, die durch Bursts über die Baseline hinaus dynamisch erweiterbar ist. Sie eignen sich für die Entwicklung, Konfiguration und das Testen des Datenbank-Migrationsprozesses sowie für regelmäßig anfallende Datenmigrationsaufgaben, die von der CPU-Burst-Fähigkeit profitieren. Die C4-Instance-Klassen sind für die Bereitstellung der größtmöglichen Prozessorleistung ausgelegt und bieten eine erheblich höhere PPS-Leistung (packet per second, Paket pro Sekunde), geringeren Netzwerk-Jitter und niedrigere Netzwerklatenz. C4-Instance-Klassen sollten Sie einsetzen, wenn Sie den Zeitaufwand für das Migrieren großer Datenbanken minimieren möchten.

Für einen vollständigen Ladevorgang wird im Normalfall nur wenig Instance-Speicher auf Ihrer AWS DMS-Replikations-Instance benötigt. Wenn Sie jedoch die Replikation zusammen mit dem vollständigen Ladevorgang ausführen, werden die Änderungen an der Quelldatenbank auf der AWS DMS-Replikations-Instance zeitgleich mit dem Ladevorgang gespeichert. Bei der Migration einer sehr großen Quelldatenbank, für die zudem während der Migration zahlreiche Aktualisierungen erfolgen, wird eine erhebliche Menge an Instance-Speicher benötigt. Die C4-Instance-Familie bietet 100 GB Instance-Speicher und die T2-Instance-Familie 50 GB. Normalerweise sind diese Speicherkapazitäten für die meisten Migrationsszenarien mehr als ausreichend.

In einigen extremen Fällen, in denen sehr große Datenbanken mit sehr hohen Transaktionsraten und aktivierter Replikation migriert werden, kann es sein, dass die AWS DMS-Replikation möglicherweise nicht Schritt halten kann. In einer solchen Situation müssen Sie unter Umständen die Änderungen an der Quelldatenbank für einige Minuten aussetzen, damit alle aufgelaufenen Änderungen die Replikation durchlaufen können, bevor Sie Ihre Anwendung erneut auf die Aurora-Zieldatenbank verweisen.

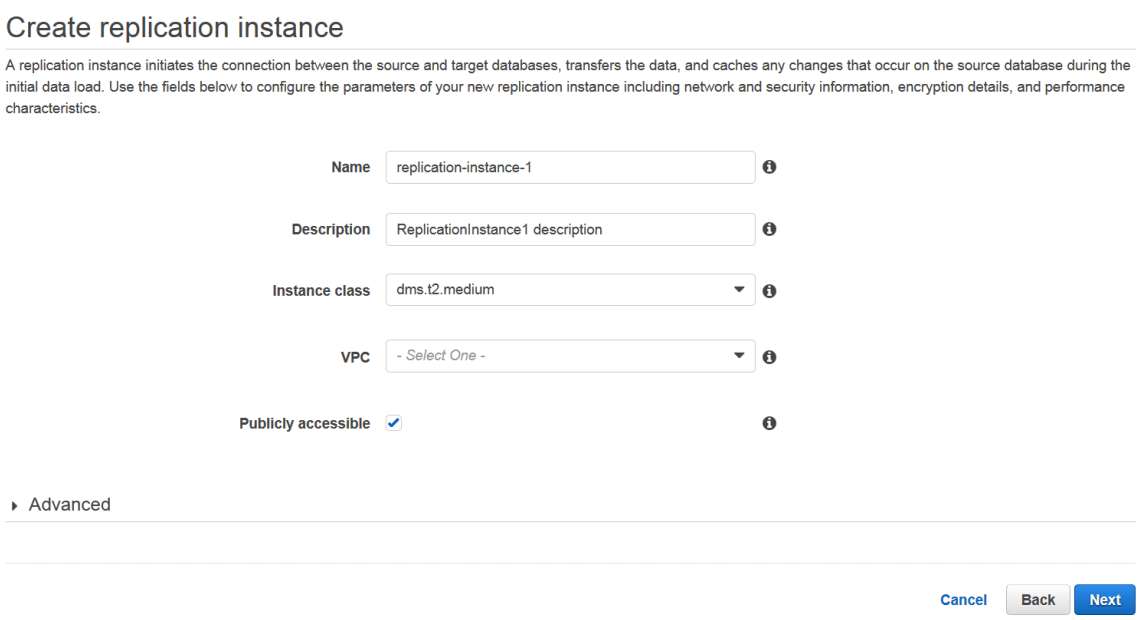

Abbildung 12: Seite zum Erstellen einer Replikations-Instance in der AWS DMS-Konsole

Definieren von Endpunkten für die Quell- und die Zieldatenbank

Ein Datenbank-Endpunkt wird von der Replikations-Instance zum Herstellen einer Verbindung zu einer Datenbank genutzt. Für eine Datenbank-Migration müssen Sie einen Endpunkt für die Quelldatenbank *und* einen Endpunkt für die Zieldatenbank erstellen. Die Datenbank-Endpunkte können entweder lokal, auf Amazon EC2 oder Amazon RDS definiert werden, wobei Quelle und Ziel aber nicht beide als lokale Endpunkte erstellt werden dürfen.

Die Verbindung der Datenbank-Endpunkte sollten Sie im Anschluss an die Definition unbedingt testen. Zum Testen greifen Sie einfach auf dieselbe Seite zurück, die Sie auch zur Erstellung von Datenbank-Endpunkten verwenden. Hierauf wird weiter unten in diesem Whitepaper eingegangen.

Hinweis: Sollten in Ihrem Quellschema Fremdschlüsseleinschränkungen vorliegen, müssen Sie beim Erstellen Ihres Zielendpunkts im Abschnitt **Advanced** unter **Extra connection attributes** Folgendes eingeben:

```
initstmt=SET FOREIGN_KEY_CHECKS=0
```

Damit werden die Fremdschlüsselprüfungen während des Ladens der Zieltabellen deaktiviert. Dadurch wird verhindert, dass der Ladevorgang durch

fehlgeschlagene Fremdschlüsselprüfungen bei teilweise geladenen Tabellen unterbrochen wird.

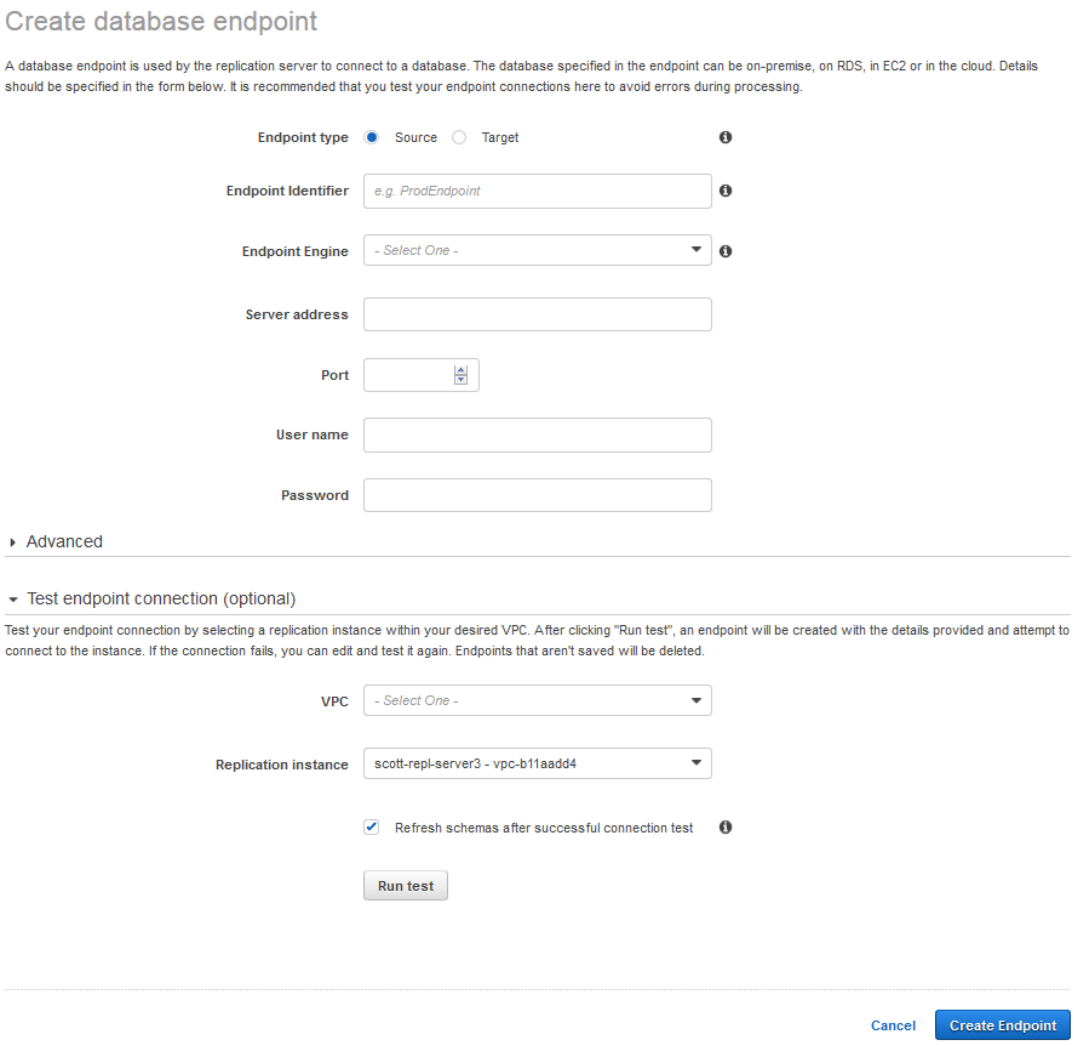

Abbildung 13: Seite zum Erstellen des Datenbank-Endpunkts in der AWS DMS-Konsole

Erstellen und Ausführen einer Migrationsaufgabe

Nachdem Sie Ihre Quelldatenbank erstellt und den Endpunkt der Quell- und Zieldatenbank getestet haben, können Sie eine Aufgabe für die Datenmigration erstellen. Beim Erstellen dieser Aufgabe geben Sie die Replikations-Instance an, die Sie erstellt haben, die (zuvor beschriebene) Datenbank-Migrationsmethode, den Endpunkt der Quelldatenbank und den Endpunkt der Zieldatenbank für Ihren Amazon Aurora-Datenbank-Cluster.

Wenn Sie in der Zieldatenbank bereits das vollständige Schema erstellt haben, sollten Sie in den **Task Settings** unter **Target table preparation mode** anstelle des Standardwerts **Drop tables on target** die Option **Do nothing** aktivieren. Beim Standardwert **Drop tables on target** werden die Tabellen verworfen und neu erstellt, wobei Aspekte Ihrer Schemadefinition, beispielsweise Fremdschlüsseleinschränkungen, verloren gehen können.

Beim Erstellen einer Aufgabe können Sie Tabellenzuweisungen erstellen, die das Schema der Quelle sowie die entsprechenden Tabellen definieren, die zum Zielendpunkt migriert werden sollen. Die standardmäßige Zuweisungsmethode migriert alle Quelltabellen in die Zieltabellen mit demselben Namen, sofern vorhanden. Andernfalls wird die Quelltabelle bzw. werden die Quelltabellen im Ziel erstellt (je nachdem, mit welchen Aufgabeneinstellungen Sie arbeiten). Wenn Sie nur bestimmte Tabellen migrieren möchten oder mehr Kontrolle über das Feld und den Vorgang der Tabellenzuweisung haben möchten, können Sie außerdem (unter Verwendung einer JSON-Datei) benutzerdefinierte Zuweisungen erstellen. Sie können ferner wählen, ob Sie nur ein Schema oder alle Schemas von Ihrem Quellendpunkt migrieren wollen.

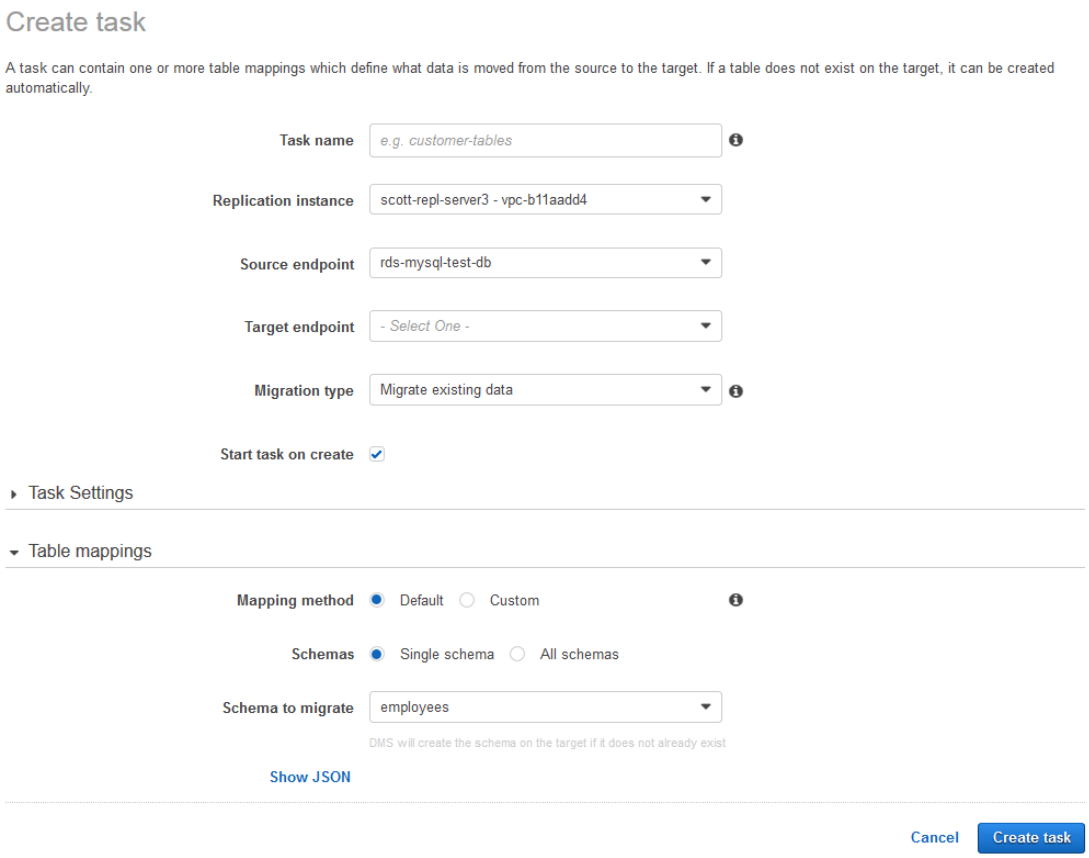

Abbildung 14: Seite zum Erstellen der Aufgabe in der AWS DMS-Konsole

Anhand der AWS Management Console können Sie den Fortschritt der AWS Database Migration Service (AWS DMS)-Aufgaben überwachen. Außerdem können Sie die Ressourcen und die genutzten Netzwerkverbindungen überwachen. Die AWS DMS-Konsole zeigt grundlegende Statistiken für jede Aufgabe an, darunter den Aufgabenstatus, Prozent abgeschlossen, verstrichene Zeit und Tabellenstatistiken, wie die folgende Abbildung zeigt.

Zusätzlich können Sie eine Aufgabe auswählen und die zugehörige Leistungsmetrik anzeigen, einschließlich Durchsatz, Anzahl der pro Sekunde migrierten Datensätze, Festplatten- und Speichernutzung sowie Latenz.

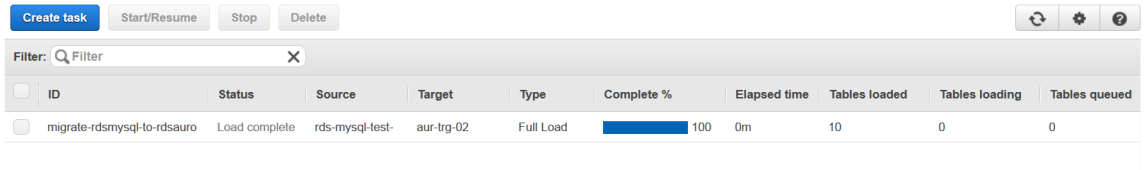

Abbildung 15: Aufgabenstatus in AWS DMS-Konsole

Testen und Umstellung

Sobald das Schema und die Daten erfolgreich von der Quelldatenbank zu Amazon Aurora migriert sind, können Sie Ihren Migrationsprozess durchgehend testen. Der Testansatz sollte nach jeder Testmigration verfeinert werden. Ihr endgültiger Migrationsplan sollte einen Testplan umfassen, der sicherstellt, dass die migrierte Datenbank in angemessener Weise getestet wird.

Migrationstests

Testkategorie	Zweck
Grundlegende Akzeptanztests	Diese Tests vor der Umstellung sollten nach Abschluss des Prozesses zur Datenmigration automatisch durchgeführt werden. Ihr Zweck besteht vornehmlich darin, zu überprüfen, ob die Datenmigration erfolgreich war. Im Folgenden sind einige gängige Ergebnisse dieser Tests aufgeführt: • Gesamtzahl der verarbeiteten Elemente • Gesamtzahl der importierten Elemente • Gesamtzahl der übersprungenen Elemente • Gesamtzahl der Warnungen • Gesamtzahl der Fehler Sollte eine dieser bei den Tests gemeldeten Summen von den erwarteten Werten abweichen, bedeutet dies, dass die Migration nicht erfolgreich war und die Probleme gelöst werden müssen, bevor mit dem nächsten Schritt im Testvorgang oder dem nächsten Testlauf fortgefahren werden kann.
Funktionstests	Bei diesen Tests im Anschluss an die Umstellung wird die Funktionalität der Anwendung(en) unter Verwendung von Aurora für die Datenspeicherung geprüft. Sie umfassen eine Kombination aus automatisierten und manuellen Tests. Die Funktionstests dienen vornehmlich dazu, Probleme in der Anwendung zu identifizieren, die durch die Migration der Daten zu Aurora bedingt sind.
Nicht funktionsbezogene Tests	Bei diesen Tests im Anschluss an die Umstellung werden die nicht funktionsbezogenen Merkmale der Anwendung bewertet, zum Beispiel die Leistung unter verschiedenen Lasten.
Benutzerakzeptanztests	Diese Tests im Anschluss an die Umstellung sollten von den Endbenutzern der Anwendung ausgeführt werden, sobald die Datenmigration und Umstellung vollständig abgeschlossen sind. Auf der Basis dieser Tests sollen die Endbenutzer entscheiden, ob die Anwendung in ihrer primären Funktion im Unternehmen nutzbar ist.

Umstellung

Wenn die Migration und alle Tests abgeschlossen sind, können Sie in Ihrer Anwendung auf die Amazon Aurora-Datenbank verweisen. Diese Phase der Migration wird als *Umstellung* bezeichnet. Wenn die Planung und die Testphase ordnungsgemäß ausgeführt wurden, sollten bei der Umstellung keine unerwarteten Probleme auftreten.

Maßnahmen vor der Umstellung

- Wählen Sie ein Umstellungsfenster: Bestimmen Sie einen Zeitraum, innerhalb dessen Sie die Umstellung auf die neue Datenbank mit minimaler Unterbrechung des Betriebs ausführen können. In der Regel würden Sie einen Zeitraum mit niedriger Aktivität für die Datenbank auswählen (normalerweise in der Nacht und/oder am Wochenende).
- Stellen Sie sicher, dass alle Änderungen auf dem neuesten Stand sind: Wenn Sie sich für einen Ansatz mit nahezu keiner Ausfallzeit bei der Migration und Replikation von der Quell- zur Zieldatenbank entschieden haben, stellen Sie sicher, dass alle Änderungen an der Datenbank nachgeführt wurden und Ihre Zieldatenbank keinen nennenswerten Rückstand gegenüber der Quelldatenbank aufweist.
- Bereiten Sie Skripts für die Änderungen in der Anwendungskonfiguration vor: Für die Umstellung müssen Sie die Datenbankverbindungsdetails in den Konfigurationsdateien Ihrer Anwendung ändern. Bei großen und komplexen Anwendungen können an mehreren Stellen Aktualisierungen für Verbindungsdetails erforderlich sein. Stellen Sie sicher, dass die nötigen Skripts für eine schnelle und zuverlässige Aktualisierung der Verbindungskonfiguration einsatzbereit sind.
- Beenden Sie die Anwendung: Beenden Sie die Anwendungsprozesse in der Quelldatenbank und versetzen Sie die Quelldatenbank in den schreibgeschützten Modus, sodass keine weiteren Schreibvorgänge in der Quelldatenbank vorgenommen werden können. Sollten die Änderungen in der Quelldatenbank noch nicht vollständig in die Zieldatenbank überführt worden sein, warten Sie, bis die Zieldatenbank dem neuesten Änderungsstand entspricht.

- Führen Sie Tests vor der Umstellung aus: Führen Sie die automatisierten Tests vor der Umstellung aus, um sicherzustellen, dass die Datenmigration erfolgreich war.

Umstellung

- Führen Sie die Umstellung durch: Wenn die Tests vor der Umstellung erfolgreich abgeschlossen wurden, können Sie jetzt in Ihrer Anwendung auf Amazon Aurora verweisen. Führen Sie die in der Phase vor der Umstellung erstellten Skripts aus, um die Anwendungskonfiguration dahingehend zu ändern, dass sie nun auf die neue Aurora-Datenbank verweist.
- Starten Sie die Anwendung: An diesem Punkt können Sie Ihre Anwendung starten. Sollten Sie die Möglichkeit haben, den Benutzerzugriff auf die Anwendung zu unterbinden, tun Sie dies, bis Sie Ihre Tests nach der Umstellung durchgeführt haben.

Tests nach der Umstellung

- Führen Sie Tests nach der Umstellung durch: Führen Sie die vordefinierten automatisierten oder manuellen Tests durch, um sicherzustellen, dass Ihre Anwendung ordnungsgemäß mit der neuen Datenbank zusammenarbeitet. Es empfiehlt sich, zunächst die schreibgeschützte Funktionalität der Datenbank zu testen, bevor Tests ausgeführt werden, bei denen in die Datenbank geschrieben wird.
- Gewähren Sie den Benutzern Zugriff bei sorgfältiger Überwachung: Wenn Ihre Testfälle erfolgreich ausgeführt wurden, können Sie den Benutzern Zugriff auf die Anwendung gewähren und damit den Migrationsprozess abschließen. In dieser Phase sollten Anwendung und Datenbank sorgfältig überwacht werden.

Fazit

Amazon Aurora ist eine leistungsfähige, hochverfügbare Datenbank der Enterprise-Klasse, die für den Einsatz in der Cloud entwickelt wurde. Mit Amazon Aurora kann eine bessere Leistung und höhere Verfügbarkeit erreicht werden als bei anderen Open-Source-Datenbanken. Gleichzeitig fallen niedrigere Kosten an als bei den meisten kommerziell genutzten Datenbanken. In diesem Whitepaper werden Strategien vorgestellt, mit denen sich die beste Methode für die Migration von Datenbanken zu Amazon Aurora bestimmen lässt. Ferner werden die Schritte zur Planung und Umsetzung dieser Migrationsmethoden

erläutert. Für heterogene Migrationsszenarien werden insbesondere AWS Database Migration Service (AWS DMS) und AWS Schema Conversion Tool empfohlen. Mit diesen leistungsstarken Tools können Sie die Kosten und die Komplexität von Datenbankmigrationen ganz erheblich reduzieren.

Mitwirkende

Dieses Dokument ist unter der Mitarbeit folgender Personen und Organisationen entstanden:

- Puneet Agarwal, Lösungsarchitekt, Amazon Web Services
- Scott Williams, Lösungsarchitekt, Amazon Web Services

Weitere Informationen

Die folgenden Ressourcen bieten Ihnen zusätzliche Informationen:

- Amazon Aurora – Produktdetails
- Amazon Aurora – Häufig gestellte Fragen
- Amazon Database Migration Service
- Amazon Database Migration Service – Häufig gestellte Fragen

Anmerkungen

[1] https://aws.amazon.com/rds/aurora/

[2] http://aws.amazon.com/rds/aurora/pricing/

[3] https://do.awsstatic.com/product-marketing/Aurora/Aurora_Export_Import_Best_Practices_v1-3.pdf

[4] http://docs.aws.amazon.com/pt_br/AmazonRDS/latest/UserGuide/Aurora.Replication.html#Aurora.Overview.Replication.MySQLReplication

[5] http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Aurora.Migrate.html#USER_ImportAurora.PreImport

[6] http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_CreateSnapshot.html

[7] http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_CopySnapshot.html

[8] http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Aurora.Migrate.html#USER_ImportAuroraCluster.Console

[9] http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Aurora.Connect.html

[10] https://dev.mysql.com/doc/refman/5.6/en/mysqldump.html

[11] http://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/Welcome.html

[12] http://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_SchemaConversionTool.GettingStarted.html

[13] http://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_SchemaConversionTool.Installing.html

[14] http://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_SchemaConversionTool.Installing.html#CHAP_SchemaConversionTool.Installing.JDBCDrivers

[15] https://forums.aws.amazon.com/forum.jspa?forumID=208

[16] http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Aurora.CreateInstance.html

[17] http://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_SchemaConversionTool.BestPractices.html

[18] http://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.html

[19] http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Aurora.CreateInstance.html