Migrazione dei database ad Amazon Aurora

Giugno 2016

Note

Il presente documento è fornito a solo scopo informativo. In esso sono illustrate le attuali offerte di prodotti e le prassi di AWS alla data di pubblicazione del documento, offerte che sono soggette a modifica senza preavviso. È responsabilità dei clienti effettuare una propria valutazione indipendente delle informazioni contenute nel presente documento e dell'uso dei prodotti o dei servizi di AWS, ciascuno dei quali viene fornito "così com'è", senza garanzie di alcun tipo, né esplicite né implicite. Il presente documento non dà origine a garanzie, rappresentazioni, impegni contrattuali, condizioni o assicurazioni da parte di AWS, delle sue società affiliate, dei suoi fornitori o dei licenzianti. Le responsabilità di AWS nei confronti dei propri clienti sono definite dai contratti AWS e il presente documento non costituisce parte né modifica qualsivoglia contratto tra AWS e i suoi clienti.

Indice

Sintesi

Amazon Aurora è un motore di database relazionale di livello enterprise compatibile con MySQL. Amazon Aurora è un database cloud nativo che supera molte delle limitazioni dei motori di database relazionali tradizionali. L'obiettivo di questo whitepaper è quello di evidenziare le best practise per la migrazione dei database esistenti ad Amazon Aurora. Presenta considerazioni sulla migrazione e la procedura dettagliata per la migrazione di database open source e commerciali ad Amazon Aurora con interruzioni minime per le applicazioni.

Presentazione di Amazon Aurora

Per decenni, i database relazionali tradizionali sono stati la scelta principali per lo storage e la persistenza dei dati. Questi sistemi di database continuano a fare affidamento su architetture monolitiche e non sono stati progettati per sfruttare i vantaggi offerti dall'infrastruttura cloud. Queste architetture monolitiche presentano numerose sfide, soprattutto in termini di costo, flessibilità e disponibilità. Per affrontare tali sfide, AWS ha riprogettato il database relazionale per l'infrastruttura cloud e presentato Amazon Aurora.

Amazon Aurora è un motore di database relazionale compatibile con MySQL che unisce la velocità, la disponibilità e la sicurezza dei database commerciali di fascia alta con la semplicità e la convenienza di quelli open source. Aurora fornisce prestazioni fino a cinque volte superiori rispetto a MySQL e database commerciali di fascia alta con prestazioni equivalenti. Il prezzo di Amazon Aurora è un decimo del costo dei motori commerciali.

Amazon Aurora è disponibile tramite la piattaforma Amazon Relational Database Service (Amazon RDS). Come altri database Amazon RDS, Aurora è un servizio di database completamente gestito. Con la piattaforma Amazon RDS, la maggior

parte delle attività di gestione del database quali effettuare il provisioning dell'hardware, applicazione di patch software, impostazione, configurazione, monitoraggio e backup sono completamente automatizzate.

Amazon Aurora è costruito per carichi di lavoro mission-critical ed è altamente disponibile per impostazione predefinita. Un cluster di database Aurora abbraccia più zone di disponibilità in una regione, in modo da fornire subito la durabilità e la tolleranza ai guasti per i dati su più data center fisici. Una zona di disponibilità è composta da uno o più data center ad alta disponibilità gestiti da Amazon. Le zone di disponibilità sono isolate tra loro e sono collegate tramite i collegamenti a bassa latenza. Ogni segmento del volume del database viene replicato sei volte su queste zone di disponibilità.

I volumi di cluster di Aurora aumentano automaticamente con l'aumentare dei dati del database senza influire sulla performance o sulla disponibilità; perciò non occorre effettuare la stima e il provisioning di grandi quantità di storage di database in anticipo. Un volume di cluster di Aurora può crescere a una dimensione massima di 64 terabyte (TB). Si paga solo per lo spazio utilizzato in un volume di cluster di Aurora.

La funzionalità di backup automatico di Aurora supporta il ripristino point-in-time dei dati, consentendo di ripristinare il database a qualsiasi punto nel tempo compreso nel periodo di retention, fino agli ultimi cinque minuti. I backup automatici vengono archiviati in Amazon Simple Storage Service (Amazon S3), che è stato progettato per una durabilità del 99,999999999%. I backup di Amazon Aurora sono automatici, incrementali e continui e non hanno alcun impatto sulle prestazioni del database.

Per le applicazioni che richiedono repliche di sola lettura, è possibile creare fino a 15 repliche per database Aurora con latenza di replica molto bassa. Queste repliche condividono lo stesso storage sottostante dell'istanza di origine, riducendo i costi ed eliminando la necessità di eseguire operazioni di scrittura ai nodi replica.

Amazon Aurora è altamente sicuro e permette di crittografare i database mediante le chiavi create e controllate dall'utente tramite AWS Key Management Service (AWS KMS). Su un'istanza database in esecuzione con crittografia Amazon Aurora, i dati salvati a riposo nello storage sottostante vengono criptati, così come snapshot, repliche e backup automatici nello stesso cluster. Amazon Aurora utilizza il protocollo SSL (AES-256) per la protezione dei dati in transito.

Per un elenco completo delle caratteristiche di Aurora, vedere la pagina dei prodotti Amazon Aurora.[1] Date le numerose funzioni e la convenienza, Amazon Aurora è sempre più considerato il database di riferimento per applicazioni mission-critical.

Considerazioni sulla migrazione dei database

Un database rappresenta un componente fondamentale dell'architettura della maggior parte delle applicazioni. La migrazione del database a una nuova piattaforma è un evento significativo in un ciclo di vita dell'applicazione e può avere un impatto sulla funzionalità, le prestazioni e l'affidabilità dell'applicazione. È necessario fare alcune considerazioni importanti prima di avviare il primo progetto di migrazione ad Amazon Aurora.

Fasi di migrazione

Poiché le migrazioni di database tendono ad essere complesse, consigliamo di adottare un approccio a fasi, iterativo.

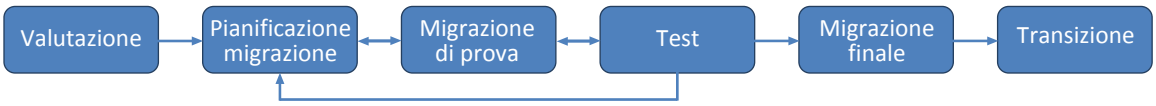

Figura 1: Fasi di migrazione

Considerazioni sull'applicazione

Valutare le caratteristiche di Aurora

Anche se la maggior parte delle applicazioni possono essere programmate per funzionare con molti motori di database relazionali, è necessario accertarsi che l'applicazione funzioni con Amazon Aurora. Amazon Aurora è stato progettato per essere compatibile con MySQL 5.6. Pertanto, la maggior parte di codici, applicazioni, driver e strumenti oggi usati con i database MySQL potranno essere impiegati con Aurora con modifiche minime o nulle.

Tuttavia, alcune caratteristiche di MySQL, come il motore di storage MyISAM, non sono disponibili con Amazon Aurora. Inoltre, data la natura gestita del servizio Aurora, l'accesso SSH ai nodi del database è limitato, e potrebbe

influenzare la capacità di installare strumenti di terze parti o plug-in sull'host del database.

Considerazioni sulle prestazioni

Le prestazioni del database sono un fattore chiave per la migrazione di un database a una nuova piattaforma. Pertanto, i progetti di migrazione di database più riusciti iniziano con le valutazioni delle prestazioni della nuova piattaforma di database. Anche se l'esecuzione dei benchmark Sysbench e TPC-C offre una discreta idea delle prestazioni globali del database, questi benchmark non sono in grado di emulare i modelli dei dati di accesso delle applicazioni. Per ulteriori risultati utili, testare le prestazioni del database per i carichi di lavoro urgenti eseguendo le query (o un sottoinsieme di query) direttamente sulla nuova piattaforma.

Considerare queste strategie:

- Se il database attuale è MySQL, migrare ad Amazon Aurora con tempi di inattività e valutare le prestazioni con una versione di prova o temporanea dell'applicazione oppure ripetendo il carico di lavoro di produzione.

- Se si utilizza un motore non conforme a MySQL, è possibile copiare in modo selettivo le tabelle più occupate su Amazon Aurora e testare le query per tali tabelle. Questo offre un buon punto di partenza. Ovviamente, fare il test dopo aver completato la migrazione dei dati fornisce un'immagine completa delle prestazioni reali dell'applicazione sulla nuova piattaforma.

Amazon Aurora offre prestazioni equiparabili con motori commerciali e un notevole miglioramento rispetto alle prestazioni di MySQL. Questo avviene integrando il motore di database con un livello di storage virtualizzato basato su SSD concepito per carichi di lavoro di database. Ciò consente di ridurre le operazioni di scrittura sul sistema di storage, riduce al minimo le contese per il lock ed elimina i ritardi creati dai thread di processo del database. I nostri test con SysBench su istanze r3.8xlarge dimostrano che Amazon Aurora fornisce oltre 585.000 letture al secondo e 107.000 operazioni di scrittura al secondo, cinque volte superiore a MySQL, eseguendo lo stesso benchmark sullo stesso hardware.

Un settore in cui Amazon Aurora risulta migliore rispetto al tradizionale MySQL sono i carichi di lavoro altamente simultanei. Per massimizzare il throughput del carico di lavoro su Amazon Aurora, consigliamo di progettare le proprie applicazioni per gestire un numero elevato di query simultanee.

Considerazioni sullo sharding e sulla replica di lettura

Se il database attuale è shardato su più nodi, è possibile avere l'opportunità di combinare questi shard in un singolo database Aurora durante la migrazione. Una singola istanza Amazon Aurora può essere estesa fino a 64 TB, supporta migliaia di tabelle e un numero di letture e scritture notevolmente superiore rispetto a un database MySQL standard.

Per le applicazioni che effettuano molte operazioni di lettura/scrittura è consigliabile utilizzare le repliche di lettura Aurora al fine di sottrarre i carichi di lavoro di sola lettura dal nodo del database principale. In questo modo è possibile migliorare la contemporaneità del database primario per le operazioni di scrittura e migliorare le prestazioni di lettura e scrittura in generale. Utilizzare le repliche di lettura consente inoltre di ridurre i costi in una configurazione Multi-AZ, poiché potrebbe essere possibile usare le istanze più piccole per l'istanza primaria, aggiungendo al contempo capacità di failover al cluster di database. Le repliche di lettura di Aurora offrono un ritardo di replica prossimo allo zero e consentono di creare fino a 15 repliche di lettura.

Considerazioni sull'affidabilità

Una considerazione importante per i database è l'elevata disponibilità e il ripristino di emergenza. Determinare i requisiti di RTO (Recovery Time Objective, obiettivo del tempo di ripristino) e RPO (Recovery Point Objective, obiettivo del punto di ripristino) dell'applicazione. Con Amazon Aurora, è possibile migliorare entrambi questi fattori.

Amazon Aurora riduce i tempi di riavvio del database a meno di 60 secondi nella maggior parte degli scenari di crash del database. Aurora, inoltre, consente di spostare la cache di buffer dal processo di database e la rende immediatamente disponibile al momento del riavvio. In rari scenari di guasti hardware e della zona di disponibilità, il ripristino viene gestito automaticamente dalla piattaforma del database.

Aurora è stato creato per fornire un RPO zero all'interno di una regione AWS, che costituisce un miglioramento importante nei sistemi di database locali. Aurora mantiene sei copie dei dati su tre zone di disponibilità e cerca automaticamente di effettuare un ripristino completo del database senza perdita di dati. Nel caso in cui i dati non fossero disponibili all'interno di Amazon Aurora, è possibile

ripristinarli da uno snapshot DB o eseguire un'operazione di ripristino temporizzato a una nuova istanza.

Considerazioni su costo e licenza

Il possesso e l'utilizzo di database comporta costi associati. Prima di pianificare una migrazione di database, è indispensabile effettuare un'analisi del costo totale di proprietà (TCO) della nuova piattaforma di database. La migrazione a una nuova piattaforma di database dovrebbe ridurre il costo totale di proprietà, fornendo alle applicazioni in uso caratteristiche simili o migliori. Se si utilizza un motore di database open source (MySQL, Postgres), i costi sono principalmente correlati all'hardware, alla gestione dei server e alle attività di gestione del database. Tuttavia, se si utilizza un motore di database commerciale (Oracle, SQL Server, DB2 ecc.), una notevole parte del costo è riferito alla licenza del database.

Poiché Aurora è disponibile a un decimo del costo dei motori commerciali, passando ad Aurora, molte applicazioni sono in grado di ridurre in modo significativo il TCO. Anche se si utilizza un motore open source come MySQL o Postgres, le alte prestazioni e le repliche di lettura dual-purpose di Aurora consentono di ottenere notevoli risparmi. Consultare la pagina dei prezzi di Amazon RDS per Aurora per ulteriori informazioni.[2]

Altre considerazioni sulla migrazione

Una volta considerata l'idoneità delle applicazioni, le prestazioni, il TCO e l'affidabilità, occorre valutare l'impatto di una migrazione alla nuova piattaforma.

Fare una stima del lavoro necessario per la modifica dei codici

È importante valutare la quantità delle modifiche di codici e schemi necessaria durante la migrazione del database ad Amazon Aurora. In caso di migrazione da database compatibili con MySQL, le modifiche di codici necessarie sono trascurabili. Tuttavia, quando la migrazione avviene da motori non MySQL, può essere necessario effettuare modifiche di schemi e codici. Lo Schema Conversion Tool AWS, illustrato più avanti in questo documento, può aiutare a valutare il lavoro.

Disponibilità delle applicazioni durante la migrazione

È possibile migrare ad Amazon Aurora adottando un approccio con tempi di inattività prevedibili con la propria applicazione oppure prossimi allo zero. L'approccio scelto dipende dalla dimensione del database e dai requisiti di

disponibilità delle applicazioni. In ogni caso, consigliamo di valutare l'impatto del processo di migrazione sull'applicazione e a livello aziendale prima di iniziare una migrazione di database. I paragrafi seguenti descrivono in dettaglio entrambi gli approcci.

Pianificazione del processo di migrazione di database

La sezione precedente ha trattato alcune delle principali considerazioni di cui tenere conto durante la migrazione dei database ad Amazon Aurora. Una volta stabilito che Aurora è la soluzione ideale per l'applicazione, è necessario decidere l'approccio di migrazione preliminare e creare un piano di migrazione del database.

Migrazione omogenea

Se il database di origine è un database compatibile con MySQL 5.6 (MySQL, MariaDB, Percona, ecc.), la migrazione ad Aurora è molto semplice.

Migrazione omogenea con tempi di inattività

Se l'applicazione utilizzata è in grado di gestire un periodo di tempi di inattività prevedibile durante le ore non di punta, questo è l'approccio più semplice e vivamente consigliato. La maggior parte dei progetti di migrazione di database rientrano in questa categoria poiché la maggior parte delle applicazioni dispone già di una finestra di manutenzione ben definita. Per migrare i database con tempi di inattività sono disponibili le seguenti opzioni.

- **Migrazione snapshot RDS** – Se il database di origine è in esecuzione su Amazon RDS MySQL 5.6, è possibile semplicemente migrare uno snapshot di quel database ad Amazon Aurora. Per migrazioni con tempi di inattività è necessario interrompere l'applicazione o interrompere la scrittura sul database mentre lo snapshot e la migrazione sono in corso. Il tempo necessario per eseguire la migrazione dipende principalmente dalle dimensioni del database e può essere stabilito prima della migrazione di produzione eseguendo una migrazione di prova. L'opzione di migrazione snapshot è spiegata nella sezione Migrazione snapshot RDS. È inoltre possibile eseguire una migrazione con tempi di inattività prossimi allo zero utilizzando la migrazione dello snapshot e la replica dei log binari, descritte nella sezione successiva.

- **Migrazione con strumenti MySQL nativi.** È possibile utilizzare gli strumenti nativi MySQL per migrare dati e schema ad Aurora. Questa opzione è di grande valore quando si necessita di un maggiore controllo sul processo di migrazione di database, quando si preferisce usare strumenti nativi MySQL e altri metodi di migrazione non funzionano altrettanto bene per il caso specifico. Per le best practice relative a questa opzione, scaricare il whitepaper Amazon RDS for Aurora Export/Import Performance Best Practices.[3]

- **Migrazione utilizzando AWS Database Migration Service (AWS DMS).** La migrazione unica utilizzando AWS DMS è un altro strumento per trasferire il database di origine ad Amazon Aurora. Prima di poter utilizzare AWS DMS, è necessario copiare lo schema del database dall'origine alla destinazione utilizzando gli strumenti nativi MySQL . Per il processo dettagliato, consultare la sezione relativa alla migrazione dei dati. Utilizzare AWS DMS è un'ottima opzione se non si ha esperienza degli strumenti nativi MySQL.

Migrazione omogenea con tempi di inattività prossimi a zero

In alcuni scenari l'opzione migliore potrebbe essere effettuare una migrazione del database ad Aurora con tempi di inattività minimi. Di seguito sono riportati due esempi:

- quando il database è relativamente grande e il tempo di migrazione utilizzando le opzioni con tempi di inattività supera la finestra di manutenzione dell'applicazione
- quando si desidera eseguire i database di origine e di destinazione in parallelo per l'esecuzione di test

In questi casi, è possibile replicare le modifiche dal database MySQL di origine ad Aurora in tempo reale utilizzando la replica. Sono disponibili due opzioni:

- **Migrazione con tempi di inattività prossimi a zero utilizzando la replica binlog di MySQL.** Amazon Aurora supporta la tradizionale replica dei log binari di MySQL. Se si utilizza il database MySQL, probabilmente si possiede familiarità con la configurazione di replica binlog classica. Se questo è il caso e si desidera un maggiore controllo sul processo di migrazione, la procedura più semplice per migrare ad Aurora consisterà nel caricare una volta il database utilizzando gli strumenti nativi e nell'effettuare la replica dei log binari.

- **Migrazione con tempi di inattività prossimi a zero utilizzando AWS Database Migration Service (AWS DMS).** Oltre a supportare la migrazione in una volta sola, AWS DMS, supporta anche la replica dei dati in tempo reale utilizzando l'acquisizione dei dati delle modifiche (CDC) dall'origine alla destinazione. AWS DMS si occupa delle complessità correlate alla copia dei dati iniziali, della configurazione delle istanze di replica e del monitoraggio della replica. Dopo la migrazione iniziale del database, il database di destinazione rimane sincronizzato con quello di origine per tutto il tempo desiderato. Se non si ha familiarità con la replica dei log binari, la migrazione con AWS DMS è l'alternativa migliore per una migrazione ad Amazon Aurora omogenea con tempi di inattività prossimi a zero. Consultare la sezione Introduzione e approccio generale ad AWS DMS.
- **Migrazione con tempi di inattività prossimi a zero utilizzando la migrazione snapshot RDS e la replica binlog di MySQL.** Se il database di origine è in esecuzione su Amazon RDS MySQL 5.6, è possibile migrare uno snapshot di quel database ad Amazon Aurora e avviare la replica binlog tra l'origine e la destinazione dopo la migrazione dello snapshot. Per ulteriori informazioni su questa opzione di migrazione, consultare Replica con Amazon Aurora nella *Guida per l'utente di Amazon RDS*.[4]

Migrazione eterogenea

Se si desidera migrare un database non compatibile con MySQL (Oracle, SQL Server, PostgreSQL ecc.) ad Amazon Aurora, diverse opzioni di migrazione consentono di eseguire questa operazione in modo rapido e semplice.

Migrazione dello schema

La migrazione dello schema da un database non compatibile con MySQL ad Amazon Aurora può essere effettuata utilizzando lo Schema Conversion Tool di AWS. Questo strumento è un'applicazione desktop che consente di convertire lo schema da un database Oracle, Microsoft SQL Server o PostgreSQL a un'istanza di database MySQL di Amazon RDS o a un cluster Amazon Aurora DB. Nei casi in cui lo schema del database di origine non possa essere convertito automaticamente e completamente, lo Schema Conversion Tool AWS fornisce linee guida su come creare lo schema equivalente nel database target di Amazon RDS. Per ulteriori informazioni, consultare la sezione Migrazione dello schema del database.

Migrazione dei dati

Oltre a supportare migrazioni omogenee con tempi di inattività prossimi a zero, AWS Database Migration Service (AWS DMS) supporta inoltre la replica continua in database eterogenei; questa è l'opzione preferita per spostare il database di origine al database di destinazione, sia per migrazioni con tempi di inattività che per migrazioni con tempi di inattività prossimi a zero. Dopo aver avviato la migrazione, AWS DMS gestisce tutte le complessità del processo di migrazione come la trasformazione del tipo di dati, la compressione e il trasferimento parallelo (per velocizzare il trasferimento dei dati) garantendo al contempo che le modifiche dei dati al database di origine che si verificano durante il processo di migrazione vengano automaticamente replicate nella destinazione.

Oltre a AWS DMS, è possibile utilizzare vari strumenti di terze parti come Attunity Replicate, Tungsten Replicator, Oracle Golden Gate, ecc. per migrare i propri dati ad Amazon Aurora. Indipendentemente dallo strumento scelto, prendere in considerazione le prestazioni e i costi di licenza prima di finalizzare il set di strumenti per la migrazione.

Migrazione di database di grandi dimensioni ad Amazon Aurora

La migrazione di set di dati di grandi dimensioni presenta sfide uniche in ogni progetto di migrazione di database. Molti progetti di migrazione di database di grandi dimensioni di successo utilizzano una combinazione delle seguenti strategie:

- **Migrazione con la replica continua.** I database di grandi dimensioni in genere hanno requisiti di tempi di inattività estesi durante il trasferimento di dati dall'origine alla destinazione. Per ridurre i tempi di inattività, è possibile caricare prima i dati baseline dall'origine alla destinazione e quindi abilitare la replica (utilizzando strumenti nativi di MySQL, AWS DMS o strumenti di terze parti) per recuperare le modifiche.
- **Copiare le tabelle statiche per prime.** Se il database si basa su tabelle statiche di grandi dimensioni con dati di riferimento, è possibile eseguire la migrazione di questi tipi di tabelle al database di destinazione prima di migrare il set di dati attivo. È possibile utilizzare AWS DMS per copiare le tabelle in modo selettivo o esportare e importare le tabelle manualmente.

- **Migrazione multifase.** La migrazione di database di grandi dimensioni con migliaia di tabelle può essere suddivisa in più fasi. Ad esempio, è possibile spostare un set di tabelle senza query cross join ogni fine settimana finché il database di origine non è completamente migrato al database di destinazione. Si noti che, per completare questa operazione, è necessario modificare l'applicazione affinché si connetta a due database contemporaneamente mentre il set di dati è su due nodi distinti. Anche se non si tratta di un modello di migrazione comune, è comunque un'opzione.
- **Pulizia del database.** Molti database di grandi dimensioni contengono dati e tabelle che rimangono inutilizzati. In molti casi, gli sviluppatori e gli amministratori di database conservano copie di backup delle tabelle nello stesso database, oppure semplicemente dimenticano di cancellare le tabelle inutilizzate. Indipendentemente dal motivo, un progetto di migrazione di database offre l'opportunità di pulire il database esistente prima della migrazione. Se alcune tabelle non vengono utilizzate, è possibile cancellarle o archiviarle in un altro database. È inoltre possibile cancellare i dati obsoleti dalle tabelle di grandi dimensioni oppure archiviare tali dati su flat file.

Consolidamento partizioni e shard su Amazon Aurora

Se si stanno eseguendo più partizioni shard o funzionali del database per ottenere prestazioni elevate, è possibile consolidare queste partizioni o shard su un solo database Aurora. Una singola istanza Amazon Aurora può essere estesa fino a 64 TB, supporta migliaia di tabelle e un numero di letture e scritture notevolmente superiore rispetto a un database MySQL standard. Il consolidamento di queste partizioni su una singola istanza di Aurora non solo consente di ridurre il costo totale di proprietà e semplificare la gestione del database, ma anche di migliorare sensibilmente le prestazioni delle query tra partizioni.

- **Partizioni funzionali.** L'esecuzione di partizioni funzionali significa dedicare nodi diversi ad attività diverse. Ad esempio, in un'applicazione di e-commerce, è possibile predisporre un nodo di database per il catalogo dei prodotti e un altro nodo di database per acquisire ed elaborare gli ordini. Di conseguenza, queste partizioni hanno solitamente schemi distinti e non sovrapposti.

  Strategia di consolidamento. Eseguire la migrazione di ciascuna partizione funzionale all'istanza di destinazione di Aurora come schema distinto. Se il database di origine è compatibile con i requisiti di MySQL,

utilizzare gli strumenti nativi di MySQL per migrare lo schema e quindi utilizzare AWS DMS per migrare i dati, in una sola volta o in modo continuo utilizzando la replica. Se il database di origine non è compatibile MySQL, utilizzare AWS Schema Conversion Tool per migrare gli schemi ad Aurora e utilizzare AWS DMS per il caricamento in una sola volta o la replica continua.

- **Shard di dati.** Se si ha lo stesso schema con di set di dati distinti su più nodi, si sta utilizzando lo sharding (partizionamento) del database. Ad esempio, un servizio di blog a traffico elevato può distribuire l'attività e i dati degli utenti su più shard del database mantenendo lo stesso schema di tabelle.

  Strategia di consolidamento. Poiché tutti gli shard condividono lo stesso schema del database, è necessario creare lo schema di destinazione solo una volta. Se si sta utilizzando un database compatibile con MySQL, utilizzare gli strumenti nativi per migrare lo schema del database ad Aurora. Se si sta usando un database MySQL, utilizzare AWS Schema Conversion Tool per migrare lo schema del database ad Aurora. Dopo la migrazione dello schema del database, è consigliabile arrestare le operazioni di scrittura sugli shard del database e utilizzare gli strumenti nativi o un caricamento di dati singolo di AWS DMS per eseguire la migrazione di un singolo shard ad Aurora. Se le operazioni di scrittura sull'applicazione non possono essere interrotte per un lungo periodo, è possibile continuare a utilizzare AWS DMS con la replica, ma solo dopo corretta pianificazione e testing.

Panoramica delle opzioni di migrazione

Tipo di database di origine	Migrazione con tempi di inattività	Migrazione con tempi di inattività prossimi a zero
Amazon RDS MySQL	Opzione 1: migrazione dello snapshot RDS Opzione 2: migrazione manuale di utilizzando gli strumenti nativi* Opzione 3: migrazione dello schema con gli strumenti nativi e caricamento di dati con AWS DMS	Opzione 1: Migrazione utilizzando strumenti nativi + replica binlog Opzione 2: Migrazione snapshot RDS + replica binlog Opzione 3: Migrazione schema utilizzando gli strumenti nativi + AWS DMS per il trasferimento dei dati

Tipo di database di origine	Migrazione con tempi di inattività	Migrazione con tempi di inattività prossimi a zero
MySQL Amazon EC2 o in loco	Opzione 1: Migrazione utilizzando strumenti nativi Opzione 2: Migrazione schema con strumenti nativi + AWS DMS per il caricamento dei dati	Opzione 1: Migrazione utilizzando strumenti nativi + replica binlog Opzione 2: Migrazione schema utilizzando gli strumenti nativi + AWS DMS per il trasferimento dei dati
Server Oracle/SQL	Opzione 1: AWS Schema Conversion Tool + AWS DMS (consigliato) Opzione 2: Strumento manuale o di terze parti per la conversione dello schema + caricamento dati manuale o di terze parti nel target	Opzione 1: AWS Schema Conversion Tool + AWS DMS (consigliato) Opzione 2: Strumento manuale o di terze parti per la conversione dello schema + caricamento dati manuale o di terze parti nel target + strumento di terze parti per la replica
Altri database non MySQL	Opzione: Strumento manuale o di terze parti per la conversione dello schema + caricamento dati manuale o di terze parti nel target	Opzione: Strumento manuale o di terze parti per la conversione dello schema + caricamento dati manuale o di terze parti nel target + strumento di terze parti per la replica (GoldenGate, ecc.)

* Strumenti nativi Mysql: mysqldump, SELECT INTO OUTFILE, strumenti di terze parti come mydumper/myloader

Migrazione dello snapshot RDS

Per utilizzare la funzione di migrazione di snapshot di RDS per passare ad Aurora, il database MySQL deve essere in esecuzione su Amazon RDS MySQL 5.6 ed è necessario effettuare uno snapshot RDS del database. Questo metodo di migrazione non funziona con database locali o database in esecuzione su Amazon Elastic Compute Cloud (Amazon EC2). Inoltre, se si utilizza il database Amazon RDS MySQL di una versione precedente a 5.6, è necessario eseguire l'aggiornamento a 5.6 come prerequisito.

Il principale vantaggio di questo metodo di migrazione è che è il più semplice e richiede il minor numero di passaggi. In particolare, insieme a tutti i dati del database, trasferisce anche tutti gli oggetti dello schema, gli indici secondari e le procedure salvate.

Durante la migrazione dello snapshot senza replica dei log binari, il database di origine deve essere offline o in modalità di sola lettura (in modo che non vengano eseguite modifiche al database di origine durante la migrazione). Per stimare i tempi di inattività, è sufficiente utilizzare l'attuale snapshot del database per

eseguire una migrazione di prova. Se il tempo di migrazione risulta adatto alle proprie esigenze di tempi di inattività, questo potrebbe essere il metodo migliore da utilizzare nel caso specifico. Si noti che, in alcuni casi, la migrazione con AWS DMS o con gli strumenti di migrazione nativi può risultare più veloce rispetto alla migrazione dello snapshot.

Se tempi di inattività prolungati non sono accettabili, può essere possibile effettuare una migrazione con tempi di inattività prossimi a zero trasferendo prima uno snapshot del database RDS ad Amazon Aurora mentre viene aggiornato il database di origine, e aggiornandolo poi mediante la replica dei file binari da MySQL ad Aurora.

È possibile eseguire la migrazione di uno snapshot di database manuale o automatizzato. La procedura generale è indicata di seguito:

1. Determinare la quantità di spazio necessaria per migrare l'istanza Amazon RDS MySQL 5.6 a un cluster di database Aurora. Per ulteriori informazioni, consultare la sezione seguente.
2. Utilizzare la console di Amazon RDS per creare lo snapshot nella regione in cui è ubicata l'istanza di Amazon RDS MySQL 5.6.
3. Utilizzare la funzione di **Migrate Database** nella console per creare un cluster di database Amazon Aurora, che sarà popolato utilizzando gli snapshot di database dell'istanza originale di MySQL 5.6.

Nota: la conversione di alcune tabelle MyISAM potrebbe causare errori e richiedere modifiche manuali. Ad esempio, il motore di InnoDB non consente un campo autoincrementale come parte di una chiave composta. Inoltre, gli indici spaziali non sono attualmente supportati.

Stima dei requisiti di spazio per la migrazione snapshot

Quando si esegue la migrazione di uno snapshot di un'istanza database MySQL a un cluster di database Aurora, Aurora usa un volume Amazon Elastic Block Store (Amazon EBS) per formattare i dati dallo snapshot prima di effettuarne la migrazione. Ci sono alcuni casi in cui è necessario spazio aggiuntivo per formattare i dati per la migrazione. Le due caratteristiche che potrebbero causare problemi di spazio durante la migrazione sono le tabelle MyISAM e l'uso dell'opzione `ROW_FORMAT = COMPRESSED`. Se non si utilizza una di queste funzioni nel database di origine, è possibile ignorare questa sezione poiché non dovrebbero esserci problemi di spazio. Durante la migrazione, le tabelle MyISAM

vengono convertite a InnoDB ed eventuali tabelle compresse vengono decompresse. Di conseguenza, deve esservi spazio sufficiente per le copie aggiuntive di tali tabelle.

Le dimensioni del volume di migrazione dipendono dalle dimensioni assegnate al database MySQL di origine da cui è stato effettuato lo snapshot. Pertanto, se si dispone di tabelle MyISAM o compresse che costituiscono una piccola percentuale delle dimensioni complessive del database e vi è spazio disponibile nel database originale, la migrazione dovrebbe riuscire senza incontrare problemi di spazio. Tuttavia, se il database originale non dispone di spazio sufficiente per memorizzare una copia di tabelle MyISAM convertite e un'altra copia (non compressa) delle tabelle, il volume di migrazione non sarà sufficiente. In questo caso, è necessario modificare il database Amazon RDS MySQL di origine aumentando lo spazio ad esso allocato per consentire le copie aggiuntive di queste tabelle, effettuare un nuovo snapshot del database e quindi migrare il nuovo snapshot.

Alla migrazione dei dati nel cluster di database, osservare le seguenti linee guida e limitazioni:

- Anche se Amazon Aurora supporta fino a 64 TB di storage, il processo di migrazione di uno snapshot a un cluster di database Aurora è limitato dalle dimensioni del volume Amazon EBS dello snapshot, ovvero 6 TB al massimo.
- Le tabelle non-MyISAM nel database di origine possono avere una dimensione fino a 6 TB. Tuttavia, a causa di requisiti di spazio aggiuntivi durante la conversione, assicurarsi che nessuna delle tabelle MyISAM e compresse di cui si effettua la migrazione dalla propria istanza di database MySQL superi i 3 TB.

È possibile modificare il proprio schema del database (convertire le tabelle MyISAM e InnoDB e rimuovere `ROW_FORMAT=COMPRESSED`) prima di migrare ad Amazon Aurora. Questo può essere utile nei seguenti casi:

- Per velocizzare il processo di migrazione.
- Quando non si è sicuri di quanto spazio occorre.
- Si è tentata la migrazione dei dati ma non è riuscita a causa di mancanza di spazio.

Assicurasi di non apportare queste modifiche nel database di produzione Amazon RDS MySQL, ma su un'istanza di database che è stata ripristinata dallo snapshot di produzione. Per ulteriori informazioni su questa operazione, vedere Riduzione della quantità di spazio richiesto per la migrazione dei dati ad Amazon Aurora nella *Guida per l'utente di Amazon RDS*.[5]

Migrazione di uno snapshot di database utilizzando la console

È possibile eseguire la migrazione di uno snapshot di database di un'istanza database di Amazon RDS MySQL per creare un cluster di database Aurora. Il nuovo cluster di database viene popolato con i dati provenienti dall'istanza Amazon RDS MySQL DB originale. Gli snapshot di database devono essere stati effettuati da un'istanza database di RDS che utilizza MySQL 5.6 e non devono essere crittografati. Per ulteriori informazioni su come creare uno snapshot di database, vedere Creazione di uno snapshot di database nella *Guida per l'utente di Amazon RDS*.[6]

Se lo snapshot di database non è nella regione in cui si desidera collocare il cluster di database Aurora, è possibile usare la console di Amazon RDS per copiare lo snapshot di database in quella regione. Per ulteriori informazioni sulla copia di uno snapshot di database, vedere Copia di uno snapshot di database in *Guida per l'utente di Amazon RDS*.[7]

Per eseguire la migrazione di uno snapshot di database MySQL 5.6 utilizzando la console, procedere come segue:

1. Accedere alla console di gestione AWS e aprire la console di Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere **Snapshots**.
3. Nella pagina **Snapshots**, scegliere lo snapshot Amazon RDS MySQL 5.6 che si desidera migrare in un cluster di database Aurora.
4. Scegliere **Migrate Database**.
5. Nella pagina **Migrate Database**, specificare i valori che soddisfano i requisiti di elaborazione e di ambiente come illustrato di seguito. Per una descrizione di queste opzioni, vedere Migrazione di uno snapshot di database mediante console nella *Guida per l'utente di Amazon RDS*.[8]

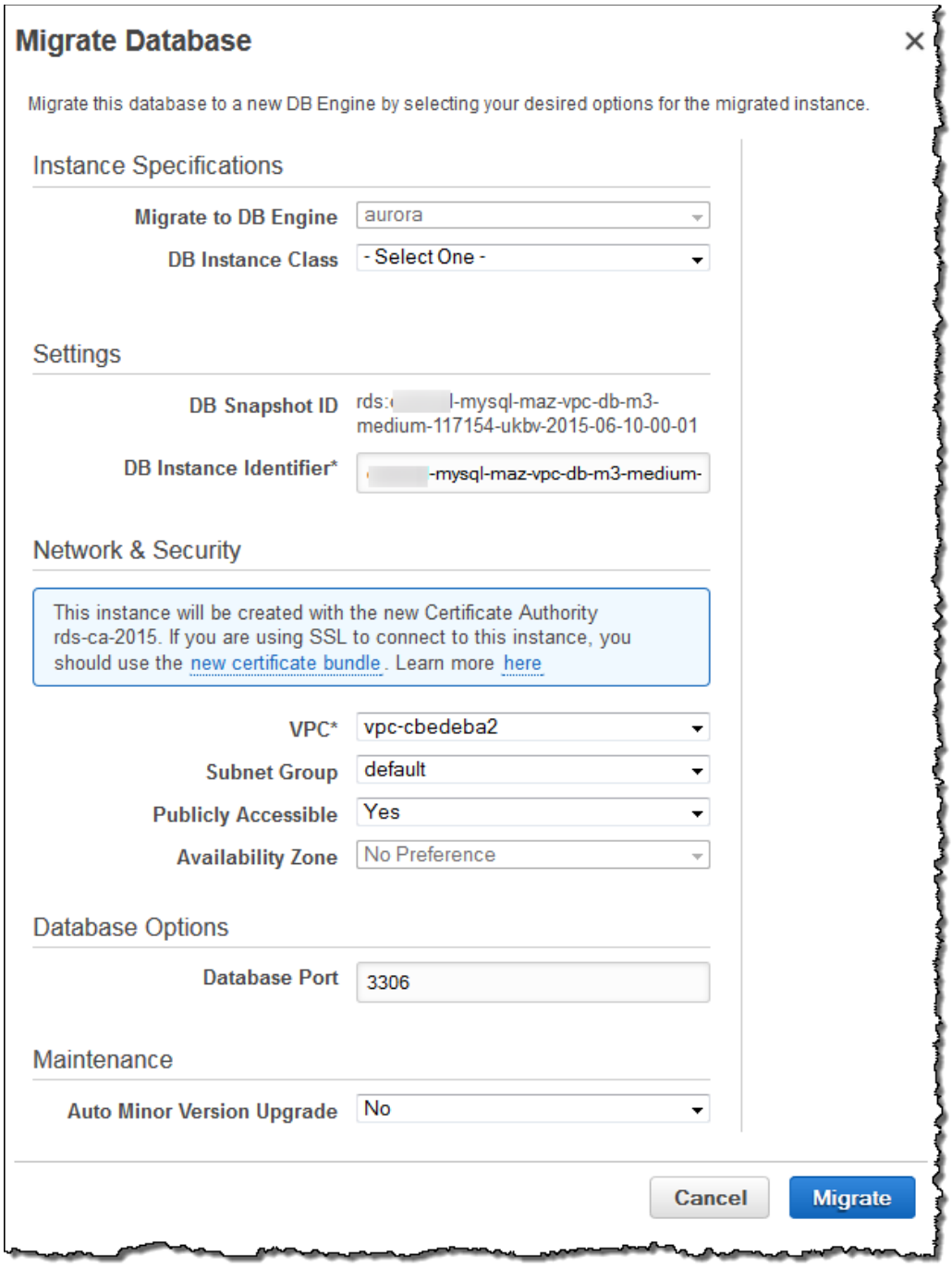

Figura 2: Migrazione di snapshot

6. Fare clic su **Migrate** per migrare lo snapshot di database.

Nell'elenco delle istanze, fare clic sull'icona freccia appropriata per visualizzare i dettagli del cluster di database e monitorare l'avanzamento del processo di migrazione. In questo pannello vengono visualizzati i dettagli del cluster endpoint utilizzato per connettersi all'istanza principale del cluster di database.

Per ulteriori informazioni sulla connessione a un cluster di database Amazon Aurora, consultare la pagina Connessione a un cluster di database Amazon Aurora nella *Guida per l'utente di Amazon RDS*.[9]

Migrazione dello schema del database

La migrazione dello snapshot di database di RDS migra sia l'intero schema sia i dati alla nuova istanza di Aurora. Tuttavia, se la posizione del database di origine o i requisiti di tempo di attività dell'applicazione non consentono l'uso della migrazione dello snapshot, è necessario innanzitutto migrare lo schema del database dal database di origine al database di destinazione prima di poter spostare i dati effettivi. Uno schema di database è una struttura che rappresenta la visualizzazione logica dell'intero database e in genere include quanto segue:

- Oggetti di storage di database: tabelle, colonne, restrizioni, indici, sequenze, tipi definiti dall'utente e tipi di dati
- Oggetti di codice del database: funzioni, procedure, package, trigger, viste, viste materializzate, eventi, funzioni scalari SQL, funzioni inline SQL, funzioni tabelle SQL, attributi, variabili, costanti, tipi di tabella, tipi pubblici, tipi privati, cursori, eccezioni, parametri e altri oggetti

Nella maggior parte dei casi, lo schema del database rimane relativamente statico, quindi non sono necessari tempi di inattività durante la fase di migrazione dello schema del database. Lo schema dal database di origine può essere estratto mentre il database di origine è in esecuzione senza compromettere le prestazioni. Se l'applicazione o gli sviluppatori apportano modifiche frequenti allo schema del database, assicurarsi che le modifiche siano interrotte mentre la migrazione è in corso o vengano considerate durante il processo di migrazione dello schema.

A seconda del tipo di database di origine, è possibile usare le tecniche descritte nelle sezioni seguenti per migrare lo schema del database. Come prerequisito per la migrazione degli schemi, è necessario disporre di un database Aurora di destinazione creato e disponibile.

Migrazione schema omogenea

Se il database di origine è compatibile con MySQL 5.6 ed è in esecuzione su Amazon RDS, Amazon EC2, o al di fuori di AWS, è possibile utilizzare gli strumenti nativi di MySQL per esportare e importare lo schema.

- **Esportazione di schema di database.** È possibile utilizzare la client utility mysqldump per esportare lo schema del database. Per eseguire questa utility, è necessario connettersi al database di origine e reindirizzare l'output del comando `mysqldump` a un flat file. L'opzione `–no-data` garantisce che venga esportato solo lo schema del database e non le tabelle di dati. Per il riferimento completo sui comandi `mysqldump` vedere mysqldump—Un programma di backup del database.[10]

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

- **Importazione di schema del database in Aurora.** Per importare lo schema alla propria istanza di Aurora, connettersi al database Aurora da un client a riga di comando MySQL (o un corrispondente client Windows) e indirizzare i contenuti del file di esportazione in MySQL.

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

Nota:

- Se il database di origine contiene procedure memorizzate, trigger e visualizzazioni, occorre rimuovere la sintassi `DEFINER` dal file dump. Un semplice comando Perl è indicato di seguito. In questo modo vengono creati tutti gli allarmi, le visualizzazioni e le procedure memorizzate con l'attuale utente connesso come `DEFINER`. Assicurasi di valutare le implicazioni di sicurezza che può comportare.

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

- Amazon Aurora supporta solo le tabelle InnoDB. Se il database di origine contiene tabelle MyISAM, Aurora cambia automaticamente il motore di InnoDB quando viene eseguito il comando `CREATE TABLE`.

- Amazon Aurora non supporta le tabelle compresse (ovvero le tabelle create con `ROW_FORMAT=COMPRESSED`). Se si dispone di tabelle compresse nel database di origine, Aurora cambia automaticamente da `ROW_FORMAT` a `COMPACT` quando viene eseguito il comando `CREATE TABLE`.

Una volta importato correttamente lo schema in Amazon Aurora dal database di origine compatibile con MySQL 5.6, è possibile copiare i dati effettivi dall'origine alla destinazione. Per ulteriori informazioni, consultare Introduzione e approccio generale a AWS DMS più avanti in questo documento.

Migrazione schema eterogenea

Se il database di origine non è compatibile con MySQL, è necessario convertire lo schema in un formato compatibile con Amazon Aurora. La conversione dello schema da un motore di database a un altro un'operazione di non poco conto e può comportare la riscrittura di alcune parti del database e il codice dell'applicazione. Sono disponibili due opzioni principali per la conversione e la migrazione dello schema ad Amazon Aurora:

- **AWS Schema Conversion Tool.** AWS Schema Conversion Tool semplifica le migrazioni di database eterogenee consentendo la conversione automatica dello schema del database di origine e la maggior parte del codice personalizzato, incluse visualizzazioni, procedure memorizzate e funzioni, a un formato compatibile con il database di destinazione.[11] Eventuali codici che non possono essere convertiti automaticamente vengono contrassegnati in modo chiaro per poter intervenire manualmente. È possibile usare questo strumento per convertire i database di origine in esecuzione su Oracle o Microsoft SQL Server a un database di destinazione Amazon Aurora, MySQL oppure PostgreSQL in Amazon RDS o Amazon EC2. L'utilizzo di AWS Schema Conversion Tool per convertire lo schema Oracle, SQL Server o PostgreSQL a un formato compatibile con Aurora è il metodo preferito.
- **Migrazione manuale dello schema e strumenti di terze parti.** Se il database di origine non è Oracle, SQL Server o PostgreSQL, è possibile eseguire manualmente la migrazione dello schema del database di origine ad Aurora o utilizzare gli strumenti di terze parti per la migrazione dello schema in un formato compatibile con MySQL 5.6. La migrazione manuale dello schema può essere un processo piuttosto complesso in base alle dimensioni e alla complessità dello schema di origine. Nella maggior parte dei casi, tuttavia, la conversione manuale dello schema vale lo sforzo

considerato il risparmio sui costi, i vantaggi in prestazioni e altri miglioramenti possibili con Amazon Aurora.

Migrazione schema utilizzando AWS Schema Conversion Tool

AWS Schema Conversion Tool offre un'interfaccia utente basata sul progetto per convertire automaticamente lo schema del database di origine in un formato compatibile con Amazon Aurora. È consigliabile utilizzare AWS Schema Conversion Tool per valutare la migrazione di database e per la migrazione pilota prima della migrazione di produzione effettiva.

La seguente è una descrizione generale delle fasi di utilizzo di AWS Schema Conversion Tool. Per istruzioni dettagliate, vedere la sezione Getting Started della *Guida per l'utente di AWS Schema Conversion Tool*.[12]

1. In primo luogo, installare lo strumento. AWS Schema Conversion Tool è disponibile per Microsoft Windows, Mac OS X, Ubuntu Linux e Fedora Linux.

   Istruzioni dettagliate per il download a l'installazione sono disponibili nella sezione installazione e aggiornamento della guida per l'utente.[13] La posizione di installazione di AWS Schema Conversion Tool è importante. Per poter convertire e applicare lo schema lo strumento deve connettersi direttamente a entrambi i database di origine e di destinazione. Assicurarsi che il desktop in cui si installa AWS Schema Conversion Tool disponga di connettività di rete con i database di origine e di destinazione.

2. Installare i driver JDBC. AWS Schema Conversion Tool utilizza i driver JDBC per la connessione al database di origine e di destinazione. Per utilizzare questo strumento, è necessario scaricare i driver JDBC per il desktop locale. Per le istruzioni per il download dei driver, visitare Driver database richiesti nella *Guida per l'utente di AWS Schema Conversion Tool User Guide*.[14] Inoltre, controllare le istruzioni per l'impostazione dei driver JDCB per i diversi motori di database su Forum AWS per l'utilizzo di AWS Schema Conversion Tool.[15]

3. Creare un database di destinazione. Crea un database di destinazione Amazon Aurora. Per istruzioni su come creare un database Amazon Aurora, consultare l'argomento relativo alla creazione di un cluster di database Amazon Aurora nella *Guida per l'utente di Amazon RDS*.[16]

4. Aprire AWS Schema Conversion Tool e avviare la **New Project Wizard**.

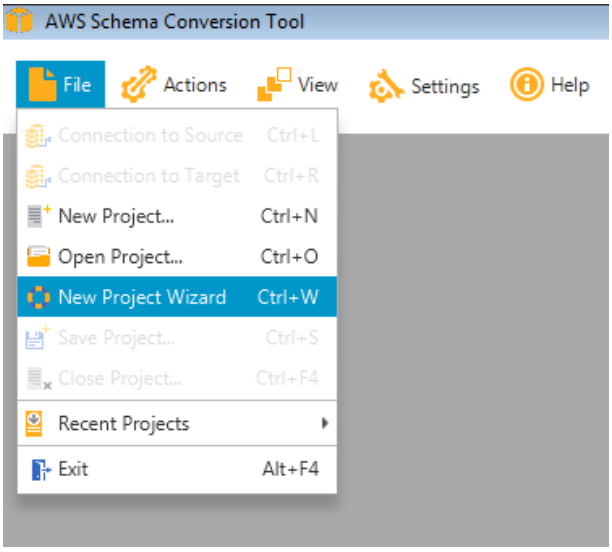

Figura 3: Creazione nuovo progetto AWS Schema Conversion Tool

5. Configurare il database di origine e testare la connettività tra AWS Schema Conversion Tool e il database di origine. Il database di origine deve essere raggiungibile dal desktop perché possa funzionare, perciò accertarsi di disporre della rete e delle impostazioni del firewall appropriate.

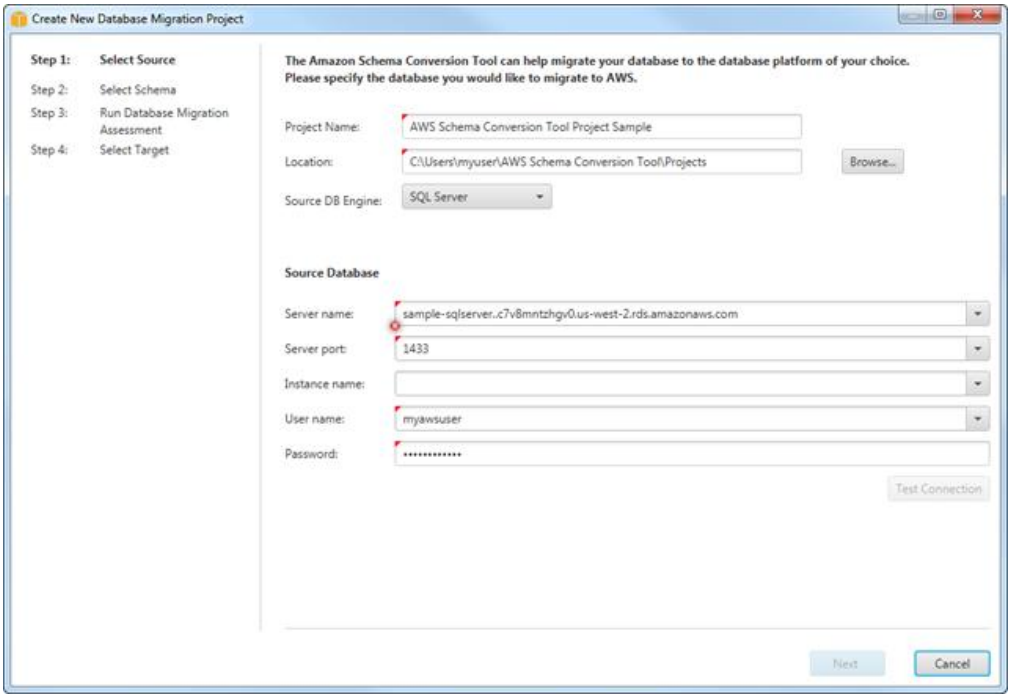

Figura 4: Procedura guidata creazione nuovo progetto di migrazione database

6. Nella schermata successiva, selezionare lo schema del database di origine che si desidera convertire ad Amazon Aurora.

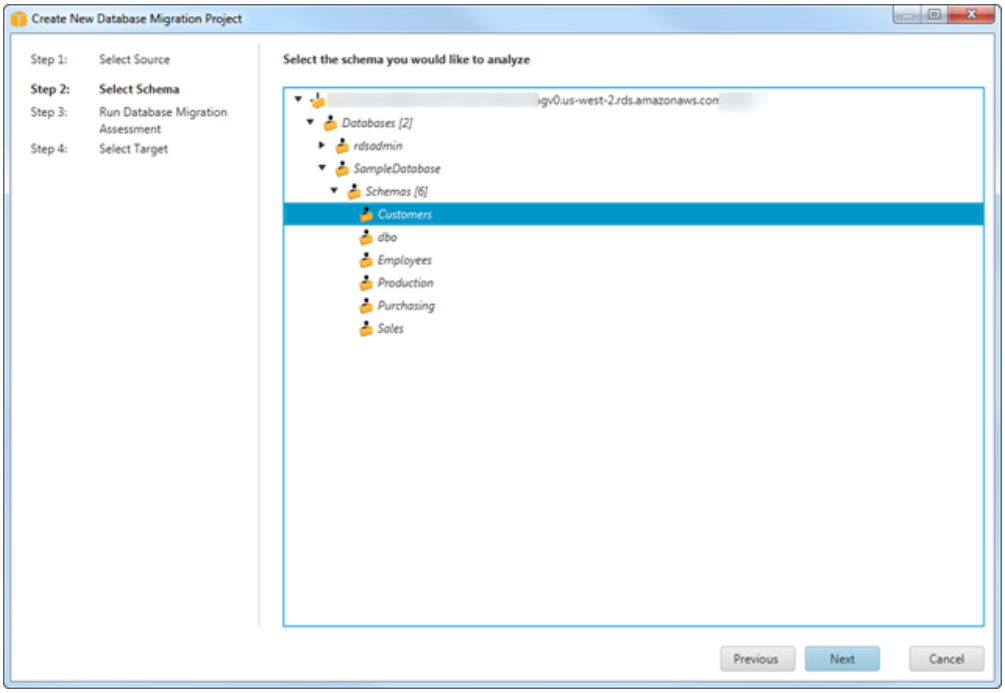

Figura 5: Procedura guidata selezione fase di migrazione dello schema

7. Eseguire il report di valutazione della migrazione del database. Questo report fornisce informazioni importanti relative alla conversione dello schema del database di origine per l'istanza Amazon Aurora di destinazione. Riepiloga tutte le attività di conversione dello schema e i dettagli delle azioni per le parti dello schema che non possono essere convertite automaticamente ad Aurora. Il report include inoltre le stime della quantità di lavoro che sarà necessario per scrivere il codice equivalente nel proprio database di destinazione che non è stato possibile convertire automaticamente.

   Fare clic su **Next** per configurare il database di destinazione. È possibile visualizzare nuovamente il report di migrazione in un secondo momento.

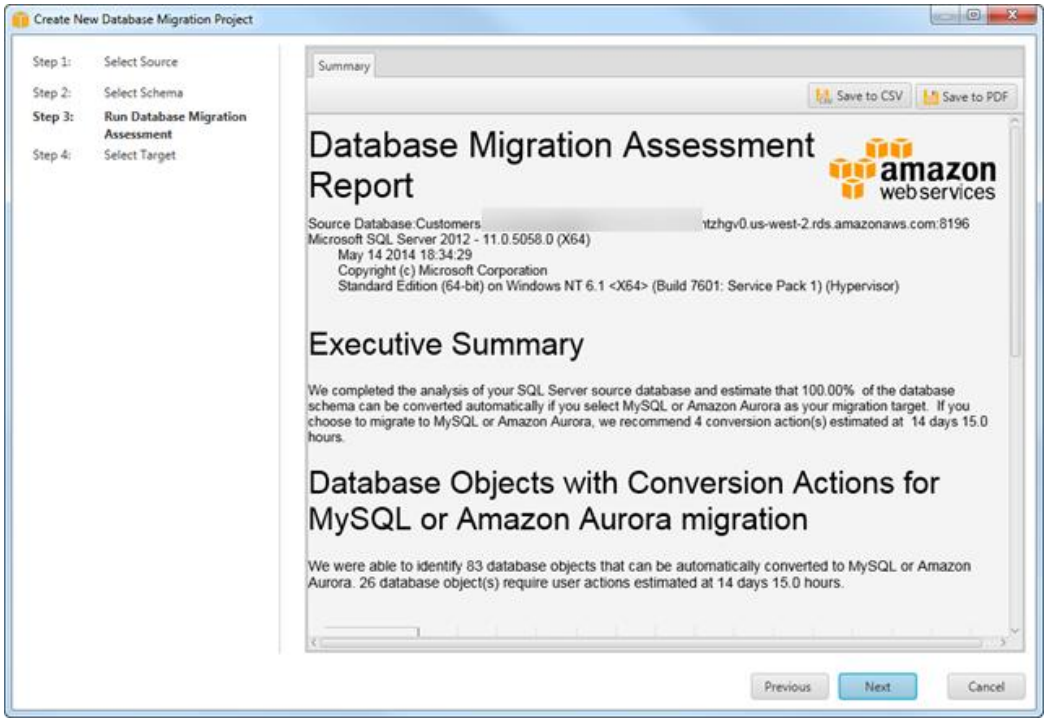

Figura 6: Report migrazione

8. Configurare il database Amazon Aurora target e testare la connettività tra AWS Schema Conversion Tool e il database di origine. Il database di destinazione deve essere raggiungibile dal desktop perché possa funzionare, perciò accertarsi di disporre della rete e delle impostazioni del firewall appropriate. Fare clic su **Finish** per aprire la finestra del progetto.
9. Una volta aperta la finestra del progetto, è già stata stabilita una connessione ai database di origine e di destinazione ed è il momento di valutare il report dettagliato ed effettuare la migrazione dello schema.
10. Nel riquadro sinistro che mostra lo schema dal database di origine, scegliere un oggetto dello schema per il quale creare un report di valutazione. Fare clic con il pulsante destro del mouse sull'oggetto e scegliere **Create Report**.

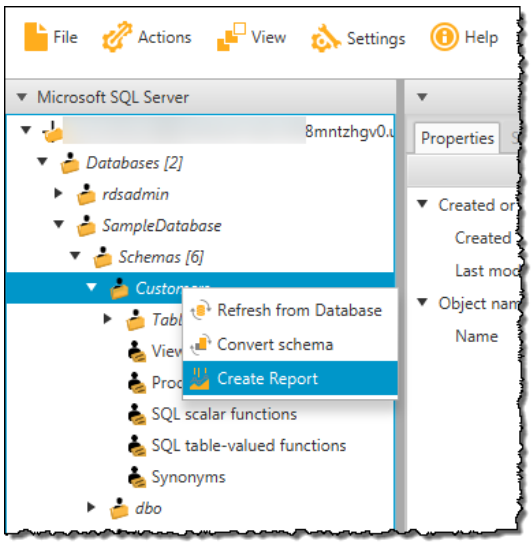

Figura 7: Creazione report di migrazione

La scheda **Summary** visualizza la sintesi delle informazioni del report di valutazione della migrazione del database. Mostra gli elementi che sono stati convertiti automaticamente e gli elementi che non hanno potuto essere convertiti automaticamente.

Per gli elementi dello schema che non hanno potuto essere convertiti automaticamente al motore di database di destinazione, il riepilogo include una stima delle risorse che sono necessarie per creare uno schema equivalente al database di origine all'interno dell'istanza di database di destinazione. Il rapporto suddivide il tempo stimato per convertire questi elementi dello schema come segue:

- **Semplice**: operazioni che possono essere completate in meno di 1 ora.
- **Medio** - Azioni più complesse e che possono essere completate da una a quattro ore.
- **Significativo** - Azioni che sono molto complesse e il cui completamento richiede oltre quattro ore.

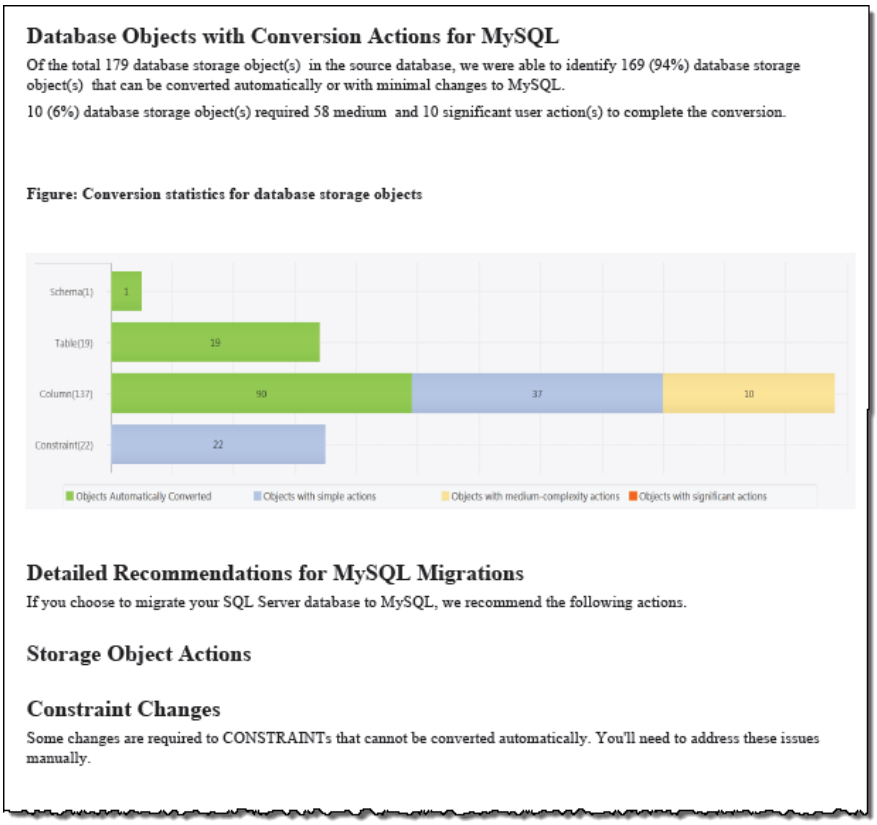

Figura 8: Report migrazione

Importante: Questo report è uno strumento prezioso per valutare le risorse necessarie per il progetto di migrazione del database. Esaminare il report di valutazione in dettaglio per determinare quali modifiche di codice sono necessarie nello schema del database e l'impatto di tali modifiche possono avere sulla funzionalità e sul progetto dell'applicazione.

11. Il passaggio successivo è la conversione dello schema. Lo schema convertito non viene immediatamente applicato al database di destinazione. Al contrario, viene archiviato localmente finché non si applica esplicitamente lo schema convertito al database di destinazione. Per convertire lo schema dal database di origine, scegliere un oggetto dello schema da convertire dal riquadro sinistro del proprio progetto. Fare clic con il pulsante destro del mouse sull'oggetto e scegliere **Converti schema**, come illustrato di seguito.

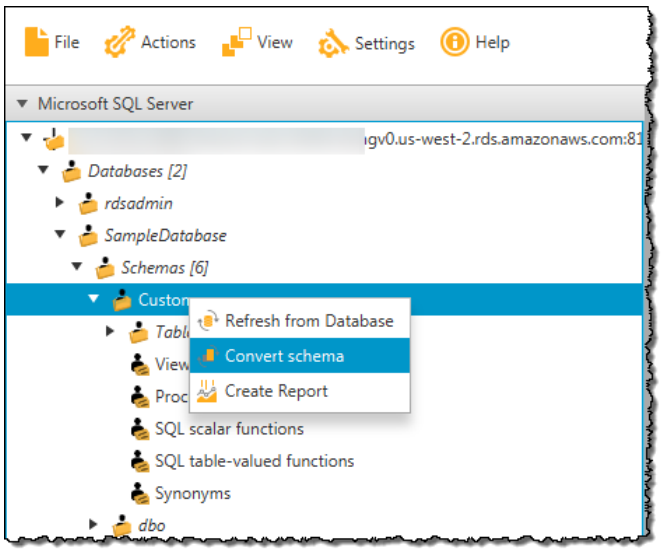

Figura 9: Converti schema

Questa azione aggiunge lo schema convertito al riquadro di destra della finestra del progetto e mostra gli oggetti che sono stati convertiti automaticamente da AWS Schema Conversion Tool.

12. È possibile rispondere alle voci di azione nel report di valutazione in diversi modi:

- Aggiungi schema equivalente manualmente. È possibile scrivere la porzione di schema che può essere convertita automaticamente all'istanza di database di destinazione scegliendo **Apply to database** nel riquadro a destra del progetto. Lo schema che viene aggiunto all'istanza di database di destinazione non includerà gli elementi che non hanno potuto essere convertiti automaticamente. Tali elementi sono elencati nel report di valutazione della migrazione del database.

  Dopo aver applicato lo schema alla propria istanza di database di destinazione, è possibile creare manualmente lo schema nell'istanza di database di destinazione per gli elementi che non hanno potuto essere convertiti automaticamente. In alcuni casi, potrebbe non essere possibile creare uno schema equivalente nella propria istanza di database di destinazione. Potrebbe essere necessario riprogettare una parte dell'applicazione e del database per utilizzare la funzionalità disponibile dal motore di database per la propria istanza di database di destinazione. In altri casi, è sufficiente ignorare lo schema che non può essere convertito automaticamente.

Attenzione: Se si crea manualmente lo schema nella propria istanza di database di destinazione, non scegliere **Apply to database** finché non è stata salvata una copia di qualsiasi intervento manuale effettuato. L'applicazione dello schema dal proprio progetto all'istanza di database di destinazione sovrascrive lo schema con lo stesso nome dell'istanza database di destinazione e si perdono gli eventuali aggiornamenti effettuati manualmente.

- Modificare lo schema del database di origine e aggiornare lo schema del progetto. Per alcuni elementi, è consigliabile modificare lo schema del database di origine in modo che sia compatibile con l'architettura dell'applicazione e che possa anche essere convertito automaticamente al motore di database dell'istanza di database di destinazione. Dopo l'aggiornamento dello schema del database di origine e la verifica che gli aggiornamenti sono compatibili con la propria applicazione, scegliere **Refresh from Database** nel riquadro sinistro del progetto per aggiornare lo schema dal database di origine. È quindi possibile convertire lo schema aggiornato e generare i report di valutazione di migrazione del database. L'azione per lo schema aggiornato non è più visualizzata.

13. Se si desidera applicare la conversione dello schema convertito all'istanza di Aurora di destinazione, scegliere l'elemento dello schema dal riquadro a destra del proprio progetto. Fare clic con il pulsante destro del mouse sull'elemento dello schema e scegliere **Apply to database**, come illustrato nella seguente figura.

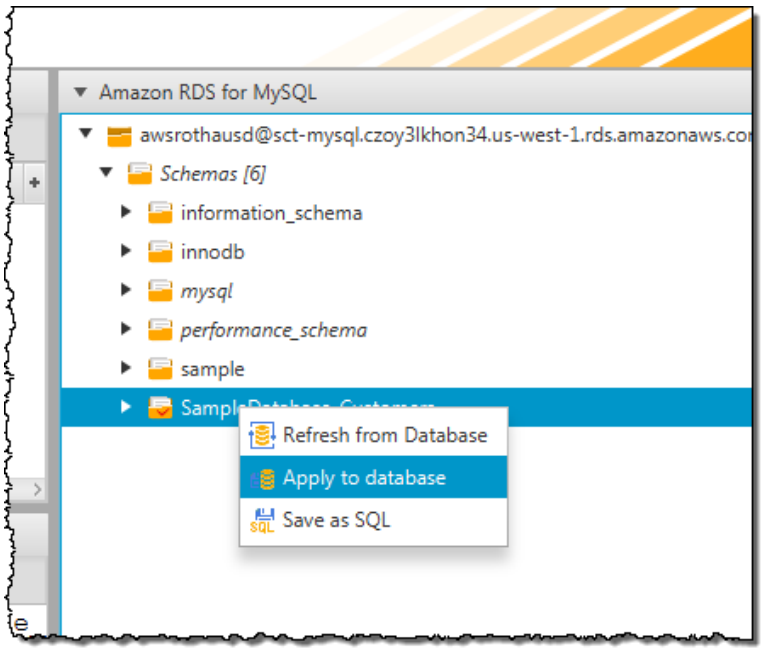

Figura 10: Applicazione schema a database

> **Nota:** La prima volta che si applica lo schema convertito alla propria istanza di database di destinazione, AWS Schema Conversion Tool aggiunge un ulteriore schema (`AWS_ORACLE_EXT` o `AWS_SQLSERVER_EXT`) all'istanza di database di destinazione. Questo schema implementa funzioni di sistema del database di origine che sono necessarie alla scrittura dello schema convertito sulla propria istanza di database di destinazione. Non modificare questo schema altrimenti lo schema convertito che viene scritto nell'istanza di database di destinazione potrebbe presentare anomalie. Quando lo schema è completamente migrato all'istanza di database di destinazione e non è più necessario utilizzare AWS Schema Conversion Tool, è possibile eliminare lo schema `AWS_ORACLE_EXT` o `AWS_SQLSERVER_EXT`.

AWS Schema Conversion è uno strumento di facile utilizzo per il toolkit di migrazione. Per ulteriori best practice correlate a AWS Schema Conversion Tool, consultare l'argomento Best Practice nella *Guida per l'utente di AWS Schema Conversion Tool*.[17]

Migrazione dei dati

Dopo aver copiato lo schema dal database di origine al database di destinazione Aurora, è possibile eseguire la migrazione effettiva dei dati da origine a destinazione. Sebbene la migrazione dei dati possa essere effettuata utilizzando diversi strumenti, consigliamo di spostare i dati utilizzando AWS Database Migration Service (AWS DMS) che fornisce la semplicità e la funzionalità necessarie per l'attività in corso.

Introduzione e approccio generale a AWS DMS

AWS Database Migration Service (AWS DMS) consente ai clienti di eseguire la migrazione dei database di produzione ad AWS con tempi di inattività minimi. È possibile mantenere le applicazioni in esecuzione durante la migrazione del database. Inoltre, AWS Database Migration Service garantisce che le modifiche a dati del database di origine che si verificano durante e dopo la migrazione vengano replicate in modo continuo nella destinazione. È possibile configurare le attività di migrazione in pochi minuti nella Console di gestione AWS. AWS Database Migration Service è in grado di eseguire la migrazione dei dati da e verso diffuse piattaforme di database, ad esempio Oracle, SQL Server, MySQL, PostgreSQL, Amazon Aurora, MariaDB e Amazon Redshift.

Il servizio supporta sia migrazioni omogenee, ad esempio da Oracle a Oracle, sia migrazioni eterogenee tra diverse piattaforme di database, ad esempio da Oracle ad Amazon Aurora o da SQL Server a MySQL. È possibile eseguire migrazioni una tantum oppure mantenere la replica continua tra i database senza che il cliente debba installare o configurare qualsiasi software complesso.

AWS DMS funziona con i database disponibili in loco, in esecuzione su Amazon EC2 o in esecuzione su Amazon RDS. Tuttavia, AWS DMS non funziona in situazioni in cui sia il database di origine sia il database di destinazione sono locali; un endpoint deve essere in AWS.

AWS DMS supporta versioni specifiche di Oracle, SQL Server, Amazon Aurora, MySQL e PostgreSQL. Per le versioni attualmente supportate, consultare la Guida per l'utente di AWS Database Migration Service.[18] Tuttavia, questo whitepaper tratta esclusivamente Amazon Aurora come destinazione di migrazione.

Metodi di migrazione

AWS DMS offre tre metodi di migrazione dei dati:

Migrazione dei dati esistenti. Questo metodo crea le tabelle nel database di destinazione, definisce automaticamente i metadati necessari alla destinazione e popola le tabelle con i dati dal database di origine (noto anche come "carico completo"). I dati dalle tabelle vengono caricati in parallelo per migliorare l'efficienza. Le tabelle sono create solo in caso di migrazioni omogenee, e gli indici secondari non vengono creati automaticamente da AWS DMS. Continuare a leggere per ulteriori dettagli.

Migrazione dei dati esistenti e replica delle modifiche in corso. Questo metodo non ha un pieno carico, come descritto sopra, e in aggiunta acquisisce qualsiasi modifica in corso apportata al database di origine durante il carico completo e la memorizza sulla istanza di replica. Una volta terminato il carico completo, le modifiche salvate vengono applicate al database di destinazione finché non risulta aggiornato con il database di origine. Inoltre, qualsiasi modifica in corso effettuata al database di origine continuerà ad essere replicata sul database di destinazione mantenere la sincronizzazione. Questo metodo di migrazione è molto utile quando si desidera eseguire la migrazione di un database con un'esigua disponibilità di tempi di inattività.

Solo replica delle modifiche dei dati. Questo metodo legge solamente le modifiche dal file di log di ripristino del database di origine e le applica al database di destinazione in modo continuo. Se il database di destinazione non è disponibile, tali modifiche sono conservate sull'istanza di replica finché la destinazione non sarà disponibile.

Quando AWS DMS sta eseguendo una migrazione a carico completo, l'elaborazione inserisce un carico sulle tabelle nel database di origine, che potrebbe influenzare le prestazioni delle applicazioni che utilizzano questo database nello stesso momento. Se questo è un problema, e non è possibile arrestare le applicazioni durante la migrazione, è possibile considerare i seguenti approcci:

- Eseguire la migrazione in un momento in cui il carico dell'applicazione sul database è al punto più basso.
- Creare una replica di lettura del database di origine e quindi eseguire la migrazione AWS DMS dalla replica di lettura.

Procedura di migrazione

Il quadro generale per l'utilizzo di AWS DMS è il seguente:

1. Creare un database di destinazione.
2. Copiare lo schema.
3. Creare un'istanza di replica AWS DMS.
4. Definire gli endpoint del database di origine e di destinazione.
5. Creare ed eseguire un'attività di migrazione.

Creazione del database di destinazione

Creare il proprio cluster di database Amazon Aurora di destinazione utilizzando la procedura descritta in Creazione di un cluster di database Amazon Aurora nella *Guida per l'utente di Amazon RDS*.[19] È consigliabile creare il database di destinazione nella regione e con un tipo di istanza che soddisfi i requisiti aziendali. Inoltre, per migliorare le prestazioni di migrazione, verificare che il database di destinazione non disponga di implementazione Multi-AZ abilitata; è possibile abilitarla al termine del ciclo di carico.

Copia schema

Inoltre, è consigliabile creare lo schema in questo database di destinazione. AWS DMS supporta la migrazione di schemi di base, tra cui la creazione di tabelle e chiavi primarie. Tuttavia, AWS DMS non crea automaticamente indici secondari, chiavi esterne, procedure salvate, account utente e così via nel database di destinazione. Per ulteriori informazioni, vedere la sezione Migrazione dello schema del database.

Creare un'istanza di replica AWS DMS

Per utilizzare il servizio AWS DMS, è necessario creare un'istanza di replica DMS AWS, che viene eseguita nel VPC. Questa istanza legge i dati dal database di origine, esegue le mappature della tabella specificata e scrive i dati nel database di destinazione. In generale, utilizzando un'istanza di replica di dimensioni maggiori si accelera la migrazione del database (anche se la velocità dell'operazione può essere limitata anche da altri fattori, come la capacità dei database di origine e di destinazione, la latenza di connessione, eccetera). Inoltre, l'istanza di replica può essere arrestata una volta completata la migrazione del database.

Figura 11: AWS Database Migration Service

AWS DMS attualmente supporta le classi di istanze T2 e C4 per le istanze di replica. Le classi di istanza T2 sono istanze standard a basso costo progettate per fornire un livello base di prestazioni della CPU, con la possibilità di incrementare le prestazioni oltre tale livello. Sono idonee per lo sviluppo, la configurazione e il testing del database del processo di migrazione, nonché per la migrazione periodica dei dati, che può trarre vantaggio dalla possibilità di aumentare le prestazioni della CPU. Le classi di istanza C4 sono progettate per fornire il massimo livello di prestazioni del processore e conseguire prestazioni PPS (pacchetti per secondo) notevolmente superiori, jitter di rete più ridotti e ridurre la latenza di rete. È consigliabile utilizzare le classi di istanze C4, nel caso in cui si

esegua la migrazione di database di grandi dimensioni e si desideri ridurre al minimo la migrazione.

Di solito, l'esecuzione di un pieno carico non richiede una notevole quantità di storage dell'istanza sull'istanza di replica AWS DMS. Tuttavia, se si sta eseguendo la replica con il carico completo, le modifiche al database di origine vengono archiviate sull'istanza di replica AWS DMS mentre avviene il carico completo. Se si esegue la migrazione di un database di origine di dimensioni molto grandi, che inoltre riceve numerosi aggiornamenti durante il processo di migrazione, può venire utilizzata una notevole quantità di storage dell'istanza. La famiglia di istanze C4 viene fornita con 100 GB di storage dell'istanza e la famiglia di istanze T2 viene fornita con 50 GB. Normalmente questi volumi di storage devono essere più che sufficienti per la maggior parte degli scenari di migrazione.

Inoltre, in alcuni casi estremi in cui i database di dimensioni molto grandi con percentuali di transazioni molto elevate vengono migrati con la replica abilitata, è possibile che la replica AWS DMS non sia in grado di recuperare nel tempo. Se si verifica questo problema, può essere necessario arrestare le modifiche al database di origine per un certo numero di minuti per consentire alla replica di recuperare prima di ripuntare l'applicazione al database di destinazione Aurora.

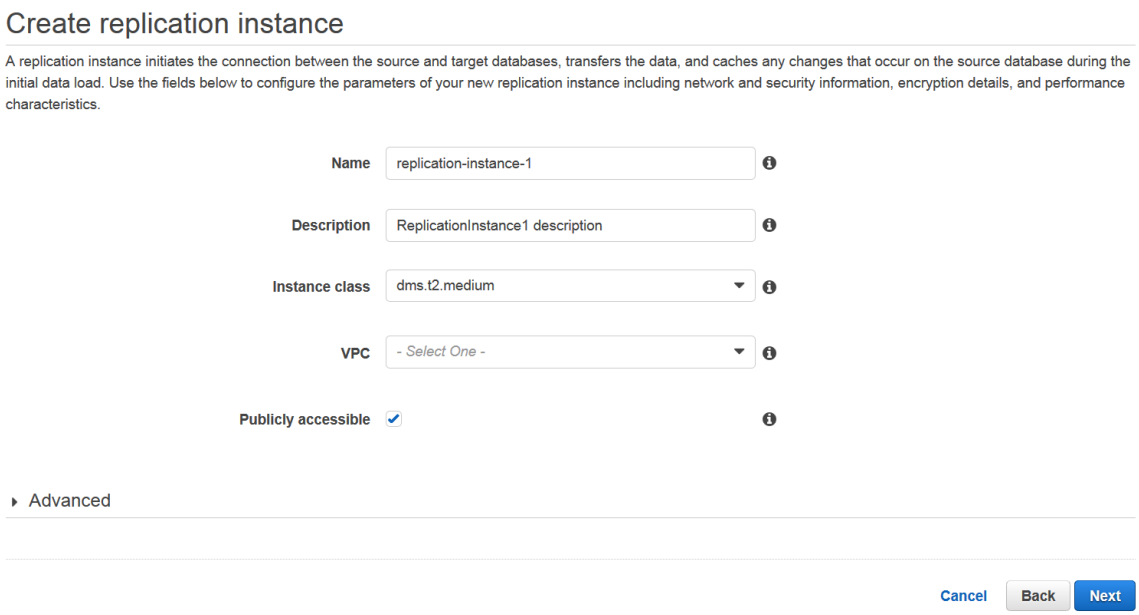

Figura 12: Pagina Creazione istanza di replica nella console AWS DMS

Definizione degli endpoint del database di origine e di destinazione

Un endpoint del database viene utilizzato dall'istanza di replica per connettersi a un database. Per eseguire la migrazione di un database, è necessario creare un endpoint per *entrambi* i database di origine e di destinazione. Gli endpoint dei database specificati possono essere in locale, in esecuzione su Amazon EC2 o in esecuzione su Amazon RDS, ma l'origine e la destinazione non possono essere in locale.

È consigliabile testare la connessione dell'endpoint del database dopo averlo definito. La stessa pagina utilizzata per creare un endpoint di database può essere utilizzata anche per testarlo, come descritto più avanti in questo documento.

Nota: In presenza di vincoli di chiave esterna nel proprio schema di origine, quando si crea l'endpoint di destinazione è necessario immettere quanto segue in **Extra connection attributes** nella sezione **Advanced**:

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

Questo disabilita i controlli di chiave esterna durante il caricamento delle tabelle di destinazione. In questo modo si impedisce l'interruzione del caricamento a causa di controlli di chiave esterna non riusciti su tabelle parzialmente caricate.

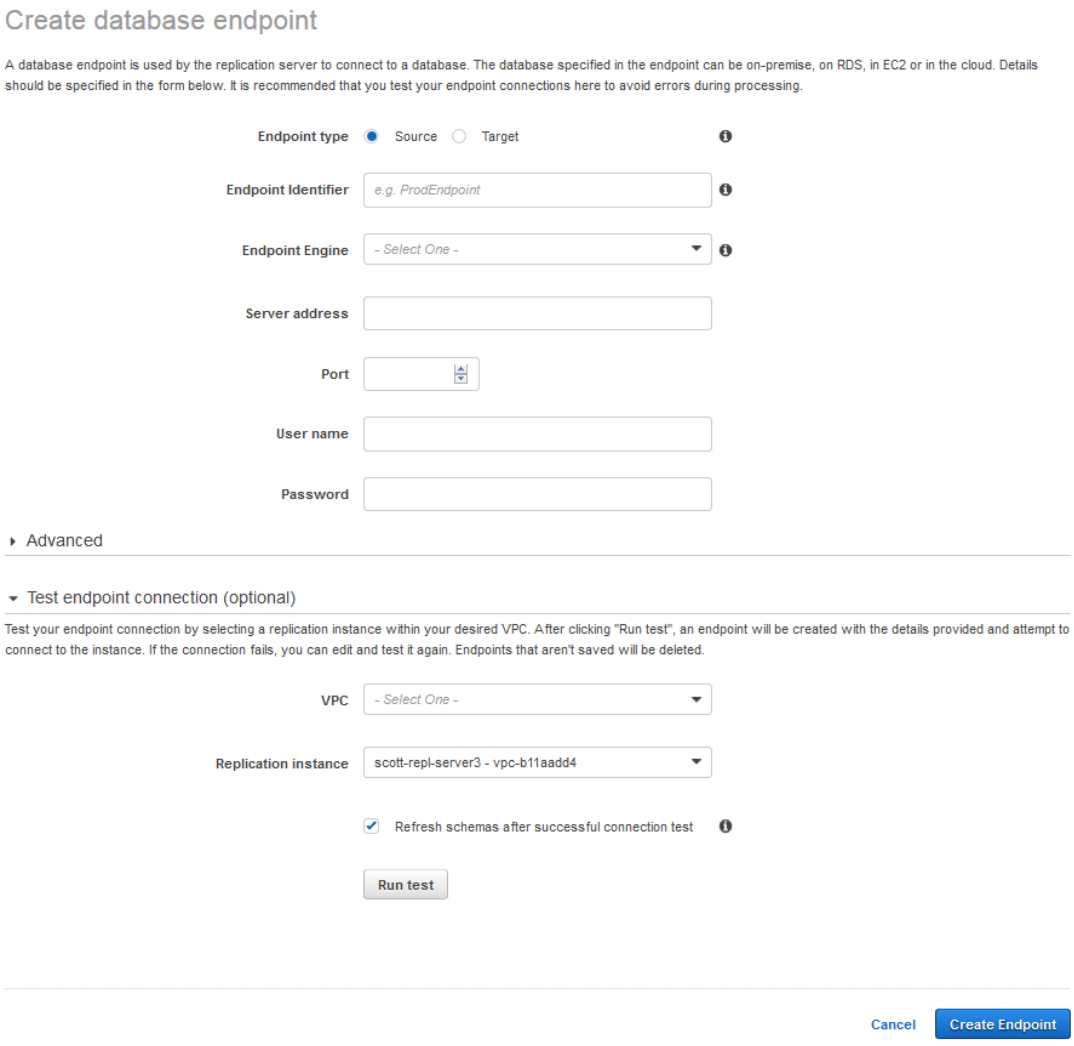

Figura 13: Pagina Creazione endpoint di database nella console di AWS DMS

Creare ed eseguire un'attività di migrazione

Dopo aver creato e testato l'endpoint del database di origine e l'endpoint del database di destinazione, è possibile creare un'attività per eseguire la migrazione dei dati. Quando si crea un'attività, è necessario specificare l'istanza di replica creata, il tipo di metodo di migrazione del database (descritto in precedenza), l'endpoint del database di origine e di destinazione per il cluster di database Amazon Aurora.

Inoltre, nella sezione **Task Settings**, se è già stato creato lo schema completo nel database di destinazione, è necessario modificare la **Target table preparation mode** a **Do nothing** anziché utilizzarne il valore di default di **Drop tables on target**. Quest'ultimo potrebbe comportare la perdita di aspetti della definizione

dello schema, come i vincoli di chiave esterna, durante il rilascio e la ricreazione delle tabelle.

Quando viene creata un'attività, è possibile creare mappature delle tabelle che specifichino lo schema di origine insieme alle tabelle corrispondenti di cui effettuare la migrazione all'endpoint di destinazione. Il metodo di mappatura predefinito effettua la migrazione di tutte le tabelle di origine alle tabelle di destinazione con lo stesso nome, se esistenti. In caso contrario, crea la tabella, o le tabelle, di origine sulla destinazione (a seconda delle impostazioni delle attività). Inoltre, è possibile creare mappature personalizzate (utilizzando un file JSON) se si desidera migrare solo alcune tabelle o se si desidera disporre di maggiore controllo sul campo e sul processo di mappatura tabelle. È inoltre possibile scegliere di migrare solo uno schema o tutti gli schemi dal proprio endpoint di origine.

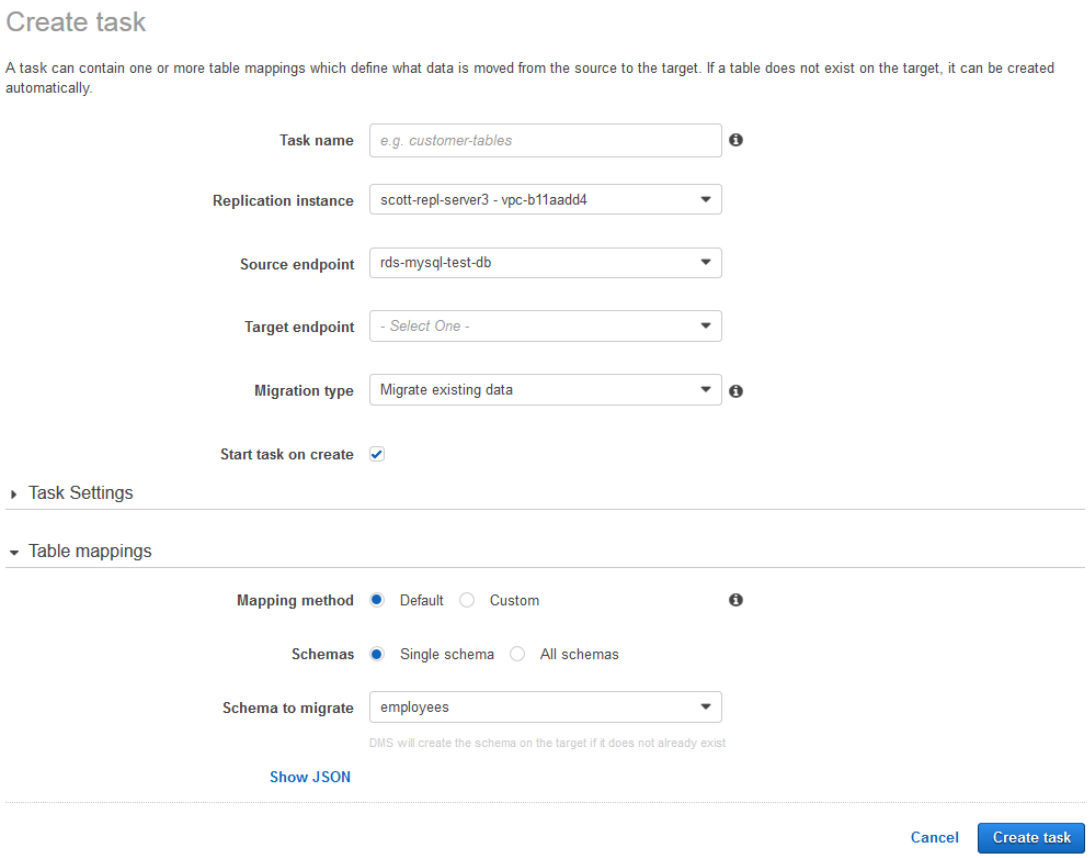

Figura 14: Creazione pagina attività della console di AWS DMS

È possibile utilizzare la console di gestione AWS per monitorare l'avanzamento delle attività AWS Database Migration Service (AWS DMS). È inoltre possibile monitorare le risorse e la connettività di rete utilizzate. La console AWS DMS mostra le statistiche di base per ogni attività, inclusi lo stato del task, la percentuale di completamento, il tempo trascorso e le statistiche di tabella, come mostra l'immagine seguente.

Inoltre, è possibile selezionare un'attività e visualizzare i parametri delle prestazioni per l'attività, tra cui il throughput, i dati al secondo migrati, l'uso del disco e della memoria e la latenza.

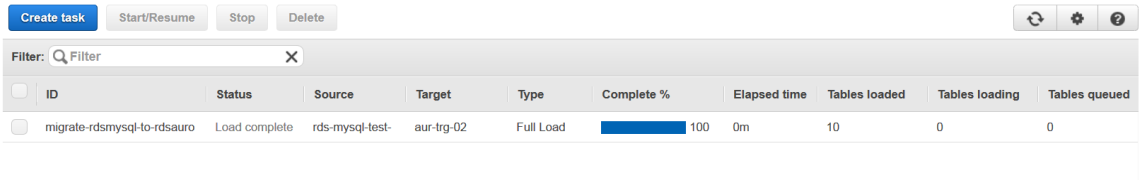

Figura 15: Stato delle attività nella console di AWS DMS

Test e interruzione

Dopo la migrazione dello schema e dei dati dal database di origine ad Amazon Aurora, è ora possibile eseguire il testing dall'inizio alla fine del processo di migrazione. L'approccio di prova dovrebbe essere perfezionato dopo ciascuna prova di migrazione e il piano di migrazione finale dovrebbe includere un piano di testing che garantisca il testing adeguato del database migrato.

Test di migrazione

Categoria di test	Scopo
Prove di accettazione di base	Questi test pre-transizione devono essere eseguiti automaticamente al completamento del processo di migrazione dei dati. Il loro scopo principale è verificare se la migrazione dei dati è riuscita. Di seguito sono elencati alcuni risultati comuni di questi test: • Numero totale di elementi elaborati • Numero totale di elementi importati • Numero totale di elementi saltati • Numero totale di avvertenze • Numero totale di errori Se uno qualsiasi di questi totali ottenuti dai test si scosta dai valori previsti, significa che la migrazione non è andata a buon fine ed è necessario risolvere i problemi prima di passare alla fase successiva del processo o al successivo ciclo di prove.

Categoria di test	Scopo
Test funzionali	Questi test successivi alla transizione verificano la funzionalità dell'applicazione utilizzando Aurora per lo storage di dati. Includono una combinazione di test automatici e manuali. Lo scopo principale del test funzionali è identificare i problemi nell'applicazione causati dalla migrazione dei dati ad Aurora.
Test non funzionali	Questi test successivi alla transizione valutano le caratteristiche non funzionali dell'applicazione, ad esempio le prestazioni con i livelli di carico.
Test di accettazione utente	Questi test successivi alla transizione devono essere eseguiti dall'utente finale dell'applicazione quando la migrazione finale dei dati e la transizione sono state completate. L'obiettivo di questi test è consentire agli utenti finali di decidere se l'applicazione è sufficientemente utilizzabile per soddisfare la relativa funzione primaria nell'organizzazione.

Transizione

Una volta completata la migrazione finale e il testing, è il momento di puntare la propria applicazione al database Amazon Aurora. Questa fase di migrazione è noto come *transizione*. Se la pianificazione e i test sono stati eseguiti correttamente, la transizione non dovrebbe generare problemi imprevisti.

Azioni pre-transizione

- Scegliere una finestra di transizione: Identificare un intervallo di tempo in cui è possibile eseguire la transizione al nuovo database con minima interruzione delle attività aziendali. È possibile selezionare un periodo di bassa attività per il database (in genere di notte e/o nel fine settimana).
- Verificare che le modifiche siano recuperate: Se è stato adottato un approccio di migrazione con tempi di inattività prossimi a zero per replicare le modifiche dal database di origine a quello di destinazione, accertarsi che tutte le modifiche del database siano recuperate e che il database di destinazione non sia eccessivamente in ritardo rispetto a quello di origine.
- Preparare gli script per modificare la configurazione dell'applicazione: Per completare la transizione, è necessario modificare i dettagli di connessione del database nei file di configurazione dell'applicazione. Applicazioni di grandi dimensioni e complesse potrebbero richiedere l'aggiornamento dei dettagli di connessione in più luoghi. Verificare che gli script necessari

siano pronti per aggiornare la configurazione di connessione in modo rapido e affidabile.

- Interrompere l'applicazione: Interrompere i processi dell'applicazione del database di origine e collocare il database di origine in modalità di sola lettura, in modo che su questo non possano essere effettuate altre operazioni di scrittura. Se le modifiche del database di origine non sono completamente recuperate nel database di destinazione, attendere alcuni minuti per consentire il completamento della propagazione di tali modifiche al database di destinazione.
- Eseguire le prove di pre-transizione: Eseguire le prove di pre-transizione automatizzate per assicurarsi che la migrazione dei dati sia stata completata.

Transizione

- Eseguire la transizione: Se i controlli pre-transizione sono stati completati correttamente, è ora possibile puntare l'applicazione ad Amazon Aurora. Eseguire gli script creati nella fase di pre-transizione per modificare la configurazione dell'applicazione in modo che punti al nuovo database Aurora.
- Avviare l'applicazione: A questo punto, è possibile avviare l'applicazione. Se è possibile impedire agli utenti di accedere all'applicazione mentre l'applicazione è in esecuzione, esercitare tale opzione finché non sono stati eseguiti i controlli successivi alla transizione.

Controlli successivi alla transizione

- Eseguire i test successivi alla transizione: Eseguire i test di esempio predefiniti automatici o manuali per verificare il corretto funzionamento dell'applicazione con il nuovo database secondo quanto previsto. È una buona strategia iniziare con i test della funzionalità di sola lettura del database prima di eseguire i test in scrittura sul database.
- Abilitare l'accesso degli utenti e monitorare attentamente: Se i test di esempio sono stati eseguiti correttamente, è possibile consentire l'accesso degli utenti all'applicazione per completare il processo di migrazione. In questo momento, entrambi l'applicazione e il database devono essere monitorati attentamente.

Conclusioni

Amazon Aurora è un database ad alte prestazioni, disponibilità elevata e di classe enterprise creato per il cloud. L'utilizzo di Amazon Aurora è in grado di generare prestazioni migliori e una maggiore disponibilità rispetto ad altri database open-source e di ridurre i costi rispetto alla maggior parte dei database di livello commerciale. Questo documento propone strategie per identificare il metodo migliore per effettuare la migrazione dei database ad Amazon Aurora e descrive in dettaglio le procedure per la pianificazione e l'esecuzione di tali migrazioni. In particolare, AWS Database Migration Service (AWS DMS) e lo Schema Conversion Tool AWS sono strumenti consigliati per gli scenari di migrazione eterogenea. Questi potenti strumenti sono in grado di ridurre i costi e la complessità delle migrazioni di database.

Collaboratori

Le persone e le organizzazioni indicate di seguito hanno collaborato alla stesura di questo documento:

- Puneet Agarwal, Solutions Architect, Amazon Web Services
- Scott Williams, Solutions Architect, Amazon Web Services

Approfondimenti

Per ulteriori informazioni, consultare le seguenti fonti:

- Dettagli prodotto Amazon Aurora
- Domande frequenti su Amazon Aurora
- Amazon Database Migration Service
- Domande frequenti su Amazon Database Migration Service

Note

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

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

[3] https://d0.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