Scomposizione del database su AWS

AWS Guida prescrittiva

AWS Guida prescrittiva: Scomposizione del database su AWS

Table of Contents

Scomposizione del database su AWS

Philippe Wanner e Saurabh Sharma, Amazon Web Services

Ottobre 2025 (storia del documento)

La modernizzazione dei database, in particolare la scomposizione dei database monolitici, è un flusso di lavoro fondamentale per le organizzazioni che desiderano migliorare l'agilità, la scalabilità e le prestazioni dei propri sistemi di gestione dei dati. Man mano che le aziende crescono e le loro esigenze di dati diventano più complesse, i database monolitici tradizionali spesso faticano a tenere il passo. Ciò comporta rallentamenti delle prestazioni, problemi di manutenzione e difficoltà di adattamento ai mutevoli requisiti aziendali.

Di seguito sono elencate le problematiche più comuni relative ai database monolitici:

- Disallineamento dei domini aziendali: i database monolitici spesso non riescono ad allineare la tecnologia con domini aziendali distinti, il che può limitare la crescita organizzativa.
- Limiti di scalabilità: i sistemi spesso raggiungono i limiti di scalabilità, il che crea barriere all'espansione aziendale.
- Rigidità architettonica: le strutture strettamente collegate rendono difficile l'aggiornamento di componenti specifici senza influire sull'intero sistema.
- Degrado delle prestazioni: l'aumento dei carichi di dati e l'aumento della concorrenza tra gli utenti spesso portano a un deterioramento delle prestazioni del sistema.

Di seguito sono riportati i vantaggi della decomposizione del database:

- Maggiore agilità aziendale: la scomposizione consente un rapido adattamento alle mutevoli esigenze aziendali e supporta la scalabilità indipendente.
- Prestazioni ottimizzate: la decomposizione consente di creare soluzioni di database specializzate, personalizzate in base a casi d'uso specifici e scalabili indipendentemente da ciascun database.
- Migliore gestione dei costi: la scomposizione consente un utilizzo più efficiente delle risorse e riduce i costi operativi.
- Opzioni di licenza flessibili: la decomposizione crea opportunità di transizione da costose licenze proprietarie ad alternative open source.
- Abilitazione all'innovazione: la decomposizione facilita l'adozione di database creati appositamente per carichi di lavoro specifici.

Destinatari principali

Questa guida aiuta gli architetti di database, gli architetti di soluzioni cloud, i team di sviluppo di applicazioni e gli architetti aziendali. È progettata per aiutarti a scomporre database monolitici in archivi dati allineati ai microservizi, implementare architetture di database basate sul dominio, pianificare strategie di migrazione dei database e scalare le operazioni dei database per soddisfare le crescenti esigenze aziendali. Per comprendere i concetti e le raccomandazioni di questa guida, è necessario conoscere i principi dei database relazionali e NoSQL AWS , i servizi di database gestiti e i modelli di architettura dei microservizi. Questa guida ha lo scopo di aiutare le organizzazioni che si trovano nelle fasi iniziali di un progetto di scomposizione del database.

Obiettivi

Questa guida può aiutare l'organizzazione a raggiungere i seguenti obiettivi:

- Raccogli i requisiti per scomporre l'architettura di destinazione.
- Sviluppa una metodologia sistematica per la valutazione del rischio e la comunicazione.
- Crea un piano di decomposizione.
- Definisci metriche di successo, indicatori chiave di performance (KPIs), una strategia di mitigazione e un piano di continuità aziendale.
- Stabilisci una migliore elasticità del carico di lavoro che ti aiuti a seguire la domanda aziendale.
- Scopri come adottare database specializzati per casi d'uso specifici, che favoriscono l'innovazione.
- Rafforza la sicurezza e la governance dei dati della tua organizzazione.
- Riduci i costi attraverso quanto segue:
  - Costi di licenza ridotti
  - Riduzione del legame con il fornitore
  - Accesso migliorato al supporto e alle innovazioni della comunità più ampia
  - Possibilità di scegliere diverse tecnologie di database per diversi componenti
  - Migrazione graduale, che riduce i rischi e ripartisce i costi nel tempo
  - Migliore utilizzo delle risorse

Sfide comuni e responsabilità di gestione per la scomposizione del database

La decomposizione del database è un processo complesso che richiede un'attenta pianificazione, esecuzione e gestione. Quando le organizzazioni cercano di modernizzare la propria infrastruttura di dati, spesso incontrano una miriade di sfide che possono influire sul successo dei loro progetti. Questa sezione descrive gli ostacoli più comuni e introduce un approccio strutturato per superarli.

Sfide comuni

Un progetto di scomposizione del database deve affrontare diverse sfide a livello tecnico, personale e aziendale. Sul fronte tecnico, garantire la coerenza dei dati tra i sistemi distribuiti rappresenta un ostacolo significativo. Può inoltre avere potenziali impatti sulle prestazioni e sulla stabilità durante il periodo di transizione ed è necessario integrarsi perfettamente con i sistemi esistenti. Le sfide legate alle persone includono la curva di apprendimento associata al nuovo sistema, la potenziale resistenza ai cambiamenti da parte dei dipendenti e la disponibilità delle risorse necessarie. Dal punto di vista aziendale, il progetto deve far fronte ai rischi di rispetto delle tempistiche, ai vincoli di budget e alla potenziale interruzione delle attività aziendali durante il processo di migrazione.

Definizione di ruoli e responsabilità

Alla luce di queste sfide complesse che riguardano la dimensione tecnica, personale e aziendale, stabilire ruoli e responsabilità chiari diventa fondamentale per il successo del progetto. Una matrice responsabile, responsabile, consultata e informata (RACI) fornisce la struttura necessaria per affrontare queste sfide. Definisce esplicitamente chi prende le decisioni, chi svolge il lavoro, chi fornisce input e chi deve rimanere informato in ogni fase della scomposizione. Questa chiarezza aiuta a prevenire i ritardi causati da processi decisionali ambigui, incoraggia il coinvolgimento appropriato delle parti interessate e crea responsabilità per i risultati chiave. Senza un tale quadro, i team potrebbero avere difficoltà a far fronte a sovrapposizioni di responsabilità, mancate comunicazioni e percorsi di escalation poco chiari, problemi che potrebbero esacerbare le complessità tecniche esistenti e le sfide di gestione del cambiamento, aumentando al contempo il rischio di scadenze e sforamenti di budget.

Il seguente esempio di matrice RACI è un punto di partenza che può aiutarvi a chiarire i ruoli e le responsabilità potenziali all'interno dell'organizzazione.

Compito o attività	Responsabile del progetto	Architetto	Sviluppatore	Stakeholder
Identifica i risultati e le sfide aziendali	A/R	R	C	–
Definisci l'ambito e identifica i requisiti	A	R	C	C/I
Identifica le metriche di successo del progetto	A	R	C	I
Crea ed esegui il piano di comunicazione	A/R	C	C	I
Definisci l'architettura di destinazione	I	A/R	C	–
Controlla l'accesso al database	I	A/R	R	–
Crea ed esegui il piano di continuità aziendale	A/R	C	I	–
Analizza la coesione e l'accoppiamento	I	A/R	R	I
Sposta la logica aziendale (ad esempio le	I	A	R	–

stored procedure) dal database al livello applicativo				
Disaccoppia le relazioni tra tabelle, note come join	I	A	R	–

Definizione dell'ambito e dei requisiti per la scomposizione del database

Quando si definisce l'ambito e si identificano i requisiti per il progetto di scomposizione del database, è necessario lavorare a ritroso rispetto alle esigenze dell'organizzazione. Ciò richiede un approccio sistematico che bilanci la fattibilità tecnica con il valore aziendale. Questa fase iniziale pone le basi per l'intero processo e aiuta a garantire che gli obiettivi del progetto siano in linea con gli obiettivi e le capacità dell'organizzazione.

Questa sezione contiene i seguenti argomenti:

Stabilire un framework di analisi di base

La definizione dell'ambito inizia con un flusso di lavoro sistematico che guida l'analisi attraverso quattro fasi interconnesse. Questo approccio completo garantisce che gli sforzi di scomposizione del database siano basati su una comprensione approfondita dei sistemi e dei requisiti operativi esistenti. Di seguito sono riportate le fasi del framework di analisi di base:

1. Analisi degli attori: identifica accuratamente tutti i sistemi e le applicazioni che interagiscono con il database. Ciò comporta la mappatura sia dei produttori che eseguono le operazioni di scrittura sia dei consumatori che gestiscono le operazioni di lettura, documentando al contempo i modelli di accesso, le frequenze e i periodi di picco di utilizzo. Questa visione incentrata sul cliente aiuta a comprendere l'impatto di eventuali modifiche e a identificare i percorsi critici che richiedono un'attenzione speciale durante la scomposizione.
2. Analisi delle attività: approfondisci le operazioni specifiche eseguite da ciascun attore. Crei matrici dettagliate di creazione, lettura, aggiornamento ed eliminazione (CRUD) per ogni sistema

e identifichi a quali tabelle accedono e in che modo. Questa analisi consente di scoprire i limiti naturali della decomposizione e mette in evidenza le aree in cui è possibile semplificare l'architettura corrente.

3. Mappatura delle dipendenze: documenta le dipendenze dirette e indirette tra i sistemi, creando visualizzazioni chiare dei flussi e delle relazioni di dati. Questo aiuta a identificare i potenziali punti di rottura e le aree in cui è necessaria un'attenta pianificazione per guadagnare fiducia. L'analisi considera sia le dipendenze tecniche, come le tabelle condivise e le chiavi esterne, sia le dipendenze dei processi aziendali, come le sequenze dei flussi di lavoro e i requisiti di reporting.
4. Requisiti di coerenza: esamina le esigenze di coerenza di ogni operazione con standard elevati. Determina quali operazioni richiedono una coerenza immediata, come le transazioni finanziarie. Altre operazioni possono funzionare con eventuale coerenza, come gli aggiornamenti delle analisi. Questa analisi influenza direttamente la scelta dei modelli di decomposizione e le decisioni architettoniche durante l'intero progetto.

Definizione dei limiti del sistema per la scomposizione del database

I confini del sistema sono perimetri logici che definiscono dove finisce un sistema e inizia un altro, e comprendono la proprietà dei dati, i modelli di accesso e i punti di integrazione. Quando definisci i confini del sistema, fai scelte ponderate ma decisive che bilanciano la pianificazione completa con le esigenze pratiche di implementazione. Considerate il database come un'unità logica che potrebbe estendersi su più database o schemi fisici. Questa definizione dei confini consente di raggiungere i seguenti obiettivi critici:

- Identifica tutti gli attori esterni e i relativi modelli di interazione
- Mappa in modo completo le dipendenze in entrata e in uscita
- Documenta i vincoli tecnici e operativi
- Delinea chiaramente la portata dello sforzo di decomposizione

Considerando i cicli di rilascio

La comprensione dei cicli di rilascio è fondamentale per pianificare la scomposizione del database. Controlla i tempi di rinnovo sia per il sistema di destinazione che per tutti i sistemi dipendenti. Identifica le opportunità per cambiamenti coordinati. Prendete in considerazione qualsiasi smantellamento pianificato dei sistemi connessi perché ciò potrebbe influenzare la vostra strategia di decomposizione. Tenete conto delle finestre di modifica e dei vincoli di implementazione esistenti per

ridurre al minimo le interruzioni delle attività aziendali. Assicurati che il tuo piano di implementazione sia in linea con le pianificazioni di rilascio su tutti i sistemi connessi.

Valutazione dei vincoli tecnici per la scomposizione del database

Prima di procedere con la scomposizione del database, valutate le principali limitazioni tecniche che determineranno il vostro approccio alla modernizzazione. Esamina le funzionalità del tuo attuale stack tecnologico, comprese le versioni del database, i framework, i requisiti prestazionali e gli accordi sui livelli di servizio. Prendi in considerazione i requisiti di sicurezza e conformità, in particolare per i settori regolamentati. Rivedi i volumi di dati attuali, le proiezioni di crescita e gli strumenti di migrazione disponibili per prendere decisioni informate sulla scalabilità. Infine, confermate i vostri diritti di accesso al codice sorgente e alle modifiche del sistema, poiché queste determineranno le strategie di decomposizione praticabili.

Comprensione del contesto organizzativo

La corretta scomposizione del database richiede la comprensione del più ampio panorama organizzativo in cui opera il sistema. Mappa le dipendenze interdipartimentali e stabilisci canali di comunicazione chiari tra i team. Valuta le capacità tecniche del tuo team e identifica eventuali esigenze di formazione o lacune di competenze che devi colmare. Prendi in considerazione le implicazioni relative alla gestione delle modifiche, tra cui la gestione delle transizioni e il mantenimento della continuità aziendale. Valuta le risorse disponibili e gli eventuali vincoli, come i limiti di budget o di personale. Infine, allinea la tua strategia di scomposizione alle aspettative e alle priorità degli stakeholder per promuovere un supporto continuo durante tutto il progetto.

Valutazione del rischio di decomposizione del database

Una valutazione completa del rischio è essenziale per il successo della scomposizione del database. Valuta attentamente i rischi, come l'integrità dei dati durante la migrazione, il potenziale peggioramento delle prestazioni del sistema, i possibili errori di integrazione e le vulnerabilità di sicurezza. Queste sfide tecniche devono essere bilanciate rispetto ai rischi aziendali, tra cui potenziali interruzioni operative, limitazioni delle risorse, ritardi nelle tempistiche e vincoli di budget. Per ogni rischio identificato, sviluppate strategie di mitigazione e piani di emergenza specifici per mantenere lo slancio del progetto proteggendo al contempo le operazioni aziendali.

Crea una matrice di rischio che valuti sia l'impatto che la probabilità di potenziali problemi. Collabora con i team tecnici e gli stakeholder aziendali per identificare i rischi, stabilire soglie di intervento

chiare e sviluppare strategie di mitigazione specifiche. Ad esempio, considera il rischio di perdita di dati come elevato impatto e bassa probabilità e richiede solide strategie di backup. Un lieve peggioramento delle prestazioni potrebbe avere un impatto medio e un'alta probabilità e richiede un monitoraggio proattivo.

Stabilisci cicli regolari di revisione del rischio per rivalutare le priorità e adattare i piani di mitigazione man mano che il progetto si evolve. Questo approccio sistematico assicura che le risorse siano concentrate sui rischi più critici, mantenendo al contempo chiari percorsi di escalation per le problematiche emergenti.

Definizione dei criteri di successo per la scomposizione del database

I criteri di successo per la scomposizione del database devono essere chiaramente definiti e misurabili su più dimensioni. Dal punto di vista aziendale, stabilite obiettivi specifici per la riduzione dei costi, il miglioramento time-to-market della disponibilità del sistema e la soddisfazione del cliente. Il successo tecnico deve essere misurato attraverso miglioramenti quantificabili delle prestazioni del sistema, dell'efficienza di implementazione, della coerenza dei dati e dell'affidabilità complessiva. Per il processo di migrazione, definisci requisiti rigorosi per l'assenza di perdite di dati, limiti accettabili per le interruzioni dell'attività, rispetto del budget e rispetto delle tempistiche.

Documenta accuratamente questi criteri mantenendo metriche di base e target, metodologie di misurazione chiare e programmi di revisione regolari. Assegna proprietari chiari per ogni metrica di successo e mappa le dipendenze tra le diverse metriche. Questo approccio globale alla misurazione del successo allinea i risultati tecnici ai risultati aziendali, mantenendo al contempo la responsabilità durante tutto il percorso di decomposizione.

Controllo dell'accesso al database durante la decomposizione

Molte organizzazioni si trovano ad affrontare uno scenario comune: un database centrale che è cresciuto in modo organico nel corso di molti anni e al quale accedono direttamente più servizi e team. Ciò crea diversi problemi critici:

- Crescita incontrollata: man mano che i team aggiungono continuamente nuove funzionalità e modificano gli schemi, il database diventa sempre più complesso e difficile da gestire.
- Problemi relativi alle prestazioni: anche con miglioramenti hardware, il carico crescente alla fine minaccia di superare le capacità del database. Impossibilità di ottimizzare le query a causa della complessità dello schema o della mancanza di competenze. Impossibile prevedere o spiegare le prestazioni del sistema.
- Paralisi da decomposizione: diventa quasi impossibile dividere o rifattorizzare il database mentre viene modificato attivamente da più team.

> **Note**
>
> I sistemi di database monolitici spesso riutilizzano le stesse credenziali per applicazioni o servizi o per l'amministrazione. Ciò comporta una scarsa tracciabilità del database. L'impostazione di ruoli dedicati e l'adozione del principio del privilegio minimo possono aiutarti ad aumentare la sicurezza e la disponibilità.

Quando si ha a che fare con un database monolitico diventato ingombrante, uno dei modelli più efficaci per controllare l'accesso è chiamato servizio di wrapper del database. Rappresenta un primo passo strategico nella gestione di sistemi di database complessi. Stabilisce un accesso controllato al database e consente una modernizzazione graduale, riducendo al contempo i rischi. Questo approccio crea una base per miglioramenti incrementali fornendo una chiara visibilità sui modelli di utilizzo dei dati e sulle dipendenze. È un'architettura di transizione che funge da passo verso la decomposizione completa del database. Il servizio wrapper offre la stabilità e il controllo necessari per affrontare con successo questo percorso.

Questa sezione contiene i seguenti argomenti:

- Controllo dell'accesso con il modello del servizio wrapper del database

- Controllo dell'accesso con il pattern CQRS

Controllo dell'accesso con il modello del servizio wrapper del database

Un servizio wrapper è un livello di servizio che funge da facciata per il database. Questo approccio è particolarmente utile quando è necessario mantenere le funzionalità esistenti preparandosi per future decomposizioni. Questo schema segue un principio semplice: quando qualcosa è troppo disordinato, inizia a contenere il disordine. Il servizio wrapper diventa l'unico modo autorizzato per accedere al database, fornendo un'interfaccia controllata e nascondendo la complessità sottostante.

Utilizzate questo modello quando la decomposizione immediata del database non è possibile a causa di schemi complessi o quando più servizi richiedono un accesso continuo ai dati. È particolarmente utile durante i periodi di transizione perché offre il tempo necessario per un accurato refactoring mantenendo al contempo la stabilità del sistema. Il modello funziona bene quando si consolida la proprietà dei dati a team specifici o quando nuove applicazioni richiedono viste aggregate su più tabelle.

Ad esempio, applica questo schema quando:

- La complessità dello schema impedisce la separazione immediata
- Più team hanno bisogno di un accesso continuo ai dati
- È preferibile una modernizzazione graduale
- La ristrutturazione del team richiede una chiara proprietà dei dati
- Le nuove applicazioni richiedono viste di dati consolidate

Vantaggi e limiti del modello di servizio wrapper del database

Di seguito sono riportati i vantaggi del modello wrapper del database:

- Crescita controllata: il servizio wrapper impedisce ulteriori aggiunte incontrollate allo schema del database.
- Confini chiari: il processo di implementazione consente di stabilire chiari limiti di proprietà e responsabilità.
- Libertà di refactoring: un servizio wrapper consente di apportare modifiche interne senza influire sui consumatori.

- Migliore osservabilità: un servizio wrapper è un punto unico per il monitoraggio e la registrazione.
- Test semplificati: un servizio wrapper semplifica l'utilizzo dei servizi per creare versioni simulate semplificate per i test.

Di seguito sono riportate le limitazioni del modello wrapper del database.

- Accoppiamento tecnologico: un servizio wrapper funziona meglio quando utilizza lo stesso stack tecnologico dei servizi che lo utilizzano.
- Sovraccarico iniziale: il servizio wrapper richiede un'infrastruttura aggiuntiva che potrebbe influire sulle prestazioni.
- Impegno di migrazione: per implementare il servizio wrapper, è necessario coordinarsi tra i team in modo da abbandonare l'accesso diretto.
- Prestazioni: se il servizio di wrapping registra un traffico elevato, un utilizzo intenso o un accesso frequente, i servizi che utilizzano potrebbero avere prestazioni scadenti. Oltre al database, il servizio wrapper deve gestire l'impaginazione, i cursori e le connessioni al database. A seconda del caso d'uso, potrebbe non scalare bene e potrebbe non essere adatto ai carichi di lavoro di estrazione, trasformazione e caricamento (ETL).

Implementazione del modello di servizio wrapper del database

Esistono due fasi per implementare il modello di servizio wrapper del database. Innanzitutto, si crea il servizio wrapper del database. Quindi, dirigete tutti gli accessi attraverso di esso e documentate i modelli di accesso.

Fase 1: creazione del servizio wrapper del database

Crea un livello di servizio leggero che funga da guardiano del tuo database. Inizialmente, dovrebbe rispecchiare tutte le funzionalità esistenti. Questo servizio wrapper diventa il punto di accesso obbligatorio per tutte le operazioni del database, che converte le dipendenze dirette del database in dipendenze a livello di servizio. Implementa la registrazione e il monitoraggio dettagliati a questo livello per tenere traccia dei modelli di utilizzo, delle metriche delle prestazioni e delle frequenze di accesso. Mantieni le procedure archiviate esistenti, ma assicurati che siano accessibili solo tramite questa nuova interfaccia di servizio.

Fase 2: implementazione del controllo degli accessi

Reindirizza sistematicamente tutti gli accessi al database tramite il servizio wrapper, quindi revoca le autorizzazioni dirette al database dai sistemi esterni che accedono direttamente al database. Documenta ogni modello di accesso e dipendenza durante la migrazione dei servizi. Questo accesso controllato consente il refactoring interno dei componenti del database senza interferire con gli utenti esterni. Ad esempio, inizia con operazioni a basso rischio e di sola lettura anziché flussi di lavoro transazionali complessi.

Fase 3: monitoraggio delle prestazioni del database

Utilizza il servizio wrapper come punto di monitoraggio centralizzato per le prestazioni del database. Tieni traccia delle metriche chiave, inclusi i tempi di risposta alle query, i modelli di utilizzo, i tassi di errore e l'utilizzo delle risorse. Imposta avvisi per soglie prestazionali e modelli insoliti. Ad esempio, monitora le query a esecuzione lenta, l'utilizzo del pool di connessioni e la velocità effettiva delle transazioni per identificare in modo proattivo potenziali problemi.

Utilizza questa visualizzazione consolidata per ottimizzare le prestazioni del database tramite l'ottimizzazione delle query, gli aggiustamenti dell'allocazione delle risorse e l'analisi dei modelli di utilizzo. La natura centralizzata del servizio wrapper semplifica l'implementazione dei miglioramenti e la convalida del loro impatto su tutti i consumatori, mantenendo al contempo standard prestazionali coerenti.

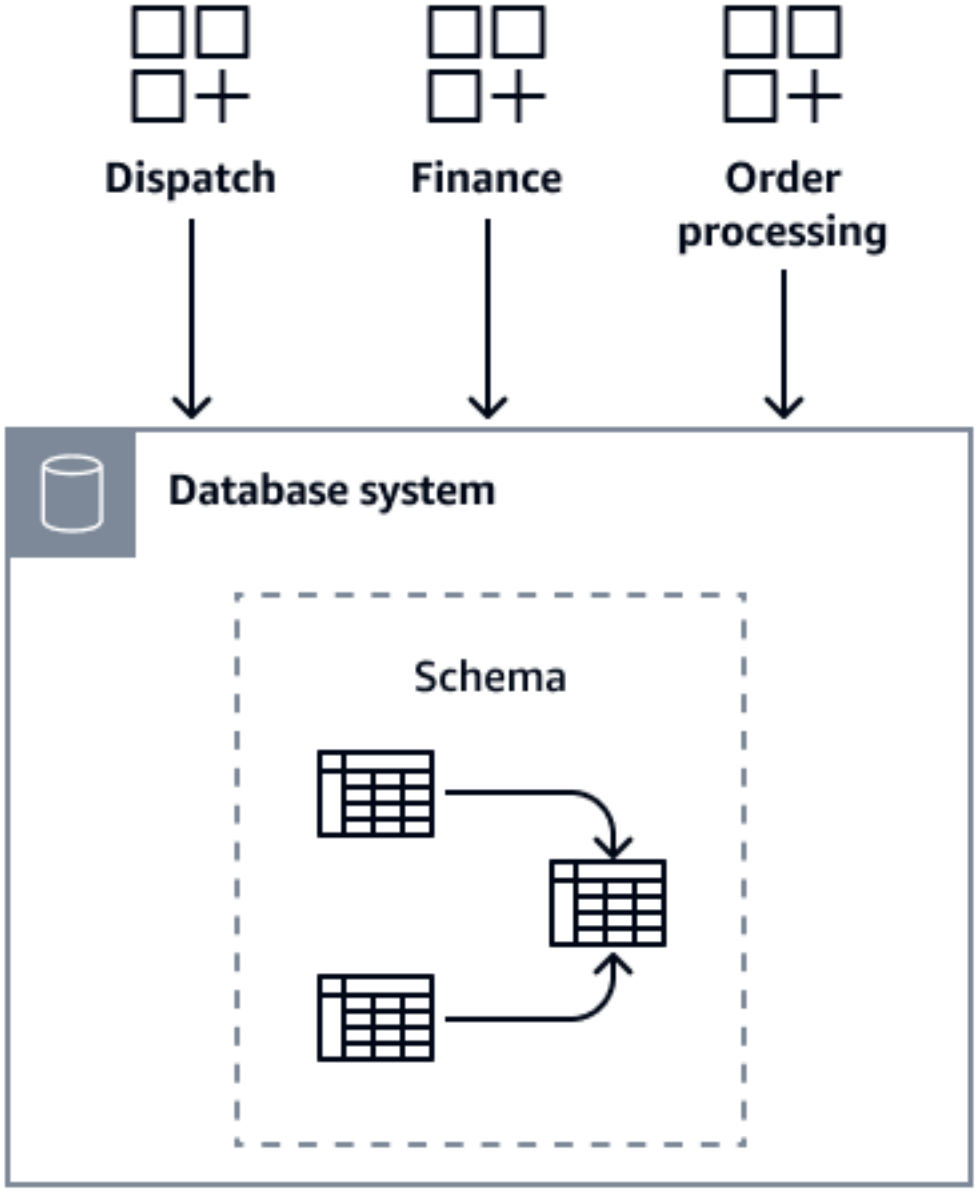
Le migliori pratiche per l'implementazione di un servizio di wrapper di database

Le seguenti best practice possono aiutarti a implementare un servizio di database wrapper:

- Inizia in piccolo: inizia con un wrapper minimo che si limita a trasferire tramite proxy le funzionalità esistenti
- Mantieni la stabilità: mantieni stabile l'interfaccia di servizio apportando miglioramenti interni
- Monitora l'utilizzo: implementa un monitoraggio completo per comprendere i modelli di accesso
- Proprietà chiara: assegna un team dedicato alla manutenzione sia del wrapper che dello schema sottostante
- Incoraggia l'archiviazione locale: motiva i team a archiviare i propri dati nei propri database

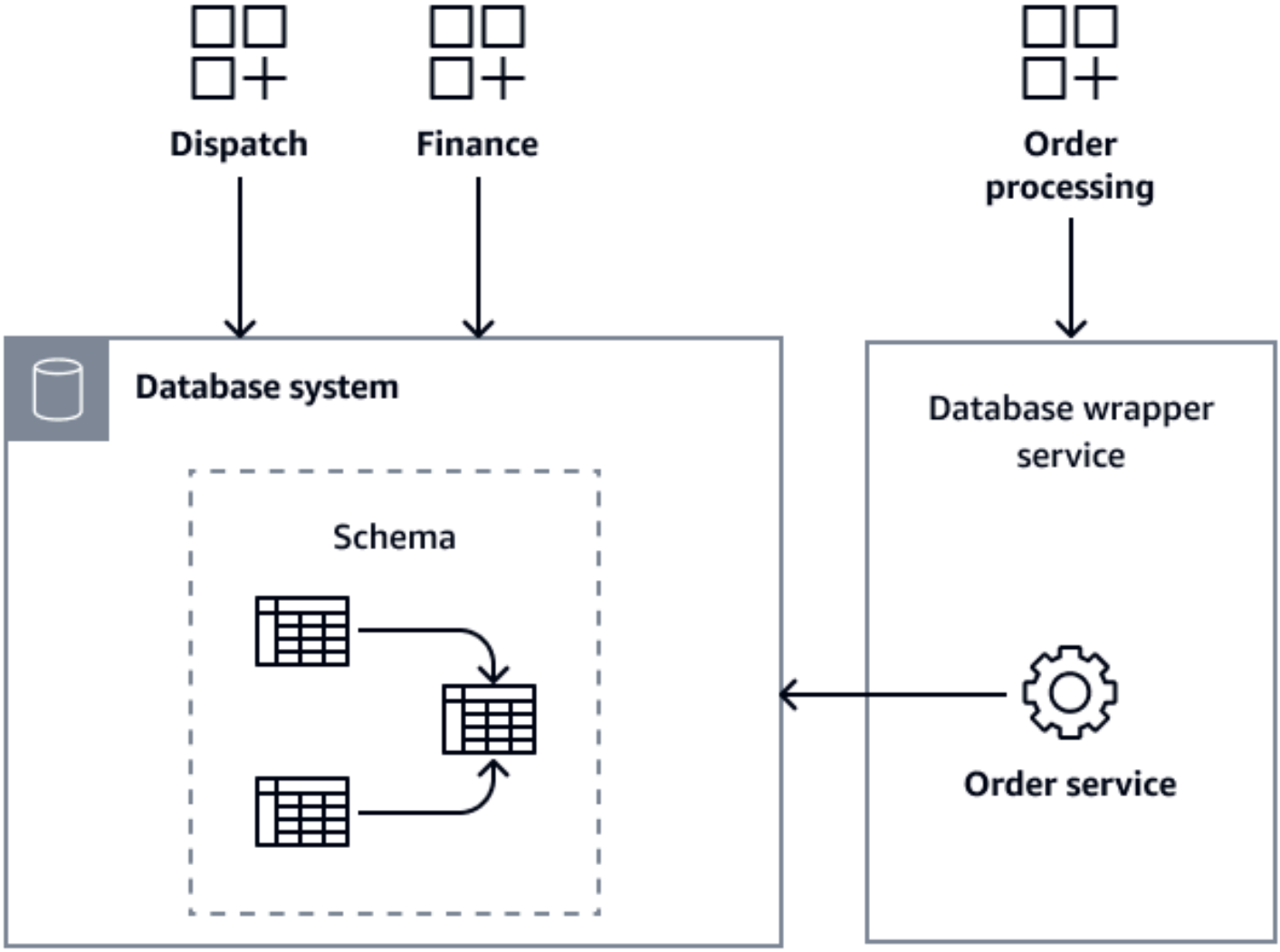
Esempio basato su scenari

Questa sezione descrive un esempio di come un'azienda fittizia, denominata AnyCompany Books, potrebbe utilizzare il modello wrapper del database per controllare l'accesso al proprio sistema di database monolitico. In AnyCompany Books, ci sono tre servizi fondamentali: Dispatch, Finance ed Order Processing. Questi servizi condividono l'accesso a un database centrale. Ogni servizio è gestito da un team diverso. Nel tempo, modificano in modo indipendente lo schema del database per soddisfare le loro esigenze specifiche. Ciò ha portato a una rete intricata di dipendenze e a una struttura di database sempre più complessa.



L'applicativo o l'architetto aziendale dell'azienda riconosce la necessità di scomporre questo database monolitico. Il loro obiettivo è fornire a ciascun servizio il proprio database dedicato per migliorare la manutenibilità e ridurre le dipendenze tra i team. Tuttavia, devono affrontare una sfida importante: è quasi impossibile scomporre il database mentre tutti e tre i team continuano a modificarlo attivamente per i progetti in corso. Le continue modifiche allo schema e la mancanza di coordinamento tra i team rendono estremamente rischioso tentare qualsiasi ristrutturazione significativa.

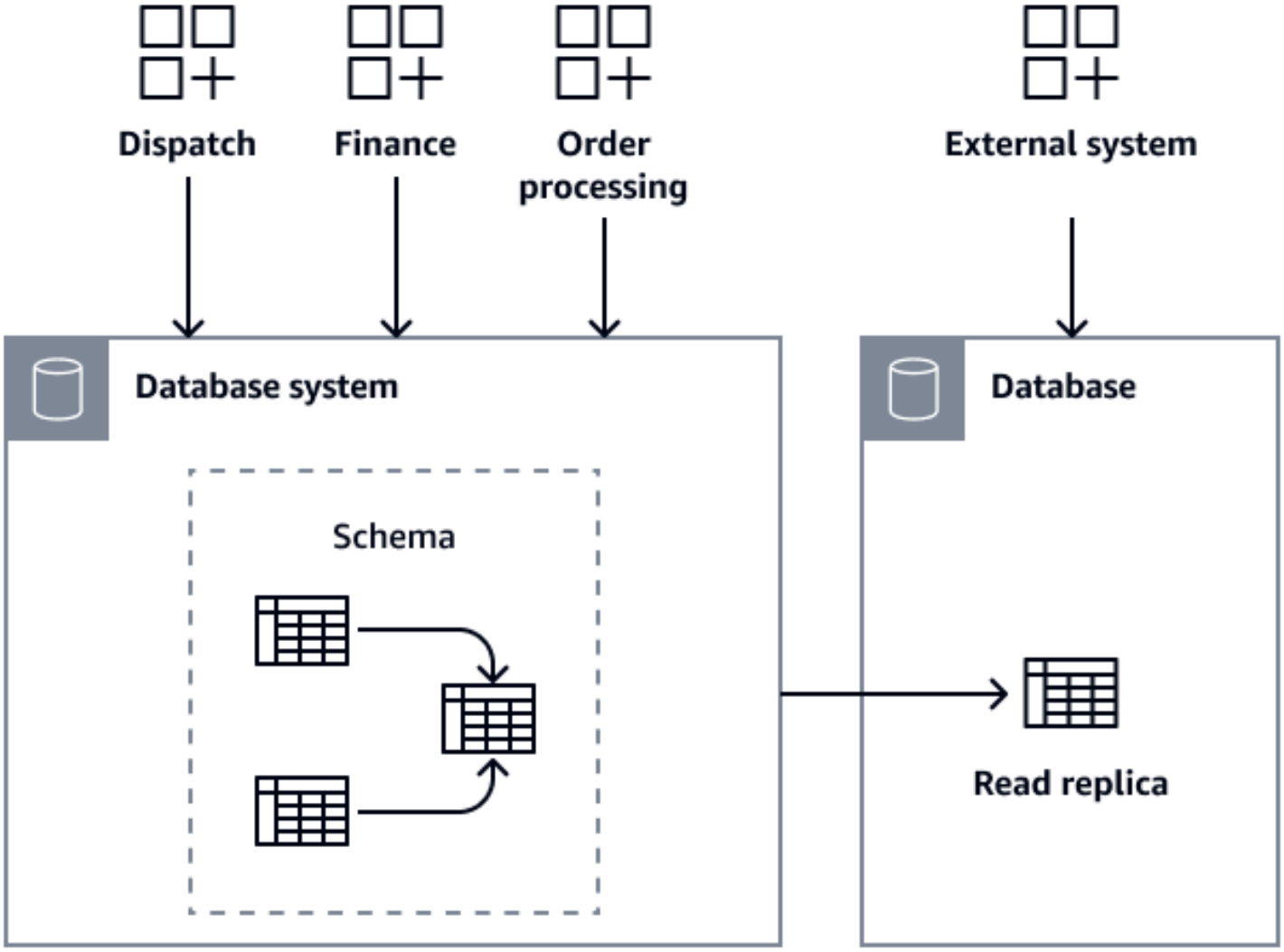
L'architetto utilizza il modello di servizio wrapper del database per iniziare a controllare l'accesso al database monolitico. Innanzitutto, hanno impostato il servizio wrapper del database per un particolare modulo, chiamato servizio Order. Quindi, reindirizzano il servizio di elaborazione degli ordini per accedere al servizio wrapper anziché accedere direttamente al database. L'immagine seguente mostra l'infrastruttura modificata.



Controllo dell'accesso con il pattern CQRS

Un altro modello che puoi utilizzare per isolare i sistemi esterni che si connettono a questo database centrale è la segregazione della responsabilità delle query di comando (CQRS). Se alcuni sistemi esterni si connettono al database centrale principalmente per operazioni di lettura, ad esempio analisi, reportistica o altre operazioni che richiedono molta lettura, è possibile creare archivi dati separati ottimizzati per la lettura.

Questo modello isola efficacemente questi sistemi esterni dagli impatti della decomposizione del database e delle modifiche allo schema. Mantenendo repliche di lettura dedicate o archivi dati creati appositamente per modelli di query specifici, i team possono continuare le proprie operazioni senza risentire delle modifiche nella struttura del database principale. Ad esempio, mentre si scompone il database monolitico, i sistemi di reporting possono continuare a funzionare con le visualizzazioni dei dati esistenti e i carichi di lavoro analitici possono mantenere i modelli di query correnti attraverso archivi analitici dedicati. Questo approccio offre l'isolamento tecnico e consente l'autonomia organizzativa, poiché diversi team possono far evolvere i propri sistemi in modo indipendente senza una stretta connessione con il percorso di trasformazione del database principale.



Per ulteriori informazioni su questo modello e un esempio del suo utilizzo per disaccoppiare le relazioni tra tabelle, vedete Modello CQRS più avanti in questa guida.

Analisi della coesione e dell'accoppiamento per la scomposizione del database

Questa sezione consente di analizzare i modelli di accoppiamento e coesione nel database monolitico per guidarne la scomposizione. Comprendere come i componenti del database interagiscono e dipendono gli uni dagli altri è fondamentale per identificare i punti di interruzione naturali, valutare la complessità e pianificare un approccio di migrazione graduale. Questa analisi rivela le dipendenze nascoste, evidenzia le aree adatte alla separazione immediata e aiuta a dare priorità agli sforzi di decomposizione riducendo al minimo i rischi di trasformazione. Esaminando sia l'accoppiamento che la coesione, è possibile prendere decisioni informate sulla sequenza di separazione dei componenti al fine di mantenere la stabilità del sistema durante l'intero processo di trasformazione.

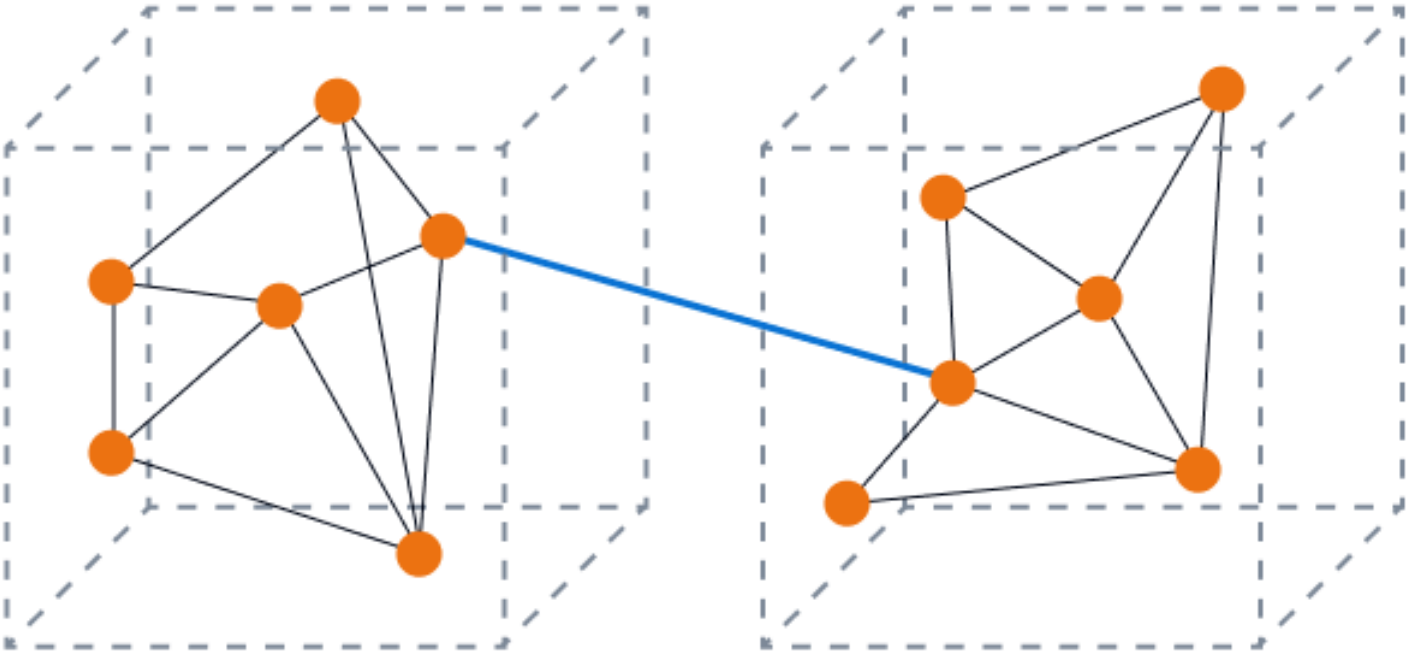
Questa sezione contiene i seguenti argomenti:

- Informazioni sulla coesione e l'accoppiamento
- Schemi di accoppiamento comuni nei database monolitici
- Modelli di coesione comuni nei database monolitici
- Implementazione di basso accoppiamento e alta coesione

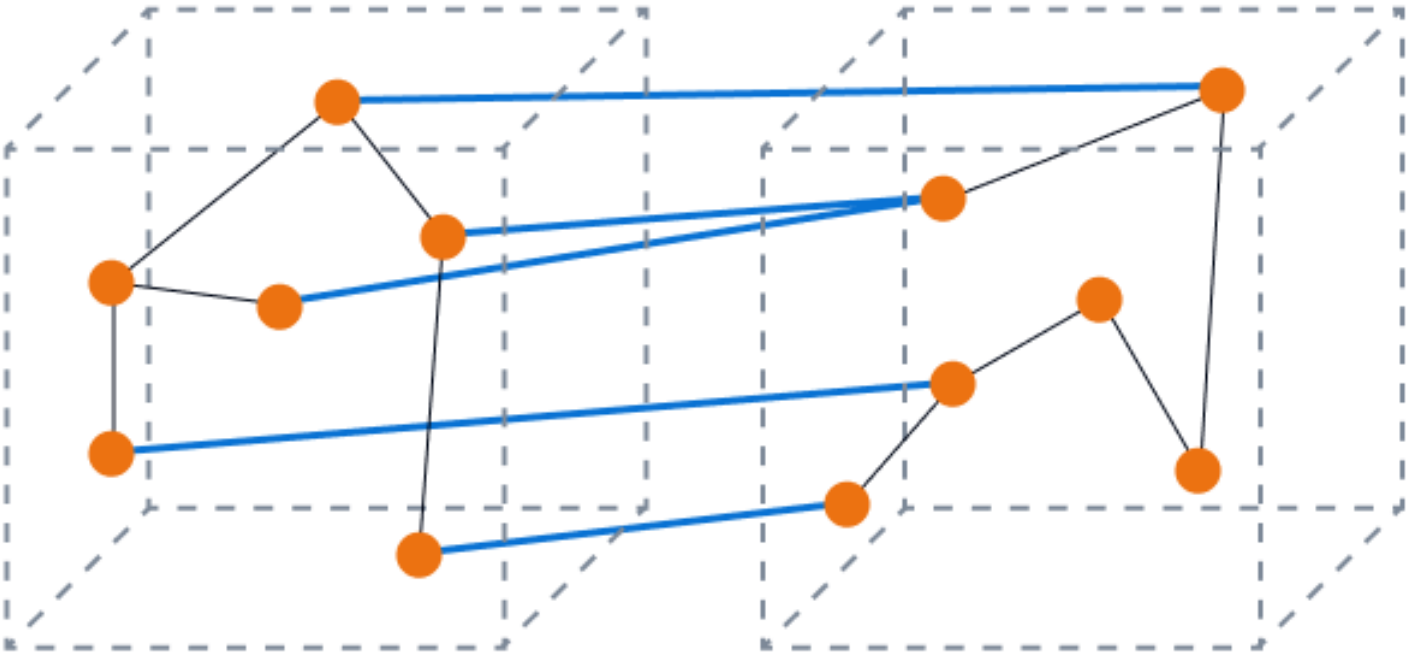
Informazioni sulla coesione e l'accoppiamento

L'accoppiamento misura il grado di interdipendenza tra i componenti del database. In un sistema ben progettato, si desidera ottenere un accoppiamento libero, in cui le modifiche a un componente abbiano un impatto minimo sugli altri. La coesione misura l'efficacia con cui gli elementi all'interno di un componente del database interagiscono per raggiungere un unico scopo ben definito. Un'elevata coesione indica che gli elementi di un componente sono strettamente correlati e focalizzati su una funzione specifica. Quando si scompone un database monolitico, è necessario analizzare sia la coesione all'interno dei singoli componenti sia l'accoppiamento tra di essi. Questa analisi consente di prendere decisioni informate su come suddividere il database mantenendo al contempo l'integrità e le prestazioni del sistema.

L'immagine seguente mostra un accoppiamento libero con un'elevata coesione. I componenti del database interagiscono per eseguire una funzione specifica e voi riducete al minimo l'impatto delle modifiche su un singolo componente. Questo è lo stato ideale.



L'immagine seguente mostra un accoppiamento elevato con una bassa coesione. I componenti del database sono disconnessi ed è molto probabile che le modifiche influiscano su altri componenti.



Schemi di accoppiamento comuni nei database monolitici

Esistono diversi modelli di accoppiamento che si trovano comunemente quando si scompone un database monolitico in database specifici per microservizi. La comprensione di questi modelli è

fondamentale per iniziative di modernizzazione dei database di successo. Questa sezione descrive ogni modello, le relative sfide e le migliori pratiche per ridurre l'accoppiamento.

Modello di accoppiamento dell'implementazione

Definizione: i componenti sono strettamente interconnessi a livello di codice e schema. Ad esempio, la modifica della struttura di una `customer` tabella influisce su `orderinventory`, e sui servizi. `billing`

Impatto della modernizzazione: ogni microservizio richiede uno schema di database e un livello di accesso ai dati dedicati.

Sfide:

- Le modifiche alle tabelle condivise influiscono su più servizi
- Alto rischio di effetti collaterali indesiderati
- Maggiore complessità dei test
- Difficile modificare i singoli componenti

Migliori pratiche per ridurre l'accoppiamento:

- Definire interfacce chiare tra i componenti
- Usa i livelli di astrazione per nascondere i dettagli di implementazione
- Implementa schemi specifici del dominio

Schema di accoppiamento temporale

Definizione: le operazioni devono essere eseguite in una sequenza specifica. Ad esempio, l'elaborazione degli ordini non può procedere fino al completamento degli aggiornamenti dell'inventario.

Impatto della modernizzazione: ogni microservizio richiede un controllo autonomo dei dati.

Sfide:

- Interruzione delle dipendenze sincrone tra i servizi
- Colli di bottiglia in termini di prestazioni
- Difficile da ottimizzare

- Elaborazione parallela limitata

Migliori pratiche per ridurre l'accoppiamento:

- Implementare l'elaborazione asincrona ove possibile
- Utilizza architetture basate sugli eventi
- Progetta per garantire la coerenza finale, se necessario

Schema di accoppiamento dell'implementazione

Definizione: i componenti del sistema devono essere distribuiti come una singola unità. Ad esempio, una modifica minore alla logica di elaborazione dei pagamenti richiede la ridistribuzione dell'intero database.

Impatto della modernizzazione: implementazioni di database indipendenti per servizio

Sfide:

- Implementazioni ad alto rischio
- Frequenza di implementazione limitata
- Procedure di rollback complesse

Migliori pratiche per ridurre l'accoppiamento:

- Suddivisione in componenti distribuibili in modo indipendente
- Implementa strategie di condivisione del database
- Utilizza modelli di distribuzione blu-verdi

Schema di accoppiamento del dominio

Definizione: i domini aziendali condividono la struttura e la logica del database. Ad esempio, i `inventory` domini `customerorder`, e condividono tabelle e stored procedure.

Impatto della modernizzazione: isolamento dei dati specifico del dominio

Sfide:

- Confini di dominio complessi
- Difficile scalare i singoli domini
- Regole aziendali intricate

Le migliori pratiche per ridurre l'accoppiamento:

- Identifica confini di dominio chiari
- Separa i dati per contesto di dominio
- Implementa servizi specifici del dominio

Modelli di coesione comuni nei database monolitici

Esistono diversi modelli di coesione che si riscontrano comunemente quando si valutano i componenti del database per la decomposizione. La comprensione di questi modelli è fondamentale per identificare componenti di database ben strutturati. Questa sezione descrive ogni modello, le sue caratteristiche e le migliori pratiche per rafforzare la coesione.

Modello di coesione funzionale

Definizione: tutti gli elementi supportano e contribuiscono direttamente all'esecuzione di un'unica funzione ben definita. Ad esempio, tutte le procedure e le tabelle memorizzate in un modulo di elaborazione dei pagamenti gestiscono solo le operazioni relative ai pagamenti.

Impatto sulla modernizzazione: modello ideale per la progettazione di database a microservizi

Sfide:

- Identificazione di confini funzionali chiari
- Separazione di componenti a uso misto
- Mantenimento di un'unica responsabilità

Migliori pratiche per rafforzare la coesione:

- Raggruppa le funzioni correlate
- Rimuovi funzionalità non correlate
- Definisci confini chiari dei componenti

Modello di coesione sequenziale

Definizione: l'output di un elemento diventa input per un altro. Ad esempio, i risultati di convalida di un ordine inserito nell'elaborazione degli ordini.

Impatto della modernizzazione: richiede un'attenta analisi del flusso di lavoro e una mappatura del flusso di dati

Sfide:

- Gestione delle dipendenze tra i passaggi
- Gestione degli scenari di errore
- Mantenimento dell'ordine del processo

Migliori pratiche per rafforzare la coesione:

- Documenta flussi di dati chiari
- Implementa una corretta gestione degli errori
- Progetta interfacce chiare tra i passaggi

Modello di coesione comunicativa

Definizione: gli elementi operano sugli stessi dati. Ad esempio, le funzioni di gestione dei profili dei clienti funzionano tutte con i dati dei clienti.

Impatto sulla modernizzazione: aiuta a identificare i limiti dei dati per la separazione dei servizi per ridurre l'accoppiamento tra i moduli

Sfide:

- Definizione della proprietà dei dati
- Gestione dell'accesso condiviso ai dati
- Mantenere la coerenza dei dati

Migliori pratiche per rafforzare la coesione:

- Definire una chiara proprietà dei dati

- Implementa modelli di accesso ai dati adeguati
- Progetta un partizionamento dei dati efficace

Modello procedurale di coesione

Definizione: gli elementi sono raggruppati perché devono essere eseguiti in un ordine specifico, ma potrebbero non essere correlati dal punto di vista funzionale. Ad esempio, nell'elaborazione degli ordini, una procedura memorizzata che gestisce sia la convalida degli ordini che la notifica all'utente viene raggruppata semplicemente perché avvengono in sequenza, anche se hanno scopi diversi e potrebbero essere gestite da servizi separati.

Impatto della modernizzazione: richiede un'attenta separazione delle procedure mantenendo al contempo il flusso dei processi

Sfide:

- Mantenimento del corretto flusso di processo dopo la decomposizione
- Identificazione dei veri limiti funzionali rispetto alle dipendenze procedurali

Migliori pratiche per rafforzare la coesione:

- Procedure separate in base al loro scopo funzionale piuttosto che all'ordine di esecuzione
- Utilizza modelli di orchestrazione per gestire il flusso dei processi
- Implementa sistemi di gestione del flusso di lavoro per sequenze complesse
- Progetta architetture basate sugli eventi per gestire le fasi del processo in modo indipendente

Modello di coesione temporale

Definizione: gli elementi sono correlati da requisiti temporali. Ad esempio, quando viene effettuato un ordine, è necessario eseguire più operazioni contemporaneamente: controllo dell'inventario, elaborazione dei pagamenti, conferma dell'ordine e notifica di spedizione devono avvenire tutte entro una finestra temporale specifica per mantenere uno stato dell'ordine coerente.

Impatto della modernizzazione: potrebbe richiedere una gestione speciale nei sistemi distribuiti

Sfide:

- Coordinamento delle dipendenze temporali tra i servizi distribuiti
- Gestione delle transazioni distribuite
- Conferma del completamento del processo su più componenti

Migliori pratiche per rafforzare la coesione:

- Implementare meccanismi e timeout di pianificazione adeguati
- Utilizza architetture basate sugli eventi con una gestione chiara delle sequenze
- Progettazione per una eventuale coerenza con i modelli di compensazione
- Implementa modelli saga per transazioni distribuite

Modello di coesione logico o casuale

Definizione: Gli elementi sono classificati logicamente per fare le stesse cose, anche se hanno relazioni deboli o prive di significato. Un esempio è l'archiviazione dei dati degli ordini dei clienti, dei conteggi delle scorte di magazzino e dei modelli di e-mail di marketing nello stesso schema di database, poiché tutti si riferiscono alle operazioni di vendita, nonostante abbiano modelli di accesso, gestione del ciclo di vita e requisiti di scalabilità diversi. Un altro esempio è la combinazione dell'elaborazione dei pagamenti degli ordini e della gestione del catalogo dei prodotti all'interno dello stesso componente del database, poiché entrambi fanno parte del sistema di e-commerce, anche se svolgono funzioni aziendali distinte con esigenze operative diverse.

Impatto della modernizzazione: dovrebbe essere rifattorizzato o riorganizzato

Sfide:

- Identificazione di modelli organizzativi migliori
- Rompere le dipendenze non necessarie
- Ristrutturazione di componenti raggruppati arbitrariamente

Migliori pratiche per rafforzare la coesione:

- Riorganizza in base a confini funzionali e domini aziendali reali
- Rimuovi i raggruppamenti arbitrari basati su relazioni superficiali
- Implementa una corretta separazione degli elementi in base alle capacità aziendali

- Allinea i componenti del database ai relativi requisiti operativi specifici

Implementazione di basso accoppiamento e alta coesione

Best practice

Le seguenti best practice possono aiutarti a ottenere un accoppiamento basso:

- Riduci al minimo le dipendenze tra i componenti del database
- Utilizza interfacce ben definite per l'interazione tra i componenti
- Evita strutture di dati globali e statali condivise

Le seguenti best practice possono aiutarti a raggiungere un'elevata coesione:

- Raggruppa i dati e le operazioni correlati
- Assicurati che ogni componente abbia un'unica e chiara responsabilità
- Mantieni confini chiari tra i diversi domini aziendali

Fase 1: mappare le dipendenze dei dati

Mappa le relazioni tra i dati e identifica i confini naturali. È possibile utilizzare strumenti, ad esempio SchemaSpy, per visualizzare il database mostrando le tabelle nel diagramma entità-relazione (ER). Ciò fornisce un'analisi statica del database e indica alcuni dei limiti e delle dipendenze chiari all'interno del database.

È inoltre possibile esportare gli schemi del database in un database grafico o in un Jupiter taccuino. Quindi, è possibile applicare algoritmi di clustering o di componenti interconnessi per identificare confini e dipendenze naturali. Altri AWS Partner strumenti, ad esempio CAST Imaging, possono aiutare a comprendere le dipendenze del database.

Fase 2: analisi dei limiti delle transazioni e dei modelli di accesso

Analizza i modelli di transazione per mantenere le proprietà di atomicità, coerenza, isolamento, durabilità (ACID) e scopri come i dati vengono accessibili e modificati. È possibile utilizzare strumenti di analisi e diagnosi del database, come Oracle Automatic Workload Repository (AWR)o PostgreSQL pg_stat_statements. Questa analisi consente di capire chi sta accedendo al database e quali sono i

limiti delle transazioni. Può anche aiutarti a comprendere la coesione e l'accoppiamento tra le tabelle in fase di esecuzione. È inoltre possibile utilizzare strumenti di monitoraggio e profilazione in grado di collegare codice e profili di esecuzione del database, ad esempio. Dynatrace AppEngine

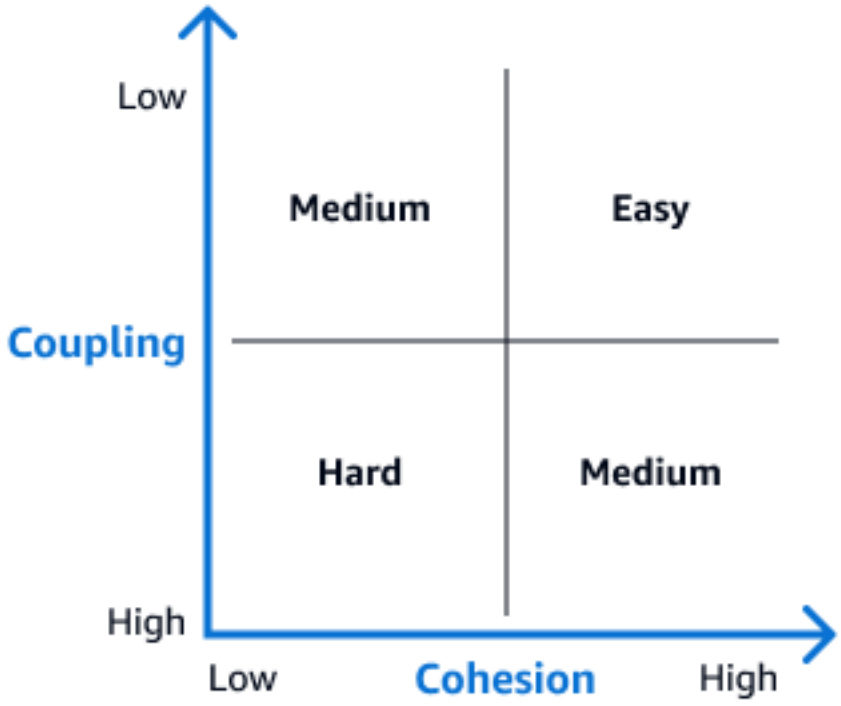
Gli strumenti di intelligenza artificiale, ad esempio vFunction, possono aiutarti a identificare i confini del dominio analizzando i confini funzionali e di dominio dell'applicazione. Sebbene analizzi vFunction principalmente il livello applicativo, le sue informazioni possono guidare la scomposizione sia dell'applicazione che del database, favorendo l'allineamento con i domini aziendali.

Fase 3: Identificazione delle tabelle autonome

Cerca tabelle che dimostrino due caratteristiche chiave:

- Elevata coesione: i contenuti della tabella sono strettamente correlati tra loro
- Basso accoppiamento: hanno una dipendenza minima dalle altre tabelle.

La seguente matrice di accoppiamento e coesione può aiutarvi a identificare la difficoltà di disaccoppiamento di ogni tabella. Le tabelle che appaiono nel quadrante in alto a destra di questa matrice sono le candidate ideali per le operazioni iniziali di disaccoppiamento perché sono le più facili da separare. In un diagramma ER, queste tabelle hanno poche relazioni con chiavi esterne o altre dipendenze. Dopo aver disaccoppiato queste tabelle, passa a tabelle con relazioni più complesse.



> **Note**
>
> La struttura del database spesso rispecchia l'architettura dell'applicazione. Le tabelle più facili da disaccoppiare a livello di database corrispondono in genere a componenti più facili da convertire in microservizi a livello di applicazione.

Migrazione della logica aziendale dal database al livello applicativo

La migrazione della logica di business da procedure, trigger e funzioni archiviati nel database ai servizi a livello di applicazione è un passaggio fondamentale nella scomposizione dei database monolitici. Questa trasformazione migliora l'autonomia del servizio, semplifica la manutenzione e migliora la scalabilità. Questa sezione fornisce indicazioni sull'analisi della logica del database, sulla pianificazione della strategia di migrazione e sull'implementazione della trasformazione mantenendo al contempo la continuità aziendale. Descrive inoltre la definizione di un piano di rollback efficace.

Questa sezione contiene i seguenti argomenti:

Fase 1: analisi della logica aziendale

Quando si modernizzano i database monolitici, è necessario innanzitutto condurre un'analisi completa della logica del database esistente. Questa fase si concentra su tre categorie principali:

- Le stored procedure spesso contengono operazioni aziendali critiche, tra cui logica di manipolazione dei dati, regole aziendali, controlli di convalida e calcoli. In quanto componenti fondamentali della logica aziendale dell'applicazione, richiedono un'attenta scomposizione. Ad esempio, le procedure archiviate di un'organizzazione finanziaria potrebbero gestire il calcolo degli interessi, la riconciliazione dei conti e i controlli di conformità.
- I trigger sono componenti chiave del database che gestiscono gli audit trail, la convalida dei dati, i calcoli e la coerenza tra tabelle. Ad esempio, un'organizzazione di vendita al dettaglio potrebbe utilizzare i trigger per gestire gli aggiornamenti dell'inventario in tutto il sistema di elaborazione degli ordini, il che dimostra la complessità delle operazioni automatizzate del database.
- Le funzioni nei database gestiscono principalmente le trasformazioni dei dati, i calcoli e le operazioni di ricerca. Spesso sono integrate in più procedure e applicazioni. Ad esempio, un'organizzazione sanitaria potrebbe utilizzare funzioni per normalizzare i dati dei pazienti o cercare codici medici.

Ogni categoria rappresenta aspetti diversi della logica aziendale incorporata nel livello del database. È necessario valutarli e pianificarli attentamente per migrarli al livello applicativo.

Durante questa fase di analisi, i clienti in genere devono affrontare tre sfide significative. Innanzitutto, emergono dipendenze complesse attraverso chiamate di procedura annidate, riferimenti tra schemi e dipendenze implicite tra dati. In secondo luogo, la gestione delle transazioni diventa fondamentale, in particolare quando si tratta di transazioni in più fasi e si mantiene la coerenza dei dati tra sistemi distribuiti. In terzo luogo, le considerazioni relative alle prestazioni devono essere valutate attentamente, in particolare per le operazioni di elaborazione in batch, gli aggiornamenti di massa dei dati e i calcoli in tempo reale che attualmente traggono vantaggio dalla vicinanza ai dati.

Per affrontare efficacemente queste sfide, è possibile utilizzare AWS Schema Conversion Tool (AWS SCT) per l'analisi iniziale e quindi utilizzare strumenti dettagliati di mappatura delle dipendenze. Questo approccio consente di comprendere l'intero ambito della logica del database e di creare una strategia di migrazione completa che mantenga la continuità aziendale durante la decomposizione.

Comprendendo a fondo questi componenti e queste sfide, è possibile pianificare meglio il percorso di modernizzazione e prendere decisioni informate sugli elementi a cui dare priorità durante la migrazione verso un'architettura basata su microservizi.

Quando analizzi i componenti del codice del database, crea una documentazione completa per ogni procedura, trigger e funzione archiviati. Inizia descrivendone chiaramente lo scopo e le funzionalità principali, comprese le regole aziendali che implementa. Descrivi in dettaglio tutti i parametri di input e output e annota i relativi tipi di dati e gli intervalli validi. Mappa le dipendenze da altri oggetti del database, sistemi esterni e processi a valle. Definisci chiaramente i limiti delle transazioni e i requisiti di isolamento per mantenere l'integrità dei dati. Documenta qualsiasi aspettativa di prestazioni, compresi i requisiti in termini di tempi di risposta e i modelli di utilizzo delle risorse. Infine, analizza i modelli di utilizzo per comprendere i carichi di picco, la frequenza di esecuzione e i periodi aziendali critici.

Fase 2: Classificazione della logica aziendale

Un'efficace scomposizione del database richiede una categorizzazione sistematica della logica del database in base a dimensioni chiave: complessità, impatto aziendale, dipendenze, modelli di utilizzo e difficoltà di migrazione. Questa classificazione consente di identificare i componenti ad alto rischio, determinare i requisiti di test e stabilire le priorità di migrazione. Ad esempio, procedure archiviate complesse con un elevato impatto aziendale e un utilizzo frequente richiedono un'attenta pianificazione e test approfonditi. Tuttavia, funzioni semplici e utilizzate raramente con dipendenze minime potrebbero essere adatte per le prime fasi di migrazione.

Questo approccio strutturato crea una tabella di marcia di migrazione equilibrata che riduce al minimo le interruzioni aziendali mantenendo al contempo la stabilità del sistema. Comprendendo queste interrelazioni, è possibile migliorare la sequenza delle attività di scomposizione e allocare le risorse in modo appropriato.

Fase 3: migrazione della logica aziendale

Dopo aver analizzato e classificato la logica aziendale, è il momento di migrarla. Esistono due approcci per la migrazione della logica aziendale da un database monolitico: spostare la logica del database al livello dell'applicazione o spostare la logica aziendale su un altro database che fa parte del microservizio.

Se si esegue la migrazione della logica aziendale all'applicazione, le tabelle del database memorizzano solo i dati e il database non contiene alcuna logica aziendale. Questo è l'approccio consigliato. Puoi utilizzare Ispirer o strumenti di intelligenza artificiale generativa, come Amazon Q Developer o Kiro, per convertire la logica di business del database per il livello applicativo, come la conversione in Java. Per ulteriori informazioni, consulta Migrare la logica di business dal database all'applicazione per un'innovazione e una flessibilità più rapide (AWS post sul blog).

Se si esegue la migrazione della logica aziendale a un altro database, è possibile utilizzare AWS Schema Conversion Tool (AWS SCT) per convertire gli schemi di database e gli oggetti di codice esistenti nel database di destinazione. Supporta servizi di AWS database appositamente progettati, come Amazon DynamoDB, Amazon Aurora e Amazon Redshift. Fornendo un rapporto di valutazione completo e funzionalità di conversione automatizzate, AWS SCT aiuta a semplificare il processo di transizione, consentendoti di concentrarti sull'ottimizzazione della nuova struttura del database per migliorare prestazioni e scalabilità. Man mano che avanzi nel progetto di modernizzazione, è in AWS SCT grado di gestire conversioni incrementali per supportare un approccio graduale, che consente di convalidare e ottimizzare ogni fase della trasformazione del database.

Strategia di rollback per la logica aziendale

Due aspetti critici di qualsiasi strategia di scomposizione sono il mantenimento della compatibilità con le versioni precedenti e l'implementazione di procedure di rollback complete. Questi elementi interagiscono per aiutare a proteggere le operazioni durante il periodo di transizione. Questa sezione descrive come gestire la compatibilità durante il processo di scomposizione e stabilire efficaci funzionalità di rollback di emergenza che proteggano da potenziali problemi.

Mantieni la compatibilità con le versioni precedenti

Durante la decomposizione del database, il mantenimento della compatibilità con le versioni precedenti è essenziale per transizioni fluide. Mantieni temporaneamente attive le procedure di database esistenti implementando gradualmente nuove funzionalità. Usa il controllo delle versioni per tenere traccia di tutte le modifiche e gestire più versioni del database contemporaneamente. Pianifica un periodo di coesistenza prolungato in cui sia il sistema di origine che quello di destinazione devono funzionare in modo affidabile. Ciò offre il tempo necessario per testare e convalidare il nuovo sistema prima di ritirare i componenti precedenti. Questo approccio riduce al minimo le interruzioni dell'attività e fornisce una rete di sicurezza per il ripristino, se necessario.

Piano di rollback di emergenza

Una strategia di rollback completa è essenziale per una scomposizione sicura del database. Implementa i flag di funzionalità nel codice per controllare quale versione della logica di business è attiva. Ciò consente di passare istantaneamente dalla nuova implementazione a quella originale senza modifiche alla distribuzione. Questo approccio offre un controllo granulare sulla transizione e consente di ripristinare rapidamente i processi in caso di problemi. Mantieni la logica originale come backup verificato e mantieni procedure di rollback dettagliate che specificano i fattori scatenanti, le responsabilità e le fasi di ripristino.

Testa regolarmente questi scenari di rollback in varie condizioni per convalidarne l'efficacia e assicurati che i team abbiano familiarità con le procedure di emergenza. I flag di funzionalità consentono inoltre l'implementazione graduale abilitando selettivamente nuove funzionalità per gruppi di utenti o transazioni specifici. Ciò fornisce un ulteriore livello di mitigazione del rischio durante la transizione.

Disaccoppiamento delle relazioni tra tabelle durante la scomposizione del database

Questa sezione fornisce indicazioni sulla suddivisione delle relazioni tra tabelle complesse e sulle operazioni JOIN durante la scomposizione monolitica del database. Un table join combina le righe di due o più tabelle in base a una colonna correlata tra di esse. L'obiettivo della separazione di queste relazioni è ridurre l'elevato accoppiamento tra le tabelle mantenendo al contempo l'integrità dei dati tra i microservizi.

Questa sezione contiene i seguenti argomenti:

- Strategia di denormalizzazione
- Reference-by-key strategia
- Modello CQRS
- Sincronizzazione dei dati basata sugli eventi
- Implementazione di alternative ai join tra tabelle
- Esempio basato su scenari

Strategia di denormalizzazione

La denormalizzazione è una strategia di progettazione di database che prevede l'introduzione intenzionale della ridondanza combinando o duplicando i dati tra le tabelle. Quando si suddivide un database di grandi dimensioni in database di piccole dimensioni, potrebbe essere opportuno duplicare alcuni dati tra i servizi. Ad esempio, l'archiviazione dei dati di base dei clienti, come nome e indirizzi e-mail, sia in un servizio di marketing che in un servizio ordini elimina la necessità di ricerche costanti tra i diversi servizi. Il servizio di marketing potrebbe aver bisogno delle preferenze dei clienti e delle informazioni di contatto per il targeting delle campagne, mentre il servizio ordini richiede gli stessi dati per l'elaborazione degli ordini e le notifiche. Sebbene ciò crei una certa ridondanza dei dati, può migliorare significativamente le prestazioni e l'indipendenza del servizio, consentendo al team di marketing di gestire le proprie campagne senza dipendere dalle ricerche in tempo reale del servizio clienti.

Quando implementate la denormalizzazione, concentratevi sui campi ad accesso frequente che identificate attraverso un'attenta analisi dei modelli di accesso ai dati. È possibile utilizzare strumenti, ad esempio Oracle AWR report opg_stat_statements, per capire quali dati vengono

comunemente recuperati insieme. Gli esperti del settore possono anche fornire informazioni preziose sui raggruppamenti di dati naturali. Ricorda che la denormalizzazione non è un all-or-nothing approccio, ma solo dati duplicati che migliorano in modo dimostrabile le prestazioni del sistema o riducono le dipendenze complesse.

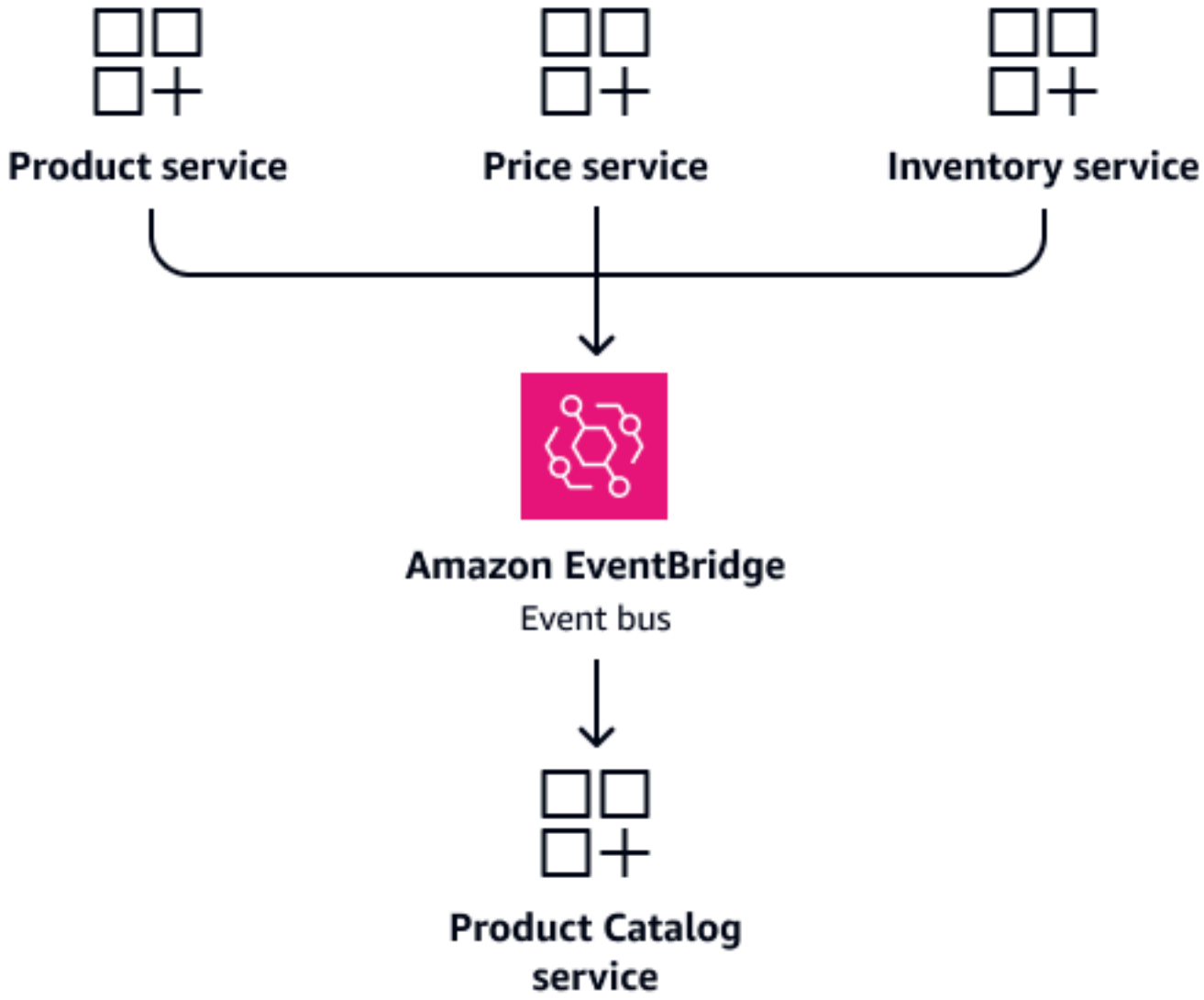
Reference-by-key strategia

Una reference-by-key strategia è un modello di progettazione di database in cui le relazioni tra le entità vengono mantenute tramite chiavi univoche anziché archiviare i dati correlati effettivi. Invece delle tradizionali relazioni con chiavi esterne, i microservizi moderni spesso archiviano solo gli identificatori univoci dei dati correlati. Ad esempio, anziché conservare tutti i dettagli del cliente nella tabella degli ordini, il servizio ordini memorizza solo l'ID cliente e recupera informazioni aggiuntive sul cliente tramite una chiamata API quando necessario. Questo approccio mantiene l'indipendenza del servizio garantendo al contempo l'accesso ai dati correlati.

Modello CQRS

Il pattern Command Query Responsibility Segregation (CQRS) separa le operazioni di lettura e scrittura di un archivio dati. Questo modello è particolarmente utile in sistemi complessi con requisiti di prestazioni elevate, in particolare quelli con carichi asimmetrici. read/write Se l'applicazione necessita spesso di dati combinati da più fonti, è possibile creare un modello CQRS dedicato anziché join complessi. Ad esempio, anziché unire `Inventory` tabelle ad ogni richiesta `ProductPricing`, è consigliabile mantenere una `Product Catalog` tabella consolidata che contenga i dati necessari. I vantaggi di questo approccio possono superare i costi della tabella aggiuntiva.

Prendiamo in considerazione uno scenario in cui `Product Price` i `Inventory` servizi necessitano spesso di informazioni sui prodotti. Invece di configurare questi servizi per accedere direttamente alle tabelle condivise, crea un `Product Catalog` servizio dedicato. Questo servizio mantiene il proprio database che contiene le informazioni consolidate sul prodotto. Funziona come un'unica fonte di verità per le domande relative ai prodotti. Quando i dettagli del prodotto, i prezzi o i livelli di inventario cambiano, i rispettivi servizi possono pubblicare eventi per aggiornare il servizio. `Product Catalog` Ciò garantisce la coerenza dei dati mantenendo al contempo l'indipendenza del servizio. L'immagine seguente mostra questa configurazione, in cui Amazon EventBridge funge da bus di eventi.



Come illustrato nella sezione successivaSincronizzazione dei dati basata sugli eventi, mantenete aggiornato il modello CQRS tramite eventi. Quando i dettagli del prodotto, i prezzi o i livelli di inventario cambiano, i rispettivi servizi pubblicano eventi. Il `Product Catalog` servizio sottoscrive questi eventi e aggiorna la sua visualizzazione consolidata. Ciò fornisce letture veloci senza unioni complesse e mantiene l'indipendenza del servizio.

Sincronizzazione dei dati basata sugli eventi

La sincronizzazione dei dati basata sugli eventi è un modello in cui le modifiche ai dati vengono acquisite e propagate come eventi, il che consente a diversi sistemi o componenti di mantenere gli stati dei dati sincronizzati. Quando i dati cambiano, invece di aggiornare immediatamente tutti i database correlati, pubblica un evento per notificare i servizi sottoscritti. Ad esempio, quando un

cliente modifica l'indirizzo di spedizione nel `Customer` servizio, un `CustomerUpdated` evento avvia gli aggiornamenti del `Order` servizio e del servizio in base alla `Delivery` pianificazione di ciascun servizio. Questo approccio sostituisce le giunzioni rigide tra tabelle con aggiornamenti flessibili e scalabili basati sugli eventi. Alcuni servizi potrebbero contenere per un breve periodo dati obsoleti, ma il compromesso è una migliore scalabilità del sistema e l'indipendenza del servizio.

Implementazione di alternative ai join tra tabelle

Inizia la scomposizione del database con le operazioni di lettura, poiché in genere sono più semplici da migrare e convalidare. Dopo che i percorsi di lettura sono stabili, affronta le operazioni di scrittura più complesse. Per requisiti critici e ad alte prestazioni, prendi in considerazione l'implementazione del modello CQRS. Utilizzate un database separato e ottimizzato per le letture e mantenetene un altro per le scritture.

Crea sistemi resilienti aggiungendo la logica di ripetizione per le chiamate tra servizi e implementando livelli di caching appropriati. Monitora attentamente le interazioni con i servizi e imposta avvisi per problemi di coerenza dei dati. L'obiettivo finale non è la perfetta coerenza ovunque, ma la creazione di servizi indipendenti che funzionino bene mantenendo al contempo una precisione dei dati accettabile per le esigenze aziendali.

La natura disaccoppiata dei microservizi introduce le seguenti nuove complessità nella gestione dei dati:

- I dati vengono distribuiti. I dati ora risiedono in database separati, gestiti da servizi indipendenti.
- La sincronizzazione in tempo reale tra i servizi è spesso poco pratica e richiede un eventuale modello di coerenza.
- Le operazioni che in precedenza si svolgevano all'interno di una singola transazione di database ora si estendono su più servizi.

Per risolvere queste sfide, procedi come segue:

- Implementa un'architettura basata sugli eventi: utilizza le code di messaggi e la pubblicazione degli eventi per propagare le modifiche ai dati tra i servizi. Per ulteriori informazioni, consulta Creazione di architetture basate sugli eventi su Serverless Land.
- Adotta lo schema di orchestrazione della saga: questo modello ti aiuta a gestire le transazioni distribuite e a mantenere l'integrità dei dati tra i servizi. Per ulteriori informazioni, consulta

Creazione di un'applicazione distribuita senza server utilizzando un modello di orchestrazione saga sui blog. AWS

- Progettazione in caso di guasto: incorpora meccanismi di ripetizione dei tentativi, interruttori automatici e transazioni di compensazione per gestire problemi di rete o guasti del servizio.
- Usa la marcatura delle versioni: monitora le versioni dei dati per gestire i conflitti e assicurati che vengano applicati gli aggiornamenti più recenti.
- Riconciliazione regolare: implementa processi periodici di sincronizzazione dei dati per catturare e correggere eventuali incongruenze.

Esempio basato su scenari

L'esempio di schema seguente ha due tabelle, una `Customer` tabella e una tabella: `Order`

```
-- Customer table
CREATE TABLE customer (
    customer_id INT PRIMARY KEY,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    email VARCHAR(255),
    phone VARCHAR(20),
    address TEXT,
    created_at TIMESTAMP
);

-- Order table
CREATE TABLE order (
    order_id INT PRIMARY KEY,
    customer_id INT,
    order_date TIMESTAMP,
    total_amount DECIMAL(10,2),
    status VARCHAR(50),
    FOREIGN KEY (customer_id) REFERENCES customers(id)
);
```

Di seguito è riportato un esempio di come è possibile utilizzare un approccio denormalizzato:

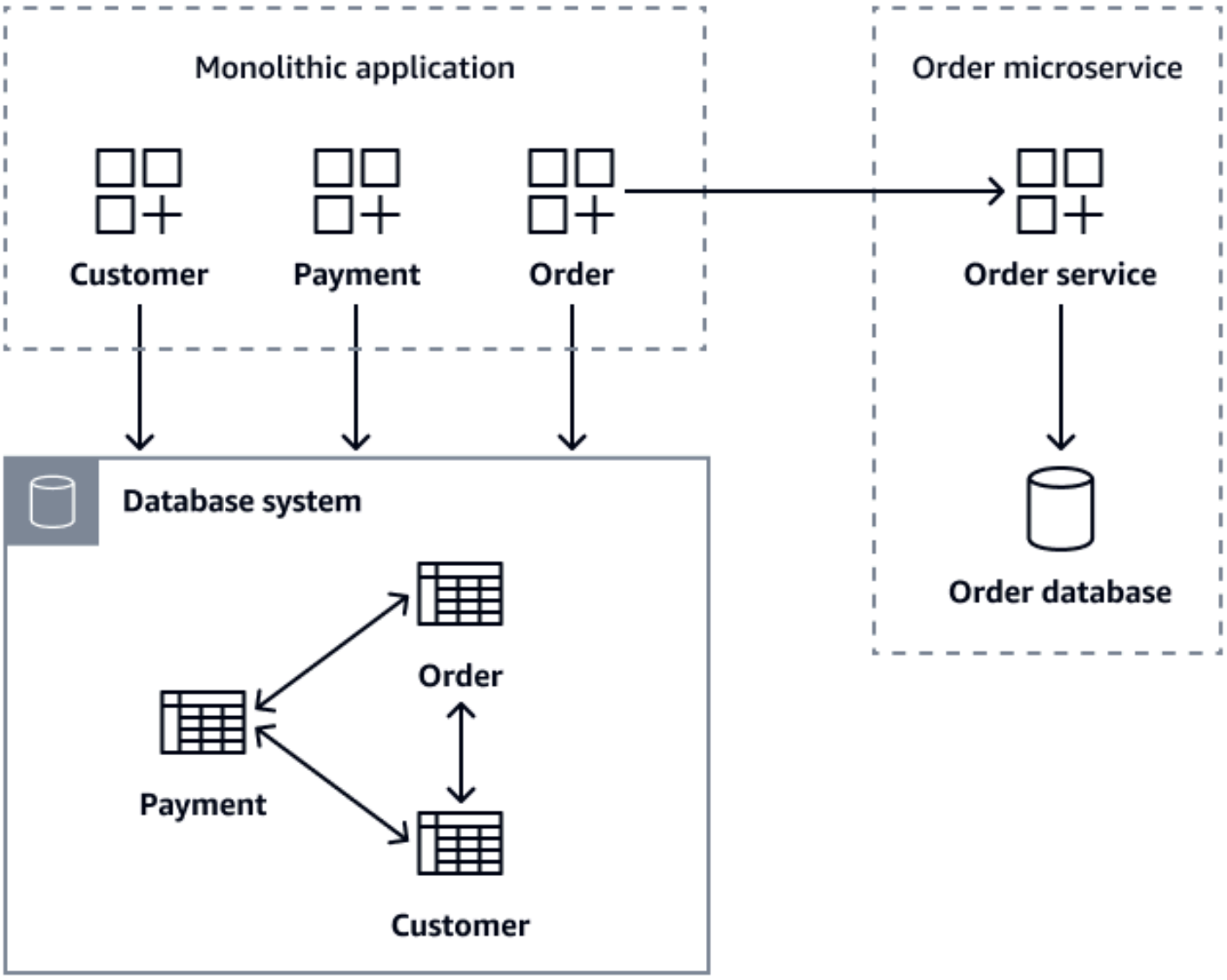
```
CREATE TABLE order (
    order_id INT PRIMARY KEY,
    customer_id INT,                  -- Reference only
```

```
    customer_first_name VARCHAR(100), -- Denormalized
    customer_last_name VARCHAR(100),  -- Denormalized
    customer_email VARCHAR(255),      -- Denormalized
    order_date TIMESTAMP,
    total_amount DECIMAL(10,2),
    status VARCHAR(50)
);
```

La nuova `Order` tabella contiene il nome e gli indirizzi e-mail dei clienti denormalizzati. `customer_id`Viene fatto riferimento a e non esiste alcun vincolo di chiave esterna nella tabella. `Customer` Di seguito sono riportati i vantaggi di questo approccio denormalizzato:

- Il `Order` servizio può visualizzare la cronologia degli ordini con i dettagli del cliente e non richiede chiamate API al microservizio. `Customer`
- Se il `Customer` servizio è inattivo, il `Order` servizio rimane perfettamente funzionante.
- Le richieste per l'elaborazione degli ordini e la creazione di report vengono eseguite più rapidamente.

Il diagramma seguente mostra un'applicazione monolitica che recupera i dati degli ordini utilizzando `getOrder(customer_id)` `getOrder(order_id)``getCustomerOders(customer_id)`, e chiamate `createOrder(Order order)` API al microservizio. `Order`



Durante la migrazione dei microservizi, è possibile mantenere la `Order` tabella nel database monolitico come misura di sicurezza transitoria, garantendo che l'applicazione precedente rimanga funzionale. Tuttavia, è fondamentale che tutte le nuove operazioni relative agli ordini vengano instradate tramite l'API dei `Order` microservizi, che mantiene il proprio database e contemporaneamente scrive nel database legacy come backup. Questo pattern a doppia scrittura fornisce una rete di sicurezza. Consente una migrazione graduale mantenendo la stabilità del sistema. Dopo che tutti i clienti hanno effettuato con successo la migrazione al nuovo microservizio, è possibile rendere obsoleta la `Order` tabella esistente nel database monolitico. Dopo aver scomposto l'applicazione monolitica e il relativo database in microservizi separati`Customer`, mantenere la coerenza dei dati diventa la sfida principale. `Order`

Le migliori pratiche per la scomposizione del database

Quando si scompone un database monolitico, le organizzazioni devono stabilire strutture chiare per tracciare i progressi, mantenere la conoscenza del sistema e affrontare le sfide emergenti. Questa sezione fornisce le migliori pratiche per misurare il successo della decomposizione, conservare la documentazione fondamentale, implementare processi di miglioramento continuo e affrontare le sfide comuni. La comprensione e l'osservanza di queste linee guida consentono di garantire che le iniziative di scomposizione del database offrano i vantaggi previsti, riducendo al minimo le interruzioni operative e il debito tecnico.

Questa sezione contiene i seguenti argomenti:

- Misurare il successo
- Requisiti di documentazione
- Strategia di miglioramento continuo
- Superamento delle sfide più comuni nella decomposizione dei database

Misurare il successo

Tieni traccia del successo della decomposizione attraverso un mix di metriche tecniche, operative e aziendali. Tecnicamente, monitora i tempi di risposta alle query, i miglioramenti dell'operatività del sistema e gli aumenti della frequenza di implementazione. Dal punto di vista operativo, misura la riduzione degli incidenti, la velocità di risoluzione dei problemi e il miglioramento dell'utilizzo delle risorse. Per lo sviluppo, monitora la velocità di implementazione delle funzionalità, l'accelerazione del ciclo di rilascio e la riduzione delle dipendenze tra i team. Gli impatti aziendali dovrebbero tradursi in una riduzione dei costi operativi, in tempi più rapidi time-to-market e in una maggiore soddisfazione dei clienti. Queste metriche vengono spesso definite durante la fase di definizione dell'ambito. Per ulteriori informazioni, consulta Definizione dell'ambito e dei requisiti per la scomposizione del database in questa guida.

Requisiti di documentazione

Conserva la documentazione sull'architettura up-to-date del sistema con limiti di servizio, flussi di dati e specifiche di interfaccia chiari. Utilizza i record decisionali relativi all'architettura (ADRs) per acquisire le decisioni tecniche chiave, inclusi il contesto, le conseguenze e le alternative prese in

considerazione. Ad esempio, documenta il motivo per cui servizi specifici sono stati separati per primi o come sono stati raggiunti determinati compromessi in termini di coerenza dei dati.

Pianifica revisioni mensili dell'architettura per valutare lo stato del sistema attraverso metriche chiave: tendenze delle prestazioni, conformità alla sicurezza e dipendenze tra i servizi. Includi il feedback dei team di sviluppo sulle sfide di integrazione e sui problemi operativi. Questo ciclo di revisione regolare consente di identificare tempestivamente i problemi emergenti e verifica che gli sforzi di scomposizione rimangano in linea con gli obiettivi aziendali.

Strategia di miglioramento continuo

Considera la scomposizione del database come un processo iterativo, non come un progetto isolato. Monitora le metriche delle prestazioni del sistema e le interazioni con i servizi per identificare le opportunità di ottimizzazione. Ogni trimestre, dai priorità alla risoluzione del debito tecnico in base all'impatto operativo e ai costi di manutenzione. Ad esempio, automatizza le operazioni di database eseguite di frequente, migliora la copertura del monitoraggio e perfeziona le procedure di implementazione sulla base di modelli appresi.

Superamento delle sfide più comuni nella decomposizione dei database

L'ottimizzazione delle prestazioni richiede un approccio multiforme. Implementa il caching strategico ai confini del servizio, ottimizza i modelli di query in base all'utilizzo effettivo e monitora continuamente le metriche chiave. Risolvi gli ostacoli alle prestazioni in modo proattivo analizzando le tendenze e stabilendo soglie di intervento chiare.

Le sfide legate alla coerenza dei dati richiedono scelte architettoniche attente. Implementa modelli basati sugli eventi per aggiornamenti su più servizi e utilizza modelli di orchestrazione Saga per transazioni complesse. Definisci confini di servizio chiari e accetta la coerenza finale laddove i requisiti aziendali lo consentano. Questo equilibrio tra coerenza e autonomia del servizio è fondamentale per una corretta decomposizione.

L'eccellenza operativa richiede l'automazione delle attività di routine e delle procedure standardizzate per tutti i servizi. Mantieni un monitoraggio completo con soglie di allarme chiare e investi nella formazione regolare del team per nuovi modelli e strumenti. Questo approccio sistematico alle operazioni promuove l'erogazione affidabile dei servizi gestendo al contempo la complessità.

Domande frequenti sulla scomposizione del database

Questa sezione completa delle domande frequenti affronta le domande e le sfide più comuni che le organizzazioni devono affrontare quando intraprendono progetti di scomposizione dei database. Dalla definizione dell'ambito e dei requisiti iniziali alla migrazione delle stored procedure, queste domande forniscono approfondimenti pratici e approcci strategici per aiutare i team ad affrontare con successo il percorso di modernizzazione del database. Che siate in fase di pianificazione o che stiate già eseguendo la vostra strategia di scomposizione, queste risposte possono aiutarvi a evitare le insidie più comuni e a implementare le migliori pratiche per ottenere risultati ottimali.

Questa sezione contiene i seguenti argomenti:

- FAQs sulla definizione dell'ambito e dei requisiti
- FAQs sul controllo dell'accesso ai database
- FAQs sull'analisi della coesione e dell'accoppiamento
- FAQs sulla migrazione della logica aziendale al livello applicativo

FAQs sulla definizione dell'ambito e dei requisiti

La Definizione dell'ambito e dei requisiti per la scomposizione del database sezione di questa guida illustra come analizzare le interazioni, mappare le dipendenze e stabilire criteri di successo. Questa sezione delle domande frequenti affronta le domande chiave sulla definizione e la gestione dei confini del progetto. Che si tratti di vincoli tecnici poco chiari, esigenze dipartimentali contrastanti o requisiti aziendali in evoluzione, queste FAQs forniscono indicazioni pratiche per mantenere un approccio equilibrato.

Questa sezione contiene le seguenti domande:

- Quanto deve essere dettagliata la definizione iniziale dell'ambito?
- Cosa succede se scopro dipendenze aggiuntive dopo l'avvio del progetto?
- Come posso gestire le parti interessate di diversi reparti che hanno requisiti contrastanti?
- Qual è il modo migliore per valutare i vincoli tecnici quando la documentazione è scadente o obsoleta?
- Come posso bilanciare le esigenze aziendali immediate con gli obiettivi tecnici a lungo termine?
- Come posso assicurarmi che i requisiti critici non vengano trascurati dalle parti interessate silenziose?

- Queste raccomandazioni si applicano ai database mainframe monolitici?

Quanto deve essere dettagliata la definizione iniziale dell'ambito?

Partendo dalle esigenze dei clienti, definisci l'ambito del progetto con dettagli sufficienti per identificare i limiti del sistema e le dipendenze critiche, mantenendo al contempo la flessibilità necessaria per l'individuazione. Mappa gli elementi essenziali, tra cui le interfacce di sistema, le principali parti interessate e i principali vincoli tecnici. Inizia in piccolo selezionando una parte del sistema limitata e a basso rischio che fornisca un valore misurabile. Questo approccio aiuta i team ad apprendere e adattare le strategie prima di affrontare componenti più complessi.

Documenta i requisiti aziendali critici che favoriscono lo sforzo di scomposizione, ma evita di specificare eccessivamente i dettagli che potrebbero cambiare durante l'implementazione. Questo approccio bilanciato assicura che i team possano procedere con chiarezza pur rimanendo adattabili alle nuove intuizioni e alle sfide che emergono durante il percorso di modernizzazione.

Cosa succede se scopro dipendenze aggiuntive dopo l'avvio del progetto?

Aspettatevi di scoprire dipendenze aggiuntive man mano che il progetto avanza. Mantieni un registro delle dipendenze in tempo reale ed esegui revisioni periodiche dell'ambito per valutare l'impatto su tempistiche e risorse. Implementa un chiaro processo di gestione delle modifiche e includi un periodo di buffer nei piani di progetto per gestire scoperte inaspettate. L'obiettivo non è prevenire le modifiche ma gestirle in modo efficace. Questo aiuta i team ad adattarsi rapidamente mantenendo lo slancio del progetto.

Come posso gestire le parti interessate di diversi reparti che hanno requisiti contrastanti?

Gestisci i requisiti dipartimentali contrastanti attraverso una chiara definizione delle priorità basata sul valore aziendale e sull'impatto sul sistema. Assicuratevi la sponsorizzazione dei dirigenti per prendere decisioni chiave e risolvere rapidamente i conflitti. Pianifica riunioni periodiche di allineamento degli stakeholder per discutere dei compromessi e mantenere la trasparenza. Documenta tutte le decisioni e le relative motivazioni per promuovere una comunicazione chiara e mantenere lo slancio del progetto. Concentra le discussioni sui vantaggi aziendali quantificabili piuttosto che sulle preferenze dei reparti.

Qual è il modo migliore per valutare i vincoli tecnici quando la documentazione è scadente o obsoleta?

Di fronte a una documentazione scadente, combinate l'analisi tradizionale con i moderni strumenti di intelligenza artificiale. Utilizzate modelli linguistici di grandi dimensioni (LLMs) per analizzare gli archivi di codice, i log e la documentazione esistente al fine di identificare modelli e potenziali vincoli. Intervistate sviluppatori e architetti di database esperti per convalidare i risultati dell'IA e scoprire vincoli non documentati. Implementa strumenti di monitoraggio con funzionalità di intelligenza artificiale avanzate per osservare il comportamento del sistema e prevedere potenziali problemi.

Crea piccoli esperimenti tecnici che convalidino le tue ipotesi. Puoi utilizzare strumenti di test basati sull'intelligenza artificiale per accelerare il processo. Documenta i risultati in una knowledge base che può essere continuamente migliorata tramite aggiornamenti assistiti dall'intelligenza artificiale. Prendi in considerazione la possibilità di coinvolgere esperti in materia per aree complesse e utilizza gli strumenti di programmazione in coppia basati sull'intelligenza artificiale per accelerare le loro attività di analisi e documentazione.

Come posso bilanciare le esigenze aziendali immediate con gli obiettivi tecnici a lungo termine?

Crea una roadmap di progetto graduale che allinei le esigenze aziendali immediate con gli obiettivi tecnici a lungo termine. Identifica tempestivamente le vittorie rapide che offrano valore tangibile in modo da aumentare la fiducia delle parti interessate. Suddividi la scomposizione in tappe fondamentali chiare. Ciascuno di essi dovrebbe offrire vantaggi aziendali misurabili, progredendo nel contempo verso gli obiettivi architettonici. Mantieni la flessibilità necessaria per soddisfare le esigenze aziendali urgenti attraverso revisioni e adeguamenti regolari della roadmap.

Come posso assicurarmi che i requisiti critici non vengano trascurati dalle parti interessate silenziose?

Mappa tutte le potenziali parti interessate all'interno dell'organizzazione, compresi i proprietari dei sistemi a valle e gli utenti indiretti. Crea più canali di feedback attraverso interviste strutturate, workshop e sessioni di revisione regolari. Crea proof-of-concepts prototipi per rendere tangibili i requisiti e innescare discussioni significative. Ad esempio, una semplice dashboard che mostra le dipendenze del sistema spesso rivela parti interessate e requisiti nascosti che inizialmente non erano evidenti.

Conduci sessioni di convalida regolari con le parti interessate che si esprimono a voce bassa e assicurati che tutti i punti di vista vengano presi in considerazione. Le informazioni critiche spesso provengono da coloro che sono più vicini alle operazioni quotidiane piuttosto che dalle voci più forti durante le riunioni di pianificazione.

Queste raccomandazioni si applicano ai database mainframe monolitici?

La metodologia descritta in questa guida si applica anche alla scomposizione di database mainframe monolitici. Le sfide principali di questi database sono la gestione dei requisiti delle varie parti interessate. Le raccomandazioni tecnologiche contenute in questa guida potrebbero applicarsi ai database mainframe monolitici. Se il mainframe dispone di un database relazionale, ad esempio un database OLTP (Online Transaction Processing), valgono molte delle raccomandazioni. Per i database di elaborazione analitica online (OLAP), come quelli utilizzati per generare report aziendali, valgono solo alcune delle raccomandazioni.

FAQs sul controllo dell'accesso ai database

Il controllo dell'accesso al database utilizzando il modello del servizio wrapper del database è discusso nella Controllo dell'accesso al database durante la decomposizione sezione di questa guida. Questa sezione delle domande frequenti affronta preoccupazioni e domande comuni sull'introduzione di un servizio di database wrapper, incluso il suo potenziale impatto sulle prestazioni, sulla gestione delle stored procedure esistenti, sulla gestione di transazioni complesse e sulla supervisione delle modifiche allo schema.

Questa sezione contiene le seguenti domande:

- Il servizio wrapper non diventerà un nuovo collo di bottiglia?
- Cosa succede alle stored procedure esistenti?
- Come posso gestire le modifiche allo schema durante la transizione?

Il servizio wrapper non diventerà un nuovo collo di bottiglia?

Sebbene il servizio database wrapper aggiunga un hop di rete aggiuntivo, l'impatto è generalmente minimo. È possibile scalare il servizio orizzontalmente e i vantaggi dell'accesso controllato in genere superano i bassi costi in termini di prestazioni. Consideratelo un compromesso temporaneo tra prestazioni e manutenibilità.

Cosa succede alle stored procedure esistenti?

Inizialmente, il servizio wrapper del database può esporre le stored procedure come metodi di servizio. Nel tempo, è possibile spostare gradualmente la logica al livello dell'applicazione, migliorando così i test e il controllo delle versioni. Migra la logica aziendale in modo incrementale per ridurre al minimo i rischi.

Come posso gestire le modifiche allo schema durante la transizione?

Centralizza il controllo delle modifiche allo schema tramite il team del servizio wrapper. Questo team è responsabile del mantenimento di una visibilità completa su tutti i consumatori. Questo team esamina le modifiche proposte per verificarne l'impatto a livello di sistema, si coordina con i team interessati e implementa le modifiche utilizzando un processo di distribuzione controllato. Ad esempio, quando si aggiungono nuovi campi, questo team dovrebbe mantenere la compatibilità con le versioni precedenti implementando valori predefiniti o consentendo inizialmente i valori Null.

Stabilisci un chiaro processo di gestione delle modifiche che includa la valutazione dell'impatto, i requisiti di test e le procedure di rollback. Utilizza gli strumenti di controllo delle versioni del database e mantieni una documentazione chiara di tutte le modifiche. Questo approccio centralizzato impedisce che le modifiche allo schema interrompano i servizi dipendenti e mantiene la stabilità del sistema.

FAQs sull'analisi della coesione e dell'accoppiamento

Comprendere e analizzare efficacemente l'accoppiamento e la coesione del database è fondamentale per una corretta decomposizione del database. L'accoppiamento e la coesione sono discussi nella Analisi della coesione e dell'accoppiamento per la scomposizione del database sezione di questa guida. Questa sezione delle domande frequenti affronta le domande chiave sull'identificazione dei livelli di granularità appropriati, sulla selezione degli strumenti di analisi giusti, sulla documentazione dei risultati e sull'assegnazione delle priorità ai problemi di accoppiamento.

Questa sezione contiene le seguenti domande:

- Come posso identificare il giusto livello di granularità durante l'analisi dell'accoppiamento?
- Quali strumenti posso utilizzare per analizzare l'accoppiamento e la coesione del database?
- Qual è il modo migliore per documentare i risultati dell'accoppiamento e della coesione?
- Come posso dare priorità ai problemi di accoppiamento da affrontare per primi?
- Come posso gestire le transazioni che riguardano più operazioni?

Come posso identificare il giusto livello di granularità durante l'analisi dell'accoppiamento?

Inizia con un'analisi ampia delle relazioni tra i database, quindi approfondisci sistematicamente per identificare i punti di separazione naturali. Utilizza gli strumenti di analisi del database per mappare le relazioni a livello di tabella, le dipendenze dello schema e i confini delle transazioni. Ad esempio, esamina i modelli di join nelle query SQL per comprendere le dipendenze di accesso ai dati. È inoltre possibile analizzare i registri delle transazioni per identificare i limiti dei processi aziendali.

Concentratevi sulle aree in cui l'accoppiamento è naturalmente minimo. Questi spesso si allineano ai confini del dominio aziendale e rappresentano punti di scomposizione ottimali. Nel determinare i limiti di servizio appropriati, prendete in considerazione sia l'accoppiamento tecnico (ad esempio tabelle condivise e chiavi esterne) sia l'accoppiamento aziendale (come i flussi di processo e le esigenze di reporting).

Quali strumenti posso utilizzare per analizzare l'accoppiamento e la coesione del database?

È possibile utilizzare una combinazione di strumenti automatici e analisi manuali per valutare l'accoppiamento e la coesione del database. I seguenti strumenti possono aiutarti in questa valutazione:

- Strumenti di visualizzazione dello schema: puoi utilizzare strumenti come SchemaSpyo pgAdminper generare diagrammi ER. Questi diagrammi rivelano le relazioni tra le tabelle e i potenziali punti di accoppiamento.
- Strumenti di analisi delle query: è possibile utilizzare pg_stat_statementso SQL Server Query Storeidentificare tabelle e modelli di accesso uniti di frequente.
- Strumenti di profilazione del database: strumenti come Oracle SQL Developero MySQL Workbenchforniscono informazioni sulle prestazioni delle query e sulle dipendenze dei dati.
- Strumenti di mappatura delle dipendenze: il AWS Schema Conversion Tool (AWS SCT) può aiutarti a visualizzare le relazioni tra schemi e a identificare componenti strettamente collegati. vFunctionpuò aiutarti a identificare i confini del dominio analizzando i confini funzionali e di dominio dell'applicazione.
- Strumenti di monitoraggio delle transazioni: è possibile utilizzare strumenti specifici del database, come Oracle Enterprise Manageror SQL Server Extended Events, per analizzare i limiti delle transazioni.

- Strumenti di migrazione della logica aziendale: puoi utilizzare Ispireri nostri strumenti di intelligenza artificiale generativa, come Amazon Q Developer o Kiro, per convertire la logica di business del database per il livello applicativo, come la conversione in Java.

Combina queste analisi automatizzate con la revisione manuale dei processi aziendali e delle conoscenze del dominio per comprendere appieno l'accoppiamento dei sistemi. Questo approccio multiforme assicura che nella strategia di scomposizione vengano prese in considerazione sia le prospettive tecniche che quelle commerciali.

Qual è il modo migliore per documentare i risultati dell'accoppiamento e della coesione?

Crea una documentazione completa che visualizzi le relazioni e i modelli di utilizzo del database. Di seguito sono riportati i tipi di risorse che è possibile utilizzare per registrare i risultati:

- Matrici di dipendenza: mappano le dipendenze delle tabelle ed evidenziano le aree ad alto accoppiamento.
- Diagrammi di relazione: utilizza i diagrammi ER per mostrare le connessioni dello schema e le relazioni con chiavi esterne.
- Mappe termiche sull'utilizzo delle tabelle: visualizza la frequenza delle query e i modelli di accesso ai dati tra le tabelle.
- Diagrammi di flusso delle transazioni: documenta le transazioni multitavola e i relativi limiti.
- Mappe dei confini dei domini: delinea i potenziali confini dei servizi in base ai domini aziendali.

Combina questi elementi in un documento e aggiornalo regolarmente man mano che la scomposizione avanza. Per i diagrammi, puoi usare strumenti come o. draw.ioLucidchart Prendi in considerazione l'implementazione di un wiki per facilitare l'accesso e la collaborazione del team. Questo approccio documentale multiforme fornisce una comprensione chiara e condivisa dell'accoppiamento e della coesione del sistema.

Come posso dare priorità ai problemi di accoppiamento da affrontare per primi?

Dai priorità ai problemi di accoppiamento sulla base di una valutazione equilibrata dei fattori aziendali e tecnici. Valuta ogni problema in base all'impatto aziendale (ad esempio ricavi ed esperienza

del cliente), al rischio tecnico (come la stabilità del sistema e l'integrità dei dati), allo sforzo di implementazione e alle capacità del team. Crea una matrice di prioritizzazione che attribuisca un punteggio a ogni problema da 1 a 5 in queste dimensioni. Questa matrice ti aiuta a identificare le opportunità più preziose con rischi gestibili.

Inizia con modifiche ad alto impatto e a basso rischio, in linea con le competenze esistenti del team. Questo ti aiuta a rafforzare la fiducia organizzativa e lo slancio per cambiamenti più complessi. Questo approccio promuove un'esecuzione realistica e massimizza il valore aziendale. Rivedi e modifica regolarmente le priorità per mantenere l'allineamento con le mutevoli esigenze aziendali e la capacità del team.

Come posso gestire le transazioni che riguardano più operazioni?

Gestisci transazioni con più operazioni attraverso un coordinamento dei livelli di servizio progettato con cura. Implementa modelli saga per transazioni distribuite complesse. Suddividili in fasi più piccole e reversibili che possono essere gestite in modo indipendente. Ad esempio, un flusso di elaborazione degli ordini potrebbe essere suddiviso in fasi separate per il controllo dell'inventario, l'elaborazione dei pagamenti e la creazione degli ordini, ciascuna con il proprio meccanismo di compensazione.

Ove possibile, riprogetta le operazioni in modo che siano più atomiche, il che riduce la necessità di transazioni distribuite. Quando le transazioni distribuite sono inevitabili, implementa solidi meccanismi di tracciamento e compensazione per promuovere la coerenza dei dati. Monitora i tassi di completamento delle transazioni e implementa procedure chiare di ripristino degli errori per mantenere l'affidabilità del sistema.

FAQs sulla migrazione della logica aziendale al livello applicativo

La migrazione della logica aziendale dal database al livello applicativo è un aspetto critico e complesso della modernizzazione del database. Questa migrazione della logica aziendale è illustrata nella Migrazione della logica aziendale dal database al livello applicativo sezione di questa guida. Questa sezione delle domande frequenti affronta le domande più comuni sulla gestione efficace di questa transizione, dalla selezione dei candidati iniziali per la migrazione alla gestione di complesse stored procedure e trigger.

Questa sezione contiene le seguenti domande:

- Come posso identificare le stored procedure da migrare per prime?
- Quali sono i rischi legati allo spostamento della logica al livello applicativo?

- Come posso mantenere le prestazioni quando sposto la logica dal database?
- Cosa devo fare con le stored procedure complesse che coinvolgono più tabelle?
- Come posso gestire i trigger del database durante la migrazione?
- Qual è il modo migliore per testare la logica aziendale migrata?
- Come posso gestire il periodo di transizione quando esistono sia la logica del database che quella dell'applicazione?
- Come posso gestire gli scenari di errore a livello di applicazione che in precedenza erano gestiti dal database?

Come posso identificare le stored procedure da migrare per prime?

Inizia identificando le stored procedure che offrono la migliore combinazione di basso rischio e alto valore di apprendimento. Concentrati su procedure che abbiano dipendenze minime, funzionalità chiare e un impatto aziendale non critico. Questi sono i candidati ideali per la migrazione iniziale perché aiutano il team a creare fiducia e a stabilire modelli. Ad esempio, scegliete procedure che gestiscono semplici operazioni sui dati rispetto a quelle che gestiscono transazioni complesse o logiche aziendali critiche.

Utilizza gli strumenti di monitoraggio del database per analizzare i modelli di utilizzo e identificare le procedure a cui si accede raramente come prime candidate. Questo approccio riduce al minimo i rischi aziendali fornendo al contempo un'esperienza preziosa per affrontare migrazioni più complesse in un secondo momento. Assegna un punteggio a ogni procedura in base alla complessità, alla criticità aziendale e ai livelli di dipendenza per creare una sequenza di migrazione prioritaria.

Quali sono i rischi legati allo spostamento della logica al livello applicativo?

Lo spostamento della logica del database al livello dell'applicazione presenta diverse sfide chiave. Le prestazioni del sistema possono peggiorare a causa dell'aumento delle chiamate di rete, in particolare per le operazioni a uso intensivo di dati che in precedenza venivano gestite all'interno del database. La gestione delle transazioni diventa più complessa e richiede un attento coordinamento per mantenere l'integrità dei dati nelle operazioni distribuite. Garantire la coerenza dei dati diventa una sfida, in particolare per le operazioni che in precedenza si basavano su vincoli a livello di database.

Altrettanto preoccupanti sono le potenziali interruzioni dell'attività durante la migrazione e la curva di apprendimento per gli sviluppatori. Riduci questi rischi attraverso una pianificazione approfondita,

test approfonditi in ambienti a più livelli e una migrazione graduale che inizia con componenti meno critici. Implementa solide procedure di monitoraggio e rollback per identificare e risolvere rapidamente i problemi di produzione.

Come posso mantenere le prestazioni quando sposto la logica dal database?

Implementa meccanismi di caching appropriati per i dati a cui si accede di frequente, ottimizza i modelli di accesso ai dati per ridurre al minimo le chiamate di rete e utilizza l'elaborazione in batch per operazioni di massa. Per quanto riguarda non-time-critical le operazioni, prendi in considerazione l'elaborazione asincrona per migliorare la reattività del sistema.

Monitora attentamente le metriche delle prestazioni delle applicazioni e ottimizzale secondo necessità. Ad esempio, è possibile sostituire più operazioni a riga singola con l'elaborazione in blocco, memorizzare nella cache i dati di riferimento che vengono modificati di rado e ottimizzare i modelli di query per ridurre il trasferimento dei dati. I test e l'ottimizzazione regolari delle prestazioni aiutano il sistema a mantenere tempi di risposta accettabili e migliorano la manutenibilità e la scalabilità.

Cosa devo fare con le stored procedure complesse che coinvolgono più tabelle?

Avvicinati a procedure memorizzate complesse e multi-tabella attraverso una scomposizione sistematica. Inizia suddividendole in componenti più piccoli e logicamente coerenti e identifica confini chiari delle transazioni e dipendenze dai dati. Crea interfacce di servizio per ogni componente logico. In questo modo è possibile migrare gradualmente senza interrompere le funzionalità esistenti.

Implementa una step-by-step migrazione, iniziando dai componenti meno accoppiati. Per procedure molto complesse, prendi in considerazione la possibilità di mantenerle temporaneamente nel database durante la migrazione di parti più semplici. Questo approccio ibrido mantiene la stabilità del sistema mentre si progredisce verso gli obiettivi architetturali. Monitora continuamente le prestazioni e le funzionalità durante la migrazione e preparati a modificare la strategia in base ai risultati.

Come posso gestire i trigger del database durante la migrazione?

Trasforma i trigger del database in gestori di eventi a livello di applicazione mantenendo la funzionalità del sistema. Sostituisci i trigger sincroni con modelli basati sugli eventi che inviano messaggi alle code per operazioni asincrone. Prendi in considerazione l'utilizzo di Amazon Simple

Notification Service (Amazon SNS) o Amazon Simple Queue Service (Amazon SQS) per le code di messaggi. Per i requisiti di audit, implementa la registrazione a livello di applicazione o utilizza le funzionalità CDC (Database Change Data Capture).

Analizza lo scopo e la criticità di ogni trigger. Alcuni trigger potrebbero essere meglio gestiti dalla logica dell'applicazione, mentre altri potrebbero richiedere modelli di sourcing degli eventi per mantenere la coerenza dei dati. Inizia con trigger semplici, come i registri di controllo, prima di affrontare quelli complessi che gestiscono le regole aziendali o l'integrità dei dati. Monitora attentamente la migrazione per assicurarti che non si verifichino perdite di funzionalità o di coerenza dei dati.

Qual è il modo migliore per testare la logica aziendale migrata?

Implementa un approccio di test a più livelli prima di implementare la logica aziendale migrata. Inizia con i test unitari per il nuovo codice applicativo, quindi aggiungi test di integrazione che coprano i flussi aziendali. end-to-end Esegui implementazioni vecchie e nuove in parallelo, quindi confronta i risultati per convalidare l'equivalenza funzionale. Eseguite test delle prestazioni in varie condizioni di carico per verificare che il comportamento del sistema corrisponda o superi le capacità precedenti.

Utilizza i flag di funzionalità per controllare la distribuzione in modo da poter ripristinare rapidamente il sistema in caso di problemi. Coinvolgi gli utenti aziendali nella convalida, in particolare per i flussi di lavoro critici. Monitora le metriche chiave durante l'implementazione iniziale e aumenta gradualmente il traffico verso la nuova implementazione. In ogni momento, mantieni la capacità di ripristinare la logica del database originale, se necessario.

Come posso gestire il periodo di transizione quando esistono sia la logica del database che quella dell'applicazione?

Quando la logica del database e dell'applicazione sono entrambe in uso, implementa i flag di funzionalità che controllano il flusso del traffico e consentono il passaggio rapido tra le vecchie e le nuove implementazioni. Mantieni un controllo rigoroso delle versioni e documenta chiaramente entrambe le implementazioni e le rispettive responsabilità. Imposta un monitoraggio completo per entrambi i sistemi per identificare rapidamente eventuali discrepanze o problemi di prestazioni.

Stabilisci procedure di rollback chiare per ogni componente migrato in modo da poter tornare alla logica originale, se necessario. Comunica regolarmente con tutte le parti interessate sullo stato della transizione, sui potenziali impatti e sulle procedure di intensificazione. Questo approccio consente di migrare gradualmente, mantenendo al contempo la stabilità del sistema e la fiducia degli stakeholder.

Come posso gestire gli scenari di errore a livello di applicazione che in precedenza erano gestiti dal database?

Sostituisci la gestione degli errori a livello di database con solidi meccanismi a livello di applicazione. Implementa interruttori automatici e riprova la logica per guasti transitori. Utilizza le transazioni di compensazione per mantenere la coerenza dei dati tra le operazioni distribuite. Ad esempio, se l'aggiornamento di un pagamento fallisce, l'applicazione dovrebbe riprovare automaticamente entro limiti definiti e avviare azioni di compensazione, se necessario.

Imposta un monitoraggio e un sistema di avvisi completi per identificare rapidamente i problemi e mantieni registri di controllo dettagliati per la risoluzione dei problemi. Progetta la gestione degli errori in modo che sia il più automatizzata possibile e definisci percorsi di escalation chiari per gli scenari che richiedono l'intervento umano. Questo approccio a più livelli offre la resilienza del sistema mantenendo al contempo l'integrità dei dati e la continuità dei processi aziendali.

Passaggi successivi per la scomposizione del database su AWS

Dopo aver implementato le strategie iniziali di scomposizione del database tramite i servizi di database wrapper e aver spostato la logica di business al livello applicativo, le organizzazioni devono pianificare la loro prossima evoluzione. Questa sezione descrive le considerazioni chiave per continuare il percorso di modernizzazione.

Questa sezione contiene i seguenti argomenti:

- Strategie incrementali per la scomposizione del database
- Considerazioni tecniche per gli ambienti di database distribuiti
- Modifiche organizzative a supporto delle architetture distribuite

Strategie incrementali per la scomposizione del database

La scomposizione del database segue un'evoluzione graduale attraverso tre fasi distinte. I team avvolgono innanzitutto il database monolitico con un servizio di wrapper del database per controllare l'accesso. Quindi iniziano a suddividere i dati in database specifici del servizio, mantenendo al contempo il database principale per le esigenze precedenti. Infine, completano la migrazione della logica aziendale per passare a database di servizi completamente indipendenti.

Durante questo percorso, i team devono implementare modelli accurati di sincronizzazione dei dati e convalidare continuamente la coerenza tra i servizi. Il monitoraggio delle prestazioni diventa fondamentale per identificare e risolvere tempestivamente potenziali problemi. Poiché i servizi si evolvono in modo indipendente, i relativi schemi devono essere ottimizzati in base ai modelli di utilizzo effettivi ed è necessario rimuovere le strutture ridondanti che si sono accumulate nel tempo.

Questo approccio incrementale aiuta a ridurre al minimo i rischi mantenendo al contempo la stabilità del sistema durante tutto il processo di trasformazione.

Considerazioni tecniche per gli ambienti di database distribuiti

In un ambiente di database distribuito, il monitoraggio delle prestazioni diventa essenziale per identificare e risolvere tempestivamente i punti deboli. I team devono implementare sistemi di

monitoraggio completi e strategie di memorizzazione nella cache per mantenere i livelli di prestazioni. Read/write la suddivisione può bilanciare efficacemente i carichi in tutto il sistema.

La coerenza dei dati richiede un'attenta orchestrazione tra i servizi distribuiti. I team dovrebbero implementare eventuali modelli di coerenza laddove appropriato e stabilire chiari limiti di proprietà dei dati. Un monitoraggio affidabile promuove l'integrità dei dati in tutti i servizi.

Inoltre, la sicurezza deve evolversi per adattarsi all'architettura distribuita. Ogni servizio richiede controlli di sicurezza dettagliati e i modelli di accesso richiedono una revisione regolare. Il monitoraggio e il controllo avanzati diventano fondamentali in questo ambiente distribuito.

Modifiche organizzative a supporto delle architetture distribuite

La struttura del team deve essere in linea con i limiti del servizio al fine di definire chiaramente la titolarità e la responsabilità. Le organizzazioni devono stabilire nuovi modelli di comunicazione e sviluppare capacità tecniche aggiuntive all'interno dei team. Questa struttura dovrebbe supportare sia la manutenzione dei servizi esistenti sia la continua evoluzione dell'architettura.

È necessario aggiornare i processi operativi per gestire l'architettura distribuita. I team devono modificare le procedure di implementazione, adattare i processi di risposta agli incidenti ed evolvere le pratiche di gestione delle modifiche per coordinarsi tra più servizi.

Resources

Le seguenti risorse e strumenti aggiuntivi possono aiutare la tua organizzazione nel percorso di scomposizione del database.

AWS Guida prescrittiva

- Migrazione dei database verso OracleCloud AWS
- Opzioni di ripiattaforma per on Oracle DatabaseAWS
- Modelli di progettazione, architetture e implementazioni del cloud

AWS post sul blog

- Migra la logica di business dal database all'applicazione per un'innovazione e una flessibilità più rapide

Servizi AWS

- AWS Application Migration Service
- AWS Database Migration Service (AWS DMS)
- Migration Evaluator
- AWS Schema Conversion Tool (AWS SCT)
- AWS Transform

Altri strumenti

- AppEngine (sito Web Dynatrace)
- Oracle Automatic Workload Repository (sito Web Oracle)
- CAST Imaging (sito Web CAST)
- Kiro (sito Web Kiro)
- pgAdmin (sito Web pgAdmin)
- pg_stat_statements (sito Web PostgreSQL)

- SchemaSpy (sito Web SchemaSpy)
- SQL Developer (sito Web Oracle)
- SQLWays (sito Web Ispirer)
- vFunction (sito Web vFunction)

Altre risorse

- Da Monolith a microservizi (sito Web) O'Reilly

Cronologia dei documenti

La tabella seguente descrive le modifiche significative apportate a questa guida. Per ricevere notifiche sugli aggiornamenti futuri, puoi abbonarti a un feed RSS.

Modifica	Descrizione	Data
Domande frequenti sui mainframe e sugli strumenti di intelligenza artificiale	Abbiamo aggiunto la sezione Questi consigli si applicano ai database mainframe monolitici? Domande frequenti e abbiamo aggiunto ulteriori informazioni sugli strumenti di intelligenza artificiale che è possibile utilizzare durante la scomposizione del database.	14 ottobre 2025
Pubblicazione iniziale	—	30 settembre 2025

AWS Glossario delle linee guida prescrittive

I seguenti sono termini di uso comune nelle strategie, nelle guide e nei modelli forniti da AWS Prescriptive Guidance. Per suggerire voci, utilizza il link Fornisci feedback alla fine del glossario.

Numeri

7 R

Sette strategie di migrazione comuni per trasferire le applicazioni sul cloud. Queste strategie si basano sulle 5 R identificate da Gartner nel 2011 e sono le seguenti:

- Rifattorizzare/riprogettare: trasferisci un'applicazione e modifica la sua architettura sfruttando appieno le funzionalità native del cloud per migliorare l'agilità, le prestazioni e la scalabilità. Ciò comporta in genere la portabilità del sistema operativo e del database. Esempio: migra il tuo database Oracle locale all'edizione compatibile con Amazon Aurora PostgreSQL.
- Ridefinire la piattaforma (lift and reshape): trasferisci un'applicazione nel cloud e introduci un certo livello di ottimizzazione per sfruttare le funzionalità del cloud. Esempio: migra il tuo database Oracle locale ad Amazon Relational Database Service (Amazon RDS) per Oracle in. Cloud AWS
- Riacquistare (drop and shop): passa a un prodotto diverso, in genere effettuando la transizione da una licenza tradizionale a un modello SaaS. Esempio: migra il tuo sistema di gestione delle relazioni con i clienti (CRM) su Salesforce.com.
- Eseguire il rehosting (lift and shift): trasferisci un'applicazione sul cloud senza apportare modifiche per sfruttare le funzionalità del cloud. Esempio: migra il database Oracle locale su Oracle su un'istanza in. EC2 Cloud AWS
- Trasferire (eseguire il rehosting a livello hypervisor): trasferisci l'infrastruttura sul cloud senza acquistare nuovo hardware, riscrivere le applicazioni o modificare le operazioni esistenti. Si esegue la migrazione dei server da una piattaforma locale a un servizio cloud per la stessa piattaforma. Esempio: migra un'applicazione suMicrosoft Hyper-V. AWS
- Riesaminare (mantenere): mantieni le applicazioni nell'ambiente di origine. Queste potrebbero includere applicazioni che richiedono una rifattorizzazione significativa che desideri rimandare a un momento successivo e applicazioni legacy che desideri mantenere, perché non vi è alcuna giustificazione aziendale per effettuarne la migrazione.
- Ritirare: disattiva o rimuovi le applicazioni che non sono più necessarie nell'ambiente di origine.

A

ABAC

Vedi controllo degli accessi basato sugli attributi.

servizi astratti

Vedi servizi gestiti.

ACIDO

Vedi atomicità, consistenza, isolamento, durata.

migrazione attiva-attiva

Un metodo di migrazione del database in cui i database di origine e di destinazione vengono mantenuti sincronizzati (utilizzando uno strumento di replica bidirezionale o operazioni di doppia scrittura) ed entrambi i database gestiscono le transazioni provenienti dalle applicazioni di connessione durante la migrazione. Questo metodo supporta la migrazione in piccoli batch controllati anziché richiedere una conversione una tantum. È più flessibile ma richiede più lavoro rispetto alla migrazione attiva-passiva.

migrazione attiva-passiva

Un metodo di migrazione del database in cui i database di origine e di destinazione vengono mantenuti sincronizzati, ma solo il database di origine gestisce le transazioni provenienti dalle applicazioni di connessione mentre i dati vengono replicati nel database di destinazione. Il database di destinazione non accetta alcuna transazione durante la migrazione.

funzione di aggregazione

Una funzione SQL che opera su un gruppo di righe e calcola un singolo valore restituito per il gruppo. Esempi di funzioni aggregate includono `SUM` e. `MAX`

Intelligenza artificiale

Vedi intelligenza artificiale.

AIOps

Guarda le operazioni di intelligenza artificiale.

anonimizzazione

Il processo di eliminazione permanente delle informazioni personali in un set di dati. L'anonimizzazione può aiutare a proteggere la privacy personale. I dati anonimi non sono più considerati dati personali.

anti-modello

Una soluzione utilizzata di frequente per un problema ricorrente in cui la soluzione è controproducente, inefficace o meno efficace di un'alternativa.

controllo delle applicazioni

Un approccio alla sicurezza che consente l'uso solo di applicazioni approvate per proteggere un sistema dal malware.

portfolio di applicazioni

Una raccolta di informazioni dettagliate su ogni applicazione utilizzata da un'organizzazione, compresi i costi di creazione e manutenzione dell'applicazione e il relativo valore aziendale. Queste informazioni sono fondamentali per il processo di scoperta e analisi del portfolio e aiutano a identificare e ad assegnare la priorità alle applicazioni da migrare, modernizzare e ottimizzare.

intelligenza artificiale (IA)

Il campo dell'informatica dedicato all'uso delle tecnologie informatiche per svolgere funzioni cognitive tipicamente associate agli esseri umani, come l'apprendimento, la risoluzione di problemi e il riconoscimento di schemi. Per ulteriori informazioni, consulta la sezione Che cos'è l'intelligenza artificiale?

operazioni di intelligenza artificiale (AIOps)

Il processo di utilizzo delle tecniche di machine learning per risolvere problemi operativi, ridurre gli incidenti operativi e l'intervento umano e aumentare la qualità del servizio. Per ulteriori informazioni su come AIOps viene utilizzata nella strategia di AWS migrazione, consulta la guida all'integrazione delle operazioni.

crittografia asimmetrica

Un algoritmo di crittografia che utilizza una coppia di chiavi, una chiave pubblica per la crittografia e una chiave privata per la decrittografia. Puoi condividere la chiave pubblica perché non viene utilizzata per la decrittografia, ma l'accesso alla chiave privata deve essere altamente limitato.

atomicità, consistenza, isolamento, durabilità (ACID)

Un insieme di proprietà del software che garantiscono la validità dei dati e l'affidabilità operativa di un database, anche in caso di errori, interruzioni di corrente o altri problemi.

Controllo degli accessi basato su attributi (ABAC)

La pratica di creare autorizzazioni dettagliate basate su attributi utente, come reparto, ruolo professionale e nome del team. Per ulteriori informazioni, consulta ABAC AWS nella documentazione AWS Identity and Access Management (IAM).

fonte di dati autorevole

Una posizione in cui è archiviata la versione principale dei dati, considerata la fonte di informazioni più affidabile. È possibile copiare i dati dalla fonte di dati autorevole in altre posizioni allo scopo di elaborarli o modificarli, ad esempio anonimizzandoli, oscurandoli o pseudonimizzandoli.

Zona di disponibilità

Una posizione distinta all'interno di un edificio Regione AWS che è isolata dai guasti in altre zone di disponibilità e offre una connettività di rete economica e a bassa latenza verso altre zone di disponibilità nella stessa regione.

AWS Cloud Adoption Framework (CAF)AWS

Un framework di linee guida e best practice AWS per aiutare le organizzazioni a sviluppare un piano efficiente ed efficace per passare con successo al cloud. AWS CAF organizza le linee guida in sei aree di interesse chiamate prospettive: business, persone, governance, piattaforma, sicurezza e operazioni. Le prospettive relative ad azienda, persone e governance si concentrano sulle competenze e sui processi aziendali; le prospettive relative alla piattaforma, alla sicurezza e alle operazioni si concentrano sulle competenze e sui processi tecnici. Ad esempio, la prospettiva relativa alle persone si rivolge alle parti interessate che gestiscono le risorse umane (HR), le funzioni del personale e la gestione del personale. In questa prospettiva, AWS CAF fornisce linee guida per lo sviluppo delle persone, la formazione e le comunicazioni per aiutare a preparare l'organizzazione all'adozione del cloud di successo. Per ulteriori informazioni, consulta il sito web di AWS CAF e il white paper AWS CAF.

AWS Workload Qualification Framework (WQF)AWS

Uno strumento che valuta i carichi di lavoro di migrazione dei database, consiglia strategie di migrazione e fornisce stime del lavoro. AWS WQF è incluso in (). AWS Schema Conversion Tool AWS SCT Analizza gli schemi di database e gli oggetti di codice, il codice dell'applicazione, le dipendenze e le caratteristiche delle prestazioni e fornisce report di valutazione.

B

bot difettoso

Un bot che ha lo scopo di interrompere o causare danni a individui o organizzazioni.

BCP

Vedi la pianificazione della continuità operativa.

grafico comportamentale

Una vista unificata, interattiva dei comportamenti delle risorse e delle interazioni nel tempo. Puoi utilizzare un grafico comportamentale con Amazon Detective per esaminare tentativi di accesso non riusciti, chiamate API sospette e azioni simili. Per ulteriori informazioni, consulta Dati in un grafico comportamentale nella documentazione di Detective.

sistema big-endian

Un sistema che memorizza per primo il byte più importante. Vedi anche endianness.

Classificazione binaria

Un processo che prevede un risultato binario (una delle due classi possibili). Ad esempio, il modello di machine learning potrebbe dover prevedere problemi come "Questa e-mail è spam o non è spam?" o "Questo prodotto è un libro o un'auto?"

filtro Bloom

Una struttura di dati probabilistica ed efficiente in termini di memoria che viene utilizzata per verificare se un elemento fa parte di un set.

implementazione blu/verde

Una strategia di implementazione in cui si creano due ambienti separati ma identici. La versione corrente dell'applicazione viene eseguita in un ambiente (blu) e la nuova versione dell'applicazione nell'altro ambiente (verde). Questa strategia consente di ripristinare rapidamente il sistema con un impatto minimo.

bot

Un'applicazione software che esegue attività automatizzate su Internet e simula l'attività o l'interazione umana. Alcuni bot sono utili o utili, come i web crawler che indicizzano le informazioni su Internet. Alcuni altri bot, noti come bot dannosi, hanno lo scopo di disturbare o causare danni a individui o organizzazioni.

botnet

Reti di bot infettate da malware e controllate da un'unica parte, nota come bot herder o bot operator. Le botnet sono il meccanismo più noto per scalare i bot e il loro impatto.

ramo

Un'area contenuta di un repository di codice. Il primo ramo creato in un repository è il ramo principale. È possibile creare un nuovo ramo a partire da un ramo esistente e quindi sviluppare funzionalità o correggere bug al suo interno. Un ramo creato per sviluppare una funzionalità viene comunemente detto ramo di funzionalità. Quando la funzionalità è pronta per il rilascio, il ramo di funzionalità viene ricongiunto al ramo principale. Per ulteriori informazioni, consulta Informazioni sulle filiali (documentazione). GitHub

accesso break-glass

In circostanze eccezionali e tramite una procedura approvata, un mezzo rapido per consentire a un utente di accedere a un sito a Account AWS cui in genere non dispone delle autorizzazioni necessarie. Per ulteriori informazioni, vedere l'indicatore Implementate break-glass procedures nella guida Well-Architected AWS .

strategia brownfield

L'infrastruttura esistente nell'ambiente. Quando si adotta una strategia brownfield per un'architettura di sistema, si progetta l'architettura in base ai vincoli dei sistemi e dell'infrastruttura attuali. Per l'espansione dell'infrastruttura esistente, è possibile combinare strategie brownfield e greenfield.

cache del buffer

L'area di memoria in cui sono archiviati i dati a cui si accede con maggiore frequenza.

capacità di business

Azioni intraprese da un'azienda per generare valore (ad esempio vendite, assistenza clienti o marketing). Le architetture dei microservizi e le decisioni di sviluppo possono essere guidate dalle capacità aziendali. Per ulteriori informazioni, consulta la sezione Organizzazione in base alle funzionalità aziendali del whitepaper Esecuzione di microservizi containerizzati su AWS.

pianificazione della continuità operativa (BCP)

Un piano che affronta il potenziale impatto di un evento che comporta l'interruzione dell'attività, come una migrazione su larga scala, sulle operazioni e consente a un'azienda di riprendere rapidamente le operazioni.

C

CAF

Vedi Cloud Adoption AWS Framework.

implementazione canaria

Il rilascio lento e incrementale di una versione agli utenti finali. Quando sei sicuro, distribuisci la nuova versione e sostituisci la versione corrente nella sua interezza.

CCoE

Vedi Cloud Center of Excellence.

CDC

Vedi Change Data Capture.

Change Data Capture (CDC)

Il processo di tracciamento delle modifiche a un'origine dati, ad esempio una tabella di database, e di registrazione dei metadati relativi alla modifica. È possibile utilizzare CDC per vari scopi, ad esempio il controllo o la replica delle modifiche in un sistema di destinazione per mantenere la sincronizzazione.

ingegneria del caos

Introduzione intenzionale di guasti o eventi dirompenti per testare la resilienza di un sistema. Puoi usare AWS Fault Injection Service (AWS FIS) per eseguire esperimenti che stressano i tuoi AWS carichi di lavoro e valutarne la risposta.

CI/CD

Vedi integrazione continua e distribuzione continua.

classificazione

Un processo di categorizzazione che aiuta a generare previsioni. I modelli di ML per problemi di classificazione prevedono un valore discreto. I valori discreti sono sempre distinti l'uno dall'altro. Ad esempio, un modello potrebbe dover valutare se in un'immagine è presente o meno un'auto.

crittografia lato client

Crittografia dei dati a livello locale, prima che il destinatario li Servizio AWS riceva.

Centro di eccellenza cloud (CCoE)

Un team multidisciplinare che guida le iniziative di adozione del cloud in tutta l'organizzazione, tra cui lo sviluppo di best practice per il cloud, la mobilitazione delle risorse, la definizione delle tempistiche di migrazione e la guida dell'organizzazione attraverso trasformazioni su larga scala. Per ulteriori informazioni, consulta gli CCoE post sull' Cloud AWS Enterprise Strategy Blog.

cloud computing

La tecnologia cloud generalmente utilizzata per l'archiviazione remota di dati e la gestione dei dispositivi IoT. Il cloud computing è generalmente collegato alla tecnologia di edge computing.

modello operativo cloud

In un'organizzazione IT, il modello operativo utilizzato per creare, maturare e ottimizzare uno o più ambienti cloud. Per ulteriori informazioni, consulta Building your Cloud Operating Model.

fasi di adozione del cloud

Le quattro fasi che le organizzazioni in genere attraversano quando migrano verso Cloud AWS:

- Progetto: esecuzione di alcuni progetti relativi al cloud per scopi di dimostrazione e apprendimento
- Fondamento: effettuare investimenti fondamentali per scalare l'adozione del cloud (ad esempio, creazione di una landing zone, definizione di una CCo E, definizione di un modello operativo)
- Migrazione: migrazione di singole applicazioni
- Reinvenzione: ottimizzazione di prodotti e servizi e innovazione nel cloud

Queste fasi sono state definite da Stephen Orban nel post sul blog The Journey Toward Cloud-First & the Stages of Adoption on the Enterprise Strategy. Cloud AWS Per informazioni su come si relazionano alla strategia di AWS migrazione, consulta la guida alla preparazione alla migrazione.

CMDB

Vedi database di gestione della configurazione.

repository di codice

Una posizione in cui il codice di origine e altri asset, come documentazione, esempi e script, vengono archiviati e aggiornati attraverso processi di controllo delle versioni. Gli archivi cloud più comuni includono GitHub oBitbucket Cloud. Ogni versione del codice è denominata ramo. In una struttura a microservizi, ogni repository è dedicato a una singola funzionalità. Una singola pipeline CI/CD può utilizzare più repository.

cache fredda

Una cache del buffer vuota, non ben popolata o contenente dati obsoleti o irrilevanti. Ciò influisce sulle prestazioni perché l'istanza di database deve leggere dalla memoria o dal disco principale, il che richiede più tempo rispetto alla lettura dalla cache del buffer.

dati freddi

Dati a cui si accede raramente e che in genere sono storici. Quando si eseguono interrogazioni di questo tipo di dati, le interrogazioni lente sono in genere accettabili. Lo spostamento di questi dati su livelli o classi di storage meno costosi e con prestazioni inferiori può ridurre i costi.

visione artificiale (CV)

Un campo dell'intelligenza artificiale che utilizza l'apprendimento automatico per analizzare ed estrarre informazioni da formati visivi come immagini e video digitali. Ad esempio, Amazon SageMaker AI fornisce algoritmi di elaborazione delle immagini per CV.

deriva della configurazione

Per un carico di lavoro, una modifica della configurazione rispetto allo stato previsto. Potrebbe causare la non conformità del carico di lavoro e in genere è graduale e involontaria.

database di gestione della configurazione (CMDB)

Un repository che archivia e gestisce le informazioni su un database e il relativo ambiente IT, inclusi i componenti hardware e software e le relative configurazioni. In genere si utilizzano i dati di un CMDB nella fase di individuazione e analisi del portafoglio della migrazione.

Pacchetto di conformità

Una raccolta di AWS Config regole e azioni correttive che puoi assemblare per personalizzare i controlli di conformità e sicurezza. È possibile distribuire un pacchetto di conformità come singola entità in una regione Account AWS and o all'interno di un'organizzazione utilizzando un modello YAML. Per ulteriori informazioni, consulta i Conformance Pack nella documentazione. AWS Config

integrazione e distribuzione continua (continuous integration and continuous delivery, CI/CD)

Il processo di automazione delle fasi di origine, compilazione, test, gestione temporanea e produzione del processo di rilascio del software. CI/CD viene comunemente descritto come una pipeline. CI/CD può aiutarvi ad automatizzare i processi, migliorare la produttività, migliorare la qualità del codice e velocizzare le consegne. Per ulteriori informazioni, consulta Vantaggi

della distribuzione continua. CD può anche significare continuous deployment (implementazione continua). Per ulteriori informazioni, consulta Distribuzione continua e implementazione continua a confronto.

CV

Vedi visione artificiale.

D

dati a riposo

Dati stazionari nella rete, ad esempio i dati archiviati.

classificazione dei dati

Un processo per identificare e classificare i dati nella rete in base alla loro criticità e sensibilità. È un componente fondamentale di qualsiasi strategia di gestione dei rischi di sicurezza informatica perché consente di determinare i controlli di protezione e conservazione appropriati per i dati. La classificazione dei dati è un componente del pilastro della sicurezza nel AWS Well-Architected Framework. Per ulteriori informazioni, consulta Classificazione dei dati.

deriva dei dati

Una variazione significativa tra i dati di produzione e i dati utilizzati per addestrare un modello di machine learning o una modifica significativa dei dati di input nel tempo. La deriva dei dati può ridurre la qualità, l'accuratezza e l'equità complessive nelle previsioni dei modelli ML.

dati in transito

Dati che si spostano attivamente attraverso la rete, ad esempio tra le risorse di rete.

rete di dati

Un framework architettonico che fornisce la proprietà distribuita e decentralizzata dei dati con gestione e governance centralizzate.

riduzione al minimo dei dati

Il principio della raccolta e del trattamento dei soli dati strettamente necessari. Praticare la riduzione al minimo dei dati in the Cloud AWS può ridurre i rischi per la privacy, i costi e l'impronta di carbonio delle analisi.

perimetro dei dati

Una serie di barriere preventive nell' AWS ambiente che aiutano a garantire che solo le identità attendibili accedano alle risorse attendibili delle reti previste. Per ulteriori informazioni, consulta Building a data perimeter on. AWS

pre-elaborazione dei dati

Trasformare i dati grezzi in un formato che possa essere facilmente analizzato dal modello di ML. La pre-elaborazione dei dati può comportare la rimozione di determinate colonne o righe e l'eliminazione di valori mancanti, incoerenti o duplicati.

provenienza dei dati

Il processo di tracciamento dell'origine e della cronologia dei dati durante il loro ciclo di vita, ad esempio il modo in cui i dati sono stati generati, trasmessi e archiviati.

soggetto dei dati

Un individuo i cui dati vengono raccolti ed elaborati.

data warehouse

Un sistema di gestione dei dati che supporta la business intelligence, come l'analisi. I data warehouse contengono in genere grandi quantità di dati storici e vengono generalmente utilizzati per interrogazioni e analisi.

linguaggio di definizione del database (DDL)

Istruzioni o comandi per creare o modificare la struttura di tabelle e oggetti in un database.

linguaggio di manipolazione del database (DML)

Istruzioni o comandi per modificare (inserire, aggiornare ed eliminare) informazioni in un database.

DDL

Vedi linguaggio di definizione del database.

deep ensemble

Combinare più modelli di deep learning per la previsione. È possibile utilizzare i deep ensemble per ottenere una previsione più accurata o per stimare l'incertezza nelle previsioni.

deep learning

Un sottocampo del ML che utilizza più livelli di reti neurali artificiali per identificare la mappatura tra i dati di input e le variabili target di interesse.

defense-in-depth

Un approccio alla sicurezza delle informazioni in cui una serie di meccanismi e controlli di sicurezza sono accuratamente stratificati su una rete di computer per proteggere la riservatezza, l'integrità e la disponibilità della rete e dei dati al suo interno. Quando si adotta questa strategia AWS, si aggiungono più controlli a diversi livelli della AWS Organizations struttura per proteggere le risorse. Ad esempio, un defense-in-depth approccio potrebbe combinare l'autenticazione a più fattori, la segmentazione della rete e la crittografia.

amministratore delegato

In AWS Organizations, un servizio compatibile può registrare un account AWS membro per amministrare gli account dell'organizzazione e gestire le autorizzazioni per quel servizio. Questo account è denominato amministratore delegato per quel servizio specifico. Per ulteriori informazioni e un elenco di servizi compatibili, consulta Servizi che funzionano con AWS Organizations nella documentazione di AWS Organizations .

implementazione

Il processo di creazione di un'applicazione, di nuove funzionalità o di correzioni di codice disponibili nell'ambiente di destinazione. L'implementazione prevede l'applicazione di modifiche in una base di codice, seguita dalla creazione e dall'esecuzione di tale base di codice negli ambienti applicativi.

Ambiente di sviluppo

Vedi ambiente.

controllo di rilevamento

Un controllo di sicurezza progettato per rilevare, registrare e avvisare dopo che si è verificato un evento. Questi controlli rappresentano una seconda linea di difesa e avvisano l'utente in caso di eventi di sicurezza che aggirano i controlli preventivi in vigore. Per ulteriori informazioni, consulta Controlli di rilevamento in Implementazione dei controlli di sicurezza in AWS.

mappatura del flusso di valore dello sviluppo (DVSM)

Un processo utilizzato per identificare e dare priorità ai vincoli che influiscono negativamente sulla velocità e sulla qualità nel ciclo di vita dello sviluppo del software. DVSM estende il processo di

mappatura del flusso di valore originariamente progettato per pratiche di produzione snella. Si concentra sulle fasi e sui team necessari per creare e trasferire valore attraverso il processo di sviluppo del software.

gemello digitale

Una rappresentazione virtuale di un sistema reale, ad esempio un edificio, una fabbrica, un'attrezzatura industriale o una linea di produzione. I gemelli digitali supportano la manutenzione predittiva, il monitoraggio remoto e l'ottimizzazione della produzione.

tabella delle dimensioni

In uno schema a stella, una tabella più piccola che contiene gli attributi dei dati quantitativi in una tabella dei fatti. Gli attributi della tabella delle dimensioni sono in genere campi di testo o numeri discreti che si comportano come testo. Questi attributi vengono comunemente utilizzati per il vincolo delle query, il filtraggio e l'etichettatura dei set di risultati.

disastro

Un evento che impedisce a un carico di lavoro o a un sistema di raggiungere gli obiettivi aziendali nella sua sede principale di implementazione. Questi eventi possono essere disastri naturali, guasti tecnici o il risultato di azioni umane, come errori di configurazione involontari o attacchi di malware.

disaster recovery (DR)

La strategia e il processo utilizzati per ridurre al minimo i tempi di inattività e la perdita di dati causati da un disastro. Per ulteriori informazioni, consulta Disaster Recovery of Workloads su AWS: Recovery in the Cloud in the AWS Well-Architected Framework.

DML

Vedi linguaggio di manipolazione del database.

progettazione basata sul dominio

Un approccio allo sviluppo di un sistema software complesso collegandone i componenti a domini in evoluzione, o obiettivi aziendali principali, perseguiti da ciascun componente. Questo concetto è stato introdotto da Eric Evans nel suo libro,Domain-Driven Design: Tackling Complexity in the Heart of Software (Boston: Addison-Wesley Professional, 2003). Per informazioni su come utilizzare la progettazione basata sul dominio con il modello del fico strangolatore (Strangler Fig), consulta la sezione Modernizzazione incrementale dei servizi Web Microsoft ASP.NET (ASMX) legacy utilizzando container e il Gateway Amazon API.

DOTT.

Vedi disaster recovery.

rilevamento della deriva

Tracciamento delle deviazioni da una configurazione di base. Ad esempio, è possibile AWS CloudFormation utilizzarlo per rilevare deviazioni nelle risorse di sistema oppure AWS Control Tower per rilevare cambiamenti nella landing zone che potrebbero influire sulla conformità ai requisiti di governance.

DVSM

Vedi la mappatura del flusso di valore dello sviluppo.

E

EDA

Vedi analisi esplorativa dei dati.

MODIFICA

Vedi scambio elettronico di dati.

edge computing

La tecnologia che aumenta la potenza di calcolo per i dispositivi intelligenti all'edge di una rete IoT. Rispetto al cloud computing, l'edge computing può ridurre la latenza di comunicazione e migliorare i tempi di risposta.

scambio elettronico di dati (EDI)

Lo scambio automatizzato di documenti aziendali tra organizzazioni. Per ulteriori informazioni, vedere Cos'è lo scambio elettronico di dati.

crittografia

Un processo di elaborazione che trasforma i dati in chiaro, leggibili dall'uomo, in testo cifrato.

chiave crittografica

Una stringa crittografica di bit randomizzati generata da un algoritmo di crittografia. Le chiavi possono variare di lunghezza e ogni chiave è progettata per essere imprevedibile e univoca.

endianità

L'ordine in cui i byte vengono archiviati nella memoria del computer. I sistemi big-endian memorizzano per primo il byte più importante. I sistemi little-endian memorizzano per primo il byte meno importante.

endpoint

Vedi service endpoint.

servizio endpoint

Un servizio che puoi ospitare in un cloud privato virtuale (VPC) da condividere con altri utenti. Puoi creare un servizio endpoint con AWS PrivateLink e concedere autorizzazioni ad altri Account AWS o a AWS Identity and Access Management (IAM) principali. Questi account o principali possono connettersi al servizio endpoint in privato creando endpoint VPC di interfaccia. Per ulteriori informazioni, consulta Creazione di un servizio endpoint nella documentazione di Amazon Virtual Private Cloud (Amazon VPC).

pianificazione delle risorse aziendali (ERP)

Un sistema che automatizza e gestisce i processi aziendali chiave (come contabilità, MES e gestione dei progetti) per un'azienda.

crittografia envelope

Il processo di crittografia di una chiave di crittografia con un'altra chiave di crittografia. Per ulteriori informazioni, vedete Envelope encryption nella documentazione AWS Key Management Service (AWS KMS).

ambiente

Un'istanza di un'applicazione in esecuzione. Di seguito sono riportati i tipi di ambiente più comuni nel cloud computing:

- ambiente di sviluppo: un'istanza di un'applicazione in esecuzione disponibile solo per il team principale responsabile della manutenzione dell'applicazione. Gli ambienti di sviluppo vengono utilizzati per testare le modifiche prima di promuoverle negli ambienti superiori. Questo tipo di ambiente viene talvolta definito ambiente di test.
- ambienti inferiori: tutti gli ambienti di sviluppo di un'applicazione, ad esempio quelli utilizzati per le build e i test iniziali.
- ambiente di produzione: un'istanza di un'applicazione in esecuzione a cui gli utenti finali possono accedere. In una CI/CD pipeline, l'ambiente di produzione è l'ultimo ambiente di distribuzione.

- ambienti superiori: tutti gli ambienti a cui possono accedere utenti diversi dal team di sviluppo principale. Si può trattare di un ambiente di produzione, ambienti di preproduzione e ambienti per i test di accettazione da parte degli utenti.

epica

Nelle metodologie agili, categorie funzionali che aiutano a organizzare e dare priorità al lavoro. Le epiche forniscono una descrizione di alto livello dei requisiti e delle attività di implementazione. Ad esempio, le epopee della sicurezza AWS CAF includono la gestione delle identità e degli accessi, i controlli investigativi, la sicurezza dell'infrastruttura, la protezione dei dati e la risposta agli incidenti. Per ulteriori informazioni sulle epiche, consulta la strategia di migrazione AWS , consulta la guida all'implementazione del programma.

ERP

Vedi pianificazione delle risorse aziendali.

analisi esplorativa dei dati (EDA)

Il processo di analisi di un set di dati per comprenderne le caratteristiche principali. Si raccolgono o si aggregano dati e quindi si eseguono indagini iniziali per trovare modelli, rilevare anomalie e verificare ipotesi. L'EDA viene eseguita calcolando statistiche di riepilogo e creando visualizzazioni di dati.

F

tabella dei fatti

Il tavolo centrale in uno schema a stella. Memorizza dati quantitativi sulle operazioni aziendali. In genere, una tabella dei fatti contiene due tipi di colonne: quelle che contengono misure e quelle che contengono una chiave esterna per una tabella di dimensioni.

fallire velocemente

Una filosofia che utilizza test frequenti e incrementali per ridurre il ciclo di vita dello sviluppo. È una parte fondamentale di un approccio agile.

limite di isolamento dei guasti

Nel Cloud AWS, un limite come una zona di disponibilità Regione AWS, un piano di controllo o un piano dati che limita l'effetto di un errore e aiuta a migliorare la resilienza dei carichi di lavoro. Per ulteriori informazioni, consulta AWS Fault Isolation Boundaries.

ramo di funzionalità

Vedi filiale.

caratteristiche

I dati di input che usi per fare una previsione. Ad esempio, in un contesto di produzione, le caratteristiche potrebbero essere immagini acquisite periodicamente dalla linea di produzione.

importanza delle caratteristiche

Quanto è importante una caratteristica per le previsioni di un modello. Di solito viene espresso come punteggio numerico che può essere calcolato con varie tecniche, come Shapley Additive Explanations (SHAP) e gradienti integrati. Per ulteriori informazioni, consulta Interpretabilità del modello di machine learning con AWS.

trasformazione delle funzionalità

Per ottimizzare i dati per il processo di machine learning, incluso l'arricchimento dei dati con fonti aggiuntive, il dimensionamento dei valori o l'estrazione di più set di informazioni da un singolo campo di dati. Ciò consente al modello di ML di trarre vantaggio dai dati. Ad esempio, se suddividi la data "2021-05-27 00:15:37" in "2021", "maggio", "giovedì" e "15", puoi aiutare l'algoritmo di apprendimento ad apprendere modelli sfumati associati a diversi componenti dei dati.

prompt con pochi scatti

Fornire a un LLM un numero limitato di esempi che dimostrino l'attività e il risultato desiderato prima di chiedergli di eseguire un'attività simile. Questa tecnica è un'applicazione dell'apprendimento contestuale, in cui i modelli imparano da esempi (immagini) incorporati nei prompt. I prompt con pochi passaggi possono essere efficaci per attività che richiedono una formattazione, un ragionamento o una conoscenza del dominio specifici. Vedi anche zero-shot prompting.

FGAC

Vedi il controllo granulare degli accessi.

controllo granulare degli accessi (FGAC)

L'uso di più condizioni per consentire o rifiutare una richiesta di accesso.

migrazione flash-cut

Un metodo di migrazione del database che utilizza la replica continua dei dati tramite l'acquisizione dei dati delle modifiche per migrare i dati nel più breve tempo possibile, anziché utilizzare un approccio graduale. L'obiettivo è ridurre al minimo i tempi di inattività.

FM

Vedi modello di base.

modello di fondazione (FM)

Una grande rete neurale di deep learning che si è addestrata su enormi set di dati generalizzati e non etichettati. FMs sono in grado di svolgere un'ampia varietà di attività generali, come comprendere il linguaggio, generare testo e immagini e conversare in linguaggio naturale. Per ulteriori informazioni, consulta Cosa sono i modelli Foundation.

G

IA generativa

Un sottoinsieme di modelli di intelligenza artificiale che sono stati addestrati su grandi quantità di dati e che possono utilizzare un semplice messaggio di testo per creare nuovi contenuti e artefatti, come immagini, video, testo e audio. Per ulteriori informazioni, consulta Cos'è l'IA generativa.

blocco geografico

Vedi restrizioni geografiche.

limitazioni geografiche (blocco geografico)

In Amazon CloudFront, un'opzione per impedire agli utenti di determinati paesi di accedere alle distribuzioni di contenuti. Puoi utilizzare un elenco consentito o un elenco di blocco per specificare i paesi approvati e vietati. Per ulteriori informazioni, consulta Limitare la distribuzione geografica dei contenuti nella CloudFront documentazione.

Flusso di lavoro di GitFlow

Un approccio in cui gli ambienti inferiori e superiori utilizzano rami diversi in un repository di codice di origine. Il flusso di lavoro Gitflow è considerato obsoleto e il flusso di lavoro basato su trunk è l'approccio moderno e preferito.

immagine dorata

Un'istantanea di un sistema o di un software utilizzata come modello per distribuire nuove istanze di quel sistema o software. Ad esempio, nella produzione, un'immagine dorata può essere utilizzata per fornire software su più dispositivi e contribuire a migliorare la velocità, la scalabilità e la produttività nelle operazioni di produzione dei dispositivi.

strategia greenfield

L'assenza di infrastrutture esistenti in un nuovo ambiente. Quando si adotta una strategia greenfield per un'architettura di sistema, è possibile selezionare tutte le nuove tecnologie senza il vincolo della compatibilità con l'infrastruttura esistente, nota anche come brownfield. Per l'espansione dell'infrastruttura esistente, è possibile combinare strategie brownfield e greenfield.

guardrail

Una regola di alto livello che aiuta a governare le risorse, le politiche e la conformità tra le unità organizzative (). OUs I guardrail preventivi applicano le policy per garantire l'allineamento agli standard di conformità. Vengono implementati utilizzando le policy di controllo dei servizi e i limiti delle autorizzazioni IAM. I guardrail di rilevamento rilevano le violazioni delle policy e i problemi di conformità e generano avvisi per porvi rimedio. Sono implementati utilizzando Amazon AWS Config AWS Security Hub CSPM GuardDuty AWS Trusted Advisor, Amazon Inspector e controlli personalizzati AWS Lambda .

H

AH

Vedi disponibilità elevata.

migrazione di database eterogenea

Migrazione del database di origine in un database di destinazione che utilizza un motore di database diverso (ad esempio, da Oracle ad Amazon Aurora). La migrazione eterogenea fa in genere parte di uno sforzo di riprogettazione e la conversione dello schema può essere un'attività complessa. AWS offre AWS SCT che aiuta con le conversioni dello schema.

alta disponibilità (HA)

La capacità di un carico di lavoro di funzionare in modo continuo, senza intervento, in caso di sfide o disastri. I sistemi HA sono progettati per il failover automatico, fornire costantemente prestazioni di alta qualità e gestire carichi e guasti diversi con un impatto minimo sulle prestazioni.

modernizzazione storica

Un approccio utilizzato per modernizzare e aggiornare i sistemi di tecnologia operativa (OT) per soddisfare meglio le esigenze dell'industria manifatturiera. Uno storico è un tipo di database utilizzato per raccogliere e archiviare dati da varie fonti in una fabbrica.

dati di blocco

Una parte di dati storici etichettati che viene trattenuta da un set di dati utilizzata per addestrare un modello di apprendimento automatico. È possibile utilizzare i dati di holdout per valutare le prestazioni del modello confrontando le previsioni del modello con i dati di holdout.

migrazione di database omogenea

Migrazione del database di origine in un database di destinazione che condivide lo stesso motore di database (ad esempio, da Microsoft SQL Server ad Amazon RDS per SQL Server). La migrazione omogenea fa in genere parte di un'operazione di rehosting o ridefinizione della piattaforma. Per migrare lo schema è possibile utilizzare le utilità native del database.

dati caldi

Dati a cui si accede frequentemente, ad esempio dati in tempo reale o dati di traduzione recenti. Questi dati richiedono in genere un livello o una classe di storage ad alte prestazioni per fornire risposte rapide alle query.

hotfix

Una soluzione urgente per un problema critico in un ambiente di produzione. A causa della sua urgenza, un hotfix viene in genere creato al di fuori del tipico DevOps flusso di lavoro di rilascio.

periodo di hypercare

Subito dopo la conversione, il periodo di tempo in cui un team di migrazione gestisce e monitora le applicazioni migrate nel cloud per risolvere eventuali problemi. In genere, questo periodo dura da 1 a 4 giorni. Al termine del periodo di hypercare, il team addetto alla migrazione in genere trasferisce la responsabilità delle applicazioni al team addetto alle operazioni cloud.

I

IaC

Considera l'infrastruttura come codice.

Policy basata su identità

Una policy allegata a uno o più principi IAM che definisce le relative autorizzazioni all'interno dell' Cloud AWS ambiente.

applicazione inattiva

Un'applicazione che prevede un uso di CPU e memoria medio compreso tra il 5% e il 20% in un periodo di 90 giorni. In un progetto di migrazione, è normale ritirare queste applicazioni o mantenerle on-premise.

IIoT

Vedi Industrial Internet of Things.

infrastruttura immutabile

Un modello che implementa una nuova infrastruttura per i carichi di lavoro di produzione anziché aggiornare, applicare patch o modificare l'infrastruttura esistente. Le infrastrutture immutabili sono intrinsecamente più coerenti, affidabili e prevedibili delle infrastrutture mutabili. Per ulteriori informazioni, consulta la best practice Deploy using immutable infrastructure in Well-Architected AWS Framework.

VPC in ingresso (ingress)

In un'architettura AWS multi-account, un VPC che accetta, ispeziona e indirizza le connessioni di rete dall'esterno di un'applicazione. La AWS Security Reference Architecture consiglia di configurare l'account di rete con funzionalità in entrata, in uscita e di ispezione VPCs per proteggere l'interfaccia bidirezionale tra l'applicazione e la rete Internet in generale.

migrazione incrementale

Una strategia di conversione in cui si esegue la migrazione dell'applicazione in piccole parti anziché eseguire una conversione singola e completa. Ad esempio, inizialmente potresti spostare solo alcuni microservizi o utenti nel nuovo sistema. Dopo aver verificato che tutto funzioni correttamente, puoi spostare in modo incrementale microservizi o utenti aggiuntivi fino alla disattivazione del sistema legacy. Questa strategia riduce i rischi associati alle migrazioni di grandi dimensioni.

Industria 4.0

Un termine introdotto da Klaus Schwab nel 2016 per riferirsi alla modernizzazione dei processi di produzione attraverso progressi in termini di connettività, dati in tempo reale, automazione, analisi e AI/ML.

infrastruttura

Tutte le risorse e gli asset contenuti nell'ambiente di un'applicazione.

infrastruttura come codice (IaC)

Il processo di provisioning e gestione dell'infrastruttura di un'applicazione tramite un insieme di file di configurazione. Il processo IaC è progettato per aiutarti a centralizzare la gestione dell'infrastruttura, a standardizzare le risorse e a dimensionare rapidamente, in modo che i nuovi ambienti siano ripetibili, affidabili e coerenti.

IIoInternet delle cose industriale (T)

L'uso di sensori e dispositivi connessi a Internet nei settori industriali, come quello manifatturiero, energetico, automobilistico, sanitario, delle scienze della vita e dell'agricoltura. Per ulteriori informazioni, vedere Creazione di una strategia di trasformazione digitale per l'Internet of Things (IIoT) industriale.

VPC di ispezione

In un'architettura AWS multi-account, un VPC centralizzato che gestisce le ispezioni del traffico di rete tra VPCs (nello stesso o in modo diverso Regioni AWS), Internet e le reti locali. La AWS Security Reference Architecture consiglia di configurare l'account di rete con informazioni in entrata, in uscita e di ispezione VPCs per proteggere l'interfaccia bidirezionale tra l'applicazione e Internet in generale.

Internet of Things (IoT)

La rete di oggetti fisici connessi con sensori o processori incorporati che comunicano con altri dispositivi e sistemi tramite Internet o una rete di comunicazione locale. Per ulteriori informazioni, consulta Cos'è l'IoT?

interpretabilità

Una caratteristica di un modello di machine learning che descrive il grado in cui un essere umano è in grado di comprendere in che modo le previsioni del modello dipendono dai suoi input. Per ulteriori informazioni, vedere Interpretabilità del modello di machine learning con. AWS

IoT

Vedi Internet of Things.

libreria di informazioni IT (ITIL)

Una serie di best practice per offrire servizi IT e allinearli ai requisiti aziendali. ITIL fornisce le basi per ITSM.

gestione dei servizi IT (ITSM)

Attività associate alla progettazione, implementazione, gestione e supporto dei servizi IT per un'organizzazione. Per informazioni sull'integrazione delle operazioni cloud con gli strumenti ITSM, consulta la guida all'integrazione delle operazioni.

ITIL

Vedi la libreria di informazioni IT.

ITSM

Vedi Gestione dei servizi IT.

L

controllo degli accessi basato su etichette (LBAC)

Un'implementazione del controllo di accesso obbligatorio (MAC) in cui agli utenti e ai dati stessi viene assegnato esplicitamente un valore di etichetta di sicurezza. L'intersezione tra l'etichetta di sicurezza utente e l'etichetta di sicurezza dei dati determina quali righe e colonne possono essere visualizzate dall'utente.

zona di destinazione

Una landing zone è un AWS ambiente multi-account ben progettato, scalabile e sicuro. Questo è un punto di partenza dal quale le organizzazioni possono avviare e distribuire rapidamente carichi di lavoro e applicazioni con fiducia nel loro ambiente di sicurezza e infrastruttura. Per ulteriori informazioni sulle zone di destinazione, consulta la sezione Configurazione di un ambiente AWS multi-account sicuro e scalabile.

modello linguistico di grandi dimensioni (LLM)

Un modello di intelligenza artificiale di deep learning preaddestrato su una grande quantità di dati. Un LLM può svolgere più attività, come rispondere a domande, riepilogare documenti, tradurre testo in altre lingue e completare frasi. Per ulteriori informazioni, consulta Cosa sono. LLMs

migrazione su larga scala

Una migrazione di 300 o più server.

BIANCO

Vedi controllo degli accessi basato su etichette.

Privilegio minimo

La best practice di sicurezza per la concessione delle autorizzazioni minime richieste per eseguire un'attività. Per ulteriori informazioni, consulta Applicazione delle autorizzazioni del privilegio minimo nella documentazione di IAM.

eseguire il rehosting (lift and shift)

Vedi 7 R.

sistema little-endian

Un sistema che memorizza per primo il byte meno importante. Vedi anche endianità.

LLM

Vedi modello linguistico di grandi dimensioni.

ambienti inferiori

Vedi ambiente.

M

machine learning (ML)

Un tipo di intelligenza artificiale che utilizza algoritmi e tecniche per il riconoscimento e l'apprendimento di schemi. Il machine learning analizza e apprende dai dati registrati, come i dati dell'Internet delle cose (IoT), per generare un modello statistico basato su modelli. Per ulteriori informazioni, consulta la sezione Machine learning.

ramo principale

Vedi filiale.

malware

Software progettato per compromettere la sicurezza o la privacy del computer. Il malware potrebbe interrompere i sistemi informatici, divulgare informazioni sensibili o ottenere accessi non autorizzati. Esempi di malware includono virus, worm, ransomware, trojan horse, spyware e keylogger.

servizi gestiti

Servizi AWS per cui AWS gestisce il livello di infrastruttura, il sistema operativo e le piattaforme e si accede agli endpoint per archiviare e recuperare i dati. Amazon Simple Storage Service

(Amazon S3) Simple Storage Service (Amazon S3) e Amazon DynamoDB sono esempi di servizi gestiti. Questi sono noti anche come servizi astratti.

sistema di esecuzione della produzione (MES)

Un sistema software per tracciare, monitorare, documentare e controllare i processi di produzione che convertono le materie prime in prodotti finiti in officina.

MAP

Vedi Migration Acceleration Program.

meccanismo

Un processo completo in cui si crea uno strumento, si promuove l'adozione dello strumento e quindi si esaminano i risultati per apportare le modifiche. Un meccanismo è un ciclo che si rafforza e si migliora man mano che funziona. Per ulteriori informazioni, consulta Creazione di meccanismi nel AWS Well-Architected Framework.

account membro

Tutti gli account Account AWS diversi dall'account di gestione che fanno parte di un'organizzazione in. AWS Organizations Un account può essere membro di una sola organizzazione alla volta.

MEH

Vedi sistema di esecuzione della produzione.

Message Queuing Telemetry Transport (MQTT)

Un protocollo di comunicazione machine-to-machine (M2M) leggero, basato sul modello di pubblicazione/sottoscrizione, per dispositivi IoT con risorse limitate.

microservizio

Un servizio piccolo e indipendente che comunica tramite canali ben definiti ed è in genere di proprietà di piccoli team autonomi. APIs Ad esempio, un sistema assicurativo potrebbe includere microservizi che si riferiscono a funzionalità aziendali, come vendite o marketing, o sottodomini, come acquisti, reclami o analisi. I vantaggi dei microservizi includono agilità, dimensionamento flessibile, facilità di implementazione, codice riutilizzabile e resilienza. Per ulteriori informazioni, consulta Integrazione dei microservizi utilizzando servizi serverless. AWS

architettura di microservizi

Un approccio alla creazione di un'applicazione con componenti indipendenti che eseguono ogni processo applicativo come microservizio. Questi microservizi comunicano attraverso un'interfaccia

ben definita utilizzando sistemi leggeri. APIs Ogni microservizio in questa architettura può essere aggiornato, distribuito e dimensionato per soddisfare la richiesta di funzioni specifiche di un'applicazione. Per ulteriori informazioni, vedere Implementazione dei microservizi su. AWS

Programma di accelerazione della migrazione (MAP)

Un AWS programma che fornisce consulenza, supporto, formazione e servizi per aiutare le organizzazioni a costruire una solida base operativa per il passaggio al cloud e per contribuire a compensare il costo iniziale delle migrazioni. MAP include una metodologia di migrazione per eseguire le migrazioni precedenti in modo metodico e un set di strumenti per automatizzare e accelerare gli scenari di migrazione comuni.

migrazione su larga scala

Il processo di trasferimento della maggior parte del portfolio di applicazioni sul cloud avviene a ondate, con più applicazioni trasferite a una velocità maggiore in ogni ondata. Questa fase utilizza le migliori pratiche e le lezioni apprese nelle fasi precedenti per implementare una fabbrica di migrazione di team, strumenti e processi per semplificare la migrazione dei carichi di lavoro attraverso l'automazione e la distribuzione agile. Questa è la terza fase della strategia di migrazione AWS.

fabbrica di migrazione

Team interfunzionali che semplificano la migrazione dei carichi di lavoro attraverso approcci automatizzati e agili. I team di Migration Factory in genere includono addetti alle operazioni, analisti e proprietari aziendali, ingegneri addetti alla migrazione, sviluppatori e DevOps professionisti che lavorano nell'ambito degli sprint. Tra il 20% e il 50% di un portfolio di applicazioni aziendali è costituito da schemi ripetuti che possono essere ottimizzati con un approccio di fabbrica. Per ulteriori informazioni, consulta la discussione sulle fabbriche di migrazione e la Guida alla fabbrica di migrazione al cloud in questo set di contenuti.

metadati di migrazione

Le informazioni sull'applicazione e sul server necessarie per completare la migrazione. Ogni modello di migrazione richiede un set diverso di metadati di migrazione. Esempi di metadati di migrazione includono la sottorete, il gruppo di sicurezza e l'account di destinazione. AWS

modello di migrazione

Un'attività di migrazione ripetibile che descrive in dettaglio la strategia di migrazione, la destinazione della migrazione e l'applicazione o il servizio di migrazione utilizzati. Esempio: riorganizza la migrazione su Amazon EC2 con AWS Application Migration Service.

Valutazione del portfolio di migrazione (MPA)

Uno strumento online che fornisce informazioni per la convalida del business case per la migrazione a. Cloud AWS MPA offre una valutazione dettagliata del portfolio (dimensionamento corretto dei server, prezzi, confronto del TCO, analisi dei costi di migrazione) e pianificazione della migrazione (analisi e raccolta dei dati delle applicazioni, raggruppamento delle applicazioni, prioritizzazione delle migrazioni e pianificazione delle ondate). Lo strumento MPA (richiede l'accesso) è disponibile gratuitamente per tutti i AWS consulenti e i consulenti dei partner APN.

valutazione della preparazione alla migrazione (MRA)

Il processo di acquisizione di informazioni sullo stato di preparazione al cloud di un'organizzazione, l'identificazione dei punti di forza e di debolezza e la creazione di un piano d'azione per colmare le lacune identificate, utilizzando il CAF. AWS Per ulteriori informazioni, consulta la guida di preparazione alla migrazione. MRA è la prima fase della strategia di migrazione AWS.

strategia di migrazione

L'approccio utilizzato per migrare un carico di lavoro verso. Cloud AWS Per ulteriori informazioni, consulta la voce 7 R in questo glossario e consulta Mobilita la tua organizzazione per accelerare le migrazioni su larga scala.

ML

Vedi machine learning.

modernizzazione

Trasformazione di un'applicazione obsoleta (legacy o monolitica) e della relativa infrastruttura in un sistema agile, elastico e altamente disponibile nel cloud per ridurre i costi, aumentare l'efficienza e sfruttare le innovazioni. Per ulteriori informazioni, vedere Strategia per la modernizzazione delle applicazioni in. Cloud AWS

valutazione della preparazione alla modernizzazione

Una valutazione che aiuta a determinare la preparazione alla modernizzazione delle applicazioni di un'organizzazione, identifica vantaggi, rischi e dipendenze e determina in che misura l'organizzazione può supportare lo stato futuro di tali applicazioni. Il risultato della valutazione è uno schema dell'architettura di destinazione, una tabella di marcia che descrive in dettaglio le fasi di sviluppo e le tappe fondamentali del processo di modernizzazione e un piano d'azione per colmare le lacune identificate. Per ulteriori informazioni, vedere Valutazione della preparazione alla modernizzazione per le applicazioni in. Cloud AWS

applicazioni monolitiche (monoliti)

Applicazioni eseguite come un unico servizio con processi strettamente collegati. Le applicazioni monolitiche presentano diversi inconvenienti. Se una funzionalità dell'applicazione registra un picco di domanda, l'intera architettura deve essere dimensionata. L'aggiunta o il miglioramento delle funzionalità di un'applicazione monolitica diventa inoltre più complessa man mano che la base di codice cresce. Per risolvere questi problemi, puoi utilizzare un'architettura di microservizi. Per ulteriori informazioni, consulta la sezione Scomposizione dei monoliti in microservizi.

MAPPA

Vedi Migration Portfolio Assessment.

MQTT

Vedi Message Queuing Telemetry Transport.

classificazione multiclasse

Un processo che aiuta a generare previsioni per più classi (prevedendo uno o più di due risultati). Ad esempio, un modello di machine learning potrebbe chiedere "Questo prodotto è un libro, un'auto o un telefono?" oppure "Quale categoria di prodotti è più interessante per questo cliente?"

infrastruttura mutabile

Un modello che aggiorna e modifica l'infrastruttura esistente per i carichi di lavoro di produzione. Per migliorare la coerenza, l'affidabilità e la prevedibilità, il AWS Well-Architected Framework consiglia l'uso di un'infrastruttura immutabile come best practice.

O

OAC

Vedi Origin Access Control.

QUERCIA

Vedi Origin Access Identity.

OCM

Vedi gestione delle modifiche organizzative.

migrazione offline

Un metodo di migrazione in cui il carico di lavoro di origine viene eliminato durante il processo di migrazione. Questo metodo prevede tempi di inattività prolungati e viene in genere utilizzato per carichi di lavoro piccoli e non critici.

OI

Vedi l'integrazione delle operazioni.

OLA

Vedi accordo a livello operativo.

migrazione online

Un metodo di migrazione in cui il carico di lavoro di origine viene copiato sul sistema di destinazione senza essere messo offline. Le applicazioni connesse al carico di lavoro possono continuare a funzionare durante la migrazione. Questo metodo comporta tempi di inattività pari a zero o comunque minimi e viene in genere utilizzato per carichi di lavoro di produzione critici.

OPC-UA

Vedi Open Process Communications - Unified Architecture.

Comunicazioni a processo aperto - Architettura unificata (OPC-UA)

Un protocollo di comunicazione machine-to-machine (M2M) per l'automazione industriale. OPC-UA fornisce uno standard di interoperabilità con schemi di crittografia, autenticazione e autorizzazione dei dati.

accordo a livello operativo (OLA)

Un accordo che chiarisce quali sono gli impegni reciproci tra i gruppi IT funzionali, a supporto di un accordo sul livello di servizio (SLA).

revisione della prontezza operativa (ORR)

Un elenco di domande e best practice associate che aiutano a comprendere, valutare, prevenire o ridurre la portata degli incidenti e dei possibili guasti. Per ulteriori informazioni, vedere Operational Readiness Reviews (ORR) nel Well-Architected AWS Framework.

tecnologia operativa (OT)

Sistemi hardware e software che interagiscono con l'ambiente fisico per controllare le operazioni, le apparecchiature e le infrastrutture industriali. Nella produzione, l'integrazione di sistemi OT e di tecnologia dell'informazione (IT) è un obiettivo chiave per le trasformazioni dell'Industria 4.0.

integrazione delle operazioni (OI)

Il processo di modernizzazione delle operazioni nel cloud, che prevede la pianificazione, l'automazione e l'integrazione della disponibilità. Per ulteriori informazioni, consulta la guida all'integrazione delle operazioni.

trail organizzativo

Un percorso creato da noi AWS CloudTrail che registra tutti gli eventi di un'organizzazione per tutti Account AWS . AWS Organizations Questo percorso viene creato in ogni Account AWS che fa parte dell'organizzazione e tiene traccia dell'attività in ogni account. Per ulteriori informazioni, consulta Creazione di un percorso per un'organizzazione nella CloudTrail documentazione.

gestione del cambiamento organizzativo (OCM)

Un framework per la gestione di trasformazioni aziendali importanti e che comportano l'interruzione delle attività dal punto di vista delle persone, della cultura e della leadership. OCM aiuta le organizzazioni a prepararsi e passare a nuovi sistemi e strategie accelerando l'adozione del cambiamento, affrontando i problemi di transizione e promuovendo cambiamenti culturali e organizzativi. Nella strategia di AWS migrazione, questo framework si chiama accelerazione delle persone, a causa della velocità di cambiamento richiesta nei progetti di adozione del cloud. Per ulteriori informazioni, consultare la Guida OCM.

controllo dell'accesso all'origine (OAC)

In CloudFront, un'opzione avanzata per limitare l'accesso per proteggere i contenuti di Amazon Simple Storage Service (Amazon S3). OAC supporta tutti i bucket S3 in generale Regioni AWS, la crittografia lato server con AWS KMS (SSE-KMS) e le richieste dinamiche e dirette al bucket S3. `PUT` `DELETE`

identità di accesso origine (OAI)

Nel CloudFront, un'opzione per limitare l'accesso per proteggere i tuoi contenuti Amazon S3. Quando usi OAI, CloudFront crea un principale con cui Amazon S3 può autenticarsi. I principali autenticati possono accedere ai contenuti in un bucket S3 solo tramite una distribuzione specifica. CloudFront Vedi anche OAC, che fornisce un controllo degli accessi più granulare e avanzato.

ORR

Vedi la revisione della prontezza operativa.

- NON

Vedi la tecnologia operativa.

VPC in uscita (egress)

In un'architettura AWS multi-account, un VPC che gestisce le connessioni di rete avviate dall'interno di un'applicazione. La AWS Security Reference Architecture consiglia di configurare l'account di rete con funzionalità in entrata, in uscita e di ispezione VPCs per proteggere l'interfaccia bidirezionale tra l'applicazione e Internet in generale.

P

limite delle autorizzazioni

Una policy di gestione IAM collegata ai principali IAM per impostare le autorizzazioni massime che l'utente o il ruolo possono avere. Per ulteriori informazioni, consulta Limiti delle autorizzazioni nella documentazione di IAM.

informazioni di identificazione personale (PII)

Informazioni che, se visualizzate direttamente o abbinate ad altri dati correlati, possono essere utilizzate per dedurre ragionevolmente l'identità di un individuo. Esempi di informazioni personali includono nomi, indirizzi e informazioni di contatto.

Informazioni che consentono l'identificazione personale degli utenti

Visualizza le informazioni di identificazione personale.

playbook

Una serie di passaggi predefiniti che raccolgono il lavoro associato alle migrazioni, come l'erogazione delle funzioni operative principali nel cloud. Un playbook può assumere la forma di script, runbook automatici o un riepilogo dei processi o dei passaggi necessari per gestire un ambiente modernizzato.

PLC

Vedi controllore logico programmabile.

PLM

Vedi la gestione del ciclo di vita del prodotto.

policy

Un oggetto in grado di definire le autorizzazioni (vedi politica basata sull'identità), specificare le condizioni di accesso (vedi politicabasata sulle risorse) o definire le autorizzazioni massime per tutti gli account di un'organizzazione in (vedi politica di controllo dei servizi). AWS Organizations

persistenza poliglotta

Scelta indipendente della tecnologia di archiviazione di dati di un microservizio in base ai modelli di accesso ai dati e ad altri requisiti. Se i microservizi utilizzano la stessa tecnologia di archiviazione di dati, possono incontrare problemi di implementazione o registrare prestazioni scadenti. I microservizi vengono implementati più facilmente e ottengono prestazioni e scalabilità migliori se utilizzano l'archivio dati più adatto alle loro esigenze. Per ulteriori informazioni, consulta la sezione Abilitazione della persistenza dei dati nei microservizi.

valutazione del portfolio

Un processo di scoperta, analisi e definizione delle priorità del portfolio di applicazioni per pianificare la migrazione. Per ulteriori informazioni, consulta la pagina Valutazione della preparazione alla migrazione.

predicate

Una condizione di interrogazione che restituisce o, in genere, si trova in una clausola `true`. `false` `WHERE`

predicato pushdown

Una tecnica di ottimizzazione delle query del database che filtra i dati della query prima del trasferimento. Ciò riduce la quantità di dati che devono essere recuperati ed elaborati dal database relazionale e migliora le prestazioni delle query.

controllo preventivo

Un controllo di sicurezza progettato per impedire il verificarsi di un evento. Questi controlli sono la prima linea di difesa per impedire accessi non autorizzati o modifiche indesiderate alla rete. Per ulteriori informazioni, consulta Controlli preventivi in Implementazione dei controlli di sicurezza in AWS.

principale

Un'entità in AWS grado di eseguire azioni e accedere alle risorse. Questa entità è in genere un utente root per un Account AWS ruolo IAM o un utente. Per ulteriori informazioni, consulta Principali in Termini e concetti dei ruoli nella documentazione di IAM.

privacy fin dalla progettazione

Un approccio di ingegneria dei sistemi che tiene conto della privacy durante l'intero processo di sviluppo.

zone ospitate private

Un contenitore che contiene informazioni su come desideri che Amazon Route 53 risponda alle query DNS per un dominio e i relativi sottodomini all'interno di uno o più. VPCs Per ulteriori informazioni, consulta Utilizzo delle zone ospitate private nella documentazione di Route 53.

controllo proattivo

Un controllo di sicurezza progettato per impedire l'implementazione di risorse non conformi. Questi controlli analizzano le risorse prima del loro provisioning. Se la risorsa non è conforme al controllo, non viene fornita. Per ulteriori informazioni, consulta la guida di riferimento sui controlli nella AWS Control Tower documentazione e consulta Controlli proattivi in Implementazione dei controlli di sicurezza su. AWS

gestione del ciclo di vita del prodotto (PLM)

La gestione dei dati e dei processi di un prodotto durante l'intero ciclo di vita, dalla progettazione, sviluppo e lancio, attraverso la crescita e la maturità, fino al declino e alla rimozione.

Ambiente di produzione

Vedi ambiente.

controllore logico programmabile (PLC)

Nella produzione, un computer altamente affidabile e adattabile che monitora le macchine e automatizza i processi di produzione.

concatenamento rapido

Utilizzo dell'output di un prompt LLM come input per il prompt successivo per generare risposte migliori. Questa tecnica viene utilizzata per suddividere un'attività complessa in sottoattività o per perfezionare o espandere iterativamente una risposta preliminare. Aiuta a migliorare l'accuratezza e la pertinenza delle risposte di un modello e consente risultati più granulari e personalizzati.

pseudonimizzazione

Il processo di sostituzione degli identificatori personali in un set di dati con valori segnaposto. La pseudonimizzazione può aiutare a proteggere la privacy personale. I dati pseudonimizzati sono ancora considerati dati personali.

publish/subscribe (pub/sub)

Un modello che consente comunicazioni asincrone tra microservizi per migliorare la scalabilità e la reattività. Ad esempio, in un MES basato su microservizi, un microservizio può pubblicare

messaggi di eventi su un canale a cui altri microservizi possono abbonarsi. Il sistema può aggiungere nuovi microservizi senza modificare il servizio di pubblicazione.

Q

Piano di query

Una serie di passaggi, come le istruzioni, utilizzati per accedere ai dati in un sistema di database relazionale SQL.

regressione del piano di query

Quando un ottimizzatore del servizio di database sceglie un piano non ottimale rispetto a prima di una determinata modifica all'ambiente di database. Questo può essere causato da modifiche a statistiche, vincoli, impostazioni dell'ambiente, associazioni dei parametri di query e aggiornamenti al motore di database.

R

Matrice RACI

Vedi responsabile, responsabile, consultato, informato (RACI).

STRACCIO

Vedi Retrieval Augmented Generation.

ransomware

Un software dannoso progettato per bloccare l'accesso a un sistema informatico o ai dati fino a quando non viene effettuato un pagamento.

Matrice RASCI

Vedi responsabile, responsabile, consultato, informato (RACI).

RCAC

Vedi controllo dell'accesso a righe e colonne.

replica di lettura

Una copia di un database utilizzata per scopi di sola lettura. È possibile indirizzare le query alla replica di lettura per ridurre il carico sul database principale.

riprogettare

Vedi 7 Rs.

obiettivo del punto di ripristino (RPO)

Il periodo di tempo massimo accettabile dall'ultimo punto di ripristino dei dati. Questo determina ciò che si considera una perdita di dati accettabile tra l'ultimo punto di ripristino e l'interruzione del servizio.

obiettivo del tempo di ripristino (RTO)

Il ritardo massimo accettabile tra l'interruzione del servizio e il ripristino del servizio.

rifattorizzare

Vedi 7 R.

Region

Una raccolta di AWS risorse in un'area geografica. Ciascuna Regione AWS è isolata e indipendente dalle altre per fornire tolleranza agli errori, stabilità e resilienza. Per ulteriori informazioni, consulta Specificare cosa può usare Regioni AWS il tuo account.

regressione

Una tecnica di ML che prevede un valore numerico. Ad esempio, per risolvere il problema "A che prezzo verrà venduta questa casa?" un modello di ML potrebbe utilizzare un modello di regressione lineare per prevedere il prezzo di vendita di una casa sulla base di dati noti sulla casa (ad esempio, la metratura).

riospitare

Vedi 7 R.

rilascio

In un processo di implementazione, l'atto di promuovere modifiche a un ambiente di produzione.

trasferisco

Vedi 7 Rs.

ripiattaforma

Vedi 7 Rs.

riacquisto

Vedi 7 Rs.

resilienza

La capacità di un'applicazione di resistere o ripristinare le interruzioni. L'elevata disponibilità e il disaster recovery sono considerazioni comuni quando si pianifica la resilienza in. Cloud AWSPer ulteriori informazioni, vedere Cloud AWS Resilience.

policy basata su risorse

Una policy associata a una risorsa, ad esempio un bucket Amazon S3, un endpoint o una chiave di crittografia. Questo tipo di policy specifica a quali principali è consentito l'accesso, le azioni supportate e qualsiasi altra condizione che deve essere soddisfatta.

matrice di assegnazione di responsabilità (RACI)

Una matrice che definisce i ruoli e le responsabilità di tutte le parti coinvolte nelle attività di migrazione e nelle operazioni cloud. Il nome della matrice deriva dai tipi di responsabilità definiti nella matrice: responsabile (R), responsabile (A), consultato (C) e informato (I). Il tipo di supporto (S) è facoltativo. Se includi il supporto, la matrice viene chiamata matrice RASCI e, se la escludi, viene chiamata matrice RACI.

controllo reattivo

Un controllo di sicurezza progettato per favorire la correzione di eventi avversi o deviazioni dalla baseline di sicurezza. Per ulteriori informazioni, consulta Controlli reattivi in Implementazione dei controlli di sicurezza in AWS.

retain

Vedi 7 R.

andare in pensione

Vedi 7 Rs.

Retrieval Augmented Generation (RAG)

Una tecnologia di intelligenza artificiale generativa in cui un LLM fa riferimento a una fonte di dati autorevole esterna alle sue fonti di dati di formazione prima di generare una risposta. Ad esempio, un modello RAG potrebbe eseguire una ricerca semantica nella knowledge base o nei dati personalizzati di un'organizzazione. Per ulteriori informazioni, consulta Cos'è il RAG.

rotazione

Processo di aggiornamento periodico di un segreto per rendere più difficile l'accesso alle credenziali da parte di un utente malintenzionato.

controllo dell'accesso a righe e colonne (RCAC)

L'uso di espressioni SQL di base e flessibili con regole di accesso definite. RCAC è costituito da autorizzazioni di riga e maschere di colonna.

RPO

Vedi obiettivo del punto di ripristino.

VERSO

Vedi obiettivo del tempo di ripristino.

runbook

Un insieme di procedure manuali o automatizzate necessarie per eseguire un'attività specifica. In genere sono progettati per semplificare operazioni o procedure ripetitive con tassi di errore elevati.

S

SAML 2.0

Uno standard aperto utilizzato da molti provider di identità (IdPs). Questa funzionalità abilita il single sign-on (SSO) federato, in modo che gli utenti possano accedere Console di gestione AWS o chiamare le operazioni AWS API senza che tu debba creare un utente in IAM per tutti i membri dell'organizzazione. Per ulteriori informazioni sulla federazione basata su SAML 2.0, consulta Informazioni sulla federazione basata su SAML 2.0 nella documentazione di IAM.

SCADA

Vedi controllo di supervisione e acquisizione dati.

SCP

Vedi la politica di controllo del servizio.

Secret

In Gestione dei segreti AWS, informazioni riservate o riservate, come una password o le credenziali utente, archiviate in forma crittografata. È costituito dal valore segreto e dai relativi

metadati. Il valore segreto può essere binario, una stringa singola o più stringhe. Per ulteriori informazioni, consulta Cosa c'è in un segreto di Secrets Manager? nella documentazione di Secrets Manager.

sicurezza fin dalla progettazione

Un approccio di ingegneria dei sistemi che tiene conto della sicurezza durante l'intero processo di sviluppo.

controllo di sicurezza

Un guardrail tecnico o amministrativo che impedisce, rileva o riduce la capacità di un autore di minacce di sfruttare una vulnerabilità di sicurezza. Esistono quattro tipi principali di controlli di sicurezza: preventivi, investigativi, reattivi e proattivi.

rafforzamento della sicurezza

Il processo di riduzione della superficie di attacco per renderla più resistente agli attacchi. Può includere azioni come la rimozione di risorse che non sono più necessarie, l'implementazione di best practice di sicurezza che prevedono la concessione del privilegio minimo o la disattivazione di funzionalità non necessarie nei file di configurazione.

sistema di gestione delle informazioni e degli eventi di sicurezza (SIEM)

Strumenti e servizi che combinano sistemi di gestione delle informazioni di sicurezza (SIM) e sistemi di gestione degli eventi di sicurezza (SEM). Un sistema SIEM raccoglie, monitora e analizza i dati da server, reti, dispositivi e altre fonti per rilevare minacce e violazioni della sicurezza e generare avvisi.

automazione della risposta alla sicurezza

Un'azione predefinita e programmata progettata per rispondere o porre rimedio automaticamente a un evento di sicurezza. Queste automazioni fungono da controlli di sicurezza investigativi o reattivi che aiutano a implementare le migliori pratiche di sicurezza. AWS Esempi di azioni di risposta automatizzate includono la modifica di un gruppo di sicurezza VPC, l'applicazione di patch a un'istanza EC2 Amazon o la rotazione delle credenziali.

Crittografia lato server

Crittografia dei dati a destinazione, da parte di chi li riceve. Servizio AWS

Policy di controllo dei servizi (SCP)

Una politica che fornisce il controllo centralizzato sulle autorizzazioni per tutti gli account di un'organizzazione in. AWS Organizations SCPs definire barriere o fissare limiti alle azioni

che un amministratore può delegare a utenti o ruoli. È possibile utilizzarli SCPs come elenchi consentiti o elenchi di rifiuto, per specificare quali servizi o azioni sono consentiti o proibiti. Per ulteriori informazioni, consulta le politiche di controllo del servizio nella AWS Organizations documentazione.

endpoint del servizio

L'URL del punto di ingresso per un Servizio AWS. Puoi utilizzare l'endpoint per connetterti a livello di programmazione al servizio di destinazione. Per ulteriori informazioni, consulta Endpoint del Servizio AWS nei Riferimenti generali di AWS.

accordo sul livello di servizio (SLA)

Un accordo che chiarisce ciò che un team IT promette di offrire ai propri clienti, ad esempio l'operatività e le prestazioni del servizio.

indicatore del livello di servizio (SLI)

Misurazione di un aspetto prestazionale di un servizio, ad esempio il tasso di errore, la disponibilità o la velocità effettiva.

obiettivo a livello di servizio (SLO)

Una metrica target che rappresenta lo stato di un servizio, misurato da un indicatore del livello di servizio.

Modello di responsabilità condivisa

Un modello che descrive la responsabilità condivisa AWS per la sicurezza e la conformità del cloud. AWS è responsabile della sicurezza del cloud, mentre tu sei responsabile della sicurezza nel cloud. Per ulteriori informazioni, consulta Modello di responsabilità condivisa.

SIEM

Vedi il sistema di gestione delle informazioni e degli eventi sulla sicurezza.

punto di errore singolo (SPOF)

Un guasto in un singolo componente critico di un'applicazione che può disturbare il sistema.

SLAM

Vedi il contratto sul livello di servizio.

SLI

Vedi l'indicatore del livello di servizio.

LENTA

Vedi obiettivo del livello di servizio.

split-and-seed modello

Un modello per dimensionare e accelerare i progetti di modernizzazione. Man mano che vengono definite nuove funzionalità e versioni dei prodotti, il team principale si divide per creare nuovi team di prodotto. Questo aiuta a dimensionare le capacità e i servizi dell'organizzazione, migliora la produttività degli sviluppatori e supporta una rapida innovazione. Per ulteriori informazioni, vedere Approccio graduale alla modernizzazione delle applicazioni in. Cloud AWS

SPOF

Vedi punto di errore singolo.

schema a stella

Una struttura organizzativa di database che utilizza un'unica tabella dei fatti di grandi dimensioni per archiviare i dati transazionali o misurati e utilizza una o più tabelle dimensionali più piccole per memorizzare gli attributi dei dati. Questa struttura è progettata per l'uso in un data warehouse o per scopi di business intelligence.

modello del fico strangolatore

Un approccio alla modernizzazione dei sistemi monolitici mediante la riscrittura e la sostituzione incrementali delle funzionalità del sistema fino alla disattivazione del sistema legacy. Questo modello utilizza l'analogia di una pianta di fico che cresce fino a diventare un albero robusto e alla fine annienta e sostituisce il suo ospite. Il modello è stato introdotto da Martin Fowler come metodo per gestire il rischio durante la riscrittura di sistemi monolitici. Per un esempio di come applicare questo modello, consulta Modernizzazione incrementale dei servizi Web legacy di Microsoft ASP.NET (ASMX) mediante container e Gateway Amazon API.

sottorete

Un intervallo di indirizzi IP nel VPC. Una sottorete deve risiedere in una singola zona di disponibilità.

controllo di supervisione e acquisizione dati (SCADA)

Nella produzione, un sistema che utilizza hardware e software per monitorare gli asset fisici e le operazioni di produzione.

crittografia simmetrica

Un algoritmo di crittografia che utilizza la stessa chiave per crittografare e decrittografare i dati.

test sintetici

Test di un sistema in modo da simulare le interazioni degli utenti per rilevare potenziali problemi o monitorare le prestazioni. Puoi usare Amazon CloudWatch Synthetics per creare questi test.

prompt di sistema

Una tecnica per fornire contesto, istruzioni o linee guida a un LLM per indirizzarne il comportamento. I prompt di sistema aiutano a impostare il contesto e stabilire regole per le interazioni con gli utenti.

T

tags

Coppie chiave-valore che fungono da metadati per l'organizzazione delle risorse. AWS Con i tag è possibile a gestire, identificare, organizzare, cercare e filtrare le risorse. Per ulteriori informazioni, consulta Tagging delle risorse AWS.

variabile di destinazione

Il valore che stai cercando di prevedere nel machine learning supervisionato. Questo è indicato anche come variabile di risultato. Ad esempio, in un ambiente di produzione la variabile di destinazione potrebbe essere un difetto del prodotto.

elenco di attività

Uno strumento che viene utilizzato per tenere traccia dei progressi tramite un runbook. Un elenco di attività contiene una panoramica del runbook e un elenco di attività generali da completare. Per ogni attività generale, include la quantità stimata di tempo richiesta, il proprietario e lo stato di avanzamento.

Ambiente di test

Vedi ambiente.

training

Fornire dati da cui trarre ispirazione dal modello di machine learning. I dati di training devono contenere la risposta corretta. L'algoritmo di apprendimento trova nei dati di addestramento i pattern che mappano gli attributi dei dati di input al target (la risposta che si desidera prevedere). Produce un modello di ML che acquisisce questi modelli. Puoi quindi utilizzare il modello di ML per creare previsioni su nuovi dati di cui non si conosce il target.

Transit Gateway

Un hub di transito di rete che puoi utilizzare per interconnettere le tue reti VPCs e quelle locali. Per ulteriori informazioni, consulta Cos'è un gateway di transito nella AWS Transit Gateway documentazione.

flusso di lavoro basato su trunk

Un approccio in cui gli sviluppatori creano e testano le funzionalità localmente in un ramo di funzionalità e quindi uniscono tali modifiche al ramo principale. Il ramo principale viene quindi integrato negli ambienti di sviluppo, preproduzione e produzione, in sequenza.

Accesso attendibile

Concessione delle autorizzazioni a un servizio specificato dall'utente per eseguire attività all'interno dell'organizzazione AWS Organizations e nei suoi account per conto dell'utente. Il servizio attendibile crea un ruolo collegato al servizio in ogni account, quando tale ruolo è necessario, per eseguire attività di gestione per conto dell'utente. Per ulteriori informazioni, consulta Utilizzo AWS Organizations con altri AWS servizi nella AWS Organizations documentazione.

regolazione

Modificare alcuni aspetti del processo di training per migliorare la precisione del modello di ML. Ad esempio, puoi addestrare il modello di ML generando un set di etichette, aggiungendo etichette e quindi ripetendo questi passaggi più volte con impostazioni diverse per ottimizzare il modello.

team da due pizze

Una piccola DevOps squadra che puoi sfamare con due pizze. Un team composto da due persone garantisce la migliore opportunità possibile di collaborazione nello sviluppo del software.

U

incertezza

Un concetto che si riferisce a informazioni imprecise, incomplete o sconosciute che possono minare l'affidabilità dei modelli di machine learning predittivi. Esistono due tipi di incertezza: l'incertezza epistemica, che è causata da dati limitati e incompleti, mentre l'incertezza aleatoria è causata dal rumore e dalla casualità insiti nei dati. Per ulteriori informazioni, consulta la guida Quantificazione dell'incertezza nei sistemi di deep learning.

compiti indifferenziati

Conosciuto anche come sollevamento di carichi pesanti, è un lavoro necessario per creare e far funzionare un'applicazione, ma che non apporta valore diretto all'utente finale né offre vantaggi competitivi. Esempi di attività indifferenziate includono l'approvvigionamento, la manutenzione e la pianificazione della capacità.

ambienti superiori

Vedi ambiente.

V

vacuum

Un'operazione di manutenzione del database che prevede la pulizia dopo aggiornamenti incrementali per recuperare lo spazio di archiviazione e migliorare le prestazioni.

controllo delle versioni

Processi e strumenti che tengono traccia delle modifiche, ad esempio le modifiche al codice di origine in un repository.

Peering VPC

Una connessione tra due VPCs che consente di indirizzare il traffico utilizzando indirizzi IP privati. Per ulteriori informazioni, consulta Che cos'è il peering VPC? nella documentazione di Amazon VPC.

vulnerabilità

Un difetto software o hardware che compromette la sicurezza del sistema.

W

cache calda

Una cache del buffer che contiene dati correnti e pertinenti a cui si accede frequentemente. L'istanza di database può leggere dalla cache del buffer, il che richiede meno tempo rispetto alla lettura dalla memoria dal disco principale.

dati caldi

Dati a cui si accede raramente. Quando si eseguono interrogazioni di questo tipo di dati, in genere sono accettabili query moderatamente lente.

funzione finestra

Una funzione SQL che esegue un calcolo su un gruppo di righe che si riferiscono in qualche modo al record corrente. Le funzioni della finestra sono utili per l'elaborazione di attività, come il calcolo di una media mobile o l'accesso al valore delle righe in base alla posizione relativa della riga corrente.

Carico di lavoro

Una raccolta di risorse e codice che fornisce valore aziendale, ad esempio un'applicazione rivolta ai clienti o un processo back-end.

flusso di lavoro

Gruppi funzionali in un progetto di migrazione responsabili di una serie specifica di attività. Ogni flusso di lavoro è indipendente ma supporta gli altri flussi di lavoro del progetto. Ad esempio, il flusso di lavoro del portfolio è responsabile della definizione delle priorità delle applicazioni, della pianificazione delle ondate e della raccolta dei metadati di migrazione. Il flusso di lavoro del portfolio fornisce queste risorse al flusso di lavoro di migrazione, che quindi migra i server e le applicazioni.

VERME

Vedi scrivere una volta, leggere molti.

WQF

Vedi AWS Workload Qualification Framework.

scrivi una volta, leggi molte (WORM)

Un modello di storage che scrive i dati una sola volta e ne impedisce l'eliminazione o la modifica. Gli utenti autorizzati possono leggere i dati tutte le volte che è necessario, ma non possono modificarli. Questa infrastruttura di archiviazione dei dati è considerata immutabile.

Z

exploit zero-day

Un attacco, in genere malware, che sfrutta una vulnerabilità zero-day.

vulnerabilità zero-day

Un difetto o una vulnerabilità assoluta in un sistema di produzione. Gli autori delle minacce possono utilizzare questo tipo di vulnerabilità per attaccare il sistema. Gli sviluppatori vengono spesso a conoscenza della vulnerabilità causata dall'attacco.

prompt zero-shot

Fornire a un LLM le istruzioni per eseguire un'attività ma non esempi (immagini) che possano aiutarla. Il LLM deve utilizzare le sue conoscenze pre-addestrate per gestire l'attività. L'efficacia del prompt zero-shot dipende dalla complessità dell'attività e dalla qualità del prompt. Vedi anche few-shot prompting.

applicazione zombie

Un'applicazione che prevede un utilizzo CPU e memoria inferiore al 5%. In un progetto di migrazione, è normale ritirare queste applicazioni.

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.