WHITE PAPER | APRILE 2016

Guida dettagliata al testing ETL completamente automatizzato

ca technologies

Sezione 1

Il ruolo critico di ETL per l'azienda moderna

Fin dal suo ingresso nel mondo del data warehousing e della business intelligence, Extract, Transform, Load (ETL) è diventato un processo onnipresente nel contesto software. Come suggerisce il nome, una routine ETL si compone di tre fasi distinte, che spesso si svolgono in parallelo: i dati vengono estratti da una o più fonti di dati; vengono convertiti nello stato richiesto; vengono caricati nella destinazione desiderata, di solito un data warehouse, data mart o database. Una outine ETL sviluppata include spesso anche gestione degli errori, infrastruttura di logging e ambiente di routine.[1]

Fino a oggi, ETL è stato utilizzato soprattutto per preparare i dati, spesso ampi e disparati, per l'analisi e la business intelligence. Tuttavia, il suo impiego si sta espandendo al di là del semplice trasferimento dei dati: la migrazione verso nuovi sistemi è un'applicazione sempre più comune, così come la gestione di integrazioni, ordinamenti e unioni di dati.[2]

ETL è pertanto una caratteristica critica del ciclo di sviluppo moderno, in rapida evoluzione, dove in un dato momento più release e versioni vengono realizzate in parallelo. Le aziende devono essere in grado di migliorare costantemente, integrare e innovare il proprio software, mettendo i dati a disposizione di tester e sviluppatori nella condizione giusta per ogni iterazione e release. I dati saranno estratti dalle stesse fonti, ma devono essere trasformati per soddisfare le esigenze specifiche di ogni singolo team. Questo è particolarmente vero se un'azienda si sta impegnando ad essere "agile", o di implementare con successo la continuous delivery.

Un buon esempio del ruolo fondamentale di ETL nell'ambito della continuous delivery viene da una grande banca multinazionale, con cui CA Technologies ha collaborato. La banca stava per procedere a un'acquisizione, e doveva eseguire la migrazione di clienti, prodotti e dati finanziari delle entità acquisite nella propria infrastruttura esistente. Questo significava dover recuperare, convertire e convalidare 80 file di inserimento, per poi caricarli nei sistemi di back-end delle banche. Questi dati dovevano quindi essere disponibili per 47 progetti in parallelo, mantenendone l'integrità referenziale. In questo caso, ETL è stato essenziale per consentire il parallelismo necessario per il successo della continuous delivery.

Tuttavia, nonostante l'uso e l'importanza sempre maggiori di ETL, anche i test relativi riflettono lo stato del testing in generale: sono troppo lenti e manuali, eppure consentono a una quantità inaccettabile di difetti di arrivare in produzione. Questo documento illustra le sfide tipicamente affrontate quando ci si avvicina all'approccio ai test ETL. Viene quindi illustrato un approccio alternativo, ampiamente basato su modelli, considerando come potrebbe rendere i test ETL molto più efficienti, efficaci e sistematici.

Sezione 2

L'approccio tipico ai test ETL e le sfide comuni affrontate

Nella convalida delle regole di trasformazione ETL, i tester creano di solito un insieme di codice ombra, lo utilizzano per trasformare i dati e quindi per confrontare i risultati effettivi con quelli attesi. Di solito, gli script ETL o il codice SQL viene copiato manualmente nei dati di origine ed eseguito, registrando i risultati. Lo stesso script viene poi copiato nei dati di destinazione, sempre registrando i risultati. I due insiemi di risultati (effettivi e previsti) vengono poi confrontati, per verificare che la trasformazione dei dati sia avvenuta in modo corretto.

Il problema principale: complessità e testabilità

La questione di fondo alla base di questa convalida manuale è che le routine ETL, per loro stessa natura, diventano rapidamente molto complesse. Al crescere del business, e della gamma e quantità di dati raccolti, anche le regole ETL devono crescere per poterli gestire. Nella cosiddetta "era dell'informazione", questa crescita sta avvenendo più velocemente di quanto i metodi di test tradizionali siano in grado di gestire. In realtà, la quantità di informazioni raccolte dalle aziende basate sui dati è cresciuta così rapidamente che il 90% dei dati al mondo è stato raccolto solo negli ultimi due anni[3], mentre la quantità di dati raccolti dall'azienda media raddoppia ogni anno.[4]

La complessità dei sistemi progettati per raccogliere, trasferire, manipolare e presentare questi dati cresce esponenzialmente a ogni ulteriore decisione. Questo include le regole ETL, e i fattori che possono influenzare la complessità delle trasformazioni sono numerosi:

- il numero e la gamma delle fonti di dati coinvolte, compresi tipi di database relazionali e non relazionali e file flat;
- il numero e la gamma delle destinazioni di dati;
- il numero e la gamma delle trasformazioni semplici e complesse, dalle semplici ricerche alle unioni attive alla normalizzazione;
- il numero di trasformazioni riutilizzabili, frammenti di codice e unioni;
- il numero di tabelle create.[5]

Tutti questi fattori sono esacerbati dall'attenzione attualmente concentrata sulle soluzioni quasi in tempo reale, e dalle complessità aggiuntive che comportano.[6]

La documentazione non aiuta

Questa complessità crescente influenza direttamente la testabilità delle routine ETL. La situazione risulta particolarmente problematica per i test ETL, dato che le regole di trasformazione sono in genere memorizzate all'interno di documentazione scadente, nella quale i risultati previsti non sono esplicitati. Le regole sono spesso progettate durante la stessa fase di sviluppo, e altrettanto spesso memorizzate in documenti o fogli di lavoro scritti; o, peggio, esistono solo nella mente di sviluppatori e tester.[7] In questo caso, non esiste una documentazione effettiva da cui derivare in modo affidabile i casi di test (cioè il codice ombra).

All'interno di un team di business intelligence con cui abbiamo lavorato, i requisiti erano archiviati come documenti scritti, mentre i casi di test venivano conservati in fogli di calcolo. Questa documentazione statica costituiva un "muro di parole", da cui i passaggi logici delle routine ETL dovevano essere in qualche modo decifrati. I documenti erano focalizzati sulla metodologia "happy path" e non contenevano condizioni negative, quindi circa l'80% della logica possibile che deve essere testata nel sistema medio era assente. Questa documentazione incompleta e ambigua faceva sì che i tester non avessero modo di comprendere con facilità o precisione le routine ETL.

Troppo spesso, erano costretti a colmare i vuoti; ma quando commettevano un errore, nelle routine ETL comparivano difetti. Dati non validi venivano quindi copiati nella destinazione, anche se il codice e i casi di test riflettevano un'interpretazione plausibile della documentazione relativa ai requisiti.

"Garbage in, garbage out": derivazione manuale dei casi di test e risultati attesi

In realtà, un significativo 56% dei difetti che arrivano in produzione può essere fatto risalire ad ambiguità nella documentazione relativa ai requisiti.[8] Questo è in parte dovuto al fatto che i casi di test e i risultati attesi sono derivati manualmente da documentazione di scarsa qualità: un processo che richiede un tempo lunghissimo, e che di solito porta a una copertura di test scadente.

Qualità

La derivazione manuale è ad hoc e non sistematica, e porta di solito alla creazione dei casi di test che vengono in mente ai tester. Data la complessità delle routine ETL, cui abbiamo accennato, combinata con la scarsa qualità della documentazione disponibile, è impossibile aspettarsi che anche il tester di maggior talento riesca a creare tutti i singoli test necessari per convalidare le possibili combinazioni di dati. Ad esempio, se un semplice sistema con 32 nodi e 62 margini è stato progettato in modo lineare, possono esistere 1.073.741.824 possibili percorsi all'interno della sua logica; sicuramente più di quelli concepibili da una singola persona.

La derivazione ad hoc porta quindi a undertesting e overtesting significativi, situazioni in cui viene testata soltanto una frazione della logica possibile applicata in una routine ETL. Il testing negativo sarà particolarmente impegnativo e i casi di test, come la documentazione, di solito si concentrano quasi esclusivamente sugli happy path. Tuttavia, sono questi valori anomali e risultati inattesi che devono essere testati, così come è imperativo che le routine ETL respingano questi dati non validi.

Una società di servizi finanziari con cui CA Technologies ha lavorato, ad esempio, si affidava a 11 casi di test con una copertura di appena il 16%. Si tratta di un numero abbastanza standard, e dalle nostre verifiche è emerso che una copertura di test funzionale del 10-20% rappresenta la norma. Un altro progetto aziendale era in overtesting di 18 volte, con casi di test che si accumulavano nel tentativo di testare a fondo il sistema, senza tuttavia raggiungere mai la massima copertura. L'esecuzione dei 150 casi di test in eccesso, presso un fornitore in outsourcing costava 26.000 dollari.[9]

Il risultato di questa scarsa copertura è l'introduzione di difetti nel codice, dove la correzione risulta lunga e costosa: secondo alcuni studi, può costare 40-1000 volte di più in termini di risorse[10] e 50 volte di più in termini di tempo[11] correggere un bug durante i test, rispetto a una fase precedente. Peggio ancora, i bug potrebbero passare inosservati, e dati non validi venire copiati nella destinazione live, dove possono minacciare l'integrità del sistema. Non solo: di fronte a una documentazione statica, i tester non dispongono di modalità affidabili per misurare la copertura dei propri casi di test; semplicemente non possono affermare quanto, o quanto poco, di una data routine sia oggetto di test, né attribuire priorità ai test in base alla criticità.

Tempo e sforzi: i test non riescono semplicemente a tenere il passo

Scrivere i casi di test a partire dalla documentazione descritta richiede anche enormi risorse, in termini di tempo e di sforzi. Nell'esempio precedente, creare gli 11 casi di test richiedeva 6 ore, mentre l'overtesting dilagante in azienda consumava ancora più tempo. A questo spreco di tempo per la progettazione manuale dei casi di test si aggiunge il tempo che deve poi essere dedicato a confrontare i risultati effettivi e quelli attesi.

Confrontare gli ampi campi singoli con i risultati attesi richiede molto tempo, data la quantità di dati prodotti da una routine ETL complessa, e il fatto che i dati di origine saranno spesso memorizzati in una gamma di tipi diversi di database e di file. È anche un'operazione molto difficile, in quanto i dati trasformati devono essere convalidati su più livelli:

- I tester devono verificare la completezza dei dati, assicurandosi che i conteggi della fonte di dati e della destinazione corrispondano;
- L'integrità dei dati deve essere garantita, controllando che i dati di destinazione siano allineati con quelli di origine;
- La trasformazione deve corrispondere alle regole di business;
- La coerenza dei dati deve essere garantita, identificando eventuali duplicati imprevisti;
- L'integrità referenziale deve essere mantenuta, individuando tutti i record orfani o le chiavi esterne mancanti.[12]

A volte si scende a compromessi, limitandosi a convalidare solo un set di dati campione. Tuttavia, questo compromette anche la completezza dei test ETL, e quindi l'affidabilità delle trasformazioni. Dato il ruolo di molte routine ETL nelle operations business-critical, questo compromesso è inaccettabile. La qualità è ulteriormente influenzata dalla propensione agli errori dei confronti manuali, soprattutto quando i risultati attesi sono definiti male o, peggio, non sono definiti affatto indipendentemente dal codice ombra utilizzato nei test. In questo caso, i tester tendono a presumere che un test sia riuscito, a meno che il risultato effettivo sia particolarmente anomalo: senza risultati attesi predefiniti, tenderanno a dare per scontato che il risultato effettivo sia quello atteso[13], e quindi non saranno in grado di determinare la validità dei dati in modo affidabile.

Il problema dei dati

Finora, ci siamo concentrati sui problemi derivanti dalla derivazione dei test (codice ombra) necessari per convalidare le regole ETL. Tuttavia, una volta derivati i casi di test, i tester hanno bisogno di dati sorgente fittizi da inserire nel sistema. Questa è un'altra causa frequente di colli di bottiglia e di difetti.

Sono disponibili tutti i dati necessari per testare routine ETL complesse?

Disporre di una quantità sufficiente di dati "non validi" è un fattore critico per l'efficacia dei test ETL, in quanto è fondamentale che, quando utilizzata, una regola di ETL rifiuti questi dati e li invii all'utente appropriato, nel formato appropriato. Se non vengono respinti, questi dati non validi possono creare difetti, o addirittura provocare il collasso del sistema.

I "dati non validi" in questo contesto possono essere definiti in vari modi, che corrispondono al modo in cui i dati devono essere convalidati dai tester. Può trattarsi dei dati che, in base alle regole di business non dovrebbero mai essere accettati; ad esempio, valori negativi in un carrello degli acquisti online, in assenza di buoni. Potrebbero essere i dati che minacciano l'integrità referenziale di un magazzino, come dati mancanti interdipendenti o obbligatori, o dati mancanti tra quelli in ingresso.[14] I dati di test che vengono inseriti attraverso una regola di convalida ETL devono pertanto includere l'intera gamma di dati non validi, al fine di consentire una copertura di test funzionale del 100%.

Questi dati raramente sono inclusi nelle fonti di dati di produzione di cui viene eseguito il provisioning ai team di test in molte aziende. Questo perché i dati di produzione sono derivati da scenari "di ordinaria amministrazione" già verificatisi in passato, e di conseguenza per loro natura finiscono per escludere i dati non validi. Non contengono risultati inaspettati, anomali o condizioni estreme necessarie per i test ETL, ma si focalizzano invece sugli "happy path". In effetti, da nostri controlli sui dati di produzione è emersa una copertura che di norma è di appena il 10-20%. Ironicamente, meglio viene creata una routine, meno i "dati non validi" saranno consentiti, il che significa che ci saranno meno dati di diversità sufficiente a testare in modo completo le regole ETL per il futuro.

I dati sono disponibili quando serve?

Un'altra questione importante per la convalida ETL è la disponibilità dei dati. I dati di origine possono essere derivati da 50 fonti diverse in tutta l'azienda. Il problema dei test ETL, e dei test in generale, è che sono concepiti come una serie di fasi lineari, quindi i team di test sono costretti ad attendere i dati mentre vengono utilizzati da un altro team.

Prendiamo l'esempio di un flusso di migrazione nel settore bancario, dove i dati vengono presi da una banca e convertiti nei sistemi di un'altra, utilizzando uno strumento di riconciliazione. In ogni fase, i dati devono essere convalidati, per verificare che siano stati convertiti correttamente nel framework finanziario, che il numero di conto sia stato recuperato, che sia garantita la correttezza ora su ora e così via. Questo processo può essere suddiviso in più fasi separate, dall'input di base, alla de-duplica e alla preparazione, alla propagazione e alla registrazione dei dati. Inoltre, possono essere coinvolti più team, inclusi team ETL e non ETL, in particolare quelli attivi sul mainframe.

Se i dati da tutte le origini in tutta l'azienda non sono disponibili per tutti i team in parallelo, i ritardi si accumuleranno mentre i team rimangono inattivi e in attesa. I tester scriveranno il proprio codice fantasma, e quindi non disporranno dei dati di origine per convalidare una regola ETL, in uso da parte di un altro team. In effetti, abbiamo riscontrato che il tester medio può trascorrere anche il 50% del suo tempo aspettando, manipolando o creando dati. Questo può costituire ben il 20% dell'SDLC globale.

Cosa avviene quando le regole cambiano?

Derivare manualmente casi di test e dati da requisiti statici è un'attività molto poco reattiva al cambiamento. Le routine ETL cambiano rapidamente all'evolversi del business, e il volume e la gamma dei dati raccolti cresce con quest'ultimo. I test ETL non riescono però a tenere il passo con questo cambiamento continuo.

Probabilmente la singola causa principale di ritardi nei progetti, in questo caso, è l'esigenza di controllare e aggiornare i casi di test esistenti quando le routine si modificano. I tester non hanno modo di identificare automaticamente l'impatto di una modifica apportata a requisiti statici e casi di test. Devono invece controllare ogni caso di test esistente a mano, senza alcun modo di misurare se la copertura è effettivamente mantenuta o meno.

Per il team di business intelligence citato sopra, che conservava requisiti e casi di test rispettivamente in documenti scritti e fogli di calcolo, il cambiamento risultava particolarmente problematico. Un tester impiegava ben 7,5 ore per controllare e aggiornare una serie di casi di test in caso di modifica apportata a una singola regola ETL. In un'altra azienda con cui abbiamo lavorato, due tester hanno impiegato due giorni per controllare ogni caso di test esistente dopo una modifica ai requisiti.

Sezione 3

L'alternativa possibile: il testing ETL completamente automatizzato

È evidente, quindi, che finché i test vengono derivati manualmente, e i risultati confrontati manualmente, i test ETL non potranno tenere il passo con la velocità con cui i requisiti di business si modificano costantemente. Qui di seguito viene illustrata una possibile strategia per migliorare l'efficienza e l'efficacia dei test ETL. Si tratta di una strategia basata su requisiti e su modelli, che mira a "spostare a sinistra" lo sforzo di test, e a integrare la qualità nel ciclo di vita ETL fin dall'inizio. Questo approccio basato su modelli introduce l'automazione in tutte le fasi di test e sviluppo, rendendo i test ETL totalmente reattivi al cambiamento continuo.

1) Partire da un modello formale

L'introduzione della modellazione formale nei test ETL offre il vantaggio fondamentale di "spostare a sinistra" lo sforzo di test; tutte le attività di test/sviluppo successive possono quindi essere derivate dallo sforzo iniziale di associazione tra una regola ETL e un modello. Il modello formale diventa quindi la chiave di volta di una convalida ETL completamente automatizzata.

La modellazione formale, però, contribuisce anche a risolvere la questione più specifica citata sopra, collegata all'ambiguità e all'incompletezza dei requisiti. Aiuta a conservare la testabilità nonostante la complessità sempre crescente delle regole ETL, affinché i tester possano comprendere rapidamente e visivamente la logica esatta che deve essere oggetto di test. I tester possono individuare facilmente i dati validi e non che devono essere immessi per testare a fondo una regola di trasformazione, e quale dovrebbe essere il risultato atteso in ciascun caso.

Un modello di diagramma di flusso, ad esempio, suddivide l'altrimenti ingombrante "muro di parole" della documentazione in segmenti gestibili. Esso riduce l'ETL alla sua logica di causa ed effetto, associandolo a una serie di dichiarazioni "se A, allora B", collegate all'interno di una gerarchia di processi.[15] Ognuno di questi passaggi diventa in effetti una componente di test, comunicando al tester cosa esattamente deve essere convalidato. La modellazione delle routine ETL come diagramma di flusso elimina così l'ambiguità dalla documentazione dei requisiti, lavorando per evitare il 56% dei difetti che da essa derivano.

All'aumentare della complessità delle routine ETL, il diagramma di flusso funge da punto di riferimento unico. Diversamente da ciò che avviene con i documenti scritti e i diagrammi "statici", eventuale logica aggiuntiva può essere facilmente aggiunta al modello. Non solo: l'astrazione di routine altamente complesse è possibile utilizzando la tecnologia dei flussi secondari, aumentando così la testabilità. I componenti di livello inferiore possono essere incorporati all'interno dei flussi principali, in modo che routine numerose che compongono un insieme estremamente complesso di regole ETL possano essere consolidate in un unico diagramma visivo.

Oltre a ridurre l'ambiguità, la modellazione tramite diagrammi di flusso aiuta anche a combattere l'incompletezza. Costringe l'autore dei modelli a pensare in termini di vincoli, condizioni negative, restrizioni e condizioni di limite, chiedendosi "che cosa succede se questa causa o trigger non è presente?" Egli deve quindi sistematicamente pensare a percorsi negativi, gestendo il testing negativo che dovrebbe costituire la maggioranza della convalida ETL. È inoltre possibile applicare algoritmi di verifica della completezza, in quanto il modello formale è un diagramma matematicamente preciso della regola ETL.

Questo elimina la logica mancante dei "dangling else", così da poter derivare casi di test che coprono il 100% delle possibili combinazioni di dati (da notare, tuttavia, è il fatto che ci saranno quasi sempre più combinazioni che possono essere eseguite come test in modo economicamente sostenibile; le tecniche di ottimizzazione saranno discusse in seguito). Un altro grande vantaggio è che i risultati attesi possono essere definiti nel modello, indipendentemente dai casi di test. In altre parole, con un diagramma di flusso, l'utente può definire il modello in modo che includa gli input di limite e propagare il proprio risultato atteso ad altri end-point del modello. Questo definisce chiaramente ciò che deve essere accettato e rifiutato da una regola di convalida, evitando che i tester suppongano erroneamente che i test sono riusciti quando il risultato atteso non è esplicito.

Va notato che l'adozione dei test basati su modelli per la convalida ETL non richiede la piena adozione di un approccio basato sui requisiti al test e allo sviluppo a livello dell'intera azienda. Non richiede un cambiamento essenziale: nella nostra esperienza, bastano 90 minuti per modellare una routine ETL come diagramma di flusso "attivo". Questo modello può essere quindi utilizzato dal team ETL o di test solo per i test, e per sottoporre nuovamente a test la stessa routine.

2) Derivare automaticamente casi di test dal modello basato su diagrammi di flusso

L'introduzione dei test basati su modelli è in grado di automatizzare una delle componenti manuali più importanti dei test ETL: la progettazione dei casi di test. I tester non hanno più bisogno di scrivere codice fantasma o di copiare manualmente codice SQL dal database di origine a quello di destinazione. I percorsi all'interno del diagramma di flusso diventano invece i casi di test, utilizzabili per integrare i dati nelle regole di trasformazione. Questi possono essere derivati sistematicamente, in un modo non possibile quando si scrive codice a partire da requisiti statici.

Questa derivazione automatica è possibile perché al diagramma di flusso può essere sovrapposta tutta la logica funzionale presente in un sistema. Algoritmi matematici automatizzati possono quindi essere applicati, per individuare ogni possibile percorso all'interno del modello, generando casi di test che coprono ogni combinazione di input e output (l'analisi di cause ed effetti o omotopica può essere usata a questo scopo).

Poiché i casi di test sono collegati direttamente al modello, coprono tutta la logica in esso definita. Di conseguenza forniscono il 100% della copertura funzionale, in modo che utilizzare un modello di diagramma di flusso per contribuire a una documentazione completa equivale a lavorare per testare completamente una routine ETL. Un ulteriore vantaggio di questo metodo è che l'attività di test diventa misurabile. Perché ogni possibile caso di test può essere derivato, i tester possono determinare esattamente la copertura funzionale fornita da un dato insieme di casi di test.

Ottimizzazione: più attività di test, meno test

Algoritmi di ottimizzazione automatici possono quindi essere applicati, per ridurre il numero di casi di test al minimo, pur mantenendo la massima copertura funzionale. Queste tecniche combinatorie sono rese possibili grazie alla struttura logica del diagramma di flusso, in cui un solo passaggio nella logica di causa ed effetto (un blocco nel diagramma di flusso/componente di test) potrebbe essere incluso in più percorsi all'interno del flusso. Il testing completo della routine ETL diventa allora questione di testare ogni singolo blocco (operatore), utilizzando una delle varie tecniche di ottimizzazione esistenti (Tutti i margini, Tutti i nodi, Tutti i margini in/out, Tutte le coppie). Una serie di 3 casi di test, ad esempio, potrebbe essere sufficiente per testare a fondo la logica applicata su 5 percorsi.

Presso la società di servizi finanziari citata sopra, questo si è rivelato estremamente utile per ridurre l'overtesting, abbreviare i cicli di test e migliorare la qualità dei test. Includendo il tempo necessario per modellare il diagramma di flusso, ad esempio, ci sono voluti 40 minuti per creare 19 casi di test con copertura del 95%, rispetto ai 150 casi di test con l'80% di copertura e un overtesting di 18 volte della situazione precedente. In un altro progetto, la creazione di 17 casi di test con copertura del 100% ha richiesto 2 ore: un netto miglioramento rispetto alla copertura del 16% ottenuta, prima, in 6 ore.

3) Creare automaticamente i dati necessari per eseguire i test

Una volta creati i test, per eseguirli i tester hanno bisogno di dati che coprano il 100% dei test possibili. Questi dati possono anche essere ricavati direttamente dal modello, e possono essere creati automaticamente, o provenire da più fonti contemporaneamente.

Un motore sintetico di generazione di dati come CA Test Data Manager offre diversi modi per creare i dati richiesti utilizzando i test basati su modelli come base per i test ETL. Questo perché, in aggiunta alla logica funzionale, al diagramma di flusso possono essere sovrapposti tutti i dati presenti in un sistema. In altre parole, al momento della modellazione delle regole ETL, i nomi di output, le variabili e i valori predefiniti possono essere definiti per ogni singolo nodo. Quando i casi di test vengono creati, i dati necessari per eseguirli possono essere generati automaticamente dai valori predefiniti, unitamente ai relativi risultati attesi.

In alternativa, grazie all'impiego di CA Agile Requirements Designer (in precedenza Grid Tools Agile Designer), i dati di sistema possono essere creati rapidamente con lo strumento Data Painter. Esso fornisce un elenco completo di funzioni di generazione dei dati combinabili, tabelle di seed, variabili di sistema e variabili predefinite. Queste possono poi essere utilizzate per creare dati a copertura di ogni scenario possibile, inclusi "dati non validi" e percorsi negativi. Dato che ogni percorso è semplicemente un altro punto di dati, i dati necessari per testare sistematicamente la capacità di una routine ETL di rifiutare i dati non validi possono essere creati in modo completamente sintetico.

Infine, i dati esistenti possono essere reperiti in più sistemi di back-end in pochi minuti, mediante mining automatizzato dei dati. Questo utilizza l'analisi statistica per individuare schemi nei database, in modo da poter estrarre gruppi di modelli, dipendenze o record anomali.

4) Eseguire il provisioning dei dati in pochi minuti, già in associazione con i test corretti

Di solito, una combinazione di dati di produzione esistenti e di dati sintetici sarà preferibile, laddove la generazione sintetica sia utilizzata per arrivare a una copertura del 100%. A essere essenziale per l'efficienza dei test ETL è che i dati "gold copy" siano memorizzati in modo intelligente, in modo che gli stessi insiemi di dati possano essere richiesti, clonati e trasmessi in parallelo. Questo elimina i ritardi causati dai vincoli di dati.

Il primo passo per lo storage intelligente dei dati è la creazione di un mart di test, dove i dati vengono "abbinati" a test specifici. A ogni test sono assegnati dati esatti, mentre i dati sono abbinati a criteri stabili e definiti, non a chiavi specifiche. L'abbinamento tra test e dati è in grado di eliminare il tempo altrimenti da dedicare alla ricerca dei dati all'interno di un'origine di produzione di grandi dimensioni, dato che è possibile recuperare automaticamente i dati dal warehouse dei dati di test, oppure estrarli in pochi minuti da più sistemi di back-end.

Il warehouse dei dati di test funge da libreria centrale, in cui i dati vengono memorizzati come risorse riutilizzabili, unitamente alle query associate necessarie per estrarli. I pool di dati possono quindi essere richiesti e ricevuti in pochi minuti, in collegamento con i casi di test corretti e i risultati attesi. Più test vengono eseguiti, più aumenteranno le dimensioni della libreria, fino a quando praticamente ogni richiesta di dati potrà essere eseguita in pochissimo tempo.

È essenziale inoltre il fatto che i dati possano essere integrati in più sistemi contemporaneamente, nonché clonati durante il provisioning. Questo significa che gli insiemi di dati da più database sorgente sono disponibili per più team in parallelo. I test ETL non sono più un processo lineare, e i ritardi creati dai vincoli di dati vengono eliminati. I dati originali possono essere conservati all'esecuzione di modifiche al modello, consentendo ai team di lavorare su più release e versioni in parallelo. Questo "controllo delle versioni" implica anche che le modifiche apportate alle routine ETL si riflettono automaticamente nei dati, fornendo ai team di test i dati aggiornati di cui hanno bisogno per testare in modo rigoroso le trasformazioni.

5) Eseguire i dati rispetto alle regole e confrontare automaticamente i risultati

Una volta che i tester dispongono dei test necessari per testare a fondo una routine ETL, e dei dati necessari per eseguirli, anche la convalida deve essere automatizzata per fare sì che i test ETL tengano il passo con il mutare dei requisiti.

Utilizzo di un motore di orchestrazione dei dati

Un modo per ottenere questo risultato è l'utilizzo di un motore di automazione dei test. CA Technologies offre il vantaggio di fungere anche da motore di orchestrazione dei dati, il che significa che i dati, abbinati a test e risultati attesi specifici, possono essere estratti e inseriti attraverso una regola di convalida. Si tratta di una "automazione basata sui dati" in cui, ad esempio, un harness di test creato in un motore di orchestrazione dei dati può estrarre ogni riga di un file XML, definito nel diagramma di flusso, ed eseguirla come test.

Questo fornirà quindi un risultato pass/fail, sulla base dei risultati attesi definiti nel modello. Questo automatizza sia l'esecuzione dei test che il confronto tra risultati effettivi e attesi. I tester non devono più copiare manualmente gli script dalla destinazione all'origine di dati, e possono evitare anche il processo, complesso e soggetto a errori, di confronto con ogni singolo campo generato dalla trasformazione.

Utilizzo di CA Test Data Manager

In alternativa, una volta che i risultati attesi sono stati definiti, CA Test Data Manager può essere utilizzato per creare automaticamente report pass/fail basati su di essi. Questo automatizza confronti di dati che sarebbero altrimenti manuali; rimane tuttavia l'esigenza di copiare il codice SQL dall'origine alla destinazione.

In primo luogo, i dati di sintesi sono definiti nel sistema di origine, utilizzando le varie tecniche di cui sopra. Un pool di dati può quindi essere creato per simulare il processo ETL, copiando i dati dall'origine alla destinazione. Un insieme valido di dati può essere copiato, per verificare che non venga respinto, così come un insieme di dati con errori, per assicurarsi che vengano respinti i dati non validi. Con la creazione di un'ulteriore tabella per la memorizzazione delle condizioni di test e dei risultati, e la creazione di una tabella basata su pool di dati con condizioni di dati e casi di test, i risultati attesi possono essere confrontati automaticamente con i risultati effettivi. Eventuali dati mancanti o errati possono quindi essere identificati a colpo d'occhio.

6) Implementare automaticamente il cambiamento

Uno dei più grandi vantaggi dei test basati su modelli per la convalida ETL è la capacità di reagire rapidamente al cambiamento. Poiché i casi di test, i dati e i requisiti sono così strettamente collegati, una modifica apportata al modello può essere trasmessa automaticamente ai casi di test e ai dati associati. Questo significa che i test riescono a tenere il passo con la sempre maggiore complessità delle routine ETL, senza che si creino colli di bottiglia nella pipeline di application delivery.

Con la modellazione tramite diagrammi di flusso, l'implementazione di una modifica diventa rapida e semplice quanto l'aggiunta di un nuovo blocco al diagramma di flusso. È quindi possibile applicare algoritmi di controllo della completezza, per convalidare il modello, e per essere certi che ogni componente logico sia stato collegato correttamente all'interno dell'intero flusso. Se si utilizza CA Agile Requirements Designer, è possibile utilizzare Path Impact Analyzer, per identificare ulteriormente l'impatto della modifica sui percorsi all'interno del diagramma di flusso. I casi di test interessati possono quindi essere rimossi o corretti automaticamente, generando in automatico gli eventuali nuovi test necessari per mantenere una copertura funzionale del 100%.

Questo elimina il tempo speso per controllare e aggiornare i test a mano, mentre è dimostrato che la de-duplica automatizzata abbrevia i cicli di test del 30%. Per il team di business intelligence sopra citato, i test basati su modelli hanno apportato enormi risparmi di tempo nel processo ETL. A fronte delle 7,5 ore necessarie a un tester per controllare e aggiornare i casi di test quando era stata modificata una singola regola ETL, l'operazione ha richiesto a CA Agile Requirements Designer appena 2 minuti. Non solo: sul totale dei casi di test, solo 3 erano stati effettivamente interessati e dovevano quindi essere aggiornati.

Come illustrato, il controllo delle versioni dei dati implica anche che ai tester vengono forniti i dati aggiornati di cui hanno bisogno per testare a fondo una routine ETL, anche dopo la sua modifica. Poiché i dati di test sono riconducibili al modello, quando i requisiti cambiano, le modifiche vengono automaticamente trasmesse agli insiemi di dati rilevanti. I dati sono disponibili in tutte le release e le versioni in parallelo, mentre i dati necessari per test di regressione efficienti vengono conservati nel warehouse dei dati di test. Dati interessanti, oppure i rari dati non validi, possono inoltre essere bloccati, per impedire che vengano utilizzati da un altro team, e che vadano persi durante un aggiornamento.

Sezione 4

Riepilogo

L'introduzione di un più elevato grado di automazione nei test ETL è fondamentale per qualsiasi azienda che miri alla continuous delivery di software di alta qualità. La convalida ETL continua a richiedere un grado particolarmente elevato di sforzo manuale, dalla scrittura di codice fantasma a partire dai requisiti statici, al sourcing dei dati richiesti e al confronto dei risultati. I test basati su modelli e la gestione intelligente dei dati di test possono essere utilizzati per automatizzare tutte e ciascuna di queste attività, consentendo a più team di lavorare in parallelo a partire dalle stesse fonti di dati.

I test basati su modelli "spostano a sinistra" lo sforzo richiesto dai test ETL, concentrando la maggioranza del lavoro nella fase di progettazione. Da quel momento, ogni risorsa necessaria per i test ETL può essere derivata automaticamente e in un tempo limitato. Casi di test che forniscono una copertura funzionale al 100% possono essere generati automaticamente, e sono collegati a risultati attesi definiti in modo indipendente. Ogni test è ulteriormente "abbinato" agli esatti dati di origine necessari per eseguirlo che, se conservati in warehouse di dati di test, possono essere erogati a più team in parallelo.

Dato lo stretto legame che si crea tra i test, i dati e i requisiti, i test necessari per verificare nuovamente una routine ETL possono essere eseguiti rapidamente dopo l'esecuzione di una modifica. Il tempo dedicato alla creazione del modello iniziale è quindi rapidamente compensato da quello risparmiato potendo evitare di creare, aggiornare e ripetere manualmente i test. La generazione basata su modelli offre anche il sostanziale vantaggio della riutilizzabilità, dato che i componenti di test possono essere memorizzati come risorse condivisibili, collegate a dati e risultati attesi, nel warehouse dei dati di test. Più test vengono eseguiti, più la libreria si amplia, fino a quando testare regole ETL nuove o aggiornate diventa rapido e semplice quanto la selezione di componenti esistenti.

I test ETL non creano più colli di bottiglia nell'application delivery e possono tenere il passo con il ritmo di crescita dei business basati sui dati. La testabilità di routine sempre più complesse viene conservata, in modo che i test possano gestire la gamma e il volume dei dati raccolti, e non impedisce la continuous delivery di applicazioni di qualità.

Entra in contatto con CA Technologies all'indirizzo ca.com/it

CA Technologies (NASDAQ: CA) crea software che promuove l'innovazione all'interno delle aziende, consentendo loro di sfruttare le opportunità offerte dall'economia delle applicazioni. Il software rappresenta il cuore di qualsiasi business, in ogni settore. Dalla pianificazione allo sviluppo, fino alla gestione e alla sicurezza, CA Technologies lavora con le aziende di tutto il mondo per cambiare il nostro modo di vivere, interagire e comunicare, in ambienti mobile, cloud pubblici e privati, distribuiti e mainframe. Per ulteriori informazioni sui programmi a supporto dei nostri clienti, visita **ca.com/customer-success**. Per ulteriori informazioni, visita il sito **ca.com/it**.

1 Jean-Pierre Dijcks, *Why does it take forever to build ETL processes?*, estratto il 24/07/2015 da **https://blogs.oracle.com/datawarehousing/entry/why_does_it_take_forever_to_bu**

2 Alan R. Earls, *The State of ETL: Extract, Transform and Load Technology*, estratto il 21/07/2015 da **http://data-informed.com/the-state-of-etl-extract-transform-and-load-technology/**

3 IBM, estratto il 20/07/2015 da **http://www-01.ibm.com/software/data/bigdata/what-is-big-data.html**

4 Jacek Becla and Daniel L. Wang, Lessons Learned from managing a Petabyte, P. 4. Estratto il 19/02/2015 da **http://www.slac.stanford.edu/BFROOT/www/Public/Computing/Databases/proceedings/**

5 **http://etlcode.com/index.php/utility/etl_complexity_calculator**

6 Jean-Pierre Dijcks, *Why does it take forever to build ETL processes?*, estratto il 24/07/2015 da **https://blogs.oracle.com/datawarehousing/entry/why_does_it_take_forever_to_bu**

7 ETL Guru, *ETL Startegy to store data validation rules*, estratto il 22/07/2015 da **http://etlguru.com/?p=22**

8 Bender RBT, *Requirements Based Testing Process Overview*, estratto il 05/03/2015 da **http://benderrbt.com/Bender-Requirements%20Based%20Testing%20Process%20Overview.pdf**

9 Huw Price, *Test Case Calamity*, estratto il 21/07/2015 da **https://communities.ca.com/community/ca-agile-requirements-designer/blog/2016/02/24/test-case-calamity**

10 Bender RBT, *Requirements Based Testing Process Overview*

11 Software Testing Class, *Why testing should start early in software development life cycle?*, estratto il 06/03/2015 da **http://www.softwaretestingclass.com/why-testing-should-start-early-in-software-development-life-cycle/**

12 datagaps, *ETL Testing Challenges*, estratto il 24/07/2015 da **http://www.datagaps.com/etl-testing-challenges**

13 Robin F. Goldsmith, *Four Tips for Effective Software Testing*, estratto il 20/07/2015 da **http://searchsoftwarequality.techtarget.com/photostory/4500248704/Four-tips-for-effective-software-testing/2/Define-expected-software-testing-results-independently**

14 Jagdish Malani, ETL: *How to handle bad data*, estratto il 24/07/2015 da **http://blog.aditi.com/data/etl-how-to-handle-bad-data/**

15 Philip Howard, *Automated Test Data Generation Report*, P. 6. Estratto il 22/07/2015 da **http://www.agile-designer.com/wpcms/wp-content/uploads/2014/10/00002233-Automated-test-case-generation.pdf**