LIBRO BLANCO | ABRIL DE 2016

Automatización total de las pruebas de ETL: guía paso a paso

ca technologies

Sección 1

La función clave del estándar ETL para las organizaciones modernas

Desde que irrumpió en el mundo del almacenamiento de datos y la inteligencia de negocio, el estándar Extract, Transform, Load o ETL (Extracción, Transformación, Carga) se ha convertido en un proceso ubicuo dentro del ámbito del software. Como su propio nombre sugiere, una rutina de ETL consta de tres fases distintas, que a menudo se producen en paralelo: se extraen datos de una o varias fuentes, se convierten al estado que se requiere y se cargan en el destino deseado, que normalmente es un almacén de datos, un data mart o una base de datos. Una rutina de ETL desarrollada también suele incluir una infraestructura de inicio de creación de registros, la manipulación de errores y el entorno de rutina.[1]

Hasta el momento, el estándar ETL se ha utilizado principalmente para preparar datos de gran volumen y naturaleza dispar que serían explotados con fines de análisis e inteligencia de negocio. Sin embargo, su utilización gana terreno más allá del mero transporte de datos, siendo la migración de datos para nuevos sistemas una aplicación cada vez más común, que se une a la manipulación de integraciones, clasificaciones y fusiones de datos.[2]

Por lo tanto, ETL es una característica esencial del moderno y acelerado ciclo de vida del desarrollo, donde en cualquier momento dado se trabaja en varias versiones y lanzamientos en paralelo. Las organizaciones deben ser capaces de mejorar, integrar e innovar de forma constante su software, poniendo a disposición de los responsables de pruebas y desarrolladores datos de prueba adecuados para cada iteración y lanzamiento. Los datos se obtendrán de las mismas fuentes, pero se deben transformar para adaptarlos a los requisitos específicos de cada equipo individual. Esto es especialmente cierto si una organización se esfuerza por adoptar una metodología ágil o está intentando implementar con éxito la entrega continua.

Un buen ejemplo del papel crítico que desempeña el estándar ETL en el ámbito de la entrega continua lo encontramos en un gran banco multinacional con el que ha trabajado CA Technologies. El banco se hallaba inmerso en el proceso de una adquisición y tenía que migrar los clientes de la entidad comprada, así como sus productos y actividades financieras para incorporar todo a la infraestructura que ya existía. Esto implicaba que fuese necesario recuperar, convertir y validar 80 archivos de inserción antes de cargarlos en los sistemas back-end del banco. Además, estos datos deberían estar disponibles para 47 proyectos en paralelo, manteniendo la integridad referencial de los datos. En este caso, el estándar ETL fue fundamental para posibilitar el trabajo en paralelo, requisito para implantar con éxito la entrega continua.

No obstante, a pesar del incremento en el uso y la importancia de ETL, las pruebas de ETL reflejan el estado general de la ejecución de pruebas en general: son demasiado lentas, exigen demasiada intervención manual y, a pesar de todo, permiten que llegue a la fase de producción una cantidad inaceptable de defectos. Este artículo se ocupará de definir los retos que nos encontramos en un enfoque típico de las pruebas de ETL y explorar las dificultades halladas. A continuación, estableceremos un enfoque alternativo, basado ampliamente en los modelos y sopesaremos cómo podría favorecer a las pruebas de ETL para que fuesen más eficientes, eficaces y sistemáticas.

Sección 2

El enfoque típico de las pruebas de ETL y los retos comunes detectados

Al validar reglas de transformación de ETL, los responsables de las pruebas suelen crear un conjunto de código sombra. Lo utilizan para transformar los datos y después comparan los resultados reales con los resultados esperados. Habitualmente se copian scripts o fragmentos de SQL sobre los datos de origen y se ejecutan para seguidamente registrar los resultados. El mismo script se copia a continuación sobre los datos de destino y también se registran los resultados. Posteriormente, se comparan los dos conjuntos de resultados (reales y previstos) para validar que los datos se hayan transformado correctamente.

El problema de raíz: complejidad y posibilidad de realizar pruebas

El problema subyacente a esa validación manual es que las rutinas de ETL, por su naturaleza intrínseca, no tardan en volverse muy complejas. A medida que crece el negocio y aumenta la variedad y el volumen de datos que recopila, también crecen las reglas de ETL con el fin de manejarlo. En lo que se ha dado en llamar "la Era de la información", este crecimiento se produce con más velocidad de la que son capaces de admitir los métodos de pruebas tradicionales. De hecho, la propia cantidad de información que recopilan las organizaciones que trabajan sobre datos ha subido tan rápidamente que el 90 % de los datos del mundo se han recogido durante los últimos dos años[3], mientras que el volumen de datos reunido por una organización media se duplica cada año.[4]

La complejidad de los sistemas diseñados para recoger, transferir, procesar y presentar estos datos aumenta de forma exponencial con cada decisión que se añade. Esto incluye las reglas de ETL. Hay numerosos factores que pueden incidir sobre la complejidad de las transformaciones:

- El número y la variedad de las fuentes de datos involucradas, incluidos los tipos de bases de datos relacionales y no relacionales, así como los archivos sin formato.
- El número y la variedad de los destinos de datos.
- El número y la variedad de las transformaciones simples y complejas, desde sencillas consultas hasta uniones activas y funciones de normalización.
- El número de transformaciones reutilizables, fragmentos de código y uniones.
- El número de tablas creadas.[5]

Todos estos factores se ven acentuados por la atención que actualmente se presta a las soluciones cercanas al tiempo real, así como por las complicaciones que estas últimas conllevan.[6]

La documentación no ayuda

Esta creciente complejidad repercute directamente sobre la posibilidad de someter a pruebas las rutinas de ETL. Se trata de algo especialmente problemático para las pruebas de ETL, ya es frecuente que las reglas de transformación se almacenen en una documentación deficiente, que carece de indicaciones claras sobre los resultados esperados. A menudo, las reglas se diseñan durante la propia fase de desarrollo y con frecuencia se almacenan en documentos escritos u hojas de cálculo... o aún peor, es posible que ni siquiera existan fuera de la mente de los desarrolladores y responsables de pruebas.[7] En este caso, no existe ninguna documentación real que se pueda tomar como base para derivar con seguridad los casos de prueba (es decir, el código sombra).

En un equipo de inteligencia de negocio con el que trabajamos, los requisitos se almacenaban como documentos escritos, mientras que los casos de pruebas se guardaban en hojas de cálculo. Esta documentación estática se presentaba como un gran bloque de texto, del que había que descifrar los pasos lógicos de las rutinas de ETL. Los documentos se centraban en "la ruta más feliz" y no contenían condiciones negativas, así que se omitía alrededor del 80 % del contenido lógico posible que sería preciso poner a prueba en un sistema medio. La documentación, ambigua e incompleta, provocaba que los responsables de pruebas no tuviesen manera de comprender con facilidad o exactitud las rutinas de ETL.

Con demasiada frecuencia, los responsables de pruebas se limitaban a rellenar los huecos, pero cuando se equivocaban, se filtraban defectos que se incorporaban a las rutinas de ETL. A continuación, los datos no válidos se copiaban en el destino, aunque el código y los casos de prueba reflejasen una interpretación plausible de la documentación sobre requisitos.

“Basura para dentro, basura para fuera”: la derivación manual de los casos de prueba y los resultados previstos

De hecho, el origen de un 56 % de los defectos que se filtran hasta acabar en producción (un volumen masivo) se puede rastrear hasta la ambigüedad en la documentación de requisitos.[8] En parte ello se debe a que los casos de prueba y los resultados previstos se derivan manualmente de la documentación, que es insuficiente: un proceso que consume cantidades ingentes de tiempo y suele desembocar en un bajo nivel de cobertura para las pruebas.

Calidad

La derivación manual es un proceso ad hoc y nada sistemático, que habitualmente provoca que la creación de los casos de prueba tenga lugar en la mente de los responsables de pruebas. Vista la complejidad de las rutinas de ETL que hemos discutido, combinada con el bajo nivel de la documentación disponible, no sería justo esperar que un responsable de pruebas crease todas las comprobaciones necesarias para validar las posibles combinaciones de datos, por más talento que tuviese. Por ejemplo, si un sistema simple con 32 nodos y 62 bordes se diseña de forma lineal, hay 1 073 741 824 posibles rutas a través de ese conjunto lógico, más de las que puede plantearse ninguna persona.

Por lo tanto, la derivación ad hoc causa problemas masivos debidos a pruebas excesivas o insuficientes, donde solamente se analiza una fracción de todo el contenido lógico posible involucrado en una rutina de ETL. Las pruebas negativas supondrán un reto concreto y los casos de prueba, como la documentación, normalmente se centran en las rutas más favorables de forma casi exclusiva. Sin embargo, son precisamente los resultados inesperados los que se deben poner a prueba, ya que es imperativo que las rutinas de ETL rechacen los “datos malos”.

Una empresa de servicios financieros con la que ha colaborado CA Technologies, por ejemplo, depositó su confianza en 11 casos de prueba con una cobertura de pruebas de tan solo el 16 %. Esta cifra es bastante estándar y nuestras auditorías han detectado que la norma es que la cobertura funcional de las pruebas se sitúe entre el 10 y el 20 %. En otro proyecto de la empresa, la carga de pruebas era excesiva, 18 veces por encima de lo necesario, ya que se acumulaban casos de prueba apilándolos en un esfuerzo por poner a prueba el sistema por completo, pero no se conseguía alcanzar la cobertura máxima. Los 150 casos de prueba extra supusieron un coste de 26 000 dólares para la empresa, que optó por ejecutarlos con un proveedor externo.[9]

El resultado de una cobertura tan deficiente es que algunos defectos se introducen en el código, donde corregirlos exige mucho tiempo y dinero: distintos estudios han constatado que puede costar entre 40 y 1000 veces más recursos[10] y 50 veces más tiempo[11] reparar un error durante la realización de pruebas, en comparación con lo que costaría si se detectase antes. Pero aún puede ser peor: los errores pueden pasar sin ser detectados, de modo que los datos no válidos se copien en el destino, donde pueden poner en entredicho la integridad del sistema. Todavía hay más: al enfrentarse a una documentación estática, los responsables de pruebas no tienen un método fiable para medir la cobertura de sus casos de prueba. Sencillamente, no pueden decir con confianza qué porción de una rutina dada están poniendo a prueba ni pueden asignar prioridad a las pruebas basándose en su grado de importancia.

Tiempo y esfuerzo: sencillamente, las pruebas no pueden seguir el ritmo impuesto

Redactar casos de prueba a partir de este tipo de documentación también consume grandes cantidades de tiempo y trabajo. En el ejemplo anterior, fueron necesarias seis horas para crear 11 casos de prueba, sin olvidar que la sobreutilización descontrolada de las pruebas en la empresa llevó todavía más tiempo. Este tiempo malgastado en el diseño manual de casos de prueba se agrava al sumarle el tiempo que es preciso dedicar a comparar los resultados reales y previstos.

Comparar los vastos campos individuales con los resultados esperados requiere muchísimo tiempo, dada la cantidad de datos que arroja una rutina de ETL compleja y el hecho de que los datos de origen a menudo se almacenan en una diversidad de bases de tipos de archivos y bases de datos. También resulta muy difícil porque los datos transformados deben validarse en varios niveles:

- Los responsables de pruebas deben comprobar si los datos están completos, asegurándose de que coincidan los recuentos de orígenes y destinos de datos.
- Debe garantizarse la integridad de los datos, verificando que los datos de destino sean coherentes con los datos de origen.
- La transformación debe encajar con las reglas del negocio.
- Es preciso que se garantice la coherencia de los datos y que se identifiquen los duplicados inesperados.
- Debe mantenerse la integridad referencial y detectar las claves ajenas que falten o los registros huérfanos.[12]

A veces hay que ceder y alcanzar un compromiso para validar solamente un conjunto de datos de muestra. Sin embargo, esto también pone en entredicho la completitud de las pruebas de ETL, por lo cual resulta afectada la fiabilidad de las transformaciones. Dada la función que desempeñan muchas rutinas de ETL en operaciones esenciales del negocio, tal concesión es inaceptable. La calidad sufre también por la naturaleza de las comparaciones manuales, propensas a los errores, especialmente si los resultados esperados no se han definido con precisión o, todavía peor, si ni siquiera se definen de forma independiente respecto del código sombra utilizado para realizar pruebas. En este caso, los responsables de pruebas tienden a presumir que se ha pasado una prueba, salvo si los resultados reales son especialmente extraños: sin resultados esperados predefinidos, es probable que asuman que el resultado real es el resultado que se esperaba[13], así que no se puede determinar con confianza la validez de los datos.

El problema de los datos

Hasta el momento, nos hemos centrado en las dificultades topadas al derivar las pruebas (código sombra) necesarias para validar las reglas de ETL. Sin embargo, después de derivar los casos de prueba, los responsables de las pruebas necesitan datos de origen simulados que sirvan para alimentar el sistema. Este es otro de los puntos donde es corriente ver cuellos de botella y defectos.

¿Tiene todos los datos necesarios para poner a prueba las complejas rutinas de ETL?

Disponer de un volumen suficiente de datos "malos" es esencial para realizar pruebas de ETL con eficacia, ya que es primordial que, al efectuar la operación, haya una regla de ETL que rechace esos datos y los remita al usuario apropiado con el formato adecuado. Si no se rechazan, es probable que los datos malos generen defectos o incluso provoquen el colapso del sistema.

En este contexto, se puede definir los "datos malos" de varias maneras, que corresponden a las formas en que deben validarlos los responsables de las pruebas. Puede tratarse de datos que, basándose en las reglas empresariales, nunca se deberían aceptar. Por ejemplo, valores negativos en una cesta de compra en línea cuando no hay presente ningún vale. Puede tratarse de datos que amenacen la integridad referencial de un almacén, como que falten datos obligatorios o interdependientes, o bien que entre los propios datos entrantes se detecte la ausencia de ciertos datos.[14] Por tanto, los datos de pruebas con que se alimenta una regla de validación de ETL deben contener el rango completo de datos no válidos, con el fin de posibilitar una cobertura funcional 100 % de las pruebas.

No es nada frecuente encontrar datos así entre los orígenes de datos de producción que todavía se proporcionan a los equipos de pruebas en muchas organizaciones. Esto se debe a que los datos de producción se obtienen de escenarios donde los negocios transcurren según lo habitual. Escenarios que han ocurrido anteriormente, así que son datos saneados por su propia naturaleza, que excluye los datos malos. Estos datos no contienen los resultados inesperados, datos atípicos, ni condiciones de límites necesarias para realizar pruebas de ETL y, en su lugar, se centran en la "ruta más feliz". De hecho, nuestras auditorías hechas sobre datos de producción han detectado que lo normal es que la cobertura oscile tan solo entre el 10 y el 20 %. Lo irónico es que, cuanto mejor se haya diseñado y construido la rutina, menos "datos malos" se habrán admitido, lo que significa que se cuenta con menos datos de variedad suficiente para probar por completo las reglas de ETL posteriormente.

¿Los datos están disponibles cuando los necesita?

Otro gran problema de la validación de ETL es la disponibilidad de los datos. Se pueden obtener datos de origen desde 50 fuentes diferentes dentro de una misma empresa. El problema para las pruebas de ETL y para las pruebas en general es que se perciben como una serie de etapas lineales, de modo que los equipos de prueba se ven obligados a esperar por los datos mientras los está utilizando otro equipo.

Tomemos el ejemplo de una cadena de migración bancaria, donde los datos se toman de un banco y se convierten a los sistemas de otra entidad mediante una herramienta de conciliación. En cada etapa hay que validar los datos, comprobar si se han convertido correctamente al marco de trabajo de control financiero, verificar que se recuperó el número de cuenta, que se registró total corrección de hora a hora, etc. Este proceso podría caer en varias fases diferentes, desde la introducción básica de datos, pasando por la deduplicación y la preparación, hasta la propagación y la reserva de los datos para fines específicos. Además, podrían implicarse varios equipos, incluidos tanto equipos de ETL como otros ajenos a esta metodología, particularmente los que trabajen con el mainframe.

Si los datos de cada origen de datos a lo largo y ancho de la empresa no están disponibles para todos los equipos en paralelo, surgirán retrasos mientras los equipos esperan sentados sin nada que hacer. Los responsables de pruebas escribirán un código fantasma y luego no tendrán los datos de origen precisos para validar una regla de ETL, ya que los estará utilizando otro equipo. De hecho, hemos descubierto que el responsable de pruebas medio puede dedicar el 50 % de su tiempo a esperar, buscar, manipular o crear datos. Esto puede suponer nada menos que el 20 % del tiempo total que dura el ciclo de vida de desarrollo del software.

¿Qué sucede cuando cambian las reglas?

Derivar manualmente los casos de prueba y los datos a partir de requisitos estáticos constituye un método que dificulta mucho la labor de reaccionar ante los cambios. Las rutinas de ETL cambian a la misma velocidad que evoluciona el negocio y el volumen y la variedad de los datos que se recopilan aumentan con ello. Cuando ocurre un cambio así de constante, no obstante, las pruebas de ETL no son capaces de seguir este ritmo.

Se podría decir que la mayor causa de los retrasos en los proyectos en esta instancia es tener que comprobar y actualizar los casos de prueba existentes cuando las rutinas sufren modificaciones. Los responsables de pruebas no tienen ninguna manera de identificar automáticamente el impacto de un cambio realizado en los requisitos estáticos y los casos de prueba. En vez de ello, están obligados a comprobar todos los casos de prueba existentes a mano, sin ninguna forma de medir que se ha mantenido realmente la cobertura.

Con el equipo de inteligencia de negocio mencionada anteriormente, donde los requisitos y los casos de prueba se almacenaban en documentos escritos y hojas de cálculo respectivamente, los cambios resultaron especialmente problemáticos. Un responsable de pruebas tuvo que emplear 7,5 horas para verificar y actualizar un conjunto de casos de prueba cuando se realizó una modificación en una sola regla de ETL. En otra organización con la que colaborábamos, dos probadores dedicaron dos días a comprobar todos los casos de prueba existentes cuando se produjo un cambio en los requisitos.

Sección 3

La alternativa viable: pruebas de ETL totalmente automatizadas

Está claro entonces que, mientras para derivar los casos de prueba y comparar los resultados se recurra a métodos manuales, las pruebas de ETL no podrán seguir el ritmo al que cambian constantemente los requisitos del negocio. Más abajo figura una posible estrategia para mejorar la eficiencia y la eficacia de las pruebas de ETL. Se trata de una estrategia basada en requisitos y en modelos, diseñada para que el trabajo de realizar las comprobaciones se aplique en fases más tempranas y para incorporar la calidad en el ciclo de vida de ETL desde sus primeros pasos. Un enfoque así, basado en modelos, introduce la automatización en todas las etapas de las pruebas y el desarrollo, al tiempo que confiere a las pruebas de ETL una capacidad total de reacción frente a los cambios constantes.

1) Empezar por un modelo formal

Introducir el modelado formal en las pruebas de ETL ofrece una ventaja fundamental: adelanta el trabajo de las comprobaciones a fases más tempranas, donde es posible derivar todos los activos de pruebas y desarrollo subsiguientes a partir del esfuerzo inicial, dedicado a asignar una regla de ETL a un modelo. Por lo tanto, el modelo formal se convierte en la piedra angular de la validación totalmente automatizada de ETL.

No obstante, el modelado formal también ayuda a resolver el problema más específico que hemos citado anteriormente, debido a que los requisitos son ambiguos e incompletos. Contribuye a mantener la posibilidad de realizar pruebas a pesar de la creciente complejidad de las reglas ETL, que aumenta sin parar. De este modo, los responsables de pruebas pueden comprender de forma rápida y visual cuál es la lógica exacta que es preciso comprobar. Pueden saber fácilmente tanto los datos válidos como los datos no válidos que se deberían introducir para probar por completo una regla de transformación y qué resultado esperado debería aparecer en cada instancia.

Un modelo de diagramas de flujo, por ejemplo, descompone ese inmenso bloque de texto que constituye la documentación en fragmentos digeribles, que de otra forma sería poco manejable. Reduce el ETL a su lógica de causa y efecto, relacionándolo con una serie de enunciados del tipo "¿Qué sucedería si se diesen estas condiciones y luego sucediese esto?", enlazadas en una jerarquía de procesos.[15] Cada uno de estos pasos efectivos se convierte en un componente de las pruebas, que comunica al probador exactamente qué es necesario validar. Por lo tanto, modelar rutinas de ETL como un diagrama de flujo elimina la ambigüedad de la documentación de los requisitos, lo que contribuye a evitar el 56 % de los defectos que hunden sus raíces ahí.

A medida que las rutinas de ETL ganan en complejidad, el diagrama de flujo sirve como punto único de referencia. En contraste con los diagramas y documentos escritos "estáticos", resulta fácil agregar más contenido lógico al modelo. Por si fuera poco, posibilita la abstracción de rutinas muy complicadas utilizando tecnología de flujos secundarios, lo que incrementa las posibilidades de realizar pruebas. Los componentes de niveles inferiores se pueden incrustar dentro de los flujos maestros, de forma que las numerosas rutinas que componen un conjunto muy complejo de reglas de ETL se pueden consolidar en un único diagrama visual.

Además de reducir la ambigüedad, el modelado de diagramas de flujo también contribuye a combatir la incompletitud. Obliga al creador del modelo a pensar en términos de restricciones, condiciones negativas, limitaciones y condiciones límite, planteándose qué sucede si cierta causa o elemento desencadenante no está presente. Por tanto, deben concebir de forma sistemática las rutas negativas, dejando espacio para las pruebas negativas, que deberían constituir la mayor parte de la validación de ETL. También se pueden aplicar algoritmos de comprobación de la completitud, ya que el modelo formal es un diagrama matemáticamente preciso de la regla de ETL.

Esto elimina contenido lógico que falta, como los "flecos", de modo que se hace posible derivar casos de prueba que cubren el 100 % de las posibles combinaciones de datos. Debemos tener en cuenta, sin embargo, que siempre habrá más combinaciones de las que sea viable ejecutar como pruebas. Más adelante abordaremos las técnicas de optimización. Otra de las grandes ventajas es que los resultados esperados se pueden definir en el modelo, con independencia de los casos de prueba. Dicho de otra forma, con un diagrama de flujo, el usuario puede definir el modelo para incluir las entradas de límites y propagar su resultado previsto a diferentes puntos finales dentro del modelo. Esto define con claridad qué se debería aceptar y rechazar una regla de validación, de manera que los responsables de pruebas no asuman incorrectamente que se han superado las pruebas cuando el resultado esperado no es explícito.

Debemos tener en cuenta que adoptar la realización de pruebas basado en modelos para la validación de ETL no requiere la adopción total de un enfoque impulsado por requisitos para las pruebas y el desarrollo en toda la empresa. No exige un cambio radical y, de acuerdo con nuestra experiencia, no lleva más de 90 minutos modelar una rutina de ETL como un diagrama de flujo "activo". Este modelo puede ser utilizado a continuación por el método ETL o el equipo de pruebas para el propósito exclusivo de comprobar y volver a comprobar la rutina misma.

2) Derivación automática de casos del modelo de diagrama de flujo

La introducción del método de realización de pruebas basado en modelos puede automatizar uno de los principales elementos manuales de las pruebas de ETL: el diseño de casos de prueba. Ya no es necesario que el responsable de pruebas escriba código fantasma ni copie manualmente fragmentos de SQL desde la base de datos fuente hasta la de destino. En vez de ello, las rutas que recorren el diagrama de flujo se convierten en los casos de prueba, que se pueden utilizar para alimentar con datos las reglas de transformación. Estas se pueden derivar sistemáticamente, de una forma que sería imposible al escribir el código partiendo de requisitos estáticos.

Esta derivación automática es posible debido a que el diagrama de flujo se puede revestir con toda la lógica funcional involucrada en un sistema. A continuación, se pueden aplicar los algoritmos matemáticos automatizados para identificar todas las rutas posibles a través del modelo, lo que genera casos de prueba que cubren todas las combinaciones de entradas y salidas (para hacer esto se puede recurrir al análisis de causas y efectos o análisis homotópico).

Dado que los casos de prueba están relacionados directamente con el propio modelo, cubren todo el contenido lógico que está definido en él. Por lo tanto, proporcionan una cobertura funcional del 100 %, así que utilizar un modelo de diagramas de flujo para avanzar hacia una documentación completa es equivalente a trabajar con el fin de probar por completo una rutina de ETL. Una ventaja adicional de este método es que las pruebas se vuelven mensurables. Puesto que es posible derivar todos los casos de prueba posibles, los responsables de las pruebas pueden determinar con exactitud qué grado de cobertura funcional proporciona un conjunto dado de casos de prueba.

Optimización: comprobar más con menos pruebas

A continuación, los algoritmos de optimización automatizados se pueden aplicar para reducir el número de casos de prueba hasta el mínimo, al tiempo que se retiene la máxima cobertura funcional. Estas técnicas combinatorias son posibles en virtud de la estructura lógica del diagrama de flujo, donde un solo paso en la lógica de causa y efecto (un bloque dentro del diagrama de flujo/componente de prueba) puede figurar en varias rutas que atraviesan el flujo. Probar por completo la rutina de ETL se convierte entonces en cuestión de comprobar cada bloque (operador) individual, utilizando una de las varias técnicas de optimización existentes (Todos los bordes, Todos los nodos, Todos los bordes de entrada/salida y Todos los pares). Por ejemplo, un conjunto de tres casos de prueba podría ser suficiente para probar a fondo la lógica involucrada en cinco rutas.

En la compañía de servicios financieros mencionada anteriormente, esto reveló ser valiosísimo para reducir las pruebas excesivas, acortar los ciclos de pruebas y mejorar la calidad de las pruebas. Incluido el tiempo destinado a modelar el diagrama de flujo, por ejemplo, llevó 40 minutos crear 19 casos de prueba con una cobertura del 95 %, en contraste con los 150 casos de prueba que alcanzaban anteriormente una cobertura del 80 % y multiplicaban por 18 el exceso de pruebas. En otro proyecto, llevó dos horas crear 17 casos de prueba, con una cobertura del 100 %: una mejora drástica respecto al 16 % de cobertura logrado anteriormente tras seis horas de trabajo.

3) Creación automática de los datos necesarios para ejecutar las pruebas

Una vez creados los casos de prueba, los responsables de las pruebas requieren datos que puedan cubrir el 100 % de las posibles pruebas con el fin de ejecutarlas. Estos datos se pueden derivar directamente del propio modelo y se pueden crear automáticamente u obtenerlos de varias fuentes simultáneamente.

Un motor de generación de datos sintéticos como CA Test Data Manager ofrece distintas maneras de crear los datos exigidos cuando se aplica el método de pruebas basado en modelos para impulsar las pruebas de ETL. Esto se debe a que, además de la lógica funcional, un diagrama de flujo también se puede cumplimentar con todos los datos implicados en un sistema. En otras palabras, a medida que se modelan las reglas de ETL, se pueden definir nombres de salida, variables y valores predeterminados para cada nodo individual. Cuando se crean los casos de prueba, los datos necesarios para ejecutarlos se pueden generar automáticamente a partir de valores predeterminados, junto con los resultados esperados relevantes.

De manera alternativa, mediante CA Agile Requirements Designer (anteriormente, Agile Designer de Grid Tools), pueden crearse con rapidez datos del sistema con la herramienta Data Painter. Esto proporciona una lista completa de funciones de generación de datos combinables, tablas raíz, variables de sistema y variables predeterminadas. Se pueden utilizar para crear datos que cubran todos los escenarios posibles, que incluyan rutas negativas y "datos malos". Como cada ruta no es más que otro punto de datos, los datos requeridos para probar sistemáticamente la capacidad de una rutina de ETL para rechazar datos inválidos se pueden crear de forma totalmente sintética.

Finalmente, los datos existentes se pueden encontrar a partir de diversos sistemas back-end en cuestión de minutos, utilizando la minería de datos automatizada. Esta variante aplica el análisis estadístico para descubrir patrones en las bases de datos, de modo que sea posible extraer los grupos de patrones, dependencias o registros inusuales.

4) Aprovisionamiento de los datos en minutos, "asignados" a las pruebas correspondientes

Normalmente será preferible trabajar con una combinación de datos existentes, datos de producción y datos sintéticos, donde la generación sintética se emplee para subir la cobertura hasta el 100 %. El factor crítico para que las pruebas de ETL sean eficientes es que los datos de la "copia de oro" se guarden de manera inteligente, para que se puedan solicitar, clonar y entregar en paralelo los mismos conjuntos de datos. A su vez, esto elimina los retrasos provocados por las restricciones de los datos.

El primer paso de cara al almacenamiento de datos inteligente es crear un "test mart", una suerte de repertorio de pruebas donde los datos se "asignen" a pruebas específicas. Cada prueba recibe datos exactos asignados y esos datos se corresponden con criterios estables y definidos, no con claves específicas. Los datos de la correspondencia de pruebas permiten ahorrarse el tiempo que, de otra forma, se gastaría en buscar datos dentro de la gran fuente de datos de producción, ya que se pueden recuperar automáticamente desde el almacén de datos para pruebas o se pueden extraer en minutos desde varios sistemas back-end.

El almacén de datos de pruebas sirve como biblioteca central, donde se almacenan los datos como activos reutilizables, junto a las consultas asociadas que son necesarias para extraerlos. Se pueden solicitar y recibir agrupaciones de datos en cuestión de minutos, ligadas a los casos de prueba y los resultados esperados correspondientes. Cuantas más pruebas se realicen, más crece esta biblioteca, hasta que prácticamente sea posible ejecutar todas las solicitudes de datos en una fracción del tiempo gastado inicialmente.

Un detalle crucial es que los datos permitan alimentar varios sistemas simultáneamente y se clonen a medida que se suministren. Esto significa que los conjuntos de datos de muchas bases de datos fuente están disponibles para varios equipos en paralelo. Las pruebas de ETL no constituyen ya un proceso lineal y desaparecen los largos retrasos provocados por las restricciones en los datos. Los datos originales se pueden mantener a medida que se efectúan cambios en el modelo, lo que habilita a los equipos para trabajar sobre varias versiones y lanzamientos en paralelo. Este "control de versiones" también implica que los cambios introducidos en las rutinas de ETL se reflejan automáticamente en los datos, lo que proporciona a los equipos de pruebas los datos actualizados que necesitan para comprobar las transformaciones con rigor.

5) Ejecución de datos respecto a las reglas y comparación automática de los resultados

Una vez que los responsables dispongan de las pruebas necesarias para comprobar a fondo una rutina de ETL y los datos precisos para ejecutarlas, la validación en sí también debe automatizarse si el método de pruebas de ETL es capaz de seguir el ritmo de los requisitos cambiantes.

Uso de un motor de orquestación de datos

Una forma de hacerlo es emplear un motor de automatización de pruebas. CA Technologies ofrece la ventaja de actuar también como motor de orquestación de datos, lo que implica que los datos, asignados a las pruebas específicas y los resultados esperados, se pueden tomar para alimentar una regla de validación. Esta es la "automatización impulsada por datos", donde por ejemplo, un agente de pruebas creado en un motor de orquestación de datos puede tomar cada una de las filas de un archivo XML, definido en el diagrama de flujo, y ejecutarlo como una prueba.

Esto proporcionará un resultado de prueba superada/no superada, basado en los resultados esperados definidos en el modelo. Así se automatizan tanto la ejecución de las pruebas como la comparación de los resultados reales y previstos. Los responsables de pruebas ya no tienen que copiar manualmente los scripts desde el destino de los datos hasta la fuente de los datos. Además, también evitan el proceso de tener que comparar cada campo individual producido por la transformación, muy laborioso y propenso a errores.

Uso de CA Test Data Manager

Como alternativa, después de definir los resultados esperados, se puede utilizar CA Test Data Manager para crear automáticamente informes de prueba superada/no superada basados en tales resultados. Así se automatizan las comparaciones de datos, que de otra manera serían manuales. Ahora bien, aún es preciso copiar fragmentos de SQL de la fuente al destino.

En primer lugar, se definen los datos sintéticos en el sistema fuente por medio de diversas técnicas enumeradas anteriormente. A continuación, se puede crear una agrupación de datos para simular el proceso ETL, copiando los datos de la fuente al destino. Es posible copiar un conjunto de datos válido para comprobar que no se rechaza, además de un conjunto de datos con errores para asegurarnos de que los datos no válidos sí se rechazan. Mediante la creación de otra tabla para almacenar las condiciones de prueba y los resultados y con la creación de una tabla basada en agrupaciones de datos con casos de prueba y condiciones de datos, se puede comparar automáticamente los datos esperados con los resultados reales. Cualquier dato ausente o erróneo se puede detectar entonces de un vistazo.

6) Implementación automática de cambios

Una de las mayores ventajas del método de pruebas basado en modelos para la validación de ETL es la capacidad de reaccionar ante los cambios a gran velocidad. Dado que los casos de prueba, los datos y los requisitos están tan estrechamente relacionados, si se realiza un cambio en el modelo, se puede reflejar automáticamente en los casos de prueba y los datos asociados. Esta trazabilidad implica que, a medida las rutinas de ETL se vuelven cada vez más complejas a gran velocidad, las pruebas son capaces de adaptarse al ritmo y no se originan cuellos de botella en la segmentación de la entrega de aplicaciones.

Con el modelado de diagramas de flujo, implementar un cambio se vuelve en una tarea tan rápida y simple como añadir un bloque nuevo al diagrama de flujo. A continuación, se pueden aplicar los algoritmos de comprobación de la completitud para validar el modelo y asegurarse de que todos los elementos de la lógica se hayan conectado correctamente en un flujo completo. Si se utiliza CA Agile Requirements Designer, se puede emplear el analizador Path Impact Analyzer para identificar, además, el impacto de los cambios sobre las rutas que recorren el diagrama de flujo. Entonces, los casos de prueba afectados se pueden eliminar o reparar automáticamente. Cualquier prueba nueva necesitaría retener el 100 % de la cobertura funcional generada automáticamente.

Así se ahorra el tiempo que antes se desperdiciaba en comprobar y actualizar las pruebas a mano, mientras que se ha demostrado que la deduplicación automatizada acorta los ciclos de pruebas en un 30 %. En el equipo de inteligencia de negocio mencionado más arriba, el método de pruebas basado en modelos permitió lograr grandes ahorros de tiempo en el proceso de ETL. Mientras que cuando cambiaba una sola regla de ETL hacían falta 7,5 horas para comprobar y actualizar los casos de prueba, a CA Agile Requirements Designer solamente le cuesta dos minutos. Es más; del total de casos de prueba, solamente tres habían sufrido realmente los efectos y debían, por tanto, actualizarse.

Como hemos descrito, el control de versión de los datos también implica que los probadores reciban los datos actualizados que necesitan para comprobar por completo una rutina de ETL, aunque haya cambiado. Dado que los datos de prueba se pueden rastrear hasta el modelo, cuando cambian los requisitos, los cambios se reflejan automáticamente en los conjuntos de datos relevantes. Los datos están disponibles en varias versiones y lanzamientos en paralelo, mientras que los datos necesarios para realizar pruebas de regresión eficaces se conservan en el almacén de datos de pruebas. También es posible bloquear los datos considerados interesantes o “malos” e infrecuentes, para evitar que otro equipo los procese y prevenir el riesgo de que se pierdan durante una actualización de datos.

Sección 4

Resumen

La introducción de un grado de automatización más elevado en las pruebas de ETL es imprescindible para cualquier organización que aspire a implantar la entrega continua de software de alta calidad. En la validación de ETL, se mantiene un grado de esfuerzo manual especialmente alto, desde la necesidad de escribir manualmente código fantasma a partir de requisitos estáticos hasta la obtención de los datos requeridos y la comparación de los resultados. Se pueden aplicar el método de pruebas basado en modelos y la gestión inteligente de los datos de pruebas para automatizar todas y cada una de estas tareas, al tiempo que se facilita que numerosos equipos trabajen en paralelo sobre las mismas fuentes de datos.

El método de pruebas basado en modelos adelanta el trabajo de las pruebas de ETL, concentrando la mayor parte del esfuerzo en la etapa de diseño. A partir de ahí, todos los activos necesarios para la realización de pruebas de ETL se pueden derivar automáticamente en un corto período de tiempo. Se pueden generar casos de prueba que proporcionen una cobertura funcional del 100 % y se vinculan a resultados esperados definidos de forma independiente. Además, cada prueba se "asigna" también a los datos de origen exactos que son precisos para ejecutarla. Si dichos datos se guardan en un almacén de datos de pruebas, se cuenta con la opción de suministrarlos a numerosos equipos en paralelo.

Debido al estrecho vínculo creado entre las pruebas, los datos y los requisitos, tras producirse un cambio es posible ejecutar rápidamente las pruebas necesarias para volver a comprobar una rutina de ETL. Por lo tanto, el tiempo invertido en crear el modelo inicial queda rápidamente compensado por el tiempo que nos ahorramos al no tener que crear, actualizar y volver a ejecutar manualmente las pruebas. La generación basada en modelos ofrece la reusabilidad, otra ventaja sustancial, gracias a la cual los componentes de las pruebas se pueden preservar en el almacén de datos de pruebas como activos aptos para compartir, vinculados a los datos y los resultados esperados. Cuantas más pruebas ejecutemos, más crecerá la biblioteca, hasta que la comprobación de reglas de ETL nuevas o actualizadas se convierta en una tarea tan fácil y rápida como una mera selección entre los componentes existentes.

La realización de pruebas de ETL ya no provoca un cuello de botella en la entrega de aplicaciones y es capaz de seguir el ritmo al que crecen los negocios impulsados por datos. Se conserva la posibilidad de poner a prueba rutinas cada vez más complejas, de modo que las pruebas pueden manipular la variedad y el volumen de datos recopilados, sin que se imposibilite la entrega continua de aplicaciones de calidad.

Comuníquese con CA Technologies en ca.com/es

CA Technologies (NASDAQ: CA) crea software que impulsa la transformación de las empresas y les permite aprovechar las oportunidades que brinda la economía de las aplicaciones. El software se encuentra en el corazón de cada empresa, sea cual sea su sector. Desde la planificación hasta la gestión y la seguridad, pasando por el desarrollo, CA trabaja con empresas de todo el mundo para cambiar la forma en que vivimos, realizamos transacciones y nos comunicamos, ya sea a través de la nube pública, la nube privada, plataformas móviles, entornos de mainframe o entornos distribuidos. Para obtener más información sobre nuestros programas de éxito de clientes, visite **ca.com/customer-success**. Para obtener más información, visite **ca.com/es**.

1 Jean-Pierre Dijcks, *Why does it take forever to build ETL processes?*, consultado el 24/07/2015 en **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*, consultado el 21/07/2015 en **http://data-informed.com/the-state-of-etl-extract-transform-and-load-technology/**

3 IBM, consultado el 20/07/2015 en **http://www-01.ibm.com/software/data/bigdata/what-is-big-data.html**

4 Jacek Becla y Daniel L. Wang, Lessons Learned from managing a Petabyte, P. 4. Consultado el 19/02/2015 en **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?*, consultado el 24/07/2015 en **https://blogs.oracle.com/datawarehousing/entry/why_does_it_take_forever_to_bu**

7 ETL Guru, *ETL Startegy to store data validation rules*, consultado el 22/07/2015 en **http://etlguru.com/?p=22**

8 Bender RBT, *Requirements Based Testing Process Overview*, consultado el 05/03/2015 en **http://benderrbt.com/Bender-Requirements%20Based%20Testing%20Process%20Overview.pdf**

9 Huw Price, *Test Case Calamity*, consultado el 21/07/2015 en **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?*, consultado el 06/03/2015 en **http://www.softwaretestingclass.com/why-testing-should-start-early-in-software-development-life-cycle/**

12 datagaps, *ETL Testing Challenges*, consultado el 24/07/2015 en **http://www.datagaps.com/etl-testing-challenges**

13 Robin F. Goldsmith, *Four Tips for Effective Software Testing*, consultado el 20/07/2015 en **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*, consultado el 24/07/2015 en **http://blog.aditi.com/data/etl-how-to-handle-bad-data/**

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

CS200-196717_0416