LIVRE BLANC | AVRIL 2016

Tests ETL entièrement automatisés : guide détaillé

ca technologies

Section 1

Rôle essentiel du processus ETL dans une organisation moderne

Depuis son avènement dans le monde du stockage de données et de l'aide à la décision, le processus ETL (Extract, Transform, Load - Extraction, transformation, chargement) est devenu un composant essentiel de la planète logicielle. Comme son nom le suggère, le processus ETL compte trois étapes distinctes, qui se déroulent souvent en parallèle : les données sont extraites à partir d'une ou de plusieurs sources de données ; elles sont converties pour obtenir l'état requis ; elles sont chargées dans la cible souhaitée, généralement un entrepôt, un magasin ou une base de données. Une routine ETL évoluée inclut aussi souvent une infrastructure de journalisation, pour le traitement des erreurs, ainsi qu'un environnement de routine[1].

Jusqu'à présent, le processus ETL était utilisé avant tout pour préparer des données volumineuses et hétérogènes en vue de leur analyse et dans le cadre d'une aide à la décision. Toutefois, son usage s'est maintenant élargi au-delà du simple déplacement de données, avec notamment la migration de données pour les nouveaux systèmes, une application de plus en plus courante, et le traitement de l'intégration, du tri et de la fusion de données[2].

Le processus ETL est aujourd'hui une fonctionnalité clé du cycle de vie de développement moderne, en rapide évolution, au cours duquel de multiples versions sont développées en parallèle à un instant T. Les organisations doivent être capables d'améliorer et d'intégrer leurs logiciels, ainsi que d'innover en permanence, en mettant à disposition des testeurs et des développeurs des données à l'état approprié, pour chaque version et itération. Les données sont tirées des mêmes sources, mais doivent être transformées pour répondre aux exigences spécifiques de chaque équipe. Cela est tout particulièrement important lorsque l'organisation tente d'adopter une approche « Agile » ou de réussir la mise en œuvre de la livraison continue.

Le rôle critique du processus ETL dans la livraison continue a été parfaitement illustré lors d'une mission de CA Technologies auprès d'une banque multinationale. Cette banque était en pleine phase d'acquisition et devait effectuer la migration des clients, des produits et des outils financiers de la banque acquise dans son infrastructure existante. Cela impliquait la récupération, la conversion et la validation de 80 fichiers d'insertion, avant de les charger dans les systèmes back-end de la banque. Ces données devaient ensuite être mises à disposition pour 47 projets distincts menés en parallèle, tout en préservant leur intégrité référentielle. Dans ce cas précis, le processus ETL était essentiel pour obtenir le parallélisme nécessaire au succès d'une livraison continue.

Cependant, malgré l'utilisation et l'importance grandissantes du processus ETL, les tests ETL reflètent l'état des processus de test dans leur ensemble, à savoir qu'ils sont trop lents et trop manuels, tout en laissant passer un nombre inacceptable d'erreurs en production. Le présent dossier explorera les défis rencontrés dans une approche classique des tests ETL. Une approche alternative, globalement orientée modèle, sera ensuite proposée, avec pour objectif final d'optimiser l'efficacité des tests ETL et de les rendre systématiques.

Section 2

Approche classique des tests ETL et les défis généralement rencontrés

Lors de la validation des règles de transformation ETL, les testeurs créent généralement un ensemble de code fantôme, s'en servent pour transformer les données, puis comparent les résultats obtenus aux résultats attendus. Habituellement, le script ETL ou SQL est copié manuellement dans les données sources puis exécuté, et les résultats sont enregistrés. Le même script est ensuite copié dans les données cibles, et les résultats sont enregistrés. Les deux ensembles de résultats (réels et attendus) sont alors comparés afin de confirmer que les données ont été correctement transformées.

La problématique initiale : complexité et testabilité

La problématique sous-jacente à ces processus de validation manuels est que les routines ETL, de par leur nature, deviennent rapidement extrêmement complexes. À mesure que l'activité de l'entreprise se développe, et que la diversité et le volume des données recueillies s'étoffent, les règles ETL se développent en parallèle pour pouvoir les prendre en charge. Dans ce que l'on appelle aujourd'hui « l'ère de l'information », cette croissance se produit à un rythme trop rapide pour permettre un suivi par les méthodes de test traditionnelles. De fait, le volume absolu d'informations collectées par les organisations orientées données a augmenté si rapidement que 90 % des données dans le monde ont été collectées au cours des deux dernières années seulement[3], tandis que le volume de données moyen collecté par une organisation double chaque année[4].

La complexité des systèmes conçus pour collecter, transférer, exploiter et présenter ces données s'est accrue de manière exponentielle, avec chaque décision ajoutée. Cela inclut les règles ETL, mais de nombreux facteurs peuvent avoir un impact sur la complexité des transformations :

- Le nombre et la diversité des sources de données impliquées, y compris les types de base de données relationnelle et non relationnelle, ainsi que les fichiers plats
- Le nombre et la diversité des cibles de données
- Le nombre et la diversité des transformations simples et complexes, des recherches simples aux unions actives et à la normalisation
- Le nombre de transformations, extraits de code et liaisons réutilisables
- Le nombre de tables créées[5]

Tous ces facteurs sont exacerbés par l'orientation actuelle vers des solutions en temps quasi réel, et les complications que cela entraîne[6].

La documentation ne suffit pas

Cette complexité croissante a un impact direct sur la testabilité des routines ETL. Elle est particulièrement problématique pour les tests ETL, car les règles de transformation sont généralement stockées dans une documentation de piètre qualité, peu explicite quant aux résultats attendus. Les règles sont souvent conçues lors de la phase même de développement et fréquemment consignées dans des documents écrits, des pages de tableur ou, pire encore, n'ont jamais été couchées sur papier par les développeurs et testeurs[7]. Dans ce cas, il n'existe aucune documentation réelle à partir de laquelle dériver en toute confiance les scénarios de test (ex., code fantôme).

Une équipe d'aide à la décision avec laquelle nous avons travaillé consignait les exigences sous forme de documents écrits, tandis que les scénarios de test étaient consignés dans des feuilles de calcul. Cette documentation statique, véritables blocs de texte indigestes, ne permettait pas d'identifier facilement les étapes logiques des routines ETL. En outre, ces documentations n'illustraient que des scénarios optimistes et n'intégraient aucune condition négative, ce qui excluait de facto près de 80 % de la logique possible à tester dans un système standard. Face à ces documentations aussi incomplètes et ambiguës, il était impossible aux testeurs de comprendre facilement ou avec exactitude les routines ETL.

Trop souvent, ils devaient eux-mêmes combler ces manques, mais s'ils se trompaient, des erreurs étaient introduites dans les routines ETL. Des données non valides étaient alors copiées dans la cible, même si le code et les scénarios de test reflétaient une interprétation plausible de la documentation relative aux exigences.

« Garbage in, garbage out (GIGO) » – Dérivation manuelle des scénarios de test et résultats attendus

56 % des anomalies et erreurs qui parviennent en production sont imputables à l'ambiguïté de la documentation de définition des exigences[8]. Ce chiffre est en partie dû au fait que les scénarios de test et les résultats attendus sont dérivés manuellement d'une documentation de mauvaise qualité : un processus extrêmement fastidieux qui entraîne généralement à son tour une couverture de test médiocre.

Qualité

La dérivation manuelle est un processus ad hoc et non systématique, qui entraîne généralement la création de tous les scénarios de test possibles et imaginables qui passent par l'esprit des testeurs. Au vu de la complexité des routines ETL, évoquée précédemment, combinée à la mauvaise qualité de la documentation proposée, il serait injuste d'attendre des testeurs, même les plus talentueux, de pouvoir générer tous les tests nécessaires pour valider les combinaisons de données possibles. Par exemple, si un système simple de 32 nœuds et 62 arcs a été conçu de manière linéaire, il peut y exister pas moins de 1 073 741 824 voies logiques possibles.

La dérivation ad hoc entraîne donc un véritable problème de sous-test et de sur-test, où seulement une fraction de la logique possible d'une routine ETL est testée. Les tests négatifs sont particulièrement délicats et les scénarios de test, comme la documentation, sont souvent centrés presque exclusivement sur les cas de figure optimistes. Toutefois, ce sont bien les valeurs hors normes et les résultats imprévus qui doivent faire l'objet de tests, car il est impératif que les routines ETL rejettent ces « données erronées ».

Une société de services financiers avec laquelle CA Technologies a travaillé, par exemple, utilisait seulement 11 scénarios de test, pour une couverture totale de tout juste 16 %. Ce chiffre est assez standard et nos audits ont permis de déterminer que la norme d'une couverture de test fonctionnelle se situe entre 10 et 20 %. Dans le cadre d'un autre projet, cette même société présentait un sur-test de facteur 18, en raison du cumul de scénarios de test visant à tester la totalité du système, sans toutefois obtenir une couverture optimale. L'exécution de ces 150 scénarios de test supplémentaires par un fournisseur extérieur leur a coûté 26 000 $[9].

La conséquence d'une couverture de test aussi médiocre est que des défauts sont intégrés au code, où leur résolution est bien plus longue et coûteuse ; des études ont ainsi démontré que la correction d'un bogue lors des phases de test coûtait entre 40 et 1 000 fois plus en termes de ressources[10] et prenait 50 fois plus de temps[11] que la même correction lors des phases précédentes. Pire encore, des bogues pourraient passer inaperçus et des données non valides seraient copiées dans le produit en temps réel, menaçant l'intégrité du système. En outre, dans le cas d'une documentation statique, les testeurs ne disposent d'aucun moyen fiable pour mesurer la couverture de leurs scénarios de test ; ils ne peuvent tout simplement pas déterminer avec certitude quelle proportion de leur routine est testée, ni hiérarchiser les tests en fonction de leur criticité.

Du temps et du travail : impossible de suivre le rythme

Rédiger des scénarios de test à partir d'une telle documentation est également très coûteux en termes de temps et de travail. Dans l'exemple précédent, il a fallu six heures pour créer onze scénarios de test, tandis que la tendance galopante de sur-test a pris encore plus de temps. Ce temps perdu à la conception manuelle des scénarios de test se cumule au temps qu'il faut ensuite passer à comparer les résultats réels aux résultats attendus.

Comparer les vastes champs individuels aux résultats attendus est une tâche extrêmement longue et fastidieuse, au vu du volume de données produit par une routine ETL complexe, et du fait que les données sources sont souvent stockées dans une multitude de fichiers et bases de données. C'est également une tâche très difficile, car les données transformées doivent être validées à plusieurs niveaux :

- Les testeurs doivent vérifier l'exhaustivité des données, en s'assurant que le volume de données sources et le volume de données cibles correspondent.
- L'intégrité des données doit être confirmée, en vérifiant que les données cibles sont cohérentes avec les données sources.
- La transformation doit satisfaire aux règles métier.
- La cohérence des données doit être garantie, en identifiant tout doublon imprévu.
- L'intégrité référentielle doit être préservée, en identifiant tout enregistrement orphelin ou clé étrangère manquante[12].

Parfois, l'organisation choisit le compromis et ne valide qu'un ensemble de données échantillonné. Cependant, cela met en péril la rigueur des tests ETL, et par là-même la fiabilité des transformations. Au vu du rôle joué par de nombreuses routines ETL dans des opérations métier critiques, un tel compromis est inacceptable. La qualité est d'autant plus mise à mal par la nature même des comparaisons manuelles, propices aux erreurs, tout particulièrement lorsque les résultats attendus ne sont pas clairement définis ou, pire encore, ne sont pas définis indépendamment du code fantôme utilisé lors des tests. Dans pareil cas, les testeurs ont tendance à partir du principe que le test a réussi, à moins que les résultats réels soient particulièrement étonnants ; si aucun résultat attendu n'a été prédéfini, les testeurs ont tendance à supposer que les résultats réels correspondent aux résultats attendus[13], et ne pourront établir, avec aucune certitude, la validité des données.

La problématique des données

Jusqu'à présent, nous avons principalement parlé des problèmes rencontrés lors de la dérivation des tests (code fantôme), nécessaire pour valider les règles ETL. Cependant, une fois les scénarios de test dérivés, les testeurs ont besoin de données sources fictives à injecter dans le système. C'est là une autre cause fréquente de goulets d'étranglement et de défauts.

Disposez-vous de toutes les données nécessaires pour tester vos routines ETL complexes ?

Disposer d'un volume suffisant de données « non valides » est essentiel pour garantir l'efficacité des tests ETL, de même qu'il est crucial que, lors de son application, une règle ETL rejette ces données et les envoie à l'utilisateur intéressé, au format approprié. Si elles ne sont pas rejetées, ces données non valides généreront vraisemblablement des anomalies, voire une panne globale du système.

Dans ce contexte, le terme « données non valides » peut être défini de différentes façons, en fonction des données qui doivent être validées par les testeurs. Il peut s'agir des données qui, sur la base des règles métier établies, ne doivent jamais être acceptées ; par exemple des valeurs négatives dans un panier d'achat, en l'absence d'un bon d'achat. Il peut s'agir de données mettant en péril l'intégrité référentielle d'un entrepôt, par exemple des données obligatoires ou interdépendantes manquantes, ou bien des données manquantes parmi les données entrantes proprement dites[14]. Les données de test injectées via une règle de validation ETL doivent donc contenir toute la palette requise de données non valides, afin de permettre une couverture fonctionnelle de 100 %.

Nombreuses sont les organisations qui ne fournissent pas aux équipes de test ces données parmi les sources de données de production. Cela est dû au fait que les données de production sont tirées de scénarios métier optimisés, basés sur des situations passées, et donc conçues à la base pour exclure ce type de données non valides. Elles ne contiennent pas de résultats imprévus, de valeurs hors normes ou de conditions limites, pourtant nécessaire aux tests ETL. Elles sont plutôt orientées sur le fonctionnement optimal et optimiste. Nos audits sur les données de production ont permis d'établir que celles-ci offraient en moyenne une couverture de 10 à 20 %. L'ironie est que, plus la routine créée est efficace, moins le volume de données non valides passé entre les mailles du filet sera important et donc moins les données mises à disposition par la suite seront variées pour tester entièrement les règles ETL.

Les données sont-elles disponibles lorsque vous en avez besoin ?

Une autre problématique majeure de la validation ETL est la disponibilité des données. Les données sources peuvent être récupérées dans 50 sources différentes, sur l'ensemble de l'entreprise. Le problème des tests ETL, et des tests en général, est qu'ils sont considérés comme une suite d'étapes linéaires, et les équipes de test doivent attendre les données pendant que celles-ci sont utilisées par une autre équipe.

Prenons l'exemple d'une chaîne de migration dans le domaine bancaire, où les données sont récupérées auprès d'une banque, puis converties dans l'autre système, à l'aide d'un outil de rapprochement. À chaque étape, les données doivent être validées afin de vérifier qu'elles ont été correctement converties dans le cadre de contrôle financier, que le numéro de compte a bien été récupéré, que les données étaient correctes heure après heure, etc. Ce processus peut passer par des phases distinctes, de la saisie basique à la déduplication et à la préparation, en passant par la propagation et la réservation des données. Plusieurs équipes peuvent être impliquées, notamment des équipes ETL et non-ETL, particulièrement celles travaillant sur le mainframe.

Si les données provenant de toute l'entreprise ne sont pas mises à disposition des équipes en parallèle, les retards s'accumuleront tandis que les équipes passeront leur temps à attendre. Les testeurs rédigeront leur code fantôme, mais ne disposeront pas des données sources nécessaires pour valider une règle ETL, car celles-ci seront utilisées par une autre équipe. En réalité, nous avons observé que les testeurs passaient en moyenne 50 % de leur temps à attendre, à rechercher, à manipuler ou à créer des données. Cela peut représenter jusqu'à 20 % du temps de cycle (SDLC) total.

Que se passe-t-il lorsque les règles changent ?

Dériver manuellement les scénarios de test et les données à partir d'exigences statiques est une procédure qu'il est très difficile de changer de manière réactive. Les routines ETL changent aussi vite que l'activité de l'entreprise évolue. Le volume et la diversité des données qu'elles collectent augmentent en conséquence. Lorsque des changements constants de ce type ont lieu, cependant, les tests ETL ne peuvent pas suivre le rythme.

La principale cause de retard d'un projet dans cette situation est sans aucun doute la nécessité de vérifier et de mettre à jour les scénarios de test existants lorsque les routines changent. Les testeurs n'ont aucun moyen d'identifier automatiquement l'impact d'un changement apporté aux exigences statiques et aux scénarios de test. Ils doivent donc vérifier manuellement chaque scénario de test existant, sans moyen pour confirmer que la couverture reste la même.

Pour l'équipe d'aide à la décision mentionnée précédemment, où les exigences et les scénarios de test étaient conservés dans des documents écrits et des feuilles de calcul, respectivement, tout changement était particulièrement problématique. Un testeur avait besoin de 7,5 heures pour vérifier et mettre à jour l'ensemble de scénarios de test, en cas de changement au niveau d'une seule règle ETL. Dans une autre organisation avec laquelle nous avons travaillé, il fallait à deux testeurs deux journées complètes de travail pour vérifier chaque scénario de test existant, en cas de modification apportée aux exigences définies.

Section 3

L'alternative viable : des tests ETL entièrement automatisés

Il apparaît clairement maintenant que, tant que la dérivation des scénarios de test et la comparaison des résultats seront effectuées manuellement, les tests ETL ne pourront pas suivre le rythme de l'évolution constante des exigences métier. Voici donc une stratégie possible pour améliorer l'efficacité des tests ETL. Cette stratégie orientée modèle et basée sur les exigences a été conçue pour transférer en amont le travail de test et apporter de la qualité au cycle de vie ETL dès le départ. Cette approche orientée modèle introduit l'automatisation à toutes les étapes des tests et du développement, tout en rendant les tests ETL ultra-réactifs face aux changements constants.

1) Commencer avec un modèle formel

Intégrer la modélisation formelle aux tests ETL offre un avantage fondamental : cela permet de transférer en amont le travail de test, afin que toutes les ressources ultérieures de test/développement puissent être dérivées du travail initial de mappage d'une règle ETL dans un modèle. Ce modèle formel devient la pierre angulaire d'une validation ETL entièrement automatisée.

Toutefois, la modélisation formelle aide aussi à résoudre les problèmes plus spécifiques abordés précédemment, à savoir l'ambiguïté et la non-exhaustivité des exigences. Elle aide à préserver la testabilité malgré la complexité sans cesse croissante des règles ETL, afin que les testeurs puissent comprendre rapidement, visuellement et avec précision la logique à tester. Ils peuvent ainsi déterminer facilement les données valides et non valides à intégrer pour tester entièrement une règle de transformation, et quels sont les résultats attendus pour chaque instance.

Un modèle d'organigramme, par exemple, permet de décomposer les documentations indigestes en sections plus compréhensibles. Il réduit le processus ETL à une logique de cause et d'effet, le mappant en une série de déclarations logiques, organisées en une hiérarchie de processus[15]. Chacune de ces étapes devient un composant de test, qui indique au testeur exactement ce qu'il doit valider. La modélisation des routines ETL sous forme d'organigramme élimine donc l'ambiguïté dans la documentation des exigences, dans le but d'éviter les 56 % de défauts qui en découlent.

À mesure que les routines ETL se complexifient, l'organigramme sert de point de référence unique. Par opposition aux diagrammes et aux documents écrits statiques, une logique complémentaire peut facilement être ajoutée à ce modèle. Qui plus est, l'abstraction des routines extrêmement complexes est possible grâce à une technologie de sous-flux, ce qui améliore encore la testabilité. Les composants de bas niveau peuvent être intégrés dans les flux principaux, notamment les nombreuses routines qui composent les ensembles ultra-complexes de règles ETL, qui peuvent être regroupées en un schéma unique, plus visuel.

Outre qu'elle réduit l'ambiguïté, la modélisation par organigramme aide également à lutter contre la non-exhaustivité dans la définition des exigences. Elle force le créateur du modèle à penser en termes de contraintes, de conditions négatives, de limitations et de conditions limites, en se posant la question : « Que se passe-t-il si j'élimine cette cause ou ce déclencheur ? ». Il doit donc systématiquement prévoir des scénarios négatifs régissant les tests négatifs, qui constituent la majeure partie de la validation ETL. Des algorithmes de vérification de l'exhaustivité peuvent également être appliqués, car le modèle formel est un schéma mathématiquement précis de la règle ETL.

Cela élimine les logiques manquantes, telles que les conditions « autres » ambigües, afin que les scénarios de test qui couvrent 100 % des combinaisons de données possibles puissent être dérivés. À noter cependant qu'il existera presque toujours plus de combinaisons qu'il ne sera viable d'en exécuter en tant que tests. Nous parlerons plus tard des techniques d'optimisation. Autre avantage majeur : les résultats attendus peuvent être définis dans le modèle, indépendamment des scénarios de test. En d'autres termes, avec un organigramme, l'utilisateur peut définir le modèle afin d'inclure les entrées de limite et de propager les résultats attendus aux différents points terminaux du modèle. Cela définit clairement ce qui doit être rejeté et accepté par une règle de validation, afin que les testeurs ne supposent pas à tort que les tests ont réussi lorsque les résultats attendus ne sont pas explicites.

À noter qu'adopter des tests orientés modèle pour la validation ETL n'exige pas l'adoption globale d'une approche orientée exigences pour les tests et le développement à l'échelle de l'entreprise. Il n'est ainsi pas nécessaire de changer le fonctionnement global de l'organisation et, d'après notre expérience, il faut seulement 90 minutes pour modéliser une routine ETL en un organigramme « actif ». Ce modèle peut ensuite être utilisé par l'équipe ETL ou de test aux seules fins de tester et de retester la routine proprement dite.

2) Dériver automatiquement les scénarios de test à partir du modèle d'organigramme

L'introduction de tests orientés modèle permet d'automatiser l'une des principales tâches manuelles des tests ETL : la conception des scénarios de test. Le testeur n'a plus besoin d'écrire un code fantôme ni de copier manuellement des instructions SQL de la base de données source dans la cible. Au lieu de cela, les chemins de l'organigramme deviennent les scénarios de test, qui peuvent être utilisés pour injecter des données dans les règles de transformation. Ils peuvent alors être dérivés automatiquement, ce qui est impossible lorsque l'écriture du code se base sur des exigences statiques.

Cette dérivation automatique est possible car l'organigramme peut être mappé sur toute la logique fonctionnelle d'un système. Vous pouvez ensuite appliquer des algorithmes mathématiques automatisés, afin d'identifier tous les chemins possibles au sein du modèle, générant des scénarios de test couvrant toutes les combinaisons d'entrées et de sorties (une analyse des homotopes ou de la cause et de l'effet peut être utilisée pour ce faire).

Les scénarios de test étant liés directement au modèle proprement dit, ils couvrent l'ensemble de la logique qui y est définie. Ils offrent ainsi une couverture fonctionnelle de 100 %. L'utilisation d'un modèle d'organigramme pour obtenir une documentation complète équivaut donc à viser une couverture de test totale de la routine ETL. Autre avantage de cette méthode, les tests deviennent mesurables. Étant donné qu'il est possible de dériver chaque scénario de test possible, les testeurs peuvent déterminer avec exactitude quelle couverture fonctionnelle offre un ensemble de scénarios de test donné.

Optimisation : une couverture plus élevée avec moins de tests

Vous pouvez appliquer des algorithmes d'optimisation automatisés afin de réduire au minimum le nombre de scénarios de test, tout en conservant une couverture fonctionnelle optimale. Ces techniques combinatoires sont rendues possibles par la structure logique de l'organigramme, où une simple étape dans la logique de cause et d'effet (un bloc d'organigramme/ un composant de test) peut générer plusieurs chemins dans le flux. Un test exhaustif de la routine ETL revient ainsi à tester chaque bloc (opérateur), en utilisant l'une des multiples techniques d'optimisation existantes (Tous les arcs, Tous les nœuds, Tous les arcs d'entrée/de sortie, Toutes les paires). Un ensemble de trois scénarios de test, par exemple, peut ainsi être suffisant pour tester l'ensemble de la logique de cinq chemins.

Au sein de la société de services financiers mentionnée précédemment, cette approche s'est avérée extrêmement précieuse pour réduire le sur-test, raccourcir les cycles de test et améliorer la qualité des tests. Par exemple, en incluant le temps nécessaire pour modéliser l'organigramme, il a fallu 40 minutes pour créer 19 scénarios de test, pour une couverture globale de 95 %, par rapport aux 150 scénarios précédents offrant seulement 80 % de couverture pour un sur-test de facteur 18. Dans un autre projet, il a fallu deux heures pour créer 17 scénarios de test avec une couverture de 100 % : une amélioration spectaculaire par rapport à la couverture de 16 % auparavant obtenue en six heures de travail.

3) Créer automatiquement les données nécessaires à l'exécution des tests

Une fois les scénarios de test créés, les testeurs ont besoin de données pouvant couvrir 100 % des tests possibles, afin de les exécuter. Ces données peuvent également être directement tirées du modèle proprement dit, et créées automatiquement ou récupérées simultanément à partir de plusieurs sources.

Un moteur de génération de données synthétiques tel que CA Test Data Manager offre différentes options pour créer les données souhaitées, lors de l'utilisation de tests ETL orientés modèle. Cela est dû au fait que, outre sa logique fonctionnelle, l'organigramme peut également être mappé avec toutes les données impliquées dans un système. En d'autres termes, lors de la modélisation des règles ETL, les noms de sortie, les variables et les valeurs par défaut peuvent être définies pour chaque nœud. Lors de la création des scénarios de test, les données nécessaires pour leur exécution peuvent être automatiquement générées à partir des valeurs par défaut, en même temps que les résultats attendus pertinents.

Sinon, CA Agile Requirements Designer (anciennement Grid Tools Agile Designer) permet de créer rapidement des données système à l'aide de l'outil Data Painter. Cela offre une palette complète de fonctions de génération de données, de tables d'amorçage, de variables système et de variables par défaut. Tous ces éléments peuvent ensuite être utilisés pour créer des données couvrant tous les cas de figure possibles, y compris les données non valides et les chemins négatifs. Étant donné que chaque chemin correspond simplement à un autre point de données, il est possible de créer de manière entièrement synthétique toutes les données requises pour tester systématiquement la capacité d'une routine ETL à rejeter les données non valides.

Enfin, les données existantes peuvent être récupérées dans plusieurs systèmes back-end en quelques minutes, grâce à l'exploration automatique des données. Cette fonction utilise une analyse statistique pour identifier des tendances dans les bases de données, afin de pouvoir extraire des groupes de tendances, des dépendances ou des enregistrements inhabituels.

4) Provisionner les données en quelques minutes, en fonction des tests appropriés

Mieux vaut privilégier une combinaison de données de production existantes et de données synthétiques, lorsqu'une fonction de génération synthétique est utilisée pour obtenir une couverture de 100 %. L'important pour garantir l'efficacité des tests ETL est que les données de production (« Gold Copy ») soient stockées intelligemment, afin que les mêmes ensembles de données puissent être demandés, clonés et transmis en parallèle. Cela évite les retards causés par des contraintes de données.

La première étape pour stocker intelligemment les données est la création d'un magasin de données de test, dans lequel les données sont associées à des tests spécifiques (mise en correspondance). À chaque test sont affectées des données précises, tandis que les données sont associées à des critères stables et définis, et non des clés spécifiques. La mise en correspondance des données et des tests élimine le temps perdu à effectuer des recherches dans de vastes sources de données de production, car les données peuvent être récupérées automatiquement dans l'entrepôt des données de test, ou bien recherchées en quelques minutes dans de multiples systèmes back-end grâce à la fonction d'exploration.

L'entrepôt des données de test sert de bibliothèque centrale, dans laquelle les données sont stockées sous forme de ressources réutilisables, associées aux requêtes nécessaires pour les extraire. Il est ensuite possible de demander et d'obtenir des pools de données en quelques minutes, en association avec les scénarios de test et les résultats attendus correspondants. Plus le nombre de tests exécuté est important, plus la bibliothèque sera conséquente, jusqu'à ce que la quasi-totalité des demandes de données puisse être exécutée en quelques instants.

Autre élément crucial, les données peuvent être injectées dans plusieurs systèmes simultanément et sont clonées au moment du provisioning. Cela signifie que des ensembles de données provenant de multiples bases de données sources sont disponibles pour plusieurs équipes, en parallèle. Les tests ETL ne constituent plus alors un processus linéaire et les longs retards dus aux contraintes de données sont évités. Les données d'origine peuvent être conservées lorsque des modifications sont apportées au modèle, ce qui permet aux équipes de travailler sur plusieurs versions en parallèle. Le processus de contrôle de version implique également que les changements apportés aux routines ETL se reflètent automatiquement dans les données, offrant aux équipes de test les données à jour dont elles ont besoin pour tester les transformations de manière rigoureuse.

5) Exécuter les données sur la base des règles et comparer automatiquement les résultats

Une fois que les testeurs disposent des tests nécessaires pour tester une routine ETL de manière exhaustive, ainsi que des données nécessaires à leur exécution, la validation proprement dite doit aussi être automatisée si les tests ETL veulent pouvoir suivre le rythme des exigences changeantes.

Utilisation d'un moteur d'orchestration des données

L'une des méthodes possibles pour atteindre cet objectif est d'utiliser un moteur d'automatisation des tests. La solution CA Technologies offre l'avantage de se doubler d'un moteur d'orchestration des données, qui permet de récupérer et d'injecter via une règle de validation les données associées à des tests et des résultats attendus spécifiques. Il s'agit là d'une automatisation orientée données (Data-Driven Automation) où, par exemple, un atelier de test créé avec un moteur d'orchestration des données exécute chaque ligne d'un fichier XML, défini dans l'organigramme, en tant que test.

Cela permet d'obtenir un résultat réussite/échec sur la base des résultats attendus définis dans le modèle. Sont ainsi automatisées l'exécution des tests et la comparaison entre les résultats réels et les résultats attendus. Les testeurs n'ont plus à copier manuellement des scripts de la source de données dans la cible, et évitent également le processus fastidieux et propice aux erreurs d'une comparaison de chaque champ produit par la transformation.

Utilisation de CA Test Data Manager

Vous avez également la possibilité, une fois les résultats attendus définis, d'utiliser CA Test Data Manager pour créer automatiquement des rapports réussite/échec sur cette base. Cela automatise la comparaison des données, habituellement manuelle, mais il reste encore à copier les instructions SQL de la source dans la cible.

Tout d'abord, des données synthétiques sont définies dans le système source, à l'aide des diverses techniques présentées précédemment. Vous pouvez ensuite créer un pool de données pour simuler le processus ETL, en copiant les données de la source dans la cible. Un ensemble de données valide peut être copié, afin de vérifier qu'il n'est pas rejeté, ainsi qu'un ensemble de données comportant des erreurs, pour vérifier que les données non valides sont rejetées. En créant une autre table pour stocker les conditions de test et les résultats, et en créant une table basée sur un pool de données avec les scénarios de test et les conditions de données, vous pouvez comparer automatiquement les résultats réels et les résultats attendus. Cela permet d'identifier d'un coup d'œil toutes les données manquantes ou erronées.

6) Mettre en œuvre automatiquement les changements

L'un des principaux avantages des tests orientés modèle pour la validation ETL est la capacité à réagir efficacement face à des changements rapides. Du fait du lien étroit entre les scénarios de test, les données et les exigences, tout changement apporté au modèle est automatiquement reflété dans les scénarios de test et les données associées. Cette traçabilité permet aux tests de suivre le rythme de la complexification croissante des routines ETL, sans générer de goulets d'étranglement dans le pipeline de livraison des applications.

Avec la modélisation par organigramme, mettre en œuvre un changement est aussi rapide et facile que d'ajouter un nouveau bloc à l'organigramme. Vous pouvez ensuite appliquer des algorithmes de vérification de l'exhaustivité, afin de valider le modèle et de veiller à ce que chaque élément logique soit correctement intégré à un flux global. Si vous utilisez CA Agile Requirements Designer, l'outil Path Impact Analyzer peut servir à identifier l'impact des changements sur les chemins de l'organigramme. Les scénarios de test affectés peuvent alors être supprimés ou réparés automatiquement, avec la génération automatique des nouveaux tests requis pour préserver la couverture fonctionnelle de 100 %, le cas échéant.

Cela élimine le temps passé à contrôler et à mettre à jour manuellement les tests, alors qu'une automatisation de la déduplication offre une réduction avérée de cycles de test de 30 %. Au sein de l'équipe d'aide à la décision mentionnée précédemment, l'adoption de tests orientés modèle a permis des gains de temps très importants dans le processus ETL. Alors qu'il fallait auparavant 7,5 heures pour vérifier et mettre à jour les scénarios de test en cas de modification d'une seule règle ETL, 2 minutes suffisent désormais grâce à CA Agile Requirements Designer. Mieux encore, sur le nombre total de scénarios de test, seuls trois ont réellement été affectés et ont nécessité une mise à jour.

Comme nous l'avons expliqué précédemment, le contrôle de version des données permet également aux testeurs de recevoir les données à jour dont ils ont besoin pour tester entièrement une routine ETL, même en cas de modification. Du fait de la traçabilité des données de test via le modèle, lorsque les exigences changent, ces changements sont répercutés automatiquement sur les ensembles de données correspondants. Les données sont disponibles sur l'ensemble des versions en parallèle, tandis que les données nécessaires pour effectuer des tests de régression efficaces sont préservées dans l'entrepôt des données de test. Les données non valides rares ou intéressantes peuvent également être verrouillées, pour empêcher qu'elles soient utilisées par une autre équipe et pour éviter qu'elles ne se perdent durant une actualisation des données.

Section 4

Résumé

Mettre en œuvre un plus haut degré d'automatisation dans les tests ETL est essentiel pour toutes les organisations qui souhaitent effectuer la livraison continue d'applications logicielles de qualité. La validation ETL impose encore de lourdes tâches manuelles, qu'il s'agisse de l'écriture du code fantôme ou de la définition statique des exigences, en passant par la recherche de la source des données requises et la comparaison des résultats. Les tests orientés modèle et la gestion intelligente des données de test peuvent être utilisés pour automatiser chacune de ces tâches, tout en permettant aux nombreuses équipes de travailler sur les mêmes sources de données, en parallèle.

Les tests orientés modèle transfèrent en amont le processus de test ETL, concentrant le gros du travail lors de la phase de conception. Il est ainsi possible de dériver automatiquement et en un temps record toutes les ressources nécessaires à l'exécution des tests ETL. Vous pouvez générer automatiquement des scénarios de test offrant une couverture fonctionnelle de 100 % et les associer à des résultats attendus, définis de manière indépendante. Chaque test est ensuite associé à des données sources précises, qui sont nécessaires à son exécution, lesquelles peuvent être provisionnées à plusieurs équipes en parallèle si elles sont stockées dans l'entrepôt des données de test.

Du fait du lien étroit établi entre les tests, les données et les exigences, il est possible d'exécuter rapidement les tests nécessaires pour retester une routine ETL, en cas de modification. Le temps passé à la création du modèle initial est donc rapidement amorti grâce aux gains de temps obtenus en évitant d'avoir à créer, mettre à jour et réexécuter manuellement les tests. La génération orientée modèle offre également l'énorme avantage de composants réutilisables, où les éléments de test peuvent être stockés dans l'entrepôt des données de test en tant que ressources partageables, associées aux données et aux résultats attendus. Plus le nombre de tests exécutés est important, plus la bibliothèque sera conséquente, jusqu'à ce que le test des règles ETL nouvelles ou mises à jour devienne aussi simple et rapide que la sélection de composants existants.

Les tests ETL ne génèrent désormais plus de goulets d'étranglement dans la livraison des applications et sont à même de suivre le rythme de la croissance des entreprises orientées données. La testabilité de routines de plus en plus complexes est préservée, afin que les tests puissent prendre en charge la diversité et le volume des données collectées, sans entraver la livraison continue d'applications de qualité.

Restez connecté à CA Technologies sur ca.com/fr

CA Technologies (NASDAQ : CA) fournit les logiciels qui aident les entreprises à opérer leur transformation numérique. Dans tous les secteurs, les modèles économiques des entreprises sont redéfinis par les applications. Partout, une application sert d'interface entre une entreprise et un utilisateur. CA Technologies aide ces entreprises à saisir les opportunités créées par cette révolution numérique et à naviguer dans « l'Économie des applications ». Grâce à ses logiciels pour planifier, développer, gérer la performance et la sécurité des applications, CA Technologies aide ainsi ces entreprises à devenir plus productives, à offrir une meilleure qualité d'expérience à leurs utilisateurs, et leur ouvre de nouveaux relais de croissance et de compétitivité sur tous les environnements : mobile, Cloud, distribué ou mainframe. Pour en savoir plus sur nos programmes de succès client, rendez-vous sur le site **ca.com/customer-success**. Pour en savoir plus, rendez-vous sur **ca.com/fr**.

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

3 IBM, en date du 20/07/2015 via le site **http://www-01.ibm.com/software/data/bigdata/what-is-big-data.html**

4 Jacek Becla et Daniel L. Wang, Lessons Learned from managing a Petabyte, P. 4. en date du 19/02/2015, via le site **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?*, en date du 24/07/2015 via le site **https://blogs.oracle.com/datawarehousing/entry/why_does_it_take_forever_to_bu**

7 ETL Guru, *ETL Startegy to store data validation rules*, en date du 22/07/2015 via le site **http://etlguru.com/?p=22**

8 Bender RBT, *Requirements Based Testing Process Overview*, en date du 05/03/2015 via le site **http://benderrbt.com/Bender-Requirements%20Based%20Testing%20Process%20Overview.pdf**

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

12 datagaps, *ETL Testing Challenges*, en date du 24/07/2015 via le site **http://www.datagaps.com/etl-testing-challenges**

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

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

CS200-196717_0416