WHITE PAPER | | 2016 年 6 月

マスキングとサブセット化からの進化：テスト・データ管理の価値の実現

Huw Price
CA Technologies

ca technologies

目次

セクション 1

はじめに

TDM（Test Data Management、テスト・データ管理）のコンセプトは新しいものではありませんが、しばしば見過ごされ、過小評価され、誤解されていることがあり、TDM がビジネスにもたらす価値が完全には実現されていません。多くの組織やベンダにとって、TDM は単なる環境上の問題にすぎず、本番データのコピー、マスキング、および（場合によっては）サブセット化のみを意味します。こうして移行されたデータは、QA と開発環境で使用する「ゴールド･コピー」とみなされ、このコピーをより迅速に管理すること、そして新しいコピーのニーズに対応することが、TDM の効率的な利用の目的となっています。

合成データ生成機能は、使用されていても、独立したプロジェクト単位でしか使用されていません。こうした状況は多くの場合、データ生成は個々のチームやプロジェクトには役立つものの、企業全体からすれば有益なものではない、といった考えに裏打ちされています。現代の企業は通常、大規模で複雑なデータベースを所有しており、そこには数千万件のレコードがさまざまな形式で保存され、様々に異なるツールセットが使用されています。その理由は、実用的な合成データを作成するときのように複雑なデータをプロファイリングしてモデル化するよりも、既存の本番データをコピーする方がはるかに簡単だからです。

しかし、このような前提には疑問があり、優れた TDM を有効活用したいと考えている組織は、企業規模の合成データの生成を再評価すると共に、データの保存、管理、プロビジョニングする方法を見直すべきです。合成データの生成は、時間、品質、および資金面で効果があるだけでなく、本番データを完全にマスキングするよりも簡単かつ安全であることがしばしば証明されています。ただしそれには、適切なテクノロジ、プロセス、およびチームの構造改革が必要です。

本書では、組織でよく利用されるプロセス、テクノロジ、およびチーム構造を踏まえ、TDM の一般的な問題点と潜在的利益を環境の観点から比較対照します 1。

セクション 2

ロジスティクスのみに依存する TDM アプローチの欠点

テストおよび開発環境で本番データを利用している組織では、マスキングとサブセット化は必須の作業です。マスキングは現行のデータ保護規制を遵守するために必要であり、本番データのサブセット化は、実現不可能なほど高額なインフラストラクチャ・コストの削減に役立ちます。

ただし、TDM がマスキングとサブセット化のみに終始してしまうと、ビジネスにとっての TDM の真価は実現できません。このことは、2 つの前提条件に基づいて説明することができます。第一に、組織が収集したデータには個人を特定できる情報が含まれており、それを識別可能な形式で本番環境から持ち出すことは法律で禁じられています。第二に、マスキングされたデータはテストと開発を目的としているため、実行すべきすべてのテスト･ケース、ならびにどのような新機能にも対応できるよう、十分なデータ量が必要とされます。

これらの前提条件を突き詰めていくと、テストおよび開発環境の「目的にかなう」安全なデータをプロビジョニングする方法として、マスキングが実際には簡単でも効果的でもない理由はすぐにわかります。次に、一般的な組織で採用されているチーム構造、テクノロジ、プロセスを調べると、本番データの対象範囲が限られていることに加え、それらのデータが通常どのように管理されているかがわかります。これらのことから、マスキングとサブセット化だけに依存していると、プロジェクトでよく見られる問題点、すなわち本番での遅滞、予算超過、および欠陥が解消されないことは明らかです。

マスキングは単純なソリューションではない

まず、複雑な IT 資産においては、テストおよび開発環境用に本番データを安全にコピーし、マスキングするのがより簡単な方法である、という考え方には問題があります。実際には、照会先としての完全性を維持しながら本番データをマスキングする作業は、大抵の場合データのモデリングよりも負担が大きいうえ、効果的なマスキングを行うには、必ずデータを先にプロファイリングする必要があります。さらに、データの複雑な部分がしばしばマスキングされず、一種の欠陥として残っているため、マスキングは多くの場合、最も安全な方法であるとはいえません。

非本番環境で使用するためにデータをマスキングする場合は、たとえ機密情報がマスキングされていたとしても、元データのカラム間の関係性を維持する必要があります。単純なレベルでは、これは照会先としての完全性を維持することを意味します。しかし複雑なレベルでは、入り組んだカラムの関係性を維持することを意味します。たとえば、複数のカラムの合計を計算する［合計］フィールドがある場合、マスキングされたデータ内でもそれらのカラムが連携する必要があります。最も複雑なのが、時間データの相互関係と因果関係です。たとえば時間ベースの合計値などは、一律にマスキングするのがきわめて困難です。

さらに、これらの関係性はデータベース内（システム内）だけで一律にマスキングすればよいわけではありません。一般的な大企業は、複数のデータベース・タイプや複雑で異種混合のアプリケーションを利用しており、それら複数のデータベース間（システム間）でも関係性を維持する必要があります。データが複雑であるほど、関係性の維持が難しくなると同時に、相関付けする情報が増えるので不正利用されやすくなります。

データのマスキングではほとんどの場合、コンテンツに主眼が置かれ、システム間およびシステム内の関係性は、参照先としての完全性を維持する目的でのみ重視されます。コンテンツ、カラムの関係性、および時間的関係性を同時に維持することが難しい場合、いずれか一つが犠牲になります。たとえばカラム間の時間的な関係性は、未処理のまま残されることがよくあります。これは法的または技術的に機密度の高いコンテンツではありませんが、保存されたこれらの情報を使用して外部情報との相関付けが行われると、機密情報が識別される可能性があります。

マスキングされた一連のトランザクション・ログを例にとってみましょう。攻撃者は、ある人物が一度に一定量のトランザクションを作成したことを知っています。この情報は、マスキングされた機密扱いのコンテンツには含まれていませんが、マスキングされたデータベースには時間に関する情報が残っている場合があります。なぜならコンテンツ、数値の合計、およびトランザクション時間を一律にマスキングするのはきわめて難しいからです。システム間およびシステム内の完全性は維持されているため、機密情報が最初に識別されてしまうと、その情報はデータベースおよびシステム全体で識別可能になります。これにより、データの匿名性は事実上なくなってしまいます。

あらゆる複雑な手段を駆使してデータのプロファイリングとマスキングを行ったとしても、それらのデータが十分に安全であるとはいえません。なぜなら、情報はその後もさらに複雑な状態で保存されるからです。一例を挙げると、攻撃者は時間的手法、たとえばスナップショットを比較するなどしてシステム内とシステム間の情報、および因果関係を示す情報を結合し、トランザクション・ログの因果的カスケード効果を特定できます。その結果、運用中にデータがどのように変更されるかを推測したり、因果的カスケード効果（データベース上で実行されるトリガなど）があればそれを推測したりすることも可能になります。この場合も、マスキングされたデータには、最も脆弱なリンク程度の強度しかありません。この情報がひとたび特定されると、攻撃者はカラム間の関係性をまたいだ活動ができるようになり、業務上機密扱いの要件や、さらに厄介な場合は個人を特定できる情報が解読されてしまいます。

ほぼすべての組織のデータに何らかの個人情報が含まれていることを考慮すれば、データを非本番環境にプロビジョニングする方法として、マスキングは安全とも効率的とも言えません。企業に自社のデータをセキュリティで完全に保護する必要性が増大していることを考えればなおさらです。2013 年、データ侵害に対する平均的な罰金額は、13% 増加して 350 万ドルになりました（5、Ponemon、2014 年）。一方、EU General Data Protection Regulation（EU 一般データ保護規則）が新たに制定され、収集した目的以外での個人情報の使用を禁じている現行法の法的強制力が強化されます。

Symantec の「State of Privacy Report 2015」によると、欧州の消費者の 88% が、データ・セキュリティを考慮して買い物の場所や方法を決めており、マスキングされたデータを非本番環境で使用するリスクの増加は抑えられています。ここまでで、本番データをテストおよび開発環境へ移行する法的リスクが明らかになりましたので、その効果と効率性についてみていきましょう。

人材

データ依存関係

組織が TDM の専任チームを配置していることはめったにありません。代わりに、個々のチームがそれぞれのデータを管理、検索、作成しています。そのため各チームは独自のテストまたは開発環境で互いに独立して活動していますが、同じデータ・ソースを利用することも多々あります。このように一元管理と協力体制が不十分な状況では、データ依存による制約が生じ、1 つのチームがデータベースに変更を加えると、すべてのチームがその影響を受けることになります。その結果、明確な理由もわからないままテストに失敗し、チームは失敗の原因がコードの欠陥によるものかデータ・エラーであるのかを特定できないため、フラストレーションが蓄積し、作業の遅滞ややり直しを余儀なくされます。チームはテスト・データのバージョン管理やパラメータ化の能力も不足していることが多く、ほかのチームがデータベースを破損した場合、それをロールバックすることができません。

そのうえ、SDLC は逐次的な作業とみなされ、連続した一連の段階で構成されているため、1 つのチームが作業を終えたら次のチームにそれを引き渡します。その結果、「上流」での遅延によって、開発およびテスト・チームのプロジェクトが停滞することがよくあります。開発およびテスト・チームはデータが使用可能になるまで、あるいはほかのチームが仕様とデータの供給を終えるまで待機させられるため、SDLC の多くの時間がデータの待機に費やされます。

このような並行性の不備は、多くの組織が継続的デリバリを推進し、チームが SDLC のあらゆる段階でデータを必要としている現状に逆行しています。そのため、たとえばシステムの新バージョンで既存のデータを使いつつ、旧バージョンで耐用期間に関する開発を行う場合に、効率的な作業ができません。チームに新しい開発環境が与えられても、そこには必要なデータがありません。

テクノロジ

システムの依存関係

データ依存関係に加えて、ハードウェアとシステムの制約も、現代のテストおよび開発環境チームによく見られる問題です。アプリケーションは過去 20 年の間にどんどん複雑化し、組織の内外にあるほかのシステムとの依存関係は増大する一方です。こうした依存関係は、それによってサービスが不安定になったり使用できなかったりするため、システムをテストしたいときにはしばしば障害となります。使用したい環境に対してほかのチームが優先権を持っている場合や、テストで指定されたとおりのデータを取得できない場合もあり、チームは環境をフルに利用することができません。このような制約は、一見するとデータと直接関係がないように思われますが、その解決策として TDM インフラストラクチャがいかに重要であるかを後ほど説明します。

データ・ストレージ

現代の組織は大量の本番データを保存しています。これらのデータベースをコピーし、開発マシンで利用するには、費用も時間もかかります。こうしたデータの保存には、ハードウェア、ライセンス、およびサポートの費用など、高額なインフラストラクチャ・コストが発生します。サーバへの負荷も増大し続けています。なぜなら、サーバは各ジョブに対して大量のデータを一斉に供給すると同時に、接続を開いてファイルを公開し、データ挿入を処理する必要があるからです 2。

これらのコストは、データの管理方法を見直さない限り削減できる見込みはありません。平均的な業務で収集および保存されるデータの量は毎年倍増しており 3、「ビッグデータ」の到来は、組織がテラバイトではなくペタバイト単位での処理を求められていることを意味します。その結果、データ・ストレージは IT 予算の中で最も急速な増大を見せている項目の一つであり、近年のデータ・ストレージ業界は 20% もの年間成長率を達成しているとみられています 4。以上のことから、組織は本番データを個別にコピーする必要があるのかどうかを判断する必要があります。そして、単一のデータベースに対して複数のコピーを作成しがちな組織があることを考えれば、その答えはノーです。

データ・マイニングとプロファイリング

特にアジャイル開発のコンテキストや継続的デリバリのフレームワークにおいては、半自動または全自動テクノロジがなければ、十分な試験を実施したソフトウェアを予定どおりに提供することを目指しているチームは、データ検出で大きな困難に直面します。試験担当者は、作業時間の半分もの時間をデータ検出に費やすことがあり、そのために、コストのかかる欠陥をなくして本番へ投入するために必要なすべてのテストを実施することと、ソフトウェアを期限どおり提供することの間で、不本意な妥協を強いられます。

一元管理されたウェアハウスやリポジトリではなく、制御されていないスプレッドシートでデータがばらばらに保存されていれば（相互参照や索引作成などがほとんど、またはまったく機能しない）、状況はさらに悪化します。さらに、大抵の場合はデータベースが適切に文書化されておらず、組織にはテスト・データの属性や関連する DQL クエリをまとめた辞書がありません。したがって、データ・マイニングと割り当てが十分に機能せず、チームは標準テンプレートまたはフォーム、あるいはチームに必要な特定の基準に基づいてデータを要求することができません。その結果、チームは多くの場合、個々のテスト・ケースや要件に適した小さなデータ・セットを手作業で検索する必要があります。こうした作業は時間がかかりエラーの起こりやすいプロセスです。

データ・プロファイリングおよびマイニング用の自動化ツールの不備によって、コンプライアンス違反のリスクも増大します。制御されていないスプレッドシートにデータが保存されていると、データに対応するカラム・フィールドがない場合や、該当するすべてのフィールドがすでに一杯である場合、機密情報はどこででも（たとえば注釈カラムなど）見つかる可能性があります。所定のフィールドが検索できなければ、それらの情報が見つかる可能性は低く、したがって非本番環境に移動されることもあります。このような状況は、データはそれを収集した目的でのみ使用できることを定めたデータ保護規則に違反しています。これには平均 350 万ドルの重い罰金 5 が科せられるおそれがあり、会社の収益は大幅に低下する可能性があります。

プロセス

本番データは「ゴールド・コピー」ではない

テスト・ケースと要件に合致するデータを検索できなければ、非本番環境で本番データを使用するうえで大きな問題が生じます。つまり、組織が受け取る可能性のある大半のデータ要求、中でも開発環境からのデータ要求にまったく対応することができません。前述したように、テストおよび開発チームは開発のあらゆる段階でデータを必要とします。したがって、正しく設計された「ゴールド・コピー」データベースには、テストを繰り返し実施

するための標準的なデータ・セットと、あらゆるテストに対応するために必要なデータが格納されている必要があります。さらに、これまでの全データだけでなく「悪いデータ」も含めて、本番に類似した最新の状態である必要があります。本番データが満たす条件は 2 つだけ、すなわち本番に類似していることと、以前のデータをすべて格納していることです。したがって、本番データは「ゴールド・コピー」ではありません。

大半の本番データは非常によく似ており、「通常業務」のトランザクションに対応しています。また、本来の特性上、システム障害の原因となるデータを除外するよう構成されています。したがって、悪いデータは含まれておらず、テストの実行中、新しいシナリオやネガティブ・パスは無視されます。ただし、システム障害の一般的な原因は、予想外の結果、異常値、および境界条件であるため、開発とテストでは、通常は起こり得ないケース、不正なパス、予想外のシナリオを試験することを目的とすべきです。サブリング手法だけに依存していると、欠陥が本番に持ち込まれ、その修正に 1,000 倍 6、解決に 50 倍の時間がかかります 7。

さらに環境の変化や、IT チームへのビジネス・ニーズが絶えず変化することを考慮すると、本番からサンプリングされたデータはほとんどの場合、最新ではありません。データは環境と切り離して保存されることはないため、環境が変化した場合はシステムまたはバージョンの更新が必要です。これらの更新によって、複数の本番データ・ソースから作成されているデータやシナリオが破損し、有効なデータ・セットが失われる可能性があります。失われたデータは、単調な手作業によって再作成する必要があります。多くの場合、このようなシステムの更新は非常に時間がかかる傾向もあります。たとえば CA の提携組織では、システム更新のプロセスに最大 9 か月かかりました。

データを手動で作成すれば一時的には修正ができ、当面のテスト・ケースは実行できるようになります。しかし、こうしたデータは特定の要件やテスト・ケースを想定して作成されるため、すぐに使えなくなります。たとえば、為替レートや株価動向が良い例です。こうしたデータは日単位で古くなります。つまり、手動で作成されたデータは再利用できず、通常は「処分」されてしまいます。このため、テストのたびに新しいデータを苦労して作成する必要があり、それによってテストが繰り延べになったり、開発サイクルの次の段階まで据え置かれたりしている間に、プロジェクトの遅れが増長することになります。

複数のデータベースを効果的にマスキングするには、最初にデータのプロファイリングが必要であるが、それにもかかわらずデータのセキュリティや十分な品質は得られないとしたら、次のような疑問が生じます。組織はデータを正しくプロファイリングした後、なぜ合成によって必要なデータを作成しないのでしょうか。

セクション 3

理想的な代替策 : 人材、プロセス、テクノロジ

優れた TDM ポリシーには、企業規模のデータを管理するためのアプローチが採用されています。このアプローチは体系化され、次が非常に重要ですが、一元管理されています。このアプローチは、前述した問題点の多くを解決するだけではありません。適切なテクノロジと組み合わせれば、テストおよび開発環境にデータをプロビジョニングする方法として、多くの場合、より安価で効率的であり、さらには簡単でさえあります。

有益なデータを資産として環境の外で保存し、必要に応じて環境へ戻すことで、TDM の環境問題は解消されます。次に問題となるのが、データを転送する方法です。このデータ転送も、データ・ストレージを一元管理するアプローチによって効率的に実行できます。データは再利用可能な資産としてモデル化され、必要に応じて明確なサブセットとして抽出することができます。そこで、必須操作のマスキングとサブセット化だけで済ませずに合成データを生成すると、戦略的目標が明らかになります。この目標を広範囲な TDM ポリシーに盛り込むことで、十分な試験を実施したソフトウェアを予定通り、予算内で提供することが可能になります。

テクノロジ

自動データ・プロファイリング

すでに述べてきたように、本番データを効果的にマスキングするには、最初にデータのプロファイリングが必要ですが、プロファイリングを行ってもマスキングされたデータが完全に安全なわけではなく、テストや開発に必要なデータは提供されません。ただし、自動化テクノロジを利用すれば、複雑な IT 環境でのデータ・プロファイリングの作業を軽減できます。データをプロファイリングするには、まずそのデータを「登録」し、可能な限り多くのメタデータを収集する必要があります。メタデータにはテーブル名、カラム名、カラム・タイプなどが含まれます。非リレーショナル・データベースにもメタデータは存在し、メインフレーム・システムの場合はコピーブック、固定幅または区切り文字のフラット・ファイルの場合はマッピング・ドキュメントの形式をとります。

登録が完了すると、計算に基づくデータ検出アルゴリズムを適用できます。たとえば CA Test Data Manager（旧 Grid-Tools の Data Maker）では、まず個々のスキーマにアルゴリズムが適用され、個人を特定できる情報（PII）を識別し、必要に応じてデータベースの関係性をリバース・エンジニアリングします。この処理が完了すると、システムを相互に結合することができます。そのために、CA Test Data Manager ではキューブ・ビューを使用します。これにより、データに内在する最も複雑な関係性もプロファイリングし、多次元のデータ・セットを作成することができます。キューブの各次元はデータの属性を表しています。こうしたプロファイリングを行うことで、組織はどのようなデータがあり、それがどこに保存されているかを正確に把握し、さらには機能網羅率の差異を特定することもできます。

合成データの生成

存在するデータを正確に把握し、テスト環境に必要な追加データを特定すると、不足しているデータを自動的に生成できます。実際の各環境はもう一つのデータ・ポイントと考えられるので、多様な機能に 100% 対応したデータをモデル化し、作成することができます。このデータには、これまでに発生していない将来的なシナリオや、「悪いデータ」、異常値、予想外の結果も含まれます。これにより、有効なネガティブ・テストや、新システムまたはサブシステムの開発が可能になります。テストは体系的に実施できるため、試験者が想定していなかった結果やシナリオによってシステムに支障が生じることはなく、本番に持ち込まれる前に欠陥を検出できます。

テストを繰り返し実行できるだけの十分なデータがない場合は、自動化テクノロジを使用して大量のデータを作成することもできます。CA Test Data Manager は、RDBM または ERP API レイヤと直接連携する自動化ツールを備え、処理能力の許す限り高速でデータを生成します。大量のスクリプトにより、組織のデータ量はインフラストラクチャの処理能力と同じ速度で倍増します。つまりは、こうした自動化テクノロジによって、ネガティブ・テストも含めたテストの実行に必要なすべてのデータ、および繰り返しテストを実施できる十分な量のデータを使用して、本番に近いデータを高速で作成できるということです。

データ管理の一元化

これまで述べてきた技術面での改善により、本書が定義する「ゴールド・コピー」（「本番データはゴールド・コピーではない」を参照）を作成する手段の大半はすでに確立されています。そのうえで、再利用可能なオブジェクトとしてモデル化されたデータを中央のテスト・データ・ウェアハウスで保存すれば、テストおよび開発環境に適した特定のデータ・サブセットを、必要に応じてすばやく特定することも可能になります。

データ・クローニング

「テスト・マート」またはテスト・データ・ウェアハウスでデータがオブジェクトとしてモデル化され、データ資産の辞書と関連するクエリが設定されたら、特定のデータ・サブセットを指定してクローニングし、テストおよび開発環境に投入することができます。

たとえば CA Test Data Manager のデータ・クローニング・モジュールは、複数の相互に関連した本番および開発システムから、整合性のあるテスト・データの小規模なセットを抽出します。これにより、大規模で複雑なデータベースをコピーして移動するという、時間と費用のかかるプロセスを排除できます。必要なデータだけを抽出、コピー、および提供できるということは、膨大なサイズの本番データのコピーを維持する必要がなくなるということです。

データ・ウェアハウスを一元管理し、データをクローニングできれば、データのプロビジョニングと消費を分離して、チーム間のデータ依存関係を解消することもできます。これはつまり、データをクローニングして複数のチームに並行して配信できることを意味し、「上流」のデータが使用可能になるまで待機することで生じる遅延は解消され、データの変更によってチーム間に生じる悪影響も回避できます。

柔軟で再利用可能なオブジェクトとしてデータをモデル化し、1 か所で保存すると、バグや関心のあるシナリオを簡単に再構築することもできます。バグ・レポートでデータを明確に記述すれば、柔軟かつ高速なクローニング技術により、データが枯渇することなく、複雑で例外的なテストを繰り返し実行できます。データを結合する必要がなく、関心のあるデータ・セットが失われることがないため、データ更新の際には特に有益です。

ハードウェアの制約

合成データ生成機能は、仮想化ツールキットと組み合わせることで、ハードウェアやシステムの制限の解消にも役立ちます。システムのメタデータを使用してメッセージ・レイヤをシミュレーションし、現実に則したサービス・メッセージ（フラット・ファイルのほか、Soap、REST、MQ ファイルも含む）を正確にプロファイリングし、作成することができます。その場合は、仮想マシンで自動データ生成エンジンが動作し、要求 / 応答ペアなどの現実的なメッセージ応答が入力されます。

さらに、マシン全体を仮想化すると、複数の開発環境を作成することもできます。これはつまり、相互依存関係にあるコンポーネントがない場合でも、チームは複数の環境で作業でき、上流での遅延を回避すると同時に、コストのかかるレガシー・システムやハードウェアもテスト用に仮想化できることを意味します。

プロセス

並行開発と再利用性

特定のデータ属性に基づいてデータを検出し、クローニングできれば、環境の問題だけでなく、TDM のもう一つの懸案事項、すなわち個々の試験者またはテスト・チームにデータを効率よくプロビジョニングするという問題も解決できます。データは必要に応じて同時に要求、共有、および再利用できます。

CA Test Data Manager が提供しているような、一元化された Web ベースの「オンデマンド・テスト・データ」ポータルにアクセスすると、試験者および開発者は現在のタスクに必要なデータだけを正確に要求できます。特定の条件（テスト・データ属性）が提出されると、ポータルはバッチ・エンジンにジョブを送信し、バッチ・エンジンはバックエンド・システムで適切なデータを検索するか、データをクローニングして転送します。これにより、手動でデータを検索したり作成したりする必要がなくなるため、データ要求に対応する時間が大幅に短縮されます。

フォーム上の要求を標準化すると、チーム間で処理を再利用できるのでさらに効果的です。たとえば、以前に作成された合成データがある場合、その入力値をパラメータ化し、ポータルのドロップダウン・リストで表示されるようにすると、テスト・ケースが違っていてもすべての人がそのデータを要求できます。テスト・データのほか、データ挿入ハーネス、ユニット・テスト、仮想資産、自動スクリプトなども、保存して今後の作業の構成要素として使用できます。

バージョン・コントロール

優れたバージョン・コントロールは、新しいシステムを予定どおりに予算内で継続的に開発するために必要な並行処理を可能にします。たとえば CA Test Data Manager のテスト・データ・ウェアハウスでは、レポジトリからデータをコピーすることで、以前のバージョンを示すポインタと共にそのデータを「継承」できます。これにより、複数のバージョンにまたがるデータの進化に対応できます。特定のチームに対してデータをロックし、その間にデータを新旧のバージョンへ簡単に移動し、異なるバージョン間でデータを調整することができるからです。1 か所で変更が行われると、元データを維持したまま、その変更が新旧のバージョンに「反映」されます。たとえばあるチームが、データベース全体に新しいカラムを追加する必要があるとします。CA Test Data Manager では、ハードコーディングされた親がある場合は、それにリンクされているすべての子を検索してデフォルト値を設定したり、シーケンスや標準機能を使用してデータを生成したりすることができます。

人材

最後に、チームの構造改革は、上記の技術および手順的な改善をしばしば補うことができ、さらには前述した環境上の問題点の解消にも役立ちます。専任チームによる TDM の一元管理は、企業のニーズに効率よく対応できる中央データ・ストレージ、管理、およびプロビジョニング用のリソースを備えることと相関関係にあります。このチームは、データのプロビジョニングやデータそのものの管理、さらには必要に応じて新規データの作成とデータ・プロファイリングも行います。

これにより、チーム間のデータ依存関係による制約を回避するだけでなく、データ要求やバグ・レポートを 1 か所にまとめることもできます。そうすることで、品質ゲートを適用しながら、データのオーナーシップを IT セキュリティ・チームに一元化することができます。このような管理は、CA Test Data Manager の Test Data on Demand ポータルの動的フォーム作成機能でサポートされています。この機能では、役割ベースのアクセス権がさらに高度化されており、機密データはそれを要求した承認済みスタッフのみにプロビジョニングされます。

セクション 4

参考資料

1 非本番環境で実際に本番データを使用するうえでの問題点は、別の文書で取り上げられています。たとえば、「Reduce Time to Market with Test Data Management」および「How better Test Data Management is the only way to drive Continuous Delivery」（どちらも Huw Price 著）を参照してください。

2 Jacek Becla、Daniel L. Wang、「Lessons Learned from managing a Petabyte」4 ページ。2015 年 2 月 19 日、http://www.slac.stanford.edu/BFROOT/www/Public/Computing/Databases/proceedings/ から入手。

3「Lessons Learned from managing a Petabyte」

4 http://www.computerweekly.com/feature/Meeting-the-demand-for-data-storage

5 http://www.ponemon.org/blog/ponemon-institute-releases-2014-cost-of-data-breach-global-analysis

6 http://benderrbt.com/Bender-Requirements%20Based%20Testing%20Process%20Overview.pdf

7 http://www.softwaretestingclass.com/why-testing-should-start-early-in-software-development-lifecycle/

セクション 5

CA Technologies のメリット

CA Technologies（NASDAQ：CA）は、複雑な IT 環境の管理と保護に役立つ IT 管理ソリューションを提供し、アジャイル開発のビジネス・サービスを支援します。CA Technologies のソフトウェアと SaaS ソリューションを活用することで、データセンタからクラウドに至るまで革新を加速し、インフラストラクチャを変革し、データとアイデンティティを保護できます。CA Technologies はそのテクノロジにより、お客様が必要な成果と期待どおりのビジネス・バリューを実現できるようにします。お客様を成功に導くプログラムの詳細については、**ca.com/jp/customer-success** をご覧ください。CA Technologies の詳細については、**ca.com/jp** を参照してください。

セクション 6

著者について

約 30 年に及ぶキャリアを持つ Huw Price は、米国とヨーロッパの複数のソフトウェア会社で技術アーキテクトの責任者として、多国籍の銀行、大手公共事業および医療会社の高度なアーキテクチャ設計をサポートしてきました。長年にわたってテスト自動化ツールを専門とし、ソフトウェア業界で使用されるテスト・モデルの交代を促す多数の革新的な製品を発表し、QA Guild の「IT Director of the Year 2010」に選ばれています。現在は国際的に有名なイベントで講演を行うほか、「Professional Tester」や「CIO Magazine」をはじめとする多数の技術雑誌に執筆しています。

Price が最後に設立した Grid-Tools は、2015 年 6 月に CA Technologies に買収されました。同社は約 10 年間、大企業のテスト戦略アプローチの変革に努めてきました。Grid-Tools は、Price の明確なビジョンに基づくアプローチとリーダーシップによってデータ中心の強力なアプローチをテストに導入し、Price が発案した「データ・オブジェクト」、「データの引き継ぎ」、「中央のテスト・データ・ウェアハウス」などの新しいコンセプトを発表しています。

CS200-205127