KONTROLA APLIKACJI

WSTĘP

Kontrole aplikacji mogą być szerokie i wieloaspektowe. Biorąc pod uwagę złożoność wielu aplikacji, żaden pojedynczy system nie może zapewnić wszystkich potrzebnych elementów sterujących. W ramach projektowania nowej lub zmian w istniejącej aplikacji istnieje szereg środków kontroli, które należy wziąć pod uwagę, aby zapewnić ochronę podstawowych danych przed zagrożeniami dla sześciu podstawowych atrybutów informacji (poufność, kontrola, integralność, autentyczność, dostępność i użyteczność), jak omówiono w rozdziale 3 niniejszego podręcznika. Podobnie jak w przypadku wszystkich innych obszarów bezpieczeństwa informacji, strategia obrony w głąb kontroli aplikacji jest koniecznością. Na przykład, czy serwer WWW powinien być publicznie dostępny? Jeśli nie, wówczas kontrola ta musiałaby zostać uwzględniona w projekcie wniosku. Podstawy kontroli aplikacji dzielą się na trzy obszary:

1. Architektura i projektowanie systemów

2. Dane wprowadzane przez użytkownika i uprawnienia

3. Kontrola bazy danych i danych bazowych

Nowoczesne systemy charakteryzują się ogromną złożonością ze względu na zmiany w architekturze i presję kosztów na działy IT. Ze względu na tę złożoność istnieje możliwość posiadania tak wielu elementów sterujących, że ostatecznie będą one zastępować się nawzajem, aw niektórych przypadkach ograniczać ich użyteczność. Najłatwiej jest zaprojektować nową aplikację sieciową i wziąć pod uwagę wszystkie niezbędne kontrole, ale taka sama należyta staranność jest wymagana w przypadku zmian w istniejących aplikacjach, niezależnie od tego, czy dodawane są nowe funkcje, czy też zachodzą zmiany w procesach biznesowych, które mogą wymagać dużych zmian uprawnień. Wszystkie zmiany w aplikacjach muszą być zgodne ze ścisłą metodą projektowania, niezależnie od tego, czy jest to zwinna, kaskadowa czy inna metodologia. Należy zauważyć, że każda warstwa architektury odgrywa rolę w aplikacjach i każda warstwa musi mieć pewne kontrole. Chociaż ten rozdział koncentruje się na aplikacjach do przetwarzania danych, podobne problemy pojawiają się w systemach kontroli nadzorczej i akwizycji danych (SCADA) do sterowania procesami w czasie rzeczywistym.

ARCHITEKTURA I PROJEKTOWANIE SYSTEMÓW

Architektura i projekt stanowią pierwszą linię obrony aplikacji. Konfiguracja projektu i systemów jest najważniejsza. Ochrona i sposób, w jaki kontrole wpływają na aplikację, muszą być brane pod uwagę w podstawowej architekturze i projekcie. Formanty architektury można podzielić na kilka kategorii, ale obejmują one wszystkie poziomy systemu obsługującego aplikację. Są to standardowe najlepsze praktyki dotyczące systemów. Kwestie, które należy wziąć pod uwagę podczas projektowania kontrolek systemowych dla aplikacji, obejmują:

1. Jakie podstawowe procesy muszą działać w systemie w celu obsługi aplikacji? Zalecenie: Ogranicz procesy tylko do tych, które są wymagane.

2. Jakie porty muszą być otwarte, aby system działał poprawnie z obsługą aplikacji? Zalecenie: Ogranicz porty tylko do tych, które są wymagane.

3. Kto ma bezpośredni dostęp do systemu i dlaczego? Zalecenia: Ćwicz z najmniejszymi uprawnieniami i wymagaj specjalnego zatwierdzenia w przypadku wyższego ryzyka lub dostępu administracyjnego. Regularnie przeglądaj wszystkie dostępy, ale jeśli dostęp wiąże się z większym ryzykiem, może być wymagany częstszy przegląd i weryfikacja dostępu.

4. Jakie kontrole i procesy będą stosowane w celu wsparcia aplikacji? Podobnie, jakie kontrole będą stosowane w przypadku kontroli zmian lub łatania? Zalecenia: Zdefiniuj, śledź i weryfikuj procesy kontroli zmian dla systemów i aplikacji, których te systemy wymagają. Więcej informacji na temat zarządzania poprawkami można znaleźć w rozdziale 40 tego podręcznika.

5. W jaki sposób wzajemnie ze sobą współpracujące systemy są weryfikowane, aby upewnić się, że są tym, za kogo się podają? Zalecenia: Cyfrowe certyfikaty i dwukierunkowe SSL mogą być wykorzystane do zapewnienia ważności podstawowych systemów. Wiedzieć i rozumieć, jakie systemy mogą komunikować się z innymi systemami i dlaczego.

WKŁAD I UPRAWNIENIA

Z perspektywy tworzenia aplikacji największą kontrolą, którą należy wziąć pod uwagę, są dane wprowadzane przez użytkownika. Dane wprowadzane przez użytkownika dzielą się na dwa odrębne obszary. Jakie prawa lub uprawnienia ma lub powinien mieć użytkownik i jakie są ograniczenia dla danych wejściowych, które użytkownik może mieć? Wymagania biznesowe aplikacji zazwyczaj determinują te specyfikacje. Przykładem może być pole wprowadzania adresu. Firma może zastrzec, że pole ma 50 znaków i że dostęp do wprowadzania, zmiany lub usuwania pola mają tylko określone typy użytkowników. Pytania dotyczące uprawnień, które należy zadać, obejmują:

* Co użytkownik lub grupy użytkowników mogą robić w aplikacji?

* Co użytkownik powinien mieć możliwość przeglądania, tworzenia, aktualizowania lub usuwania?

* Czy istnieją określone funkcjonalności, których dany użytkownik lub grupy użytkowników nie mogą wykonywać ze względu na swoją obecną rolę w organizacji?

* Z perspektywy sprawozdawczości należy zadać te same pytania.

W ramach projektowania aplikacji na wszystkie te pytania należy odpowiedzieć w oparciu o wymagania biznesowe. Z drugiej strony każde pole wejściowe w ramach tych uprawnień musi być zdefiniowane w określonych granicach, a te ograniczenia muszą być przenoszone od początkowego wejścia do leżącej u jego podstaw struktury danych. Zarówno aplikacja, jak i bazowa baza danych mogą kontrolować limity i wprowadzane dane. Kontynuując powyższy przykład, 50-znakowe pole adresowe powinno zawierać tylko 50 znaków. Długość pola powinna wynosić 50 i musi zostać zakończona walidacja danych wejściowych, aby upewnić się, że wprowadzane są tylko prawidłowe lub zatwierdzone znaki. Kontrola ta może być oparta zarówno na poziomie pola aplikacji, jak i bazy danych, w której będą przechowywane dane. Ograniczenie długości danych wejściowych jest kluczową obroną przed atakami iniekcyjnymi i przepełnienia bufora. Zwykle sprawdzanie poprawności danych odbywało się bliżej wprowadzania danych przez użytkownika po prostu ze względu na szybkość sieci, a ostatnio sprawdzanie poprawności użytkownika w czasie rzeczywistym zapewnia użytkownikowi lepsze wrażenia (np. sprawdzanie poprawności podczas wprowadzania pola w porównaniu z przyciskiem „prześlij" formularza internetowego). Najlepsza praktyka wymaga sprawdzania poprawności danych wejściowych na serwerze przed ich przetworzeniem. Wstępne kontrole można przeprowadzić na formularzu internetowym lub w polu w aplikacji komputerowej, ale walidacja po stronie serwera jest lepsza, ponieważ jest bliższa systemom wewnętrznym. Systemy mogą akceptować dane wejściowe zarówno od użytkowników, jak i od systemów; na przykład proces wsadowy może pobierać dane wejściowe z jednego źródła, manipulować nimi i przenosić je do innego źródła. Dane wejściowe, czy to wprowadzone przez użytkownika, czy przez system, muszą zostać zweryfikowane. Kontrola bazy danych działa jako wsparcie kontroli aplikacji. Jeśli na przykład w aplikacji nie występowała walidacja na poziomie pól, podstawowa baza danych mogłaby służyć do ochrony przed złymi danymi. Podobnie

jak w powyższym przykładzie, jeśli walidacja na poziomie pola i walidacja po stronie serwera nie powiodła się, pole bazy danych można zdefiniować jako pole 50-znakowe. Większość systemów baz danych będzie obcinać i ignorować dodatkowe dane wprowadzane w polu.

Wejście

Jak wspomniano powyżej, wszystkie dane wejściowe muszą zostać oczyszczone i zweryfikowane, aby upewnić się, że odbierane i przetwarzane są tylko akceptowalne wartości. Dotyczy to zarówno danych wprowadzanych przez użytkownika, jak i danych wprowadzanych przez systemy. Tej samej myśli należy użyć w perspektywie bazy danych. Oczywiście ta metoda wiąże się z pewnymi wyzwaniami. Jeśli wszystkie 50-znakowe pola adresów muszą zostać zaktualizowane, aby akceptowały 60, wymagane są wielokrotne zmiany i dokładniejsze testy. Organizacje mogą zdecydować, że dane aplikacji nie są krytyczne i zaprojektować je z mniejszą kontrolą. Może to prowadzić do problemów po prostu dlatego, że osoba atakująca może użyć aplikacji z mniejszą liczbą kontrolek jako punktu przeskoku, aby dostać się do aplikacji z bardziej wrażliwymi danymi. Nie można wystarczająco podkreślić znaczenia sprawdzania poprawności danych dla wszystkich danych wprowadzanych przez użytkownika. Wielu atakom na systemy można było zapobiec, gdyby przeprowadzono odpowiednią i wszechstronną walidację danych (np. wielu atakom typu cross-site scripting (XSS) i cross-site request forgery (XSRF) [zobacz rozdział 16 tego podręcznika] można zapobiec dzięki walidacji). Nieakceptowanie niektórych znanych złośliwych znaków, kodowanie niektórych znaków, blokowanie niektórych kombinacji ciągów i zrozumienie wektorów ataku aplikacji są ważne dla zapewnienia bezpieczeństwa aplikacji i zachowania integralności. Projektanci aplikacji muszą współpracować ze swoimi partnerami biznesowymi i wyjaśniać problemy z dopuszczaniem pewnych typów i sekwencji znaków. Podobnie jak w przypadku wszystkich środków kontroli bezpieczeństwa informacji, potrzebnych jest wiele poziomów kontroli. Nie jest właściwe tworzenie aplikacji, w których sprawdzanie poprawności danych wejściowych odbywa się tylko na poziomie pola. Aby zapewnić szczegółową strategię obrony dla aplikacji, należy użyć wielu kontrolek.

Uprawnienia użytkownika

Szczególnie zniechęcające są uprawnienia użytkowników do aplikacji. Definiowanie i ograniczanie dostępu przy jednoczesnym umożliwieniu użytkownikom wykonywania wymaganych zadań może być ogromnym wyzwaniem. Kontrole aplikacji są najważniejsze przy rozważaniu, co użytkownik może zrobić w aplikacji. Uprawnienia są również ważne przy rozważaniu podziału obowiązków w ramach tej samej aplikacji, ponieważ użytkownicy zmieniają role w organizacji, a nawet jaki dostęp, jeśli w ogóle, miałby programista do środowiska produkcyjnego. Prawa dostępu użytkownika to uprawnienia nadawane użytkownikom lub grupom do wykonywania funkcji w aplikacji. Na przykład użytkownik, który powinien mieć dostęp tylko do odczytu danych, nie powinien otrzymywać uprawnień do konta umożliwiającego aktualizacje. Dobry projekt wykorzystuje kontrolę systemów, aplikacji i baz danych, aby umożliwić lub ograniczyć użytkownikom wykonywanie różnych funkcji. Jeśli rolą osoby w organizacji jest programista, czy należy pozwolić jej na migrację kodu do środowiska produkcyjnego? Najlepsze praktyki mówią nam, że nie powinny; jednak każda organizacja jest inna, a zespoły programistyczne mogą również migrować kod do środowiska produkcyjnego. Ważną rzeczą jest:

1. Dowiedz się, gdzie przyznawane są uprawnienia, komu i dlaczego.

2. Cyklicznie weryfikuj uprawnienia.

3. Z perspektywy programistycznej wymień uprawnienia jako wymagania w planie testów i zweryfikuj, czy testy zakończyły się pomyślnie.

Praktykuj najmniejszy przywilej

Najlepszą praktyką jest stosowanie jak najmniejszych uprawnień do uprawnień użytkowników, ponieważ powinno to być stosowane we wszystkich innych obszarach technologii w celu zachowania kontroli nad danymi i procesami. Sposób, w jaki bazowy system jest zbudowany, wielkość organizacji i kontrole regulacyjne będą dyktować wykonalność osiągnięcia stanu najlepszych praktyk. W wielu przypadkach wymagania biznesowe nie uwzględniają, które konkretne typy użytkowników powinny mieć dostęp do jakich funkcji. Ten brak jest nie do utrzymania, po prostu dlatego, że aplikacje ciągle się zmieniają, dodawane są nowe funkcje, aktualizowana jest starsza architektura i uwzględniane są nowe technologie. Należy dokładnie rozważyć przyznanie użytkownikom uprawnień wymaganych do wykonywania ich funkcji i kontroli. Konieczne jest również przeprowadzenie kompleksowych testów, aby upewnić się, że nie występuje pełzanie dostępu podczas wprowadzania zmian w aplikacji.

Rozwiązanie problemu

W przypadku mniejszych aplikacji można łatwo zdefiniować różne wymagania biznesowe za pomocą funkcji, które powinny mieć dostęp. Prosta macierz może zrobić na przykład:

Business Function	Business Role	Technology	Application or database role
Add new account	Phone representative	Insert into certain tables	App: phone rep DB: phone rep
Update account	Phone rep (with review) or data input associate	Update certain tables	App: update account DB: update account
Confirm phone representative	Phone rep manager	Read certain tables Update certain tables	App: phone rep manager DB: phone rep manager

Po zdefiniowaniu wymagań biznesowych i umieszczeniu ich w macierzy znacznie łatwiej jest zarządzać istniejącymi funkcjami i wszelkimi przyszłymi zmianami oraz zrozumieć, gdzie funkcje krytyczne mają nadawać tylko wymagane uprawnienia.

BAZY DANYCH I PODSTAWOWE KONTROLE DANYCH

Technologia baz danych umożliwia szczegółową kontrolę informacji za pomocą kontroli dostępu opartej na rolach (RBAC), jak omówiono w rozdziale 9 niniejszego podręcznika. Ponadto bazy danych zapewniają wielopłaszczyznowe podejście do kontroli aplikacji. Nie tylko można posegregować wszystkich użytkowników w zależności od potrzeb biznesowych, ale projekt bazy danych może zapewnić pewien poziom kontroli za pomocą projektu i układu tabeli. Dobry projekt i użytkowanie bazy danych może pomóc w ochronie bezpieczeństwa danych. Relacyjne bazy danych są oparte na podstawowym modelu teoretycznym autorstwa E. F. Codda, opracowanym w latach 70. XX wieku i szeroko stosowanym obecnie oraz w dającej się przewidzieć przyszłości. Systemy zarządzania relacyjnymi bazami danych (RDBMS) to zestawy programów, które udostępniają administratorom baz danych (DBA) narzędzia do wykonywania następujących zadań:

* Twórz struktury baz danych (układy plików lub tabel, ekrany lub formularze).

* Wprowadź informacje do struktur.

* Ustal odsyłacze między plikami lub tabelami.

* Manipuluj (sortuj, indeksuj i podsumowuj) informacje w strukturach.

* Importuj informacje z nierelacyjnych struktur baz danych i eksportuj informacje do struktur innych niż bazy danych. Pozwala to na interfejsy między aplikacjami korzystającymi z RDBMS i aplikacjami wykorzystującymi konwencjonalne struktury plików

* Narzędzia repozytorium danych i słownika danych. Narzędzia te dokumentują bazy danych i mogą zawierać informacje opisowe (metadane) dotyczące wszystkich zasobów zawartych w środowisku systemów informatycznych. Słowniki danych mogą obsługiwać standardy dokumentacji. Wspólne struktury plików lub tabel, wspólne moduły programów i wspólne definicje pól znacząco przyczyniają się do zmniejszenia zamieszania i przerw w komunikacji w przedsiębiorstwie

* Bazy danych przenoszą większość dynamicznie zmieniających się wymagań dotyczących formatów danych i edycji danych z kodu źródłowego programów aplikacyjnych do łatwych do zlokalizowania i edytowania pojedynczych kryteriów. Zamiast zmuszać programistów do wyszukiwania i modyfikowania wielu wystąpień ograniczeń danych, administratorzy baz danych mogą współpracować z programistami w celu zmiany poszczególnych kryteriów, mając pewność, że wszystkie programy korzystające z RDBMS pomyślnie wykonają zmiany.

Wszystkie te narzędzia pomagają utrzymać bezpieczeństwo i integralność bazy danych. Istnieją dwie podstawowe metody organizowania plików i baz danych pod kątem dostępu i przetwarzania: wsadowe i online. Pliki wsadowe, które można zapisać w różnych językach JCL, wykonują instrukcje bez dalszej interwencji człowieka po ich uruchomieniu, podczas gdy metody online obejmują interakcję człowieka z programami w wielu punktach. Ochrona plików online i baz danych wymaga dodatkowego planowania podczas projektowania systemów korzystających z nich oraz specjalnych środków ostrożności podczas ich używania. Ochrona plików wsadowych jest prostsza, ponieważ tworzenie kopii zapasowych jest nieodłączną częścią procesu wsadowego. Jak omówiono wcześniej, walidacja danych przed wprowadzeniem informacji do bazy danych musi zostać zakończona niezależnie od tego, czy wpisu dokonuje użytkownik, czy system. Należy dołożyć wszelkich starań, aby dane były wprowadzane lub przesyłane i zatwierdzane szybko. Brak poprawnych danych w systemie szybko opóźnia tworzenie i wyświetlanie informacji, co jest głównym celem prawie wszystkich systemów. Gdy niewłaściwa uwaga zostanie skierowana na sprawdzanie poprawności danych wprowadzanych do systemu, system tworzy nieprawidłowe informacje. Akronim GIGO (garbage in, trash out) od dawna jest częścią narodowego leksykonu języka potocznego. Niewłaściwe sprawdzanie poprawności i dopuszczanie śmieci do bazy danych może mieć dalszy wpływ na inne aplikacje lub procesy korzystające z danych. Walidacja na wszystkich poziomach aplikacji ma kluczowe znaczenie.

Uszkodzenie danych

Uszkodzenie danych oznacza nieprawidłowe dane — naruszenie integralności danych. Uszkodzenie może wystąpić z powodu czynników fizycznych lub błędów logicznych w programach. Dobre kontrole aplikacji nakazują ochronę baz danych i ich podstawowej architektury przed uszkodzeniem danych. Dane mogą zostać uszkodzone przez problemy ze sprzętem, błędy w oprogramowaniu lub podczas równoczesnego dostępu dwóch lub więcej użytkowników lub procesów.

Fizyczne uszkodzenie danych

Uszkodzenie fizyczne występuje w wyniku awarii lub innych awarii sprzętu, takiego jak komputery i sprzęt sieciowy, zwłaszcza urządzeń pamięci masowej, takich jak dyski magnetyczne, taśmy i kasety lub napędy optyczne. Do uszkodzenia danych podczas transmisji może dojść w wyniku zakłóceń elektromagnetycznych kabli komunikacyjnych lub szumu o częstotliwości radiowej, który wpływa na transmisję bezprzewodową. Kable światłowodowe są podatne na przesłuchy i zakłócenia spowodowane fizycznymi zagięciami lub załamaniami kabli; dyski optyczne są podatne na zabrudzenia

i ścieranie; napędy dysków magnetycznych mogą zostać uszkodzone przez fizyczne wstrząsy (upuszczenie, intensywne wibracje); dyski półprzewodnikowe nie radzą sobie dobrze po zanurzeniu w płynie. Oprócz problemów z nośnikami transmisyjnymi lub pamięciowymi, nieprawidłowe ustawienia lub skutki zużycia sprzętu mogą powodować błędy. Przykłady obejmują niewspółosiowość głowic magnetycznych na dyskach i taśmach; złe styki w sprzęcie do transmisji bezprzewodowej, powodujące szum; i niewłaściwego pozycjonowania laserów w ośrodkach optycznych. Fizyczne uszkodzenie zazwyczaj pokazuje dane zapisane niewłaściwie w blokach, a nie w pojedynczych polach zdefiniowanych przez aplikacje. Takie fizyczne uszkodzenie zwykle nie ma związku z uszkodzonymi zapisami poza fizyczną bliskością dysku; dlatego główną oznaką fizycznego uszkodzenia jest uszkodzenie plików z zupełnie innych aplikacji — na przykład blok rekordów bazy danych sąsiadujący z blokiem plików tekstowych.

Uszkodzenie danych logicznych

Uszkodzenie logiczne występuje w wyniku błędów programistycznych, takich jak nieprawidłowe sumy, złe formuły arytmetyczne, nieprawidłowe warunki logiczne, błędne dane w tabelach przeglądowych, warunki poza zakresem umożliwiające odczyt i zapis w niewłaściwych obszarach pamięci, brak sprawdzania poprawności danych i skutki działania złośliwego oprogramowania. Uszkodzenie logiczne zwykle pokazuje błędy w tym samym polu w wielu rekordach. Inna klasa błędów logicznych pokazuje złe wartości dla warunków brzegowych (np. najmniejszą lub największą możliwą wartość), ale prawidłowe dane w zakresie. Takie błędy rzadko przekraczają granice aplikacji, chyba że istnieje logiczny związek między uszkodzonymi plikami; na przykład błąd w arkuszu kalkulacyjnym (prawie zawsze spowodowany błędami wejściowymi lub programowymi popełnionymi przez użytkownika) może rozprzestrzenić się do plików tekstowych, jeśli do wstawienia wyników arkusza kalkulacyjnego do dokumentu zostanie użyte łączenie obiektów i osadzanie. Inne dokumenty pozostaną nienaruszone przez takie błędy logiczne.

Podsystemy zarządzania bazami danych

W latach sześćdziesiątych XX wieku złożone systemy, takie jak te oparte na zapisach księgowych, zmuszały programistów do definiowania własnych struktur plików, aby reprezentować relacje między danymi; na przykład plik nagłówka zamówienia byłby połączony z odpowiednimi rekordami szczegółów zamówienia poprzez zakodowane na stałe relacje w programach aplikacyjnych. Każdy programista lub zespół programistów musiał zdefiniować własne ścieżki dostępu do danych i jawnie je zakodować. Koordynacja dostępu do wielu indywidualnie nazwanych plików przyprawiała programistów o ból głowy. Na przykład łatwo było popełnić błędy, takie jak zapomnienie o zwiększeniu liczników reprezentujących liczbę rekordów szczegółowych odpowiadających rekordowi głównemu (indeksowi) (np. liczba wierszy w nagłówkach zamówień niezgodna z rzeczywistą liczbą wierszy szczegółowych). Ponieważ w takich systemach nie było żadnej szczególnej ochrony plików, łatwo było przez pomyłkę zastąpić lub usunąć poszczególne pliki, co doprowadziło do ogromnego uszkodzenia logicznego. Usunięcie rekordów nagłówkowych może pozostawić fragment niedostępnych szczegółów (sieroty) w powiązanych plikach. Programy musiały aktualizować wskaźniki w celu tworzenia łańcuchów do przodu i do tyłu. Kopie zapasowe czasami kończyły się niepowodzeniem z powodu błędu operatora, więc nie wszystkie powiązane pliki zostały uwzględnione. Co więcej, każdy system miał swoje własne, unikalne metody zarządzania danymi, co powodowało problemy z utrzymaniem i uczeniem się. Pod koniec lat sześćdziesiątych wiele dużych sklepów programistycznych definiowało swoje metody dostępu do danych za pomocą procedur bibliotecznych, z których wszyscy programiści mogli współdzielić, ale nadal istniały duże inwestycje w poznanie nowych zasad za każdym razem, gdy programista zmieniał pracę. Po opublikowaniu modelu relacyjnych baz danych E. F. Codda na początku lat 70. w dziedzinie programowania nastąpił gwałtowny wzrost wdrażania podsystemów zarządzania bazami danych, w

których interfejs do bazy danych kontrolował funkcje użytkowe, takie jak indeksowanie, wskazywanie i tworzenie łańcuchów. Przy odpowiedniej konfiguracji typowy DBMS chroni wszystkie pliki, zwane zestawami danych, przed przypadkowym skasowaniem i wymuszałby, aby wszystkie odczyty, zapisy, dołączenia i blokady były pośredniczone przez procedury DBMS. Zwykli użytkownicy nie powinni mieć dostępu do bazowej infrastruktury bazy danych — tylko do danych za pośrednictwem DBMS. DBMS zapewnia kontrolowany dostęp do danych, a także zazwyczaj zapewnia narzędzia do tworzenia kopii zapasowych, aby zapewnić, że wszystkie zestawy danych zostaną skopiowane razem i zapobiec przypadkowemu przywróceniu niewłaściwej wersji zestawu danych. Wreszcie, DBMS zwykle zapewnia funkcje rejestrowania w celu przechowywania rejestrów różnych rodzajów dostępu do bazy danych. Wszystkie te funkcje zapewniają kontrolę nad danymi. Jedną z najważniejszych zasad egzekwowanych przez system DBMS jest integralność referencyjna zapobiegająca uszkodzeniu typowych danych logicznych, automatyczne ograniczenia unikalności w celu wykluczenia duplikatów i konfliktów rekordów oraz blokowanie w celu zapewnienia bezpiecznego współbieżnego dostępu.

* Więzy integralności. Integralność referencyjna w projekcie DBMS zapewnia, że każdy zależny rekord ma wartość klucza podstawowego, która odpowiada istniejącemu kluczowi podstawowemu w pliku głównym.

* Na przykład w bazie danych zamówień kluczami podstawowymi są numery zamówień w pliku nagłówkowym (często nazywanym głównym zamówieniem).

* Każdy rekord nagłówka zamówienia zawiera unikalne informacje o zamówieniu, takie jak numer klienta, data złożenia, łączna cena materiałów, podatki i koszty wysyłki.

* Plik szczegółów zamówienia zawiera zależne rekordy, z których każdy można zlokalizować, używając jego numeru zamówienia jako klucza podstawowego.

* Każdy rekord szczegółowy zawiera informacje o określonej części odpowiedniego zamówienia, takie jak numer pozycji, zamówiona ilość, cena, cena rozszerzona oraz specjalne opłaty lub rabaty.

* Jeśli rekord nagłówka zamówienia ma zostać usunięty, należy najpierw usunąć wszystkie rekordy szczegółów zamówienia; w przeciwnym razie rekordy szczegółowe nie miałyby możliwości zlokalizowania ich za pomocą wartości klucza podstawowego.

* Podobnie nie można dodać żadnego rekordu szczegółowego, chyba że istnieje już rekord główny z tym samym kluczem podstawowym.

* Ograniczenia wyjątkowości. Nowoczesny DBMS umożliwia konfigurację niepowtarzalnych kluczy podstawowych; na przykład w bazie danych zamówień numer zamówienia byłby zazwyczaj niepowtarzalnym lub unikalnym kluczem, ponieważ nigdy nie powinny istnieć dwa zamówienia z tym samym numerem identyfikacyjnym. Ustawienie właściwości unikalności uniemożliwiłoby dodanie drugiego rekordu nagłówka o tej samej wartości w polu numeru zamówienia, co inne zamówienie.

Rozproszone bazy danych

Rozproszona baza danych to taka, która jest przechowywana w różnych bazach danych na różnych platformach komputerowych. Bazy danych mogą być rozproszone w wielu witrynach przy użyciu kilku różnych architektur. Najprostszym i najbardziej podatnym na awarie jest pojedynczy serwer bazy danych, który przechowuje bazę danych i udostępnia jej zawartość w kilku sieciach lokalnych. Za każdym razem, gdy serwer lub baza danych jest w trybie offline lub podczas awarii, wszyscy klienci tracą dostęp do bazy danych. Tworzenie kopii zapasowych i odzyskiwanie bazy danych na jednym serwerze odbywa się zgodnie z procedurami opisanymi wcześniej. Drugą architekturą jest serwer

replikowanej bazy danych, w którym na wielu serwerach znajdują się zduplikowane kopie bazy danych. Ten typ systemu umożliwia klientom dostęp do informacji z dowolnej z kilku kopii bazy danych. W środowisku replikowanego serwera klienci mogą nadal mieć dostęp do bazy danych, gdy jeden z serwerów jest w trybie offline. Takie zreplikowane systemy mają zazwyczaj dodatkową zaletę polegającą na możliwości równoważenia ruchu transakcyjnego, tak aby żaden serwer nie był przeciążony. Kompromisem jest proces synchronizacji bazy danych, który zwiększa złożoność architektury replikowanego serwera. Chociaż tworzenie kopii zapasowych i odzyskiwanie rozproszonej bazy danych może przebiegać zgodnie z procedurami opisanymi wcześniej, jest to skomplikowane ze względu na wymaganie synchronizacji. Trzecia architektura jest znana jako serwer partycjonowanej bazy danych, w której określone podzbiory bazy danych znajdują się na co najmniej dwóch serwerach baz danych. Na jednym serwerze może znajdować się marketingowa baza danych, na drugim — baza danych księgowości i finansów, a na trzecim — baza danych inwentaryzacji. Ponieważ na ogół nie jest możliwe posiadanie wzajemnie wykluczających się podzbiorów baz danych, synchronizacja musi nadal być częścią tej architektury. Tworzenie kopii zapasowych i odzyskiwanie partycjonowanych baz danych wymaga zastosowania procedur opisanych powyżej dla każdego z podzbiorów. Istnieje inna rozproszona architektura znana jako stowarzyszone serwery baz danych. Ta architektura jest używana do obsługi baz danych na dwóch lub więcej serwerach, składających się z zwykle niekompatybilnych modeli przechowywania, takich jak modele hierarchiczne i relacyjne obsługiwane przez różne systemy DBMS. W architekturze federacyjnej pojedyncza ujednolicona definicja bazy danych lub schemat jest tworzona i przechowywana na połączonym serwerze bazy danych. Ten serwer działa jako interfejs między aplikacjami a bazami danych rezydującymi na innych serwerach. Zapytania i inne transakcje są wysyłane do połączonego serwera bazy danych, który tłumaczy je na zapytania i transakcje do podstawowych baz danych. Następnie odpowiedzi wracają z podstawowych baz danych do ujednoliconego schematu w celu sformułowania odpowiedzi dla użytkownika. Procedury tworzenia kopii zapasowych i odzyskiwania, takie jak te opisane powyżej, są używane w bazowych bazach danych. Chociaż architektura stowarzyszonej bazy danych może być złożona i kosztowna w utrzymaniu, może być mniej złożona i tańsza niż proces obsługi aplikacji dla każdego bazowego systemu DBMS. Architektura federacyjna jest często wykorzystywana przy wdrażaniu hurtowni danych, które służą do wydobywania informacji z wielu wewnętrznych i zewnętrznych baz danych w celu wspierania podejmowania decyzji zarządczych.

Kontrole bazy danych w celu zachowania integralności

Jednoczesny dostęp do danych może powodować problemy z integralnością, chyba że dostęp jest szeregowany, aby zapewnić, że jeden proces na raz uzyskuje dostęp do danego zasobu.

Zablokuj aktualizację

Gdy więcej niż jeden użytkownik uzyskuje dostęp do bazy danych, mogą wystąpić konflikty dotyczące korzystania z określonych rekordów. Klasyczny przykład występuje w bazie danych inwentaryzacji, gdzie w inwentarzu znajduje się 15 jednostek części 1.

* Użytkownik Albert musi usunąć z inwentarza pięć jednostek części 1, pozostawiając w sumie 10. Program inwentaryzacji odczytuje rekord inwentaryzacji części 1 i modyfikuje rekord 1, aby pokazywał tylko 10 jednostek.

* Jeśli jednak w tym czasie użytkownik Betty potrzebuje trzech jednostek części 1, rekord inwentarza nadal pokazuje 15 dostępnych jednostek, ponieważ użytkownik Albert nie zaktualizował jeszcze tego rekordu.

* Po zakończeniu aktualizacji programu użytkownika Albert rekord pokazuje dostępnych 10 jednostek, ale po tym, jak program użytkownika Betty nadpisuje ten rekord, aby pokazać 12 dostępnych jednostek dla części 1, suma zapasów jest błędna o pięć jednostek.

Aby uniknąć tego rodzaju uszkodzeń logicznych, system DBMS zapewnia funkcje blokowania części bazy danych. W przykładzie dotyczącym inwentaryzacji program użytkownika A zablokowałby rekord inwentaryzacji dla części 1 (lub całego zbioru danych inwentaryzacji) do czasu zakończenia aktualizacji. W ten sposób program użytkownika B musiałby czekać, aż program użytkownika A odblokuje dane zanim będzie można działać na rekordzie inwentarza. Oczywistym objawem złej strategii blokowania jest rozbieżność między wartością w bazie danych a wartością w świecie rzeczywistym. Jednak taka rozbieżność sama w sobie nie jest dowodem logicznej korupcji, ponieważ ta sama rozbieżność może wynikać ze zdarzeń w świecie rzeczywistym, które nie są odzwierciedlone w bazie danych. W przykładzie z inwentarzem rzeczywisty inwentarz mógł zostać zmniejszony w wyniku kradzieży lub powiększony przez niezarejestrowane dodanie materiałów.

Bezwarunkowe a warunkowe blokowanie

Istnieją dwa rodzaje strategii blokowania: blokowanie warunkowe i blokowanie bezwarunkowe.

* Blokowanie warunkowe próbuje uzyskać blokadę, ale jeśli wymagany rekord lub cały zestaw danych jest już zablokowany, DBMS zwraca kontrolę do programu wywołującego ze wskaźnikiem stanu tego warunku. Program aplikacyjny można następnie zapisać w pętli, aż do uzyskania blokady.

* Żądanie bezwarunkowej blokady zawiesza program do momentu przyznania blokady. DBMS lub system operacyjny zapewnia automatyczne kolejkowanie przy użyciu kolejki pierwsze weszło, pierwsze wyszło.

Zakleszczenia

Bezwarunkowe blokowanie niesie ze sobą ryzyko, jeśli wiele zasobów jest blokowanych przez programy.

* Na przykład, jeśli program A bezwarunkowo zablokuje zasób 1, a program B zablokuje zasób 2, problem wystąpi, gdy program A spróbuje zablokować zasób 2, podczas gdy program B spróbuje zablokować zasób 1.

* Żaden program nie zwolni zasobu, który zablokował, dopóki ten nie zostanie zwolniony, więc oba będą czekać w nieskończoność lub do momentu, gdy jeden z programów zostanie przymusowo zakończony.

* Taka sytuacja jest znana jako impas lub bardziej barwnie jako śmiertelny uścisk.

* Jeśli programiści nalegają na stosowanie bezwarunkowego blokowania, strategia zapobiegania impasowi polega na zapewnieniu, że wszystkie programy uzyskujące dostęp do bazy danych muszą blokować zasoby w tej samej kolejności (np. następnie odblokować 1).

Inne strategie polegają na utrzymywaniu transakcji tak krótko, jak to możliwe i unikaniu konieczności interakcji operatora, które utrzymywałyby rekordy zablokowane przez długi czas.

Zatwierdzenie dwufazowe

Czasami wiele rekordów lub zestawów danych musi zostać zablokowanych, aby można było ukończyć złożone transakcje. Na przykład w szpitalnej bazie danych systemów klinicznych wypisanie pacjenta może wymagać modyfikacji w zestawach danych, takich jak pacjent-główny, szczegóły leczenia, główny

i szczegóły przypisania pielęgniarki, główny i szczegółowy przydział lekarza oraz zestawy danych dla funkcji finansowych. Zablokowanie wszystkiego, co może być potrzebne, i czekanie, aż człowiek wprowadzi wszystkie odpowiednie dane, może zająć od kilku sekund do kilku minut, podczas których wszystkie rekordy, których to dotyczy, byłyby zablokowane i niedostępne dla wszystkich innych osób w systemie. W skrajnych przypadkach, gdyby operator opuścił w trakcie transakcji, inni użytkownicy mogliby zostać zablokowani w dużych częściach bazy danych na nieokreślony czas, nawet na kilka godzin. Opóźnienia wynikające z takiego blokowania interwencji człowieka doprowadziły do zasady, że żadna transakcja nie może być blokowana wokół interwencji człowieka. Innym problemem związanym z blokowaniem interwencji człowieka jest to, że awaria systemu może zakończyć przetwarzanie, gdy baza danych jest w niespójnym stanie. Na przykład w przypadku inwentaryzacji DBMS mógł zaktualizować szczegóły zamówienia, dodając element, ale nie zaktualizował jeszcze nagłówka zamówienia, aby pokazać nowy całkowity koszt zamówienia. Aby zmniejszyć prawdopodobieństwo wystąpienia takiego zdarzenia, DBMS może wspierać zatwierdzanie dwufazowe jako pomoc w dokonywaniu zmian tak szybko, jak to możliwe, a tym samym zmniejszać okno podatności na przerwanie.

* W zatwierdzaniu dwufazowym DBMS uzyskuje kopie wszystkich rekordów potrzebnych, gdy operator rozpoczyna transakcję.

* Gdy operator wykona wszystkie niezbędne kroki w celu sfinalizowania transakcji, DBMS blokuje i ponownie odczytuje wszystkie zmienione rekordy oraz porównuje aktualne wartości z wartościami początkowymi; jeśli nie ma różnic, DBMS dokonuje wszystkich wymaganych zmian i natychmiast odblokowuje wszystkie rekordy.

* Jednakże, jeśli bieżące wartości zostały zmodyfikowane od czasu, gdy operator zażądał kopii początkowych, wtedy jakiś inny proces był aktywny, więc DBMS zgłasza, że konieczna jest weryfikacja.

* Operator ma zazwyczaj możliwość wyboru sposobu postępowania; na przykład, jeśli w magazynie nie ma żadnych pozycji, może być konieczne opóźnienie lub anulowanie zamówienia, natomiast jeśli jest wystarczająco dużo pozycji, operator musi jedynie ponownie zainicjować transakcję z nowymi wartościami początkowymi.

Pliki kopii zapasowej bazy danych i dzienniki systemowe

Kiedy informacje w pliku online lub bazie danych są aktualizowane, stare informacje, to znaczy informacje, które znajdowały się w rekordzie przed wprowadzeniem zmiany, są nakładane; jeśli nie zostaną podjęte kroki, znika bez śladu. Z tego powodu wiele systemów DBMS umożliwia kopiowanie obrazu oryginalnego rekordu do pliku dziennika transakcji. W innych przypadkach, gdy musi być zachowana dokładna historia określonej grupy rekordów, jak w przypadku danych ubezpieczeniowych i medycznych, aplikacja może dodawać nowe rekordy, ale nie usuwać starych. Gdyby pliki i bazy danych online nigdy nie zostały uszkodzone ani utracone, utrata danych nie byłaby problemem przy prawidłowej konfiguracji bazy danych. Jednak w niedoskonałym świecie należy podjąć kroki, aby można było odzyskać uszkodzone lub utracone pliki i bazy danych online. Więcej informacji na temat tworzenia kopii zapasowych i odzyskiwania danych można znaleźć w rozdziale 57 niniejszego podręcznika. W celu odzyskania plików online i baz danych, należy najpierw wykonać okresowe kopie zapasowe oraz kopie dziennika rekordów, które zostały zaktualizowane w czasie pomiędzy wykonaniem kopii zapasowych. Częstotliwość wykonywania kopii zapasowej zależy od tego, jak dynamiczne są pliki lub bazy danych. W większości przedsiębiorstw codziennie kopiowana jest znaczna liczba całkowitych lub częściowych plików i baz danych. Nierzadko zdarza się, że działy operacji komputerowych spędzają kilka godzin dziennie na tworzeniu kopii zapasowych. Za każdym razem, gdy tworzona jest kopia zapasowa pliku lub bazy danych, w danym momencie istnieją dwie poprawne

kopie. Aby wyjaśnić, jak działają kopie zapasowe plików i baz danych online oraz rejestrowanie w systemie, pomyśl o pojedynczym pliku online. Plik zostaje przeniesiony do trybu offline i nie jest już dostępny dla transakcji online o godzinie 4:00 i tworzona jest kopia. Obie kopie tego pliku są wówczas identyczne. O godzinie 6:00 kopia zapasowa jest zakończona, a oryginalny plik na dysku jest przywracany do trybu online. Transakcje ponownie zaczną aktualizować ten plik. Od tego momentu z każdą transakcją aktualizacji różnice między tym plikiem a jego kopią zapasową rosną. O godzinie 14:00, jeśli z jakiegoś powodu plik nie nadaje się już do użytku, plik kopii zapasowej jest opóźniony o osiem godzin. W tym momencie plik dziennika staje się krytyczny w procesie przywracania. Plik dziennika jest uporządkowany według czasu i zawiera kopie zaktualizowanych rekordów zarówno przed, jak i po wykonaniu aktualizacji. Zawiera również kopie rekordu transakcji

Odzyskiwanie i ponowne uruchomienie

Po ustaleniu, że plik online lub baza danych została uszkodzona lub zniszczona, inicjowana jest procedura znana jako odzyskiwanie i ponowne uruchamianie. Pierwszym krokiem w tej procedurze jest skopiowanie kopii zapasowej z powrotem na dysk w celu utworzenia nowego oryginału w momencie utworzenia tej kopii zapasowej. Następny krok wykorzystuje plik dziennika do ponownego zastosowania, w sekwencji czasowej, wszystkich transakcji, które zostały wykonane od czasu wykonania kopii zapasowej. Współczesne systemy baz danych mogą zawierać pliki, których nie można przełączyć w tryb offline. Są online 24 godziny na dobę, siedem dni w tygodniu. W celu wykonywania kopii zapasowych, okresowo kopiowane są części bazy danych (kopia dynamiczna). Koncepcyjnie bazę danych można podzielić na części. Kopie zapasowe każdej części można tworzyć oddzielnie w różnych okresach czasu. Można opracować wiele schematów tworzenia kopii zapasowych niektórych rekordów. Na przykład skopiuj do pliku kopii zapasowej co piąty rekord bazy danych w jednym okresie, powiedzmy rekordy 5, 10, 15, 20 itd. Nieco później skopiuj do innego pliku kopii zapasowej rekordy 1, 6, 11, 16, 21 , i tak dalej. Jeśli wystąpi konflikt między kopiowaniem rekordu w celu wykonania kopii zapasowej a transakcją próbującą zaktualizować rekord, należy ustanowić odpowiednią procedurę, aby jedno lub drugie miało miejsce w pierwszej kolejności. Procedury przywracania i ponownego uruchamiania, a także wycofywania działają w ten sam sposób, z dodatkową złożonością polegającą na ustalaniu priorytetów, dla których w przypadku wystąpienia konfliktów wykonywane są pierwsze czynności. Mimo że te konflikty zwiększają złożoność, są one rozwiązywane podczas tworzenia procedur odzyskiwania, ponownego uruchamiania i wycofywania. Innym podejściem do tworzenia kopii zapasowych takich krytycznych plików jest kopia w tle, która kopiuje dane nawet wtedy, gdy pliki są w użyciu. Oprogramowanie do kopiowania w tle omija elementy systemu plików, aby móc uzyskać dostęp do zablokowanych rekordów.

Wycofanie się

Plik dziennika jest również używany w procesie znanym jako wycofanie. Ten proces jest inicjowany, gdy transakcje aktualizacji online nie zostaną zakończone po wykonaniu niekompletnych lub częściowych aktualizacji plików lub baz danych. Na przykład transakcja aktualizacji, w której w trzech oddzielnych plikach występują uzupełnienia do trzech pól, ma mieć miejsce. Po wykonaniu dwóch z trzech aktualizacji transakcja kończy się nieprawidłowo z powodu awarii programu. Ostatecznie program zostaje poprawiony, ale trzeba coś zrobić, aby cofnąć dwie częściowe aktualizacje. W przeciwnym razie, gdy program zostanie naprawiony, a transakcja zostanie ponownie uruchomiona, te dwie aktualizacje zostaną ponownie zastosowane, co spowoduje błędne powielenie. Procedura backout inicjowana jest dla tych transakcji, które gdyby nie zostały prawidłowo zakończone, generowałyby błędy. Odzyskiwanie przy użyciu plików dziennika wymaga oznaczenia początku i końca każdej transakcji; jeśli zapisy w pliku dziennika pokazują rozpoczęcie transakcji bez odpowiadającego

jej zakończenia, procesy odzyskiwania mogą uznać transakcję za niezakończoną. Takie znaczniki odpowiadają tak zwanym punktom kontrolnym w projekcie programu.

Odzyskiwanie do przodu

Innym podejściem do odzyskiwania jest rozpoczęcie od znanego dobrego stanu i ponowne wykonanie wszystkich transakcji, o których wiadomo, że zostały wykonane poprawnie. To przywracanie zmian wymaga zsynchronizowania kopii zapasowej bazy danych i pliku dziennika transakcji, tak aby pierwszy rekord w pliku dziennika transakcji reprezentował pierwszą transakcję następującą bezpośrednio po utworzeniu kopii zapasowej bazy danych. Dysponując tymi danymi, możliwe jest zrekonstruowanie wszystkich modyfikacji aż do ostatniej kompletnej transakcji, ale z wyłączeniem tej transakcji. Wszystkie niekompletne transakcje są odrzucane, chociaż program odzyskiwania zwykle drukuje wszystkie dostępne szczegóły transakcji, które nie zostały zakończone.

PLIKI WSTĘPNE

Systemy operacyjne mają własne narzędzia do pisania i wykonywania plików wsadowych, czasami nazywanych skryptami. Pliki wsadowe mogą być na tyle złożone, że wyglądają jak programy z testami logicznymi, rozgałęzieniami i tak dalej.

Tworzenie pliku kopii zapasowej

Ochrona plików, które są aktualizowane przez programy przetwarzania wsadowego, jest automatyczna, ponieważ plik jest całkowicie kopiowany, a oryginał pozostaje w formie, w jakiej znajdował się przed wykonaniem programu. Dlatego każdy cykl przetwarzania pozostawia własną kopię zapasową. Nazwa przetwarzanie wsadowe pochodzi od idei sekwencyjnego przetwarzania transakcji w grupie. Zawsze są dwa lub więcej plików. Istnieje plik główny, który jest aktualizowany, oraz jeden lub więcej plików transakcji, które zawierają informacje używane do aktualizacji pliku głównego. Proces kopiuje rekordy akt głównych, które nie mają działań związanych z aktualizacją, do nowego zbioru głównego, aktualizuje akta główne, które mają działania, i kopiuje je do nowego zbioru głównego i nie kopiuje tych akt głównych, które są oznaczone jako usunięte w pliku działań . Po zakończeniu procesu dostępne są co najmniej trzy pliki: oryginalny wzorzec, plik(i) działań i nowy wzorzec. Kopie zapasowe to oryginalny plik główny i plik(i) aktywności. W następnym cyklu przetwarzania nowy wzorzec staje się danymi wejściowymi wraz z plikami czynności tego cyklu. Jeśli wystąpi problem z plikiem głównym, plik główny poprzedniego cyklu i wszystkie kolejne pliki działań mogą zostać użyte do wygenerowania zaktualizowanego pliku głównego. Częstą praktyką jest przechowywanie dwóch lub trzech generacji plików głównych i plików działań. Termin generacja jest często stosowany w odniesieniu do cykli przetwarzania wsadowego — generacja 0, generacja 1, generacja 2 — lub jako dziadek, ojciec i syn. W ostatnich dziesięcioleciach te ostatnie terminy zostały zastąpione przez dziadka, rodzica i dziecko.

Kontrole audytu

Innym środkiem bezpieczeństwa stosowanym podczas pracy z plikami wsadowymi jest użycie sum kontrolnych, aby upewnić się, że proces wsadowy został wykonany z dokładnością. Konkretne kontrole mogą obejmować zliczanie rekordów. Mogą również istnieć sumy kontrolne wartości w określonych polach rekordów wejściowych do porównania z sumami w danych wyjściowych, po przetworzeniu dodawania i odejmowania. Na przykład w systemie płac informacje z kart czasu pracy są przesyłane do komputera w celu przetworzenia. Wraz z kartami czasu pracy znajduje się przekaz, który zawiera liczbę kart czasu pracy oraz sumy wszystkich godzin prostych i godzin nadliczbowych, zebrane przez dział, który stworzył karty czasu pracy. Przeprowadzana jest procedura, w której sumy kontrolne na

przekazie są porównywane z sumami faktycznie wprowadzonymi z kart czasu pracy, aby zapewnić poprawność przed rozpoczęciem procesu płacowego. W przypadku napotkania rozbieżności należy je zbadać i poprawić przed podjęciem kolejnego kroku w przetwarzaniu listy płac.

INTEGRALNOŚĆ I WALIDACJA DANYCH

W poniższych sekcjach przedstawiono krótki przegląd walidacji; jednak bardziej obszerne omówienie tego tematu można znaleźć w rozdziale 47 niniejszego podręcznika dotyczącym bezpieczeństwa operacji i kontroli produkcji.

Kontrola walidacji

Niezależnie od tego, czy aplikacja aktualizuje swoje pliki za pomocą przetwarzania wsadowego lub online, czy też w kombinacji obu tych metod, najważniejsza jest walidacja danych wejściowych. Termin GIGO (garbage in, trash out) nie jest już tak często słyszany jak kiedyś, ale jest jak zawsze prawdziwy. Większość współczesnych systemów korzysta z plików i baz danych online, z informacjami wprowadzanymi interaktywnie, co pozwala na walidację u źródła. Istnieje kilka specyficznych technik walidacji, które zostały zastosowane w celu zmniejszenia ilości niepoprawnych informacji wprowadzanych do plików i baz danych.

Metody identyfikowania błędów wejściowych i nieautoryzowanych modyfikacji

Błędy wprowadzania danych są powszechne.

* Czwórki i dziewiątki, jedynki i siódemki oraz szóstki i zera można łatwo pomylić ze sobą.

* Operatorzy mogą pominąć cyfrę lub wstawić obcą.

* Transpozycja to inny rodzaj błędu, który jest popełniany od czasu do czasu. Na przykład operator wpisu zamówienia może wprowadzić 3286 zamiast 3826 (błąd transpozycji) lub być może 7790 zamiast 7796 (błąd transkrypcji).

* Takie błędy byłyby zgłaszane, gdy obliczenie cyfry kontrolnej daje liczbę, która różni się od wprowadzonej cyfry kontrolnej.

* Cyfra kontrolna to dodatkowa cyfra dodawana do generowanej zmiennej jako sufiks.

* Na przykład pięciocyfrowy numer giełdowy może mieć dodatkową cyfrę; szósta cyfra lub cyfra kontrolna, jak to się nazywa, jest obliczana przez zastosowanie algorytmu (wykonywanie pewnych działań arytmetycznych) na pierwszych pięciu cyfrach, co daje wynik jednocyfrowy.

* Aby zminimalizować błędy wprowadzania, cyfry kontrolne mogą być używane w dowolnych zmiennych generowanych przez program; na przykład numery części, numery pracowników i kody różnych typów.

Pojedyncza cyfra kontrolna może czasami ukryć istnienie podwójnych lub wielokrotnych błędów wejściowych w łańcuchu wejściowym ze względu na stosunkowo proste schematy arytmetyczne. Bardziej złożone wersje cyfry kontrolnej generują sekwencję cyfr zwaną sumą kontrolną, która zapewnia większą moc pomagającą w identyfikowaniu błędów wprowadzania lub oszustw; numery kart kredytowych zazwyczaj zawierają czterocyfrową sumę kontrolną na końcu numeru. Na przykład typowy numer karty kredytowej VISA ma format 1111 2222 3333 4444; czwarty blok (w naszym przykładzie 4444) jest obliczany przy użyciu tajnego algorytmu jako funkcja pozostałych trzech bloków. Przestępcy komputerowi często próbowali (i czasami im się to udawało) odtworzyć algorytm sumy kontrolnej w celu utworzenia numerów VISA, które w rzeczywistości należą do nieznanych ofiar.

Organizacje zajmujące się bezpieczeństwem kart kredytowych nauczyły się stosować złożone metody heurystyczne do identyfikowania mało prawdopodobnych transakcji; na przykład, jeśli prawowity właściciel właśnie zatankował swój zbiornik paliwa na często używanej stacji paliw w Vermont, jest mało prawdopodobne, aby transakcja pięć minut później na 1000 wełnianych koców w firmie produkcyjnej we Włoszech była uzasadniona, niezależnie od dokładności szukam numeru VISA. Rozszerzenie sumy kontrolnej, suma skrótu, jest powszechnym narzędziem do identyfikacji logicznego lub fizycznego uszkodzenia danych. Suma skrótu to po prostu bezsensowna suma wartości liczbowych, takich jak numery części, a nie znacząca suma ilości. Ponowne obliczenie sumy skrótu może zwykle wskazać, czy dane zostały uszkodzone od czasu ostatniego obliczenia sumy skrótu. Podpis cyfrowy to wygenerowana kryptograficznie wartość oparta na kluczu szyfrowania. Zastosowanie tego samego procesu podpisu cyfrowego do danych powinno wygenerować identyczny podpis. Ponadto, zastosowanie kryptosystemu klucza publicznego umożliwia weryfikację autentyczności oraz integralności podpisywanych danych.

Kontrola zasięgu

Kontrole zakresu oferują inny sposób sprawdzania poprawności informacji u źródła. W sytuacjach wprowadzania ilości i wartości walutowych istnieje możliwość ustalenia limitów dolnych i górnych. Na przykład sprawdzenie zakresu można zastosować do zamówionej ilości, aby upewnić się, że wprowadzona wartość mieści się w określonych granicach, takich jak nie mniej niż 10 lub nie więcej niż 60 określonej pozycji. Chociaż nie jest to całkowicie odporne na błędy, takie ograniczenia mogą przynajmniej wyeliminować oczywiste błędy i zredukować te, które nie są tak rażące. Sprawdzanie zakresu z wartościami 10 i 40 wyeliminuje dodatkowy błąd 0, gdy operator wprowadzi 100 lub 600 albo 1 lub 4.

Sprawdzanie ważności za pomocą tabel

Korzystanie z tabel wartości lub prawidłowych kodów jest jednym z najlepszych sposobów zapewnienia, że do plików i baz danych wprowadzane są tylko poprawne informacje. We współczesnych plikach i bazach danych typ danych jest określany jako właściwość pola w fazie projektowania, a właściwości te można wykorzystać do upewnienia się, że informacje tekstowe nie zostaną wprowadzone do pól numerycznych i odwrotnie. Właściwości są również używane do odfiltrowywania nieprawidłowych konfiguracji danych. Chociaż niemożliwe jest odfiltrowanie błędnie zapisanych nazwisk i innych otwartych typów informacji, możliwe jest odfiltrowanie nieprawidłowych kodów stanu i innych naruszeń standardów. Na przykład tabela zawierająca prawidłowe kody stanu w danym kraju może być wykorzystana do upewnienia się, że w polu stanu można wprowadzić tylko te kody. Inne takie jednostki (np. kody departamentów, kody lub typy produktów oraz klasy cenowe) są głównymi kandydatami do walidacji tabeli. Ze stołami możliwe jest również tworzenie stołów kombinowanych. Na przykład, jeśli określony kod produktu mieści się w określonym przedziale, dopuszczalne mogą być tylko cztery klasy cenowe. Konkretnie, załóżmy, że gdy kod produktu mieści się w przedziale od 600 do 699, wówczas klasą cenową mogą być tylko R, PR, OC i CD. Takie tabele służą do sprawdzania poprawności informacji w momencie ich wstępnego wprowadzania do plików i baz danych. Wprowadzanie nieprawidłowych informacji, celowych lub nieumyślnych, jest blokowane. Tabele zabronionych kombinacji są szczególnie ważne w systemach sterowania w czasie rzeczywistym dla procesów, które mogą pójść poważnie źle w wyniku złych danych wejściowych. Na przykład w grudniu 1992 r. eksplozja w fabryce chemicznej w Holandii została przypisana błędowi wprowadzania danych, któremu można było zapobiec, sprawdzając poprawność tabeli z zabronionymi mieszaninami chemicznymi. Poniżej znajduje się fragment raportu opublikowanego w RISKS 14.22 (4 stycznia 1993) przez Meine van der Meulen z Holenderskiej Organizacji Stosowanych Badań Naukowych, Departament Bezpieczeństwa Przemysłowego:

Wypadek rozpoczął się od błędu w pisaniu recepty przez pracownika aborcyjnego. Zamiast zbiornika 632 wpisał zbiornik 634. W zbiorniku 632 przechowywana była żywica klasyczna (UN-1268) i zwykle stosowana w procesie wsadowym. W zbiorniku 634 przechowywano DCDP (dicyklopentadien). Operator, który musiał sprawdzić, czy zawartość zbiornika jest [sic] zgodna z receptą, napełnił reaktor niewłaściwymi chemikaliami. … nastąpił ciężki wybuch, w wyniku którego zginęło 3 strażaków zakładowej straży pożarnej, a 11 pracowników zostało rannych [w tym] 4 strażaków zakładowej straży pożarnej. Szkody oszacowano na kilka [10] milionów guldenów NL

Tabele powinny być generalnie używane tylko do informacji, które są względnie statyczne, ze względu na konieczność konserwacji. Częsta konserwacja tabeli może prowadzić do błędów, które mają efekt kaskadowy. Jeśli tabela jest niepoprawna, wszystkie informacje rzekomo potwierdzone przez tę tabelę mogą być nieprawidłowe w plikach i bazach danych.

Narzędzia diagnostyczne

Programy produkcyjne muszą działać na dających się zweryfikować poprawnych danych. Każdy program produkcyjny powinien zawierać narzędzia diagnostyczne do skanowania baz danych w poszukiwaniu logicznie niemożliwych wartości lub odchyleń od rzeczywistości. Na przykład procedura diagnostyczna może sprawdzić, czy suma wszystkich nagłówków jest zgodna z sumą obliczoną na podstawie wartości połączonych pozycji. Ponieważ starsze dane mogły zostać zaakceptowane zgodnie ze starszymi regułami walidacji zakresu, wszystkie rekordy należy sprawdzić, aby upewnić się, że są zgodne z bieżącymi ograniczeniami zakresu. Podobnie wszelkie inne ograniczenia logiczne w systemie aplikacji powinny być wyraźnie uwzględnione w programie diagnostycznym. Wyjście z programu powinno zawierać wszystkie szczegóły odpowiednie do identyfikacji i poprawienia błędnych danych. Na przykład, zamiast po prostu wskazywać „ZŁA WARTOŚĆ W POLU CENY, ZAPIS 1234”, dane wyjściowe powinny wyjaśniać coś w stylu: „CENA ZA POZYCJĘ 234 = 45,67 USD, KTÓRA PRZEKRACZA KONFIGUROWANY LIMIT 40,00 USD WYKAZANY W REJESCIE GŁÓWNYM ZAPASÓW 78999”.

UWAGI KOŃCOWE

W innym zastosowaniu zasady 80/20, 20 procent prac projektowych i rozwojowych pochłania obsługa 80 procent danych (normalnych i poprawnych) przechodzących przez system, podczas gdy 80 procent prac projektowych i rozwojowych jest wydatkowanych za obsługę 20 procent danych (błędów, wyjątków i nietypowych sytuacji) przechodzących przez system. Można śmiało powiedzieć, że na walidację oraz procedury tworzenia kopii zapasowych i odtwarzania należy poświęcić więcej wysiłku niż na te procesy, które obsługują zweryfikowane dane i rutynowe zadania przetwarzania aplikacji. Dobra kontrola aplikacji obejmuje sprawdzanie poprawności danych na wszystkich poziomach architektury, a także staranne rozważenie uprawnień przyznanych użytkownikom z perspektywy aplikacji, systemów i baz danych. Nie wystarczy skupić się tylko na danych wejściowych użytkownika i zaufanych danych wejściowych z innych systemów. W dzisiejszych złożonych systemach istnieje zbyt wiele obszarów, w których dane mogą zostać przypadkowo lub złośliwie zmodyfikowane, aby można było ufać jakimkolwiek wejściom bez weryfikacji. Ponadto organizacje muszą wykorzystać wszystkie funkcje systemów baz danych, aby nie tylko zachować integralność, ale także zrozumieć mechanizmy tworzenia kopii zapasowych i odzyskiwania danych. Aby poradzić sobie z awariami systemu, należy wdrożyć odpowiednie procesy i procedury tworzenia kopii zapasowych i odzyskiwania danych. Te procesy i procedury obejmują zarówno odzyskiwanie po awarii, jak i odzyskiwanie pojedynczego systemu lub aplikacji. Aby zapewnić możliwość przywrócenia witryn, przedsiębiorstw lub aplikacji po awariach, należy wprowadzić procesy i procedury podczas procesu projektowania i rozwoju oraz aktualizować je zgodnie z wymaganiami sytuacji.