Komentarze

Bazy danych: Gry i zabawy z Triggerami (i nie tylko)

Spis treści

[Refleksje 3](#_Toc40081826)

[One upmanship 3](#_Toc40081827)

# Refleksje

Tu macie zbiorcze refleksje

## Aktualizacja wpisu <> update

Aktualizacja wpisu to zarówno UPDATE jak i DELETE.

## Wszystko na raz

Tu się dzieje dość sporo na raz:

CREATE TRIGGER trigger\_widok1

ON trigger\_widok

INSTEAD OF **INSERT,UPDATE,DELETE**

ALBO

*„Nie można stworzyć 3 triggerów na jednym widoku co komplikuje zadanie”*

Można. Trzeba. Lepiej to rozbić na 3 osobne triggery

## To całkiem ciekawa refleksja 1

„zamiast bezpośredniej zmiany wartości na widoku jest dokonywana zmiana poprzez trigger na tabeli macierzystej, powoduje to że tabela inserted i deleted zawiera tylko kolumny z widoku w których brak UsersId i MovieId, co powoduje problemy z modyfikacją konkretnego rekordu „

I tak, i nie – to, że 1) gro BD stawia z automatu indeksy na kluczach głównych 2) ludzie nie ograniają BD i wszędzie używają klucze surogaty (Id) jako klucze główne… to powoduje, że faktycznie zjawisko „problemy z modyfikacją konkretnego rekordu” występuje.

Ale BD przechowująca filmy raczej będzie chciała mieć efektywne wyszukiwanie na podstawie tytułu filmu więc pewnie ten indeks tam się pojawi – i automatycznie spostrzeżenie będzie nieprawdziwe.

## Nie robić takich złączeń!



## To całkiem ciekawa refleksja 2

„Triggery użyte w odpowiedni sposób są bardzo pomocne. Dzięki nim możemy zapobiegać dostawaniu się o bazy danych, które są nieodpowiednie oraz na bieżąco kontrolować aktywność wykonywanych operacji na bazie danych w czasie sesji. Istnieje możliwość powiadomienia innych programów o konieczności wykonania niezbędnych operacji, gdy w tabeli nastąpią zmiany.”

I tak, i nie – ogólnie ludzie nie ogarniają BD i konsensus środowiska to: „Triggery są złe”. Jak ktoś ogarnia to konsensus jest „Triggery AFTER/BEFORE są złe”. Wszystkie elementy aktywne BD wpływają na ‘szybkość’ BD – najczęściej mierzonej w Transakcjach na Sekundę (TpS) – kalkulacja jest prosta. Jeśli tabela bez triggerów wykonuje 1000 transakcji na sekundę, to dziennikowanie realizowane przez trigger będzie wymagać dwukrotnie więcej czasu tj. miara spadnie do 500 tps-ów.

I tu wchodzimy na temat listy 5 – ten sam spadek wydajności widoczny jest gdy położymy indeksy tj. N indeksów spowolni nam N-krotnie zapis/update tabeli (w pewnych warunkach). Oczywiście szkopuł w tym, że indeksy poprawiają prędkość odczytu, więc czasem też i prędkość operacji UPDATE(WHERE) – dlatego indeksy nikogo nie bolą, a wręcz chcemy je mieć.

Tylko gdy mamy 2 tabele (A i B) i każda ma 5 indeksów + istnieje trigger np. INSERT A->B, to wsadzenie rekordu do tabeli A jest **10 wolniejsze** niż do bezindeksowej/beztriggerowej tabeli. **Dlatego triggery bolą, bo kaskadowo zwiększają wymagany pobór IO – a im dłużej transakcja się wykonuje, tym większe prawdopodobieństwo zablokowania innej transakcji (izolacje z listy 3). TA SYNERGIA BARDZO SZYBKO JEST ZAUWAŻALNA PRZY PEWNYM OBCIĄŻENIU BD I TO ONA ZABIJA SERWERY.**

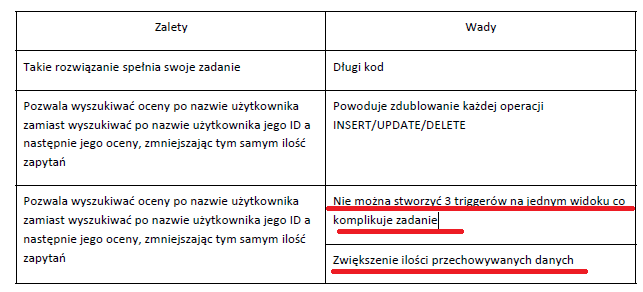
**JEŚLI NIE POTRAFICIE PROGRAMOWAĆ TRIGGERÓW – W SENSIE, NIE UMIECIE PRZEWIDZIEĆ WPŁYWU NA IO/BLOKADY TO LEPIEJ LOGIKĘ TRIGGERÓW PRZENIEŚĆ NA WARSTWĘ APLIKACJI (KOD).**

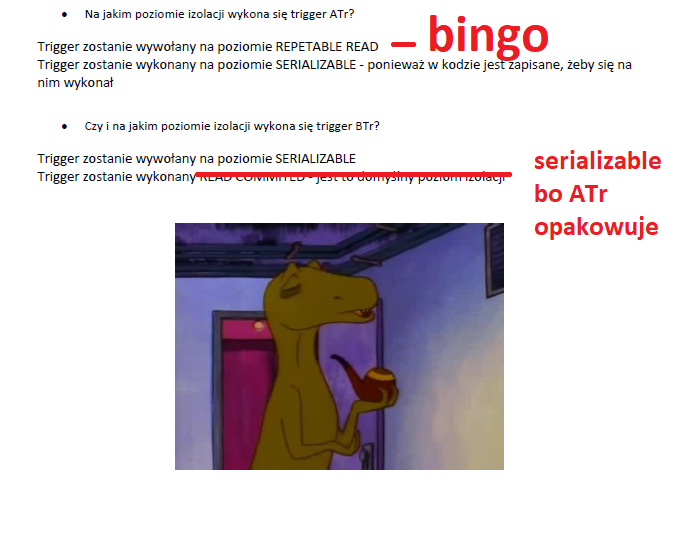
## To całkiem ciekawa refleksja 3

*„Triggery są bardzo przydatną funkcją, dzięki nim możemy reagować na zmiany w tabeli. Triggery pozwalają na zapisywanie zmian w bazie danych, przez co są powszechnie używane.”*

**( ͡° ͜ʖ ͡°) patrz tyrada wyżej**

## Do 158369





Ogólnie git.

## ?

Niestety występowały błędy i w rezultacie nie udało się rozwiązać tego zadania. Nie jestem do końca pewien zrozumienia przeze mnie jego treści, więc pozostawiam je bez rozwiązania.

## O\_o

Również przechowywanie tych danych w całkowicie innej tabeli wydaję mi się byd **mało optymalnym** rozwiązaniem i jestem przekonany, że istnieje lepsze rozwiązanie tego problemu.

## wydajność - Skoro już o tym mowa

Łukasz pisze odnośnie baz temporalnych:

*Na pewno do wad tego rozwiązania należy czasochłonność wykonania polecenia UPDATE (chociaż najprawdopodobniej jest to związane z chwilową ‘’niewydolnością’’ mojego komputera). Zaletą tego rozwiązania, że jest również to, że zasada działanie tej operacji wydaje się być dużo bardziej zrozumiała, a przede wszystkim bardziej spójna od tej przedstawiona w zadaniu 1****. Do wad tego rozwiązania, może należeć fakt, że każda zmiana przechowywana jest w tabeli, co może powodować obciążenie bazy danych przy bardzo dużej ilości rekordów*** *czy chociażby skumulowanym ataku – w takim momencie można by było zadbać o możliwą maksymalną ilość oceny dla danego użytkownika w zadanym przedziale czasowym.*

To nie jest prawda, a przynajmniej wpływ nie jest tak duży jak się może wydawać na początku. Pamiętajcie o tym, że tabele temporalne w SQL Serverze to tak naprawdę 2 tabele: „gorąca” z aktualnymi danymi oraz historyczna. Tabela historyczna jest skompresowana wg. **stron** (8 kB jednostka alokacji). Jest bardzo duże ryzyko, że aktualizując rekord – aktualizujemy np. 1 kolumnę – reszta pozostaje bez zmian. W takim przypadku kompresja jest **bardzo** **efektywna**. Dlatego narzut jest mały – ogólnie zalety przewyższają wady.

## 160763 – zadanie 2

*„Nie widzę żadnych wad tego rozwiązania” ( ͡° ͜ʖ ͡°).*

Są. Dodatkowa warstwa abstrakcji pomiędzy aplikacją, a fizycznymi strukturami BD. No i czasem, CZASEM zachodzi potrzeba operowania na identyfikatorach.

## A propo zadania 3

Mogliście zapytać – nie bardzo interesuje mnie trigger UPDATE/DELETE.

Zadanie wzięło się z case-a który pojawił się na początku maja w firmie w której pracuje. Mogłem nadmienić, że rozwiązanie ma na celu obsługę bazy tzw. Hurtowni Danych. Bazy takie są zazwyczaj append-only (tylko inserty).



## Do 160767

W zad 1 zaleciałeś hardcorem: DDL-ka w triggerze: BARDZO RYZYKOWNE

W zad 2: *Po trzecie, czytając instrukcję można wtedy wywnioskować, że podobna sytuacja ma mieć miejsce przy akcji DELETE, co jest niedopuszczalne*. – zgadzam się

## Do 160769 - Zadanie 3

Wzór rozwiązania zadania 3:

CREATE TRIGGER zmiana ON Users\_Info INSTEAD OF INSERT AS

BEGIN

INSERT INTO Users\_Info2~~(name)~~

SELECT i.name FROM inserted i

END

## Do 157760

### Ad 3

Niedopowiedzenia w zadaniach tworzą piękne interpretacje zadania – i do tej pory Twoja najbardziej mi się spodobała. Tak jak pisałem wcześniej – zadanie wzięło się z case-u w firmie gdy potrzebowaliśmy ultra-quick-real-time-fix. Tylko u nas były 2 systemy:

Źródłowy -> Baza danych ze zdarzeniami -> Hurtownia

No i problem był taki jak pisałeś:

*Ten sposób ma jeden mankament - jeśli nie chcemy zmieniać nic w kodzie aplikacji operującej na tej bazie danych, to zapytania* ***SELECT*** *będą mogły odnosić się tylko do początkowej tabeli Users\_Info.*

Rekomendowałem postawienie bazy danych o innej nazwie, ale **takich samych nazwach tabel** i podmianę connection stringa od strony hurtowni danych:

Źródłowy -> Baza danych ze zdarzeniami (interceptor) -> Zminifikowana BD -> Hurtownia

Rozważaliśmy też to co Ty zaproponowałeś: unię na widoku.

### Ad 4

Dokładnie tak trzeba było zrobić (stack). Pytanie filozoficzne jest bardzo dobre.

### Ad 5

A to nieprawda; *„nie można ocenić czy aktualna kombinacja userId, movieId znajduję się w bazie na podstawie samej tabeli temporalnej”.* Bo temporalne tabele w SQL Serverze to 2 tabele.

## DO 160766

Obcinałem punkty za brak triggera update lub delete w zadaniu 1 …. Ale zawsze się trafi ktoś taki jak Ty – pewnie grasz w LOL-a na defence-ie: „*Zastanawiające może być, czy trigger wyłącznie dla aktualizacji rekordow jest wystarczający. Jeżeli przyjmiemy następujący scenariusz działania bazy: raz utworzona informacja o wystawieniu oceny zostaje na zawsze w bazie, to znaczy użytkownik, ktory kasuje swoją ocenę filmu, w rzeczywistości ustawia ją jako 0.0 (lub bezpieczniej na NULL – co oczywiście wymaga usunięcia ograniczenia NOT NULL z kolumn rating), to przechwytywanie operacji aktualizacji w zupełności wystarczy.”*

## d\*pa debugging

Przypomnijcie mi jak będę drukował do dziekanatu, żeby **nie** wysyłać tego sprawozdania