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.**

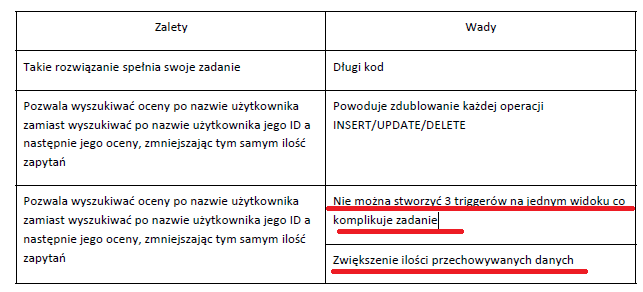
**JEŚLI NIE POTRAFICIE PROGRAMOWAĆ TRIGGERÓW – W SENSIE, NIE UMIECIE PRZEWIDZIEĆ WPŁYWU NA IO/BLOKADY TO LEPIEJ LOGIKĘ TRIGGERÓW PRZENIEŚĆ NA WARSTWĘ APLIKACJI (KOD).**

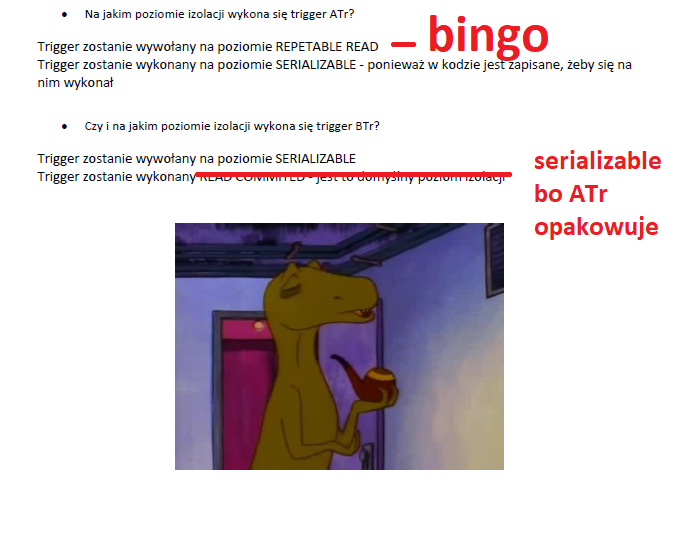
## To całkiem ciekawa refleksja 3

*„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.