Bazy Danych 2024
Lista zadań

Do zdobycia jest 9 punktów ale do maksymalnej punktacji z ćwiczeń wliczymy tylko 7.

1. (1 pkt.) Nadleśnictwo posiada bazę danych umożliwiającą sporządzanie oraz przechowywanie tekstowych opisów poszczególnych sektorów lasu. Każdy pracownik leśny podczas obchodu lasu dysponuje aplikacją mobilną działającą w następujący sposób: w razie potrzeby edycji opisu bieżacego sektora aplikacja łączy się z bazą danych, przesyła lokalizację i pobiera krotkę dotyczącą tego sektora (w ramach pojedynczej transakcji typu READ ONLY). Pracownik może edytować pobrany opis, a po zakończeniu edycji uruchamiana jest nowa transakcja, która zapisuje wprowadzone zmiany w bazie.

Od pewnego czasu wprowadzono obowiązek aby każdy obszar leśny był regularnie wizytowany przez komisję składającą się ze specjalistów ds. ochrony przyrody, ds. gospodarki leśnej oraz innych. Specjaliści często zgłaszają, że wprowadzane przez nich modyfikacje opisów sektorów znikają mimo, że nikt nie przyznaje się do usuwania czegokolwiek. Prowadzi to do częstych nieporozumień i powstawania teorii spiskowych (np. notatka ekologa o gnieździe rzadkiego ptaka znika, a w jej miejsce pojawia się wycena drewna możliwego do pozyskania w przypadku wycinki).

Wiadomo, że system bazy danych stosuje protokół gwarantujący szeregowalność. Co w takim razie może być przyczyną problemu? W jaki sposób można się go pozbyć? Zaproponuj rozwiązanie, które pozwoli na maksymalną współbieżność(np. możesz zaproponować zmianę schematu bazy).

Uczelniana Baza Danych

Rozważmy sytuację pewnej uczelni gdzie używana jest baza PostgreSQL. Dla każdego z następujących przypadków określ, który poziom izolacji najlepiej zastosować. Jakie będą problemy/wady jeśli zostanie wybrany inny (niższy lub wyższy) poziom niż zalecany przez Ciebie? Wytłumacz co się dzieje powołując się na dokumentację PostgreSQL. Sprawdź eksperymentalnie (zawsze jeśli to możliwe) przed zajęciami czy Twoje rozwiązanie odpowiada rzeczywistości - uruchom odpowiednie zapytania w dwóch równoległych sesjach `psql` i zobacz co się stanie. Przypomnij w jaki sposób każdy z poziomów izolacji jest implementowany w PostgreSQL.

2. (1 pkt.) Aby sprawdzić poprawność danych aplikacja wylicza sumaryczną liczbę przepracowanych godzin dla wszystkich pracowników uczelni, a następnie sumaryczną liczbę przepracowanych godzin przez pracowników każdego z wydziałów. Aplikacja następnie sprawdza czy obliczone wyniki są równe. Równolegle odnotowywane są przepracowane godziny pracownikom i departamentom, każda

dopisana liczba godzin w pojedynczej transakcji obejmującej operację UPDATE dla jednej i drugiej tabeli.

```
BEGIN ISOLATION LEVEL ______;
    SELECT SUM(hours) FROM employees;
    SELECT SUM(hours) FROM departments;
COMMIT;
```

3. **(1 pkt.)** Chcemy zapewnić, że podczas przydzielania zajęć pracownikowi nie zostanie zlecone więcej godzin niż wynosi jego pensum. Zakładamy, że każdy pracownik prowadzi zajęcia na wielu kierunkach, a każdy kierunek ma swojego dyrektora dydaktycznego i wszyscy ci dyrektorzy układają plan w nocy przed deadlinem. Parametry zapytania ustalane przez aplikację: `:newhours, :emp`. W relacji `plan` kolumna `teacher` pamięta identyfikator pracownika, a `hours` łączną liczbę przydzielonych mu godzin. Zakładamy, że w danym momencie w bazie uruchamiane są wyłącznie transakcje takie jak poniższa.

```
BEGIN ISOLATION LEVEL ______;
  UPDATE plan SET hours = hours + :newhours
      WHERE teacher = :emp;
  SELECT hours INTO asum
      FROM plan;
      WHERE teacher = :emp;
  IF (asum <= 210)
    COMMIT;
  ELSE
    ROLLBACK;
```

Przy okazji, jak w naturalny sposób powyższy warunek wymusić nie przejmując się poziomami izolacji?

4. **(1 pkt.)** Podobnie jak w poprzednim zadaniu chcemy zapewnić, że podczas przydzielania zajęć pracownikowi nie zostanie zlecone więcej godzin niż wynosi jego pensum. Parametry zapytania ustalane przez aplikację: `:emp, :newhours, :course`. Tym razem inny jest schemat bazy: każda krotka w relacji `plan` ma 3 kolumny: `teacher` z identyfikatorem pracownika, `course` z identyfikatorem przedmiotu, a `hours` liczbę przydzielonych godzin z danego przedmiotu. Dodanie pracownikowi godzin polega na dopisaniu nowej krotki do tabeli `plan`. Zakładamy, że w danym momencie w bazie uruchamiane są wyłącznie transakcje takie jak poniższa.

```
BEGIN ISOLATION LEVEL ______;
  INSERT INTO plan(teacher, course, hours)
       VALUES (:emp, :course, :newhours);
  SELECT SUM(hours) INTO asum
       FROM plan;
       WHERE teacher = :emp;
```

```
IF (asum <= 210)
  COMMIT;
ELSE
  ROLLBACK;
```

Czy ten sam naturalny sposób zapewnieniający spełnienie warunku na maksymalne pensum może być wykorzystany również dla tego scenariusza?

5. **(1 pkt.)** Aplikacja przed zapisaniem studenta na termin egzamin upewnia się, że student nie zapisał się na więcej niż 2 terminy z egzaminu z podanego przedmiotu.

```
BEGIN ISOLATION LEVEL _______;
  SELECT COUNT(*) INTO anumber FROM enrol
     WHERE student = :student
       AND course = :course;
  IF (anumber <= 1)
    INSERT INTO enrol VALUES (:student, :course, :newslot);
COMMIT
```

6. **(1 pkt.)** Nowy dyrektor Instytutu postanowił przyznać 10% podwyżki wszystkim pracownikom, którzy w 4-letnim okresie ewaluacyjnym opublikowali dokładnie 3 lub 4 publikacje naukowe (wiadomo, pracownik z mniej niż 3 pracami może spowodować pogorszenie wyniku ewaluacji, a z więcej niż 4 naraża społeczeństwo na niepotrzebne koszty). Spodziewamy się, że poza naszą transakcją biblioteka cały czas dopisuje kolejne publikacje (których liczba jest zapamiętywana w kolumnie `no_of_articles`), a dział kadr zatrudnia i zwalnia pracowników.

```
BEGIN ISOLATION LEVEL _______;
UPDATE employees
 SET remuneration = 1.10 * remuneration
 WHERE no_of_articles BETWEEN 3 AND 4;
COMMIT
```

7. **(1 pkt.)** Mamy bazę z dwiema tabelami:

```
Bombiarze(id INT PRIMARY KEY, stopien INT),
Agenci(id INT PRIMARY KEY, bombiarz INT).
```

Bombiarze są podejrzewani o podkładanie bomb w autobusach, agenci ich śledzą, monitorują stopień zagrożenia (atrybut `stopien`) i w przypadku zebrania odpowiednich dowodów zatrzymują do wyjaśnienia. Można założyć, że jesli bombiarz ma przydzielonego agenta to nie będzie w stanie podłożyć bomby. Agenci są dobrze zakonspirowani i czasem ich atrybut `bombiarz` ma wartość NULL - nie wiemy wtedy, którego bombiarza śledzi agent (a nawet czy w ogóle jakiegoś śledzi).

Co dokładnie wypisze poniższe zapytanie? Czy ono jest właściwe jeśli chcemy otrzymać listę bombiarzy, o których wiadomo *na pewno*, że nie mają przydzielonego agenta? A co jeśli chcemy otrzymać listę bombiarzy, którzy *być może* nie mają przydzielonego agenta?

```
SELECT id FROM Bombiarze
WHERE id NOT IN (SELECT bombiarz FROM Agenci)
```

Odpowiedzi uzasadnij. Napisz zapytania zwracające poprawne listy dla obu wariantów. Przetestuj jak szybko działają Twoje zapytania (dla bazy z milionem Bombiarzy i Agentów) i popraw je jeśli ich działanie trwa zbyt długo.

Indeksowanie

Zaproponuj indeksy, które przyspieszą w sprawdzianowej bazie danych wykonywanie operacji opisanych poniżej. Wytłumacz dlaczego proponowane przez Ciebie rozwiązanie jest odpowiednie, wykonaj eksperyment i opisz jego wyniki.

Do wylosowania danych testowych (nie odtwarzaj całej bazy, tylko te tabele/kolumny, które są potrzebne), możesz np. zmodyfikować następujący kod. Użyj odpowiednio dużych danych (≥5 mln krotek).

```
INSERT INTO points(p, ts)
   SELECT point(n*random()/50000, n*random()/50000),
   NOW() + (random() * (interval '30 days')) + '5 days'
   FROM generate_series(1,5000000) AS n;
```

8. **(1 pkt.)** Usuwanie krotek z tabeli `company`, do których nie odwołuje się żadna oferta (poprzez `company_id`). Załóżmy, że takie krotki są obecne, ale nieliczne, tak jak w stanie początkowym bazy. Dlaczego większy zysk czasowy z zaproponowanego indeksu jest dla operacji sprawdzania bezpieczeństwa usunięcia krotki w obecności klucza obcego (trigger dla klucza obcego) niż z, tak naprawdę tożsamego, wybierania krotek z `company`, do których nie odwołuje się już żadna oferta.

   Zapoznaj się z komentarzem do tego zadania: 4. sprawdzian SQL..

9. **(1 pkt.)** Szybkie wybieranie ofert opublikowanych w podanym przedziale dni (np. od 1 do 10 września 2023). Czy gdyby takie zapytanie było priorytetem w Twoim systemie to mogłoby zostać przypieszone z użyciem polecenia CLUSTER? Jaki wpływ miałoby wydanie takiego polecenia na inne zapytania?