Bazy Danych 2019
Lista zadań

1. (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).

Czy poniższe zapytanie poprawnie wypisuje listę bombiarzy, którzy nie mają przydzielonego agenta?

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

Odpowiedź uzasadnij. Jeśli uważasz zapytanie za niepoprawne to je popraw.

2. (1 pkt.) Jak odpowiedź na zapytanie `SELECT <expr> FROM Bombiarze` zależy od wyrażenia `<expr>` oraz od tego czy kolumna `stopien` zawiera wartości NULL?

Jako `<expr>` wypróbuj: `stopien` (ile krotek jest w odpowiedzi?), `count(*)`, `count(stopien)`, `count(distinct stopien)`, `max(stopien)`, `sum(stopien)`.

3. (1 pkt.) Przeczytaj opis implementacji transakcji w pewnej grafowej bazie danych: *Each transaction is represented as an in-memory object whose state represents writes to the database. This object is supported by a lock manager, which applies write locks to nodes and relationships as they are created, updated, and deleted. On transaction rollback, the transaction object is discarded and the write locks released, whereas on successful completion the transaction is committed to disk.*

Committing data to disk uses a Write Ahead Log, whereby changes are appended as actionable entries in the active transaction log. On transaction commit (...) a commit entry will be written to the log. This causes the log to be flushed to disk, thereby making the changes durable. Once the disk flush has occurred, the changes are applied to the graph itself. After all the changes have been applied to the graph, any write locks associated with the transaction are released.

Załóżmy na potrzeby naszej analizy, że w przypadku operacji czytania transakcja najpierw próbuje odczytać dane ze swojego stanu (czyli widzi własne modyfikacje danych), a jeśli dane nie były modyfikowane to czyta je bezpośrednio z bazy

respektując write-locki innych transakcji (tj. oczekując na zakończenie transakcji modyfikującej dane, które zamierza przeczytać).

Co można powiedzieć o poziomie izolacji osiąganym dzięki powyższej implementacji? Przedyskuj możliwość wystąpienia następujących problemów: *dirty read, nonrepeatable read, phantom read, serialization anomaly* —definicje tych pojęć znajdziesz w dokumentacji postgresql (link był podany na wykładzie).

4. (1 pkt.) Aplikacja do zarządzania lasem używa bazy PostgreSQL. W bazie znajduje się m.in. tabela `drzewo(id, gatunek, lat, lng, opis)`. Z kodu aplikacji do zarządzania lasem wynika, że w celu podania sumy liczby drzew wszystkich gatunków chronionych, aplikacja wysyła polecenie sql:

BEGIN TRANSACTION ISOLATION LEVEL READ COMMITTED;

a następnie dla wszystkich gatunków chronionych, po kolei, pyta bazy danych, ile drzew danego gatunku w niej jest oraz zwraca sumę uzyskanych drzew. Czy mamy gwarancję, że uzyskany w ten sposób wynik odzwierciedla prawdziwą liczbę drzew chronionych (w jakimkolwiek momencie)? Co się zmieni jeśli aplikacja wyśle polecenie

BEGIN TRANSACTION ISOLATION LEVEL REPEATABLE READ; ?

Uzasadnij odpowiedzi.

5. (1 pkt.) Pracownicy leśni dysponują aplikacją mobilną działającą w następujący sposób: w razie potrzeby edycji opisu jakiegoś drzewa aplikacja łączy się z bazą danych z poprzedniego zadania, przesyła lokalizację i pobiera krotki dotyczące drzew rosnących w pobliżu (w ramach pojedynczej transakcji typu READ ONLY). Pracownicy mogą wtedy edytować pobrane opisy, które następnie zbiorczo, w pojedynczej transakcji, zapisywane są 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 drzew znikają mimo, że nikt tych modyfikacji nie usuwał. 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).

W jaki sposób można pozbyć się powyższego problemu? Obecnie wszystkie transakcje są uruchamiane jako READ COMMITTED. Czy zmiana poziomu izolacji transakcji na REPEATABLE READ wystarczy? A na SERIALIZABLE? Zaproponuj rozwiązanie problemu, który pozwoli na maksymalną współbieżność.

6. (1 pkt.) Rozważmy złączenie relacji `semestr, przedmiot_semestr, przedmiot, grupa, uzytkownik (alias prac), wybor, uzytkownik (alias stud)` z bazy danych zapisów, które posłuży nam do wybrania trójek:

```
(prac.nazwisko,stud.nazwisko,semestr.nazwa)
```

takich, że dany student uczęszcza na jakieś zajęcia do danego pracownika w danym semestrze. Sprawdź (użyj EXPLAIN), w jakiej kolejności PostgreSQL wykona powyższy ciąg złączeń i czy zależy on od porządku, w jakim podajesz (łączysz) relacje na liście FROM. Przedstaw plan zapytania PostgreSQL w postaci drzewa. Następnie dodaj do zapytania:

- warunek na nazwisko pracownika;
- warunek na nazwisko studenta.

Jak każdy z warunków wpływa na drzewo zapytania?

7. (2 pkt.) Rozważmy tabele `student(id INT PRIMARY KEY, name TEXT, gender TEXT, birthday DATE)` oraz `zapisy(id INT PRIMARY KEY, sid INT, course INT)`. W tabelach przechowujemy dane o wszystkich studentach w Polsce i ich aktualnych zapisach na zajęcia w macierzystych uczelniach (na tej podstawie przyjmij założenia co do rozmiaru i dystrybucji danych).

Jakie indeksy najlepiej utworzyć w PostgreSQL aby przyspieszyć wykonywanie poniższego zapytania? Który indeks będzie najbardziej przydatny jeśli możesz utworzyć tylko jeden (oczywiście nie licząc już istniejących indeksów dla kluczy głównych)? Identyfikatory `:gender`, `:age`, `:course` oznaczają parametry uzupełniane przez aplikację.

Przedstaw kilka propozycji, wygeneruj dane testowe (zastanów się jak), przeprowadź odpowiednie eksperymenty i podaj ich wyniki.

```
SELECT * FROM person p JOIN zapisy z ON (p.id=z.sid)
WHERE p.gender=:gender AND EXTRACT(year from age(p.birthday))=:age AND
      z.course=:course;
```