Etap 2B

Grupa B6 - Loty

Hanna Grodzicka

Mateusz Najda

Aktywne reguły

1. **Usunięcie pracowników**

   Zestaw C (insert ⇒ delete)

   Opis: Usunięcie pracowników zatrudnionych przed powstaniem ich linii lotniczej.

   Zdarzenia inicjujące: Dodanie linii lotniczej.

   Warunki uruchomienia: —

   Działanie: Po dodaniu nowej linii lotniczej następuje usunięcie wszystkich pracowników zatrudnionych przed założeniem linii lotniczej, do której są przypisani, tj. u których `employee.employment_date` < `airline.establishment_date`.

   Szacunek złożoności: Wybranie 15,960 pracowników (1 złączenie + 1 instrukcja warunkowa), a następnie ich usunięcie.

   ```
   CREATE TRIGGER delete_employees
   AFTER INSERT ON airline
   BEGIN
     DELETE (SELECT *
           FROM employee e
             INNER JOIN airline a ON a.id = e.airline_id
           WHERE e.employment_date < a.establishment_date);
   END;
   ```

   Przykładowe zdarzenie uruchamiające:

   ```
   INSERT INTO AIRLINE (id, company, code, country, establishment_date)
   VALUES (3001, "Raven", "RA-47", "Brazil", null);
   ```

2. **Okresowe dodawanie nowych lotów (reguła czasowa)**

   Opis: Okresowe dodawanie nowych lotów do tabeli `Flight`.

   Zdarzenia inicjujące: Upływ określonej jednostki czasowej, czyli 1 godziny.

Warunki uruchomienia: W tabeli `Flight` musi być przynajmniej 1 lot, w tabeli `Plane` przynajmniej 1 samolot oraz w tabeli `Airport` przynajmniej 1 lotnisko.

Działanie: Dodanie lotu do tabeli `Flight`, którego kolumny są częściowo tworzone na podstawienie istniejących danych z tabel `Plane`, `Flight` i `Airport`.

Szacunek złożoności: Przy dodaniu lotu następują 3 proste podzapytania (min/max), 1 instrukcja warunkowa, 2-krotne pobranie czasu.

```
BEGIN
DBMS_SCHEDULER.CREATE_JOB (
   job_name             => 'add_flights',
   job_type             => 'PLSQL_BLOCK',
   job_action           => 'BEGIN
INSERT INTO flight
(id, flight_number, destination_country, destination_city,
 departure, arrival, passengers, status, delay, is_national,
 plane_id, airport_id)
SELECT f.id+1, "123", "Brazil", "Campinas",
  SYSDATE, SYSDATE + 3, 123, "CONFIRMED", 0,
  (CASE WHEN f.is_national = 0 THEN 1 ELSE 0 END),
  (SELECT MAX(id) FROM plane),
  (SELECT MIN(id) FROM airport)
FROM flight f
WHERE f.id = (SELECT MAX(id) FROM flight);
END;',
   start_date           => SYSTIMESTAMP,
   repeat_interval      => 'freq=hourly; byminute=0; bysecond=0;',
   enabled              =>  TRUE,
   comments             => 'Add new flight every hour');
END;
```

3. **Dodawanie nowych pracowników po zakupie samolotu**

Zestaw A (insert ⇒ insert)

Opis: Dodanie pracowników przy dodawaniu samolotu do floty

Zdarzenia inicjujące: Dodanie samolotu

Warunki uruchomienia: —

Działanie: Dodanie 3,000 nowych pracowników do tabeli `Employee`

Szacunek złożoności: wybranie 3,000 pracowników zgrupowanych wg linii lotniczej; podczas dodania nowego pracownika na podstawie wybranego wykonanie prostych operacji (`MAX`, 2*`MIN`, 2*`MEDIAN`, `ROUND`, `AVG`, `VARIANCE`, `COUNT`) na wybranym zbiorze.

```
CREATE TRIGGER insert_employees
AFTER INSERT ON flight
BEGIN

  INSERT INTO employee
  (id,
   name,
```

```
    surname,
    birthdate,
    gender,
    occupation,
    employment_date,
    flown_hours,
    airline_id)
  SELECT employee_id_seq.NEXTVAL, a, b, c, d, f, g, h, i
  FROM (SELECT MAX(e.name) a,
               MIN(e.surname) b,
               MEDIAN(birthdate) + 10 c,
               ROUND(AVG(CASE WHEN e.gender = 1 THEN 1 ELSE 0 END)) d,
               MIN(e.occupation) f,
               MEDIAN(employment_date) g,
               VARIANCE(flown_hours) / COUNT(*) h,
               e.airline_id i
        FROM employee e
        GROUP BY e.airline_id);

END;
```

Przykładowe zdarzenie uruchamiające:

```
INSERT INTO flight
(id, flight_number, destination_country, destination_city,
 departure, arrival, passengers, status, delay, is_national,
 plane_id, airport_id)
VALUES
(30001, "flight_number", "destination_country", "destination_city",
to_date('2019-12-11 09:21:41', 'YYYY-MM-DD HH24:MI:SS'),
to_date('2019-12-11 09:21:41', 'YYYY-MM-DD HH24:MI:SS'),
1, "status", 14.88, 0, 1, 1);
```

4. **Aktualizacja liczby przelatanych godzin u pracowników**

Zestaw A (update ⇒ update)

Opis: Aktualizacja liczby przelatanych godzin.

Zdarzenia inicjujące: Zaktualizowanie czasu przylotu (`arrival`) rekordu z tabeli `Flight`.

Warunki uruchomienia: —

Działanie: Obliczenie długości lotu na podstawie daty i dodanie tej wartości do `flown_hours` każdego pracownika odbywającego ten lot.

Szacunek złożoności: Wybranie 24,000 pracowników (w tym dwa złączenia tabel), wyliczenie dla każdego z nich sumę (`SUM`) długości trwania odbytych lotów i zaktualizowanie tej wartości.

```
CREATE TRIGGER update_flown_hours
AFTER UPDATE OF arrival ON flight
BEGIN

  UPDATE employee eu
```

```
  SET eu.flown_hours = (SELECT SUM(24 * (f.arrival - f.departure))
                        FROM flight f
                        JOIN flight_employee fe ON f.id = fe.flight_id
                        JOIN employee e ON e.id = fe.employee_id);

END;
```

Przykładowe zdarzenie uruchamiające:

```
UPDATE flight
SET arrival = to_date('2019-12-11 09:21:41', 'YYYY-MM-DD HH24:MI:SS');
```

5. **Sprawdzenie, czy można dodać lot do tabeli (check constraint)**

   Opis: Anulowanie dodawania lotu, jeśli liczba pasażerów przekracza liczbę dostępnych miejsc w samolocie obsługującym lot.

   Zdarzenia inicjujące: Dodanie lotu.

   Warunki uruchomienia: Liczba pasażerów przekracza liczbę dostępnych miejsc w samolocie przypisanym do lotu.

   Działanie: Dodawanie lotu jest anulowane.

   Szacunek złożoności: Wybranie 1 samolotu na podstawie jego `id` oraz jedna instrukcja warunkowa.

```
CREATE TRIGGER add_flight_when
BEFORE INSERT ON flight
DECLARE max_passengers float;
BEGIN

  SELECT seat_count INTO max_passengers FROM plane
  WHERE :new.plane_id = plane.id;

  IF (max_passengers < :new.passengers)
    THEN raise_application_error(-20000, "Number of passengers exceeds number of seats!")
  END IF;

END;
```

Przykładowe zdarzenie powodujące błąd:

```
INSERT INTO flight
(id, flight_number, destination_country, destination_city,
 departure, arrival, passengers, status, delay, is_national,
 plane_id, airport_id)
VALUES
(30001, "flight_number", "destination_country", "destination_city",
to_date('2019-12-11 09:21:41', 'YYYY-MM-DD HH24:MI:SS'),
to_date('2019-12-11 09:21:41', 'YYYY-MM-DD HH24:MI:SS'),
2137, "status", 14.88, 0, 1, 1);
```

Zestaw A

Aa Numer próby	# Próbka obciążenia [s]	# Aktywne reguły [s]
1	5.31	5.39
2	5.7	5.45
3	5.65	5.88
4	5.51	5.55
5	5.61	5.48

$$\sigma_{probek} = 0.13792751719653, \quad \sigma^2_{probek} = 0.019024$$

$$\sigma_{regul} = 0.17285832349066, \quad \sigma^2_{regul} = 0.02988$$

Zestaw B

Aa Numer próby	# Próbka obciążenia [s]	# Aktywne reguły [s]
1	3.5	4.48
2	3.03	3.63
3	3.2	4.1
4	3.16	4.05
5	2.99	3.62

$$\sigma_{probek} = 0.17984437717093, \quad \sigma^2_{probek} = 0.032344$$

$$\sigma_{regul} = 0.3228993651279, \quad \sigma^2_{regul} = 0.104264$$

Zestaw C

Aa Numer próby	# Próbka obciążenia [s]	# Aktywne reguły [s]
1	27.3	30.37
2	29.38	29.08
3	26.81	29.72
4	26.52	31.25
5	26.52	27.33

$$\sigma_{probek} = 1.0754645507872, \quad \sigma^2_{probek} = 1.156624$$

$$\sigma_{regul} = 1.3221648913808, \quad \sigma^2_{regul} = 1.74812$$

Średnie pomiary czasu

Zestaw	Próbka obciążenia [s]	Aktywne reguły [s]
A	5.556 ±0.121	5.55 ±0.152
B	3.176 ±0.158	3.976 ±0.283
C	27.306 ±0.943	29.55 ±1.159

Błędy pomiarowe są podane przy poziomie zaufania 95% (istotność równa 5%).

Eksperymenty

Eksperyment 1

W celu przetestowania braku własności stopu zdefiniowano 3 wyzwalacze, wszystkie wykonywane po dodaniu rekordu do testowej tabeli. Każdy z wyzwalaczy także dodaje kolejny rekord.

```
create table foo (bar varchar2(255), baz timestamp);

create or replace trigger triggerA after insert on foo
begin
    insert into foo values ('A', current_timestamp);
    dbms_output.put_line('A');
end;
/

create or replace trigger triggerB after insert on foo
begin
    insert into foo values ('B', current_timestamp);
    dbms_output.put_line('B');
end;
/

create or replace trigger triggerC after insert on foo
begin
    insert into foo values ('C', current_timestamp);
    dbms_output.put_line('C');
end;
/

insert into foo values ('test', current_timestamp);

select * from foo;

drop trigger triggerC;
drop trigger triggerB;
drop trigger triggerA;

drop table foo;
```

Wynik wywołania powyższego kodu:

```
Error starting at line : 24 in command -
insert into foo values ('test', current_timestamp)
Error report -
ORA-00036: maximum number of recursive SQL levels (50) exceeded
ORA-06512: at "SYSTEM.TRIGGERC", line 2
ORA-04088: error during execution of trigger 'SYSTEM.TRIGGERC'
ORA-06512: at "SYSTEM.TRIGGERC", line 2
ORA-04088: error during execution of trigger 'SYSTEM.TRIGGERC'
ORA-06512: at "SYSTEM.TRIGGERC", line 2
ORA-04088: error during execution of trigger 'SYSTEM.TRIGGERC'
```

Eksperyment pokazuje, że **maksymalna liczba wywołań w bazie danych Oracle wynosi 50**. Po wykonaniu większej liczby pojawia się błąd o przekroczeniu maksymalnej liczby wywołań rekursywnych. Warto zauważyć, że **w przypadku wystąpienia błędu transakcja nie zostanie zapisana**. Wszystkie zmiany nie zostaną wykonane, w naszym przykładzie tabela `foo` będzie na końcu pusta.

Eksperyment 2

Zdefiniowano 3 wyzwalacze, wszystkie uruchamiane przy operacji dodania rekordu do tabeli testowej. Każdy z wyzwalaczy wypisuje tekst na strumień wyjściowy.

```
create table foo (bar varchar2(255));

create or replace trigger triggerA after insert on foo
begin
    dbms_output.put_line('A');
end;
/

create or replace trigger triggerB after insert on foo
follows triggerA
begin
    dbms_output.put_line('B');
end;
/

create or replace trigger triggerC after insert on foo
follows triggerB
begin
    dbms_output.put_line('C');
end;
/

insert into foo values ('test');

drop trigger triggerC;
drop trigger triggerB;
drop trigger triggerA;
```

```
drop table foo;
```

W bazie danych Oracle do sterowania kolejnością wykonywania służy słowo kluczowe `FOLLOWS`. Według dokumentacji, bez niego baza danych **nie zapewnia ustalonej kolejności wykonywania reguł**. Jednak w naszym przypadku, po usunięciu słowa kluczowego `FOLLOWS`, kolejność była zawsze taka sama (C B A).

Wynik działania powyższego kodu:

```
A
B
C
```

Po usunięciu słów kluczowych `FOLLOWS`:

```
C
B
A
```