Bazy Danych

Piotr Cięgotura

298740

Ćwiczenie: wydajność złączeń i zagnieżdżeń

1. Parametry komputera:

   CPU: Intel(R) Core(TM) i7-6500U CPU @ 2.5GHz
   RAM: Pamięć DDR3 8GB (1600 MHZ),
   SSD: SSDPR-CX400-256-G2
   S.O.: Windows 10.
   Jako systemy zarządzania bazami danych wybrano oprogramowanie wolno dostępne MySQL Workbench, mysql Ver 15.1 Distrib 10.4.28-MariaDB, for Win64 (AMD64),

2. Kod:

```
CREATE DATABASE Dane;

--Tabele dziesięć i milion:

CREATE TABLE dane.Dziesienc (

cyfra INT,
bit  INT

);

DELIMITER $$
CREATE PROCEDURE WypelnijLosowo()
 BEGIN
DECLARE no INT;
  SET no = 0;
  loop1: LOOP

    INSERT INTO dane.Dziesienc (cyfra, bit) VALUES ((no), ROUND(RAND()));
SET no = no +1;
    IF no >9 THEN
     LEAVE loop1;
    END IF;
 END LOOP loop1;
SELECT no;
END $$
DELIMITER ;

CALL WypelnijLosowo();

CREATE TABLE dane.Milion(liczba int,cyfra int, bit int);
INSERT INTO  dane.Milion SELECT a1.cyfra +10* a2.cyfra +100*a3.cyfra + 1000*a4.cyfra +
10000*a5.cyfra + 10000*a6.cyfra AS liczba , a1.cyfra AS cyfra, a1.bit AS bit
FROM dane.Dziesienc a1, dane.Dziesienc a2, dane.Dziesienc a3, dane.Dziesienc a4,
dane.Dziesienc a5, dane.Dziesienc a6 ;
```

```
--Znormalizowana geochronologiczna baza danych:

CREATE TABLE dane.GeoEon (

id_eon INT AUTO_INCREMENT PRIMARY KEY,
nazwa_eon  VARCHAR(25)

);

INSERT INTO dane.geoeon (nazwa_eon) VALUES ('FANEROZOIK');

#drufa tabela

CREATE TABLE dane.geoera (

id_era INT AUTO_INCREMENT PRIMARY KEY,
id_eon INT NOT NULL,
FOREIGN KEY (id_eon) REFERENCES dane.geoeon(id_eon),
nazwa_era  VARCHAR(25)

);

INSERT INTO dane.geoera (id_eon,nazwa_era) VALUES
(1,'Kenozoik'),(1,'Mezozoik'),(1,'Paleozoik');

#Trzecia tabela

CREATE TABLE dane.geookres (

id_okres INT AUTO_INCREMENT PRIMARY KEY,
id_era INT NOT NULL,
FOREIGN KEY (id_era) REFERENCES dane.geoera(id_era),
nazwa_okres  VARCHAR(25)

);

INSERT INTO dane.geookres (id_era,nazwa_okres) VALUES
(1,'Czwartorząd'),
(1,'Neogen'),
(1,'Paleogen'),
(2,'Kread'),
(2,'Jura'),
(2,'Trias'),
(3,'Perm'),
(3,'Karbon'),
(3,'Dewon');
```

#czwarta tabela

```
CREATE TABLE dane.geoepoka (

id_epoka INT AUTO_INCREMENT PRIMARY KEY,
id_okres INT NOT NULL,
FOREIGN KEY (id_okres) REFERENCES dane.geookres(id_okres),
nazwa_epoka  VARCHAR(25)

);

INSERT INTO dane.geoepoka (id_okres,nazwa_epoka) VALUES
(1,'Halocen'),
(1,'Plejstocen'),
(2,'Pilocen'),
(2,'Miocen'),
(3,'Oligocen'),
(3,'Eocen'),
(3,'Paleocen'),
(4,'Gorna'),
(4,'Dolna'),
(5,'Gorna'),
(5,'Srodkowa'),
(5,'Dolna'),
(6,'Gorna'),
(6,'Srodkowa'),
(6,'Dolna'),
(7,'Gorny'),
(7,'Dolny'),
(8,'Gorny'),
(8,'Dolny'),
(9,'Gorny'),
(9,'Srodkowy'),
(9,'Dolny');

CREATE TABLE dane.geopietro (

id_pietro INT AUTO_INCREMENT PRIMARY KEY,
id_epoka INT NOT NULL,
FOREIGN KEY (id_epoka) REFERENCES dane.geoepoka(id_epoka),
nazwa_pietro  INT

);
```

--procedura do wypelnienia tabeli geopietro:

```
DELIMITER $$
CREATE PROCEDURE WypelnijGeopietro()
 BEGIN
DECLARE no INT;
  SET no = 0;
  loop1: LOOP
IF no >0 AND no <=10000 THEN
   INSERT INTO dane.geopietro (id_epoka,nazwa_pietro) VALUES (1, no);
END IF;
IF no >10000 AND no <=1800000 THEN
   INSERT INTO dane.geopietro (id_epoka,nazwa_pietro) VALUES (2, no);
END IF;
IF no >1800000 AND no <=5333000 THEN
   INSERT INTO dane.geopietro (id_epoka,nazwa_pietro) VALUES (3, no);
END IF;
IF no >5333000 AND no <=22500000 THEN
   INSERT INTO dane.geopietro (id_epoka,nazwa_pietro) VALUES (4, no);
END IF;
IF no >22500000 AND no <=33900000 THEN
   INSERT INTO dane.geopietro (id_epoka,nazwa_pietro) VALUES (5, no);
END IF;
IF no >33900000 AND no <=56000000 THEN
   INSERT INTO dane.geopietro (id_epoka,nazwa_pietro) VALUES (6, no);
END IF;
IF no >56000000 AND no <=65000000 THEN
   INSERT INTO dane.geopietro (id_epoka,nazwa_pietro) VALUES (7, no);
END IF;
IF no >65000000 AND no <=100500000 THEN
   INSERT INTO dane.geopietro (id_epoka,nazwa_pietro) VALUES (8, no);
END IF;
IF no >100500000 AND no <=140000000 THEN
   INSERT INTO dane.geopietro (id_epoka,nazwa_pietro) VALUES (9, no);
END IF;
IF no >140000000 AND no <=163500000 THEN
   INSERT INTO dane.geopietro (id_epoka,nazwa_pietro) VALUES (10, no);
END IF;
IF no >163500000 AND no <=174100000 THEN
   INSERT INTO dane.geopietro (id_epoka,nazwa_pietro) VALUES (11, no);
END IF;
IF no >174100000 AND no <=195000000 THEN
   INSERT INTO dane.geopietro (id_epoka,nazwa_pietro) VALUES (12, no);
END IF;
IF no >195000000 AND no <=237000000 THEN
   INSERT INTO dane.geopietro (id_epoka,nazwa_pietro) VALUES (13, no);
END IF;
IF no >237000000 AND no <=247000000 THEN
```

```
  INSERT INTO dane.geopietro (id_epoka,nazwa_pietro) VALUES (14, no);
END IF;
IF no >247000000 AND no <=251000000 THEN
  INSERT INTO dane.geopietro (id_epoka,nazwa_pietro) VALUES (15, no);
END IF;
IF no >251000000 AND no <=273000000 THEN
  INSERT INTO dane.geopietro (id_epoka,nazwa_pietro) VALUES (16, no);
END IF;
IF no >273000000 AND no <=299000000 THEN
  INSERT INTO dane.geopietro (id_epoka,nazwa_pietro) VALUES (17, no);
END IF;
IF no >299000000 AND no <=318000000 THEN
  INSERT INTO dane.geopietro (id_epoka,nazwa_pietro) VALUES (18, no);
END IF;
IF no >318000000 AND no <=359000000 THEN
  INSERT INTO dane.geopietro (id_epoka,nazwa_pietro) VALUES (19, no);
END IF;
IF no >359000000 AND no <=385000000 THEN
  INSERT INTO dane.geopietro (id_epoka,nazwa_pietro) VALUES (20, no);
END IF;
IF no >385000000 AND no <=391000000 THEN
  INSERT INTO dane.geopietro (id_epoka,nazwa_pietro) VALUES (21, no);
END IF;
IF no >391000000 AND no <=416000000 THEN
  INSERT INTO dane.geopietro (id_epoka,nazwa_pietro) VALUES (22, no);
END IF;

          SET no = no +416;
   IF no >416000000 THEN
    LEAVE loop1;
   END IF;
 END LOOP loop1;
SELECT no;
END $$
DELIMITER ;

CALL WypelnijGeopietro();


--Utworzenie tabeli zdemoralizowanej:

CREATE TABLE dane.geotabela AS (SELECT * FROM geopietro NATURAL JOIN geoepoka
NATURAL
JOIN geookres NATURAL JOIN geoera NATURAL JOIN geoeon );
```

```
--Utworzenie tabeli milion z indeksem
CREATE TABLE d.milionindex AS
SELECT * FROM d.milion;

ALTER TABLE d.milionindex
ADD INDEX ind (liczba, cyfra, bit);
```

3. Badanie:

   W badaniu wydajności baz danych użyłem 4 zapytań, odpowiednio na tabeli danych bez indeksów i z indeksami.

```
SELECT COUNT(*) FROM dane.milion INNER JOIN dane.geotabela ON
(mod(dane.milion.liczba,68)=(dane.geotabela.id_pietro));


SELECT COUNT(*) FROM Milion INNER JOIN GeoPietro ON
(mod(Milion.liczba,68)=GeoPietro.id_pietro) NATURAL JOIN GeoEpoka NATURAL JOIN
GeoOkres NATURAL JOIN GeoEra NATURAL JOIN GeoEon;

SELECT COUNT(*) FROM Milion WHERE mod(Milion.liczba,68)=
(SELECT id_pietro FROM GeoTabela WHERE mod(Milion.liczba,68)=(id_pietro));

SELECT COUNT(*) FROM d.milion WHERE mod(d.milion.liczba,68) IN
(SELECT d.geopietro.id_pietro FROM d.geopietro NATURAL JOIN d.geoepoka NATURAL JOIN
d.geookres NATURAL JOIN d.geoera NATURAL JOIN d.geoeon);
```

4. Wyniki

	1. Zapytanie		2. Zapytanie		3. Zapytanie		4. Zapytanie	
Bez indeksów	MIN	ŚR	MIN	ŚR	MIN	ŚR	MIN	ŚR
MySQL	5.61[s]	6.107	4.844	5.0784	25337.672(po tym czasie zapytanie dalej się wykonywało, anulowałem je)	25337.672	3.562	4.1718
Z Indeksami								
MySQL	5.781	6.5442	3.047	3.2718	15117.5(po tym czasie zapytanie dalej się wykonywało, anulowałem je)	15117.5	3.438	3.4846

5. Wnioski

Widać że wprowadzenie indeksów poprawiło wyniki wykonywania się zapytań, z pominięciem pierwszego przypadku w którym długość wykonywania zapytania się zwiększyła. Widać też że normalizacja tabeli przyspiesza szybkość wykonywania zapytania, porównując 3. zapytanie z 4. występuje przepaść.