Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчет

По дисциплине: “Управление информационными ресурсами”

Лабораторная работа №4

“Исследование принципов партицирования баз данных”

Выполнил:

ст.гр. ИС/б-17-2

Долженко И.А.

Проверил:

Дрозин А.Ю.

Севастополь

2020

1 ЦЕЛЬ РАБОТЫ

Исследовать способы партицирования таблиц баз данных и их влияние на скорость доступа к данным. Изучить основы партицирования на примере MySQL.

2 ПОСТАНОВКА ЗАДАЧИ

Используя методику, описанную в приложении, создать простой HTTP + MySQL сервер и реализовать партицирование.

|  |  |  |
| --- | --- | --- |
| Вариант | Название таблицы (в скобках указан перечень полей) | Кол-во строк в таблице |
| 2 | blogs (id, title, posts\_count) | 200, 700, 1300, 1900 |

3 ТЕКСТ ПРОГРАММЫ

Главный файл index.js:

const express = require("express");

const mysql = require("mysql2/promise");

const now = require("performance-now");

const server\_host = "localhost";

const server\_port = "3000";

const conn = {

    host: "localhost",

    database: "lab4",

    user: "root",

    password: "root"

};

const test\_id = 100;

const queries = [

    {

        table: "transactions\_no\_part\_200",

        sql: `SELECT SQL\_NO\_CACHE \* FROM transactions\_no\_part\_200 LIMIT 1`

    },

    {

        table: "transactions\_no\_part\_200",

        sql: `SELECT SQL\_NO\_CACHE \* FROM transactions\_no\_part\_200 WHERE id = ${test\_id} LIMIT 1`

    },

    {

        table: "transactions\_part\_200",

        sql: `SELECT SQL\_NO\_CACHE \* FROM transactions\_part\_200 LIMIT 1`

    },

    {

        table: "transactions\_part\_200",

        sql: `SELECT SQL\_NO\_CACHE \* FROM transactions\_part\_200 WHERE id = ${test\_id} LIMIT 1`

    },

    {

        table: "transactions\_no\_part\_700",

        sql: `SELECT SQL\_NO\_CACHE \* FROM transactions\_no\_part\_700 LIMIT 1`

    },

    {

        table: "transactions\_no\_part\_700",

        sql: `SELECT SQL\_NO\_CACHE \* FROM transactions\_no\_part\_700 WHERE id = ${test\_id} LIMIT 1`

    },

    {

        table: "transactions\_part\_700",

        sql: `SELECT SQL\_NO\_CACHE \* FROM transactions\_part\_700 LIMIT 1`

    },

    {

        table: "transactions\_part\_700",

        sql: `SELECT SQL\_NO\_CACHE \* FROM transactions\_part\_700 WHERE id = ${test\_id} LIMIT 1`

    },

    {

        table: "transactions\_no\_part\_1300",

        sql: `SELECT SQL\_NO\_CACHE \* FROM transactions\_no\_part\_1300 LIMIT 1`

    },

    {

        table: "transactions\_no\_part\_1300",

        sql: `SELECT SQL\_NO\_CACHE \* FROM transactions\_no\_part\_1300 WHERE id = ${test\_id} LIMIT 1`

    },

    {

        table: "transactions\_part\_1300",

        sql: `SELECT SQL\_NO\_CACHE \* FROM transactions\_part\_1300 LIMIT 1`

    },

    {

        table: "transactions\_part\_1300",

        sql: `SELECT SQL\_NO\_CACHE \* FROM transactions\_part\_1300 WHERE id = ${test\_id} LIMIT 1`

    },

    {

        table: "transactions\_no\_part\_1900",

        sql: `SELECT SQL\_NO\_CACHE \* FROM transactions\_no\_part\_1900 LIMIT 1`

    },

    {

        table: "transactions\_no\_part\_1900",

        sql: `SELECT SQL\_NO\_CACHE \* FROM transactions\_no\_part\_1900 WHERE id = ${test\_id} LIMIT 1`

    },

    {

        table: "transactions\_part\_1900",

        sql: `SELECT SQL\_NO\_CACHE \* FROM transactions\_part\_1900 LIMIT 1`

    },

    {

        table: "transactions\_part\_1900",

        sql: `SELECT SQL\_NO\_CACHE \* FROM transactions\_part\_1900 WHERE id = ${test\_id} LIMIT 1`

    }

];

async function run() {

    const db = *await* mysql.createConnection(conn);

    const app = express();

    app.set("view engine", "pug");

    app.get(

        "/",

        wrapAsync(async (req, res, next) => {

            let results = [];

*for* (let i in queries) {

                const [row] = *await* db.execute(

                    `SELECT COUNT(\*) AS n FROM ${queries[i].table}`

                );

                start\_time = now();

                const rows = *await* db.execute(queries[i].sql);

                end\_time = now();

                results.push({

                    n: row[0].n,

                    sql: queries[i].sql,

                    table: queries[i].table,

                    time: String((end\_time - start\_time).toFixed(3))

                });

            }

            res.render("results", { header: `Query results`, results: results });

        })

    );

    app.listen(server\_port, server\_host, () => {

        console.log(`Server running at http://${server\_host}:${server\_port}/`);

    });

}

function wrapAsync(fn) {

*return* function (req, res, next) {

        fn(req, res, next).catch(next);

    };

}

run();

Файл database.sql:

*-- Удалить старую базу и создать новую*

DROP DATABASE IF EXISTS lab4;

CREATE DATABASE lab4;

*-- Использовать созданную базу*

USE lab4;

*-- blogs (id, title, posts\_count)*

*-- Создать таблицу без партицирования*

CREATE TABLE IF NOT EXISTS `transactions\_no\_part\_200` (

    `id` INTEGER NOT NULL AUTO\_INCREMENT,

    `title` VARCHAR(20) NOT NULL DEFAULT 'this title',

    `posts\_count` INTEGER NOT NULL,

    PRIMARY KEY (`id`)

) ENGINE = InnoDB;

*-- Создать таблицу с партицированием*

CREATE TABLE IF NOT EXISTS `transactions\_part\_200` (

    `id` INTEGER NOT NULL AUTO\_INCREMENT,

    `title` VARCHAR(20) NOT NULL DEFAULT 'this title',

    `posts\_count` INTEGER NOT NULL,

    PRIMARY KEY (`id`)

) ENGINE = InnoDB

PARTITION BY HASH(`id`)

PARTITIONS 4;

*-- Создать таблицу без партицирования*

CREATE TABLE IF NOT EXISTS `transactions\_no\_part\_700` (

    `id` INTEGER NOT NULL AUTO\_INCREMENT,

    `title` VARCHAR(20) NOT NULL DEFAULT 'this title',

    `posts\_count` INTEGER NOT NULL,

    PRIMARY KEY (`id`)

) ENGINE = InnoDB;

*-- Создать таблицу с партицированием*

CREATE TABLE IF NOT EXISTS `transactions\_part\_700` (

    `id` INTEGER NOT NULL AUTO\_INCREMENT,

    `title` VARCHAR(20) NOT NULL DEFAULT 'this title',

    `posts\_count` INTEGER NOT NULL,

    PRIMARY KEY (`id`)

) ENGINE = InnoDB

PARTITION BY HASH(`id`)

PARTITIONS 4;

*-- Создать таблицу без партицирования*

CREATE TABLE IF NOT EXISTS `transactions\_no\_part\_1300` (

    `id` INTEGER NOT NULL AUTO\_INCREMENT,

    `title` VARCHAR(20) NOT NULL DEFAULT 'this title',

    `posts\_count` INTEGER NOT NULL,

    PRIMARY KEY (`id`)

) ENGINE = InnoDB;

*-- Создать таблицу с партицированием*

CREATE TABLE IF NOT EXISTS `transactions\_part\_1300` (

    `id` INTEGER NOT NULL AUTO\_INCREMENT,

    `title` VARCHAR(20) NOT NULL DEFAULT 'this title',

    `posts\_count` INTEGER NOT NULL,

    PRIMARY KEY (`id`)

) ENGINE = InnoDB

PARTITION BY HASH(`id`)

PARTITIONS 4;

*-- Создать таблицу без партицирования*

CREATE TABLE IF NOT EXISTS `transactions\_no\_part\_1900` (

    `id` INTEGER NOT NULL AUTO\_INCREMENT,

    `title` VARCHAR(20) NOT NULL DEFAULT 'this title',

    `posts\_count` INTEGER NOT NULL,

    PRIMARY KEY (`id`)

) ENGINE = InnoDB;

*-- Создать таблицу с партицированием*

CREATE TABLE IF NOT EXISTS `transactions\_part\_1900` (

    `id` INTEGER NOT NULL AUTO\_INCREMENT,

    `title` VARCHAR(20) NOT NULL DEFAULT 'this title',

    `posts\_count` INTEGER NOT NULL,

    PRIMARY KEY (`id`)

) ENGINE = InnoDB

PARTITION BY HASH(`id`)

PARTITIONS 4;

*-- Создать процедуру для генерации случайных данных*

DELIMITER //

CREATE PROCEDURE `generate\_data\_200` (IN items\_count INT)

BEGIN

    DECLARE i INT DEFAULT 0;

    WHILE i < items\_count DO

        INSERT INTO `transactions\_no\_part\_200` (`posts\_count`) VALUES (RAND() \* 200);

        SET i = i + 1;

    END WHILE;

*-- Скопировать те же данные в партицированную таблицу*

    INSERT INTO `transactions\_part\_200`

        SELECT \* FROM `transactions\_no\_part\_200`;

END//

DELIMITER ;

*-- Создать процедуру для генерации случайных данных*

DELIMITER //

CREATE PROCEDURE `generate\_data\_700` (IN items\_count INT)

BEGIN

    DECLARE i INT DEFAULT 0;

    WHILE i < items\_count DO

        INSERT INTO `transactions\_no\_part\_700` (`posts\_count`) VALUES (RAND() \* 700);

        SET i = i + 1;

    END WHILE;

*-- Скопировать те же данные в партицированную таблицу*

    INSERT INTO `transactions\_part\_700`

        SELECT \* FROM `transactions\_no\_part\_700`;

END//

DELIMITER ;

*-- Создать процедуру для генерации случайных данных*

DELIMITER //

CREATE PROCEDURE `generate\_data\_1300` (IN items\_count INT)

BEGIN

DECLARE i INT DEFAULT 0;

    WHILE i < items\_count DO

        INSERT INTO `transactions\_no\_part\_1300` (`posts\_count`) VALUES (RAND() \* 1300);

        SET i = i + 1;

    END WHILE;

*-- Скопировать те же данные в партицированную таблицу*

    INSERT INTO `transactions\_part\_1300`

        SELECT \* FROM `transactions\_no\_part\_1300`;

END//

DELIMITER ;

*-- Создать процедуру для генерации случайных данных*

DELIMITER //

CREATE PROCEDURE `generate\_data\_1900` (IN items\_count INT)

BEGIN

    DECLARE i INT DEFAULT 0;

    WHILE i < items\_count DO

        INSERT INTO `transactions\_no\_part\_1900` (`posts\_count`) VALUES (RAND() \* 1900);

        SET i = i + 1;

    END WHILE;

*-- Скопировать те же данные в партицированную таблицу*

    INSERT INTO `transactions\_part\_1900`

        SELECT \* FROM `transactions\_no\_part\_1900`;

END//

DELIMITER ;

*-- Заполнить таблицу данными*

CALL generate\_data\_200(200);

CALL generate\_data\_700(700);

CALL generate\_data\_1300(1300);

CALL generate\_data\_1900(1900);

*-- Удалить процедуру*

DROP PROCEDURE `generate\_data\_200`;

DROP PROCEDURE `generate\_data\_700`;

DROP PROCEDURE `generate\_data\_1300`;

DROP PROCEDURE `generate\_data\_1900`;

Файл results.pug:

html

    head

        title='Лабораторная работа 4'

        style

            include style.css

    body

        h2=header

        table

            tr

                th='Records'

                th='SQL query'

                th='SQL table'

                th='Time (µs)'

                for res in results

                    tr

                        td=res.n

                        td=res.sql

                        td=res.table

                        td=res.time

        br

Файл style.css:

table {

    border-collapse: collapse;

}

th,

td {

    border: 1px solid blue;

    padding: 5px;

}

4 РЕЗУЛЬТАТЫ

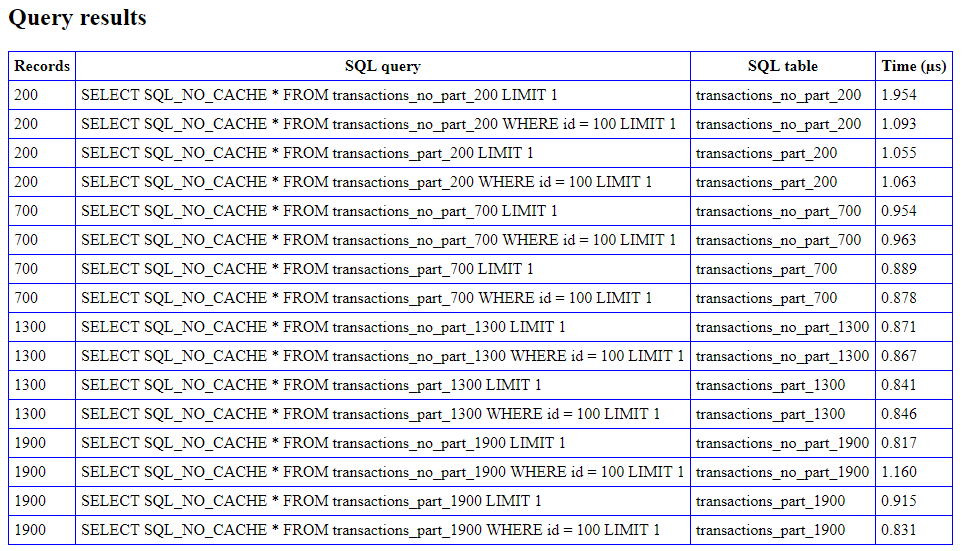


Рисунок 1 – Таблица с результатами выполнения запросов

ВЫВОД

В ходе выполнения лабораторной работы были исследованы способы партицирования таблиц баз данных и их влияние на скорость доступа к данным. Изучены основы партицирования на примере MySQL.