Modul Praktikum 1

Teknologi Basis Data

Persiapan Tuning

PROGRAM STUDI SAINS DATA FAKULTAS SAINS

INSTITUT TEKNOLOGI SUMATERA

2025

Persiapan Tuning

1. Tujuan

a. Memahami konsep dan cara melakukan tunning dengan indexing dan setting DBMS
b. Mampu membandingkan response time sebelum dan sesudah melakukan indexing pada database

2. Dasar Teori

2.1. Tunning Database System

Tunnig merupakan salah satu cara pada database system untuk meningkatkan performa kinerja database Tunning database system dapat dilakukan dengan pendekatan beberapa level:

a. Hardware
b. Sistem Operasi
c. Database Manajement System
d. Schema basis data
e. Indexing
f. Query

2.2. Indexing

Indeks untuk sebuah file dalam sistem basis data bekerja dengan cara yang hampir sama seperti indeks dalam buku teks ini. Jika kita ingin mempelajari topik tertentu (yang ditentukan oleh sebuah kata atau frasa) dalam buku ini, kita dapat mencari topik tersebut di indeks yang terletak di bagian belakang buku, menemukan halaman-halaman di mana topik tersebut muncul, lalu membaca halaman-halaman tersebut untuk mendapatkan informasi yang kita cari. Kata-kata dalam indeks tersusun dalam urutan yang teratur, sehingga memudahkan kita menemukan kata yang diinginkan. Selain itu, ukuran indeks jauh lebih kecil dibandingkan dengan seluruh buku, sehingga pencarian menjadi lebih mudah dan cepat. Konsep indeks dalam sistem basis data dengan membandingkannya dengan indeks pada buku teks.

1. Perbandingan dengan Indeks Buku
    - Saat mencari suatu topik dalam buku, kita bisa melihat indeks di bagian belakang buku.
    - Indeks tersebut berisi kata kunci (keyword) yang sudah diurutkan secara alfabetis.
    - Setelah menemukan kata kunci, kita melihat nomor halaman yang berkaitan dan membaca isinya.
2. Indeks dalam Basis Data
    - Indeks dalam basis data bekerja dengan cara yang mirip dengan indeks buku.

- Saat mencari data dalam database, indeks membantu menemukan lokasi penyimpanan data dengan lebih cepat.
- Indeks tersusun dalam urutan tertentu (misalnya alfabetis atau numerik) untuk mempercepat pencarian.
- Ukuran indeks lebih kecil dibandingkan keseluruhan data, sehingga pencarian lebih efisien.

3. Praktikum

3.1.Tujuan

Kegiatan praktikum akan menganalisis performa query dalam sistem basis data dengan menggunakan indexing dan tuning DBMS.

3.2.Tools

1. Java Programming
2. DBMS Seperti Mysql, Postgress, MariaDB dll.

3.3.Langkah

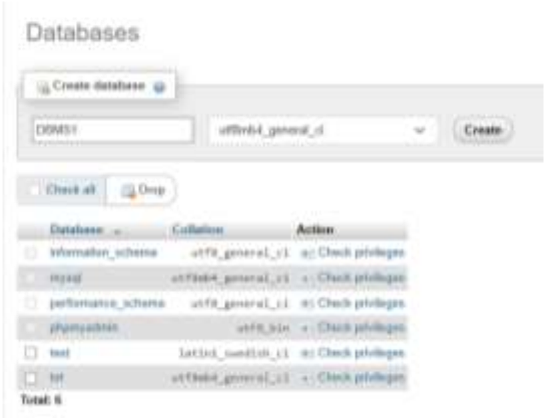
1. Pastikan DBMS sudah terinstall dan sedang dalam keadaan run
2. Tuliskan Query DDL berikut:

```
create table classroom
     (building varchar(15),
      room_number varchar(7),
      capacity numeric(4,0),
      primary key (building, room_number)
     );
create table department
     (dept_name varchar(20),
      Building varchar(15),
      Budget numeric(12,2) check (budget > 0),
      primary key (dept_name)
     );
create table course
     (course_id varchar(8),
      Title varchar(50),
      dept_name varchar(20),
      credits numeric(2,0) check (credits > 0),
      primary key (course_id), foreign key (dept_name)
references department(dept_name) on delete set null
     );
create table instructor
     (ID varchar(5),
      Name varchar(20) not null,
      dept_name varchar(20),
      salary numeric(8,2) check (salary > 29000),
      primary  key  (ID),  foreign  key  (dept_name)
references department(dept_name) on delete set null
     );
```

```
create table section
     (course_id varchar(8),
         sec_id varchar(8),
      semester varchar(6)
          check   (semester   in   ('Fall',   'Winter',
'Spring', 'Summer')), year              numeric(4,0)
check  (year  >  1701  and  year  <  2100),  building
varchar(15),  room_number  varchar(7),  time_slot_id
varchar(4), primary key (course_id, sec_id, semester,
year),foreign     key     (course_id)     references
course(course_id)  on  delete  cascade,  foreign  key
(building, room_number) references classroom(building,
room_number) on delete set null
     );
create table teaches
     (ID varchar(5),
      course_id varchar(8),
      sec_id varchar(8),
      semester varchar(6),
      year numeric(4,0),
      primary  key  (ID,  course_id,  sec_id,  semester,
year), foreign key (course_id,sec_id, semester, year)
references  section(course_id,sec_id,  semester,  year)
on   delete   cascade,   foreign   key   (ID)   references
instructor(ID) on delete cascade
     );
create table student
     (ID varchar(5),
      Name varchar(20) not null,
      dept_name varchar(20),
      tot_cred numeric(3,0) check (tot_cred >= 0),
      primary   key   (ID),   foreign   key   (dept_name)
references department(dept_name) on delete set null
     );
create table takes
     (ID varchar(5),
      course_id varchar(8),
      sec_id varchar(8),
      semester varchar(6),
      year numeric(4,0),
      grade varchar(2),
      primary  key  (ID,  course_id,  sec_id,  semester,
year), foreign key (course_id,sec_id, semester, year)
references  section(course_id,sec_id,  semester,  year)
on   delete   cascade,   foreign   key   (ID)   references
student(ID) on delete cascade
     );
```

```
create table advisor
     (s_ID varchar(5),
      i_ID varchar(5),
      primary   key   (s_ID),   foreign   key   (i_ID)
references instructor (ID) on delete set null, foreign
key (s_ID) references student (ID) on delete cascade
     );
create table prereq
     (course_id varchar(8),
      prereq_id varchar(8),
      primary key (course_id, prereq_id), foreign key
(course_id)  references  course(course_id)  on  delete
cascade,    foreign    key    (prereq_id)    references
course(course_id)
     )
```

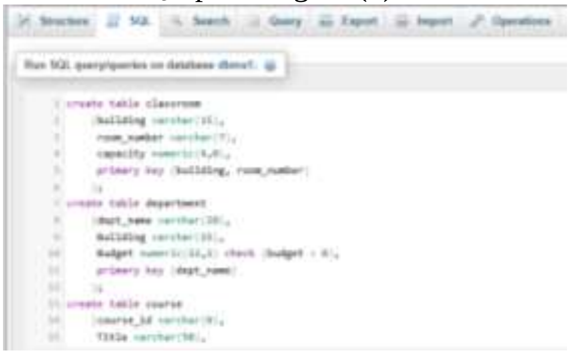
3. Buat Schema Database. Contoh DBMS1



	Database	Collation	Action
☐	information_schema	utf8_general_ci	Check privileges
☐	mysql	utf8mb4_general_ci	Check privileges
☐	performance_schema	utf8_general_ci	Check privileges
☐	phpmyadmin	utf8_bin	Check privileges
☐	test	latin1_swedish_ci	Check privileges
☐	[illegible]	utf8mb4_general_ci	Check privileges

Total: 6

4. Eksekusi SQL pada bagian (2)



```
create table classroom
    (building varchar(15),
     room_number varchar(7),
     capacity numeric(4,0),
     primary key (building, room_number)
    );
create table department
    (dept_name varchar(20),
     building varchar(15),
     budget numeric(12,2) check (budget > 0),
     primary key (dept_name)
    );
create table course
    (course_id varchar(8),
     title varchar(50),
```

5. Tuliskan code berikut dalam bahasa pemrograman Java.

```java
import java.util.Scanner;
import java.io.*;
import java.util.Random;
import java.text.DecimalFormat;

public class tableGen {
    private static int maxClassroom = 100;
    private static int maxStudent = 4000;
    private static int maxDepartment = 50;
    private static int maxRoom = 1000;
    private static int maxCourse = 1000;
    private static int maxSection = 2000;
    private static int maxAdvisor = 4000;
    private static int maxInstructor = 1000;
    private static int maxTeaches = 5000;
    private static int maxTakes= 40000;
    private static int maxPrereq =  1000;
    private static int maxBuilding = 100;
    private static int maxName = 4200;
    private static int maxDept = 100;
    private static int maxTitle = 1000;
    private static double maxSalary = 100000.0;
    private static double maxBudget = 999999.0;
    private static int maxID = 99999;
    private static Random rnd = new Random();
    private static String[] buildingArray = new
String[maxBuilding];
    private static String[] nameArray = new String[maxName];
    private static String[] deptNameArray = new String[maxDept];
    private static String[] titleArray = new String[maxTitle];
    private static Boolean[] teachesArray = new
Boolean[maxSection];
    private static int[] nextCourseSection = new int[1000]; //
starts at 0
    private static DecimalFormat moneyFormat = new
DecimalFormat("0.00");

    /**
     * primary keys
     */
    // classroom
    private static String[] classroomBuilding = new
String[maxClassroom];
    private static int[] classroomRoom = new int[maxRoom];
    // department
    private static String[] departmentDeptName = new
String[maxDepartment];
    // course
    private static int[] courseCourseID = new int[maxCourse];
    // store course dept for integrity check
    private static String[] courseDept = new String[maxCourse];
    // instructor
    private static int[] instructorID = new int[maxInstructor];
    // store instructor dept for integrity check
    private static String[] instructorDept = new
String[maxInstructor];
    // section
    private static int[] sectionCourseID = new int[maxSection];
    private static int[] sectionSectionID = new int[maxSection];
    private static String[] sectionSemester = new
String[maxSection];
    private static int[] sectionYear = new int[maxSection];
    // teaches
    private static int[] teachesID = new int[maxTeaches];
    private static int[] teachesCourseID = new int[maxTeaches];
    private static int[] teachesSectionID = new int[maxTeaches];
    private static String[] teachesSemester = new
String[maxSection];
    private static int[] teachesYear = new int[maxSection];
    // student
    private static int[] studentID = new int[maxStudent];
    // takes
    private static int[] takesID = new int[maxTakes];
    private static int[] takesCourseID = new int[maxTakes];
    private static int[] takesSectionID = new int[maxTakes];
    private static String[] takesSemester = new
String[maxTakes];
    private static int[] takesYear = new int[maxTakes];
    // advisor
    private static int[] advisorID = new int[maxAdvisor];

    private static String squote(int val) {
        return ("'"+val+"'");
    }

    private static String squote(String text) {
        return ("'"+text+"'");
    }

    private static Scanner openFile(String fileName) {
        Scanner in = null;
        try {
            in = new Scanner(new FileInputStream ("ref/" +
fileName));
        }
        catch(FileNotFoundException e) {
            System.out.println ("Could not open the file"+
e.getMessage());
            System.exit (0);
        }
        return in;
    }

    private static PrintWriter outputFile(String fileName) {
        PrintWriter out = null;
        try {
            out = new PrintWriter (new FileOutputStream ("sql/"
+ fileName));
        }
        catch (FileNotFoundException e) {
            System.out.println ("Could not open the file");
            System.exit (0);
        }
        return out;
    }
    private static void fillArrays() {
        int i;
        // fill buildingArray
        Scanner in = openFile("buildingNames");
        i = 0;
        while (in.hasNext() && i < maxBuilding) {
            buildingArray[i++] = in.nextLine();
        }
        if (i < maxBuilding) maxBuilding = i;
        in.close();
        // fill nameArray
        in = openFile("personNames");
        i = 0;
        while (in.hasNext() && i < maxName) {
            nameArray[i++] = in.nextLine();
        }
        if (i < maxName) maxName = i;
        in.close();
        // fill deptNameArray
        in = openFile("deptNames");
        i = 0;
        while (in.hasNext() && i < maxDept) {
            deptNameArray[i++] = in.nextLine();
        }
        if (i < maxDept) maxDept = i;
```

```
            in.close();
            // fill titleArray
            in = openFile("courseTitles");
            i = 0;
            while (in.hasNext() && i < maxTitle) {
                titleArray[i++] = in.nextLine();
            }
            if (i < maxTitle) maxTitle= i;
            in.close();
        }
        private static String getBuilding() {
            // building char(15)
            return buildingArray[rnd.nextInt(maxBuilding)];
        }
        private static int getRoom() {
            // room_no char(7)
            return rnd.nextInt(maxRoom); // stick with 3 digit rooms
        }
        private static int getCapacity() {
            // capacity numeric(4,0)
            // we'll use 226 as max and bias it towards small rooms
            int i = rnd.nextInt(16);
            return 10 + rnd.nextInt(1+i*i);
        }
        private static String getBudget() {
            // budget numeric(12,2)
            return moneyFormat.format(maxBudget*rnd.nextFloat());
        }
        private static String getTitle() {
            // title char(50)
            return titleArray[rnd.nextInt(maxTitle)];
        }
        private static int getCredits() {
            // credits numeric(2,0)
            return rnd.nextBoolean() ? 3 : 4;
        }
        private static int getID() {
            // ID numeric(9,0)
            return rnd.nextInt(maxID+1);
        }
        private static String getName() {
            // name char(20)
            return nameArray[rnd.nextInt(maxName)];
        }
        private static String getDeptName() {
```

```
        // dept_name char(20)
        return deptNameArray[rnd.nextInt(maxDept)];
    }
    private static String getSalary() {
        // salary numeric(8,2)
        return moneyFormat.format(29000 +
maxSalary*rnd.nextFloat());
    }
    private static int getCourseID() {
        // course_id char(8)
        // we'll have courses numbered 100 - 999
        return 101 + rnd.nextInt(899);
    }
    private static int getSecID(int courseID) {
        // sec_id char(8)
        // first section is section 1, then 2, by course number
independent of year
        return ++nextCourseSection[courseID];
    }
    private static String getSemester() {
        // semester char(6) check semester in ('Fall', 'Winter',
'Spring', 'Summer')
        return rnd.nextBoolean() ? "Fall" : "Spring";
    }
    private static int getYear() {
        // year numeric(4,0) check (year > 1759 and year < 2100)
        // stick to 2001 through 2010
        return 2001 + rnd.nextInt(10);
    }
    private static int getTotCred() {
        // tot_cred numeric(3,0)
        return rnd.nextInt(130);
    }
    private static String getGrade() {
        // grade char(2)
        int pM = rnd.nextInt(3);
        int lG = rnd.nextInt(3); // just A, B, C
        String x;
        if (pM == 0) x = "-";
        else if (pM == 1) x = " ";
        else x = "+";
        char y = (char) ((int)'A' + lG);
        return y + x;
    }
    private static String getTimeSlot() {
        // time_slot_id char(4)
        // this table is fixed with slots A through P
        char x = (char) ((int)'A' + rnd.nextInt(16));
        return "" + x;
    }
    public static void main(String[] args) {
        int classroom = 10;
        int department = 10;
        int course = 200;
        int instructor = 50;
        int teaches = 100;
        int advisor = 100;
        int student = 100;
        int section = 200;
        int takes = 200;
        int prereq = 100;
        int timeSlot = 10;
        int i = 0, j = 0, r = 0, c = 0, x = 0, y = 0;
        boolean tryValue = true;
        String b = "", d = "", s = "";
        fillArrays();

        // fill classroom
        PrintWriter out = outputFile("all.sql");
        for (i = 0; i < classroom; i++) {
            tryValue = true;
            while (tryValue) {
                b = getBuilding();
                r = getRoom();
                tryValue = false;
                for (j = 0; j < i; j++) {
                    if (b.equals(classroomBuilding[j]) && r ==
classroomRoom[j])
                        tryValue = true;
                }
            }
            classroomBuilding[i] = b;
            classroomRoom[i] = r;
            s = squote(b) + ", " + r + ", " + getCapacity();
            System.out.println(s);
            out.println ("insert into classroom values(" + s
+");");
        }

        // fill department
        for (i = 0; i < department; i++) {
            tryValue = true;
            while (tryValue) {
                d = getDeptName();
                tryValue = false;
                for (j = 0; j < i; j++) {
                    System.out.println(d +"/"+ 
departmentDeptName[j]);
                    if (d.equals(departmentDeptName[j]))
                        tryValue = true;
                }
            }
            departmentDeptName[i] =  d;
            s = squote(d) + ", " + squote(getBuilding()) + ", " 
+ getBudget();
            System.out.println(s);
            out.println ("insert into department values(" + s 
+");");
        }

        // fill course
        for (i = 0; i < course; i++) {
            tryValue = true;
            while (tryValue) {
                c = getCourseID();
                tryValue = false;
                for (j = 0; j < i; j++) {
                    if (c == courseCourseID[j]) tryValue = true;
                }
            }
            courseCourseID[i] = c;
            courseDept[c]=departmentDeptName[rnd.nextInt(departm
ent)];
            s = squote(c) + ", " + squote(getTitle()) + ", " + 
squote(courseDept[c]) + ", " + getCredits();
            System.out.println(s);
            out.println ("insert into course values(" + s 
+");");
        }

        // fill instructor
        for (i = 0; i < instructor; i++) {
            tryValue = true;
            while (tryValue) {
                c = getID();
```

```
                    tryValue = false;
                    for (j = 0; j < i; j++) {
                        if (c == instructorID[j]) tryValue = true;
                    }
                }
                instructorID[i] = c;
                instructorDept[i]=departmentDeptName[rnd.nextInt(dep
artment)];
                s = squote(c) + ", " + squote(getName()) + ", " +
squote(instructorDept[i] ) + ", " + getSalary();
                System.out.println(s);
                out.println ("insert into instructor values(" + s
+");");
            }

            // fill section
            for (i = 0; i < section; i++) {
                c = courseCourseID[rnd.nextInt(course)];
                r = getSecID(c);
                j = rnd.nextInt(classroom);
                sectionCourseID[i] = c;
                sectionSectionID[i] = r;
                b = getSemester();
                sectionSemester[i] = b;
                x = getYear();
                sectionYear[i] = x;
                s = squote(c)  + ", " + squote(r) + ", " + squote(b)
+ ", " + x +
                        ", " + squote(classroomBuilding[j]) + ", " +
squote(classroomRoom[j]) + ", " + squote(getTimeSlot());
                System.out.println(s);
                out.println ("insert into section values(" + s
+");");
            }

            // fill teaches
            for (i = 0; i < maxSection; i++) teachesArray[i] =
false;
            for (i = 0; i < teaches; i++) {
                tryValue = true;
                while (tryValue) {
                    j = rnd.nextInt(section);
                    tryValue = teachesArray[j];
                }
                teachesArray[j] = true;
                c = sectionCourseID[j];
                r = sectionSectionID[j];
                b = sectionSemester[j];
                x = sectionYear[j];
                //  insist that instructor be in same department as
    the course being taught
                int chosenInstructor = -1;
                for (y = 0; y < instructor; y++) {
                    if (instructorDept[y] == courseDept[c]) {
                        if (chosenInstructor == -1) chosenInstructor
    = y;
                        else if (rnd.nextInt(10) > 1)
    chosenInstructor = y;
                    }
                }
                y = chosenInstructor == -1 ? rnd.nextInt(instructor)
    : chosenInstructor;
                if (instructorDept[y] == courseDept[c])
    System.out.println("teaching outside dept: " +
                        y + " " + c);
                s = squote(instructorID[y]) + ", " + squote(c) + ",
    " + squote(r) + ", " + squote(b) + ", " + x;
                System.out.println(s);
                out.println ("insert into teaches values(" + s
    +");");
            }

            // fill student
            for (i = 0; i < student; i++) {
                tryValue = true;
                while (tryValue) {
                    c = getID();
                    tryValue = false;
                    for (j = 0; j < i; j++) {
                        if (c == studentID[j]) tryValue = true;
                    }
                }
                studentID[i] = c;
                s = squote(c) + ", " + squote(getName()) + ", " +
    squote(departmentDeptName[rnd.nextInt(department)])
                        + ", " + getTotCred();
                System.out.println(s);
                out.println ("insert into student values(" + s
    +");");
```

```
        }

        // fill takes
        for (i = 0; i < takes; i++) {
            tryValue = true;
            while (tryValue) {
                j = rnd.nextInt(student);
                x = studentID[j];
                j = rnd.nextInt(section);
                c = sectionCourseID[j];
                r = sectionSectionID[j];
                b = sectionSemester[j];
                y = sectionYear[j];
                tryValue = false;
                for (j = 0; j < i; j++) {
                    if (x == takesID[j] && c == takesCourseID[j]
&& r == takesSectionID[j])
                        tryValue = true;
                }
            }
            takesID[i] = x;
            takesCourseID[i] = c;
            takesSectionID[i] = r;
            takesSemester[i] = b;
            takesYear[i] = y;
            s = squote(x) + ", " + squote(c) + ", " + squote(r)
+ ", " + squote(b) + ", " + y + ", " + squote(getGrade());
            System.out.println(s);
            out.println ("insert into takes values(" + s +");");
        }

        // fill advisor
        for (i = 0; i < student; i++) {
            s = squote(studentID[i]) + ", " +
squote(instructorID[rnd.nextInt(instructor)]) ;
            System.out.println(s);
            out.println ("insert into advisor values(" + s
+");");
        }

        // fill prereq
        for (i = 0; i < prereq; i++) {
            s = squote(courseCourseID[rnd.nextInt(course)]) + ",
" + squote(courseCourseID[rnd.nextInt(course)]);
            System.out.println(s);
            out.println ("insert into prereq values(" + s
+");");
        }
        out.close();
    }
}
```

6. Compile dan jalan code tersebut pada komputer anda.

3.4.Tahapan percobaan

1. Generate data dengan menjalankan code
2. Lakukan perubahan pada beberapa nilai
3. Hitung response time dari query yang diberikan
4. Lakukan Tunning. Tunning yang dilakukan yaitu tunnig dengan indexing dan setting DBMS.