

Opdracht 3 - Performance

Verslag

Analyse 8 - Advanced Databases (2014-2015)

Namen:	HoYe Lam, Rinesh Ramadhin
Studentnummer:	0876814, 0882447
Klas:	INF2D
Vak:	INFANL01-8
Opdracht:	Performance
Datum:	27 – 05 - 2015

Inhoud

Simulatie: Gebruikers die data invoeren.....	2
Simulatie: gebruikers die data opvragen.....	8
Simulatie: Indexen.....	11

Simulatie: Gebruikers die data invoeren

Hieronder vindt U de JAVA code die we hebben geschreven om gelijktijdig meerdere data in te voeren in de POSTGRES database. Er wordt per iteratie geprint hoelang elke thread erover doet.

We hebben in elke thread een aantal INSERTS uitgevoerd. De gemeten tijd per iteratie ligt tussen de 0 en 4 milliseconden.

De voornamen van de studenten zijn random gegenereerd, zodat we unieke waarden hebben voor gebruik bij de index simulatie.

Uiteindelijk wordt de tijd per iteratie uitgeprint. Aan het eind van elke thread wordt de gemiddelde tijd van de iteraties uitgeprint.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class DataInvoegen {
    // decladerdeer variabele
    //Connectie Variables
    public static String url = "jdbc:postgresql://127.0.0.1:5432/postgres";
    public static String username = "postgres";
    public static String password = "hoye";

    public static Random rand = new Random();

    public static void main(String[] args) {
        // Maak en start thread 1
        new Thread(new Runnable() {
            @Override
            public void run() {
                Connection connection = connect();
                // Decladeer Thread 1 variables
                List<String> groep = new ArrayList<String>();
                List<String> module = new ArrayList<String>();
                List<Long> totale_duur_thread1 = new ArrayList<Long>();

                long begintijd;
                long eindtijd;
                long duurInms;

                // Onze eerste klas en module
                Statement st1;
                try {

                    st1 = (Statement) connection.createStatement();
                    st1.executeUpdate("INSERT INTO Groep "
                        + "VALUES ('INF1D', '2015-05-05', '2016-01-01')");
                    groep.add("INF1D");

                    st1.executeUpdate("INSERT INTO Cursussen "
                        + "VALUES ('INFANL-01', 'TJANG', 'analyse', '2015-05-05', '2016-01-01')");
                    module.add("INFANL-01");

                } catch (SQLException e2) {
                    // TODO Auto-generated catch block
                    e2.printStackTrace();
                }

                // Begin 600 iteraties van student en klas toevoegen
                for (int i = 0; i < 600; i++) {

                    // sla begin tijd op
                    begintijd = System.currentTimeMillis();

                    // random string
                    char[] chars = "abcdefghijklmnopqrstuvwxyz".toCharArray();
                    StringBuilder sb = new StringBuilder();
                    for (int nu = 0; nu < 6; nu++) {
```

```

        char c = chars[rand.nextInt(chars.length)];
        sb.append(c);
    }

    // transactie
    try {
        Statement st = (Statement) connection.createStatement();

        int studentnummer = 1876814 + i;
        int modulenummer = i + 10;
        String klas = "INF" + i + "D";
        String naam = sb.toString();
        String modulenaam = "INFANL-" + modulenummer;

        st.executeUpdate("INSERT INTO Student "
            + "VALUES ("
            + studentnummer
            + ","
            + naam
            + "," + 'Lam' + "," + '1996-02-28' + ',' + 'man' + ',' + 'wijnhaven',106
            + ',' + 'Rotterdam', '2222AA', '0639387945')");

        int n = rand.nextInt(30) + 1;
        if (n == 20) {
            groep.add(klas);
            st.executeUpdate("INSERT INTO Groep " + "VALUES ("
                + klas + "," + '2015-05-05' + ',' + '2016-01-01')");
        }

        // student aan willekeurige klas
        st.executeUpdate("INSERT INTO Groep_has_Student "
            + "VALUES ("
            + groep.get(rand.nextInt(groep.size()))
            + "," + studentnummer + ")");

        int z = rand.nextInt(30) + 1;
        if (z == 20) {
            module.add(modulenaam);
            st.executeUpdate("INSERT INTO Cursussen "
                + "VALUES ("
                + modulenaam
                + "," + 'TJANG' + ',' + 'analyse' + ',' + '2015-05-05' + ',' + '2016-01-01')");

            int a = rand.nextInt(100) + 1;
            if (a <= 15) {
                for (int b = 0; b < groep.size(); b++) {
                    st.executeUpdate("INSERT INTO Cursussen_has_Groep "
                        + "VALUES ("
                        + groep.get(b)
                        + "," + modulenaam + ")");
                }
            }
        }
    }

    } catch (SQLException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }

    // sla eind tijd op
    eindtijd = System.currentTimeMillis();

    // Bereken duur
    duurInms = eindtijd - begintijd;
    System.out.println("Gebruiker 1 : " + duurInms);

    totale_duur_thread1.add(duurInms);
}

int i;
float totale_duur = 0;
for (i = 1; i < totale_duur_thread1.size(); i++) {
    totale_duur += totale_duur_thread1.get(i);
}
System.out.println("thread 1 totale duur : " + totale_duur);
System.out.println("thread 1 gemiddeld: " + (totale_duur / totale_duur_thread1.size()));

try {
    connection.commit();
    connection.close();
} catch (SQLException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

```

```

    }

    try {
        Thread.sleep((long) (Math.random() * 1000));
    } catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

}, "Thread 1").start();

// Maak en start thread 2
new Thread(new Runnable() {
    @Override
    public void run() {
        Connection connection = connect();
        // Decladeer Thread 2 variables
        List<String> groep = new ArrayList<String>();
        List<String> module = new ArrayList<String>();
        List<Long> totale_duur_thread2 = new ArrayList<Long>();

        // Decladeer variables
        long begintijd;
        long eindtijd;
        long duurInms;

        // Onze eerste klas en module
        Statement st1;
        try {
            st1 = (Statement) connection.createStatement();
            st1.executeUpdate("INSERT INTO Groep "
                + "VALUES ('INF1B', '2015-05-05', '2016-01-01')");
            groep.add("INF1B");

            st1.executeUpdate("INSERT INTO Cursussen "
                + "VALUES ('INFDEV-01', 'TJANG', 'analyse', '2015-05-05', '2016-01-01')");
            module.add("INFDEV-01");
        } catch (SQLException e2) {
            // TODO Auto-generated catch block
            e2.printStackTrace();
        }

        // Begin 600 iteraties van student en klas toevoegen
        for (int i = 0; i < 600; i++) {

            // sla begin tijd op
            begintijd = System.currentTimeMillis();

            // random string
            char[] chars = "abcdefghijklmnopqrstuvwxyz".toCharArray();
            StringBuilder sb = new StringBuilder();
            for (int nu = 0; nu < 6; nu++) {
                char c = chars[rand.nextInt(chars.length)];
                sb.append(c);
            }

            // transactie
            try {
                Statement st = (Statement) connection.createStatement();

                int studentnummer = 2876814 + i;
                int modulenummer = i + 10;
                String klas = "INF" + i + "B";
                String naam = sb.toString();
                String modulenaam = "INFDEV-" + modulenummer;

                st.executeUpdate("INSERT INTO Student "
                    + "VALUES ("
                    + studentnummer
                    + ", "
                    + naam
                    + ", 'Lam', '1996-02-28', 'man', 'wijnhaven', 106
                    + ", 'Rotterdam', '2222AA', '0639387945')");

                int n = rand.nextInt(30) + 1;
                if (n == 20) {
                    groep.add(klas);
                    st.executeUpdate("INSERT INTO Groep " + "VALUES ("
                        + klas + ", '2015-05-05', '2016-01-01')");
                }

                // student aan willekeurige klas
                st.executeUpdate("INSERT INTO Groep_has_Student "
                    + "VALUES ("

```

```

        + groep.get(rand.nextInt(groep.size()))
        + "'','" + studentnummer + "'");

int z = rand.nextInt(30) + 1;
if (z == 20) {
    module.add(modulenaam);
    st.executeUpdate("INSERT INTO Cursussen "
        + "VALUES ('"
        + modulenaam
        + "', 'TJANG','analyse','2015-05-05', '2016-01-01')");
    int a = rand.nextInt(100) + 1;

    if (a <= 15) {
        for (int b = 0; b < groep.size(); b++) {

            st.executeUpdate("INSERT INTO Cursussen_has_Groep "
                + "VALUES ('"
                + groep.get(b)
                + "',''" + modulenaam + "'");

        }
    }

}

} catch (SQLException e1) {
    // TODO Auto-generated catch block
    e1.printStackTrace();
}

// sla eind tijd op
eindtijd = System.currentTimeMillis();

// Bereken duur
duurInms = eindtijd - begintijd;
System.out.println("Gebruiker 2 : " + duurInms);

totale_duur_thread2.add(duurInms);
}

int i;
float totale_duur = 0;
for (i = 1; i < totale_duur_thread2.size(); i++) {
    totale_duur += totale_duur_thread2.get(i);
}
System.out.println("thread 2 totale duur : " + totale_duur);
System.out.println("thread 2 gemiddeld: " + (totale_duur / totale_duur_thread2.size()));
try {
    connection.commit();
    connection.close();
} catch (SQLException e1) {
    // TODO Auto-generated catch block
    e1.printStackTrace();
}

try {
    Thread.sleep((long) (Math.random() * 1000));
} catch (InterruptedException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

}
}, "Thread 2").start();

// Maak en start thread 3
new Thread(new Runnable() {
    @Override
    public void run() {
        Connection connection = connect();
        List<String> groep = new ArrayList<String>();
        List<String> module = new ArrayList<String>();

        // Decladeer variables
        long begintijd;
        long eindtijd;
        long duurInms;

        List<Long> totale_duur_thread3 = new ArrayList<Long>();

        // Onze eerste klas en module
        Statement st1;
        try {
            st1 = (Statement) connection.createStatement();
            st1.executeUpdate("INSERT INTO Groep "
                + "VALUES ('INF1C', '2015-05-05', '2016-01-01')");
            groep.add("INF1C");

```

```

        st1.executeUpdate("INSERT INTO Cursussen "
            + "VALUES ('INFONZ-01','TJANG','analyse','2015-05-05', '2016-01-01')");
        module.add("INFONZ-01");

    } catch (SQLException e2) {
        // TODO Auto-generated catch block
        e2.printStackTrace();
    }

    // Begin 600 iteraties van student en klas toevoegen
    for (int i = 0; i < 600; i++) {

        // sla begin tijd op
        begintijd = System.currentTimeMillis();

        // random string
        char[] chars = "abcdefghijklmnopqrstuvwxyz".toCharArray();
        StringBuilder sb = new StringBuilder();
        for (int nu = 0; nu < 6; nu++) {
            char c = chars[rand.nextInt(chars.length)];
            sb.append(c);
        }

        // transactie
        try {
            Statement st = (Statement) connection.createStatement();

            int studentnummer = 3876814 + i;
            int modulenummer = i + 10;
            String klas = "INF" + i + "C";
            String naam = sb.toString();
            String modulenaam = "INFONZ-" + modulenummer;

            st.executeUpdate("INSERT INTO Student "
                + "VALUES ("
                + studentnummer
                + ","
                + naam
                + "','Lam','','1996-02-28','man','wijnhaven',106
                + "','Rotterdam','2222AA','0639387945')");

            int n = rand.nextInt(30) + 1;
            if (n == 20) {
                groep.add(klas);
                st.executeUpdate("INSERT INTO Groep " + "VALUES ("
                    + klas + "','2015-05-05', '2016-01-01')");
            }

            // student aan willekeurige klas
            st.executeUpdate("INSERT INTO Groep_has_Student "
                + "VALUES ("
                + groep.get(rand.nextInt(groep.size()))
                + "','" + studentnummer + "')");

            int z = rand.nextInt(30) + 1;
            if (z == 20) {
                module.add(modulenaam);
                st.executeUpdate("INSERT INTO Cursussen "
                    + "VALUES ("
                    + modulenaam
                    + "','TJANG','analyse','2015-05-05', '2016-01-01')");
                int a = rand.nextInt(100) + 1;

                if (a <= 15) {
                    for (int b = 0; b < groep.size(); b++) {

                        st.executeUpdate("INSERT INTO Cursussen_has_Groep "
                            + "VALUES ("
                            + groep.get(b)
                            + "','" + modulenaam + "')");
                    }
                }
            }
        } catch (SQLException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }

        // sla eind tijd op
        eindtijd = System.currentTimeMillis();

        // Bereken duur
    }

```

```

        duurInms = eindtijd - begintijd;
        System.out.println("Gebruiker 3 : " + duurInms);

        totale_duur_thread3.add(duurInms);
    }

    int i;
    float totale_duur = 0;
    for (i = 1; i < totale_duur_thread3.size(); i++) {
        totale_duur += totale_duur_thread3.get(i);
    }

    System.out.println("thread 3 totale duur : " + totale_duur);
    System.out.println("thread 3 gemiddeld: " + (totale_duur / totale_duur_thread3.size()));

    try {
        connection.commit();
        connection.close();
    } catch (SQLException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }

    try {
        Thread.sleep((long) (Math.random() * 1000));
    } catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
}, "Thread 3").start();
}

//Connect functie
public static Connection connect() {
    // maak verbinding met postgres

    // maak verbinding met de driver
    try {
        Class.forName("org.postgresql.Driver");
    } catch (ClassNotFoundException e) {
        throw new RuntimeException(
            "Cannot find the driver in the classpath!", e);
    }

    // maak verbinding met de database
    Connection connection = null;
    try {
        System.out.println("Connecting database...");
        connection = DriverManager.getConnection(url, username, password);
        System.out.println("Database connected!");

        connection.setAutoCommit(false);

    } catch (Exception e) {
    }
    return connection;
}
}

```


Simulatie: gebruikers die data opvragen

Hieronder vindt U de JAVA code die we hebben geschreven om gelijktijdig meerdere data op te vragen uit de POSTGRES database. Er wordt per iteratie geprint hoelang elke thread erover doet.

We hebben in elke thread een aantal SELECTS uitgevoerd. De gemeten tijd per iteratie ligt tussen de 0 en 4 milliseconden.

Uiteindelijk wordt de tijd per iteratie uitgeprint samen met de vakken die een random gekozen student volgt / heeft gevolgt. Aan het eind van elke thread wordt de gemiddelde tijd van de iteraties uitgeprint.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class DataLezen {
    public static void main(String[] args) {
        // Maak en start thread 1
        new Thread(new Runnable() {
            @Override
            public void run() {
                Connection connection = DataInvoegeen.connect();
                // Decladeer variables
                long begintijd;
                long eindtijd;
                long duurInms;
                List<Long> totale_duur_thread = new ArrayList<Long>();
                Statement st1;

                // Begin 600 iteraties van student en klas toevoegen
                for (int i = 0; i < 600; i++) {

                    // SELECT ALLE STUDENTEN

                    try {
                        st1 = (Statement) connection.createStatement();
                        ResultSet z;
                        z = st1.executeQuery("SELECT voornaam,achternaam FROM Student ORDER BY RANDOM() LIMIT 1;");

                        String voornaam = null;
                        String achternaam = null;
                        String groepnaam = null;
                        String cursusnaam = null;

                        if (z.next()) {
                            voornaam = z.getString("voornaam");
                            achternaam = z.getString("achternaam");
                        }

                        // sla begin tijd op
                        begintijd = System.currentTimeMillis();

                        // transactie
                        ResultSet groep_ophalen;
                        groep_ophalen = st1
                            .executeQuery("SELECT groep_groepnaam FROM Student INNER JOIN Groep_has_Student ON
Student.studentnummer = Groep_has_Student.student_studentnummer WHERE voornaam = '"
                                + voornaam
                                + "' AND achternaam = '"
                                + achternaam + "';");

                        if (groep_ophalen.next()) {
                            groepnaam = groep_ophalen
                                .getString("groep_groepnaam");
                        }

                        ResultSet cursus_ophalen;
                        cursus_ophalen = st1
                            .executeQuery("SELECT * FROM Cursussen_has_Groep WHERE groep_groepnaam = '"
                                + groepnaam + "';");
```

```

        while (cursus_ophalen.next()) {
            cursusnaam = cursus_ophalen
                .getString("cursussen_cursuscode");
            System.out.println("A: " + voornaam + " "
                + achternaam + " in " + groepnaam
                + " volgt " + cursusnaam);
        }

        // sla eind tijd op
        eindtijd = System.currentTimeMillis();

        // Bereken duur
        duurInms = eindtijd - begintijd;
        System.out.println("Gebruiker 1 : " + duurInms);
        totale_duur_thread.add(duurInms);

    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

}
int i;
float totale_duur = 0;
for (i = 1; i < totale_duur_thread.size(); i++) {
    totale_duur += totale_duur_thread.get(i);
}
System.out.println("thread 1 deed : " + totale_duur);
System.out.println("thread 1 gemiddeld: " + (totale_duur / totale_duur_thread.size()));

try {
    connection.commit();
    connection.close();
} catch (SQLException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

}
}, "Thread 1").start();

// Maak en start thread 2
new Thread(new Runnable() {
    @Override
    public void run() {
        Connection connection = DataInvoeegen.connect();

        // Decladeer variables
        long begintijd;
        long eindtijd;
        long duurInms;
        List<Long> totale_duur_thread = new ArrayList<Long>();
        Statement st1;

        // Begin 600 iteraties van student en klas toevoegen
        for (int i = 0; i < 600; i++) {

            // SELECT ALLE STUDENTEN

            try {
                st1 = (Statement) connection.createStatement();
                ResultSet z;
                z = st1.executeQuery("SELECT voornaam,achternaam FROM Student ORDER BY RANDOM() LIMIT 1;");

                String voornaam = null;
                String achternaam = null;
                String groepnaam = null;
                String cursusnaam = null;

                if (z.next()) {
                    voornaam = z.getString("voornaam");
                    achternaam = z.getString("achternaam");
                }

                // sla begin tijd op
                begintijd = System.currentTimeMillis();

                // transactie
                ResultSet groep_ophalen;
                groep_ophalen = st1
                    .executeQuery("SELECT groep_groepnaam FROM Student INNER JOIN Groep_has_Student ON
Student.studentnummer = Groep_has_Student.student_studentnummer WHERE voornaam = '"
                        + voornaam
                        + "' AND achternaam = '"

```

```

        + achternaam + "';");

        if (groep_ophalen.next()) {
            groepnaam = groep_ophalen
                .getString("groep_groepnaam");
        }

        ResultSet cursus_ophalen;
        cursus_ophalen = st1
            .executeQuery("SELECT * FROM Cursussen_has_Groep WHERE groep_groepnaam = '"
                + groepnaam + "';");

        while (cursus_ophalen.next()) {
            cursusnaam = cursus_ophalen
                .getString("cursussen_cursuscode");
            System.out.println("B: " + voornaam + " "
                + achternaam + " in " + groepnaam
                + " volgt " + cursusnaam);
        }

        // sla eind tijd op
        eindtijd = System.currentTimeMillis();

        // Bereken duur
        duurInms = eindtijd - begintijd;
        System.out.println("Gebruiker 2 : " + duurInms);
        totale_duur_thread.add(duurInms);

    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

}

int i;
float totale_duur = 0;
for (i = 1; i < totale_duur_thread.size(); i++) {
    totale_duur += totale_duur_thread.get(i);
}
System.out.println("thread 2 totale duur : " + totale_duur);
System.out.println("thread 2 gemiddeld: " + (totale_duur / totale_duur_thread.size()));

try {
    connection.commit();
    connection.close();
} catch (SQLException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

}

}, "Thread 2").start();
}
}

```

Simulatie: Indexen

Hieronder vindt U de JAVA code die we hebben geschreven om aan te tonen dat het gebruiken van een index een positieve performance impact heeft. We hebben allereerst de INDEX verwijderd. Hierna hebben we alle studenten opgevraagd van wie de voornaam begint met een "a". de tijd voor het ophalen is gemeten. Hierna hebben we een INDEX gemaakt voor de "voornaam" kolom in de "student" tabel. Hierna hebben we nogmaals alle studenten opgevraagd van wie de voornaam begint met een "a". Ook hier werd de tijd gemeten. In de resultaten onderaan de code ziet U dat de INDEX wordt gebruikt en dat het een positieve impact heeft op de performance.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class DataLezenIndex {
    public static void main(String[] args) {
        // Maak en start thread 1
        new Thread(new Runnable() {
            @Override
            public void run() {
                Connection connection = DataInvoegen.connect();
                // Decladeer variables
                long begintijd_zonder_idx;
                long eindtijd_zonder_idx;
                long begintijd_met_idx;
                long eindtijd_met_idx;
                long duurInms_zonder_idx;
                long duurInms_met_idx;

                Statement st1;
                String voornaam = null;

                try {
                    st1 = (Statement) connection.createStatement();

                    //Zonder IDX duur
                    st1.executeUpdate("DROP INDEX Student_idx;");

                    // sla begin tijd op
                    begintijd_zonder_idx = System.currentTimeMillis();
                    ResultSet z;
                    z = st1.executeQuery("SELECT voornaam FROM Student WHERE voornaam LIKE 'a%';");

                    if (z.next()) {
                        voornaam = z.getString("voornaam");
                    }

                    eindtijd_zonder_idx = System.currentTimeMillis();

                    duurInms_zonder_idx = eindtijd_zonder_idx - begintijd_zonder_idx;

                    System.out.println("Zonder Index Duurt : " + duurInms_zonder_idx);

                    //Met IDX duur
                    st1.executeUpdate("CREATE UNIQUE INDEX Student_idx ON Student (voornaam);");

                    // sla begin tijd op
                    begintijd_met_idx = System.currentTimeMillis();
                    ResultSet b;
                    b = st1.executeQuery("SELECT voornaam FROM Student WHERE voornaam LIKE 'a%';");

                    if (b.next()) {
                        voornaam = b.getString("voornaam");
                    }

                    eindtijd_met_idx = System.currentTimeMillis();

                    duurInms_met_idx = eindtijd_met_idx - begintijd_met_idx;
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }).start();
    }
}
```

```

        System.out.println("Met Index Duurt : " + duurInms_met_idx);

        connection.close();
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
}, "Thread 1").start();
}
}

```

Hier ziet U dat de cost veel lager is wanneer we een INDEX gebruiken.

Output pane	
Data Output	Explain Messages History
QUERY PLAN	
text	
1	Index Only Scan using student idx on student (cost=0.28..8.29 rows=1 width=7)
2	Index Cond: (voornaam = ' ndchap'::text)

```

Problems @ Javadoc Declaration Console
<terminated> DataLezenIndex [Java Application] C:\Program Files\Java\jre1.8.0_25\bin\javaw.exe (May 27, 2015, 1:02:53 PM)
A: Loading driver...
A: Driver loaded!
A: Connecting database...
A: Database connected!
Zonder Index Duurt : 13
Met Index Duurt : 2

```