**PENGEMBANGAN SQL SIMILARITY DENGAN MULTI KUNCI JAWABAN UNTUK PROSES PENILAIAN OTOMATIS PADA SISTEM AUTOMATED ASSESSMENT MYSQL**

**PROPOSAL SKRIPSI**

**Oleh:**

**SULTAN ACHMAD QUM MASYKURO NS NIM. 1841720019**

****

**PROGRAM STUDI TEKNIK INFORMATIKA**

**JURUSAN TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI MALANG**

**2021**

**PENGEMBANGAN SQL SIMILARITY DENGAN MULTI KUNCI JAWABAN UNTUK PROSES PENILAIAN OTOMATIS PADA SISTEM AUTOMATED ASSESSMENT MYSQL**

**PROPOSAL SKRIPSI**

**Oleh:**

**SULTAN ACHMAD QUM MASYKURO NS NIM. 1841720019**

****

**PROGRAM STUDI TEKNIK INFORMATIKA**

**JURUSAN TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI MALANG**

**2021**

**HALAMAN PENGESAHAN**

**PENGEMBANGAN SQL SIMILARITY DENGAN MULTI KUNCI JAWABAN UNTUK PROSES PENILAIAN OTOMATIS PADA SISTEM AUTOMATED ASSESSMENT MYSQL**

**Disusun oleh:**

**SULTAN ACHMAD QUM MASYKURO NS NIM. 1841720019**

**Proposal Skripsi ini telah diuji pada tanggal 04 Februari 2022**

**Disetujui oleh:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1. | Pembimbing Utama | : | Putra Prima Arhandi, ST., M.Kom. NIP. 19861103 201404 1 001 | ........................... |
|  |  |  |  |  |
| 2. | Penguji I | : | Pramana Yoga Saputra, S.Kom., MMT.  NIP. 19880504 201504 1 001 | ........................... |
|  |  |  |  |  |
| 3. | Penguji II | : | Elok Nur Hamdana, S.T., M.T.  NIP. 19861002 201903 2 011 | ........................... |

Mengetahui,

|  |  |
| --- | --- |
| Ketua Jurusan  Teknologi Informasi | Ketua Program Studi  Teknik Informatika |
| Rudy Ariyanto, S.T., M.Cs. | Imam Fahrur Rozi, S.T., M.T. |
| NIP. 19711110 199903 1 002 | |  | | --- | | NIP. 19840610 200812 1 004 | |

# DAFTAR ISI

[DAFTAR ISI iv](#_Toc91094199)

[DAFTAR GAMBAR vi](#_Toc91094200)

[DAFTAR TABEL vii](#_Toc91094201)

[BAB I. PENDAHULUAN 1](#_Toc91094202)

[1.1. Latar Belakang 1](#_Toc91094203)

[1.2. Rumusan Masalah 3](#_Toc91094204)

[1.3. Batasan Masalah 4](#_Toc91094205)

[1.4. Tujuan 4](#_Toc91094206)

[1.5. Manfaat 4](#_Toc91094207)

[BAB II. LANDASAN TEORI 5](#_Toc91094208)

[2.1. Studi Literatur 5](#_Toc91094209)

[2.2. Dasar Teori 9](#_Toc91094210)

[2.2.1. MySQL 9](#_Toc91094211)

[2.2.2. SQL 9](#_Toc91094212)

[2.2.3. Automated Assessment 9](#_Toc91094213)

[2.2.4. TF-IDF 9](#_Toc91094214)

[2.2.5. Cosine Similarity 10](#_Toc91094215)

[2.2.6. SQLearn 11](#_Toc91094216)

[BAB III. METODOLOGI PENELITIAN 13](#_Toc91094217)

[3.1. Waktu dan Tempat Penelitian 13](#_Toc91094218)

[3.2. Teknik Pengumpulan Data 13](#_Toc91094219)

[3.3. Teknik Pengolahan Data 13](#_Toc91094220)

[3.3.1. Menentukan *Query* yang akan dilakukan pengecekan similaritas 14](#_Toc91094221)

[3.3.2. Mencari set of features dari query 14](#_Toc91094222)

[3.3.3. Menghitung TF-IDF 15](#_Toc91094223)

[3.3.4. Meghitung Cosine Similarity 16](#_Toc91094224)

[3.3.5. Ulangi untuk similaritas dengan *query* kunci yang lain 17](#_Toc91094225)

[3.3.6. Mencari nilai similaritas tertinggi 18](#_Toc91094226)

[3.4. Desain Sistem 18](#_Toc91094227)

[3.4.1. Use Case 18](#_Toc91094228)

[3.4.2. Activity Diagram 20](#_Toc91094229)

[3.4.3. Flowchart 23](#_Toc91094230)

[3.4.4. Arsitektur Sistem 24](#_Toc91094231)

[3.5. Uji Coba Sistem 25](#_Toc91094232)

[3.5.1. Multi Kunci Jawaban 25](#_Toc91094233)

[3.5.2. Insert Similarity 25](#_Toc91094234)

[BAB IV. JADWAL PENELITIAN 26](#_Toc91094235)

# DAFTAR GAMBAR

[Gambar 2.1 Konsep Multi Key Similarity 12](#_Toc91512455)

[Gambar 2.2 Konsep Tampilan Pengerjaan Soal Mahasiswa 12](#_Toc91512456)

[Gambar 2.3 Konsep Tampilan Tambah Soal Dosen 12](#_Toc91512457)

[Gambar 3.1 Use Case Mahasiswa 18](#_Toc94345694)

[Gambar 3.2 Use Case Dosen 19](#_Toc94345695)

[Gambar 3.3 Use Case Admin 20](#_Toc94345696)

[Gambar 3.4 Activity Diagram Mahasiswa 21](#_Toc94345697)

[Gambar 3.5 Activity Diagram Dosen 22](#_Toc94345698)

[Gambar 3.6 Flowchart Similarity 23](#_Toc94345699)

[Gambar 3.7 Arsitektur Sistem 24](#_Toc94345700)

[Gambar 3.8 Uji Coba Multi Kunci Jawaban 25](#_Toc94345701)

[Gambar 3.9 Uji Coba Insert Similarity 25](#_Toc94345702)

# DAFTAR TABEL

[Tabel 3.1 Set of Features A 14](#_Toc90588015)

[Tabel 3.2 Set of Features B 14](#_Toc90588016)

[Tabel 3.3 Hasil TF 15](#_Toc90588017)

[Tabel 3.4 Hasil DF 15](#_Toc90588018)

[Tabel 3.5 Hasil IDF 15](#_Toc90588019)

[Tabel 3.6 Hasil TF-IDF 16](#_Toc90588020)

[Tabel 3.7 Perkalian Skalar AxB 16](#_Toc90588021)

[Tabel 3.8 Hasil Total Kuadrat dan Akar Kuadrat 17](#_Toc90588022)

[Tabel 4.1 Tabel Jadwal Kegiatan 26](#_Toc90588006)

# PENDAHULUAN

## Latar Belakang

Basis Data telah menjadi dasar sumber daya informasi pengembangan dan pemanfaatan dalam masyarakat informasi, dan hal ini adalah bagian penting dari sistem informasi. Teknologi basis data digunakan di lebih dari 70% aplikasi komputer, sehingga menguasai teori basis data, dan menggunakan teknologi basis data dengan mahir telah menjadi kemampuan wajib yang harus dimiliki oleh mahasiswa yang ada di jurusan *computer science* (Jiandong et al., 2018). Basis Data merupakan salah satu mata kuliah yang terdapat di Jurusan Teknologi Informasi. Memberi pengetahuan kepada mahasiswa untuk membuat dan memanajemen data pada sistem manajemen *database* atau DBMS yang merupakan tujuan dari kurikulum Basis Data (Puspitasari et al., 2019). Berdasarkan penjelasan di atas dapat disimpulkan bahwa penting bagi mahasiswa untuk mempelajari dan memahami Basis Data.

Dalam pembelajaran Basis Data terdapat topik yang penting untuk dipelajari bagi pemula. Namun, saat proses pembelajaran banyak yang mengalami kesulitan, salah satunya adalah SQL *Command Query* (Nalintippayawong et al., 2017). Padahal, *Structure Query Language (SQL)* sering digunakan, karena SQL secara *de facto* merupakan bahasa standar untuk mengakses dan memanipulasi data, sehingga mempunyai kemampuan SQL *query* sangat dibutuhkan di industri (Obionwu et al., 2021). Kesulitan dalam belajar SQL dapat disebabkan oleh beberapa hal seperti kesalahan pada logika, ekspresi, projeksi, fungsi maupun kompleksitas query. (Taipalus & Perälä, 2019).

Proses penilaian secara manual membuat dosen dihadapkan dengan situasi sulit, antara fokus mengoreksi jawaban atau aktivitas mengajar (Luchoomun et al., 2019). Oleh karena itu dibuatlah Sistem *Automated Assessment*. Sistem *Automated Assessment* dapat bermanfaat untuk dosen maupun mahasiswa, manfaat untuk mahasiswa adalah jawaban yang diberikan oleh mahasiswa dinilai secara konsisten dan dapat mendapatkan *feedback* secara langsung sehingga dapat mengetahui apakah jawaban yang terpikirkan benar atau tidak, mahasiswa berusaha untuk mencari solusi terbaik dengan cara mengirim jawaban secara berulang dan meningkatkan jawabannya berdasarkan *feedback* yang didapatkan (Daradoumis et al., 2019).

Sistem *automated assessment* telah dibuat oleh (Puspitasari et al., 2019) dengan tujuan melakukan penilaian pada kurikulum basis data berdasarkan hasil SQL *Query*. Akan tetapi, apabila proses penilaian berdasarkan hasil *query*-nya saja, maka mahasiswa dapat melakukan kecurangan. Kecurangan tersebut dilakukan dengan cara memasukkan *query* yang hasilnya sama dengan hasil dari *query* jawaban, tetapi *query* yang dituliskan jauh berbeda dengan *query* jawaban. Dan jika mahasiswa memasukkan *query* yang dapat merusak sistem, *query* tersebut akan langsung dieksekusi oleh sistem tanpa adanya pengecekan *query* terlebih dahulu. Hal tersebut merupakan beberapa kekurangan dari sistem yang dibuat. Kekurangan tersebut telah coba diselesaikan oleh (Damarriyanto, 2021), dengan cara menerapkan pengecekan similaritas SQL *Query* jawaban menggunakan perhitungan bobot TF-IDF kemudian menghitung similaritasnya menggunakan metode *Cosine Similarity*.

*Cosine similarity* telah terbukti sebagai metrik yang handal untuk penilaian similaritas antara dua *string* (Zhou et al., 2020). Agar hasil perhitungan similaritas lebih akurat maka SQL *Query* perlu direpresentasikan ke dalam bentuk lain, salah satunya menjadi bentuk *set of features* (Arzamasova et al., 2020). dalam tahap *preprocessing* akan memiliki 2 (dua) tahap yaitu proses standarisasi query dan pembobotan fitur. Pada tahap standarisasi, query akan dibagi menjadi beberapa grup seperti *select, from, where, groupby, orderby*. Selanjutnya dilakukan proses pembobotan fitur menggunakan metode TF (Azhir et al., 2021). Pengecekan similaritas dokumen menggunakan metode pembobotan TF-IDF dan *Cosine Similarity* telah sering digunakan seperti yang dilakukan pada (Amrizal, 2018; Toulouse, 2021).

Pembobotan TF-IDF dan metode *cosine similarity* pada (Damarriyanto, 2021) masih terdapat kelemahan, yaitu pengecekan *query* Kunci Jawaban masih terbatas 1 (satu) *query* kunci jawaban. Penambahan multi kunci jawaban untuk similaritas SQL perlu dilakukan. Karena, jika kunci jawaban ada 1 (satu) saja akan terdapat kondisi dimana mahasiswa menjawab suatu soal dengan contoh *query “SELECT nama, nim FROM Mahasiswa”* dan *query* tersebut bisa saja benar jawabannya namun karena yang terdaftar di sistem *query* kunci jawaban dengan *query “SELECT \* FROM Mahasiswa”* maka query jawaban dari mahasiswa yang bisa saja benar, pada saat dilakukan pengecekan similaritas SQL oleh sistem terdeteksi menjadi salah. Dan similaritas *query* terbatas pada *query SELECT*. karena salah satu tujuan penting dalam pembelajaran SQL adalah mendorong mahasiswa untuk belajar pernyataan SQL yang dapat mempengaruhi data didalam tabel basis data (Permpool et al., 2019). Disisi lain, basis data mempunyai peran penting dalam industri, beberapa diantaranya adalah entri data dan pemrosesan data yang efektif (Hamzah et al., 2019). Dalam penerapan basis data, tentu tidak terbatas pada perintah *SELECT* saja, Salah satu perintah yang digunakan adalah *INSERT*. Sehingga penambahan pengecekan perintah *INSERT* pada sistem similaritas *query* diperlukan.

Dikarenakan sistem *multi-instance database* pada (Damarriyanto, 2021) baru mendukung untuk perintah *SELECT* dan belum sepenuhnya mendukung untuk perintah *INSERT*. Maka dibutuhkan penyesuaian sistem yang akan digunakan pada proses penilaian.

Penambahan pengecekan similaritas *query insert* dan penerapan *multi* kunci jawaban menggunakan pembobotan TF-IDF dan metode *cosine similarity* pada sistem *automated assessment* MySQL, diharapkan dapat meningkatkan akurasi dan fleksibilitas untuk soal latihan maupun kuis pada studi kasus. Berdasarkan penjelasan diatas penulis terdorong untuk mengusulkan sebuah penelitian dengan judul “Pengembangan SQL Similarity Dengan Multi Kunci Jawaban Untuk Proses Penilaian Otomatis Pada Sistem Automated Assessment MySQL”.

## Rumusan Masalah

Berdasarkan latar belakang di atas, maka rumusan masalah yang dapat diambil adalah sebagai berikut:

1. Bagaimana tingkat akurasi penilaian *query insert* pada sistem *automated assessment* MySQL apabila ditambahkan pengecekan *insert query* similaritas SQL?
2. Bagaimana pengaruh dari penerapan multi kunci jawaban pada sistem *automated assessment* MySQL?

## Batasan Masalah

Permasalahan yang dibahas pada penelitian ini memiliki beberapa batasan sebagai berikut :

1. Aplikasi ini berjalan pada platform web.
2. Basis Data yang digunakan adalah MySQL.
3. Sistem *similarity* SQL yang digunakan pada penelitian ini adalah perintah *SELECT*, dan *INSERT*
4. Uji coba aplikasi dilakukan di Politeknik Negeri Malang untuk Dosen dan Mahasiswa Jurusan Teknologi Informasi.

## Tujuan

Tujuan dari dilakukaannya skripsi dengan judul “Pengembangan SQL Similarity Dengan Multi Kunci Jawaban Untuk Proses Penilaian Otomatis Pada Sistem Automated Assessment MySQL”, adalah sebagai berikut:

1. Mengetahui tingkat akurasi *query insert* pada sistem *automated assessment* MySQL apabila ditambahkan pengecekan *insert query similaritas* SQL.
2. Mengetahui pengaruh dari penerapan multi kunci jawaban pada sistem *automated assessment* MySQL.

## Manfaat

Manfaat yang didapatkan dari penelitian ini adalah dapat mengetahui pengaruh akurasi *query insert* pada sistem *automated assessment* ketika ditambahkan pengecekan *insert query similaritas* SQL dalam proses penilaiannya serta mengetahui pengaruh dari penerapan multi kunci jawaban pada sistem *automated assessment* MySQL.

# LANDASAN TEORI

## Studi Literatur

Terdapat beberapa penelitian sebelumnya yang dapat menunjang penelitian ini, topiknya adalah mengenai *query recommendation,*, *multi instance database* dan penelitian sebelumnya mengenai *automated assessment* similaritas SQL.

Penelitian mengenai *query recommendation* telah dilakukan oleh (Toulouse, 2021) dengan penelitian yang berjudul “*Slide-recommendation for improving students SQL solutions using natural language processing*” penelitian ini bertujuan untuk membantu mahasiswa pada *Universitas Otto-von-Guericke Magdeburg* mendapatkan *feedback* *slide* rekomendasi secara otomatis. *Feedback* didapatkan dengan cara memetakan latihan SQL dengan bahan ajar mereka yang sesuai, konten yang berhubungan dengan SQL pada latihan dan *slides* dari dosen dianalisis berdasarkan *cosine similarity* menggunakan pembobotan TF-IDF. Konsep untuk menyelesaikan Dengan cara ini mahasiswa yang memiliki masalah berhasil menyelesaikan latihan akan mendapatkan petunjuk mengenai *slide* dari dosen mana yang dapat membantu latihan tersebut. Hasil dari penelitian ini adalah tingkat akurasi terbaik dari prototipe dapat dicapai hingga tingkat akurasi sebesar 0.767 dan F=0.5 sebesar 0.505.

Penelitian mengenai sistem *multi-instance database* telah dilakukan oleh (Atchariyachanvanich et al., 2017). dengan penelitian yang berjudul “*Development of a MySQL Sandbox for processing SQL statements: Case of DML and DDL statements*”. Penelitian ini bertujuan untuk membuat sistem yang dapat membantu menyelesaikan masalah pada sistem penilaian SQL yang ada dengan menambah alternatif perintah SQL yang didukung, diharapkan nantinya system yang dibuat akan berdampak pada bertambahnya kemampuan dan pemahaman mahasiswa mengenai SQL seperti *query database*, administrasi *database*, dan pemrograman *database*, *MySQL Sandbox* menggunakan *RESTful web service* untuk menyelesaikan masalah yang dihadapi dengan tiga service penting yaitu *sandbox database creation, SQL statement processing,* dan *sandbox database resetting*. Dimana proses pembuatan *database* adalah dengan cara menggunakan fungsi dari MySQL yang dapat mengatur hak akses pengguna, nama *database* dihasilkan dari id mahasiswa dan id pertanyaan, jika ditemukan nama *database* yang sama maka *database* lama akan dihapus kemudian dibuat *database* baru dengan nama yang sama, dan *database* akan terhapus secara otomatis jika tidak aktif lebih dari 24 jam, pada *SQL Statement Processing* sistem menerima id pertanyaan, id mahasiswa dan *query* dari mahasiswa yang kemudian sistem akan menjalankan *query* pada *database* yang sesuai, dan terakhir pada *Sandbox Database Resetting* digunakan untuk mengembalikan *database* kembali ke kondisi awal setelah *sandbox* *database creation* dilakukan. Hasil dari penelitian ini adalah suatu sistem bernama *MySQL Sandbox* yang dapat digunakan oleh mahasiswa belajar mengenai *database* MySQL secara bebas tanpa adanya gangguan perubahan *query* oleh orang lain dikarenakan setiap mahasiswa memiliki *databasenya* sendiri yang dapat digunakan secara bebas.

Penelitian mengenai sistem *automated assessment* pada mysql telah dilakukan oleh (Puspitasari et al., 2019). Dengan penelitian yang berjudul *“Online judge MySQL for learning process of database practice course”*. Sistem tersebut berfungsi untuk melakukan penilaian otomatis terhadap jawaban yang dimasukkan oleh mahasiswa, jawaban dari mahasiswa yang dimaksud adalah *query*. Penilaian dilakukan dengan cara membandingkan hasil *query* yang dimasukkan oleh mahasiswa dan hasil *query* yang dimasukkan oleh dosen yang dijadikan sebagai kunci jawaban. Fitur yang terdapat dari dari sistem yang dibuat antara lain mahasiswa dapat mengikuti sesi kuis dan latihan, dosen juga dapat membuat studi kasus sebagai soal pada kuis maupun latihan, mahasiswa dan dosen dapat melihat nilai yang didapatkan oleh mahasiswa setelah mahasiswa menyelesaikan sesi kuis atau latihan yang dijalankan. Hasil dari penelitian ini adalah dari hasil pengujian yang dilakukan dengan cara diuji coba kan aplikasi untuk ujian *online* pada beberapa kelas secara bersamaan dapat disimpulkan bahwa aplikasi yang dibuat dapat mengoreksi hasil tes dengan cepat dan ringan.

Penelitian mengenai *automated assessment* similaritas *query* pada MySQL telah dilakukan oleh (Damarriyanto, 2021). Dengan penelitian yang berjudul *“Penerapan SQL Similarity Dan Multi-Instance Database Untuk Proses Penilaian Otomatis Pada Sistem Automated Assessment MySQL”*. Sistem ini merupakan pengembangan dari penelitian yang dilakukan oleh (Puspitasari et al., 2019) untuk sistem *automated assessment* pada MySQL, pengembangan yang dilakukan adalah menambahkan pengecekan similaritas *query* SQL pada proses *automated assessment* untuk memperbaiki kekurangan dari sistem mengenai sistem yang dapat dicurangi oleh siswa menggunakan *query* manipulatif. Similaritas *query* dilakukan dengan cara *query* jawaban mahasiswa dan kunci jawaban diubah ke dalam bentuk *set of features* sama seperti sebelumnya, kemudian menghitung bobot setiap *features* menggunakan TF-IDF, langkah terakhir adalah menghitung nilai similaritas menggunakan *cosine similarity* Selain itu untuk dapat lebih mudah mengelola studi kasus yang banyak, sistem diimplementasikan dengan arsitektur *multi-instance database*. Hasil dari penelitian ini adalah berdasarkan hasil pengujian proses *assessment* menggunakan 372 data *query* dengan *threshold similarity* di 60%, didapatkan tingkat akurasi adalah 97,04%. Serta pengaruh dari penerapan arsitektur *multi-instance database* yang lebih mudah untuk mengelola *database* yang digunakan sebagai referensi untuk menjalankan *query* nantinya.

Pada sistem *automated assessment* similaritas *query* yang telah dibuat pada (Damarriyanto, 2021) terdapat kekurangan. Antara lain pada dukungan proses pengecekan similaritas yang masih terbatas pada perintah *SELECT* saja, jika terbatas pada melakukan latihan atau kuis tentang *query* *SELECT* saja maka hal tersebut akan menjadi kurang fleksibel karena banyak perintah *query* SQL lainnya seperti *INSERT*, *ALTER* dan lain-lain dan perintah SQL seluruhnya itu sama pentingnya untuk diketahui, serta berdasarkan percobaan sistem secara pribadi ditemukan bahwa jika menjawab menggunakan query yang menggunakan pendekatan berbeda dengan query kunci tetapi hasilnya sama dan tidak menggunakan query manipulatif akan dianggap salah oleh sistem dikarenakan query memiliki tingkat similaritas yang rendah sehingga penilaian menjadi tidak akurat.

## Dasar Teori

### MySQL

MySQL adalah salah satu DBMS (*Database Mangement System*) populer yang telah ada sejak 1995. Dan MySQL memiliki 2 versi distribusi, yaitu versi komunitas (*open source*) yang gratis dan versi *enterprise* yang perlu membayar untuk dapat menggunakan MySQL ini (Malepati et al., 2019). MySQL ini dapat dimanfaatkan untuk mengelola data dengan cepat. Contohnya untuk menyimpan, menghapus, mengubah, mengambil dan menyortir data.

### SQL

SQL adalah bahasa yang digunakan untuk memanipulasi data pada basis data relasional, dan SQL telah berkembang menjadi bahasa yang digunakan untuk memanipulasi data antar beberapa teknologi *database* (Beaulieu, 2020). Pada bahasa SQL terdapat banyak perintah, beberapa perintah yang umum digunakan dan diketahui oleh mahasiswa maupun dosen pada bidang *computer science* adalah *SELECT* dan *INSERT*. Perintah *SELECT* digunakan untuk mengambil data dari tabel yang ada, sedangkan *INSERT* digunakan untuk memasukkan data kedalam tabel yang diinginkan.

### Automated Assessment

*Automated assessment* adalah suatu sistem yangdibuat untuk membantu dosen dan mahasiswa dalam proses pembelajaran. Dikarenakan pada saat penilaian yang masih dilakukan secara manual dosen dihadapkan dengan penilaian yang repetitif dan memiliki kewajiban memberikan *feedback* mendetail sehingga membuat beban kerja dosen meningkat, disisi lain mahasiswa membutuhkan *feedback* agar dapat terus belajar dan memahami kelebihan dan kekurangan yang dimiliki dibidang tertentu (Chen et al., 2020).

### TF-IDF

TF-IDF dihitung dengan cara memberikan bobot pada setiap term atau kata dengan membagikan frekuensi kata tersebut dengan jumlah dokumen yang memiliki kata tersebut di dalamnya, Dalam perhitungan bobot menggunakan TF-IDF, dihitung terlebih dahulu nilai TF (*Term Frequency*), kemudian dihitung nilai IDF (*Inverse Document Frequency*), langkah terakhir adalah melakukan perkalian antara hasil TF dan IDF.

TF (*Term Frequency*) yaitu berapa sering kata tersebut muncul dalam sebuah dokumen sehingga semakin banyak frekuensi kemunculan kata tersebut semakin besar nilai TF, untuk mencari TF dapat menggunakan persamaan (2.1).

Dimana adalah jumlah kata dalam 1 dokumen, sedangkan adalah total kata dalam dokumen.

Sedangkan IDF (*Inverse Document Frequency*) merupakan perhitungan dari bagaimana kata tersebut didistribusikan pada koleksi dokumen, nilai IDF akan semakin besar jika frekuensi kata yang muncul sedikit. IDF dapat dihitung dengan mencari nilai DF terlebih dahulu yang merupakan banyaknya frekuensi suatu kata muncul dalam koleksi dokumen. Untuk mencari IDF dapat menggunakan Persamaan (2.2).

Dimana n adalah jumlah dari seluruh dokumen, sedangkan DF adalah jumlah dokumen yang mengandung term.

Setelah mengetahui nilai TF dan IDF, untuk mengetahui nilai TF-IDF dapat menggunakan persamaan (2.3).

Dimana TF-IDF dapat dihitung setelah mendapatkan nilai TF dan IDF terlebih dahulu, kemudian hasil TF dan IDF dikalikan (Dalaorao et al., 2019)

### Cosine Similarity

*Cosine similarity* adalah suatu metode untuk menghitung kemiripan dari 2 vektor dengan cara menghitung nilai *cosine* dari 2 dokumen vektor (Soyusiawaty & Zakaria, 2018). Model ini dapat diimplementasikan ke semua 2 teks yang ingin dibandingkan (kalimat, paragraf, atau seluruh dokumen). Namun kekurangan dari *cosine similarity* tidak dapat mengerti arti *semantic* dari teks secara sempurna.

Pada studi kasus dokumen, sebuah dokumen direpresentasikan sebagai term vektor yang dimana merujuk pada kata yang ada pada dokumen. Sebuah dokumen dapat direpresentasikan ke dalam bentuk vektor.

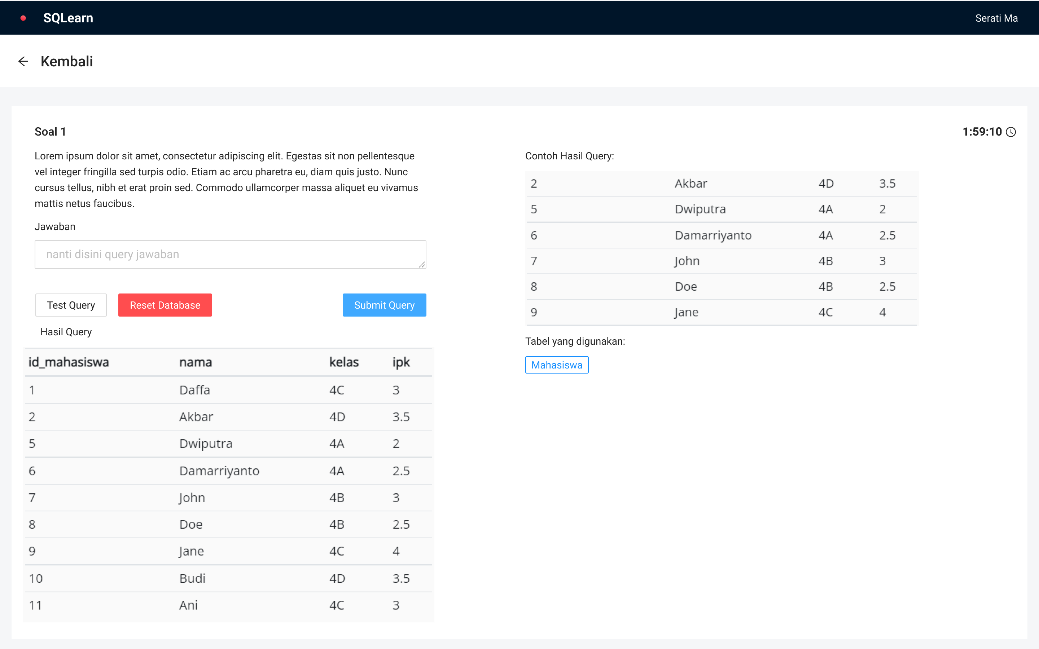
Dimana (0 <= i <= n) adalah angka desimal yang mengindikasikan frekuensi dari setiap kata dalam sebuah dokumen, serta dimensi vektor mengkorespon pada sebuah kata yang terdapat dalam dokumen. Berdasarkan similaritas vektor, kemiripan antara 2 vektor dapat menggunakan persamaan.

### SQLearn

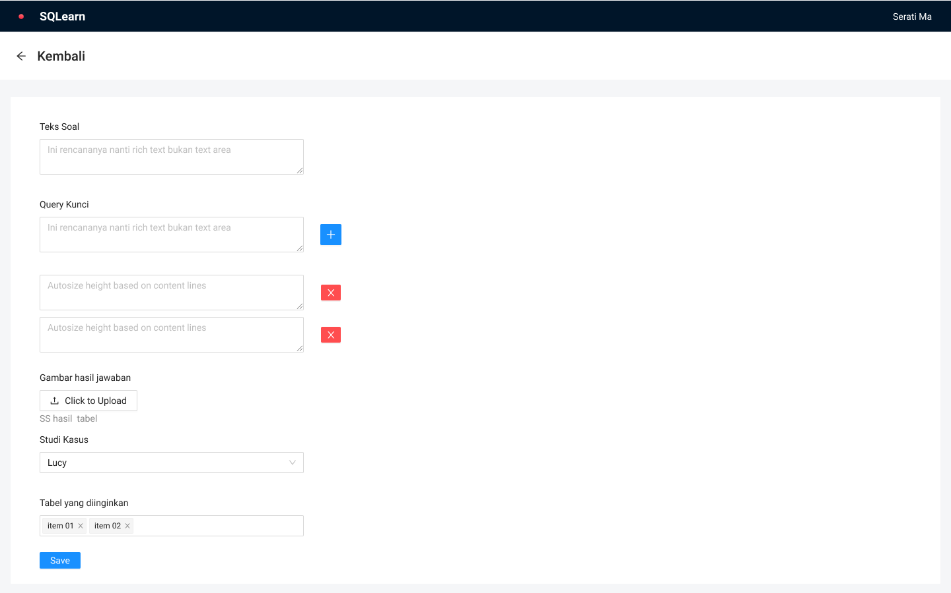
SQLearn adalah sebuah aplikasi pembelajaran SQL yang ditujukan agar mahasiswa lebih mudah memahami konsep SQL. pada aplikasi ini terdapat sebuah fitur yang memungkinkan mahasiswa untuk mengerjakan latihan soal atau kuis di dalam aplikasi ini. Untuk melakukan penilaian terhadap ujian tersebut, terdapat sistem penilaian otomatis yang diintegrasikan ke sistem. Sistem penilaian otomatis ini dibuat dengan mengimplementasikan TF-IDF dan perhitungan metode *cosine similarity* untuk melakukan pengecekan similaritas *query*. *Query* yang dimasukkan oleh mahasiswa akan dibandingkan dengan beberapa *query* kunci yang disediakan oleh dosen kemudian diambil nilai similaritas tertinggi dari hasil similaritasnya, jika nilai similaritas tidak melebihi batas *threshold* maka jawaban mahasiswa otomatis dianggap salah, tetapi jika melebihi batas *threshold* maka *query* yang dimasukkan oleh mahasiswa akan dieksekusi kemudian hasilnya akan dibandingkan dengan hasil dari eksekusi *query* kunci. Pertama dosen membuat studi kasus yang dapat digunakan untuk pembuatan soal dan dosen dapat menambahkan kunci jawaban lebih dari 1 (satu) agar penilaian lebih akurat, mahasiswa perlu menjawab soal dengan cara memasukkan *query* pada kolom jawaban, mahasiswa dapat mencoba *query* yang dimasukkan dengan cara menekan tombol “*Test Query*”, jika sudah yakin akan jawabannya dapat menekan tombol “*Submit Query*”, jika mahasiswa merasa salah dalam melakukan percobaan dan ingin memulai kembali kondisi database maka dapat menekan tombol “Reset Database”. mahasiswa dapat melihat nilai yang didapatkan langsung setelah menyelesaikan sesi. Berdasarkan penjelasan diatas, konsep aplikasi SQLearn seperti berikut:



Gambar 2.1 Konsep Multi Key Similarity



Gambar 2.2 Konsep Tampilan Pengerjaan Soal Mahasiswa



Gambar 2.3 Konsep Tampilan Tambah Soal Dosen

# METODOLOGI PENELITIAN

## Waktu dan Tempat Penelitian

Penelitian dilakukan di kampus Politeknik Negeri Malang. Penelitian dilaksanakan selama 4 bulan dimulai pada bulan Januari 2022 sampai dengan April 2022.

## Teknik Pengumpulan Data

Data penelitian yang diolah akan berupa *query* yang biasa digunakan untuk latihan maupun kuis pada mata kuliah basis data.

1. Penelitian Lapangan (*Field Research*)

Wawancara (*interview*) dengan cara tanya jawab langsung dengan pihak yang bersangkutan, guna memperoleh informasi yang akurat, dan narasumber yang akan diwawancarai adalah dosen mata kuliah basis data Jurusan Teknologi Informasi Politeknik Negeri Malang, dimana penulis menyiapkan pertanyaan mengenai perbandingan pembelajaran praktikum dan kuis yang diterapkan, baik dari segi waktu, kemudahan dan akurasi.

1. Penelitian Kepustakaan (*Library Research*)

Penelitian yang dilakukan dengan cara membaca dan mempelajari literatur, dengan maksud untuk menempatkan landasan teori mengenai masalah pokok yang akan dibahas.

## Teknik Pengolahan Data

Untuk dapat menggunakan metode *cosine similarity* pada sistem *automated assessment similarity* harus menyiapkan minimal 2 *query*, yaitu *query* *input* dari mahasiswa dan *query* kunci, jika terdapat lebih dari 1 *query* kunci maka sistem akan membandingkan satu persatu terlebih dahulu *similarity* antara *query* *input* mahasiswa dengan beberapa *query* kunci yang disiapkan, kemudian mencari nilai *similaritas* tertinggi dari nilai *similaritas* yang ada. *Output* dari metode ini berupa nilai *similarity* tertinggi dari *query* *input* dengan beberapa *query* kunci yang ada.

Sebagai contoh terdapat 3 *query* dengan detail 1 *query* *input* dari mahasiswa dan 2 *query* kunci dari dosen seperti berikut:

*Query* *input* = “INSERT INTO mahasiswa VALUES (“YASA”, 1123144, “MI-2M”)”

*Query* kunci 1 = “INSERT INTO mahasiswa (nama, nim, kelas) VALUES (“YASA”, 1123144, “MI-2M”)”

*Query* kunci 2 = “INSERT INTO mahasiswa VALUES (“YASA”, 1123144, “MI-2M”)”

### Menentukan *Query* yang akan dilakukan pengecekan similaritas

Langkah awal untuk memulai perhitungan *similarity* adalah membutuhkan 2 *query*. 2 *query* disini adalah 1 *query* *input* dari mahasiswa dan 1 lagi *query* kunci sebagai pembanding, *query* *input* misalnya:

A = *“INSERT INTO mahasiswa VALUES (“YASA”, 1123144, “MI-2M”)”*

Kemudian 1 lagi query dari query kunci misalnya:

B = *“INSERT INTO mahasiswa (nama, nim, kelas) VALUES (“YASA”, 1123144, “MI-2M”)”*

### Mencari set of features dari query

Setelah mengetahui *query* yang akan dilakukan proses *similarity* selanjutnya *query* akan dipisahkan dan kemudian dikelompokkan berdasarkan komponen seperti *INSERTINTO, VALUES, SELECT, FROM, SELECT, FROM, WHERE, GROUPBY,* dan *ORDERBY.*

Query tadi akan membentuk *Set of Features* seperti berikut:

1. *Set of Feature Query* A

Tabel 3.1 Set of Features A

|  |  |  |
| --- | --- | --- |
| **No** | **Feature** | **Jumlah** |
| 1 | INSERT\_mahasiswa | 1 |
| 2 | VALUES\_1\_Constant | 3 |

1. *Set of Feature Query* B

Tabel 3.2 Set of Features B

|  |  |  |
| --- | --- | --- |
| **No** | **Feature** | **Jumlah** |
| 1 | INSERT\_mahasiswa | 1 |
| 2 | INSERT\_nama | 1 |
| 3 | INSERT\_nim | 1 |
| 4 | INSERT\_kelas | 1 |
| 5 | VALUES\_1\_Constant | 3 |

### Menghitung TF-IDF

1. Menghitung TF (*Term Frequency)*

Contohnya pada *feature* “INSERT\_mahasiswa” pada *query* A maka untuk mendapatkan nilai TF akan seperti persamaan (3.1):

Sehingga hasil dari perhitungan TF adalah sebagai berikut:

Tabel 3.3 Hasil TF

|  |  |  |
| --- | --- | --- |
| **Feature** | **A** | **B** |
| INSERT\_mahasiswa | 0.25 | 0.142857 |
| INSERT\_nama | 0 | 0.142857 |
| INSERT\_nim | 0 | 0.142857 |
| INSERT\_kelas | 0 | 0.142857 |
| VALUES\_1\_Constant | 0.75 | 0.428571 |

1. Mencari DF (*Document Frequency)*

Nilai DF didapatkan dengan menggabungkan jumlah dari *feature* yang sama antara *query*, sehingga nilai DF yang didapatkan seperti berikut:

Tabel 3.4 Hasil DF

|  |  |
| --- | --- |
| **Feature** | **DF** |
| INSERT\_mahasiswa | 2 |
| INSERT\_nama | 1 |
| INSERT\_nim | 1 |
| INSERT\_kelas | 1 |
| VALUES\_1\_Constant | 2 |

1. Menghitung IDF

Contohnya pada *feature* “INSERT\_nama” maka untuk menghitung IDF dapat seperti persamaan (3.2)

Sehinga nilai IDF yang didapatkan untuk seluruh *feature* adalah sebagai berikut:

Tabel 3.5 Hasil IDF

|  |  |
| --- | --- |
| **Feature** | **IDF** |
| INSERT\_mahasiswa | 1 |
| INSERT\_nama | 1.3013 |
| INSERT\_nim | 1.3013 |
| INSERT\_kelas | 1.3013 |
| VALUES\_1\_Constant | 1 |

1. Menghitung TF-IDF

Untuk mendapatkan nilai TF-IDF dilakukan dengan cara mengkalikan nilai TF dan nilai IDF, sehinggga untuk menghitung nilai TF-IDF *feature* INSERT\_nim di *Set of Feature* B dapat dengan cara seperti pada persamaan (3.3)

Sehingga dapat diketahui nilai TF-IDF seperti berikut:

Tabel 3.6 Hasil TF-IDF

|  |  |  |
| --- | --- | --- |
| **Feature** | **A** | **B** |
| INSERT\_mahasiswa | 0.25 | 0.142857 |
| INSERT\_nama | 0 | 0.185861 |
| INSERT\_nim | 0 | 0.185861 |
| INSERT\_kelas | 0 | 0.185861 |
| VALUES\_1\_Constant | 0.75 | 0.428571 |

### Meghitung Cosine Similarity

1. Perkalian Skalar

Langkah awal menghitung similaritas menggunakan metode *cosine similarity* adalah melakukan perkalian skalar antara TF-IDF A dan TF-IDF B, atau dengan nama lain AxB dan kemudian menghitung totalnya.

Tabel 3.7 Perkalian Skalar AxB

|  |  |
| --- | --- |
| **Feature** | **AxB** |
| INSERT\_mahasiswa | 0.035714286 |
| INSERT\_nama | 0 |
| INSERT\_nim | 0 |
| INSERT\_kelas | 0 |
| VALUES\_1\_Constant | 0.3214285714 |
| **TOTAL** | 0.3571428571 |

1. Menghitung total kuadrat dan akar kuadrat

Selanjutnya adalah mencari hasil kuadrat dari TF-IDF A, TF-IDF B kemudian dihitung total untuk setiap TF-IDF, setelah didapatkan total maka akan dihitung akar kuadratnya

Tabel 3.8 Hasil Total Kuadrat dan Akar Kuadrat

|  |  |  |
| --- | --- | --- |
| **Feature** | **A** | **B** |
| INSERT\_mahasiswa | 0.0625 | 0.020408 |
| INSERT\_nama | 0 | 0.034544 |
| INSERT\_nim | 0 | 0.034544 |
| INSERT\_kelas | 0 | 0.034544 |
| VALUES\_1\_Constant | 0.5625 | 0.1836734694 |
| **TOTAL Kuadrat** | 0.625 | 0.3077150439 |
| **Akar Kuadrat** | 0.790569415 | 0.5547206899 |

1. Menghitung *Cosine Similarity*

Untuk menghitung *cosine similarity* dapat menggunakan cara total dari perkalian skalar AxB dibagi dengan hasil perkalian akar kuadrat A dan B, seperti pada persamaan (3.4)

Dari hasil persamaan (3.4) didapatkan persentase kemiripan antar *query* sebesar 81.44%

### Ulangi untuk similaritas dengan *query* kunci yang lain

Dikarenakan terdapat lebih dari 1 *query* kunci yang disediakan, maka langkah selanjutnya adalah mencari persentase kemiripan antara *query input* dengan query kunci lainnya hingga *query* kunci terakhir yang disediakan menggunakan cara yang sama. Seperti contoh untuk *query input* dengan *query* kunci ke-2 berikut.

*Query* A = “INSERT INTO mahasiswa VALUES (“YASA”, 1123144, “MI-2M”)”

*Query* B = “INSERT INTO mahasiswa VALUES (“YASA”, 1123144, “MI-2M”)”

Setelah mengetahui *query* mana yang akan dinilai kembali maka selanjutnya menghitung nilai similaritas seperti yang sudah dilakukan, sehingga untuk nilai similaritas yang ke-2 didapat hasil similaritas sebesar 100%.

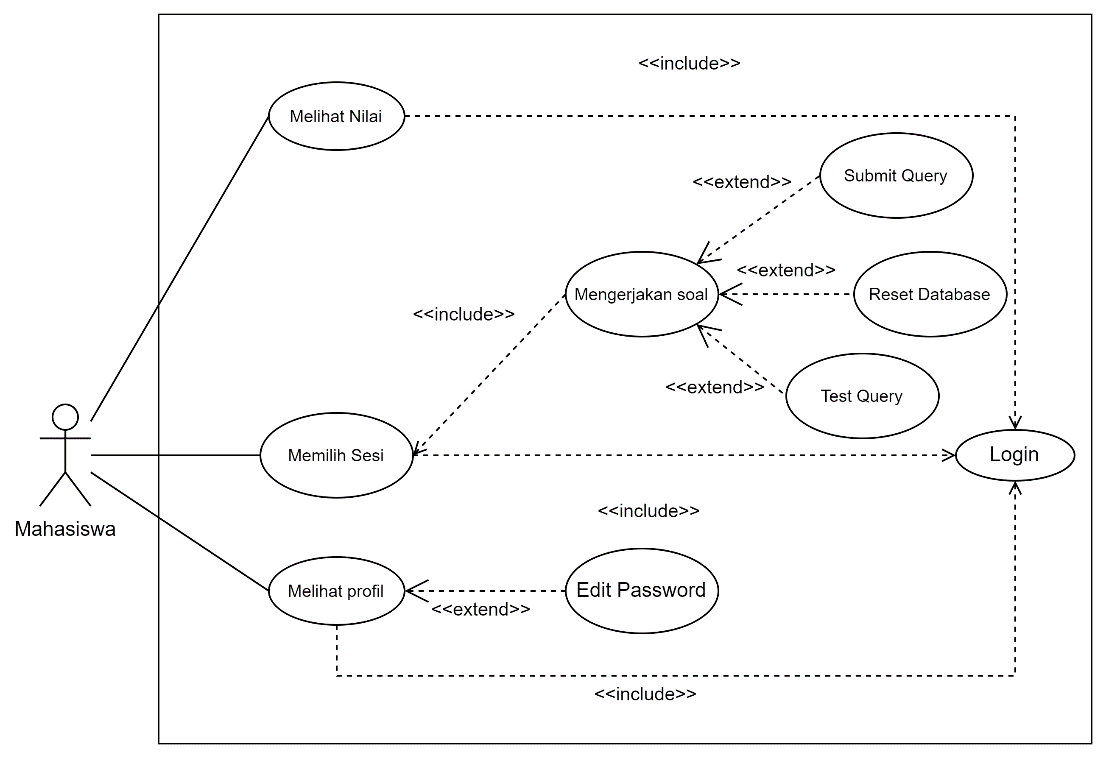
### Mencari nilai similaritas tertinggi

Untuk mencari nilai similaritas tertinggi dilakukan dengan cara membandingkan antar nilai similaritas yang sudah didapatkan. sehingga akan mendapatkan nilai similaritas tertinggi diantara seluruh nilai similaritas yang ada. Sebagai contoh dikarenakan ada 2 hasil similaritas yang didapat dengan nilai 0.8143809302 dan 1 maka dapat simpulkan nilai similaritas yang ke-2 adalah yang tertinggi dengan nilai 1 atau similaritas sebesar 100%.

## Desain Sistem

### Use Case

1. Mahasiswa



Gambar 3.1 Use Case Mahasiswa

Jika sebagai mahasiswa maka dapat melihat nilai Latihan atau kuis yang telah dilakukan sebelumnya. Mahasiswa juga dapat melihat profil dan mengubah *password* jika menginginkan untuk mengubah *password*. Mahasiswa juga dapat melihat daftar sesi yang tersedia. Dan mahasiswa juga dapat mengerjakan soal yang tersedia dan melakukan *test query* jika ingin mencoba *query* jawaban yang terpikirkan atau *submit query* jika sudah yakin dengan *query* jawaban yang telah terpikirkan jika ingin mengembalikan database kebentuk semula. Agar dapat melakukan semua hal diatas mahasiswa harus login terlebih dahulu dengan kombinasi nim dan password.

1. Dosen

Diagram

Description automatically generated

Gambar 3.2 Use Case Dosen

Jika sebagai dosen maka dapat melihat nilai seluruh mahasiswa yang telah melaksanakan sesi Latihan atau kusi dan dapat melihat nilai secara detail. Pada menambahkan studi kasus yang akan membuat database studi kasus baru dan akan digunakan untuk soal yang akan dibuat. Pada mengelola soal dapat membuat soal dan memilih studi kasus mana yang akan digunakan oleh soal yang dibuat dan memasukkan *query* kunci yang telah disiapkan. Pada mengelola paket soal dapat membuat paket soal dan memilih beberapa soal kedalam paket soal yang telah dibuat. Pada mengelola kelas dapat membuat kelas dan menambahkan mahasiswa dari daftar excel dan membuat akun yang dapat digunakan oleh mahasiswa. Pada melihat profil dapat melihat profil akun dari dosen dan mengubah *password* jika diinginkan.

Secara garis besar prosedur dosen untuk menggunakan aplikasi adalah sebagai berikut:

membuat studi kasus dengan cara mengupload file backup sql dan nama studi kasus.

membuat soal dan memasukkan kunci jawaban lebih dari 1 jika diperlukan dan memilih akan menggunakan studi kasus yang telah dibuat.

membuat paket soal dengan cara memasukkan nama paket soal dan memilih daftar soal yang telah dibuat.

menambahkan kelas jika belum tersedia dengan cara memasukkan nama kelas dan daftar siswa dalam bentuk excel.

membuat sesi dengan cara memasukkan nama, jadwal, kelas sesi dan memilih paket soal yang telah dibuat.

1. Admin

Diagram

Description automatically generated

Gambar 3.3 Use Case Admin

Jika sebagai admin maka dapat mengelola dosen yaitu menambahkan akun dosen, menghapus akun dosen. Mengelola aturan penilaian yaitu mengubah nilai yang didapatkan berdasarkan banyaknya percobaan. Mengelola *threshold* yang digunakan pada proses similarity sebelum query dibandingkan hasilnya. Dan dapat melihat profil admin serta mengubah password jika diinginkan.

### Activity Diagram

1. Activity Diagram Mahasiswa

Diagram

Description automatically generated

Gambar 3.4 Activity Diagram Mahasiswa

Dari diagram pada gambar diatas dijelaskan bahwa alur dari penggunaan aplikasi oleh mahasiswa dilakukan dengan cara mahasiwa *login* dengan kombinasi nim dan *password* kemudian memilih sesi yang tersedia dan dapat dikerjakan. Setelah itu sistem akan mendapatkan data id sesi dan id mahasiswa yang akan menghasilkan nama *database* dan menyiapkan *database* yang sesuai, kemudian *database* management akan membuat *database* sesuai dengan apa yang telah didefinisi pada sistem dan setelah itu mahasiswa dapat mulai mengerjakan soal. Ketika mahasiswa menjawab soal maka sistem akan menerima *query* dari mahasiswa dan mengambil daftar kunci jawaban yang tersimpan, kemudian selanjutnya adalah proses pengecekan similaritas untuk mendapatkan nilai similaritas tertinggi. Setelah mendapatkan nilai similaritas maka akan dicek apakah nilai similaritas diatas *threshold*, jika diatas nilai *threshold* maka *query* akan dijalankan dan dibandingkan hasilnya kemudian disimpan hasilnya, jika tidak maka akan menyimpan hasilnya beserta pesan pemberitahuan pesan sukses atau error yang akan ditampilkan. Setelah itu mahasiswa dapat mengerjakan soal selanjutnya hingga soal terakhir, setelah soal telah terjawab semua maka mahasiswa dapat menyelesaikan sesi untuk mengetahui nilai yang diperoleh.

1. Activity Diagram Dosen

Diagram, schematic

Description automatically generated

Gambar 3.5 Activity Diagram Dosen

Dari diagram pada gambar diatas dijelaskan bahwa alur dari penggunaan aplikasi oleh dosen dilakukan dengan cara dosen login menggunakan kombinasi username dan password kemudian membuat studi kasus yang akan digunakan daftar soal, pada pembuatan studi kasus sistem akan menyimpan data studi kasus yang dibuat dan *database* studi kasus akan dibuat. Kemudian membuat soal beserta kunci jawaban, soal akan menggunakan studi kasus yang telah dibuat sebelumnya, sistem akan menyimpan soal dan kunci jawaban yang dibuat oleh dosen. Setelah studi kasus dan soal beserta kunci jawaban siap maka dosen dapat membuat paket soal dan memilih kumpulan soal yang telah dibuat kedalam paket soal. Setelah paket soal sudah tersedia maka dosen dapat membuat sesi agar mahasiswa dapat melakukan latihan / kuis telah tersedia. kemudian dosen dapat melihat nilai yang dihasilkan oleh mahasiswa yang telah menyelesaikan sesi.

### Flowchart

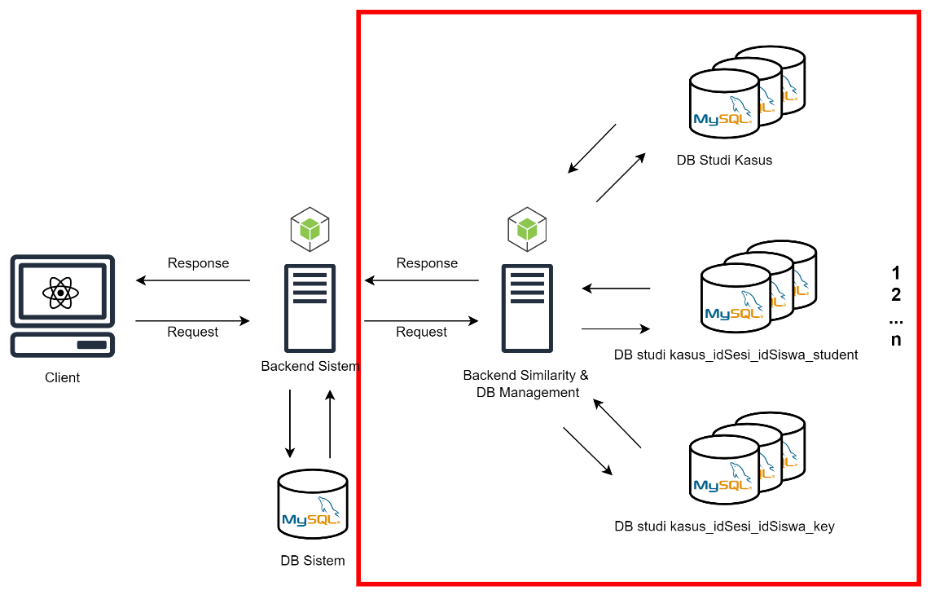
Diagram

Description automatically generated

Gambar 3.6 Flowchart Similarity

Pada gambar diatas dijelaskan bahwa proses perhitungan similaritas dimulai dari mengetahui *query input* dan daftar *query* kunci kemudian dilakukan perulangan dengan membandingkan *query input* dengan setiap *query* kunci. Pada awalnya *query* diubah menjadi *Abstract Syntax Tree (AST)* agar memudahkan proses pembuatan *Set of Features*, setelah mendapatkan *Abstract Syntax Tree (AST)* selanjutnya parsing AST menjadi *set of features*. Setelah mendapatkan *set of features* dapat menghitung TF-IDF, setelah diketahui nilai TF-IDF dapat melakukan perhitungan *Cosine Similarity.* Setelah mendapatkan nilai similaritas dari seluruh perbandingan *query* kunci dengan *query input* maka selanjutnya adalah mencari nilai similaritas tertinggi dari seluruh hasil yang ada sehingga akan didapatkan nilai similaritas tertinggi.

### Arsitektur Sistem



Gambar 3.7 Arsitektur Sistem

Berdasarkan gambar diatas, dapat dijelaskan bahwa mahasiswa dapat berinteraksi dengan aplikasi SQLearn yang akan dibuat dengan *library frontend* Next JS, Next JS dapat membagi komponen UI dari aplikasi yang dapat memudahkan pengembangan dikarenakan UI dari aplikasi akan menjadi komponen-komponen yang memiliki tanggung jawab tersendiri sehingga jika ada kesalahan pada satu komponen tidak akan mempengaruhi kepada komponen UI lainnya.

Untuk backend sistem menggunakan 2 backend sistem yang akan menggunakan *node.js* dengan *micro-framework express*. Backend pertama adalah backend sistem yang bertugas mengelola studi kasus, soal, sesi, autentikasi dll. Dan untuk backend kedua dibuat untuk proses similaritas dan *database management*, alasan memiliki 2 backend adalah untuk memudahkan proses *management database* dan komputasi similaritas agar lebih optimal.

Perbedaan dengan arsitektur sistem pada penelitian sebelumnya terletak pada sistem backend *similarity* dan *database management.* Yang nantinya akan ditambahkan multi kunci jawaban dan dukungan pengecekan perintah *INSERT* pada proses *similarity*, dan perubahan pada *database management* agar dapat mendukung menjalankan perintah *INSERT.* Sehingga *multi-instance database* tidak terbatas pada *database* studi kasus saja tetapi akan memiliki *database* untuk mahasiswa dapat menjalankan *query* secara terpisah dari mahasiswa lain dan *database* untuk menjalankan *query* kunci yang nantinya akan dibandingkan hasilnya.

Secara keseluruhan sistem, perbedaan antara sistem saat ini dengan sistem dari penelitian sebelumnya adalah nantinya mahasiswa dapat melakukan reset database untuk mengembalikan bentuk database kedalam bentuk semula, dan dapat mengubah password, dosen dapat menambahkan multi kunci jawaban dan mengubah password, admin dapat mengubah aturan penilaian dan dapat mengubah password, penambahan sistem untuk pengecekan multi kunci jawaban, similarity sql akan mendukung perintah insert dan multi instance database tidak hanya studi kasus tetapi setiap mahasiswa yang mengerjakan sesi akan memiliki database uniknya sendiri.

## Uji Coba Sistem

Pengujian akan terdiri dari pengujian untuk multi kunci jawaban dan insert similarity.

### Multi Kunci Jawaban

Uji coba sistem untuk multi kunci jawaban dilakukan dengan menggunakan data query dari penelitian sebelumnya. Data query akan dilakukan pengolahan dengan proses similarity seperti yang telah dijelaskan. Hasil yang didapat nantinya akan dibandingkan dengan hasil dari penelitian sebelumnya untuk mengetahui pengaruh dari penerapan multi kunci jawaban terhadap sistem automated assessment mysql.

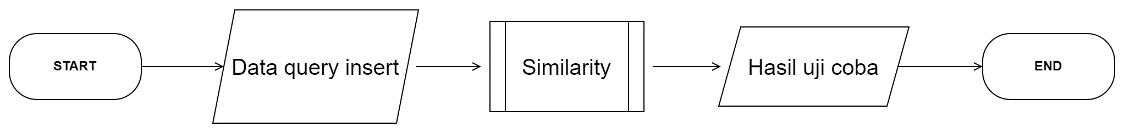
Diagram

Description automatically generated

Gambar 3.8 Uji Coba Multi Kunci Jawaban

### Insert Similarity

Uji coba sistem untuk *insert similarity query SQL* dilakukan dengan menggunakan data *query* yang didapatkan dari pengujian kepada Mahasiswa yang sedang mengikuti mata kuliah basis data di Jurusan Teknologi Informasi Politeknik Negeri Malang. Data query yang didapatkan akan dilakukan pengolahan dengan proses similarity seperti yang telah dijelaskan. Hasil dari pengolahan data query adalah nilai similarity tertinggi yang dapat digunakan untuk mengetahui nilai akurasi *query insert similarity* pada sistem *automated assessment* MySQL.



Gambar 3.9 Uji Coba Insert Similarity

# JADWAL PENELITIAN

Tabel 4.1 Tabel Jadwal Kegiatan



DAFTAR PUSTAKA

Amrizal, V. (2018). PENERAPAN METODE TERM FREQUENCY INVERSE DOCUMENT FREQUENCY (TF-IDF) DAN COSINE SIMILARITY PADA SISTEM TEMU KEMBALI INFORMASI UNTUK MENGETAHUI SYARAH HADITS BERBASIS WEB (STUDI KASUS: HADITS SHAHIH BUKHARI-MUSLIM). *JURNAL TEKNIK INFORMATIKA*, *11*(2), 149–164. https://doi.org/10.15408/jti.v11i2.8623

Arzamasova, N., Bohm, K., Goldman, B., Saaler, C., & Schaler, M. (2020). On the Usefulness of SQL-Query-Similarity Measures to Find User Interests. *IEEE Transactions on Knowledge and Data Engineering*, *32*(10), 1982–1999. https://doi.org/10.1109/TKDE.2019.2913381

Atchariyachanvanich, K., Nalintippayawong, S., & Permpool, T. (2017). Development of a MySQL Sandbox for processing SQL statements: Case of DML and DDL statements. *2017 14th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 1–6. https://doi.org/10.1109/JCSSE.2017.8025930

Azhir, E., Jafari Navimipour, N., Hosseinzadeh, M., Sharifi, A., & Darwesh, A. (2021). An automatic clustering technique for query plan recommendation. *Information Sciences*, *545*, 620–632. https://doi.org/10.1016/j.ins.2020.09.037

Beaulieu, A. (2020). *Learning SQL: Generate, Manipulate, and Retrieve Data* (3rd ed.). O’Reilly Media. https://www.oreilly.com/library/view/learning-sql-3rd/9781492057604/

Chen, H.-M., Nguyen, B.-A., Yan, Y.-X., & Dow, C.-R. (2020). Analysis of Learning Behavior in an Automated Programming Assessment Environment: A Code Quality Perspective. *IEEE Access*, *8*, 167341–167354. https://doi.org/10.1109/ACCESS.2020.3024102

Dalaorao, G. A., Sison, A. M., & Medina, R. P. (2019). Integrating Collocation as TF-IDF Enhancement to Improve Classification Accuracy. *2019 IEEE 13th International Conference on Telecommunication Systems, Services, and Applications (TSSA)*, 282–285. https://doi.org/10.1109/TSSA48701.2019.8985458

Damarriyanto, D. A. D. (2021). *Penerapan SQL Similarity dan Multi-Instance Database Untuk Proses Penilaian Otomatis Pada Sistem Automated Assessment MySQL*. Politeknik Negeri Malang.

Daradoumis, T., Marquès Puig, J. M., Arguedas, M., & Calvet Liñan, L. (2019). Analyzing students’ perceptions to improve the design of an automated assessment tool in online distributed programming. *Computers & Education*, *128*, 159–170. https://doi.org/10.1016/j.compedu.2018.09.021

Hamzah, M. L., Rukun, K., Rizal, F., Purwati, A., Hamzah, H., & Zarnelly. (2019). A review of increasing teaching and learning database subjects in computer science. *Espacios*, *40*, 6.

Jiandong, H., Jinyu, S., & Suojuan, Z. (2018). Study on the Iterative Teaching Method in Database Curriculum. *2018 9th International Conference on Information Technology in Medicine and Education (ITME)*, 615–617. https://doi.org/10.1109/ITME.2018.00141

Luchoomun, T., Chumroo, M., & Ramnarain-Seetohul, V. (2019). A Knowledge Based System for Automated Assessment of Short Structured Questions. *2019 IEEE Global Engineering Education Conference (EDUCON)*, 1349–1352. https://doi.org/10.1109/EDUCON.2019.8725139

Malepati, T., Shah, B., & Vanier, E. (2019). *Advanced MySQL 8: Discover the Full Potential of MySQL and Ensure High Performance of Your Database*. Packt Publishing Ltd. https://www.packtpub.com/product/advanced-mysql-8/9781788834445

Nalintippayawong, S., Atchariyachanvanich, K., & Julavanich, T. (2017). DBLearn: Adaptive e-learning for practical database course — An integrated architecture approach. *2017 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, 109–114. https://doi.org/10.1109/SNPD.2017.8022708

Obionwu, V., Broneske, D., Hawlitschek, A., Köppen, V., & Saake, G. (2021). SQLValidator – An Online Student Playground to Learn SQL. *Datenbank-Spektrum*, *21*(2), 73–81. https://doi.org/10.1007/s13222-021-00372-0

Permpool, T., Nalintippayawong, S., & Atchariyachanvanich, K. (2019). Interactive SQL Learning Tool with Automated Grading using MySQL Sandbox. *2019 IEEE 6th International Conference on Industrial Engineering and Applications (ICIEA)*, 928–932. https://doi.org/10.1109/IEA.2019.8715175

Puspitasari, D., Arhandi, P. P., Saputra, P. Y., Syaifudin, Y. W., Himawan, H. A., & Sholihah, P. A. (2019). Online judge MySQL for learning process of database practice course. *IOP Conference Series: Materials Science and Engineering*, *523*(1), 012046. https://doi.org/10.1088/1757-899X/523/1/012046

Soyusiawaty, D., & Zakaria, Y. (2018). Book Data Content Similarity Detector With Cosine Similarity (Case study on digilib.uad.ac.id). *2018 12th International Conference on Telecommunication Systems, Services, and Applications (TSSA)*, 1–6. https://doi.org/10.1109/TSSA.2018.8708758

Taipalus, T., & Perälä, P. (2019). What to Expect and What to Focus on in SQL Query Teaching. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 198–203. https://doi.org/10.1145/3287324.3287359

Toulouse, V. (2021). Slide-recommendation for improving students SQL solutions using natural language processing. *Otto-von-Guericke University Magdeburg*, 73.

Zhou, Y., Yang, S., li, yixiao, chen, Y., Yao, J., & Qazi, A. (2020). Does the review deserve more helpfulness when its title resembles the content? Locating helpful reviews by text mining. *Information Processing & Management*, *57*(2), 102179. https://doi.org/10.1016/j.ipm.2019.102179