



PETUNJUK PRAKTIKUM BASIS DATA

Penyusun:

Jefree Fahana, S.T., M.Kom.
Dewi Soyusiawaty, S.T., M.T.
Ir. Sri Winiarti, S.T., M.Cs.
Miftahurahma Rosyda, S.Kom., M.Eng.
Dr. Ardiansyah, S.T., M.Cs.
Ninda Khoirunnisa, S.T., M.Sc.

2025

HAK CIPTA

PETUNJUK PRAKTIKUM BASIS DATA

Copyright© 2025,

Jefree Fahana, S.T., M.Kom.

Dewi Soyusiawaty, S.T., M.T.

Ir. Sri Winiarti, S.T., M.Cs.

Miftahurahma Rosyda, S.Kom., M.Eng.

Dr. Ardiansyah, S.T., M.Cs.

Ninda Khoirunnisa, S.T., M.Sc.

Hak Cipta dilindungi Undang-Undang

Dilarang mengutip, memperbanyak atau mengedarkan isi buku ini, baik sebagian maupun seluruhnya, dalam bentuk apapun, tanpa izin tertulis dari pemilik hak cipta dan penerbit.

Diterbitkan oleh:

Program Studi S1 Informatika

Fakultas Teknologi Industri

Universitas Ahmad Dahlan

Jalan Ring Road Selatan, Tamanan, Banguntapan, Bantul Yogyakarta 55166

Penulis : Jefree Fahana, S.T., M.Kom.

Dewi Soyusiawati, S.T., M.T.

Ir. Sri Winiarti, S.T., M.Cs.

Miftahurahma Rosyda, S.Kom., M.Eng.

Dr. Ardiansyah

Ninda Khoirunnisa, S.T., M.Sc.

Editor : Laboratorium S1 Informatika, Universitas Ahmad Dahlan

Desain sampul : Laboratorium S1 Informatika, Universitas Ahmad Dahlan

Tata letak : Laboratorium S1 Informatika, Universitas Ahmad Dahlan

Ukuran/Halaman : 21 x 29,7 cm /128 halaman

Didistribusikan oleh:



Laboratorium S1 Informatika

Universitas Ahmad Dahlan

Jalan Ring Road Selatan, Tamanan, Banguntapan, Bantul Yogyakarta 55166

Indonesia



KATA PENGANTAR

Puji dan syukur ke hadirat Allah SWT yang memberikan rahmat dan hidayah sehingga penyusunan revisi ketiga Petunjuk Praktikum Basis Data Edisi Kurikulum *Outcome Based Education* (OBE) ini akhirnya bisa diselesaikan. Petunjuk Praktikum ini disusun sebagai panduan untuk pelaksanaan praktikum mata kuliah Basis Data di lingkungan Program Studi S1 Informatika Universitas Ahmad Dahlan.

Materi yang disajikan sudah diurutkan disesuaikan dengan RPS berbasis OBE matakuliah Basis Data, sehingga mahasiswa dapat dengan mudah memahami. Setiap pertemuan diberikan terlebih dahulu penjelasan mengenai teori terkait materi yang akan dilakukan dan dilanjutkan dengan kegiatan praktikum.

Penyusun menyadari masih ada kekurangan pada penulisan ini, oleh karenanya kritik dan saran yang membangun diharapkan dapat memperbaiki untuk tahun-tahun berikutnya. Terima kasih kepada semua pihak yang telah membantu baik secara langsung maupun tidak, terhadap terselesaiannya petunjuk praktikum ini.

Yogyakarta, 29 September 2025

Penyusun



DAFTAR PENYUSUN

Jefree Fahana, S.T., M.Kom.



NIDN : 0528058401
NIPM : 60160979
Jabatan Fungsional: Lektor
Bidang Minat : Sistem Informasi
Email : jefree.fahana[at]tif.uad.ac.id
Penyusun Bab 3 dan 8 Petunjuk Praktikum Basisdata Edisi OBE

Dewi Soyusiawaty, S.T., M.T



NIDN : 0530077601
NIPM : 19760730 200409 011 0951361
Jabatan Fungsional: Lektor Kepala
Bidang Minat : Sistem Cerdas, Pemrosesan Bahasa Alami
Email : dewi.soyusiawaty[at]tif.uad.ac.id
Penyusun Bab 2 dan 7 Petunjuk Praktikum Basisdata Edisi OBE

Ir. Sri Winiarti, S.T., M.Cs.



NIDN : 0516127501
NIPM : 19751216 200103 011 0880702
Jabatan Fungsional: Lektor Kepala
Bidang Keahlian : Sistem Cerdas
Email : sri.winiarti[at]tif.uad.ac.id
Penyusun Bab 1 Petunjuk Praktikum Basisdata Edisi OBE

Miftahurrahma Rosyda, S.Kom., M.Eng.



NIDN : 0515069001
NIPM : 19900615 201908 011 1029280
Jabatan Fungsional: Lektor
Bidang Keahlian : Sistem cerdas, bioinformatika, big data
Email : miftahurrahma.rosyda[at]tif.uad.ac.id
Penyusun Bab 6 Petunjuk Praktikum Basis data Edisi OBE

Dr. Ardiansyah, S.T., M.Cs.



NIDN : 0523077902
NIPM : 19790723 200309 111 0932301
Jabatan Fungsional: Lektor
Bidang Keahlian : Software Engineering
Email : ardiansyah[at]tif.uad.ac.id
Penyusun Bab 4 dan 5



Ninda Khoirunnisa, S.T., M.Sc.



NIDN : -
NIPM : 19981005 202408 011 563680
Jabatan Fungsional: Staf Pengajar
Bidang Keahlian : Data Science dan Natural Language Processing
Email : ninda[at]tif.uad.ac.id
Penyusun Bab 9-12



KONTRIBUSI PENULIS

Nomor Bab	Daftar Penulis
Bab I	Ir. Sri Winiarti, S.T., M.Cs.
Bab II	Dewi Soyusiawaty, S.T., M.T
Bab III	Jefree Fahana, S.T., M.Kom.
Bab IV	Dr. Ardiansyah
Bab V	Dr. Ardiansyah
Bab VI	Miftahurrahma Rosyda S.Kom., M.Eng.
Bab VII	Dewi Soyusiawaty, S.T., M.T
Bab VIII	Jefree Fahana, S.T., M.Kom.
Bab IX	Ninda Khoirunnisa, S.T., M.Sc.
Bab X	Ninda Khoirunnisa, S.T., M.Sc.
Bab XI	Ninda Khoirunnisa, S.T., M.Sc.
Bab XII	Ninda Khoirunnisa, S.T., M.Sc.



HALAMAN REVISI

Yang bertanda tangan di bawah ini:

Nama : Dr. Ardiansyah, S.T., M.Cs.

NIPM : 19790723 200309 111 0932301

Jabatan : Dosen Pengampu Mata Kuliah Basis Data

Dengan ini menyatakan pelaksanaan Revisi Petunjuk Praktikum Basisdata untuk Program Studi Informatika telah dilaksanakan dengan penjelasan sebagai berikut:

No	Keterangan Revisi	Tanggal Revisi	Nomor Modul
1	a. Tuliskan revisi disini. b. dst.		PP/018/V/R1
2	a. Penataan ulang struktur praktikum b. Menambahkan materi Trigger dan Keamanan Data c. Menambahkan materi trend basisdata (NO SQL)	10 Juli 2023	PP/018/VII/R2
3.	a. Menyesuaikan CPL dan CPMK berdasarkan RPS 2024 b. Menghapus materi Praktikum 6. Normalisasi c. Menambah materi Praktikum 8. Indexing	11 September 2024	PP/018/IX/R3
4	Penambahan materi NoSQL	29 September 2025	PP/018/IX/R4

Yogyakarta, 29 September 2025
Koordinator Penyusun



Dr. Ardiansyah, S.T., M.Cs.
NIPM. 19790723 200309 111 0932301



HALAMAN PERNYATAAN

Yang bertanda tangan di bawah ini:

Nama : Murein Miksa Mardhia S.T., M.T.
NIPM : 19891019 201606 011 1236278
Jabatan : Kepala Laboratorium S1 Informatika

Menerangkan dengan sesungguhnya bahwa Petunjuk Praktikum ini telah direview dan akan digunakan untuk pelaksanaan praktikum di Semester Gasal Tahun Akademik 2024/2025 di Laboratorium Praktikum S1 Informatika, Program Studi Informatika, Fakultas Teknologi Industri, Universitas Ahmad Dahlan.

Yogyakarta, 29 September 2025

Mengetahui,
Ketua Kelompok Keilmuan



Dr. Ardiansyah, S.T., M.Cs.
NIPM. 197907232003091110932301

Kepala Laboratorium Praktikum
S1 Informatika



Murein Miksa Mardhia S.T., M.T.
NIPM. 19891019 201606 011 1236278



VISI DAN MISI PRODI INFORMATIKA

VISI

Menjadi program studi yang unggul dan inovatif dalam bidang rekayasa perangkat lunak dan sistem cerdas dengan dijiwai nilai-nilai Islam

MISI

1. Mengimplementasikan nilai-nilai AIK pada semua aspek kegiatan.
2. Memajukan ilmu pengetahuan dan teknologi Rekayasa Perangkat Lunak dan Sistem cerdas melalui pendidikan, penelitian, dan pengabdian kepada masyarakat.
3. Mengembangkan kerjasama dalam pendidikan, penelitian, dan pengabdian kepada masyarakat di tingkat lokal, nasional, maupun internasional.
4. Menyelenggarakan tata kelola program studi yang unggul dan inovatif.
5. Berperan aktif dalam kegiatan yang menunjang profesi dosen.

TATA TERTIB LABORATORIUM S1 INFORMATIKA

DOSEN/KOORDINATOR PRAKTIKUM

1. Dosen harus hadir saat praktikum minimal 15 menit di awal kegiatan praktikum untuk mengisi materi dan menandatangani presensi kehadiran praktikum.
2. Dosen membuat modul praktikum, soal seleksi asisten, pre-test, post-test, dan responsi dengan berkoordinasi dengan asisten dan pengampu mata praktikum.
3. Dosen berkoordinasi dengan koordinator asisten praktikum untuk evaluasi praktikum setiap minggu.
4. Dosen menandatangani surat kontrak asisten praktikum dan koordinator asisten praktikum.
5. Dosen yang tidak hadir pada slot praktikum tertentu tanpa pemberitahuan selama 2 minggu berturut-turut mendapat teguran dari Kepala Laboratorium, apabila masih berlanjut 2 minggu berikutnya maka Kepala Laboratorium berhak mengganti koordinator praktikum pada slot tersebut.

PRAKTIKAN

1. Praktikan harus hadir 15 menit sebelum kegiatan praktikum dimulai, dan dispensasi terlambat 15 menit dengan alasan yang jelas (kecuali asisten menentukan lain dan patokan jam adalah jam yang ada di Laboratorium, terlambat lebih dari 15 menit tidak boleh masuk praktikum & dianggap Inhl).
2. Praktikan yang tidak mengikuti praktikum dengan alasan apapun, wajib mengikuti INHAL, maksimal 4 kali praktikum dan jika lebih dari 4 kali maka praktikum dianggap GAGAL.
3. Praktikan yang akan mengikuti inhal diwajibkan mendaftarkan diri dan membayar administrasi inhal kepada laboran inhal paling lambat H-1 jadwal inhal.
4. Praktikan harus berpakaian rapi sesuai dengan ketentuan Universitas, sebagai berikut:
 - a. Tidak boleh memakai Kaos Oblong, termasuk bila ditutupi Jaket/Jas Almamater (Laki-laki / Perempuan) dan Topi harus Dilepas.
 - b. Tidak Boleh memakai Baju ketat, Jilbab Minim dan rambut harus tertutup jilbab secara sempurna, tidak boleh kelihatan di jidat maupun di punggung (khusus Perempuan).
 - c. Tidak boleh memakai baju minim, saat duduk pun pinggang harus tertutup rapat (Laki-laki / Perempuan).
 - d. Laki-laki tidak boleh memakai gelang, anting-anting ataupun aksesoris Perempuan.
5. Praktikan tidak boleh makan dan minum selama kegiatan praktikum berlangsung, harus menjaga kebersihan, keamanan dan ketertiban selama mengikuti kegiatan praktikum atau selama berada di dalam laboratorium (tidak boleh membuang sampah sembarangan baik kertas, potongan kertas, bungkus permen baik di lantai karpet maupun di dalam ruang CPU).
6. Praktikan dilarang meninggalkan kegiatan praktikum tanpa seizin Asisten atau Laboran.
7. Praktikan harus meletakkan sepatu dan tas pada rak/loker yang telah disediakan.
8. Selama praktikum dilarang NGENET/NGE-GAME, kecuali mata praktikum yang membutuhkan atau menggunakan fasilitas Internet.
9. Praktikan dilarang melepas kabel jaringan atau kabel power praktikum tanpa sepengertahan laboran
10. Praktikan harus memiliki FILE Petunjuk praktikum dan digunakan pada saat praktikum dan harus siap sebelum praktikum berlangsung.
11. Praktikan dilarang melakukan kecurangan seperti mencontek atau menyalin pekerjaan praktikan yang lain saat praktikum berlangsung atau post-test yang menjadi tugas praktikum.

12. Praktikan dilarang mengubah *setting software/hardware* komputer baik menambah atau mengurangi tanpa permintaan asisten atau laboran dan melakukan sesuatu yang dapat merugikan laboratorium atau praktikum lain.
13. Asisten, Koordinator Praktikum, Kepala laboratorium dan Laboran mempunyai hak untuk menegur, memperingatkan bahkan meminta praktikan keluar ruang praktikum apabila dirasa anda mengganggu praktikan lain atau tidak melaksanakan kegiatan praktikum sebagaimana mestinya dan atau tidak mematuhi aturan lab yang berlaku.
14. Pelanggaran terhadap salah satu atau lebih dari aturan diatas maka Nilai praktikum pada pertemuan tersebut dianggap 0 (NOL) dengan status INHAL.
15. Peraturan lain mengikuti aturan yang ditetapkan oleh Fakultas pada Surat Keputusan Dekan No. F2/524/A/IX/2018.
16. Sebelum pelaksanaan praktikum membaca surat pendek.
17. Praktikan saat memasuki lab wajib menggunakan kaos kaki.
18. Praktikan dilarang mengganti wallpaper dekstop pada laboratorium yang digunakan.

ASISTEN PRAKTIKUM

1. Asisten harus hadir 15 Menit sebelum praktikum dimulai (konfirmasi ke koordinator bila mengalami keterlambatan atau berhalangan hadir).
2. Asisten yang tidak bisa hadir WAJIB mencari pengganti, dan melaporkan kepada Koordinator Asisten.
3. Asisten harus berpakaian rapi sesuai dengan ketentuan Universitas, sebagai berikut:
 - a. Tidak boleh memakai Kaos Oblong, termasuk bila ditutupi Jaket/Jas Almamater (Laki-laki / Perempuan) dan Topi harus Dilepas.
 - b. Tidak Boleh memakai Baju ketat, Jilbab Minim dan rambut harus tertutup jilbab secara sempurna, tidak boleh kelihatan di jidat maupun di punggung (khusus Perempuan).
 - c. Tidak boleh memakai baju minim, saat duduk pun pinggang harus tertutup rapat (Laki-laki / Perempuan).
 - d. Laki-laki tidak boleh memakai gelang, anting-anting ataupun aksesoris Perempuan.
4. Asisten harus menjaga kebersihan, keamanan dan ketertiban selama mengikuti kegiatan praktikum atau selama berada di laboratorium, menegur atau mengingatkan jika ada praktikan yang tidak dapat menjaga kebersihan, ketertiban atau kesopanan.
5. Asisten harus dapat merapikan dan mengamankan presensi praktikum, Kartu Nilai serta tertib dalam memasukan/Input nilai secara Online/Offline.
6. Asisten mencatat dan merekap praktikan dengan status INHAL setiap minggu serta wajib mengumumkan mekanisme INHAL di awal pertemuan praktikum.
7. Asisten harus dapat bertindak secara profesional sebagai seorang asisten praktikum dan dapat menjadi teladan bagi praktikan.
8. Asisten harus dapat memberikan penjelasan/pemahaman yang dibutuhkan oleh praktikan berkenaan dengan materi praktikum yang diasistensi sehingga praktikan dapat melaksanakan dan mengerjakan tugas praktikum dengan baik dan jelas.
9. Asisten tidak diperkenankan mengobrol sendiri apalagi sampai membuat gaduh.
10. Asisten dimohon mengkoordinasikan untuk meminta praktikan agar mematikan komputer untuk jadwal terakhir dan sudah dilakukan penilaian terhadap hasil kerja praktikan.
11. Asisten wajib untuk mematikan LCD Projector dan komputer asisten/praktikan apabila tidak digunakan.
12. Asisten tidak diperkenankan menggunakan akses internet selain untuk kegiatan praktikum, seperti Youtube/Game/Medsos/Streaming Film di komputer praktikan.



13. Peraturan lain mengikuti aturan yang ditetapkan oleh Fakultas pada Surat Keputusan Dekan No. F2/524/A/IX/2018.
14. Sebelum pelaksanaan praktikum membaca surat pendek.
15. Asisten saat memasuki lab wajib menggunakan kaos kaki.

LAIN-LAIN

1. Pada Saat Responsi Harus menggunakan Baju Kemeja untuk Laki-laki dan Perempuan untuk Praktikan dan Asisten.
2. Ketidakhadiran praktikum dengan alasan apapun dianggap INHAL.
3. Pelaksanaan (waktu dan metode) INHAL sama seperti praktikum mingguan/reguler.
4. Izin praktikum mengikuti aturan izin SIMERU/KULIAH.
5. Yang tidak berkepentingan dengan praktikum dilarang mengganggu praktikan atau membuat keributan/kegaduhan.
6. Penggunaan lab diluar jam praktikum maksimal sampai pukul 18.00 dengan menunjukkan surat ijin dari Kepala Laboratorium Prodi Informatika.

Yogyakarta, 29 September 2025

Kepala Laboratorium Praktikum
S1 Informatika



Murein Miksa Mardhia S.T., M.T.
NIPM. 19891019 201606 011 1236278



DAFTAR ISI

HAK CIPTA	1
KATA PENGANTAR	2
DAFTAR PENYUSUN	3
KONTRIBUSI PENULIS	4
HALAMAN REVISI	5
HALAMAN PERNYATAAN	6
VISI DAN MISI PRODI INFORMATIKA	7
TATA TERTIB LABORATORIUM S1 INFORMATIKA	8
DAFTAR ISI	11
DAFTAR GAMBAR	12
DAFTAR TABEL	13
SKENARIO PRAKTIKUM SECARA DARING	14
PRAKTIKUM 1: IDENTIFIKASI ENTITAS UNTUK STUDI PROYEK	13
PRAKTIKUM 2: KONVERSI MODEL DATA KE TABEL	27
PRAKTIKUM 3: NORMALISASI	35
PRAKTIKUM 4: DATA DEFINITION LANGUAGE (DDL)	44
PRAKTIKUM 5: INDEKS	52
PRAKTIKUM 6: DATA MANIPULATION LANGUAGE (DML) DAN FUNGSI AGREGASI	63
PRAKTIKUM 7: RELASI TABEL DENGAN JOIN	81
PRAKTIKUM 8: TRIGGER, KEAMANAN DATA, DAN STORED PROCEDURE	92
PRAKTIKUM 9: OPERASI CREATE, READ, UPDATE, DELETE (CRUD) DALAM NOSQL	106
PRAKTIKUM 10: NOSQL – INDEKS DAN AGREGASI	128
PRAKTIKUM 11: NOSQL – PEMROGRAMAN DENGAN PYTHON	148
PRAKTIKUM 12: TREND DBMS	160
DAFTAR PUSTAKA	176

DAFTAR GAMBAR

Gambar 1.1 Label Gambar.

14

DAFTAR TABEL



SKENARIO PRAKTIKUM SECARA DARING

Nama Mata Praktikum :

Jumlah Pertemuan :

TABEL SKENARIO PRAKTIKUM DARING

Pertemuan ke	Judul Materi	Waktu <i>(Lama praktikum sampai pengumpulan posttest)</i>	Skenario Praktikum <i>(Dari pemberian pre-test, post-test dan pengumpulannya serta mencantumkan metode yang digunakan misal video, whatsapp group, Google meet atau lainnya)</i>
1			
2			
3	dst	dst	Dst



PRAKTIKUM 1: IDENTIFIKASI ENTITAS UNTUK STUDI PROYEK

Pertemuan ke : 1

Total Alokasi Waktu : 180 menit

- Materi : 30 menit
- Pre-Test : 30 menit
- Praktikum : 60 menit
- Post-Test : 60 menit

Total Bobot Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 30 %
- Post-Test : 50 %

Pemenuhan CPL dan CPMK:

CPL 09-KK02	Menguasai konsep teoritis bidang pengetahuan Ilmu Komputer/Informatika secara umum dan konsep teoritis bagian khusus dalam bidang pengetahuan tersebut secara mendalam, serta mampu memformulasikan penyelesaian masalah prosedural.
CPMK 3.3.2	Mahasiswa dapat mengimplementasikan pemodelan data

1.1. DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu mengidentifikasi dan mengimplementasikan entitas untuk studi proyek yang dikerjakan.

1.2. INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:

CPL 09-KK02	CPMK 3.3.2	Mahasiswa mampu menerapkan pemodelan data sesuai arsitektur organisasi
CPL 09-KK02	Sub CPMK 3.3.2.1	Mahasiswa mampu menerapkan pemodelan data dengan ERD

1.3. TEORI PENDUKUNG

1. Tabel dan Atribut

Identifikasi entitas merupakan tahap awal dalam perancangan basis data yang berfokus pada penentuan objek-objek utama (entity) yang akan dimodelkan. Entitas adalah segala sesuatu yang dapat diidentifikasi secara unik dalam lingkup sistem informasi, misalnya orang, benda, tempat, atau peristiwa. Tahap ini penting agar sistem informasi yang dirancang sesuai dengan kebutuhan nyata di lapangan.

Entitas merupakan individu atau objek yang memiliki sesuatu yang nyata (eksistensinya) dan dapat dibedakan dari sesuatu yang lain. Entitas biasanya dalam bentuk kata benda (Oracle Academy, 2019). Entitas dapat diklasifikasikan menjadi tiga tipe, yaitu

Tabel 1.1 Tipe Entitas

Nama Tipe	Penjelasan	Contoh
Prime	Entitas independen	Customer, Mahasiswa
Characteristic	Entitas yang muncul karena berelasi dengan entitas (<i>prime</i>) lainnya.	Pemesanan (order), Jadwal
Intersection	Entitas yang muncul karena ada dua atau lebih entitas	Item Pemesanan (order item)

Sedangkan untuk atribut adalah karakteristik atau ciri yang menjelaskan entitas sehingga dapat membedakan antara entitas satu dengan entitas lainnya (Oracle Academy, 2019). Dalam sebuah entitas pasti memiliki satu atau lebih atribut untuk mendefinisikan karakteristik dari entitas tersebut. Contoh atribut dari Entitas Mahasiswa yaitu NIM, Nama, JK.

2. Mengenal MySQL

MySQL adalah program *database server* yang mampu menerima dan mengirimkan datanya sangat cepat, multi user serta menggunakan perintah dasar SQL (Structured Query Language) (<http://mysql.com>). MySQL memiliki dua bentuk lisensi, yaitu Free Software dan Shareware. MySQL yang biasa kita gunakan adalah MySQL Free Software yang berada dibawah Lisensi GNU/GPL (General Public License). MySQL merupakan sebuah database server yang free, artinya kita bebas menggunakan database ini untuk keperluan pribadi atau usaha tanpa harus membeli atau membayar lisensinya.

MySQL pertama kali dirintis oleh seorang programmer database bernama Michael Widenius. Selain database server, MySQL juga merupakan program yang dapat mengakses suatu database MySQL yang berposisi sebagai Server, yang berarti program kita berposisi sebagai Client. Jadi MySQL adalah sebuah database yang dapat digunakan sebagai Client maupun server. Database MySQL merupakan suatu perangkat lunak database yang berbentuk database relasional atau disebut Relational Database Management System (RDBMS) yang menggunakan suatu bahasa permintaan yang bernama SQL (Structured Query Language).

Database MySQL memiliki beberapa kelebihan dibanding database lain, antara lain :

- 1) MySQL merupakan Database Management System (DBMS).
- 2) MySQL sebagai Relational Database Management System (RDBMS) atau disebut dengan database Relational.
- 3) MySQL Merupakan sebuah database server yang free, artinya kita bebas menggunakan database ini untuk keperluan pribadi atau usaha tanpa harus membeli atau membayar lisensinya.
- 4) MySQL merupakan sebuah database client.
- 5) MySQL mampu menerima query yang bertumpuk dalam satu permintaan atau Multi Threading.
- 6) MySQL merupakan Database yang mampu menyimpan data berkapasitas sangat besar hingga berukuran GigaByte sekalipun.
- 7) MySQL didukung oleh driver ODBC, artinya database MySQL dapat diakses menggunakan aplikasi apa saja termasuk berupa visual seperti Visual Basic, PHP dan Delphi.
- 8) MySQL merupakan Database Server yang multi user, artinya database ini tidak hanya digunakan oleh satu pihak orang akan tetapi dapat digunakan oleh banyak pengguna.
- 9) MySQL mendukung field yang dijadikan sebagai kunci primer dan kunci unik (Unique).



- 10) MySQL memiliki kecepatan dalam pembuatan tabel maupun update table.

3. Pemahaman Kasus pada Sistem Informasi Pengiriman Paket di PT. ABC

Tema Proyek: Sistem Informasi Pengiriman Paket.

Dari deskripsi berikut, identifikasi entitas yang relevan:

Sebuah perusahaan jasa ekspedisi ABC mengelola proses pengiriman paket. Setiap pelanggan dapat mengirim paket dengan detail pengirim dan penerima. Paket memiliki data berupa jenis barang, berat, dimensi, serta ongkos kirim yang dihitung berdasarkan jarak dan berat. Setiap paket dicatat status pengirimannya (misalnya: *diproses*, *dalam perjalanan*, *terkirim*). Kurir bertugas mengantarkan paket kepada penerima. Selain itu, perusahaan juga mencatat data kendaraan yang digunakan kurir dalam proses pengiriman.

- . Data yang disimpan dalam basis data meliputi :

Tabel 1.2 Data disimpan pada Basis Data

Data Pengirim, terdiri dari: ID_Pengirim, Nama, Alamat, Telepon/WA
Data penerima, terdiri dari: ID_Penerima, Nama, Alamat, Telepon/WA
Data paket terdiri dari: ID_Paket, Jenis_Barang, Berat, Ongkos_Kirim, Status
Data kurir terdiri dari: ID_Kurir, Nama, No_HP
Data Kendaraan terdiri dari: ID_Kendaraan, Plat_Nomor, Jenis
Data pengiriman terdiri dari: ID_Pengiriman, Tanggal, Waktu

1.4. HARDWARE DAN SOFTWARE

Hardware dan software yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. XAMPP.
3. Browser (mozilla firefox, chrome, dan lainnya).

1.5. PRE-TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL 09-KK02	CPMK 3.3.2	Tentukan jenis entitas Kuat dan entitas lemah yang terdapat dalam studi kasus yang diberikan	25
2.	CPL 09-KK02	CPMK 3.3.2	Tentukan atribut yang terlibat dalam studi kasus yang diberikan	25
3.	CPL 09-KK02	CPMK 3.3.2	Tentukan relasi yang terlibat dalam studi kasus yang diberikan	25

4.	CPL 09-KK02	CPMK 3.3.2	Mengapa entitas <i>Pengiriman</i> dapat dikategorikan sebagai entitas lemah?	25
----	-------------	------------	--	----

1.6. LANGKAH PRAKTIKUM

Aturan Penilaian (Total Skor: 100):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL 09-KK02	CPMK 3.3.2	Buat database	Hasil praktikum	20
2.	CPL 09-KK02	CPMK 3.3.2	Buat tabel dilengkapi dengan atribut, tipe data, primary key	Hasil praktikum	80

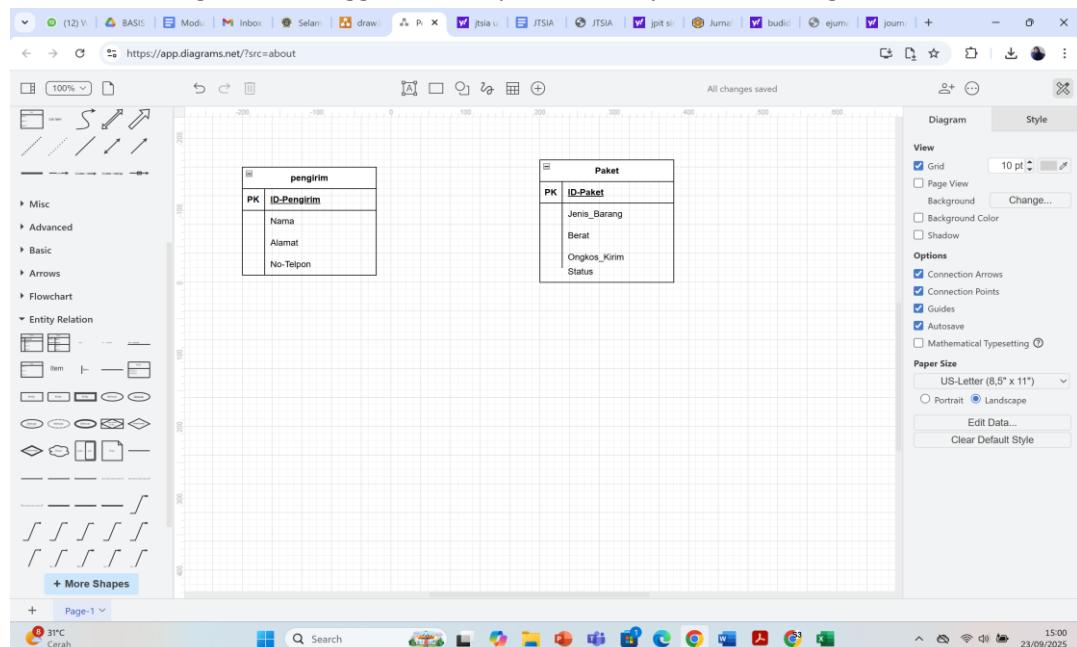
Langkah-Langkah Praktikum:

1. Lakukan Identifikasi Atribut, entitas Kuat dan Entitas lemah dari Studi kasus Sistem Informasi pengiriman paket. Isikan hasil identifikasi dalam Tabel 1.1.

Tabel 1.1. Hasil Identifikasi Entitas Studi Proyek Sistem Informasi Paket

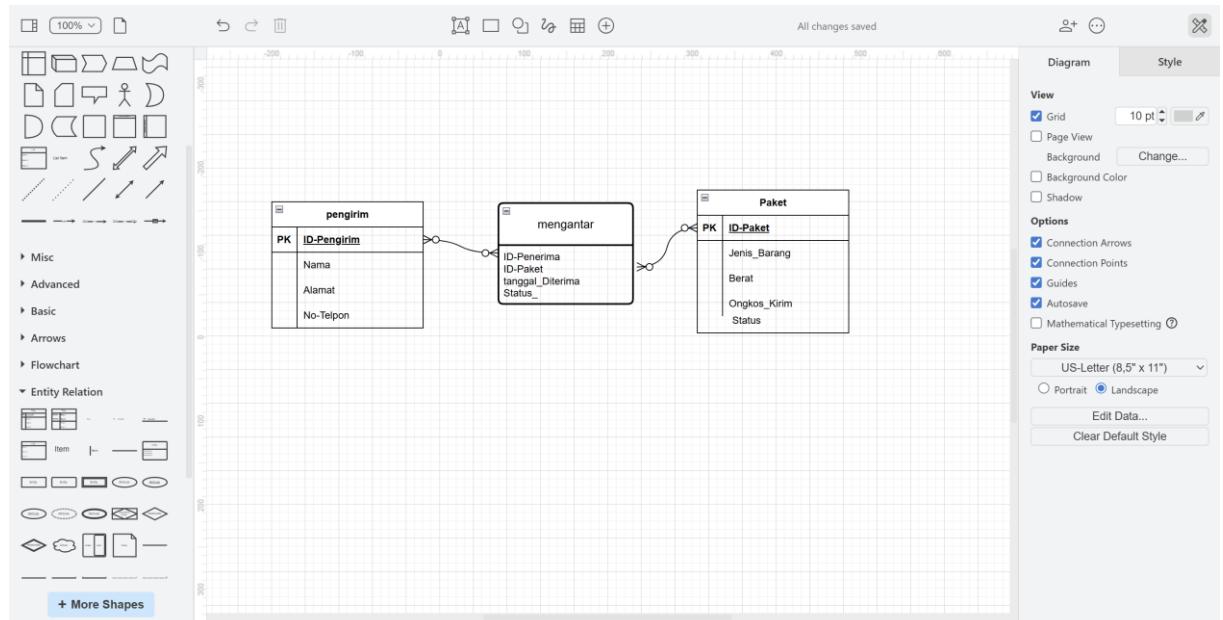
No	Nama Entitas	Atribut Utama	Jenis Entitas	
			Kuat	Lemah
2				
3				
4				
5				
6				

2. Buatlah pemodelan data menggunakan Microsoft Visio/Drawing IO dan Powerdesigner dengan studi kasus “Pengirim mengantar-paket ke penerima”. Gunakanlah tools Drawing io/Visio Drawing untuk menggambarkan pemodelannya. Perhatikan gambar 1.1.



Gambar 1.1 Tampilan Awal Drawing IO

Seperti yang telah dipraktikkan sebelumnya, buat entitas dari studi kasus di atas. Disini entitas awal yang terbentuk adalah entitas Pengirim dan paket.

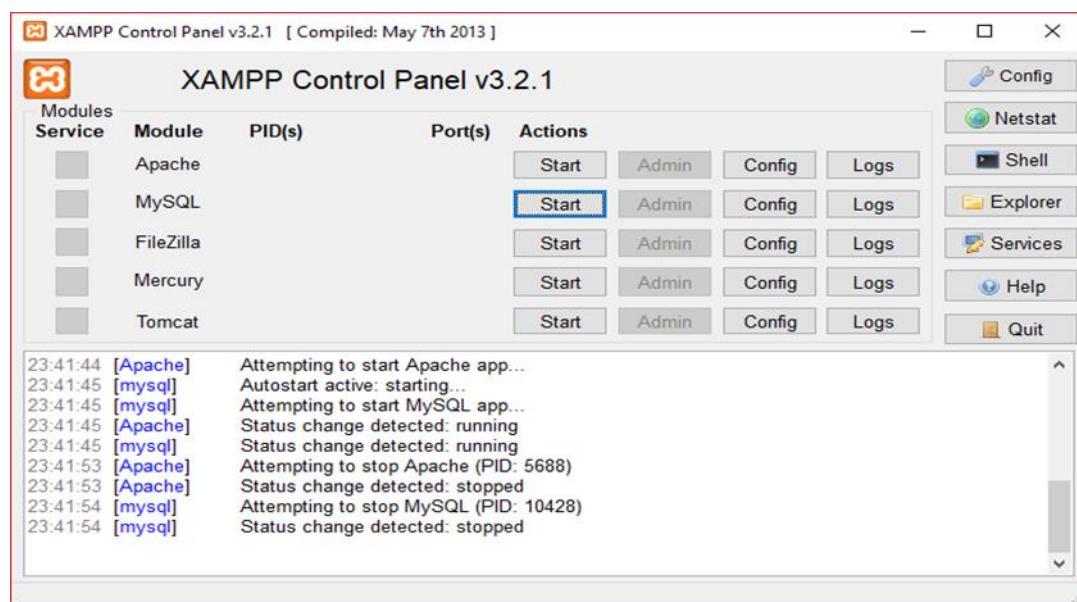


Gambar 1.2 Membuat Entitas

Karena relasi yang terbentuk adalah “many to many” maka akan dibuat 1 entitas baru yaitu entitas “mengantar” yang berisi primary key dari masing – masing entitas, primary key tersebut dijadikan sebagai foreign key pada entitas baru untuk menghubungkan kedua entitas yang telah dibuat. Sehingga hasilnya seperti yang ditunjukkan pada Gambar 1.2. Hubungkan kedua entitas yang telah dibuat dengan menggunakan shape “Relationship”. Hasil dari Gambar 1.2 s adalah hasil dari pemodelan data dari studi kasus yang ada.

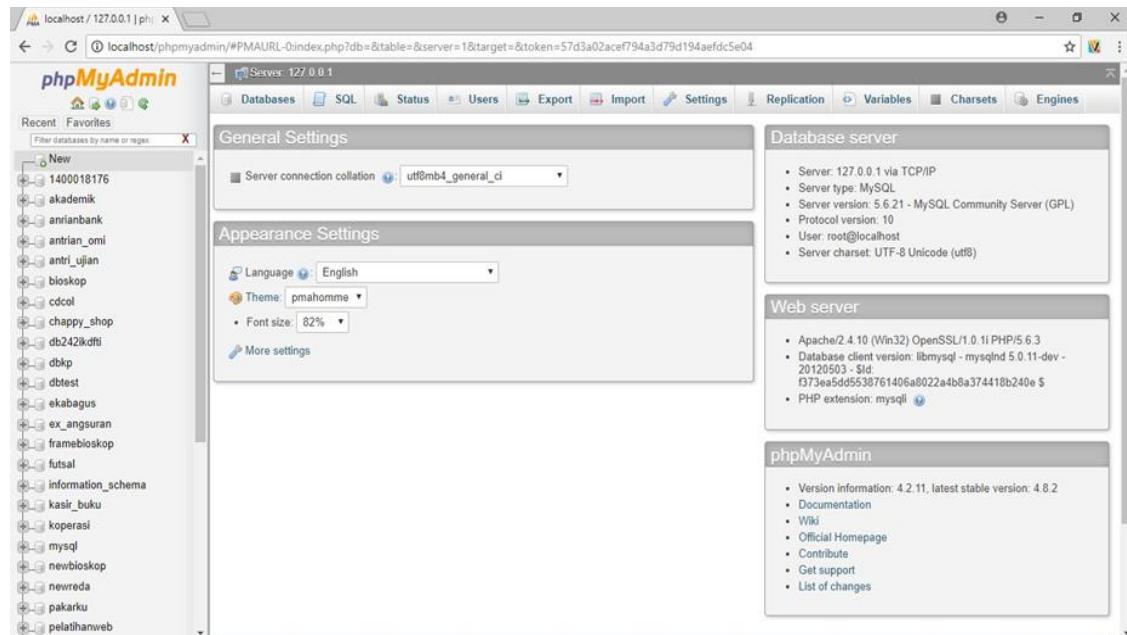
3. Mengakses PHPMyAdmin dengan XAMPP

Jalankan XAMPP Control Panel hingga muncul jendela aplikasi XAMPP Control Panel. Kemudian klik tombol Start pada modul Apache dan MySQL. Modul Apache digunakan untuk mengakses PHPMyAdmin pada browser sehingga lebih mudah dalam mengakses MySQL karena menggunakan GUI (Graphical User Interface). Modul MySQL digunakan untuk melayani request atau query yang diterima dari PHPMyAdmin.



Gambar 1. 3 Tampilan XAMPP

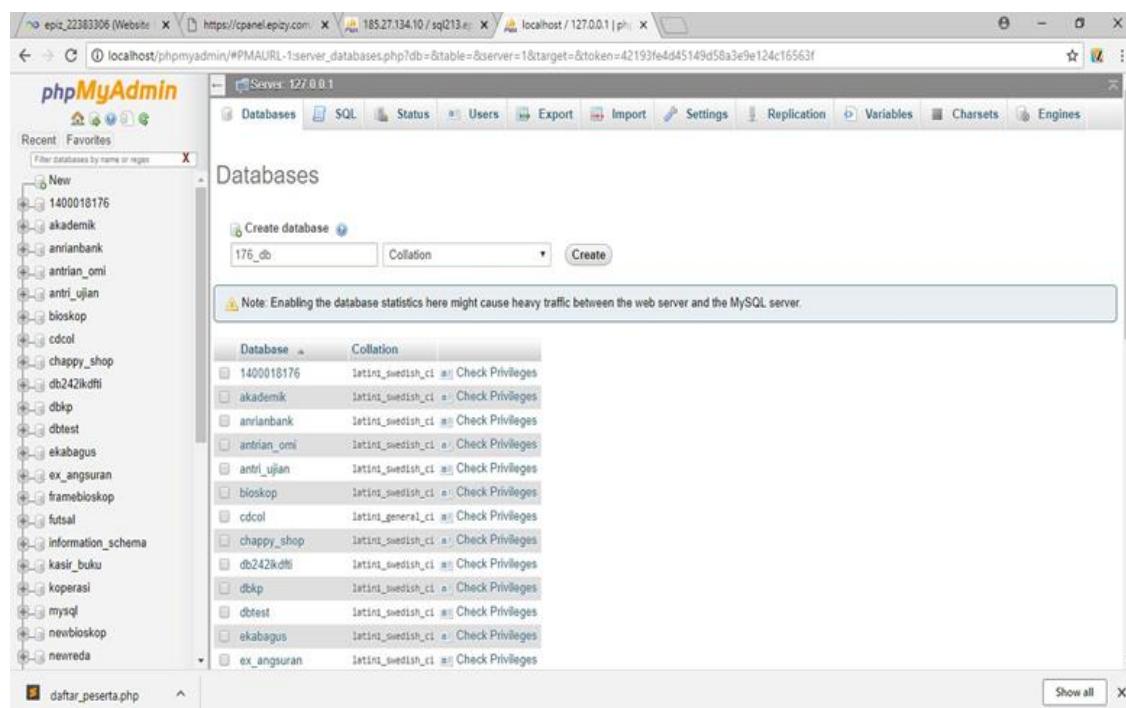
Kemudian membuka browser (Chrome, Mozilla, Opera, dll) dan mengaksesnya dengan mengetikkan “localhost/phpmyadmin” pada kolom isian URL, sehingga muncul seperti pada gambar di bawah ini.



Gambar 1. 4 Tampilan PHPMyAdmin

Membuat Database

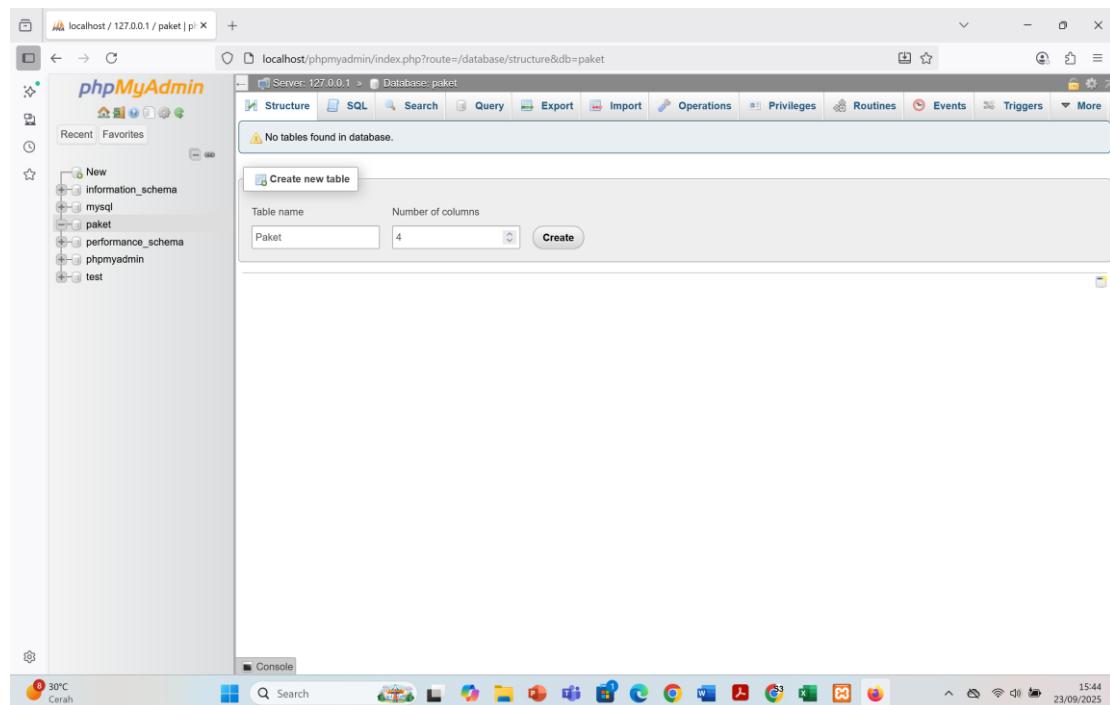
Pilih menu “new” kemudian isi nama database pada kolom yang sudah disediakan, untuk keseragaman nama database diisi dengan “3_digit_nim_terakhir_db” kemudian pilih “create”.



Gambar 1. 5 Create DB

Membuat Tabel

Membuat tabel dengan nama tabel “pengirim” yang memiliki atribut sesuai dengan atribut yang telah diidentifikasi pada Tabel 1. Langkahnya yaitu pilih menu database yang hingga muncul tampilan seperti Gambar 1.6.



Gambar 1. 6 Create Table

Kemudian mengisi form untuk memberi atribut pada tabel “paket” yang sudah dibuat. Pada entitas ini, atribut nim akan digunakan sebagai pembeda pada data yang akan dimasukkan ke dalam tabel atau sering disebut primary key.



Atribut sesuaikan dengan tabel 1 pada entitas paket dengan tipe VARCHAR dengan panjang karakter yang disesuaikan dengan keinginan programmer. Adapun atributnya dari Entitas Paket sebagai berikut: ID_Paket, Jenis_Barang, Berat, Ongkos_Kirim, dan Status. Lakukan desain Entitas dan atribut untuk entitas lainnya dan lakukan relasi tabelnya

Name	Type	Length/Values	Default	Collation	Attributes	Null	Index
ID_Paket	VARCHAR	10	None			✓	PRIMARY
Jenis_Barang	VARCHAR	50	None			✓	
Berat	FLOAT	10	None			✓	
Ongkos_Kirim	VARCHAR	12	None			✓	
Status	VARCHAR		None			✓	

Gambar 1. 7 Isi Data Table

Kemudian pada atribut ID_Paket, karena sebagai primary key maka pada index diisi dengan PRIMARY kemudian pilih “Save”.

Name	Type	Length/Values	Default	Collation	Attributes	Null	Index
ID_Paket	VARCHAR	10	None			✓	PRIMARY
Jenis_Barang	VARCHAR	50	None			✓	
Berat	FLOAT	10	None			✓	
Ongkos_Kirim	VARCHAR	12	None			✓	
Status	VARCHAR		None			✓	

Gambar 1. 8 Primary Key



1.7. POST TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

Lakukanlah identifikasi pada kasus proyek system informasi berikut ini:

Studi kasus proyek: Sistem Informasi Klinik Kesehatan

Sebuah klinik kesehatan mengelola layanan untuk pasien rawat jalan. Pasien melakukan pendaftaran, kemudian diperiksa oleh dokter, dan mendapatkan resep obat. Klinik juga mencatat data obat, stok, serta transaksi pembayaran. Dokter memiliki jadwal praktik yang harus dikelola agar pasien bisa melakukan reservasi dengan lebih teratur. Data yang bisa disimpan adalah data Pasien, Dokter, Pendaftaran, Resep, Obat, Pembayaran, Jadwal.

No	CPL	CPMK	Pertanyaan	Skor
1	CPL 09-KK02	CPMK 3.3.2	Lakukan identifikasi jenis entitas dari kasus proyek tersebut sertakan atribut dari masing-masing entitas.	15
2	CPL 09-KK02	CPMK 3.3.2	Kelompokkan entitas menjadi entitas kuat dan entitas lemah (jika ada).	15
3	CPL 09-KK02	CPMK 3.3.2	Nyatakan model data dari kasus yang diberikan untuk	20
4	CPL 09-KK02	CPMK 3.3.2	Implementasikan entitas dan atribut yang sudah ditentukan ke dalam Phpmyadminnya dengan ketentuan isian datanya	50
			Total	100

1.8. HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL 09-KK02	CPMK 3.3.2	20%		
2.	Praktik	CPL 09-KK02	CPMK 3.3.2	30%		
3.	Post-Test	CPL 09-KK02	CPMK 3.3.2	50%		
Total Nilai						



LEMBAR JAWABAN PRE-TEST DAN POST-TEST PRAKTIKUM

Nama : NIM :	Asisten: Paraf Asisten:	Tanggal: Nilai:
-----------------	----------------------------	--------------------

--

PRAKTIKUM 2: KONVERSI MODEL DATA KE TABEL

Pertemuan ke : 2

Total Alokasi Waktu : 180 menit

- Materi : 30 menit
- Pre-Test : 30 menit
- Praktikum : 60 menit
- Post-Test : 60 menit

Total Bobot Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 30 %
- Post-Test : 50 %

Pemenuhan CPL dan CPMK:

CPL 09-KK02	Mampu menerapkan konsep teoritis bidang area Informatika terkait matematika dasar dan ilmu komputer untuk memodelkan masalah dan meningkatkan produktivitas
CPMK 3.3.2	Mahasiswa dapat mengimplementasikan pemodelan data dan mengkonversikan ke tabel

2.1. DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu:

1. Menerapkan konsep teoritis bidang area Informatika terkait matematika dasar dan ilmu komputer untuk memodelkan masalah dan meningkatkan produktivitas dengan menggunakan asesmen Pre Test untuk materi terkait Konversi model data ke Tabel
2. Berpikir logis, kritis, sistematis dan inovatif, dan mampu mengambil keputusan secara tepat di bidang keahliannya melalui asesmen Pre test.
3. Mengkaji/menganalisis implikasi pengembangan atau implementasi ilmu pengetahuan teknologi, menyusun deskripsi saintifik hasil kajian untuk pemecahan masalah terkait Konversi Model data ke Tabel dengan asesmen praktik dan post test
4. Memilih, membuat dan menerapkan teknik, sumber daya, penggunaan perangkat teknik modern dan implementasi teknologi informasi untuk memecahkan masalah terkait konversi model data ke Tabel dengan Asesmen Post Test dan Praktik.

2.2. INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:

CPL 09- KK02	CPMK 3.3.2	Kemampuan Mahasiswa dalam mengimplementasikan pemodelan data dan mengkonversikannya ke tabel
--------------------	---------------	--

2.3. TEORI PENDUKUNG

Di dalam basis data yang menjadi pusat perhatian dan intisari sistem adalah tabel dan relasinya. Istilah tabel ini muncul dari abstraksi data pada level fisik. Tabel ini sama artinya dengan entitas dari model data pada level konseptual. Setiap orang bisa membuat tabel tetapi membuat tabel yang baik tidak semua orang dapat melakukannya. Kebutuhan akan membuat tabel yang baik ini melahirkan beberapa teori atau metode antara lain adalah pemetaan ER to tabel dan Normalisasi.

Uraian materi di bawah ini menjelaskan pemetaan ER model ke relasi tabel sedangkan Algoritma atau Langkah-langkah yang dilakukan untuk memetakan ER diagram ke tabel relasional yaitu sebagai berikut:

- 1) Untuk setiap entitas kuat, buat tabel baru Entitas Kuat yang menyertakan seluruh simple atribut dan simple atribut dari composite atribut yang ada. Pilih salah satu atribut kunci sebagai primary key.
- 2) Untuk setiap entitas lemah Entitas Lemah, buat tabel baru Entitas Lemah dengan mengikutsertakan seluruh simple atribut. Tambahkan primary key dari entitas kuatnya (owner entity type) yang akan digunakan sebagai primary key bersama-sama partial key dari entitas lemah.
- 3) Untuk setiap multivalued atribut R, buatlah tabel baru R yang menyertakan atribut dari multivalue tersebut. Tambahkan primary key dari relasi yang memiliki multivalue tersebut. Kedua atribut tersebut membentuk primary key dari tabel R.
- 4) Untuk setiap relasi binary 1:1, tambahkan primary key dari sisi yang lebih “ringan” ke sisi (entitas) yang lebih “berat”. Suatu sisi dianggap lebih “berat” timbangannya apabila mempunyai partisipasi total. Tambahkan juga simple atribut yang terdapat pada relasi tersebut ke sisi yang lebih “berat”. Apabila kedua partisipasi adalah sama-sama total atau sama-sama partial, maka dua entitas tersebut boleh digabung menjadi satu tabel.
- 5) Untuk setiap relasi binary 1:N yang tidak melibatkan entitas lemah, tentukan mana sisi yang lebih “berat” (sisi N). Tambahkan primary key dari sisi yang “ringan” ke tabel sisi yang lebih “berat”. Tambahkan juga seluruh simple atribut yang terdapat pada relasi biner tersebut.
- 6) Untuk setiap relasi binary M:N, buatlah tabel baru R dengan atribut seluruh simple atribut yang terdapat pada relasi biner tersebut. Tambahkan primary key yang terdapat pada kedua sisi ke tabel R. Kedua foreign key yang didapat dari kedua sisi tersebut digabung menjadi satu membentuk primary key dari tabel R.
- 7) Untuk setiap relasi lebih dari dua entitas, n-nary (ternary), meliputi dua alternatif yaitu:
 - a. Buatlah tabel R yang menyertakan seluruh primary key dari entitas yang ikut serta. Sejumlah n foreign key tersebut akan membentuk primary key untuk tabel R. Tambahkan seluruh simple atribut yang terdapat pada relasi n-ary tersebut
 - b. Mengubah bentuk relasi ternary menjadi entitas lemah, kemudian memperbaiki relasi yang terjadi antara entitas lemah tersebut dengan entitas-entitas kuatnya dan melakukan algoritma pemetaan sesuai dengan aturan mapping.

2.4. HARDWARE DAN SOFTWARE

Hardware dan software yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. Buku Petunjuk Praktikum
3. Aplikasi pemodelan: visio drawing, edraw atau Draw io

2.5. PRE-TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL 09-KK02	CPMK 3.3.2	1. Menurut Anda, mengapa penting memahami proses konversi ERD ke tabel sebelum melakukan implementasi basis data di MySQL?	25
2.	CPL 09-KK02	CPMK 3.3.2	2. Diketahui entitas Supplier (IDSupplier, NamaSupplier, Alamat, NoHP) dan Barang (KodeBarang, NamaBarang, Harga, Stok) dengan relasi 1:N "Menyuplai". Buat rancangan tabel fisik 3. Sebuah sistem kepegawaian memiliki entitas Pegawai (NIP, Nama, Jabatan, Gaji) dan entitas Proyek (IDProyek, NamaProyek, Lokasi). Hubungan antara keduanya adalah M:N "Ditugaskan", dengan atribut tambahan Peran dan DurasiHari. Konversikan ERD tersebut menjadi tabel fisik 4. Pada sistem penjualan, terdapat entitas Pelanggan (IDPelanggan, Nama, Alamat, Email) dan Pesanan (NoPesanan, TanggalPesanan, TotalHarga). Hubungan Pelanggan–Pesanan adalah 1:N. Konversikan ERD tersebut ke dalam tabel fisik, lengkap dengan penentuan primary key dan foreign key.	25 25 25
				Total Nilai 100

2.6. LANGKAH PRAKTIKUM

Bacalah Skenario kasus A dan Skenario Kasus B berikut ini, lalu selesaikanlah dengan langkah-langkah praktikum untuk penyelesaian masalah dalam Relasi model data, sehingga Anda bisa menjawab pertanyaan-pertanyaan terkait kasus A dan B.

Skenario Bisnis Kasus A

Dalam sebuah Perusahaan Arsitek yang baru berkembang, memiliki 5 DEPARTEMEN. Setiap DEPARTEMEN memiliki satu atau lebih KARYAWAN. Setiap KARYAWAN mengerjakan satu atau lebih PROYEK. Karyawan memperoleh BONUS jika dapat menyelesaikan sesuai JADWAL. Setiap Karyawan memiliki satu atau lebih JADWAL dalam PROYEK.

Dari Skenario Kasus A akan membuat Konversi dari Relasi model data menjadi Tabel, dengan cara sebagai berikut:

1. Mendefinisikan entitas Kuat dan Entitas Lemah

Pada Skenario Kasus A terdapat Entitas Kuat: DEPARTEMEN, KARYAWAN dan PROYEK, sedangkan Entitas lemah adalah BONUS karena tidak semua karyawan mendapatkan Bonus.



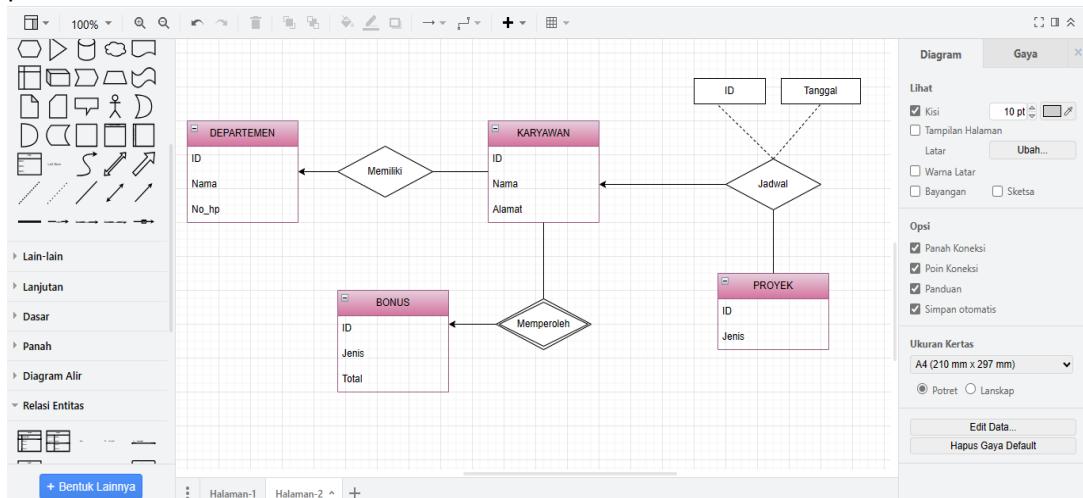
2. Menentukan Atribut, atribut kunci primer dan atribut Kunci tamu.

Entitas	Atribut	Keterangan
DEPARTEMEN	ID, Nama	PK = ID
KARYAWAN	ID, Nama, Alamat, email, No-HP	PK = ID
PROYEK	ID, Nama-Proyek	PK = ID
JADWAL	Tanggal, ID Proyek, ID Karyawan	FK = ID Proyek, ID Karyawan
BONUS	ID, Jenis, Total	PK = ID

3. Mendefinisikan tipe relasi yang terjadi antar entitas beserta Kardinalitasnya

Relasi	Atribut	Kardinalitas	Keterangan
Memiliki		one to many	FK = ID Departemen, ID Karyawan
Mengerjakan		One to many	PK = ID Karyawan, ID Proyek
Memperoleh		One to many	FK = ID Proyek, ID Karyawan
jadwal	Tanggal	One to one	FK = ID Proyek, ID Karyawan

4. Membuat desain Model Data dengan menggunakan aplikasi Drawing io. Hasilnya ditunjukkan pada Gambar 4.1.



Gambar 2.1 Relasi Model Data

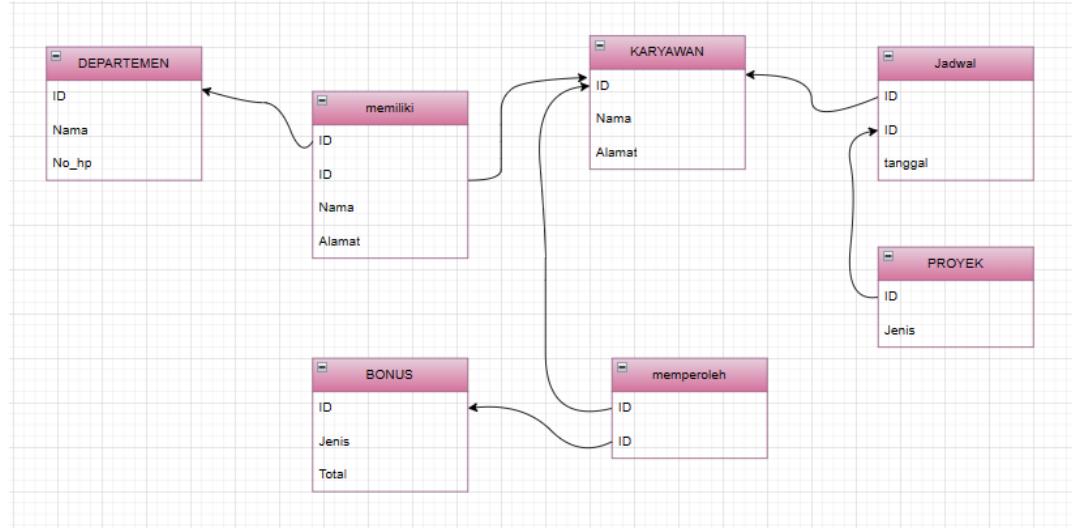
5. Membuat Struktur Fisik Tabel

Selanjutnya membuat table dengan mengacu pada Gambar 5.1 maka diidentifikasi ada 7 tabel, yaitu:

- Tabel Departemen
- Tabel proyek
- Tabel Karyawan

- d. Tabel Bonus
 - e. Tabel Jadwal
 - f. Tabel memiliki
 - g. Tabel Memperoleh
6. Membuat Pemetaan Tabel/ Relasi Tabel.

Caranya dengan mengacu hasil Gambar 4.1 dapat dibuat Pemetaan Tabelnya seperti yang disajikan pada Gambar 4.2.



Gambar 4. 2 Hasil Konversi Relasi Model Data Tabel.Mapping Table

Skenario Bisnis Kasus B

Dalam Suatu perusahaan XYZ, dilakukan pelacakan karyawan dan manajer mereka. Setiap karyawan memiliki satu manajer, termasuk Direktur Utama yang mengelola dirinya sendiri. Setiap manajer dapat mengelola beberapa karyawan. Seorang Manajer hanya memimpin satu Departemen, dan setiap karyawan menyelesaikan satu atau banyak jenis pekerjaan.

Dalam suatu perusahaan XYZ, dilakukan pelacakan karyawan dan manajer mereka.

- Setiap karyawan memiliki satu manajer, termasuk Direktur Utama yang mengelola dirinya sendiri.
- Setiap manajer dapat mengelola beberapa karyawan.
- Seorang manajer hanya memimpin satu departemen.
- Setiap karyawan dapat menyelesaikan satu atau banyak jenis pekerjaan.

1. Analisis ERD

a. Tentukan entitas utama

- Karyawan (IDKaryawan, Nama, Alamat, Jabatan, IDMajer)
- Departemen (IDDept, NamaDept)
- Pekerjaan (IDPekerjaan, NamaPekerjaan, Deskripsi)

b. Hubungan antar entitas

- Karyawan – Karyawan (self relationship 1:N → manajer dan bawahan)
- Manajer – Departemen (1:1)
- Karyawan – Pekerjaan (M:N, karena 1 karyawan bisa banyak pekerjaan, 1 pekerjaan bisa dikerjakan banyak karyawan → butuh tabel penghubung Karyawan_Pekerjaan)

2. Konversi ERD ke Basis Data Fisik (Tabel)

Buat rancangan tabel berikut:

1. Karyawan

- IDKaryawan (PK)
- Nama
- Alamat
- Jabatan
- IDManajer (FK → Karyawan.IDKaryawan)

2. Departemen

- IDDept (PK)
- NamaDept
- IDManajer (FK → Karyawan.IDKaryawan, dengan constraint unique agar 1 manajer hanya pimpin 1 dept)

3. Pekerjaan

- IDPekerjaan (PK)
- NamaPekerjaan
- Deskripsi

4. Karyawan_Pekerjaan (tabel penghubung untuk relasi M:N)

- IDKaryawan (FK → Karyawan.IDKaryawan)
- IDPekerjaan (FK → Pekerjaan.IDPekerjaan)
- (Primary Key gabungan: IDKaryawan + IDPekerjaan)

2.7. POST TEST

Pelajari Skenario Kasus C dan Skenario Kasus D, kemudian lakukan langkah praktikum 1 sampai 5 untuk menjawab pertanyaan yang diminta.

Skenario Kasus C

Sebuah Apotek menjual berbagai jenis obat. Pelanggan yang datang dapat membeli obat lebih dari satu dan dapat bertransaksi lebih dari satu kali setiap harinya. Setiap Obat ditempatkan pada rak yang berbeda. Dalam Apotek tersebut Apoteker menerima resep satu atau lebih dari Pelanggan. Setiap Apoteker meramu satu atau lebih obat. Setiap resep memiliki satu atau lebih obat. Apoteker mencatat Obat setiap hari.

Skenario Kasus D

Pemerintah Daerah memiliki satu atau lebih unit kerja. Setiap Unit kerja berada pada satu gedung. Setiap unit kerja melayani satu jenis layanan. Setiap Unit Kerja memiliki satu atau lebih Sistem informasi. Setiap Unit Kerja memiliki satu atau beberapa Karyawan. Setiap Karyawan mengakses satu atau lebih Sistem Informasi. Setiap informasi memiliki satu akun.

Berdasarkan Skenario Kasus C dan Skenario Kasus D, jawablah pertanyaan-pertanyaan di bawah ini dimana setiap pertanyaan memiliki bobot nilai yang berbeda, dengan total nilai 100.

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL 09-KK02	CPMK 3.3.2	1. Praktekkanlah langkah 1 sampai 6 pada pada Skenario Kasus C.	Jawaban hasil praktek Kasus A	50
2.	CPL 09-KK02	CPMK 3.3.2	2. Praktekkanlah langkah 1 sampai 6 pada pada Skenario Kasus BD	Jawaban hasil praktek Kasus B	50



				Total Nilai	100
--	--	--	--	--------------------	------------

2.8. HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL 09-KK02	CPMK 3.3.2	20%		
2.	Praktik	CPL 09-KK02	CPMK 3.3.2	30%		
3.	Post-Test	CPL 09-KK02	CPMK 3.3.2	50%		
Total Nilai						

Daftar Pustaka : https://repository.dinus.ac.id/docs/ajar/9.materi_ERD_.pdf

LEMBAR JAWABAN PRE-TEST DAN POST-TEST PRAKTIKUM

Nama : NIM :	Asisten: Paraf Asisten:	Tanggal: Nilai:
-------------------------------	--	----------------------------------

PRAKTIKUM 3: NORMALISASI

Pertemuan ke : 3

Total Alokasi Waktu : 180 menit

- Materi : 30 menit
- Pre-Test : 30 menit
- Praktikum : 60 menit
- Post-Test : 60 menit

Total Bobot Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 30 %
- Post-Test : 50 %

Pemenuhan CPL dan CPMK:

CPL 09-KK02	Kemampuan merancang dan menganalisis algoritma untuk menyelesaikan permasalahan organisasi secara optimal, serta memilih dan menerapkannya pada bahasa pemrograman tertentu.
CPMK 3.3.2	Menerapkan pemodelan data sesuai arsitektur organisasi
Sub CPMK 3.3.2.3	Mengimplementasikan normalisasi Data

3.1. DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu:

1. Menerapkan konsep teoritis bidang area Informatika terkait matematika dasar dan ilmu komputer untuk memodelkan masalah dan meningkatkan produktivitas dengan menggunakan asesmen praktik dan posttest untuk materi normalisasi
2. Berpikir logis, kritis, sistematis dan inovatif, dan mampu mengambil keputusan secara tepat di bidang keahliannya melalui asesmen Pre test, praktik dan post test untuk materi normalisasi
3. Mengkaji/menganalisis implikasi pengembangan atau implementasi ilmu pengetahuan teknologi, menyusun deskripsi saintifik hasil kajian untuk pemecahan masalah dengan mempertimbangkan multidisiplin ilmu dengan asesmen Pre test, praktik dan post test untuk materi normalisasi
4. Memilih, membuat dan menerapkan teknik, sumber daya, penggunaan perangkat teknik modern dan implementasi teknologi informasi untuk memecahkan masalah.

3.2. INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:

CPL 09-KK02	CPMK 3.3.2	Sub CPMK 3.3.2.3	Mahasiswa mampu menganalisis dan mengimplementasikan Normalisasi
-------------	---------------	---------------------	--

3.3. TEORI PENDUKUNG

a. Definisi Normalisasi

Normalisasi database merupakan suatu pendekatan sistematis untuk meminimalkan redundansi data pada suatu database agar database tersebut dapat bekerja dengan optimal. Jika anda seorang database administrator ketika terjadi sesuatu pada database seperti penurunan kinerja, mungkin anda akan ditanya apakah database tersebut telah di normalisasi?

b. Tujuan Normalisasi Database

Tujuan normalisasi database adalah untuk menghilangkan dan mengurangi redundansi data dan tujuan yang kedua adalah memastikan dependensi data (Data berada pada tabel yang tepat). Jika data dalam database tersebut belum di normalisasi maka akan terjadi 3 kemungkinan yang akan merugikan sistem secara keseluruhan.

1. INSERT Anomali: Situasi dimana tidak memungkinkan memasukkan beberapa jenis data secara langsung di database.
2. DELETE Anomali: Penghapusan data yang tidak sesuai dengan yang diharapkan, artinya data yang harusnya tidak terhapus mungkin ikut terhapus.
3. UPDATE Anomali: Situasi dimana nilai yang diubah menyebabkan inkonsistensi database, dalam artian data yang diubah tidak sesuai dengan yang diperintahkan atau yang diinginkan.

c. Normalisasi Dalam Database

Normalisasi database terdiri dari banyak bentuk, dalam ilmu basis data ada setidaknya 9 bentuk normalisasi yang ada yaitu 1NF, 2NF, 3NF, EKNF, BCNF, 4NF, 5NF, DKNF, dan 6NF. Namun dalam prakteknya dalam dunia industri bentuk normalisasi ini yang paling sering digunakan ada sekitar 5 bentuk.

d. Tahapan Dalam Normalisasi

Tahap-tahap normalisasi tersebut adalah:

1) Bentuk Normal ke Satu (1NF)

Syarat:

- a. Tidak ada set atribut yang berulang atau bernilai ganda, setiap atribut yang dimiliknya bersifat atomic (bernilai tunggal) untuk setiap baris.
- b. Telah ditentukannya primary key untuk tabel atau relasi.
- c. Tiap atribut hanya memiliki satu pengertian.
- d. Tiap atribut yang dapat memiliki banyak nilai sebenarnya menggambarkan entitas atau relasi yang terpisah.

2) Bentuk Normal ke Dua (2NF)

Syarat:

- a. Bentuk data telah memenuhi kriteria bentuk normal ke satu.
- b. Atribut bukan kunci (non - key atribut) haruslah memiliki ketergantungan fungsional sepenuhnya pada primary key.
- c. Kunci primer hanya mengandung satu atribut.

3) Bentuk Normal ke Tiga (3NF)

Syarat:

- a. Bentuk data telah memenuhi kriteria ke dua.
- b. Tidak boleh terdapat ketergantungan transitif terhadap kunci utama atau primary key.

4) Boyce-Codd Normal Form (BCNF)

Syarat:

Semua anomali (kesalahan data) yang tersisa dari hasil penyempurnaan kebergantungan fungsional telah dihilangkan.

5) Bentuk Normal ke Empat (4NF)

Syarat:

- Bila dan hanya bila telah berada dalam bentuk BCNF dan tidak ada multivalued dependency nontrivial.
- Multivalued Dependency Nontrivial (MVD) dipakai dalam 4NF.
- Dependency ini dipakai untuk menyatakan hubungan satu (one to many).

6) Bentuk Normal ke Lima (5NF)

Syarat: Semua anomali (kesalahan data) yang tertinggal telah dihilangkan.

Dari beberapa tahap normalisasi di atas, Bentuk Normal Pertama (1NF) sampai Normal ke Tiga (3NF), merupakan bentuk normal yang umum dipakai. Umumnya bila ketiga bentuk normal tersebut telah dipenuhi, maka persoalan anomali tidak akan muncul.

3.4. HARDWARE DAN SOFTWARE

Hardware dan software yang digunakan dalam praktikum ini yaitu:

- Komputer.
- Buku Petunjuk Praktikum
- Aplikasi pemodelan: Microsoft Word

3.5. PRE-TEST

Jawablah pertanyaan dari studi kasus berikut (**Total Skor: 100**):

Studi Kasus:

ID_Pemesanan	Nama_Pelanggan	Alamat_Pelanggan	No_HP	Nama_Restoran	Alamat_Restoran	Menu_Pesanan	Harga_Menu	Jumlah	Total_Harga	Kurir	No_HP_Kurir
1	Andi Setiawan	Jl. Malioboro 10	811	Warung Sate A	Jl. Solo 25	Sate Aym	25000	2	50000	Budi	822
2	Andi Setiawan	Jl. Malioboro 10	811	Warung Sate A	Jl. Solo 25	Es Teh	5000	1	5000	Budi	822
3	Rina Amalia	Jl. Kaliurang 77	856	Nasi Goring B	Jl. Mageling 5	Nasi Goring	20000	1	20000	Sari	833
4	Rina Amalia	Jl. Kaliurang 77	856	Nasi Goring B	Jl. Mageling 5	Teh Botol	7000	2	14000	Sari	833

Pertanyaan:

No	CPL	CPMK	SUB CPMK	Pertanyaan	Skor
1.	09-KK02	3.3.2	3.3.2.3	Jelaskan apa yang dimaksud dengan Normalisasi Tabel	25
2.	09-KK02	3.3.2	3.3.2.3	Sebutkan dan jelaskan tujuan Normalisasi	25
3.	09-KK02	3.3.2	3.3.2.3	Sebutkan dan jelaskan tahapan Normalisasi yang umum dipakai	25
4.	09-KK02	3.3.2	3.3.2.3	Lakukan analisis permasalahan (anomali) dari table studi kasus di atas	25

3.6. LANGKAH PRAKTIKUM

- Bacalah kasus terkait entitas mahasiswa yang belum ternormalisasi pada Tabel 6.1. untuk proses Normalisasi dengan MySql menggunakan perinta: insert, hapus dan update.

Tabel 4. 1 Entitas Mahasiswa UnNormal

NIM	NAMA	Sem	MaKul	NIDN	Nama Dosen
201001	Andika Saputra	1	Algoritma	1078523	Riyanti Anjani
			Struktur Data	1078523	Riyanti Anjani
201002	Biyanti Anggie	3	Struktur Data	1078523	Riyanti Anjani
			Orkom	1078523	Riyanti Anjani
			Metnum	1075047	Susan Savitri
201003	Naura Putri	5	Web	1075047	Susan Savitri
			Jarkom	1077021	Erwin Masadi
			Metnum	1075047	Susan Savitri

- Dari Table 4.1 lakukanlah bentuk normalisasi tahap 1 (1 NF). Caranya dengan menggunakan fungsi hapus, insert dan update data dari table 4.1 dengan menggunakan fungsi dalam Mysql hasilnya dapat dilihat pada Tabel 4.2

Tabel 4. 2 Bentuk Normal 1 pada Entitas Mahasiswa

NIM	NAMA	Sem	MaKul	NIDN	Nama Dosen
201001	Andika Saputra	1	Algoritma	1078523	Riyanti Anjani
201001	Andika Saputra	1	Struktur Data	1078523	Riyanti Anjani
201002	Biyanti Anggie	3	Struktur Data	1078523	Riyanti Anjani
201002	Biyanti Anggie	3	Orkom	1078523	Riyanti Anjani
201002	Biyanti Anggie	3	Metnum	1075047	Susan Savitri
201003	Naura Putri	5	Web	1075047	Susan Savitri
201003	Naura Putri	5	Jarkom	1077021	Erwin Masadi
201003	Naura Putri	5	Metnum	1075047	Susan Savitri



3. Dari Table 4.2 dilakukan normalisasi lagi karena adanya kemunculan data dengan primary yang sama berulang. Caranya dengan memecah table 4.2 menjadi Tabel dosen dan table mahasiswa (sebagai entitas baru). Untuk itu kita perlu membuat table baru. Dalam MySQL fungsi membuat table dengan perintah Create. Hasilnya dapat dilihat pada Table 4.3, Tabel 4.4., Tabel 4.5. dan Tabel 4.6

Tabel 4. 3 Tabel Dosen belum Ternormalisasi

NIDN	NAMA DOSEN	MaKul_1	MaKul_2	MaKul_3
1078523	Riyanti Anjani	Algoritma	Struktur Data	Orkom
1075047	Susan Savitri	Metnum	Web	-
1077021	Erwin Masadi	Jarkom	-	-

Tabel 4. 4 Tabel Dosen Ternormalisasi bentuk 1

NIDN	NAMA DOSEN	MaKul
1078523	Riyanti Anjani	Algoritma
1078523	Riyanti Anjani	Struktur Data
1078523	Riyanti Anjani	Orkom
1075023	Susan Savitri	Metnum
1075023	Susan Savitri	Web
1077021	Erwin Masadi	Jarkom

Tabel 4. 5 Tabel Mahasiswa Normalisasi bentuk 1

NIM	NAMA	Sem	MaKul	NIDN	Nama Dosen
201001	Andika Saputra	1	Algoritma	1078523	Riyanti Anjani
201001	Andika Saputra	1	Struktur Data	1078523	Riyanti Anjani
201002	Biyanti Anggie	3	Struktur Data	1078523	Riyanti Anjani
201002	Biyanti Anggie	3	Orkom	1078523	Riyanti Anjani
201002	Biyanti Anggie	3	Metnum	1075047	Susan Savitri
201003	Naura Putri	5	Web	1075047	Susan Savitri
201003	Naura Putri	5	Jarkom	1077021	Erwin Masadi
201003	Naura Putri	5	Metnum	1075047	Susan Savitri

4. Merubah ke bentuk Normalisasi 2.

Untuk merubah ke bentuk Normalisasi bentuk 2, dapat dilakukan dengan cara: menentukan Atribut Kunci atau Primary Key.

Untuk Tabel 4.5. Entitas Mahasiswa dapat diubah menjadi Normalisasi bentuk ke 2, caranya:

- a) Tentukan Primary Key (PK) : NIM dengan atributnya sebagai berikut:

- Atribut yang bergantung dengan PK : Nama, Semester
- Atribut yang tidak bergantung dengan PK: Makul, NIDN, NamaDosen



- b) Untuk memenuhi 2NF, atribut yang tidak bergantung dengan primary key dipecah menjadi entitas baru, sehingga entitas mahasiswa dipecah menjadi 2 entitas, yaitu; Entitas Mahasiswa 2NF (Nim, Nama, Semester) dan Entitas Ambil_MK (Makul, Nidn, NamaDosen).

Tabel 4. 6 Entitas Mahasiswa bentuk 2 belum Ternormalisasi

NIM	NAMA	Sem
201001	Andika Saputra	1
201001	Andika Saputra	1
201002	Biyanti Anggie	3
201002	Biyanti Anggie	3
201002	Biyanti Anggie	3
201003	Naura Putri	5
201003	Naura Putri	5
201003	Naura Putri	5

Terjadi kerangkapan data yang tidak diperlukan (redundansi), untuk itu, data yang sama bisa dihilangkan.

Tabel 4. 7 Entitas Mahasiswa Normalisasi bentuk 2

NIM	NAMA	Sem
2010001	Andika Saputra	1
2010002	Biyanti Anggie	3
2010003	Naura Putri	5

Untuk entitas Ambil MK hasil Normalisasi Bentuk 2 disajikan pada Tabel 6.8.

Tabel 4. 8 Tabel Entitas Ambik MK Normalisasi 2

NIM	MaKul	NIDN	Nama Dosen
201001	Algoritma	1078523	Riyanti Anjani
201001	Struktur Data	1078523	Riyanti Anjani
201002	Struktur Data	1078523	Riyanti Anjani
201002	Orkom	1078523	Riyanti Anjani
201002	Metnum	1075047	Susan Savitri
201003	Web	1075047	Susan Savitri
201003	Jarkom	1077021	Erwin Masadi

5. Mengubah menjadi Tabel Normalisasi bentuk 3

Agar Tabel 4.8 Entitas ambil Kuliah Normal 2 dapat diubah menjadi Normalisasi bentuk 3, maka dilakukan cara sebagai berikut:

1. Menentukan Primary Key (PK) : NIDN
2. Menentukan atribut yang bergantung dengan PK : Nama Dosen
3. Menentukan Atribut yang bergantung transitif dengan PK: NIM, Makul
4. Untuk memenuhi 2NF, Entitas Ambil_MK 2NF dipecah menjadi 2 Entitas, yaitu; Entitas Dosen 3NF (NIDN, Nama Dosen) dan Entitas Ambil_MK 3NF (Nim, Makul).

Tabel 4. 9 Tabel Entitas Dosen Bentuk Normal 3

NIDN	Nama Dosen
1078523	Riyanti Anjani
1075047	Susan Savitri
1075047	Susan Savitri
1077021	Erwin Masadi

Menjadi

NIDN	Nama Dosen
1078523	Riyanti Anjani
1075047	Susan Savitri
1077021	Erwin Masadi

Tabel 4. 10 Tabel Ambil MK Bentuk Normal 3

NIM	Makul	NIDN
201001	Algoritma	1078523
201001	Struktur Data	1078523
201002	Struktur Data	1078523
201002	Orkom	1078523
201002	Metnum	1075047
201003	Web	1075047
201003	Jarkom	1077021
201003	Metnum	1078523

menjadi

6. Lakukanlah dengan membuat Query untuk membuat table normalisasi 1 sampai Normalisasi 3 dengan menggunakan perintah-perintah Insert, Delete dan Update.

Aturan Penilaian (Total Skor: 100):

No	CPL	CPMK	SUB CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	09-KK02	3.3.2	3.3.2.3	Selesaikan Langkah praktikum 1	Hasil praktikum Langkah 1 (Screenshot)	25
2.	09-KK02	3.3.2	3.3.2.3	Selesaikan Langkah praktikum 2	Hasil praktikum Langkah 2 (Screenshot)	25
3.	09-KK02	3.3.2	3.3.2.3	Selesaikan Langkah praktikum 3-4	Hasil praktikum Langkah 3-4 (Screenshot)	25
4.	09-KK02	3.3.2	3.3.2.3	Selesaikan Langkah praktikum 5-6	Hasil praktikum Langkah 5-6 (Screenshot)	25

3.7. POST TEST

Perhatikan Tabel 4.11. ini merupakan table bentuk Tidak Normal dari suatu fatur penjualan.

Tabel 4. 11 Bentuk Tidak Normal dari Tabel Penjualan

No Fatur	Tanggal	Kd-Pelanggan	Nama	Kode Barang	Nama Barang	Harga	QTY
F-01	12/10/20	P-001	Tommy	K-001	Mie	10.000	5
				K-002	Gula	20.000	2
F-02	15/10/20	P-002	Susi	K-001	Mie	10.000	3
				K-003	Garam	8.000	2
F-03	16/10/20	P-003	Yanti	K-004	Tepung	15.000	1
				K-002	Gula	20.000	1

Jawablah pertanyaan berikut (**Total Skor : 100**):

No	CPL	CPMK	SUB CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	09-KK02	3.3.2	3.3.2.3	Lakukan Normalisasi dari tabel diatas sampai dihasilkan tabel yang berkualitas baik!	Jawaban tertulis	50
2.	09-KK02	3.3.2	3.3.2.3	Implementasikan tabel hasil Normalisasi pada PHPMYADMIN dan isikan datanya sesuai dengan kasus tersebut	Screenshot bukti bahwa tabel sudah saling berelasikan	50

3.8. HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	SUB CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	09-KK02	3.3.2	3.3.2.3	20%		
2.	Praktik	09-KK02	3.3.2	3.3.2.3	30%		
3.	Post-Test	09-KK02	3.3.2	3.3.2.3	50%		
						Total Nilai	

LEMBAR JAWABAN PRE-TEST DAN POST-TEST PRAKTIKUM

Nama : NIM :	Asisten: Paraf Asisten:	Tanggal: Nilai:
-------------------------------	--	----------------------------------

PRAKTIKUM 4: DATA DEFINITION LANGUAGE (DDL)

Pertemuan ke : 4

Total Alokasi Waktu : 180 menit

- Materi : 30 menit
- Pre-Test : 30 menit
- Praktikum : 60 menit
- Post-Test : 60 menit

Total Bobot Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 30 %
- Post-Test : 50 %

Pemenuhan CPL dan CPMK:

CPL 03-KU03	Mampu dalam mendefinisikan kebutuhan pengguna atau pasar terhadap kinerja (menganalisis, mengevaluasi dan mengembangkan) algoritma/metode berbasis komputer untuk mengambil keputusan secara tepat dalam konteks penyelesaian masalah di bidang Rekayasa Perangkat Lunak serta Data dan Sistem Cerdas maupun bidang lainnya, berdasarkan hasil analisis informasi dan data.
CPMK 3.3.3	Mahasiswa dapat memahami, menganalisa, dan mengimplementasikan Bahasa query basis data DDL dan DML

4.1. DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu:

1. Menerapkan konsep teoritis bidang area Informatika terkait matematika dasar dan ilmu komputer untuk memodelkan masalah dan meningkatkan produktivitas dengan menggunakan asesmen Pre Test untuk materi terkait DDL.
2. Berpikir logis, kritis, sistematis dan inovatif, dan mampu mengambil keputusan secara tepat di bidang keahliannya melalui asesmen Pre test.
3. Mengkaji/menganalisis implikasi pengembangan atau implementasi ilmu pengetahuan teknologi, menyusun deskripsi saintifik hasil kajian untuk pemecahan masalah terkait DDL dengan asesmen praktik dan post test.
4. Memilih, membuat dan menerapkan teknik, sumber daya, penggunaan perangkat teknik modern dan implementasi teknologi informasi untuk memecahkan masalah terkait DDL dengan Asesmen Post Test dan Praktik.
5. Merancang dan mengimplementasikan algoritma/metode dalam mengidentifikasi dan memecahkan masalah yang melibatkan perangkat lunak dan pemikiran komputasi.



4.2. INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:

CPL 03- KU03	CPMK 3.3.3	Kemampuan mahasiswa meningkat dalam memahami konsep data dan informasi serta dapat berfikir logis yang diterapkan dengan cara mempraktekkan konsep DDL dan DML
-----------------	---------------	--

4.3. TEORI PENDUKUNG

DDL (Data Definition Language) ialah bahasa yang digunakan untuk mendeskripsikan data dan hubungannya dalam suatu database. DDL digunakan untuk membuat dan memodifikasi struktur objek pada suatu database menggunakan perintah dan sintaks spesifik yang telah ditetapkan. Dalam pengertian umum, DDL juga digunakan untuk mengacu pada bahasa yang mendeskripsikan data.

Data Definition Language memiliki fungsi untuk melakukan hal-hal berikut:

1. Membuat/menghapus database, dinyatakan dengan perintah CREATE DATABASE dan DROP DATABASE
2. Membuat/menghapus table, dinyatakan dengan perintah CREATE TABLE dan DROP TABLE
3. Memodifikasi table, dinyatakan dengan perintah ALTER TABLE

4.4. HARDWARE DAN SOFTWARE

Hardware dan software yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. Buku Petunjuk Praktikum
3. XAMPP
4. Browser (firefox, chrome)

4.5. PRE-TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL 03- KU03	CPMK 3.3.3	1. Jelaskan pengertian DDL ?	25
2.	CPL 03- KU03	CPMK 3.3.3	2. Jelaskan perbedaan DDL dan DML?	25
3.	CPL 03- KU03	CPMK 3.3.3	3. Jelaskan fungsi DDL dalam Database?	25
4.	CPL 03- KU03	CPMK 3.3.3	4. Berikan contoh minimal 3 dari perintah DDL	25

4.6. LANGKAH PRAKTIKUM

Langkah praktikum berisi tahapan secara rinci bagaimana praktikum dijalankan dan apa hasil yang harus dicapai dari setiap langkah.

Sebelum Masuk ke perintah DDL, Buka terlebih dahulu Jendela Shell yang terdapat pada XAMPP, kemudian ketik perintah “mysql -u root” untuk mengakses mysql.

Perintah DDL:

- 1) Menampilkan database yang ada pada mysql dengan perintah SHOW databases;



```
mysql> show databases;

+-----+
| Database |
+-----+
| erp_procurement |
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.02 sec)
```

2) Membuat database

```
CREATE database <nama_database>;
```

```
mysql> create database pbo;

Query OK, 1 row affected (0.04 sec)

mysql> show databases;

+-----+
| Database |
+-----+
| erp_procurement |
| information_schema |
| mysql |
| pbo |
| performance_schema |
| sys |
+-----+
6 rows in set (0.00 sec)
```

3) Menggunakan database

```
USE <nama_database>;
```

```
mysql> use pbo;
```

4) Menghapus database

```
DROP database <nama_database>;
```

```
mysql> drop database pbo;

Query OK, 0 rows affected (0.01 sec)
```

5) Melihat tabel apa yang sudah ada dalam database yang aktif

```
SHOW tables;
```

```
mysql> show tables;

+-----+
| Tables_in_pbo |
+-----+
| produk |
+-----+
1 row in set (0.00 sec)
```

6) Membuat tabel

Format perintah untuk membuat tabel

```
CREATE TABLE table_name (
    column1 datatype constraints,
    column2 datatype constraints,
    ...
);
```

Contoh perintah untuk membuat tabel produk

```
CREATE TABLE produk (
    id INT AUTO_INCREMENT PRIMARY KEY,
    namaProduk VARCHAR(255) NOT NULL,
    harga int NOT NULL
);

Query OK, 0 rows affected (0.03 sec)
```

7) Melihat struktur table

DESCRIBE <nama_tabel>;

DESC <nama_tabel>;

```
mysql> DESCRIBE produk;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| id | int | NO | PRI | NULL | auto_increment |
| namaProduk | varchar(255) | NO | | NULL | |
| harga | int | NO | | NULL | |
+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)
```

8) Mengubah struktur tabel

Ada saatnya kita sadar kalo ternyata struktur tabel yang pernah dibuat perlu penyempurnaan, bisa dalam hal penambahan kolom, pengubahan lebar kolom, penghapusan kolom, dan sebagainya, sehingga dengan penggunaan perintah ALTER ini maka kita dapat mengubah kekurangan atau kesalahan pada saat kita membuat tabel.

ALTER TABLE nama_tabel [spesifikasi perubahan]

Parameter [spesifikasi perubahan] adalah pilihan yang digunakan untuk mengubah struktur tabel yaitu CHANGE, ADD, DROP.

ALTER TABLE nama_tabel jenis_pengubahan

- a. Menambah kolom baru

Parameter yang digunakan adalah ADD

Format perintah menambah kolom baru

```
ALTER TABLE table_name
ADD column_name datatype;
```

Perintah penambahan kolom baru email (100) di tabel employees

```
ALTER TABLE Employees
ADD email VARCHAR(100);
```

- b. Mengubah nama kolom

Parameter yang digunakan adalah CHANGE.

```
ALTER TABLE table_name CHANGE old_column_name new_column_name
column_definition;
```

Contoh perintah mengubah nama kolom dari full_name menjadi employee_name



```
ALTER TABLE Employees CHANGE full_name employee_name
VARCHAR(100);
```

c. Mengubah nama tabel

Parameter yang digunakan adalah RENAME.

```
ALTER TABLE nama_lama RENAME [TO] nama_baru;
```

```
ALTER TABLE Employees RENAME TO employee;
```

d. Mengelola konstrain pada tabel

1. Menambah primary key

Format

```
ALTER TABLE table_name ADD CONSTRAINT constraint_name
PRIMARY KEY (column_name);
```

Contoh menambah primary key untuk kolom customers

```
mysql> alter table customers add constraint pk_customer
primary key (customer_id, email);
Query OK, 0 rows affected (0.07 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

2. Menghapus primary key

```
mysql> alter table customers drop primary key;
Query OK, 4 rows affected (0.07 sec)
Records: 4 Duplicates: 0 Warnings: 0
```

3. Menambah foreign key

```
ALTER TABLE table_name
ADD CONSTRAINT constraint_name FOREIGN KEY (column_name)
REFERENCES referenced_table_name(referenced_column_name);
```

```
mysql> alter table produk add constraint fk_kategori
foreign key (kategori) references kategori(nama_kategori);
Query OK, 2 rows affected (0.08 sec)
Records: 2 Duplicates: 0 Warnings: 0
```

ON DELETE CASCADE

Ketika data di tabel *references* dihapus, semua record terkait di tabel foreign key juga akan dihapus. *Ketika data di tabel kategori dihapus, semua record terkait di tabel produk dihapus.*

```
mysql> alter table produk add constraint fk_kategori foreign key
(kategori) references kategori(nama_kategori) on delete cascade;
Query OK, 2 rows affected (0.06 sec)
Records: 2 Duplicates: 0 Warnings: 0
```

ON UPDATE CASCADE

Ketika data di tabel *references* diperbarui, semua record terkait di tabel foreign key juga akan diperbarui. *Ketika data di tabel kategori diperbarui, semua record terkait di tabel produk diperbarui.*

```
mysql> alter table produk add constraint fk_kategori foreign key
(kategori) references kategori(nama_kategori) on update cascade;
Query OK, 2 rows affected (0.07 sec)
Records: 2 Duplicates: 0 Warnings: 0
```

4. Melihat konstrain tabel



```
mysql> show create table customers;
+-----+-----+
| Table | Create Table
+-----+-----+
| customers | CREATE TABLE `customers` (
  `customer_id` int NOT NULL,
  `first_name` varchar(50) DEFAULT NULL,
  `last_name` varchar(50) DEFAULT NULL,
  `email` varchar(100) DEFAULT NULL,
  `city` varchar(50) DEFAULT NULL,
  `country` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`customer_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+-----+-----+
```

5. Menghapus foreign key

```
mysql> alter table produk drop foreign key fk_kategori;
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

6. Membuat konstrain unique

Format

```
alter table <nama_table> add constraint <nama_konstrain_unik>
unique (<nama_kolom>);
```

Contoh

```
mysql> alter table customers add constraint unique_phone
unique (phone_number);
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

7. Menghapus konstrain unique

Format

```
ALTER TABLE table_name
DROP INDEX unique_name;
```

Contoh

```
mysql> alter table customers drop index unique_phone;
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Aturan Penilaian (Total Skor: 100):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL 03-KU3	CPMK 3.3.3.	Praktekkanlah seluruh langkah-langkah	Jawaban hasil praktek (screenshot)	100

4.7. POST TEST

Jawablah pertanyaan-pertanyaan di bawah ini dimana setiap pertanyaan memiliki bobot nilai yang berbeda, dengan total nilai 100.

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL 03-	CPMK	1. Buatlah sebuah database	Jawaban hasil	100



	KU03	3.3.3	(bebas sesuai keinginan) - DDL 2. Database terdiri dari 2 buah tabel (nama dan jenis tabel bebas sesuai dengan database yang dirancang) 3. Pada masing-masing tabel, rancanglah 4 buah atribut dan tentukan primary keynya	praktek (Screenshot)	
					Total Nilai 100

4.8. HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL 03-KU03	CPMK 3.3.3	20%		
2.	Praktik	CPL 03-KU03	CPMK 3.3.3	30%		
3.	Post-Test	CPL 03-KU03	CPMK 3.3.3	50%		
Total Nilai						

LEMBAR JAWABAN PRE-TEST DAN POST-TEST PRAKTIKUM

Nama : NIM :	Asisten: Paraf Asisten:	Tanggal: Nilai:
-------------------------------	--	----------------------------------

PRAKTIKUM 5: INDEKS

Pertemuan ke : 5

Total Alokasi Waktu : 180 menit

- Materi : 30 menit
- Pre-Test : 30 menit
- Praktikum : 60 menit
- Post-Test : 60 menit

Total Bobot Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 30 %
- Post-Test : 50 %

Pemenuhan CPL dan CPMK:

CPL 03-KU3	Mempunyai kemampuan dalam mendefinisikan kebutuhan pengguna atau pasar terhadap kinerja (menganalisis, mengevaluasi dan mengembangkan) algoritma/metode berbasis komputer untuk mengambil keputusan secara tepat dalam konteks penyelesaian masalah di bidang Rekayasa Perangkat Lunak serta Data dan Sistem Cerdas maupun bidang lainnya, berdasarkan hasil analisis informasi dan data.
CPMK 3.3.3	Mengimplementasikan query basis data dan lingkungannya

5.1. DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu:

1. Menerapkan konsep teoritis bidang area Informatika terkait matematika dasar dan ilmu komputer untuk memodelkan masalah dan meningkatkan produktivitas dengan menggunakan asesmen Pre Test untuk materi terkait indeks pada tabel
2. Berpikir logis, kritis, sistematis dan inovatif, dan mampu mengambil keputusan secara tepat di bidang keahliannya melalui asesmen Pre test.
3. Mengkaji/menganalisis implikasi pengembangan atau implementasi ilmu pengetahuan teknologi, menyusun deskripsi saintifik hasil kajian untuk pemecahan masalah terkait indeks pada tabel
4. Memilih, membuat dan menerapakan teknik, sumber daya, penggunaan perangkat teknik modern dan implementasi teknologi informasi untuk memecahkan masalah terkait indeks pada tabel
5. Merancang dan mengimplementasikan algoritma/metode dalam mengidentifikasi dan memecahkan masalah yang melibatkan perangkat lunak dan pemikiran komputasi.

5.2. INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:

CPL 03- KU3	CPMK 3.3.3	Kemampuan mahasiswa meningkat sebelum dan setelah penerapan indeks beserta evaluasi performa basis data
----------------	---------------	---

5.3. TEORI PENDUKUNG

1. Pendahuluan

Indeks pada basis data bertujuan untuk meningkatkan kecepatan pencarian dan pengambilan data dari tabel. Sama seperti daftar isi di buku yang memudahkan mencari halaman tertentu, indeks membantu sistem basis data menemukan data dengan lebih cepat tanpa harus memindai seluruh tabel.

2. Tujuan Praktikum

- Memahami cara membuat dan menggunakan indeks.
- Menganalisis dampak indeks terhadap kinerja query.
- Mengetahui perbedaan antara indeks clustered dan non-clustered.
- Melakukan analisis performa pada query dengan dan tanpa indeks.

3. Konsep Dasar Indeks

- **Indeks Clustered:** Tipe indeks yang menyusun data fisik di dalam tabel berdasarkan urutan kolom tertentu. Setiap tabel hanya bisa memiliki satu index clustered.
- **Indeks Non-Clustered:** Tipe indeks yang menyimpan pointer ke data asli di dalam tabel. Sebuah tabel bisa memiliki banyak index non-clustered.
- **Primary Key:** Secara otomatis menjadi indeks clustered jika tidak dinyatakan sebaliknya.
- **Unique Index:** Menjamin bahwa kolom atau kombinasi kolom memiliki nilai yang unik.

5.4. HARDWARE DAN SOFTWARE

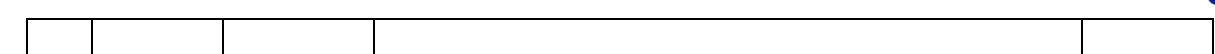
Hardware dan software yang digunakan dalam praktikum ini yaitu:

1. Komputer
2. Buku Petunjuk Praktikum
3. Command Line Interface

5.5. PRE-TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL-03- KU3	CPMK- 3.3.3	1. Jelaskan pengertian indeks!	25
2.	CPL-03- KU3	CPMK- 3.3.3	2. Berikan analogi atau contoh indeks di dunia nyata	25
3.	CPL-03- KU3	CPMK- 3.3.3	3. Jelaskan jenis-jenis indeks	25
4.	CPL-03- KU3	CPMK- 3.3.3	4. Apa manfaat indeks?	25



5.6. LANGKAH PRAKTIKUM

1. Unduh file customers.sql dari [tautan ini](#).

```
-- MySQL dump 10.13 Distrib 8.0.37, for Win64 (x86_64)

-- 
-- Host: localhost      Database: <nama database>
-- 
-- Server version 8.0.37

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!50503 SET NAMES utf8mb4 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

-- 
-- Table structure for table `customers`
-- 

DROP TABLE IF EXISTS `customers`;

/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;

CREATE TABLE `customers` (
    `customer_id` int NOT NULL,
    `first_name` varchar(50) DEFAULT NULL,
    `last_name` varchar(50) DEFAULT NULL,
    `email` varchar(100) DEFAULT NULL,
```

```

`city` varchar(50) DEFAULT NULL,
`country` varchar(50) DEFAULT NULL,
PRIMARY KEY (`customer_id`),
KEY `idx_city_country` (`city`, `country`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

/*!40101 SET character_set_client = @saved_cs_client */;

-- 
-- Dumping data for table `customers`
-- 

LOCK TABLES `customers` WRITE;

/*!40000 ALTER TABLE `customers` DISABLE KEYS */;

INSERT INTO `customers` VALUES (1,'John','Doe','john.doe@example.com','New York','USA'),(2,'Jane','Smith','jane.smith@example.com','Los Angeles','USA'),(3,'Michael','Brown','michael.brown@example.com','Chicago','USA'),(4,'Emily','Davis','emily.davis@example.com','New York','USA');

/*!40000 ALTER TABLE `customers` ENABLE KEYS */;

UNLOCK TABLES;

/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

-- 
-- Dump completed on 2024-09-07 16:48:50

```

2. Pada CLI (*command prompt*), masuk ke direktori tempat file customers.sql disimpan.
Misalnya di C:\Users\Lenovo\sql_backup>
3. Jalankan file customers.sql agar bisa membuat tabel di dalam database Anda.

Sintaks dasar: mysql -u username -p database_name < file.sql

```
C:\Users\Lenovo\sql_backup>mysql -u root -p pbo < customers.sql
```

- Masuk ke mysql, lalu periksa apakah tabel sudah berhasil dibuat dengan perintah

```
mysql> show tables;
```

```
+-----+
| Tables_in_pbo |
+-----+
| customers    |
| produk       |
+-----+
2 rows in set (0.00 sec)
```

- Periksa skema tabel dengan perintah

```
mysql> describe customers;
```

```
+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| customer_id | int      | NO   | PRI | NULL    |       |
| first_name  | varchar(50) | YES  |     | NULL    |       |
| last_name   | varchar(50) | YES  |     | NULL    |       |
| email       | varchar(100) | YES  |     | NULL    |       |
| city        | varchar(50) | YES  |     | NULL    |       |
| country     | varchar(50) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
```

```
6 rows in set (0.00 sec)
```

- Kita lihat dulu seluruh record sebelum diberikan indeks

```
mysql> SELECT * FROM customers WHERE city = 'New York';
+-----+-----+-----+-----+-----+-----+-----+
| customer_id | first_name | last_name | email           | city      | country |
+-----+-----+-----+-----+-----+-----+
|          1 | John       | Doe       | john.doe@example.com | New York | USA      |
|          4 | Emily      | Davis     | emily.davis@example.com | New York | USA      |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```



7. Gunakan EXPLAIN

```
mysql> EXPLAIN SELECT * FROM customers WHERE city = 'New York';
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table      | partitions | type | possible_keys | key   | key_len | ref    | rows | filtered | Extra
|    |             |            |           |       |             |       |         |        |       |          |       |
+-----+-----+-----+-----+-----+-----+-----+-----+
|  1 | SIMPLE      | customers  | NULL       | ALL  | NULL        | NULL  | NULL    | NULL   | 4    | 25.00    | Using
where |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.04 sec)
```

8. Gunakan EXPLAIN ANALYZE

```
mysql> EXPLAIN ANALYZE SELECT * FROM customers WHERE city = 'New York';
+-----+
| EXPLAIN
|
+-----+++
| -> Filter: (customers.city = 'New York') (cost=0.65 rows=1) (actual
time=0.0633..0.0749 rows=2 loops=1)
      -> Table scan on customers (cost=0.65 rows=4) (actual time=0.0578..0.0681 rows=4
loops=1)
|
+-----+
1 row in set (0.04 sec)
```

9. Membuat Indeks satu kolom

```
mysql> CREATE INDEX idx_city ON customers(city);
Query OK, 0 rows affected (0.11 sec)

Records: 0  Duplicates: 0  Warnings: 0
```

10. Melihat perubahan konstrain setelah ditambah indeks pada kolom City

```
mysql> show create table customers;
+-----+
| Table | Create Table
|
+-----+
| customers | CREATE TABLE `customers` (
```

```

`customer_id` int NOT NULL,
`first_name` varchar(50) DEFAULT NULL,
`last_name` varchar(50) DEFAULT NULL,
`email` varchar(100) DEFAULT NULL,
`city` varchar(50) DEFAULT NULL,
`country` varchar(50) DEFAULT NULL,
PRIMARY KEY (`customer_id`),
KEY `idx_city` (`city`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+
-----+
1 row in set (0.04 sec)

```

11. Periksa kembali menggunakan Explain

```

mysql> EXPLAIN SELECT * FROM customers WHERE city = 'New York';
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table      | partitions | type | possible_keys | key      | key_len | ref   | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|  1 | SIMPLE      | customers | NULL       | ref  | idx_city     | idx_city | 203    | const |    2 |    100.00 | NULL  |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.04 sec)

```

12. Periksa kembali menggunakan Explain Analyze

```

mysql> EXPLAIN ANALYZE SELECT * FROM customers WHERE city = 'New York';
+-----+
|EXPLAIN
|
+-----+
| -> Index lookup on customers using idx_city (city='New York')  (cost=0.7 rows=2)
(actual time=0.153..0.16 rows=2 loops=1)
|
+-----+

```



```
1 row in set (0.04 sec)
```

13. Membuat indeks untuk multi kolom

```
mysql> CREATE INDEX idx_city_country ON customers(city, country);
Query OK, 0 rows affected (0.10 sec)

Records: 0  Duplicates: 0  Warnings: 0
```

14. Analisis menggunakan Explain

```
mysql> explain SELECT * FROM customers WHERE city = 'New York' AND country = 'USA';
+-----+-----+-----+-----+-----+-----+
| id | select_type | table| partitions | type | possible_keys     | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+
|  1 | SIMPLE      | customers | NULL       | ref  | idx_city_country | idx_city_country | 406   |       | const,const | 2 |    100.00 | NULL  |
+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.00 sec)
```

15. Analisis menggunakan explain analyze

```
mysql> explain analyze SELECT * FROM customers WHERE city = 'New York' AND country = 'USA';
+-----+
| EXPLAIN
|
+-----+
| -> Index lookup on customers using idx_city_country (city='New York', country='USA')  (cost=0.7
rows=2) (actual time=0.0548..0.0593 rows=2 loops=1)
|
+-----+
1 row in set (0.00 sec)
```

16. Menghapus indeks

Sintaks dasar: `DROP INDEX <nama kolom> ON <nama tabel>;`

```
mysql> DROP INDEX idx_city ON customers;
Query OK, 0 rows affected (0.07 sec)

Records: 0  Duplicates: 0  Warnings: 0
```



17. Periksa kembali skema tabel setelah penghapusan indeks

```
mysql> show create table customers;
+-----+-----+
| Table      | Create
Table
|
+-----+-----+
| customers | CREATE TABLE `customers` (
  `customer_id` int NOT NULL,
  `first_name` varchar(50) DEFAULT NULL,
  `last_name` varchar(50) DEFAULT NULL,
  `email` varchar(100) DEFAULT NULL,
  `city` varchar(50) DEFAULT NULL,
  `country` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`customer_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+-----+-----+
```

Aturan Penilaian (Total Skor: 100):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL-03-KU3	CPMK-3.3.3	1. Praktikkanlah langkah Langkah 1-17	Jawaban hasil praktek (Screenshot)	30

5.7. POST TEST

Jawablah pertanyaan-pertanyaan di bawah ini dimana setiap pertanyaan memiliki bobot nilai yang berbeda, dengan total nilai 100

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL-03-KU3	CPMK-3.3.3	1. Tambahkan minimal 10 record ke tabel customers	Jawaban hasil praktik (Screenshot)	30
2.	CPL-03-KU3	CPMK-3.3.3	2. Analislah menggunakan EXPLAIN dan EXPLAIN ANALYZE pada saat sebelum dan setelah menggunakan indeks. Jelaskan makna tiap output keduanya 3. Berikan kesimpulan Anda!	Jawaban hasil praktik (Screenshot) Uraian	30 40
				Total Nilai	100

5.8. HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL-03-KU3	CPMK-3.3.3	20%		
2.	Praktik	CPL-03-KU3	CPMK-3.3.3	30%		
3.	Post-Test	CPL-03-KU3	CPMK-3.3.3	50%		
Total Nilai						

LEMBAR JAWABAN PRE-TEST DAN POST-TEST PRAKTIKUM

Nama : NIM :	Asisten: Paraf Asisten:	Tanggal: Nilai:
-------------------------------	--	----------------------------------

PRAKTIKUM 6: DML DAN FUNGSI AGREGASI

Pertemuan ke : 6

Total Alokasi Waktu : 180 menit

- Materi : 30 menit
- Pre-Test : 30 menit
- Praktikum : 60 menit
- Post-Test : 60 menit

Total Bobot Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 30 %
- Post-Test : 50 %

Pemenuhan CPL dan CPMK:

CPL 03-KU03	Mampu dalam mendefinisikan kebutuhan pengguna atau pasar terhadap kinerja (menganalisis, mengevaluasi dan mengembangkan) algoritma/metode berbasis komputer untuk mengambil keputusan secara tepat dalam konteks penyelesaian masalah di bidang Rekayasa Perangkat Lunak serta Data dan Sistem Cerdas maupun bidang lainnya, berdasarkan hasil analisis informasi dan data.
CPMK 3.3.3	Mahasiswa dapat memahami, menganalisa, dan mengimplementasikan Bahasa query basis data DDL dan DML

6.1. DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu:

6. Menerapkan konsep teoritis bidang area Informatika terkait matematika dasar dan ilmu komputer untuk memodelkan masalah dan meningkatkan produktivitas dengan menggunakan asesmen Pre Test untuk materi terkait DML dan Fungsi Agregasi.
7. Berpikir logis, kritis, sistematis dan inovatif, dan mampu mengambil keputusan secara tepat di bidang keahliannya melalui asesmen Pre test.
8. Mengkaji/menganalisis implikasi pengembangan atau implementasi ilmu pengetahuan teknologi, menyusun deskripsi saintifik hasil kajian untuk pemecahan masalah terkait DML dan Fungsi Agregasi dengan asesmen praktik dan post test.
9. Memilih, membuat dan menerapkan teknik, sumber daya, penggunaan perangkat teknik modern dan implementasi teknologi informasi untuk memecahkan masalah terkait DML dan Fungsi Agregasi dengan Asesmen Post Test dan Praktik.
10. Merancang dan mengimplementasikan algoritma/metode dalam mengidentifikasi dan memecahkan masalah yang melibatkan perangkat lunak dan pemikiran komputasi.

6.2. INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:



CPL 03-KU03	CPMK 3.3.3	1. Kemampuan mahasiswa meningkat dalam memahami konsep data dan informasi serta dapat berfikir logis yang diterapkan dengan cara mempraktekkan konsep DDL dan DML
CPL 03-KU03	CPMK 3.3.3	2. Kemampuan mahasiswa meningkat dalam menganalisa, memilih dan mengimplementasikan masalah terkait DDL dan DML dengan menggunakan tool pemodelan yang tersedia
CPL 03-KU03	CPMK 3.3.3	3. Kemampuan mahasiswa meningkat dalam menerapkan teknik, penggunaan perangkat teknik dengan fungsi agregasi
		4. Kemampuan mahasiswa meningkat dalam merancang dan mengimplementasikan algoritma dengan menggunakan fungsi agregasi

6.3. TEORI PENDUKUNG

Data Manipulation Language terdiri atas :

1. Mengisi tabel dengan data, dinyatakan dengan perintah INSERT
2. Mengedit data pada tabel, dinyatakan dengan perintah UPDATE
3. Menghapus data pada tabel, dinyatakan dengan perintah DELETE
4. Mencari data pada tabel, dinyatakan dengan perintah SELECT

Fungsi agregasi meliputi :

1. Menghitung banyak record
2. Menghitung total nilai suatu atribut
3. Menghitung rata-rata nilai atribut
4. Mencari nilai terbesar dari nilai atribut
5. Mencari nilai terkecil dari nilai atribut

Tabel 7. 1 Query Formal

KLAUSA	PENJELASAN
AVG	Sama dengan
COUNT	Mengetahui jumlah record
MAX	Mengetahui nilai maximal
MIN	Mengetahui nilai minimal
SUM	Menghitung jumlah data

Berikut beberapa operator yang biasanya diikuti Klausus WHERE :

Tabel 7. 2 Operator MySQL

OPERATOR	PENJELASAN
=	Sama dengan
< >, !=	Tidak sama dengan
<	Kurang dari
>	Lebih besar dari
< =	Kurang dari atau sama dengan
> =	Lebih dari atau sama dengan
! >	Tidak lebih besar dari

! <	Tidak lebih kecil dari
BETWEEN	Antara dua nilai yang ditentukan
LIKE	Menyesuaikan nilai yang ditentukan
IS NULL	Nilainya adalah NULL
IN	Nilainya ditentukan dalam sebuah daftar
NOT	Negasi dari sebuah operator perbandingan
AND	Merangkai kriteria pencarian
OR	Memastikan bahwa criteria pencarian adalah eksklusif

6.4. HARDWARE DAN SOFTWARE

Hardware dan software yang digunakan dalam praktikum ini yaitu:

4. Komputer
5. Buku Petunjuk Praktikum
6. Aplikasi pemodelan: XAMPP
7. Browser (firefox, chrome)

6.5. PRE-TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL 03-KU03	CPMK 3.3.3	1. Jelaskan pengertian DML?	15
2.	CPL 03-KU03	CPMK 3.3.3	2. Jelaskan perbedaan DDL dan DML?	10
3.	CPL 03-KU03	CPMK 3.3.3	3. Jelaskan fungsi DML?	10
4.	CPL 03-KU03	CPMK 3.3.3	4. Berikan contoh minimal 3 dari perintah DML!	15
5.	CPL 03-KU03	CPMK 3.3.3		
6.	CPL 03-KU03	CPMK 3.3.3	8. Jelaskan maksud dari count, sum, min, max, dan avg dalam fungsi agregasi	20
7.	CPL 03-KU03	CPMK 3.3.3	9. Sebutkan dan jelaskan operator dalam MySQL	20
8.	CPL 03-KU03	CPMK 3.3.3	10. Apa maksud dari Query : SELECT SUM(nilai) AS Total FROM nilai;	10

6.6. LANGKAH PRAKTIKUM

1. Perintah DML

- a) Mengisi tabel

Ada beberapa cara dalam memasukkan data yaitu dengan menyamakan kolom dan data, menyebutkan kolom, tanpa menyebutkan kolom, memasukkan hanya sebagai pada kolom. Menyamakan Kolom dan Data, perintahnya sebagai berikut:

```
INSERT INTO nama_tabel SET kolom_pertama = 'data_kolom_pertama', kolom_kedua = 'data_kolom_kedua', kolom_terakhir = 'data kolom terakhir' ;
INSERT INTO mahasiswa SET NIM = '18451', nama_mhs ='Nur Fatayah', alamat_mhs='Yogyakarta';
```

Menyebutkan Kolom, perintahnya sebagai berikut:

```
INSERT INTO nama_tabel (kolom_pertama, kolom_kedua, kolom_terakhir) VALUES
(data_kolom_pertama, data_kolom_kedua, data_kolom_terakhir);
INSERT INTO mahasiswa (NIM, nama_mhs, alamat_mhs) VALUES ('18451', 'Nur Fatayah',
'Yogyakarta');
```

Tanpa Menyebutkan Kolom

```
INSERT INTO nama_tabel VALUES (data_kolom_pertama,
data_kolom_kedua,data_kolom_terakhir);
```

```
INSERT INTO mahasiswa VALUES ('18451', Nur Fatayah', 'Yogyakarta')
```

```
MariaDB [akademik]> insert into mahasiswa values ('18451', 'Nur Fatayah', 'Yogyakarta');
Query OK, 1 row affected (0.340 sec)
```

```
MariaDB [akademik]> insert into mahasiswa values ('18324', 'Mahardika', 'Sleman');
Query OK, 1 row affected (0.071 sec)
```

```
MariaDB [akademik]>
```

b) Melihat isi tabel

Perintah ini digunakan untuk menyeleksi atau memilih atau menampilkan data-data yang ada dalam tabel. baik menampilkan semua kolom, sebagian kolom, serta berdasarkan kondisi.

1. Perintah untuk menampilkan data apa adanya, tanpa syarat, tanpa pemilihan kolom dan tanpa urutan :

```
SELECT * FROM <nama_tabel>
SELECT * FROM mahasiswa;
```

2. Membatasi jumlah record yang dibaca

Untuk membatasi record yang muncul atau untuk mencari record dengan kriteria tertentu, digunakan klausa where. Misal untuk melihat nama mahasiswa dengan nim = '18451'

```
SELECT * from mahasiswa WHERE nim = '18451'
```

Melihat data mahasiswa yang bernama 'Nur Fatayah'

```
SELECT * from mahasiswa where nama_mhs = 'Nur Fatayah'
```

Melihat data mahasiswa yang beralamat di Sleman

```
SELECT * from mahasiswa where alamat_mhs like'%Sleman'
```

Pada kriteria alamat, terlihat ada penggunaan karakter '%'. Karakter ini mengandung makna, apa pun teks yang ada akan memenuhi kriteria. Jadi '%Sleman' artinya semua string yang diakhiri kata Sleman.

```
MariaDB [akademik]> select * from mahasiswa;
+----+-----+-----+
| NIM | nama_mhs | alamat_mhs |
+----+-----+-----+
| 18324 | Mahardika | Sleman |
| 18451 | Nur Fatayah | Yogyakarta |
+----+-----+-----+
2 rows in set (0.001 sec)

MariaDB [akademik]> select * from mahasiswa where nim = '18451';
+----+-----+-----+
| NIM | nama_mhs | alamat_mhs |
+----+-----+-----+
| 18451 | Nur Fatayah | Yogyakarta |
+----+-----+-----+
1 row in set (0.001 sec)

MariaDB [akademik]> -
```

Gambar 6.1 Perintah Tampil Data

3. Membatasi jumlah field yang dibaca

Klausa order by digunakan untuk mengurutkan data yang diminta dengan query. Misal diminta untuk menampilkan nim dan nama mahasiswa yang urut oleh nim secara menaik:

`SELECT nim, nama_mhs from mahasiswa order by nim asc;`

Jika urut menurun:

`SELECT nim, nama_mhs from mahasiswa order by nim desc;`

c) Menampilkan data berurutan

Klausa order by digunakan untuk mengurutkan data yang diminta dengan query. Misal diminta untuk menampilkan nim dan nama mahasiswa yang urut oleh nim secara menaik:

`SELECT nim, nama_mhs from mahasiswa order by nim asc;`

Jika urut menurun:

`SELECT nim, nama_mhs from mahasiswa order by nim desc;`

```
MariaDB [akademik]> select nim, nama_mhs from mahasiswa order by nim asc;
+-----+-----+
| nim | nama_mhs |
+-----+-----+
| 18090 | Sofia Ananta |
| 18102 | Rahma Nadia |
| 18115 | Dian Sastro |
| 18324 | Mahardika |
| 18451 | Nur Fatayah |
+-----+-----+
5 rows in set (0.049 sec)

MariaDB [akademik]> select nim, nama_mhs from mahasiswa order by nim desc;
+-----+-----+
| nim | nama_mhs |
+-----+-----+
| 18451 | Nur Fatayah |
| 18324 | Mahardika |
| 18115 | Dian Sastro |
| 18102 | Rahma Nadia |
| 18090 | Sofia Ananta |
+-----+-----+
5 rows in set (0.001 sec)

MariaDB [akademik]> -
```

Gambar 6.2 Tampil Data Berdasarkan Urutan

d) Mengubah isi data tabel

`UPDATE nama_tabel SET kolom_pertama = 'data_kolom_pertama',`

`kolom_kedua = 'data_kolom_kedua',`

`kolom_terakhir = 'data_kolom_terakhir',`

`WHERE kondisi`

Penggunaan perintah UPDATE tanpa klausa WHERE mengakibatkan semua data dalam satu kolom akan diubah. Misal akan diubah nama ‘Nur Fatayah’ menjadi ‘N Fatayah’

`UPDATE mahasiswa SET nama_mhs = " where nim = '18451';`

```
MariaDB [akademik]> update mahasiswa set nama_mhs = 'N Fatayah' where nim='18451';
Query OK, 1 row affected (0.130 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [akademik]> select nim, nama_mhs from mahasiswa order by nim desc;
+-----+-----+
| nim | nama_mhs |
+-----+-----+
| 18451 | N Fatayah |
| 18324 | Mahardika |
| 18115 | Dian Sastro |
| 18102 | Rahma Nadia |
| 18090 | Sofia Ananta |
+-----+
5 rows in set (0.001 sec)

MariaDB [akademik]>
```

Gambar 6.3 Tampil Data Urut Abjad

- e) Menghapus isi tabel
`DELETE FROM nama_tabel WHERE kondisi`

Apabila tidak menggunakan klausua WHERE maka akan menyebabkan semua record dalam tabel terhapus. Misal akan dihapus data mahasiswa bernama ‘Sofia Ananta’

`DELETE FROM mahasiswa WHERE nim = ‘18090’`

```
MariaDB [akademik]> delete from mahasiswa where nim='18090';
Query OK, 1 row affected (0.139 sec)

MariaDB [akademik]> select * from mahasiswa;
+-----+-----+-----+
| NIM | nama_mhs | alamat_mhs |
+-----+-----+-----+
| 18102 | Rahma Nadia | Gunung Kidul |
| 18115 | Dian Sastro | Jakarta |
| 18324 | Mahardika | Sleman |
| 18451 | N Fatayah | Yogyakarta |
+-----+-----+-----+
4 rows in set (0.001 sec)

MariaDB [akademik]> -
```

Gambar 6.4 Menghapus Isi Tabel

- f) Penggunaan Limit dan Offset

Digunakan untuk membatasi jumlah baris yang ditampilkan dalam select dengan fungsi LIMIT, dan melewati dengan fungsi OFFSET. Misal mau menampilkan 3 mahasiswa urut berdasarkan nim.

```
MariaDB [akademik]> select * from mahasiswa order by nim LIMIT 3;
+-----+-----+-----+
| nim | nama       | alamat      |
+-----+-----+-----+
| 18090 | Sofia Ananta | Kulon Progo |
| 18102 | Rahma Nadia | Gunung Kidul |
| 18115 | Dian Sastro | Jakarta     |
+-----+-----+-----+
3 rows in set (0.001 sec)
```

Gambar 6.5 Perintah Select

Kemudian fungsi offset digunakan untuk melewati beberapa baris.

```
MariaDB [akademik]> select * from mahasiswa order by nim LIMIT 2 OFFSET 2;
+-----+-----+-----+
| nim | nama | alamat |
+-----+-----+-----+
| 18115 | Dian Sastro | Jakarta |
| 18324 | Mahardika | Sleman |
+-----+-----+-----+
2 rows in set (0.001 sec)
```

Gambar 6.6 Penerapan Limit dan Offset

2. Fungsi Agregasi

Buat tabel mata_kuliah dan isikan seperti gambar di bawah:

```
mysql> select * from mata_kuliah;
+-----+-----+-----+
| kode_kul | nama_kul | sks | sem |
+-----+-----+-----+
| IT0101 | Logika Inf | 3 | 1 |
| IT0102 | Studi Islam I | 2 | 1 |
| IT0103 | Kalkulus Inf | 3 | 1 |
| IT0301 | Sertifikasi | 0 | 3 |
| IT0401 | Basis Data | 3 | 4 |
| IT0801 | Tugas Akhir | 6 | 8 |
+-----+-----+-----+
6 rows in set <0.00 sec>
```

Gambar 6.7 Tampil Seluruh Data

- Menampilkan data mata kuliah yang dilaksanakan di semester 1 SELECT * FROM mata_kuliah WHERE sem = 1;
- Menampilkan data mata kuliah yang dilaksanakan selain semester 1 SELECT * FROM mata_kuliah WHERE sem <>1;
- Menampilkan data mata kuliah yang mengandung judul informatika SELECT * FROM mat_kul WHERE nama_kul LIKE '%informatika%'

```
mysql> select * from mata_kuliah where sem=1;
+-----+-----+-----+
| kode_kul | nama_kul | sks | sem |
+-----+-----+-----+
| IT0101 | Logika Inf | 3 | 1 |
| IT0102 | Studi Islam I | 2 | 1 |
| IT0103 | Kalkulus Inf | 3 | 1 |
+-----+-----+-----+
3 rows in set <0.00 sec>

mysql> select * from mata_kuliah where sem<>1;
+-----+-----+-----+
| kode_kul | nama_kul | sks | sem |
+-----+-----+-----+
| IT0301 | Sertifikasi | 0 | 3 |
| IT0401 | Basis Data | 3 | 4 |
| IT0801 | Tugas Akhir | 6 | 8 |
+-----+-----+-----+
3 rows in set <0.01 sec>

mysql> select * from mata_kuliah where nama_kul like '%Inf';
+-----+-----+-----+
| kode_kul | nama_kul | sks | sem |
+-----+-----+-----+
| IT0101 | Logika Inf | 3 | 1 |
| IT0103 | Kalkulus Inf | 3 | 1 |
+-----+-----+-----+
2 rows in set <0.00 sec>
```

Gambar 6.8 Tampil Data dengan Kondisi

- Menampilkan data mata kuliah yang mengandung judul informatika dan sksnya = 3 SELECT * FROM mat_kul WHERE nama_kul LIKE '%informatika%' AND sks = 3
- Menghitung jumlah data mata kuliah SELECT COUNT(*) from mat_kul;
- Menghitung sks yang paling sedikit, sks yang paling banyak dan rata-rata sks SELECT MIN (sks), MAX (sks), AVG (sks) from mat_kul;
- Menghitung total jumlah sks SELECT SUM (sks) from mat_kul;

```

mysql> select count(*) from mata_kuliah;
+-----+
| count(*) |
+-----+
|      6 |
+-----+
1 row in set (0.00 sec)

mysql> select min(sks), max(sks), avg(sks) from mata_kuliah;
+-----+-----+-----+
| min(sks) | max(sks) | avg(sks) |
+-----+-----+-----+
|      0   |      6   | 2.833333333333333 |
+-----+-----+-----+
1 row in set (0.00 sec)

mysql> select sum(sks) from mata_kuliah;
+-----+
| sum(sks) |
+-----+
|     17 |
+-----+
1 row in set (0.00 sec)

```

Gambar 6.9 Tampil Data dengan Kondisi SUM

3. Menerapkan Sub-Query

1. Buatlah dan Isikan tabel nilai seperti Gambar 10.1:

```
CREATE TABLE <nama_table> (<nama_kolom><tipedata>);
```

NIM	kode_kul	nilai
18102	IF0105	85
18102	IF0104	90
18451	IF0105	90
18451	IF0206	85
18321	IF0206	90
18321	IF0104	80

6 rows in set (0.001 sec)

Gambar 6.10 Nilai Mahasiswa

2. Menampilkan data nilai mahasiswa yang nilainya melebihi rata-rata nilai mata kuliah secara keseluruhan (AVG)

Langkahnya:

- 1) Membuat Query untuk mencari nilai rata-rata mata kuliah. Query ini akan digunakan sebagai acuan untuk mencari nilai yang memiliki nilai di atas rata rata, sehingga query dengan model seperti ini dinamakan Sub Query.
- 2) Dari Sub Query tersebut akan dimasukkan dalam Query utama (Main Query). Main Query ini untuk mencari nilai yang memiliki nilai di atas rata rata.
- 3) Lihat gambar 10. 2 berikut ini:

Sub Query: SELECT avg(nilai) from nilai;

```

MariaDB [akademik]> select avg(nilai) from nilai;
+-----+
| avg(nilai) |
+-----+
|    86.6667 |
+-----+
1 row in set (0.001 sec)

MariaDB [akademik]> 

```

Gambar 6.11 Sub Query

Main Query: SELECT NIM, nilai FROM nilai WHERE nilai > (SELECT avg(nilai) from nilai)

```
MariaDB [akademik]> select NIM, nilai from nilai where nilai > (select avg(nilai) from nilai);
+-----+-----+
| NIM | nilai |
+-----+-----+
| 18102 | 90 |
| 18451 | 90 |
| 18321 | 90 |
+-----+-----+
3 rows in set (0.005 sec)

MariaDB [akademik]>
```

Gambar 6.12 Main Query

3. Carilah data nilai yang nilainya sama dengan nilai terbesar (MAX)

Query membutuhkan Sub Query karena untuk dapat mencari data nilai yang diinginkan, maka nilai terbesar harus dicari terlebih dahulu.

Query ini bisa dilakukan dengan menggunakan ORDER BY dan LIMIT, tetapi hanya akan menghasilkan 1 baris saja. Bagaimana jika data yang sesuai dengan kriteria lebih dari 1 baris.

Langkahnya:

- Tampilkan Tabel Nilai

```
SELECT * FROM nilai;
```

```
MariaDB [akademik]> select * from nilai;
+-----+-----+-----+
| NIM | kode_kul | nilai |
+-----+-----+-----+
| 18102 | IF0105 | 85 |
| 18102 | IF0104 | 90 |
| 18451 | IF0105 | 90 |
| 18451 | IF0206 | 85 |
| 18321 | IF0206 | 90 |
| 18321 | IF0104 | 80 |
+-----+-----+-----+
6 rows in set (0.001 sec)

MariaDB [akademik]> -
```

Gambar 6.13 Tabel Nilai

- Tampilkan nilai maksimal dengan menggunakan Perintah MAX

```
SELECT NIM, nilai, kode_kul
```

```
FROM nilai WHERE nilai=(SELECT MAX(nilai) FROM nilai)
```

```
MariaDB [akademik]> select NIM, nilai, kode_kul
      -> from nilai where nilai=(select max(nilai) from nilai);
+-----+-----+-----+
| NIM | nilai | kode_kul |
+-----+-----+-----+
| 18102 | 90 | IF0104 |
| 18451 | 90 | IF0105 |
| 18321 | 90 | IF0206 |
+-----+-----+-----+
3 rows in set (0.001 sec)

MariaDB [akademik]>
```

Gambar 6.14 Tabel hasil Query tanpa Limit

- Gunakan Perintah ORDER BY dan LIMIT

Query di atas menampilkan data nilai lebih dari 1 baris ketika baris yang nilainya sama dengan nilai MAX(nilai) lebih dari 1 baris. Namun pada Query menggunakan ORDER BY



dan LIMIT akan menampilkan 1 baris saja karena ada penggunaan LIMIT yang dapat dilihat pada gambar 10.6.

Kekurangan dari SQL ini adalah ketika ada data nilai yang sama-sama memiliki nilai sama dengan MAX(nilai) lebih dari 1 baris namun tidak ditampilkan.

```
SELECT nim, nilai, kode_kul
FROM nilai ORDER BY nilai ASC LIMIT 1
```

```
MariaDB [akademik]> select NIM, nilai, kode_kul
      -> from nilai order by nilai asc limit 1;
+-----+-----+-----+
| NIM | nilai | kode_kul |
+-----+-----+-----+
| 18321 |    80 | IF0104 |
+-----+-----+-----+
1 row in set (0.001 sec)

MariaDB [akademik]> ■
```

Gambar 6.15 Tabel hasil Query dengan Limit

4. Menampilkan data mahasiswa yang mengambil mata kuliah ‘IF0104’ dengan IN

Query tersebut membutuhkan sub query karena harus melakukan perbandingan data mahasiswa dengan data mahasiswa yang ada di tabel mata kuliah. Berarti data mahasiswa yang ada di tabel mata kuliah harus dicari terlebih dahulu.

```
SELECT NIM, nama_mhs FROM mahasiswa
WHERE nim IN (SELECT nim FROM kuliah WHERE kode_kul = 'IF0104')
```

```
MariaDB [akademik]> select NIM, nama_mhs
      -> from mahasiswa
      -> where NIM in (select nim from nilai where kode_kul='IF0104');
+-----+-----+
| NIM | nama_mhs |
+-----+-----+
| 18102 | Rahma Nadia |
| 18321 | Fidia Fajri |
+-----+-----+
2 rows in set (0.001 sec)

MariaDB [akademik]> ■
```

Gambar 6.16 Tabel hasil Query perbandingan dua table dengan IN

Sub query bekerja untuk mencari data mahasiswa yang telah terdaftar di tabel mata kuliah (mengambil mata kuliah tersebut).

5. Menampilkan data mahasiswa yang tidak mengambil mata kuliah basis data dengan NOT IN

```
SELECT NIM, nama_mhs from mahasiswa
```

```
WHERE NIM NOT IN (SELECT NIM FROM nilai WHERE kode_kul = 'IF0104')
```



```
MariaDB [akademik]> select NIM, nama_mhs from mahasiswa
-> where NIM not in (select NIM from nilai where kode_kul = 'IF0104');
+-----+-----+
| NIM | nama_mhs |
+-----+-----+
| 18115 | Dian Sastro |
| 18210 | Aulia |
| 18324 | Mahardika |
| 18451 | N Fatayah |
+-----+-----+
4 rows in set (0.087 sec)

MariaDB [akademik]>
```

Gambar 6.17 Tabel hasil Query Perbandingan dua Tabel dengan NOT IN

4. Menerapkan Sub String

Terlebih dahulu buatlah tabel prodi berikut:

Kode_prodi	Program_Studi
10	Teknik Kimia
11	Teknik Elektro
12	Informatika
13	Teknik Industri
14	Teknologi Pangan

Tambahkan tabel mahasiswa dengan data berikut:

2403101204	Elisa Wulandari
2403101224	Citra Lestari
2403121211	Fajar Nugroho
2403121212	Gita Anggraini
2403121213	Hendra Wijaya
2403121214	Intan Maharani
2403121215	Joko Santoso
2403121216	Kiki Permatasari
2403121217	Lukman Hakim
2403121218	Maya Salsabila
2403121219	Nino Saputro
2403121220	Oktavia Siregar

2403121234	Alya Ramadhani
2403121235	Bagas Saputra
2403141214	Doni Prasetyo

Misalnya kita ingin tahu Program Studi (Prodi) untuk NIM 2403121234, berarti kita harus membaca 2 digit ketiga atau digit ke 5 dan 6 yaitu kode Prodi 02. Berdasarkan tabel berikut, kode 02 adalah Informatika.

Untuk itu, kita ambil seluruh kode prodi dari kolom nim

```
SELECT nama, SUBSTRING(nim, 5, 2) AS kode_prodi
FROM mahasiswa;
```

Hasilnya:

Elisa Wulandari	10
Citra Lestari	10
Fajar Nugroho	12
Gita Anggraini	12
Hendra Wijaya	12
Intan Maharani	12
Joko Santoso	12
Kiki Permatasari	12
Lukman Hakim	12
Maya Salsabila	12
Nino Saputro	12
Oktavia Siregar	12
Alya Ramadhani	12
Bagas Saputra	12
Doni Prasetyo	14

```
SELECT mhs.nama, SUBSTRING(mhs.nim, 5, 2) AS kode_prodi, ps.nama_prodi
FROM program_studi ps
join mahasiswa mhs on SUBSTRING(mhs.nim, 5, 2) = ps.id_prodi;
```

Hasilnya:

Elisa Wulandari	10	Teknik Kimia
Citra Lestari	10	Teknik Kimia
Fajar Nugroho	12	Informatika

Gita Anggraini	12	Informatika
Hendra Wijaya	12	Informatika
Intan Maharani	12	Informatika
Joko Santoso	12	Informatika
Kiki Permatasari	12	Informatika
Lukman Hakim	12	Informatika
Maya Salsabila	12	Informatika
Nino Saputro	12	Informatika
Oktavia Siregar	12	Informatika
Alya Ramadhani	12	Informatika
Bagas Saputra	12	Informatika
Doni Prasetyo	14	Teknologi Pangan

5. Operasi Tanggal

1. DATE()

Fungsi: Mengambil hanya bagian **tanggal** dari sebuah nilai **DATETIME**.

SQL: `SELECT DATE('2024-04-12 14:30:00') AS hanya_tanggal;`

Output: 2024-04-12

2. YEAR()

Fungsi: Mengambil bagian **tahun** dari tanggal.

SQL: `SELECT YEAR('2022-10-05') AS tahun;`

Output: 2022

3. MONTH()

Fungsi: Mengambil bagian **bulan** dari tanggal.

SQL: `SELECT MONTH('2022-10-05') AS bulan;`

Output: 10

4. DAY()

Fungsi: Mengambil bagian **hari** dari tanggal.

SQL: `SELECT DAY('2022-10-05') AS hari;`

Output: 5

5. CURDATE()

Fungsi: Mengambil **tanggal hari ini** (tanpa jam).

SQL: `SELECT CURDATE() AS tanggal_hari_ini;`

Output: 2025-04-14

6. MONTH(CURDATE())

Fungsi: Mengambil bulan dari tanggal hari ini.

SQL: `SELECT MONTH(CURDATE()) AS bulan_ini;`

Output: 4

7. YEAR(CURDATE())

Fungsi: Mengambil tahun dari tanggal hari ini.

SQL: SELECT YEAR(CURDATE()) AS tahun_ini;

Output: 2025

Contoh

Buat dan isi tabel **pic_supplier** dengan kolom **name** dan **assigned_date**

Nama	assigned_date
Ami Yulia Yuniar Tbk	2003-12-14
Ika Lailasari	1996-04-21
Salimah Yolanda	1996-09-25
Yunita Uyainah	1974-01-12
Putri Purnawati	2014-11-16
Upik Wasita	2012-04-04
Oskar Thamrin	1994-10-13
Baktianto Narpati	1979-03-10
Puput Ida Melani (Persero) Tbk	2019-09-18
Kania Padma Uyainah	1982-10-08

Kita bisa mengambil tanggal, bulan, tahun serta lama/usia PIC ditugaskan. Berikut querynya:

SELECT

```
name, assigned_date,
YEAR(assigned_date) AS tahun_penugasan,
MONTH(assigned_date) AS bulan_penugasan,
MONTHNAME(assigned_date) AS nama_bulan_penugasan,
DAY(assigned_date) AS tanggal_penugasan,
TIMESTAMPDIFF(YEAR, assigned_date, CURDATE()) AS lama_penugasan
FROM supplier_pic;
```

Name	assigned_date	tahun_penugasan	bulan_penugasan	nama_bulan_penugasan	tanggal_penugasan	lama_penugasan
Ami Yulia Yuniar Tbk	2003-12-14	2003	12	December	14	21

Ika Lailasari	1996-04-21	1996	4	April	21	28
Salimah Yolanda	1996-09-25	1996	9	September	25	28
Yunita Uyaina h	1974-01-12	1974	1	January	12	51
Putri Purnawati	2014-11-16	2014	11	November	16	10
Upik Wasita	2012-04-04	2012	4	April	4	13
Oskar Thamrin	1994-10-13	1994	10	October	13	30
Baktianto Narpati	1979-03-10	1979	3	March	10	46
Puput Ida Melani (Persero) Tbk	2019-09-18	2019	9	September	18	5
Kania Padma Uyaina h	1982-10-08	1982	10	October	8	42

Aturan Penilaian (Total Skor: 100):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL 03-KU03	CPMK 3.3.3	1. Praktekkanlah langkah Langkah DML 1-5	Jawaban hasil praktek (screenshot)	50

2.	CPL 03-KU03	CPMK 3.3.3	2. Praktekanlah langkah Fungsi Agregasi termasuk sub query	Jawaban hasil praktek (Screenshot)	50
----	-------------	------------	--	------------------------------------	----

6.7. POST TEST

Jawablah pertanyaan-pertanyaan di bawah ini dimana setiap pertanyaan memiliki bobot nilai yang berbeda, dengan total nilai 100

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL 03-KU03	CPMK 3.3.3	1. Pada database yang telah dirancang, inputkan minimal 5 buah data pada masing-masing tabel . - DML 2. Tampilkan semua data yang telah diinputkan 3. Tampilkan data yang telah diinputkan sesuai dengan kondisi tertentu (gunakan perintah (WHERE)) 4. Perbarui salah satu data pada setiap tabel	Jawaban hasil praktek (Screenshot)	50
2.	CPL 03-KU03	CPMK 3.3.3	5. Jelaskan pengertian dari insert data , update data dan delete data? 6. Tuliskan syntax dari insert, update dan delete! 7. Tuliskan syntax untuk memasukan data sebagai berikut: (Nur Fatayah, 2000018451, nur18451@webmail.uad.ac.id)	Jawaban hasil	50
					Total Nilai
					100

6.8. HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL 03-KU03	CPMK 3.3.3	20%		
2.	Praktik	CPL 03-KU03	CPMK 3.3.3	30%		
3.	Post-Test	CPL 03-KU03	CPMK 3.3.3	50%		
Total Nilai						

LEMBAR JAWABAN PRE-TEST DAN POST-TEST PRAKTIKUM

Nama : NIM :	Asisten: Paraf Asisten:	Tanggal: Nilai:
-------------------------------	--	----------------------------------

PRAKTIKUM 7: RELASI TABEL DENGAN JOIN

Pertemuan ke : 7

Total Alokasi Waktu : 180 menit

- Materi : 30 menit
- Pre-Test : 30 menit
- Praktikum : 60 menit
- Post-Test : 60 menit

Total Bobot Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 30 %
- Post-Test : 50 %

Pemenuhan CPL dan CPMK:

CPL 03-KU03	Mampu menerapkan konsep teoritis bidang area Informatika terkait matematika dasar dan ilmu komputer untuk memodelkan masalah dan meningkatkan produktivitas
CPMK 3.3.3.	Mengimplementasikan query basis data dan lingkungannya

7.1. DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu:

1. Menerapkan konsep teoritis bidang area Informatika terkait matematika dasar dan ilmu komputer untuk memodelkan masalah dan meningkatkan produktivitas dengan menggunakan asesmen Pre Test untuk materi terkait JOIN Tabel
2. Berpikir logis, kritis, sistematis dan inovatif, dan mampu mengambil keputusan secara tepat di bidang keahliannya melalui asesmen Pre test.
3. Mengkaji/menganalisis implikasi pengembangan atau implementasi ilmu pengetahuan teknologi, menyusun deskripsi saintifik hasil kajian untuk pemecahan masalah terkait JOIN Tabel dengan asesmen praktik dan post test.
4. Memilih, membuat dan menerapkan teknik, sumber daya, penggunaan perangkat teknik modern dan implementasi teknologi informasi untuk memecahkan masalah terkait JOIN Tabel dengan Asesmen Post Test dan Praktik.
5. Merancang dan mengimplementasikan algoritma/metode dalam mengidentifikasi dan memecahkan masalah yang melibatkan perangkat lunak dan pemikiran komputasi.

7.2. INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:

CPL 03-KU03	CPMK 3.3.3	1. Kemampuan mahasiswa meningkat dalam memahami konsep data dan informasi serta dapat berpikir logis yang diterapkan dengan cara mempraktekkan konsep JOIN Tabel
-------------	------------	--

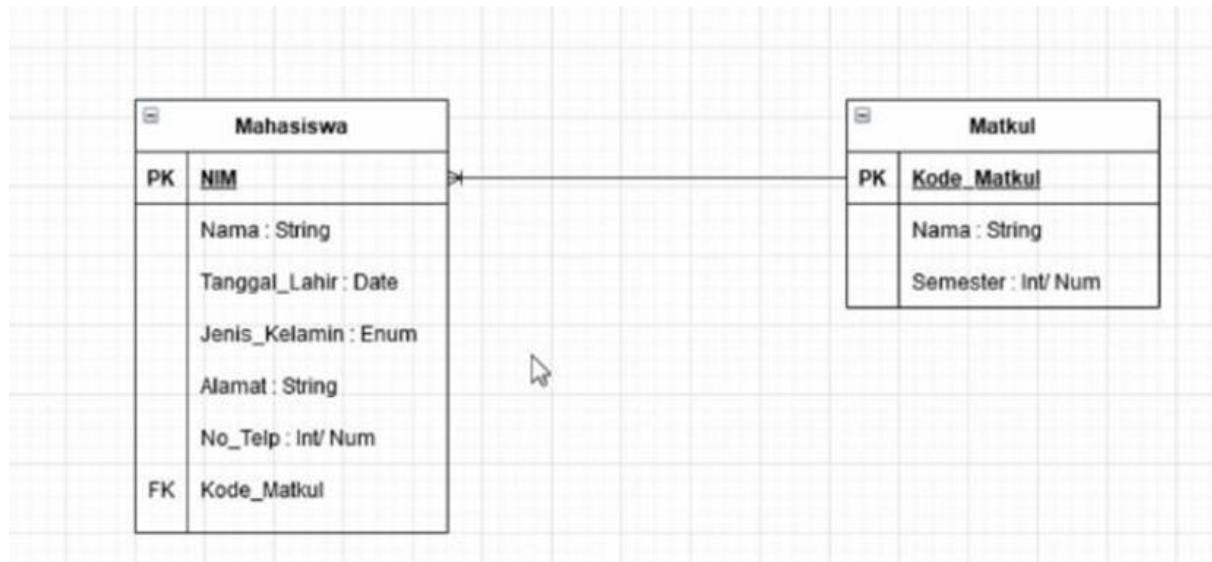
CPL 03- KU03	CPMK 3.3.3	2. Kemampuan mahasiswa meningkat dalam menganalisa, memilih dan mengimplementasikan masalah terkait JOIN Tabel dengan menggunakan tool pemodelan yang tersedia
--------------------	---------------	--

7.3. TEORI PENDUKUNG

Mekanisme join dipergunakan untuk mencari data dari beberapa tabel berdasarkan hubungan logis tabel-tabel tersebut. Macam-macam Join, adalah:

1. Inner Join merupakan himpunan dalam yaitu hasil gabungan dari dua buah tabel yang saling berelasi untuk semua record yang berpasangan.
2. Full Outer Join mengembalikan semua baris dari kedua tabel.
3. Left Outer Join menghasilkan semua baris tabel di sebelah kiri pernyataan, dan baris-baris bersesuaian dari tabel sebelah kanan pernyataan.
4. Right Outer Join menghasilkan semua baris tabel di sebelah kiri pernyataan, dan baris-baris bersesuaian dari tabel sebelah kanan pernyataan.
5. Union dipergunakan untuk menggabungkan dua buah operasi query ke dalam satu buah cursor

Berikut contoh relasi antar entitas mahasiswa dan mata kuliah.



7.4. HARDWARE DAN SOFTWARE

Hardware dan software yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. DB Maria / Mysql.
3. Browser.

7.5. PRE-TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL 03- KU03	CPMK 3.3.3	1. Jelaskan apa yang dimaksud dengan relasi database!	25



	3			
2.	CPL CPL 03- KU0 3	CPMK 3.3.3	2. Jelaskan fungsi JOIN pada database!	25
3.	CPL 03- KU03	CPMK 3.3.3	3. Sebut dan jelaskan macam macam JOIN!	25
4.	CPL 03- KU03	CPMK 3.3.3	4. Berikan satu contoh relasi table untuk Inner JOIN, Full Outer JOIN, Left Outer JOIN, Right Outer JOIN, Union ?	25

7.6. LANGKAH PRAKTIKUM

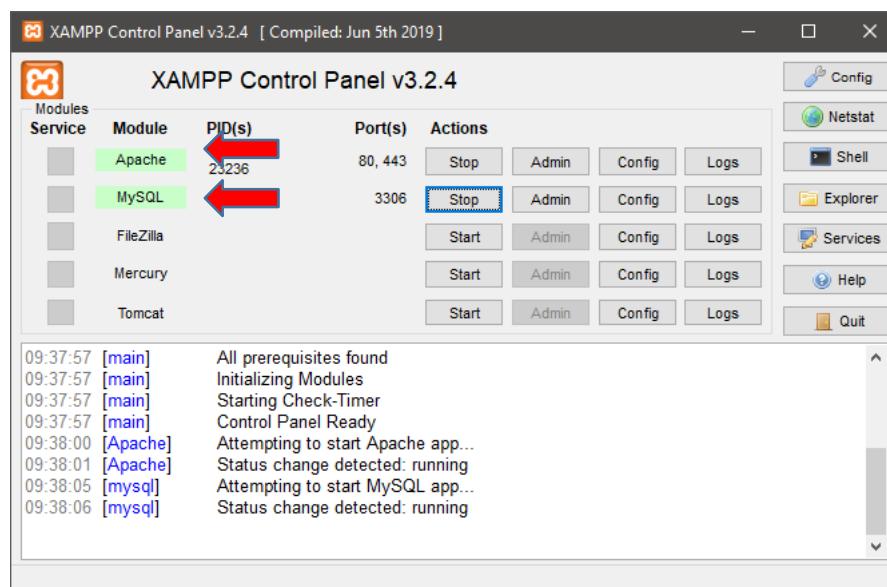
Langkah praktikum berisi tahapan secara rinci bagaimana praktikum dijalankan dan apa hasil yang harus dicapai dari setiap langkah.

Tahap I Pembuatan relasi

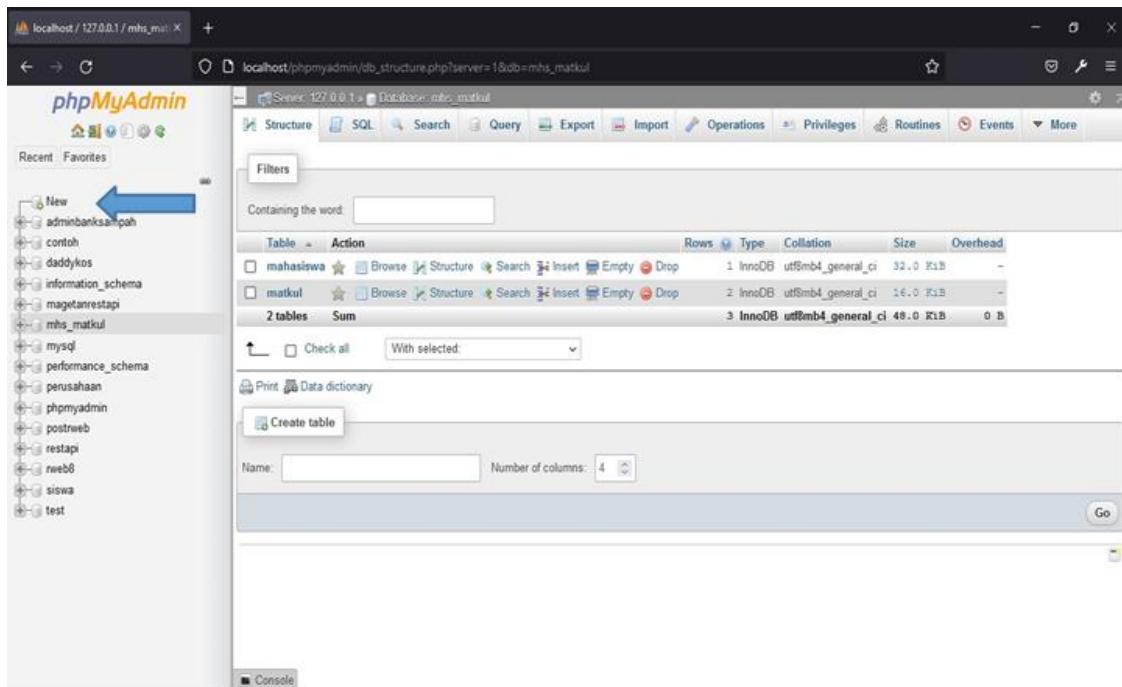
1. Pembuatan Relasi pada Mysql. Studi kasus Mahasiswa

Langkah- Langkah :

- Langkah pertama yaitu melakukan pembuatan database baru pada Mysql. Aktifkan terlebih dahulu XAMPP => Centang pada Mysql => Buka <http://localhost/phpmyadmin/>

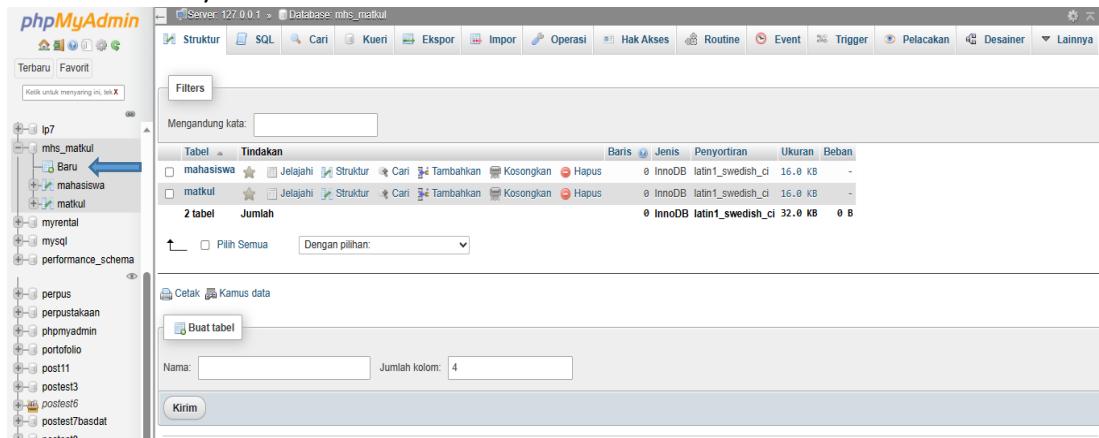


Gambar 7.1 Tampilan XAMPP

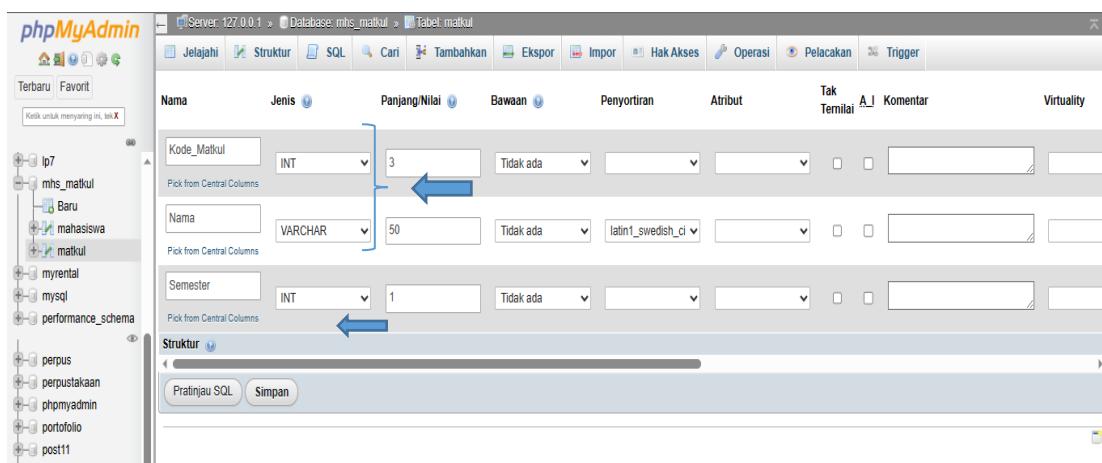


Gambar 7.2 Tampilan PHP MyAdmin

b. Langkah selanjutnya yaitu membuat tabel. Buatlah tabel sesuai dengan rancang bangun sebelumnya.

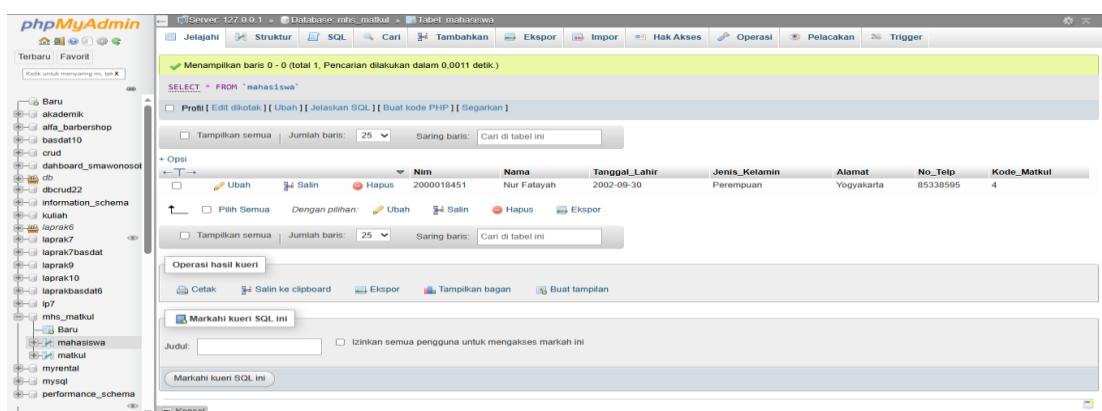


Gambar 7.3 Membuat Tabel

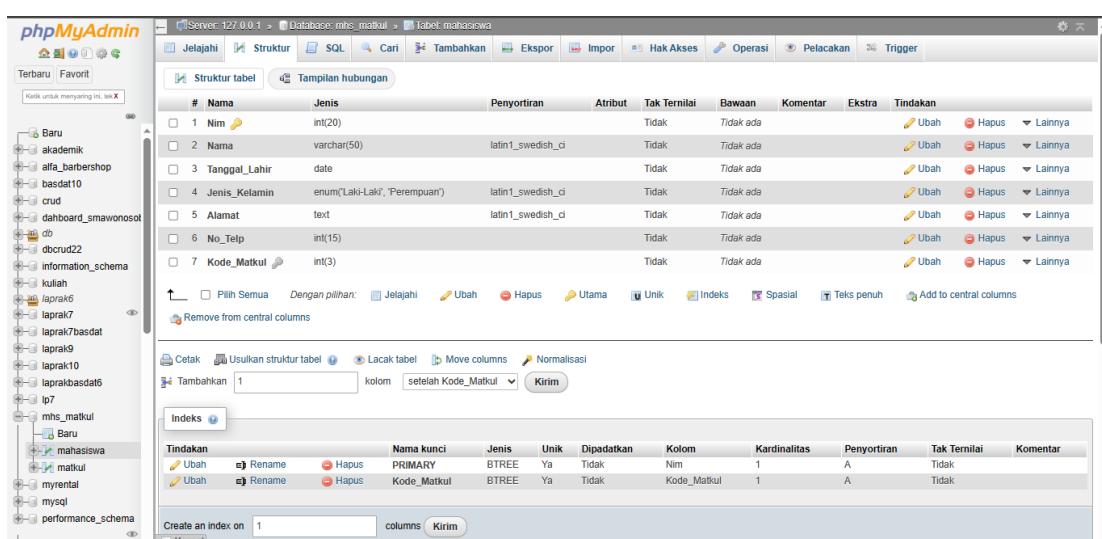


Gambar 7.4 Mengisi atribut

- c. Setelah semua tabel terbuat. Langkah selanjutnya yaitu melakukan relasi antar tabel. Pilih tabel Mahasiswa yang sebelumnya telah dibuat, kemudian Klik pada Structure, lalu klik Relation View dan setting relasi tabel sesuai struktur awal kita tadi.



Gambar 7.5 Tampilan Browse Tabel Mahasiswa



Gambar 7.6 Tampilan Struktur Tabel Mahasiswa

Isi Sesuai Relasi Struktur Awal

Gambar 7.7 Tampilan Relation View/Tampilan Hubungan

Gambar 7.8 Foreign Key ditambahkan

- d. Selanjutnya, isikan terlebih dahulu data yang ada di tabel Matakuliah. Setelah itu isilah data di tabel Mahasiswa.

Gambar 7.9 Tampilan Browse Matkul

Server: 127.0.0.1 > Database: mhs_matkul > Tabel: matkul

Kolom	Jenis	Fungsi	Tak Termilai	Nilai
Kode_Matkul	int(3)		<input checked="" type="checkbox"/>	3
Nama	varchar(50)			Basis Data
Semester	int(1)		<input checked="" type="checkbox"/>	3

Kolom	Jenis	Fungsi	Tak Termilai	Nilai
Kode_Matkul	int(3)		<input checked="" type="checkbox"/>	4
Nama	varchar(50)			KDJK
Semester	int(1)		<input checked="" type="checkbox"/>	4

Simpan		selanjutnya	kembali	Pratinjau		Kirim
--------	--	-------------	---------	-----------	--	-------

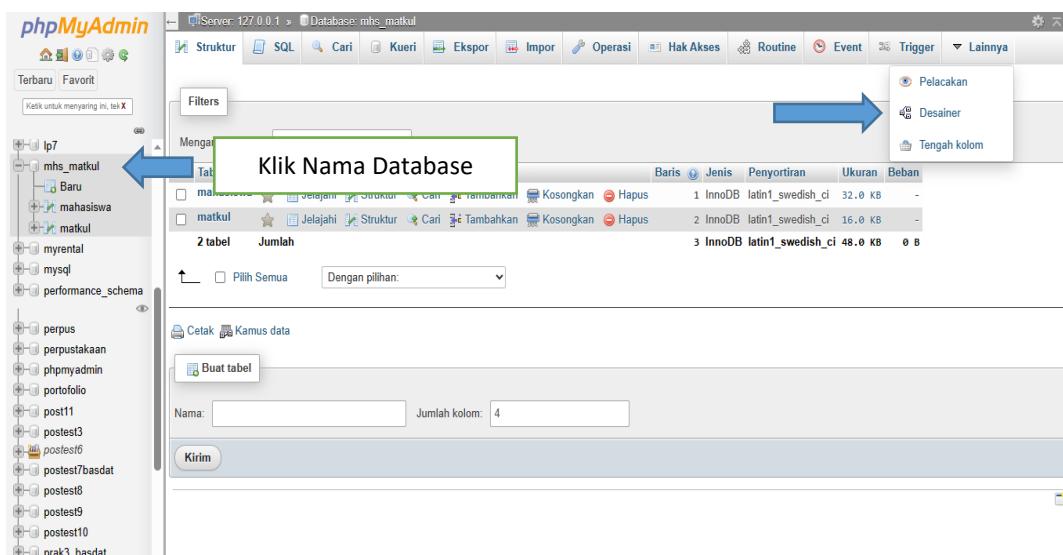
Gambar 7.10 Tampilan Insert Data Matkul

Server: 127.0.0.1 > Database: mhs_matkul > Tabel: mahasiswa

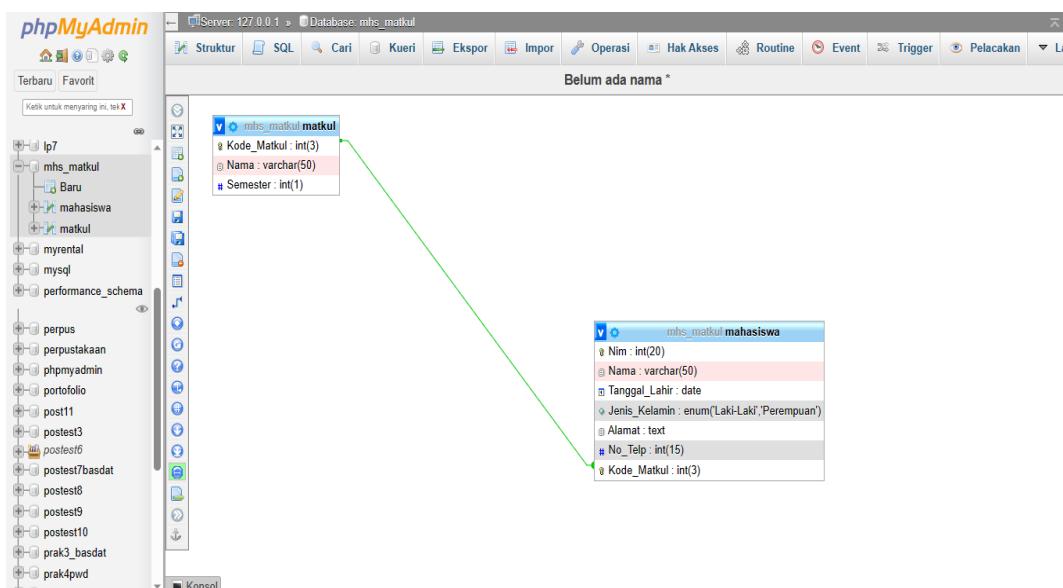
Kolom	Jenis	Fungsi	Tak Termilai	Nilai
Nim	int(20)			2000018451
Nama	varchar(50)			Nur Fatayah
Tanggal_Lahir	date			2002-09-30
Jenis_Kelamin	enum			Perempuan
Alamat	text			Yogyakarta
No_Telp	int(15)			3 - Basis Data 4 - KDJK
Kode_Matkul	int(3)			4 - KDJK

Gambar 7.11 Tampilan Insert Data Mahasiswa

- e. Maka anda dengan ini telah menyelesaikan Relasi antar tabel dan juga telah mengisi data pada setiap tabelnya. Untuk lebih melihat hasil relasi kita, bisa melihat di bagian Designer pada bagian DB.



Gambar 7.12 Tampilan Untuk Melihat Desainer



Gambar 7.13 Tampilan Designer Pada DB

Tahap II Operasi relasi menggunakan join

1. Buatlah tabel mahasiswa, nilai, dan mata_kuliah seperti pada Gambar 9.1.

```
MariaDB [akademik]> select * from mahasiswa;
+----+----+-----+
| NIM | nama_mhs | alamat_mhs |
+----+----+-----+
| 18102 | Rahma Nadia | Gunung Kidul |
| 18115 | Dian Sastro | Jakarta |
| 18210 | Aulia | Kulon Progo |
| 18321 | Fidia Fajri | Sleman |
| 18324 | Mahardika | Sleman |
| 18451 | N Fatayah | Yogyakarta |
+----+----+-----+
6 rows in set (0.065 sec)
```

```
MariaDB [akademik]> select * from nilai;
+-----+-----+-----+
| NIM | kode_kul | nilai |
+-----+-----+-----+
| 18102 | IF0105 | 85 |
| 18102 | IF0104 | 90 |
| 18451 | IF0105 | 90 |
| 18451 | IF0206 | 85 |
| 18321 | IF0206 | 90 |
| 18321 | IF0104 | 80 |
+-----+-----+-----+
6 rows in set (0.001 sec)
```

```
MariaDB [akademik]> select * from mata_kuliah;
+-----+-----+-----+-----+
| kode_kul | nama_matkul | sks | sem |
+-----+-----+-----+-----+
| IF0104 | Basis Data | 4 | 3 |
| IF0105 | Pemrograman Web Dinamis | 2 | 5 |
| IF0206 | Statistika Informatika | 4 | 3 |
+-----+-----+-----+-----+
3 rows in set (0.001 sec)
```

Gambar 7.14 Tabel Mahasiswa, Nilai, Mata kuliah

2. Lakukan Inner Join

Menampilkan nama mahasiswa, kode mata kuliah berikut nilai yang diperoleh Dengan perintah relasi atribut antar table

Select mahasiswa.NIM, mahasiswa.nama_mhs, nilai.kode_kul, nilai.nilai from mahasiswa, nilai
Where mahasiswa.nim = nilai.nim;

```
MariaDB [akademik]> select mahasiswa.NIM, mahasiswa.nama_mhs, nilai.kode_kul, nilai.nilai from mahasiswa, nilai where mahasiswa.NIM = nilai.NIM;
+-----+-----+-----+-----+
| NIM | nama_mhs | kode_kul | nilai |
+-----+-----+-----+-----+
| 18102 | Rahma Nadia | IF0105 | 85 |
| 18102 | Rahma Nadia | IF0104 | 90 |
| 18451 | N Fatayah | IF0105 | 90 |
| 18451 | N Fatayah | IF0206 | 85 |
| 18321 | Fidia Fajri | IF0206 | 90 |
| 18321 | Fidia Fajri | IF0104 | 80 |
+-----+-----+-----+-----+
6 rows in set (0.042 sec)

MariaDB [akademik]> -
```

Gambar 7.15 Perintah Inner Join 1

Dengan perintah inner join

SELECT mahasiswa.NIM, mahasiswa.nama_mhs, nilai.kode_kul, nilai.nilai FROM mahasiswa
INNER JOIN nilai ON mahasiswa.NIM = nilai.NIM;

```
MariaDB [akademik]> select mahasiswa.NIM, mahasiswa.nama_mhs, nilai.kode_kul, nilai.nilai
    -> from mahasiswa inner join nilai
    -> on mahasiswa.NIM = nilai.NIM;
+-----+-----+-----+-----+
| NIM | nama_mhs | kode_kul | nilai |
+-----+-----+-----+-----+
| 18102 | Rahma Nadia | IF0105 | 85 |
| 18102 | Rahma Nadia | IF0104 | 90 |
| 18451 | N Fatayah | IF0105 | 90 |
| 18451 | N Fatayah | IF0206 | 85 |
| 18321 | Fidia Fajri | IF0206 | 90 |
| 18321 | Fidia Fajri | IF0104 | 80 |
+-----+-----+-----+-----+
6 rows in set (0.001 sec)

MariaDB [akademik]> -
```

Gambar 7.16 Perintah Inner Join 2

Dua gambar di atas menghasilkan hasil yang sama. Terlihat bahwa pada tabel mahasiswa dan mata_kuliah, nim adalah atribut yang menghubungkan ke dua tabel tersebut. Pada perintah di

atas hanya record yang berpasangan yang dimunculkan. Pada tabel mahasiswa terdapat record dengan nim ‘18115’ dan ‘18210’ yang tidak terdapat pada tabel nilai, sehingga tidak dimunculkan.

3. Lakukan Left Join

Menampilkan semua data mahasiswa yang ada di tabel mahasiswa berikut data nilai mahasiswa

```
SELECT mahasiswa.NIM, mahasiswa.nama_mhs, nilai.kode_kul, nilai.nilai
```

```
FROM mahasiswa LEFT JOIN nilai on mahasiswa.NIM = nilai.NIM;
```

```
MariaDB [akademik]> select mahasiswa.NIM, mahasiswa.nama_mhs, nilai.kode_kul, nilai.nilai
    -> from mahasiswa left join nilai on mahasiswa.NIM = nilai.NIM;
+-----+-----+-----+
| NIM | nama_mhs | kode_kul | nilai |
+-----+-----+-----+
| 18102 | Rahma Nadia | IF0105 | 85 |
| 18102 | Rahma Nadia | IF0104 | 90 |
| 18451 | N Fatayah | IF0105 | 90 |
| 18451 | N Fatayah | IF0206 | 85 |
| 18321 | Fidia Fajri | IF0206 | 90 |
| 18321 | Fidia Fajri | IF0104 | 80 |
| 18115 | Dian Sastro | NULL | NULL |
| 18210 | Aulia | NULL | NULL |
| 18324 | Mahardika | NULL | NULL |
+-----+-----+-----+
9 rows in set (0.041 sec)

MariaDB [akademik]>
```

Gambar 7.17 Perintah Left Join

Terlihat bahwa data tabel sebelah kanan akan diisi dengan NULL karena mahasiswa yang bersangkutan tidak ada nilainya, sementara data dari tabel sebelah kiri akan ditampilkan seluruhnya.

4. Lakukan Right Join

Menampilkan semua data nilai yang ada di tabel nilai berikut nama mahasiswa.

```
SELECT mahasiswa.NIM, mahasiswa.nama_mhs, nilai.kode_kul, nilai.nilai
```

```
FROM mahasiswa RIGHT JOIN nilai on mahasiswa.NIM = nilai.NIM;
```

```
MariaDB [akademik]> select mahasiswa.NIM, mahasiswa.nama_mhs, nilai.kode_kul, nilai.nilai
    -> from mahasiswa right join nilai on mahasiswa.NIM = nilai.NIM;
+-----+-----+-----+
| NIM | nama_mhs | kode_kul | nilai |
+-----+-----+-----+
| 18102 | Rahma Nadia | IF0105 | 85 |
| 18102 | Rahma Nadia | IF0104 | 90 |
| 18451 | N Fatayah | IF0105 | 90 |
| 18451 | N Fatayah | IF0206 | 85 |
| 18321 | Fidia Fajri | IF0206 | 90 |
| 18321 | Fidia Fajri | IF0104 | 80 |
+-----+-----+-----+
6 rows in set (0.001 sec)

MariaDB [akademik]> -
```

Gambar 7.18 Perintah Right Join

Akan terlihat bahwa data dari tabel sebelah kanan yaitu tabel nilai akan ditampilkan seluruhnya.

5. Lakukan Union

```
SELECT * FROM mahasiswa UNION SELECT * FROM nilai;
```

Perintah tersebut akan menghasilkan table yang berisi penggabungan antara table mahasiswa dan nilai seperti pada gambar 9.6.

```
MariaDB [akademik]> select * from mahasiswa union select * from nilai;
+-----+-----+-----+
| NIM | nama_mhs | alamat_mhs |
+-----+-----+-----+
| 18102 | Rahma Nadia | Gunung Kidul |
| 18115 | Dian Sastro | Jakarta |
| 18210 | Aulia | Kulon Progo |
| 18321 | Fidia Fajri | Sleman |
| 18324 | Mahardika | Sleman |
| 18451 | N Fatayah | Yogyakarta |
| 18102 | IF0105 | 85 |
| 18102 | IF0104 | 90 |
| 18451 | IF0105 | 90 |
| 18451 | IF0206 | 85 |
| 18321 | IF0206 | 90 |
| 18321 | IF0104 | 80 |
+-----+-----+-----+
12 rows in set (0.001 sec)

MariaDB [akademik]> -
```

Gambar 7.19 Perintah Union

Aturan Penilaian (Total Skor: 100):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL 03-KU03	CPMK 3.3.3	Praktekkan Langkah 1 membuat tabel	Jawaban hasil praktek (Screenshot)	30
2.	CPL 03-KU03	CPMK 3.3.3	Praktekkan Langkah 2-5 melakukan inner join, Right join, left join, Union	Jawaban hasil praktek (Screenshot)	70

7.7. POST TESTJawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL 03-KU03	CPMK 3.3.3	1. Buatlah sebuah database (bebas sesuai keinginan) 2. Database terdiri dari 2 buah tabel (nama dan jenis tabel bebas sesuai dengan database yang dirancang) 3. Isilah data pada masing masing table minimal 5 data 4. Gunakan innerjoin, rightjoin, leftjoin, sesuai dengan table yang telah dibuat tersebut!	100

7.8. HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL 03-KU03	CPMK 3.3.3	20%		

2.	Praktik	CPL 03-KU03	CPMK 3.3.3	30%		
3.	Post-Test	CPL 03-KU03	CPMK 3.3.3	50%		
Total Nilai						

LEMBAR JAWABAN PRE-TEST DAN POST-TEST PRAKTIKUM

Nama : NIM :	Asisten: Paraf Asisten:	Tanggal: Nilai:
-------------------------------	--	----------------------------------

PRAKTIKUM 8: TRIGGER, KEAMANAN DATA, DAN STORED PROCEDURE

Pertemuan ke : 8

Total Alokasi Waktu : 180 menit

- Materi : 30 menit
- Pre-Test : 30 menit
- Praktikum : 60 menit
- Post-Test : 60 menit

Total Bobot Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 30 %
- Post-Test : 50 %

Pemenuhan CPL dan CPMK:

CPL 03-KU3	Mempunyai kemampuan dalam mendefinisikan kebutuhan pengguna atau pasar terhadap kinerja (menganalisis, mengevaluasi dan mengembangkan) algoritma/metode berbasis komputer untuk mengambil keputusan secara tepat dalam konteks penyelesaian masalah di bidang Rekayasa Perangkat Lunak serta Data dan Sistem Cerdas maupun bidang lainnya, berdasarkan hasil analisis informasi dan data.
CPMK 3.3.3	Mengimplementasikan query basis data dan lingkungannya
SUB CPMK 3.3.3.2	Mengimplementasikan query TRIGGER, Kemanan Data, dan STORE PROCEDURE

8.1. DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu memahami:

1. Menerapkan konsep teoritis bidang area Informatika terkait matematika dasar dan ilmu komputer untuk memodelkan masalah dan meningkatkan produktivitas dengan menggunakan asesmen Pre Test untuk materi TRIGGER, Kemanan Data, dan STORE PROCEDURE
2. Berpikir logis, kritis, sistematis dan inovatif, dan mampu mengambil keputusan secara tepat di bidang keahliannya melalui asesmen Pre test untuk materi TRIGGER, Kemanan Data dan STORE PROCEDURE
3. Mengkaji/menganalisis implikasi pengembangan atau implementasi ilmu pengetahuan teknologi, menyusun deskripsi saintifik hasil kajian untuk pemecahan masalah terkait TRIGGER dengan asesmen praktik dan post test untuk materi TRIGGER, Kemanan Data dan STORE PROCEDURE
4. Memilih, membuat dan menerapkan teknik, sumber daya, penggunaan perangkat teknik modern dan implementasi teknologi informasi untuk memecahkan masalah terkait TRIGGER dengan Asesmen Post Test dan Praktik untuk materi TRIGGER, Kemanan Data dan STORE PROCEDURE
5. Merancang dan mengimplementasikan algoritma/metode dalam mengidentifikasi dan memecahkan masalah yang melibatkan perangkat lunak dan pemikiran komputasi.

8.2. INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:

CPL 03-KU03	CPMK 3.3.3	Sub-CPMK 3.3.3.2	Kemampuan mahasiswa meningkat dalam Mengimplementasikan query basis data dan lingkungannya yang diterapkan dengan cara mempraktekkan konsep TRIGGER, Kemanan Data dan STORE PROCEDURE
-------------	------------	------------------	---

8.3. TEORI PENDUKUNG

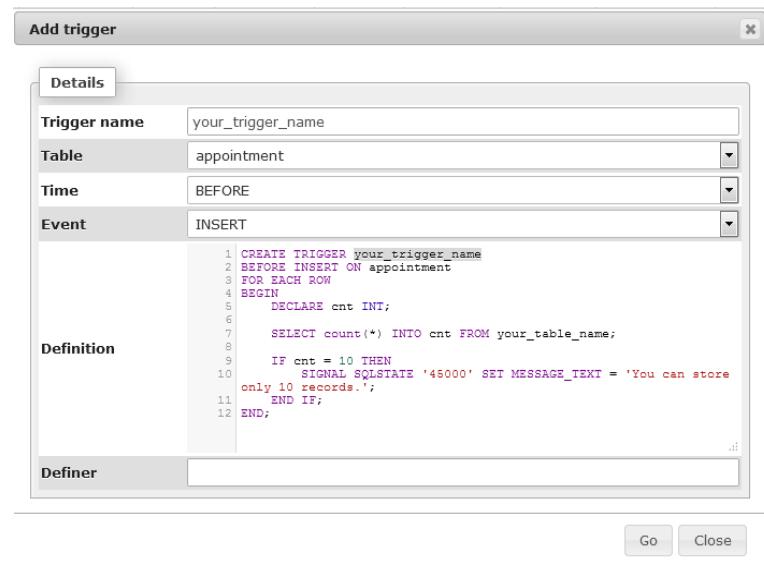
1. TRIGGER

Trigger dapat didefinisikan sebagai himpunan kode (prosedural) yang dieksekusi secara otomatis sebagai respon atas suatu kejadian berkaitan dengan tabel basis data. Kejadian (event) yang dapat membangkitkan Trigger umumnya berupa pernyataan INSERT, UPDATE, dan DELETE. Berdasarkan ruang lingkupnya, Trigger diklasifikasikan menjadi dua jenis: row trigger dan statement Trigger. Trigger baris (row) mendefinisikan aksi untuk setiap baris tabel; trigger statement (pernyataan) hanya berlaku untuk setiap pernyataan INSERT, UPDATE, atau DELETE.

Dari sisi yang lain, Trigger dapat dibedakan menjadi beberapa jenis; namun umumnya ada dua jenis: TRIGGER BEFORE dan AFTER. Sesuai penamaannya, jenis-jenis ini merepresentasikan waktu eksekusi TRIGGER — misalnya sebelum ataupun sesudah pernyataan yang berkorespondensi. Adakalanya trigger dipandang sebagai bentuk spesifik dari stored procedure (terkait pendefinisan body). Bagaimanapun, TRIGGER akan dipanggil (secara otomatis) ketika event terjadi, sedangkan stored procedure harus dipanggil secara eksplisit. Ada beberapa event yang dapat digunakan untuk melakukan eksekusi TRIGGER yaitu:

- 1) BEFORE INSERT – dijalankan ketika data dimasukkan ke dalam table.
- 2) AFTER INSERT – dijalankan setelah data masuk ke dalam table.
- 3) BEFORE UPDATE – dijalankan sebelum proses update data.
- 4) AFTER UPDATE – dijalankan setelah proses update data.
- 5) BEFORE DELETE – dijalankan sebelum proses delete data.
- 6) AFTER DELETE – dijalankan setelah proses delete data.

MySQL mendukung fitur TRIGGER termasuk juga STORED PROCEDURE dan VIEW sejak versi 5.0.2. Sebagaimana objek-objek lainnya, TRIGGER diciptakan menggunakan pernyataan CREATE. Sintaks pendefinisan trigger diperlihatkan sebagai berikut:



MySQL tidak mengizinkan multiple trigger dengan waktu aksi dan event sama per tabel. Misalkan di tabel A sudah didefinisikan trigger AFTER INSERT, maka kita tidak boleh mendefinisikan trigger AFTER INSERT lagi; namun AFTER EDIT, AFTER DELETE, atau BEFORE (INSERT, EDIT, dan DELETE) bisa diterima.

2. Keamanan Database

Database merupakan sesuatu yang sangat penting dan harus dijaga keberadaannya. Pada bagian ini akan membahas bagaimana agar keamanan database tetap terjaga untuk mengantisipasi hal-hal yang tidak diinginkan.

a) Pentingnya Keamanan

Sebagai seorang DBA, setidaknya Anda sudah memahami seperti apa database itu, dan sejauh mana keamanan yang diperlukan. Juga yang perlu Anda ketahui, kebanyakan software database yang ada di pasaran belum memiliki security secara khusus. Anda dapat mencarinya di internet, baik itu yang komersil maupun yang didistribusikan secara gratis.

b) Mengamankan Database dari Crack

Salah satu cara terbaik untuk mengamankan database dari gangguan crack yaitu menanggulangi adanya penyerang atau cracker dengan mengamankan database sistem mysql. Diantaranya yaitu:

1) Backup Database Sistem

Data terpenting pada MySQL terletak pada database sistem bernama mysql. Untuk menjaga dari segala kemungkinan, baik itu kerusakan atau gangguan dari cracker, Anda perlu memiliki file backup-nya. Sehingga jika sewaktu-waktu diperlukan, Anda tinggal me-restore ke database aslinya.

2) Proteksi Database Sistem

Cara mengamankan database sistem, yaitu dengan memberikan perintah GRANT. Dengan demikian, client hanya dapat mengakses saja tanpa bisa melakukan modifikasi. Hal ini perlu di terapkan pada database server. Selain itu, jika perlu Anda dapat men-disable perintah SHOW DATABASES sehingga client tidak dapat menampilkan database yang terdapat pada MySQL dengan mengaktifkan perintah "skip_show_database".

Untuk itu lakukan perintah seperti berikut:



1. Buka Command Prompt as Administrator dan Matikan MySQL Server dengan perintah: NET STOP MySQL.
2. Kemudian aktifkan skip_show_database, dengan menggunakan perintah berikut: mysqld –skip-show-database
3. Untuk memastikannya, masuk pada prompt mysql kemudian berikan perintah: SHOW VARIABLES;
4. Pada tampilan yang ada carilah bagian yang berisi:

```
|skip_show_database      |ON
```

5. Setelah mendapatkan skip_show_database sudah aktif, lalu keluar dari prompt mysql.
6. Jalankan kembali MySQL server seperti biasanya.
7. Selanjutnya untuk melihat hasilnya, lakukan perintah “SHOW DATABASES;” dengan menggunakan user biasa dan akan keluar hasil bahwa show databases tidak diperbolehkan pada user tamu.

c) Keamanan Transaksi Data

Transaksi data merupakan proses pertukaran data antara client dengan server. Normalnya MySQL tidak menggunakan enkripsi ketika melakukan koneksi, karena hal ini dapat menyebabkan proses koneksi client/server menjadi lambat. Hal ini juga yang menyebabkan mengapa MySQL begitu cepat dalam proses query. Meski demikian, dengan lemahnya transaksi data segala kemungkinan bisa terjadi. Misalnya saja, penyerangan yang dilakukan oleh cracker memungkinkan dapat membuka password di antara lalu lintas hubungan client ke server.

Untuk itu, jika komunikasi yang terjadi antara client dan server tersebut menggunakan jaringan yang kurang bagus, sebaiknya menggunakan komunikasi data dalam bentuk terenkripsi.

d) OpenSSL

OpenSSL (Open Secure Socket Layer) merupakan suatu proyek pengembangan protokol SSL (Secure Socket Layer) yang didasarkan pada library SSLeay.

SSL merupakan protokol yang digunakan untuk melakukan algoritma enkripsi yang menjamin keamanan data. Protokol ini menggunakan verifikasi X509, yakni suatu standar identitas yang sering digunakan pada internet.

Sejak versi 4.0 ini, MySQL memiliki program internal yang dapat digunakan untuk mengecek status OpenSSL. Sehingga Anda dapat mengetahui apakah protokol tersebut sudah aktif atau belum. Untuk itu lakukan perintah seperti berikut:

```
[c:\] XAMPP for Windows - mysql -u root
MariaDB [(none)]> show variables like "have_openssl";
+-----+-----+
| Variable_name | Value |
+-----+-----+
| have_openssl | NO   |
+-----+-----+
1 row in set (0.115 sec)

MariaDB [(none)]> -
```

Pada gambar di atas terlihat bahwa OpenSSL belum diaktifkan. Untuk itu perlu untuk mencari program tersebut dan bisa diakses melalui alamat <https://www.openssl.org/>. Jika Anda menggunakan koneksi tersebut, maka semua koneksi client server dengan TCP/IP akan dilakukan dalam bentuk terenkripsi dan keamanan transaksi data akan terjamin.

e) Privilege System

Dalam pemberian izin akses privilege system, hendaknya Anda sedikit saja melakukan perlu berhati-hati. Sekali Keteledoran, segala kemungkinan yang dapat merugikan database Anda bisa terjadi.

Berikut ini beberapa tips yang berhubungan dengan privilege system:

1. Gunakan selalu option IDENTIFIED BY PASSWORD ketika Anda menambahkan user baru dengan perintah GRANT.
2. Sebaiknya jangan berikan izin akses PROCESS privilege kepada semua user, karena tampilan perintah seperti mysqladmin processlist akan menampilkan proses aktif eksekusi query. Hal ini memungkinkan user lain yang dapat menjalankan perintah tersebut untuk melihat user lain ketika melakukan perintah penting seperti UPDATE password.
3. Jangan berikan izin akses FILE privilege kepada semua user, hal ini dapat dimanfaatkan user untuk memasukkan file ke dalam database sistem. Di mana file tersebut dapat di-setting readable sehingga dapat dibaca setiap user, dan Anda tidak dapat melakukan overwrite pada file tersebut.

3. STORE PROCEDURE

Stored Procedure adalah sekumpulan perintah SQL yang disimpan dalam basis data dan dapat dieksekusi berulang kali. Selain itu, Stored Procedure dapat diberi parameter tertentu sehingga fungsi procedure dapat digunakan lebih dinamis berdasarkan parameter. Stored Procedure digunakan untuk mengotomatisasi tugas, meningkatkan kinerja, dan menjaga konsistensi logika bisnis.

a. Keunggulan Stored Procedure:

- 1) **Kinerja:** Eksekusi lebih cepat karena disimpan dalam bentuk yang sudah dikompilasi.
- 2) **Keamanan:** Membatasi akses langsung ke tabel dan memberikan kontrol akses melalui prosedur.
- 3) **Pemeliharaan:** Logika bisnis tersentralisasi, sehingga lebih mudah dikelola.

b. Membuat STORED PROCEDURE

1. Sintak Dasar

```
DELIMITER $$  
CREATE PROCEDURE nama_procedure ()  
AS  
BEGIN  
    -- Perintah SQL  
END $$  
DELIMITER ;
```

2. Memanggil Store Procedure

```
CALL nama_procedure();
```

3. Store Procedure Menggunakan Parameter tertentu

```

DELIMITER $$

CREATE PROCEDURE nama_procedure
(
    parameter [type data] [lebar data]
)
BEGIN
    -- Perintah SQL
END$$

DELIMITER ;

```

8.4. HARDWARE DAN SOFTWARE

Hardware dan software yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. *XAMPP (Mysql, Apache, PhpMyAdmin)*
3. *Browser.*

8.5. PRE-TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

N o	CPL	CPMK	Pertanyaan	Skor
1 . .	CPL 03- KU03	CPMK 3.3.3	1. Jelaskan Konsep TRIGGER dalam basisdata?	25
2 . .	CPL 03- KU03	CPMK 3.3.3	2. Sebutkan dan jelaskan 3 contoh Event dalam TRIGGER?	25
3 . .	CPL 03- KU03	CPMK 3.3.3	3. Jelaskan fungsi TRIGGER dalam basisdata?	25
4 . . 5 . .	CPL 03- KU03	CPMK 3.3.3	4. Berikan contoh minimal 2 penggunaan TRIGGER dalam basisdata?	25

8.6. LANGKAH PRAKTIKUM

Langkah praktikum berisi tahapan secara rinci bagaimana praktikum dijalankan dan apa hasil yang harus dicapai dari setiap langkah.

LANGKAH-LANGKAH TRIGGER

1. Buatlah dan Isikan tabel produk dan table log_harga seperti tabel 11.1 dan 11.2:

Tabel 11. 1 Tabel Produk

```
MariaDB [penjualan]> desc produk;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| kode_produk | varchar(6) | NO | PRI | NULL |
| nama_produk | varchar(100) | YES | | NULL |
| harga | int(11) | YES | | NULL |
+-----+-----+-----+-----+-----+
3 rows in set (0.094 sec)

MariaDB [penjualan]>
```

Tabel 11. 2 Tabel Log Harga

```
MariaDB [penjualan]> desc log_harga;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| log_id | int(11) | NO | PRI | NULL |
| kode_produk | varchar(6) | YES | | NULL |
| harga_lama | int(11) | YES | | NULL |
| harga_baru | int(11) | YES | | NULL |
| waktu_perubahan | datetime | YES | | NULL |
+-----+-----+-----+-----+-----+
5 rows in set (0.017 sec)
```

2. Buatlah Trigger untuk mencatat perubahan harga Ketika ada perintah update pada produk. Langkahnya dengan membuat sintaks sebagai berikut:

```
DELIMITER $$  
CREATE TRIGGER before_produk_update  
BEFORE UPDATE ON produk FOR EACH ROW  
BEGIN  
    INSERT INTO log_harga  
    set kode_produk = OLD.kode_produk,  
        harga_baru=new.harga,  
        harga_lama=old.harga,  
        waktu_perubahan = NOW();  
END$$ DELIMITER ;
```

Penjelasan sintaks:

- baris 2 – Kita membuat sebuah Trigger baru dengan nama before_produk_update
- baris 3 – Pada Trigger ini kita menggunakan event BEFORE UPDATE
- baris 6 – Query SQL untuk melakukan insert data ke tabel log_harga_produk

3. Isikan data

Isikan data ke table produk dengan perintah sebagai berikut:

```
INSERT INTO `produk` VALUES ('PR001', 'TIGA BULAN JAGO LARAVEL', 200000);  
INSERT INTO `produk` VALUES ('PR002', 'SEMINGGU JAGO PHP MYSQL', 180000);
```

Setelah itu, cek isi table dengan perintah:

```
SELECT * FROM produk;
```

```
MariaDB [penjualan]> select * from produk;
+-----+-----+-----+
| kode_produk | nama_produk | harga |
+-----+-----+-----+
| PR001 | TIGA BULAN JAGO LARAVEL | 200000 |
| PR002 | SEMINGGU JAGO PHP MYSQL | 180000 |
+-----+-----+-----+
2 rows in set (0.001 sec)

MariaDB [penjualan]>
```

4. Kemudian lakukan UPDATE Data produk dengan tujuan mengganti harga PR001 dari 20000 menjadi 250000 dengan perintah.

```
UPDATE produk SET harga=250000 WHERE kode_produkt='PR001'
```

Lakukan pengecekan terhadap perintah di atas dengan memberi perintah:

```
SELECT * FROM produk;
```

```
MariaDB [penjualan]> select * from produk;
+-----+-----+-----+
| kode_produk | nama_produk | harga |
+-----+-----+-----+
| PR001 | TIGA BULAN JAGO LARAVEL | 250000 |
| PR002 | SEMINGGU JAGO PHP MYSQL | 180000 |
+-----+-----+-----+
2 rows in set (0.001 sec)

MariaDB [penjualan]>
```

5. Kemudian lakukan pengecekan pada table log_harga dengan perintah:

```
SELECT * FROM log_harga.
```

Waktu terpasang setelah dilakukan perubahan kembali, yaitu dari harga 250000 menjadi 200000, kemudian perubahan akan tercatat di dalam tabel log_harga.

```
MariaDB [penjualan]> select * from log_harga;
+-----+-----+-----+-----+
| log_id | kode_produkt | harga_lama | harga_baru | waktu_perubahan |
+-----+-----+-----+-----+
| 0 | PR001 | 250000 | 200000 | 2023-08-21 11:43:24 |
+-----+-----+-----+-----+
1 row in set (0.001 sec)

MariaDB [penjualan]>
```

6. Jika ingin mendapatkan informasi mengenai list Trigger yang ada pada suatu basisdata, maka dapat dilakukan dengan memberikan perintah: SHOW TRIGGERS
7. Sedangkan untuk melakukan penghapusan suatu Trigger dari Basisdata yang sudah dibuat yaitu dengan perintah:

```
DROP TRIGGER nama_trigger;
//contoh implementasinya
DROP TRIGGER before_produk_update;
```

LANGKAH-LANGKAH STORE PROCEDURE

Dari table produk yang sudah dibuat, kita akan membuat store procedure untuk memasukan data produk.

- Buat store procedure insert terlebih dahulu**

DELIMITER \$\$

```
CREATE PROCEDURE insertproduk
(
    kode_produk VARCHAR(6),
    nama_produk VARCHAR(100),
    harga INT(11)
)
BEGIN
    INSERT INTO produk
    VALUES (kode_produk, nama_produk, harga);

END$$
```

DELIMITER ;

2. Ingat, Jika kita ingin memasukkan record baru di table produk maka akan ada 3 parameter saat memanggil Stored Procedure insertproduk() yaitu kode_produk, nama_produk, harga
3. Memanggil Store Procedure Insert
CALL insertproduk('PR003','BELAJAR STORE PROCEDURE',23000);
4. Untuk melihatnya, silahkan lakukan perintah
SELECT * FROM produk;

Aturan Penilaian (Total Skor: 100):

No	CPL	CPMK	SUB CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL 03- KU03-	CPM K 3.3.3	CPM K 3.3.3. 2	Praktekkanlah langkah Langkah TRIGGER 1-7	Jawaban hasil praktek (Screenshot)	50
2.	CPL 03- KU03-	CPM K 3.3.3	CPM K 3.3.3	Praktekkanlah langkah Langkah STORE PROCEDRUE 1-4	Jawaban hasil praktek (Screenshot)	50

8.7. POST TEST

Jawablah pertanyaan berikut (Total Skor: 100):

No	CPL	CPMK	SUB CPMK	Pertanyaan	Dokumen Pendukung	Skor

1 .	CPL 03- KU0 3	CPMK 3.3.3	CPMK 3.3.3.2	<p>Silahkan lakukan percobaan dengan menggunakan table yang sudah ada dengan perintah Trigger:</p> <ol style="list-style-type: none"> 1. BEFORE DELETE – dijalankan sebelum proses delete data. 2. AFTER DELETE – dijalankan setelah proses delete data 	Jawaban hasil praktek disamping (Screenshot)	50
2 .				<p>Silahkan lakukan percobaan dengan menggunakan table yang sudah ada dengan memanfaatkan STORE PROCEDURE:</p> <ol style="list-style-type: none"> 3. kita ingin membuat Store Procedure untuk mencari data produk berdasarkan harga. 	Jawaban hasil praktek disamping (Screenshot)	50

8.8. HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessme nt	CPL	CPMK	SUB CPM K	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	03- KU03	3.3.3.	3.3.3. 2	20%		
2.	Praktik	03- KU03	3.3.3.	3.3.3. 2	30%		
3.	Post-Test	03- KU03	3.3.3.	3.3.3. 2	50%		
Total Nilai							

LEMBAR JAWABAN PRE-TEST DAN POST-TEST PRAKTIKUM

Nama : NIM :	Asisten: Paraf Asisten:	Tanggal: Nilai:
-----------------------------------	--	--------------------------------------

PRAKTIKUM 9: OPERASI CRUD DALAM NOSQL

Pertemuan ke : 9

Total Alokasi Waktu : 180 menit

- Materi : 30 menit
- Pre-Test : 30 menit
- Praktikum : 60 menit
- Post-Test : 60 menit

Total Bobot Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 30 %
- Post-Test : 50 %

Pemenuhan CPL dan CPMK:

CPL 03-KU3	Mempunyai kemampuan dalam mendefinisikan kebutuhan pengguna atau pasar terhadap kinerja (menganalisis, mengevaluasi dan mengembangkan) algoritma/metode berbasis komputer untuk mengambil keputusan secara tepat dalam konteks penyelesaian masalah di bidang Rekayasa Perangkat Lunak serta Data dan Sistem Cerdas maupun bidang lainnya, berdasarkan hasil analisis informasi dan data.
CPMK 3.3.3	Mengimplementasikan <i>query</i> basis data dan lingkungannya

9.1 DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu:

1. Mampu mengimplementasikan penggunaan MongoDB sebagai basis data NoSQL.

9.2 INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:

CPL 03-KU3	CPMK 3.3.3	Kemampuan mahasiswa meningkat dalam menjalankan operasi <i>create, read, update, dan delete</i> (CRUD) dalam basis data NoSQL seperti MongoDB
------------	------------	---

9.3 TEORI PENDUKUNG

1. No SQL



"Not Only SQL" atau NoSQL merupakan jenis sistem manajemen basis data yang tidak menggunakan model relasional seperti pada *Relational Database Management System* (RDBMS). Dibanding basis data relasional, NoSQL memiliki keunggulan dalam menangani sistem yang memiliki kebutuhan berikut:

- 1) *Availability* yang tinggi
- 2) *Query* dan performa cepat
- 3) Replikasi mudah
- 4) Data bervolume besar (*big data*)
- 5) *Scalability* tinggi
- 6) Skema fleksibel

Tipe-tipe basis data NoSQL adalah sebagai berikut:

- 1) *Document-based*: menyimpan data dalam format dokumen, contoh: MongoDB
- 2) *Key-value based*: menyimpan data dalam bentuk pasangan *key-value*, contoh: Redis
- 3) *Column-based*: menyimpan data berdasarkan kolom, contoh: Cassandra
- 4) *Graph-based*: menyimpan data dalam bentuk simpul/node dan edge/sisi, contoh: Neo4j

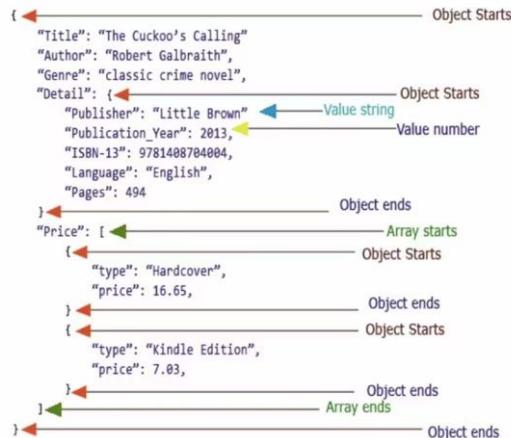
2. MongoDB

MongoDB merupakan basis data NoSQL *document-based*, di mana data disimpan dalam format BSON (Binary JSON). Adapun padanan terminologi/konsep dalam basis data SQL dan MongoDB ditunjukkan pada Tabel 9.1 di bawah ini.

Tabel 9.1 Terminologi dalam SQL dan MongoDB

SQL	MongoDB
<i>Database</i>	<i>Database</i>
Tabel	<i>Collection</i>
Baris	Dokumen
Kolom	<i>Field</i>
<i>Join</i>	<i>\$lookup</i>

Contoh dokumen dalam MongoDB ditunjukkan oleh Gambar 9.1.



Gambar 9. 1 Dokumen MongoDB



3. Operasi CRUD

1) Create

Membuat collection

```
db.createCollection(<collectionName>)
```

Contoh:

```
db.createCollection("movies")
```

Membuat dokumen

```
db.collectionName.insertOne(<dokumen>)
db.collectionName.insertMany(<dokumen-dokumen>)
db.collectionName.insertOne(<dokumen-dokumen>)
```

Catatan:

- <dokumen> : { "field": "value"} (digunakan untuk menambah satu dokumen)
- <dokumen-dokumen> : [{"field": "value"}, {"field": "value"}, ...] (digunakan untuk menambah beberapa dokumen)

Contoh:

Menambahkan tiga dokumen ke koleksi movies

```
db.movies.insertMany([
    {"title": "Ghosbusters"},
    {"title": "E.T."},
    {"title": "Blade Runner"}
]);
```

2) Read

Query digunakan untuk menjelaskan kriteria seleksi menggunakan operator, sedangkan proyeksi digunakan untuk memilih *field* mana yang ingin ditampilkan di hasil pencarian.

Mencari dokumen

```
db.collectionName.find(<query>, <proyeksi>)
```

Beberapa operasi yang dapat digunakan dalam query antara lain:

- \$eq: sama dengan
- \$ne: tidak sama dengan
- \$gt: lebih dari
- \$gte: lebih dari atau sama dengan
- \$lt: kurang dari
- \$lte: kurang dari atau sama dengan
- \$in: mencocokkan dengan nilai dalam array
- \$nin: mencocokkan dengan nilai yang tidak ada dalam array

Contoh:

Menampilkan judul film yang dirilis antara tahun 2000 - 2011

```
db.movies.find({$or: [{"year": {$gte: 2000}}, {"year": {$lte: 2011}}]}, {"title": 1})
```

Membuat query pada field dalam embedded/nested document

```
db.movies.find({"awards.win": 2})
```

Membuat query pada array, notasi dot dapat digunakan untuk merepresentasikan posisi index

```
{ "project": "coding", "staff": ["lead", "standby", "support"]}
```

```
db.movies.find({"staff.1": "standby"})
```

3) Update

Filter menotasikan kriteria seleksi dari dokumen yang akan di-update, *update* mengimplikasikan modifikasi atau perubahan yang akan diberlakukan pada dokumen yang terseleksi, dan **Option** mengimplikasikan aksi tambahan yang akan dieksekusi pada operasi.

Memodifikasi dokumen

```
db.collectionName.updateOne(<filter>, <update> action, <option>)
db.collectionName.updateMany(<filter>, <update> action, <option>)
```

Contoh:

Menambah field comment di dokumen dengan judul “A blog post”

```
db.posts.updateOne({"title": "A blog post"}, 
{$push: {"comments": 
{"name": "joe",
"email": "joe@example.com",
"content": "nice post. "}})
})
```

Menambah data baru di field yang berisi array

```
db.users.updateOne({"_id": ObjectId("4b2d75476cc613d5ee930164")},
{$addToSet: {"emails": "joe@gmail.com"}})
```

Menghapus satu data dari field berisi array

Misal field “todo” di “lists” berisi [“dishes”, “laundry”, “dry cleaning”]

```
db.lists.updateOne({}, {$pull: {"todo": "laundry"}})
```

Untuk menghapus elemen terakhir dari array tersebut dapat digunakan:

```
db.lists.updateOne({}, {$pop: {"todo": 1}})
```

Untuk menghapus elemen dari depan dapat digunakan:

```
db.lists.updateOne({}, {$pop: {"todo": -1}})
```



4) Delete

Filter menotasikan kriteria seleksi dari dokumen yang akan dihapus sedangkan Option mengimplikasikan aksi tambahan yang akan dieksekusi pada operasi.

Menghapus dokumen

```
db.collectionName.deleteOne(<filter>, <option>)
db.collectionName.deleteMany(<filter>, <option>)
```

Contoh:

Menghapus film dengan _id: 4

```
db.movies.deleteOne({"_id": 4})
```

Menghapus film dari genre “comedy”

```
db.movies.deleteMany({"genre": "comedy"})
```

9.4 HARDWARE DAN SOFTWARE

Hardware dan *software* yang digunakan dalam praktikum ini yaitu:

1. MongoDB *compass*
2. Browser (firefox, chrome)

9.5 PRE-TEST

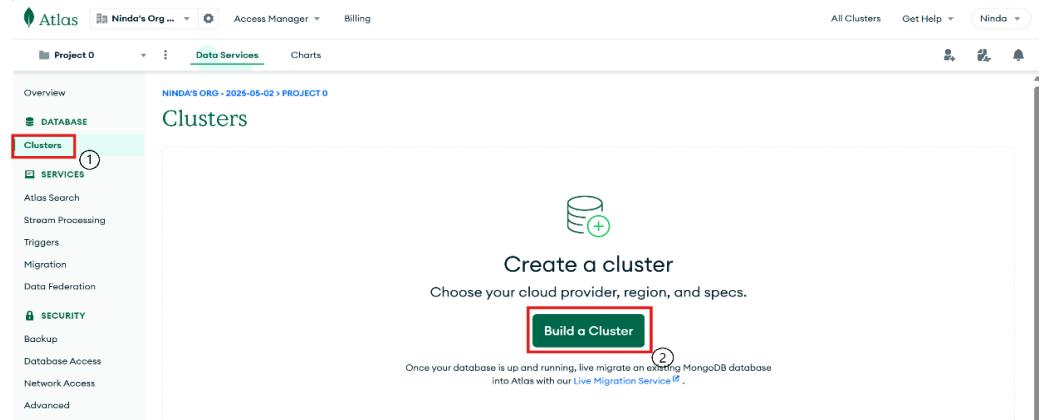
Jawablah pertanyaan berikut (**Total Skor: 100**):

N o	CPL	CPMK	Pertanyaan	Skor
1 .	CPL 03- KU03	CPMK 3.3.3	Jelaskan perbedaan antara basis data SQL dan NoSQL!	25
2 .	CPL 03- KU03	CPMK 3.3.3	Jelaskan perbedaan insertOne() dan insertMany() dalam MongoDB!	25
3 .	CPL 03- KU03	CPMK 3.3.3	Jelaskan kondisi di mana NoSQL lebih unggul dibandingkan SQL!	25
4 .	CPL 03- KU03	CPMK 3.3.3	Sebutkan minimal 4 operasi CRUD di MongoDB!	25

9.6 LANGKAH PRAKTIKUM

1. MongoDB Atlas

- 1) Buka halaman <https://www.mongodb.com/cloud/atlas/register> lalu buat akun MongoDB menggunakan email atau pilih “Sign up with Google”
- 2) Buka halaman “Cluster” dan klik “Build a Cluster”



Gambar 9. 2 Membuat cluster

3) Pilih “Free” cluster

Deploy your cluster

Use a template below or set up advanced configuration options. You can also edit these configuration options once the cluster is created.

Cluster Type	Cost	Storage	RAM	vCPU
M10	\$0.09/hour	10 GB	2 GB	2 vCPUs
Flex	From \$0.01/hour Up to \$30/month	5 GB	Shared	Shared
Free	For learning and exploring MongoDB in a cloud environment.	512 MB	Shared	Shared

Free forever! Your free cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

Gambar 9. 3 Pilih free cluster

4) Isi nama, provider, dan region di bagian “Configurations” lalu klik “Create Deployment”

Configurations

Name
You cannot change the name once the cluster is created.
Cluster0

Provider
 AWS Google Cloud Azure

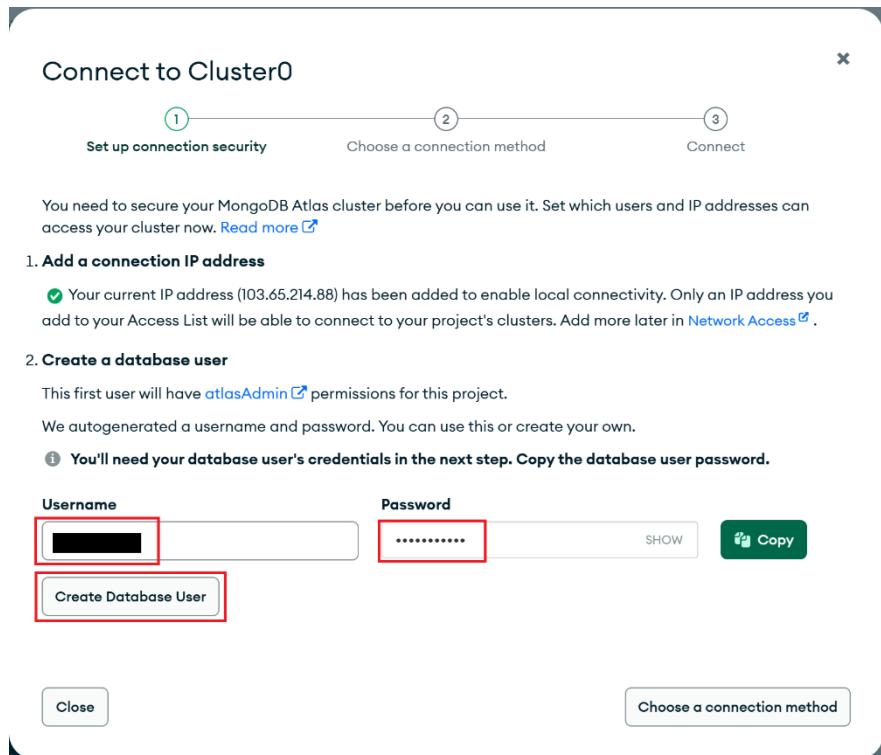
Region
Singapore (ap-southeast-1)
★ Recommended (1) Low carbon emissions (1)

Tag (optional)
Create your first tag to categorize and label your resources; more tags can be added later. [Learn more](#).

I'll do this later

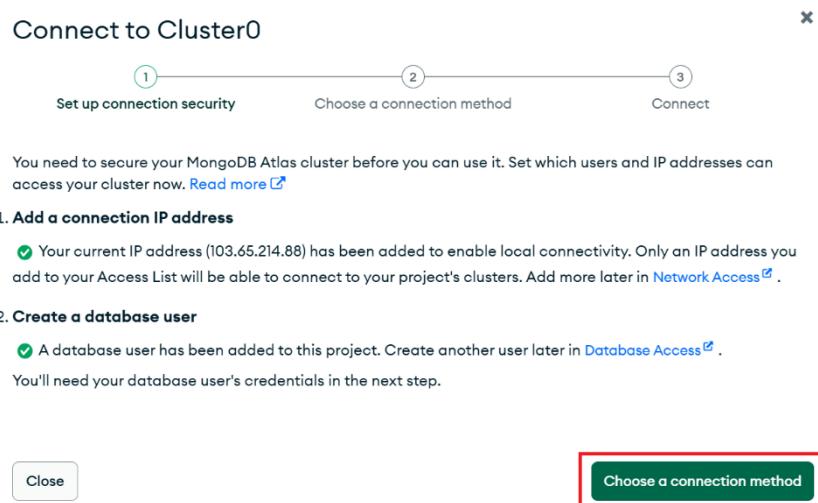
Gambar 9. 4 Mengisi name, provider, dan region saat mendeploy cluster

5) Buat database user dengan memasukkan “Username” dan “Password” lalu klik tombol “Create Database User”



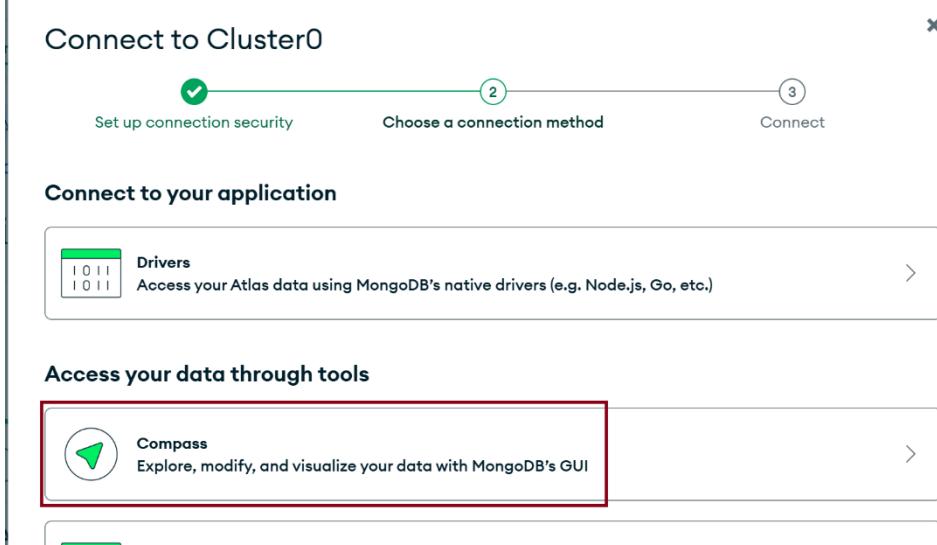
Gambar 9. 5 Mengisi Username dan Password saat mendaftarkan *database user*

- 6) Setelah membuat *database user*, klik tombol “**Choose connection method**”



Gambar 9. 6 Membuat *connection method*

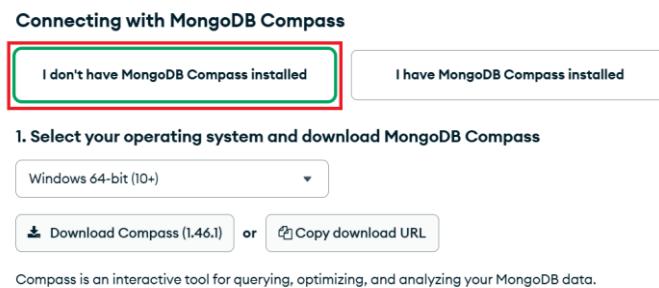
- 7) Pilih “**Compass**” untuk menggunakan **MongoDB Compass** sebagai metode koneksi



Gambar 9. 7 Memilih MongoDB Compass sebagai *connection method*

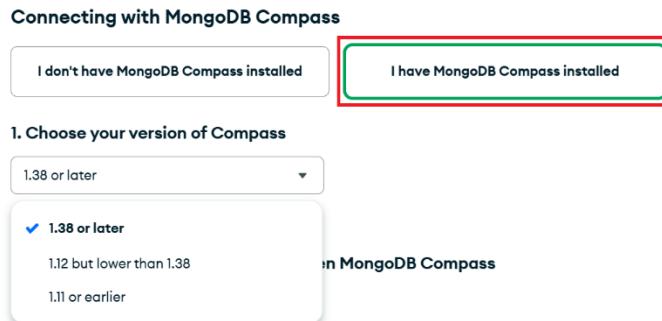
8) Salin *connection string*

- Apabila belum memasang MongoDB Compass di komputer, pilih “I don’t have MongoDB Compass installed”, pilih OS yang sesuai, lalu unduh *installer* dengan cara klik “Download Compass <ver>” atau klik “Copy download URL”



Gambar 9. 8 Belum memiliki MongoDB Compass

- Apabila sudah memasang MongoDB Compass di komputer, pilih “I have MongoDB Compass installed” lalu pilih versi MongoDB Compass yang terpasang di komputer



Gambar 9. 9 Sudah memiliki MongoDB Compass

- Salin *connection string* lalu klik “Done”

Connect to Cluster0



Connecting with MongoDB Compass

[I don't have MongoDB Compass installed](#) [I have MongoDB Compass installed](#)

1. Select your operating system and download MongoDB Compass

Windows 64-bit (10+)

[Download Compass \(1.46.1\)](#) or [Copy download URL](#)

Compass is an interactive tool for querying, optimizing, and analyzing your MongoDB data.

2. Copy the connection string, then open MongoDB Compass

Show Password ⓘ

Use this connection string in your application

mongodb+srv://[REDACTED].mongodb.net/



The password for **nindakhrnns** is included in the connection string for your first time setup. This password will not be available again after exiting this connect flow.

RESOURCES

[Connect with Compass](#)

[Access your Database Users](#)

[Import and Export Data](#)

[Troubleshoot Connections](#)

[Go Back](#)



Gambar 9. 10 Menyalin connection string

2. MongoDB Compass

- 1) Buka aplikasi MongoDB Compass
- 2) Di halaman awal, tempel/paste connection string yang disalin dari MongoDB Atlas

New Connection

Connect to a MongoDB deployment

URI ⓘ

Edit Connection String

mongodb+srv://[REDACTED].mongodb.net/

Advanced Connection Options

Save Save & Connect Connect

RESOURCES

Connect with Compass ⓘ

Access your Database Users ⓘ

Import and Export Data ⓘ

Troubleshoot Connections ⓘ

CREATE FREE CLUSTER

How do I find my connection string in Atlas?

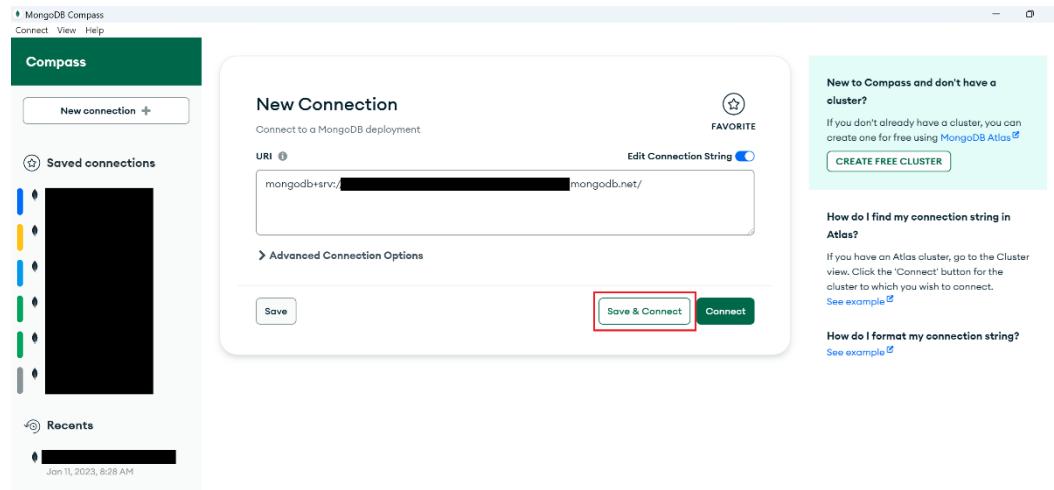
If you have an Atlas cluster, go to the Cluster view. Click the "Connect" button for the cluster to which you wish to connect. See example ⓘ

How do I format my connection string?

See example ⓘ

Gambar 9. 11 Membuat koneksi baru

3) Klik tombol “Save & Connect”



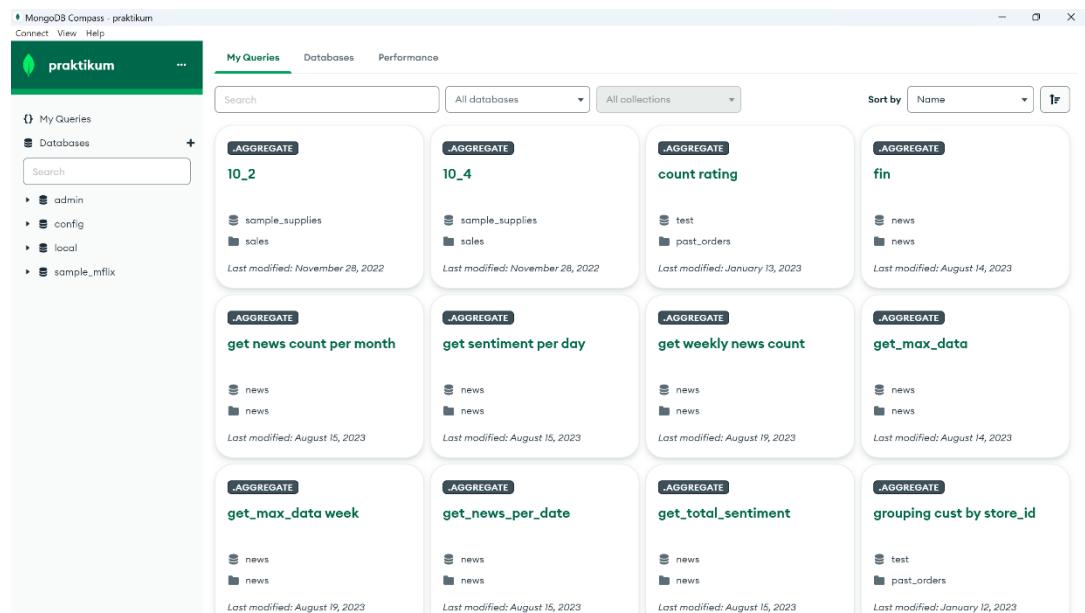
Gambar 9. 12 Menyimpan koneksi

4) Masukkan nama *connection*, misal “praktikum” lalu klik “Save & Connect”



Gambar 9. 13 Memberi nama koneksi

5) Tampilan setelah berhasil terhubung ke MongoDB adalah sebagai berikut

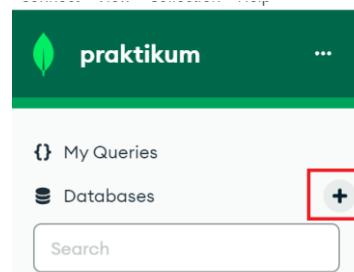


Gambar 9. 14 Berhasil membuat koneksi bernama "praktikum"

3. Operasi CRUD

1) Membuat database dan collection

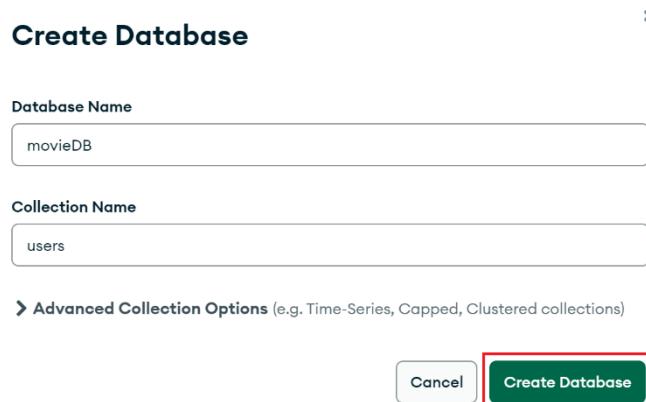
Untuk membuat database baru, klik ikon “+” di bawah ini:



Gambar 9. 15 Membuat database dari halaman awal koneksi

a. users

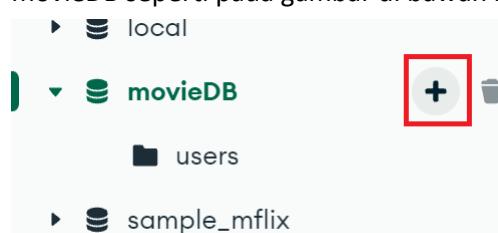
Untuk membuat collection users, klik ikon “+” di samping menu “Databases” lalu masukkan **Database Name: movieDB** dan **Collection Name: users**, lalu klik “Create Database”



Gambar 9. 16 Membuat database

b. movies

Buat collection baru “movies” dengan cara klik ikon “+” di samping nama database movieDB seperti pada gambar di bawah ini:



Gambar 9. 17 Membuat koleksi baru

Masukkan **nama collection**: “movies” lalu klik “Create Collection”

Create Collection

Collection Name

movies

» Advanced Collection Options (e.g. Time-Series, Capped, Clustered collections)

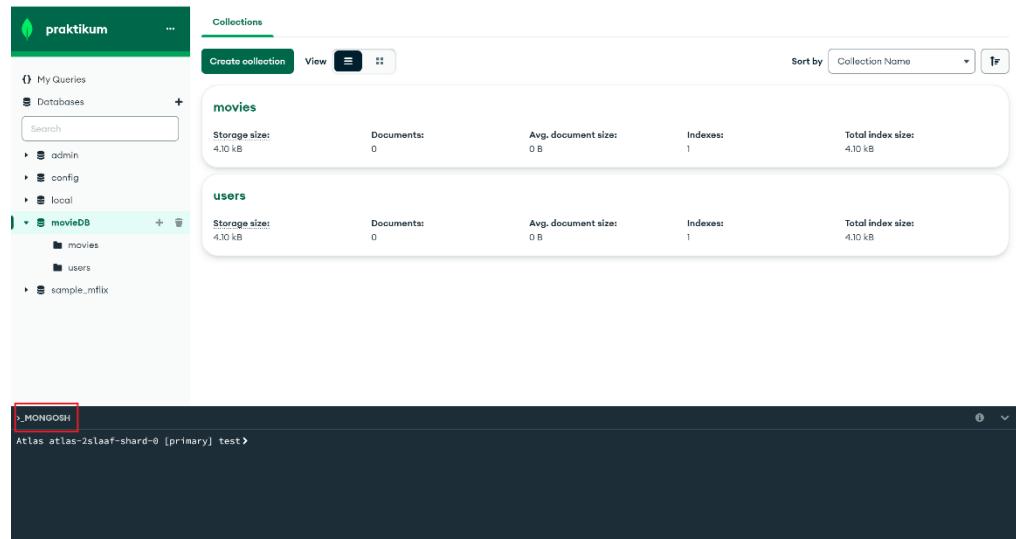
Cancel

Create Collection

Gambar 9. 18 Membuat koleksi baru bernama **movies**

c. **posts**

Pembuatan *collection* juga bisa dilakukan melalui Mongo Shell. Untuk membuka Mongo Shell, klik ikon berikut ini:



Gambar 9. 19 Membuka Mongo Shell

Membuat *collection* baru dapat dilakukan dengan operasi:

a) `use movieDB;`

Berikut ini adalah respon setelah menggunakan `movieDB` di Mongo Shell:

```
> use movieDB
< 'switched to db movieDB'
```

Gambar 9. 20 Respon setelah memilih *database*

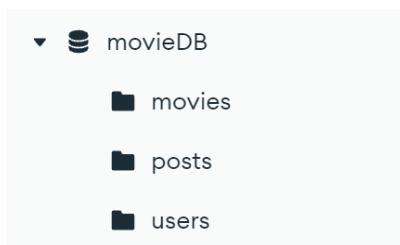
b) `db.createCollection("posts");`

Berikut ini adalah respon setelah *collection* berhasil dibuat:

```
> db.createCollection('posts')
< { ok: 1 }
```

Gambar 9. 21 Respon setelah membuat koleksi baru

- c) Collection `posts` akan tersimpan di *database* movieDB seperti yang ditunjukkan pada gambar di bawah ini:



Gambar 9. 22 Daftar koleksi

2) Menambah *document* dengan perintah

Data yang akan di-*insert* ke *document* `users` antara lain:

- i. username: GoodGuyGreg
firstname: "Good Guy"
lastname: "Greg"
- ii. username: GoodGuyGreg
firstname: "Good Guy"
lastname: "Greg"
- iii. username: JohnDoe1234
firstname: "John"
lastname: "Doe"
yearjoined: 2005
- iv. username: SamJohnSings
firstname: "Sam John"
lastname: "Sings"
rating: 4.9

Meng-*insert* dokumen satu per satu lewat Mongo Shell

```
db.users.insertOne({"username": "GoodGuyGreg", "firstname": "GoodGuy", "lastname": "Greg"})
```

```
> db.users.insertOne({"username": "GoodGuyGreg", "firstname": "GoodGuy", "lastname": "Greg"})
{
  acknowledged: true,
  insertedId: ObjectId("68147964371104c7df7c48ec")
}
```

Gambar 9. 23 Respon setelah menambah satu dokumen

Meng-*insert* beberapa dokumen sekaligus lewat Mongo Shell

```
db.users.insertMany([{"username": "GoogGuyGreg", "firstname": "GoodGuy", "lastname": "Greg"}, {"username": "JohnDoe1234", "firstname": "John", "lastname": "Doe", "yearjoined": 2005}, {"username": "SamJohnSings", "firstname": "Sam John", "lastname": "Sings", "rating": 4.9}])
```

```
> db.users.insertMany([{"username": "GoogGuyGreg", "firstname": "Greg", "lastname": "Googly", "age": 30, "city": "London", "country": "UK", "isEmployee": true}, {"username": "JohnDoe", "firstname": "John", "lastname": "Doe", "age": 25, "city": "New York", "country": "USA", "isEmployee": false}, {"username": "JaneDoe", "firstname": "Jane", "lastname": "Doe", "age": 28, "city": "Los Angeles", "country": "USA", "isEmployee": false}, {"username": "MikeSmith", "firstname": "Mike", "lastname": "Smith", "age": 32, "city": "Chicago", "country": "USA", "isEmployee": true}, {"username": "SarahJohnson", "firstname": "Sarah", "lastname": "Johnson", "age": 29, "city": "Houston", "country": "USA", "isEmployee": false}, {"username": "DavidWilson", "firstname": "David", "lastname": "Wilson", "age": 35, "city": "Phoenix", "country": "USA", "isEmployee": true}, {"username": "EmilyBrown", "firstname": "Emily", "lastname": "Brown", "age": 27, "city": "Seattle", "country": "USA", "isEmployee": false}, {"username": "AaronDavis", "firstname": "Aaron", "lastname": "Davis", "age": 31, "city": "Portland", "country": "USA", "isEmployee": true}, {"username": "BriannaGarcia", "firstname": "Brianna", "lastname": "Garcia", "age": 26, "city": "San Francisco", "country": "USA", "isEmployee": false}, {"username": "CameronHarris", "firstname": "Cameron", "lastname": "Harris", "age": 33, "city": "Austin", "country": "USA", "isEmployee": true}, {"username": "DanielleJohnson", "firstname": "Danielle", "lastname": "Johnson", "age": 29, "city": "Nashville", "country": "USA", "isEmployee": false}, {"username": "ElijahWilson", "firstname": "Elijah", "lastname": "Wilson", "age": 30, "city": "Memphis", "country": "USA", "isEmployee": true}, {"username": "FionaHarris", "firstname": "Fiona", "lastname": "Harris", "age": 28, "city": "Baltimore", "country": "USA", "isEmployee": false}, {"username": "GabeDavis", "firstname": "Gabe", "lastname": "Davis", "age": 34, "city": "Washington D.C.", "country": "USA", "isEmployee": true}, {"username": "HannahJohnson", "firstname": "Hannah", "lastname": "Johnson", "age": 27, "city": "Tampa", "country": "USA", "isEmployee": false}, {"username": "IvanWilson", "firstname": "Ivan", "lastname": "Wilson", "age": 32, "city": "Orlando", "country": "USA", "isEmployee": true}, {"username": "JasmineHarris", "firstname": "Jasmine", "lastname": "Harris", "age": 26, "city": "Miami", "country": "USA", "isEmployee": false}, {"username": "KaiWilson", "firstname": "Kai", "lastname": "Wilson", "age": 31, "city": "St. Louis", "country": "USA", "isEmployee": true}, {"username": "LiamHarris", "firstname": "Liam", "lastname": "Harris", "age": 29, "city": "Milwaukee", "country": "USA", "isEmployee": false}, {"username": "MiaWilson", "firstname": "Mia", "lastname": "Wilson", "age": 30, "city": "Des Moines", "country": "USA", "isEmployee": true}, {"username": "NatalieHarris", "firstname": "Natalie", "lastname": "Harris", "age": 28, "city": "Columbus", "country": "USA", "isEmployee": false}, {"username": "OwenWilson", "firstname": "Owen", "lastname": "Wilson", "age": 33, "city": "Nashville", "country": "USA", "isEmployee": true}, {"username": "PeytonHarris", "firstname": "Peyton", "lastname": "Harris", "age": 27, "city": "Seattle", "country": "USA", "isEmployee": false}, {"username": "QuinnWilson", "firstname": "Quinn", "lastname": "Wilson", "age": 31, "city": "Portland", "country": "USA", "isEmployee": true}, {"username": "RileyHarris", "firstname": "Riley", "lastname": "Harris", "age": 29, "city": "Phoenix", "country": "USA", "isEmployee": false}, {"username": "SavannahWilson", "firstname": "Savannah", "lastname": "Wilson", "age": 30, "city": "Austin", "country": "USA", "isEmployee": true}, {"username": "TannerHarris", "firstname": "Tanner", "lastname": "Harris", "age": 28, "city": "Nashville", "country": "USA", "isEmployee": false}, {"username": "UlyssesWilson", "firstname": "Ulysses", "lastname": "Wilson", "age": 32, "city": "Seattle", "country": "USA", "isEmployee": true}, {"username": "VivianHarris", "firstname": "Vivian", "lastname": "Harris", "age": 26, "city": "Portland", "country": "USA", "isEmployee": false}, {"username": "WadeWilson", "firstname": "Wade", "lastname": "Wilson", "age": 34, "city": "Phoenix", "country": "USA", "isEmployee": true}, {"username": "XavierHarris", "firstname": "Xavier", "lastname": "Harris", "age": 29, "city": "Austin", "country": "USA", "isEmployee": false}, {"username": "YaraWilson", "firstname": "Yara", "lastname": "Wilson", "age": 31, "city": "Nashville", "country": "USA", "isEmployee": true}, {"username": "ZaneHarris", "firstname": "Zane", "lastname": "Harris", "age": 27, "city": "Seattle", "country": "USA", "isEmployee": false}])  
< {  
  acknowledged: true,  
  insertedIds: [  
    '0': ObjectId("68147ada371104c7df7c48ed"),  
    '1': ObjectId("68147ada371104c7df7c48ee"),  
    '2': ObjectId("68147ada371104c7df7c48ef")  
  ]  
}  
Atlas atlas-2slaaf-shard-0 [primary] movieDB>
```

Gambar 9. 24 Respon setelah menambah beberapa dokumen

3) Menambah *document* dengan impor data

Data yang akan di-insert ke document `movies` disimpan dalam bentuk file `movies.json` yang dapat diunduh dari [sini](#).

- a. Buka *collection movies*
 - b. Klik tombol “**Add Data**”
 - c. Pilih “**Import File**”

The screenshot shows the MongoDB Compass interface. On the left sidebar, there's a tree view of databases: OnlineStore, admin, basisdata, config, local, and movieDB, with 'movies' selected. Below the tree are two buttons: 'ADD DATA' and 'EXPORT COLLECTION'. The main area is titled 'movieDB.movies' and contains tabs for 'Documents', 'Aggregations', 'Schema', 'Explain Plan', 'Indexes', and 'Validation'. Under 'Documents', there's a search bar with placeholder text 'Type a query: { field: 'value' }', and buttons for 'Reset', 'Find', and 'More Options'. At the bottom right of the main area, there's a small icon of a document with a green arrow pointing down.

Gambar 9. 25 Memilih "Import File"

- d. Klik “Select a file” untuk memilih file JSON “movies.json” yang berisi *document* untuk movies lalu klik “Import”

Import

To Collection movieDB.movies

Select File

Input File Type

JSON

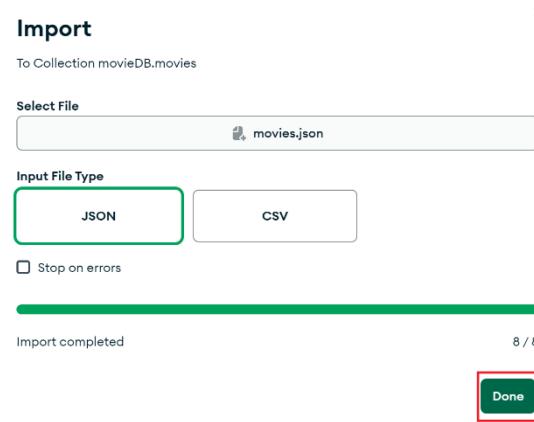
CSV

Stop on errors

Cancel Import

Gambar 9. 26 Memilih file .json/.csv

Setelah selesai, klik tombol “Done”



Gambar 9. 27 Menyelesaikan impor dokumen

4) Menambah document dengan insert data

Data yang akan di-insert ke document “posts” disimpan dalam bentuk file **posts.json** yang dapat diunduh dari [sini](#). Langkah untuk meng-insert data adalah sebagai berikut:

- Buka collection **posts** lalu klik “Add Data” dan pilih “Insert Document”

Gambar 9. 28 Memilih insert dokumen

- Salin semua teks di dalam file **posts.json** lalu *paste* di MongoDB Compass dan klik “Insert”

```

12   "username": "GoodGuyGreg",
13   "title": "Forgot",
14   "body": "Forgot to mention, I'm going with my friend."
15 },
16 {
17   "_id": {
18     "$oid": "636906fa0449edd4b1fdcd58"
19   },
20   "username": "SamJohnSings",
21   "title": "Credit Card",
22   "body": "You need a good credit rating"
23 },
24 {
25   "_id": {
26     "$oid": "636906fa0449edd4b1fdcd59"
27   },
28   "username": "GoodGuyGreg",
29   "title": "Github",
30   "body": "Forks your repo and sets to private"
31 }

```

Gambar 9. 29 Menyalin dokumen dari file **posts.json**

5) Membuat *query*

- a. Melihat semua dokumen di movies

Masukkan perintah `db.movies.find({})` ke MongoDB Shell untuk melihat semua dokumen. Query juga dapat dimasukkan di filter untuk memperoleh hasil query sebagai berikut:

movieDB.movies

8 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

Filter ⚙️ □

Reset Find ↻ More Options ➔

ADD DATA EXPORT COLLECTION

1 - 8 of 8

_id: ObjectId('636906050449edd4b1fdcd4a')
title: "Fight Club"
writer: "Chuck Palahniuk"
year: 1999
actors: Array

_id: ObjectId('636906050449edd4b1fdcd4b')
title: "Pulp Fiction"
writer: "Quentin Tarantino"
year: 1994
actors: Array

_id: ObjectId('636906050449edd4b1fdcd4c')
title: "Inglourious Basters"
writer: "Quentin Tarantino"
year: 2009
actors: Array

Gambar 9. 30 Semua dokumen di koleksi movies

- b. Melihat semua dokumen di `movies` dengan kriteria “writer” = “Quentin Tarantino”

Masukkan perintah `db.movies.find({"writer": "Quentin Tarantino"})` ke MongoDB Shell. Query juga dapat dimasukkan di *filter* untuk memperoleh hasil query sebagai berikut:

Gambar 9. 31 Menampilkan film yang ditulis oleh Quentin Tarantino

- c. Melihat semua dokumen di `movies` yang dimainkan "Brad Pitt"

Masukkan perintah `db.movies.find({ "actors": "Brad Pitt" })` ke MongoDB Shell. Query juga dapat dimasukkan di *filter* untuk memperoleh hasil query sebagai berikut:

The screenshot shows the MongoDB Compass interface with a search filter applied: `{"actors": "Brad Pitt"}`. Two documents are listed:

```

_id: ObjectId('636906050449edd4b1fdcd4a')
title: "Fight Club"
writer: "Chuck Palahniuk"
year: 1999
actors: Array
  0: "Brad Pitt"
  1: "Edward Norton"

_id: ObjectId('636906050449edd4b1fdcd4c')
title: "Inglourious Basters"
writer: "Quentin Tarantino"
year: 2009
actors: Array
  0: "Brad Pitt"
  1: "Diane Kruger"
  2: "Eli Roth"

```

Gambar 9. 32 Menampilkan film yang dimainkan oleh Brad Pitt

- d. Melihat semua dokumen di `movies` yang dirilis pada tahun 90-an

Masukkan perintah `db.movies.find({$and:[{"year": {$lt: 2000}}, {"year": {$gte: 1990}}]})` ke MongoDB Shell. Query juga dapat dimasukkan di `filter` untuk memperoleh hasil query sebagai berikut:

The screenshot shows the MongoDB Compass interface with a search filter applied: `[$and:[{"year": {$lt: 2000}}, {"year": {$gte: 1990}}]]`. Eight documents are listed:

```

_id: ObjectId('636906050449edd4b1fdcd4a')
title: "Fight Club"
writer: "Chuck Palahniuk"
year: 1999
actors: Array

_id: ObjectId('636906050449edd4b1fdcd4b')
title: "Pulp Fiction"
writer: "Quentin Tarantino"
year: 1994
actors: Array

```

Gambar 9. 33 Tampilan daftar film yang dirilis pada tahun 90-an

- e. Melihat semua film yang memiliki sinopsis

Masukkan perintah `db.movies.find({"synopsis": {$exists: true}})` ke MongoDB Shell. Query juga dapat dimasukkan di `filter` untuk memperoleh hasil query sebagai berikut:

The screenshot shows the MongoDB Compass interface with a search filter applied: `["synopsis": {$exists: true}]`. One document is listed:

```

_id: ObjectId('63690993c0449edd4b1fdcd5f')
title: "The Hobbit: The Battle of the Five Armies"
writer: "J. R. R. Tolkein"
year: 2012
franchise: "The Hobbit"
synopsis: "Bilbo and Company are forced to engage in a war against an array of co..."

```

Gambar 9. 34 Tampilan dokumen film yang memiliki sinopsis



- f. Melihat semua film yang **sinopsisnya mengandung kata “Bilbo”** dan **menampilkan “title” saja**

Masukkan perintah `db.movies.find({"synopsis": {$regex: "Bilbo"}}, {"title": 1, "_id": 0})` ke MongoDB Shell. Query juga dapat dimasukkan di *filter* untuk memperoleh hasil query sebagai berikut:

```
> db.movies.find({"synopsis": {$regex: "Bilbo"}}, {"title: 1, _id: 0})
< {
    title: 'The Hobbit: The Desolation of Smaug'
}
```

Gambar 9. 35 Tampilan dokumen film yang sinopsisnya mengandung kata “Bilbo” dan menampilkan “title” saja

- g. Melihat film yang **bukan merupakan bagian dari franchise “The Hobbit”**

Masukkan perintah `db.movies.find({"franchise": {$ne: "The Hobbit"}})` ke MongoDB Shell. Query juga dapat dimasukkan di *filter* untuk memperoleh hasil query sebagai berikut:

```
_id: ObjectId('636906050449edd4b1fdcd4c')
title: "Inglourious Basters"
writer: "Quentin Tarantino"
year: 2009
> actors: Array

-----
_id: ObjectId('636906050449edd4b1fdcd50')
title: "PeeWee Herman's Big Adventure"

-----
_id: ObjectId('636906050449edd4b1fdcd51')
title: "Avatar"
```

Gambar 9. 36 Tampilan dokumen film yang bukan merupakan bagian dari franchise “The Hobbit”

6) *Update document*

- a. Menambahkan **sinopsis** ke film dengan **judul “The Hobbit: An Unexpected Journey”**
Sebelum penambahan sinopsis:

```
_id: ObjectId('636909030449edd4b1fdcd5d')
title: "The Hobbit: An unexpected journey"
writer: "J. R. R. Tolkein"
year: 2012
franchise: "The Hobbit"
```

Gambar 9. 37 Dokumen sebelum perubahan data

Masukkan perintah `db.movies.updateOne({"title": "The Hobbit: An unexpected journey"}, {$set: {"synopsis": "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves."}})`

```
dwarves to reclaim their mountain home - and the gold within it -  
from the dragon Smaug."}}) ke MongoDB Shell.
```

Apabila *update* data berhasil, akan muncul respon berikut:

```
< {  
    acknowledged: true,  
    insertedId: null,  
    matchedCount: 1,  
    modifiedCount: 1,  
    upsertedCount: 0  
}
```

Gambar 9. 38 Respon setelah berhasil mengubah dokumen yang memenuhi kriteria/filter

Setelah penambahan sinopsis:

```
_id: ObjectId('636909030449edd4b1fdcd5d')  
title: "The Hobbit: An unexpected journey"  
writer: "J. R. R. Tolkein"  
year: 2012  
franchise: "The Hobbit"  
synopsis: "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain wit..."
```

Gambar 9. 39 Dokumen setelah penambahan data

b. **Menambahkan aktor “Samuel L. Jackson” ke film berjudul “Pulp Fiction”**

Masukkan perintah `db.movies.updateOne({"title": "Pulp Fiction"}, {$addToSet: {"actors": "Samuel L. Jackson"}})` ke MongoDB Shell. Setelah penambahan data aktor, dokumen akan menjadi seperti:

```
_id: ObjectId('636906050449edd4b1fdcd4b')  
title: "Pulp Fiction"  
writer: "Quentin Tarantino"  
year: 1994  
▼ actors: Array  
  0: "John Travolta"  
  1: "Uma Thurman"  
  2: "Samuel L. Jackson"
```

Gambar 9. 40 Setelah penambahan aktor di film "Pulp Fiction"

c. **Menghapus aktor urutan terakhir** yang memainkan film “Pulp Fiction”

Masukkan perintah `db.movies.updateOne({"title": "Pulp Fiction"}, {$pop: {"actors": 1}})` ke MongoDB Shell.

```
_id: ObjectId('636906050449edd4b1fdcd4b')  
title: "Pulp Fiction"  
writer: "Quentin Tarantino"  
year: 1994  
▼ actors: Array  
  0: "John Travolta"  
  1: "Uma Thurman"
```

Gambar 9. 41 Setelah penghapusan aktor terakhir film “Pulp Fiction”

- d. Menghapus aktor “John Travolta” dari daftar pemain film “Pulp Fiction”

Masukkan perintah `db.movies.updateOne({"title": "Pulp Fiction"}, {$pull: {"actors": "John Travolta"}})` ke MongoDB Shell

```
_id: ObjectId('636906050449edd4b1fdcd4b')
title: "Pulp Fiction"
writer: "Quentin Tarantino"
year: 1994
actors: Array
  0: "Uma Thurman"
```

Gambar 9. 42 Setelah penghapusan aktor John Travolta dari film Pulp Fiction

7) Menghapus dokumen

- a. Menghapus dokumen dengan judul “Pulp Fiction”

Masukkan perintah `db.movies.deleteOne({"title": "Pulp Fiction"})` ke Mongo Shell.

```
> db.movies.deleteOne({"title": "Pulp Fiction"})
{
  acknowledged: true,
  deletedCount: 1
}
```

Gambar 9. 43 Respon setelah menghapus dokumen film berjudul "Pulp Fiction"

- b. Menghapus film yang dimainkan oleh Brad Pitt dan dirilis sebelum tahun 2000

Masukkan perintah `db.movies.deleteOne({$and: [{"actors": "Brad Pitt"}, {"year": {$lt: 2000}}]})` ke Mongo Shell.

```
> db.movies.deleteOne({$and: [{"actors": "Brad Pitt"}, {"year": {$lt: 2000}}]})
{
  acknowledged: true,
  deletedCount: 1
}
```

Gambar 9. 44 Respon setelah menghapus dokumen film yang memenuhi kriteria

Aturan Penilaian (Total Skor: 100):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1	CPL 03- KU03	CPMK 3.3.3	1. <i>Set-up cluster</i> di MongoDB Atlas dan membuat koneksi (<i>connection</i>) baru di MongoDB Compass! (Langkah 1 – 2)	<i>Screenshot</i>	20
2	CPL 03- KU03	CPMK 3.3.3	2. Praktikkanlah langkah-langkah membuat <i>database</i> , <i>connection</i> , dan <i>document</i> ! (Langkah 3.1) sampai 3.4))	<i>Screenshot</i>	25
3	CPL 03- KU03	CPMK 3.3.3	3. Praktikkanlah langkah-langkah membuat <i>query</i> di MongoDB Compass! (Langkah 3.5))	<i>Screenshot</i>	30

4	CPL 03- KU03	CPMK 3.3.3	4. Praktikkanlah langkah-langkah meng-update dokumen di MongoDB Compass! (Langkah 3.6))	Screenshot	15
5	CPL 03- KU03	CPMK 3.3.3	5. Praktikkanlah langkah-langkah menghapus dokumen di MongoDB Compass! (Langkah 3.7))	Screenshot	10

9.7 POST TEST

Jawablah pertanyaan-pertanyaan di bawah ini dimana setiap pertanyaan memiliki bobot nilai yang berbeda, dengan total nilai 100

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1	CPL 03- KU03	CPMK 3.3.3	1. Set-up cluster di MongoDB Atlas dan membuat koneksi (<i>connection</i>) baru di MongoDB Compass! (Langkah 1 – 2)	Screenshot	20
2	CPL 03- KU03	CPMK 3.3.3	2. Praktikkanlah langkah-langkah membuat <i>database, connection, and document!</i> (Langkah 3.1 sampai 3.4))	Screenshot	25
3	CPL 03- KU03	CPMK 3.3.3	3. Praktikkanlah langkah-langkah membuat <i>query</i> di MongoDB Compass! (Langkah 3.5))	Screenshot	30
4	CPL 03- KU03	CPMK 3.3.3	4. Praktikkanlah langkah-langkah meng-update dokumen di MongoDB Compass! (Langkah 3.6))	Screenshot	15
5	CPL 03- KU03	CPMK 3.3.3	5. Praktikkanlah langkah-langkah menghapus dokumen di MongoDB Compass! (Langkah 3.7))	Screenshot	10

9.8 HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1	Pre-Test			20%		
2	Praktik			30%		
3	Post-Test			50%		
Total Nilai						

PRAKTIKUM 10: NoSQL - INDEKS DAN AGREGASI

Pertemuan ke : 10

Total Alokasi Waktu : 180 menit

- Materi : 30 menit
- Pre-Test : 30 menit
- Praktikum : 60 menit
- Post-Test : 60 menit

Total Bobot Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 30 %
- Post-Test : 50 %

Pemenuhan CPL dan CPMK:

CPL 03-KU03	Mempunyai kemampuan dalam mendefinisikan kebutuhan pengguna atau pasar terhadap kinerja (menganalisis, mengevaluasi dan mengembangkan) algoritma/metode berbasis komputer untuk mengambil keputusan secara tepat dalam konteks penyelesaian masalah di bidang Rekayasa Perangkat Lunak serta Data dan Sistem Cerdas maupun bidang lainnya, berdasarkan hasil analisis informasi dan data.
CPMK 3.3.3	Mahasiswa mampu mengimplementasikan query basis data dan lingkungannya

10.1 DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu memahami:

1. Mengimplementasikan konsep indeks dan agregasi dalam basis data non relasional.
2. Merangkai *pipeline* agregasi untuk melakukan analisis data seperti penyaringan (*filtering*), pengelompokan (*grouping*), pengurutan (*sorting*), dan perhitungan agregat.
3. Menerapkan keterampilan berpikir logis, kritis, dan sistematis dalam merancang *query* yang optimal pada studi kasus nyata
4. Menganalisis perbedaan performa eksekusi *query* sebelum dan sesudah penggunaan indeks, serta menyajikan hasil analisis secara tepat.

10.2 INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:

CPL 03-KU03	CPMK 3.3.3	1. Kemampuan mahasiswa meningkat dalam menjelaskan definisi, fungsi, dan cara kerja indeks dan agregasi pada MongoDB
CPL 03-KU03	CPMK 3.3.3	2. Kemampuan mahasiswa meningkat dalam merancang dan menjalankan <i>pipeline</i> agregasi dengan menggunakan stage \$match, \$project, \$group, \$sort, \$limit, dan \$unwind.

CPL 03- KU03	CPMK 3.3.3	3. Kemampuan mahasiswa meningkat dalam membandingkan hasil eksekusi query sebelum dan sesudah menggunakan indeks, serta menarik Kesimpulan dari perbedaan performa yang terjadi.
-----------------	---------------	--

10.3 TEORI PENDUKUNG

1. Indeks

Konsep indeks dalam basis data sama dengan indeks di dalam buku. Indeks merupakan jalan pintas yang berisi daftar berurutan yang merujuk pada konten yang ada dalam basis data. Tanpa menggunakan indeks, proses pencarian dengan *query* disebut dengan **collection scan**, yang berarti suatu server harus melakukan pencarian dari keseluruhan dokumen dalam sebuah *collection* untuk menemukan hasil *query*.

Untuk membuat indeks dalam sebuah *collection*, dapat digunakan *method* berikut ini:

Membuat indeks

```
db.<namaKoleksi>.createIndex({"<namaField>": 1})
```

Contoh 1:

Membuat indeks dari field username di collection users

```
db.<namaKoleksi>.createIndex({"<namaField>": 1})
```

Query berikut dapat digunakan untuk menjelaskan kinerja indeks.

Query

```
db.users.find({"username": "user101"}).explain("executionStats")
```

Dengan menggunakan indeks, pencarian mengacu pada tabel indeks yang terhubung dengan tabel. Dengan begitu, pencarian hanya dilakukan pada dokumen yang relevan sehingga waktu eksekusi dan jumlah dokumen yang ditelusuri akan berkurang sebagaimana yang ditunjukkan pada Gambar 10.1.

Dengan indeks

```
"executionStats": {  
  "executionSuccess": true,  
  "nReturned": 1,  
  "executionTimeMillis": 1,  
  "totalKeysExamined": 1,  
  "totalDocsExamined": 2,  
  ...}
```

Tanpa indeks

```
"executionStats": {  
  "executionSuccess": true,  
  "nReturned": 1,  
  "executionTimeMillis": 419,  
  "totalKeysExamined": 0,  
  "totalDocsExamined": 1000,  
  ...}
```

Gambar 10. 1 Perbandingan performa pencarian dengan dan tanpa indeks

Selain indeks tunggal, suatu *collection* juga dapat memiliki indeks komposit. Untuk membuat indeks komposit dapat digunakan *method* berikut ini.



Membuat Indeks Komposit

```
db.<namaKoleksi>.createIndex({"<namaField1>":1, "<namaField2>": 1})
```

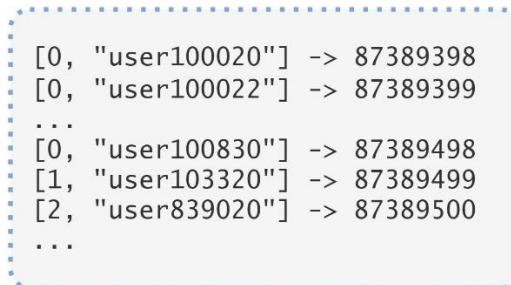
Contoh 2:

Membuat Indeks Komposit dari username dan age

```
db.users.createIndex({"age":1, "username": 1})
```

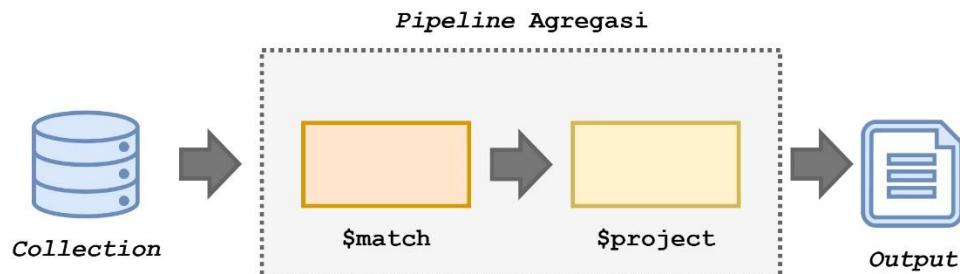
Apabila sebelumnya sudah ada indeks yang mengacu pada *field* `username` dan ada penambahan indeks baru dengan *field* lain, maka indeks baru akan dibuat.

Apabila dalam *collection* `users` dibuat indeks dengan `{"age": 1, "username": 1}`, bentuk indeks *collection* ini diilustrasikan pada Gambar 10.2 di bawah ini.



Gambar 10. 2 Bentuk indeks dengan `age` dan `username`

2. Agregasi



Gambar 10. 3 Ilustrasi agregasi

Agregasi merupakan proses mengolah data mentah menjadi informasi yang lebih ringkas dan bermakna, misalnya menghitung jumlah, rata-rata, nilai maksimum/minimum, atau mengelompokkan data berdasarkan kriteria tertentu. Pada SQL, agregasi dilakukan dengan perintah seperti `GROUP BY`, `SUM()`, `COUNT()`, dan sebagainya. Dalam MongoDB, agregasi dilakukan dengan *pipeline* yang lebih fleksibel dan memungkinkan data diproses secara bertahap.

Aggregation pipeline adalah mekanisme pemrosesan data di MongoDB yang bekerja dengan prinsip aliran data bertahap (*pipeline*). Data dokumen diproses melalui serangkaian *stage* (tahapan), di mana setiap tahapan menerima input dari tahap sebelumnya, melakukan operasi tertentu, lalu meneruskan hasilnya ke tahap berikutnya. *Pipeline* ini memungkinkan analisis data kompleks dengan cara yang modular dan efisien.



Beberapa *stage* dalam pipeline ditunjukkan pada Tabel 10.1.

Tabel 10. 1 State untuk membuat pipeline

<code>\$match</code>	Menyaring dokumen berdasarkan kondisi tertentu (seperti WHERE di SQL)
<code>\$group</code>	Mengelompokkan dokumen berdasarkan satu atau lebih <i>field</i> , lalu menerapkan fungsi agregasi seperti <code>\$sum</code> , <code>\$avg</code> , <code>\$max</code> , <code>\$min</code> (seperti GROUP BY)
<code>\$project</code>	Memilih <i>field</i> tertentu untuk ditampilkan, membuat <i>field</i> baru, atau memformat hasil keluaran (seperti SELECT)
<code>\$sort</code>	Mengurutkan dokumen berdasarkan <i>field</i> tertentu (seperti ORDER BY)
<code>\$limit</code> dan <code>\$skip</code>	Membatasi jumlah dokumen yang ditampilkan (seperti LIMIT); Melewati sejumlah dokumen
<code>\$lookup</code>	Melakukan operasi mirip JOIN antar koleksi



Gambar 10. 4 Sintaks agregasi

Untuk mengefisienkan pencarian, selalu tambah indeks pada kolom atau *field* yang akan diurutkan nilainya.

Contoh 3:

Pipeline untuk menampilkan *field* `name` dan `founded_year` dari dokumen dalam *collection* `companies` yang memiliki nilai `founded_year = 2004` dengan `$match` dan `$project`

```
db.companies.aggregate([
    {$match: {founded_year: 2004}},
    {$project: {
        _id: 0,
        name: 1,
        founded_year: 1
    }}
])
```

Contoh 4:

Pipeline untuk menampilkan *field* `name` dan `founded_year` dari dokumen dalam *collection* `companies` yang memiliki nilai `founded_year = 2004` dengan `$match`, `$project`, dan `$limit`

```
db.companies.aggregate([
    {$match: {founded_year: 2004}},
    {$limit: 5},
    {$project: {
        _id: 0,
        founded_year: 1
    }}
])
```

```
    }  
])
```

Contoh 5:

Pipeline untuk menampilkan `field name`, `ipo`, `valuation`, dan `funders` dari dokumen dalam `collection companies` yang memiliki nilai pendanaan dari organisasi dengan `permalink` "greylock" dengan `$match` dan `$project`

```
db.companies.aggregate([
  {$match: {"funding_rounds.investments.financial_org permalink": "greylock"}},
  {$project: {
    _id: 0,
    name: 1,
    ipo: "$ipo.pub_year",
    valuation: "$ipo.valuation_amount",
    funders: "$funding_rounds.investments-financial_org permalink"
  }}
])
```

Tahapan `$project` digunakan untuk mengatur ulang dan membentuk kembali *output* dokumen.

<code>"_id": 0</code>	Menghilangkan <i>field</i> <code>_id</code> dari output
<code>"name": 1</code>	Menampilkan nama perusahaan
<code>"ipo": "\$ipo.pub_year"</code>	Menampilkan tahun IPO (initial public offering) dari <i>field</i> <code>ipo.pub_year</code>
<code>"valuation": "\$ipo.valuation_amount"</code>	Menampilkan nilai valuasi IPO
<code>"funders": "\$funding_rounds.investments.financial_org permalink"</code>	Menampilkan daftar investor (<code>funders</code>) dalam bentuk <i>array</i> berdasarkan <i>field</i> <code>permalink</code>

10.4 HARDWARE DAN SOFTWARE

Hardware dan software yang digunakan dalam praktikum ini yaitu:

1. MongoDB Compass

10.5 PRE-TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL 03-	CPMK	Jelaskan pengertian dari indeks!	25

	KU03	3.3.3		
2.	CPL 03-KU03	CPMK 3.3.3	Jelaskan cara kerja indeks dan cara kerjanya dalam meningkatkan performa query!	25
3.	CPL 03-KU03	CPMK 3.3.3	Berikan contoh kasus sederhana yang memerlukan penggunaan indeks komposit!	25
4.	CPL 03-KU03	CPMK 3.3.3	Apa perbedaan peran antara stage \$match dan \$group dalam pipeline agregasi?	25

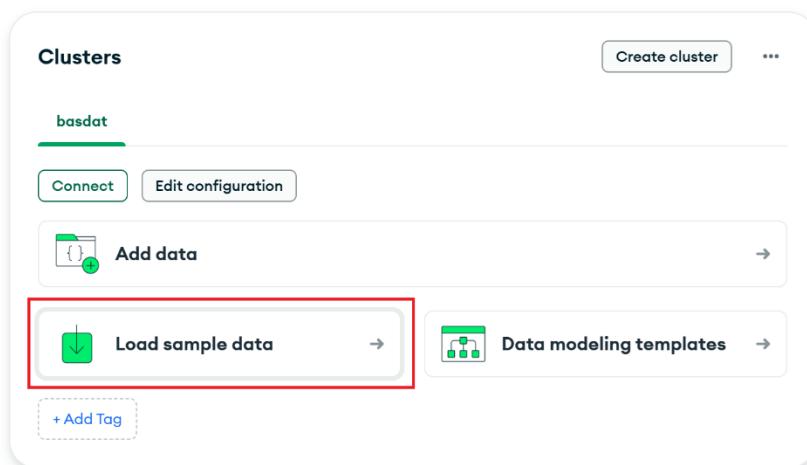
10.6 LANGKAH PRAKTIKUM

Memilih Sample Dataset di MongoDB Atlas (Pra Praktikum)

1. Buka MongoDB Atlas dan klik “Load sample data” di halaman Overview

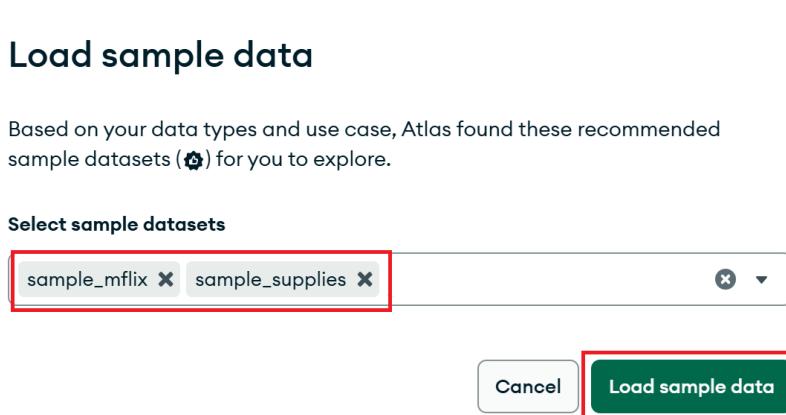
NINDA'S ORG - 2024-12-31 > PROJECT 0

Overview

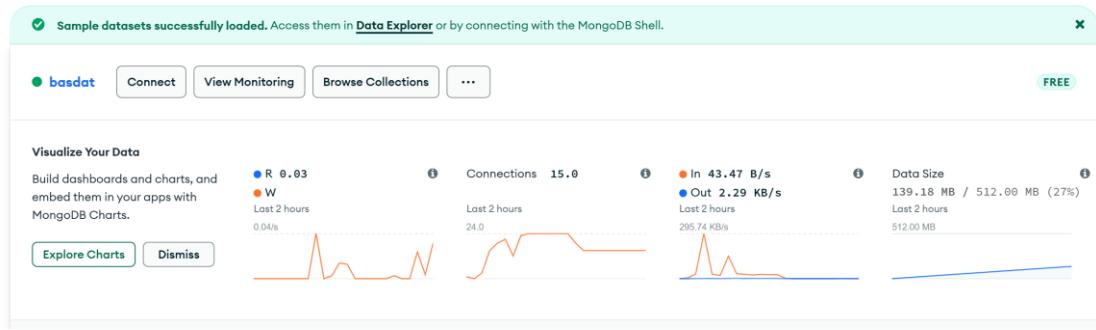


Gambar 10. 5 Pilih "Load sample data"

2. Masukkan “sample_mflix” dan “sample_supplies” di search bar lalu klik “Load sample data”



Gambar 10. 6 Memilih "sample_mflix" dan "sample_supplied"

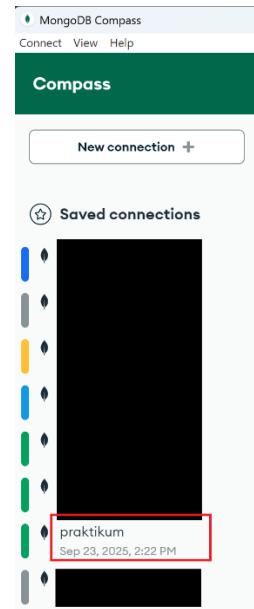


Gambar 10. 7 Sample dataset berhasil dimuat

Tunggu sampai data sampel selesai dimuat. Usahakan langkah ini sudah selesai dilakukan sebelum praktikum dimulai.

Memuat Sample Dataset di MongoDB Compass

1. Pilih connection “Praktikum” (yang sudah dibuat di pertemuan ke-9). Pastikan connection string sudah benar lalu klik “Connect”



Gambar 10. 8 Membuka koneksi Praktikum

2. Buka collection “movies” dari database “sample_mflix”

The screenshot shows the MongoDB Compass interface. On the left, there's a sidebar with 'My Queries' and 'Databases'. Under 'Databases', there are several collections: 'admin', 'basidata', 'config', 'local', 'sample_mflix' (which is expanded), 'comments', 'embedded_movies', 'movies' (highlighted with a red box), 'sessions', 'theaters', and 'users'. At the top, there's a 'Collections' bar with 'Create collection', 'View', and sorting options ('Sort by Collection Name'). Below the sidebar, the 'routes' collection is shown with its storage size (2.01 MB), document count (67 K), average document size (184.00 B), index count (1), and total index size (2.16 MB).

Gambar 10. 9 Membuka collection movies

Tugas 1: Menampilkan judul film, tahun rilis, dan durasi dari lima film tahun 2000 dengan durasi terlama

1. Klik tab “Aggregations” lalu klik tombol “+ Add Stage” untuk menambah stage baru

The screenshot shows the MongoDB Compass interface for the 'sample_mflix.movies' collection. The 'Aggregations' tab is selected (highlighted with a green box). Below it, the pipeline area says "Your pipeline is currently empty. To get started add the first stage." There are buttons for 'Explain', 'Export', 'Run', and 'More Options'. At the bottom, there's a 'PREVIEW' button, a 'STAGES' button (highlighted with a red box), and a 'TEXT' button. A large preview window shows three document snippets. At the bottom of the pipeline area, there's a green button labeled '+ Add Stage' (highlighted with a red box). Below the pipeline, there's a link to "Learn more about aggregation pipeline stages".

Gambar 10. 10 Menambah stage baru

2. Untuk menampilkan film yang dirilis tahun 2000, tambah stage `$match` lalu masukkan `year: 2000`

The screenshot shows the MongoDB Aggregation Pipeline interface. At the top, it says "sample_mflix.movies" with "21.3k DOCUMENTS" and "1 INDEXES". Below this, the "Aggregations" tab is selected. The pipeline stage is set to "\$match". The query editor contains the following MQL code:

```

1 /**
2  * query: The query in MQL.
3 */
4 {
5     year: 2000
6 }

```

The output after the \$match stage shows a sample of 10 documents. One document is highlighted with a red box around its year field value "2000". A red box also highlights the "\$match" stage in the pipeline. A red box highlights the text "Menampilkan dokumen yang sesuai query" (Displaying documents that match the query) in the preview pane.

Gambar 10. 11 Memfilter dengan tahun

Hasil pencarian berupa dokumen yang sesuai *query* akan ditampilkan di kolom sebelah kanan.

- Untuk menampilkan tahun rilis, nama film, dan durasi film saja, tambahkan stage `$project` berikut

The screenshot shows the MongoDB Aggregation Pipeline interface. The "sample_mflix.movies" database is selected. The "Aggregations" tab is selected. The pipeline stage is set to "\$project". The query editor contains the following MQL code:

```

1 /**
2  * specifications: The fields to
3  * include or exclude.
4 */
5 {
6     _id: 0,
7     title: 1,
8     year: 1,
9     runtime: 1
10}

```

The output after the \$project stage shows a sample of 10 documents. Two documents are highlighted with red boxes around their year, title, and runtime fields. A red box highlights the "\$project" stage in the pipeline. A red box highlights the text "Menampilkan dokumen yang sesuai query" (Displaying documents that match the query) in the preview pane.

Gambar 10. 12 Mengatur *field* yang ditampilkan

- Urutkan hasil pencarian dari film dengan durasi terlama sampai durasi tersingkat dengan menambah stage `$sort`

The screenshot shows the Apache Nifi interface with the following details:

- Top Bar:** The title is "sample_mflix.movies". On the right, there are statistics: "21.3k DOCUMENTS" and "1 INDEXES".
- Header:** The "Aggregations" tab is selected. Other tabs include "Documents", "Schema", "Explain Plan", "Indexes", and "Validation".
- Pipeline View:** A pipeline diagram with three main components:
 - \$match:** A green rounded rectangle.
 - \$project:** A blue rounded rectangle.
 - \$sort:** A red rounded rectangle.Buttons above the pipeline: "Pipeline" (with a dropdown), "\$match", "\$project", "\$sort", "EXPLAIN", "EXPORT", "Run", and "More Options".
- Document Preview:** Untitled - modified. Buttons: "SAVE" (with a dropdown), "+ CREATE NEW", and "EXPORT TO LANGUAGE".
 - Left Panel:** Shows the JSON document structure:

```
2  * specifications: The fields to
3  * include or exclude.
4  */
5  [
6    _id:0,
7    title:1,
8    year:1,
9    runtime:1
10 ]
```
 - Middle Panel:** Shows the output after the \$sort stage:

```
year: 2000
title: "In the Mood for Love"
runtime: 98
```
 - Right Panel:** Shows the output after the \$sort stage:

```
year: 2000
title: "State and Main"
runtime: 105
```
- Sort Stage Configuration:** A modal window titled "Output after \$sort stage (Sample of 10 documents)". It shows the sort expression: "\$sort" and the field path: "runtime: -1".
 - Left Panel:** Shows the JSON document structure:

```
1 /**
2  * Provide any number of field/order pairs.
3  */
4  [
5    runtime: -1
6  ]
```
 - Middle Panel:** Shows the sorted output:

```
runtime: 345
title: "La Commune (Paris, 1871)"
year: 2000
```
 - Right Panel:** Shows the sorted output:

```
runtime: 240
title: "The Beach Boys: An American Family"
year: 2000
```

Gambar 10. 13 Mengurutkan dokumen hasil pencarian berdasarkan runtime

Nilai -1 digunakan untuk mengurutkan dokumen berdasarkan *field runtime* dari yang nilainya terbesar ke terkecil, sedangkan nilai 1 digunakan untuk mengurutkan nilai dari terkecil ke terbesar.

5. Untuk menampilkan lima film dengan durasi terlama, tambahkan *field \$limit*

The screenshot shows the Apache Nifi interface with the following details:

- Header:** sample_mflix.movies, 21.3k DOCUMENTS, 1 INDEXES
- Tab Bar:** Documents, Aggregations (selected), Schema, Explain Plan, Indexes, Validation
- Pipeline Stage:** Pipeline → \$match → \$project → \$sort → \$limit
- Buttons:** Explain, Export, Run, More Options
- Untitled - modified:** SAVE, CREATE NEW, EXPORT TO LANGUAGE
- Preview Buttons:** PREVIEW, STAGES, TEXT
- Stage Details:** \$limit stage with a dropdown menu showing code: 1. `/**`, 2. `* Provide the number of documents to limit.`, 3. `/`, 4. `5`. The input value `5` is highlighted with a red box.
- Output Preview:** Output after \$limit stage (Sample of 5 documents). It shows a sample of 5 documents:
 - runtime: 345
 - title: "La Commune (Paris, 1871)"
 - year: 2000
- Right Preview:** A second preview window showing:
 - runtime: 240
 - title: "The Beach Boys: An American Family"
 - year: 2000

Gambar 10. 14 Membatasi jumlah dokumen yang ditampilkan

6. Untuk menyimpan hasil pencarian, klik tombol “SAVE” lalu pilih “Save As”.

sample_mflix.movies

21.3k DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

Pipeline \$match \$project \$sort \$limit

Untitled - modified

SAVE + CREATE NEW EXPORT TO LANGUAGE PREVIEW STAGES TEXT

21349 Documents in the collection

Save as Create view

Preview of document

awards: Object lastupdated: "2015-08-13 00:46:30.660000000" year: 1999 imbd: Object countries: Array type: "movie" tomatoes: Object num_mflix_comments: 0

rated: "G" awards: Object lastupdated: "2015-08-13 00:46:30.660000000" year: 1999 imbd: Object countries: Array type: "movie" tomatoes: Object

_id: ObjectId('573a1390f29313caabcd4803') plot: "Cartoon figures announce, via com strip balloons, that they will mov genres: Array runtime: 7 cast: Array num_mflix_comments: 0 poster: "https://m.media-amazon.com/images/I/51O..."

Gambar 10. 15 Menyimpan pipeline

- Beri nama *pipeline*, misal dengan “1. Lima film dengan durasi terpanjang”



Gambar 10. 16 Memberi nama pipeline

Tugas 2: Menampilkan judul film sebagai `judul_film`, jumlah kemenangan sebagai `perolehan_penghargaan` dari film yang memiliki rating lebih dari 7,5 dan ber-genre “Western”

- Untuk membuat *pipeline* baru, klik “CREATE NEW” lalu “Confirm”

sample_mflix.movies

sample_mflix.movies

21349 Documents in the collection

Documents Aggregations Schema Explain Plan Indexes Validation

Pipeline \$match \$project \$sort \$limit

1. Lima film de... **CREATE NEW** EXPORT TO LANGUAGE EXPLAIN

Gambar 10. 17 Membuat pipeline baru

Are you sure you want to create a new pipeline?

Creating this pipeline will abandon unsaved changes to the current pipeline.

Cancel **Confirm**

Gambar 10. 18 Konfirmasi membuat *pipeline* baru

2. Data *rating* sebuah film disimpan dalam object bernama `imdb`, tepatnya di *field* `rating` (`imdb.rating`). Untuk mencari film yang *rating*-nya lebih dari 7,5, dapat digunakan operator `$gt` yang berarti greater than atau lebih dari. Adapun data *genre* suatu film disimpan dalam *array* yang bernama `genres`.

Untuk membuat filter yang menampilkan film dengan *rating* lebih dari 7,5 dan ber-*genre* "Western" saja, dapat ditambahkan *stage* `$match` berikut

The screenshot shows the MongoDB Aggregations interface for a collection named `sample_mflix.movies`. The interface includes tabs for Documents, Aggregations (selected), Schema, Explain Plan, Indexes, and Validation. At the top right, it shows 21.3k DOCUMENTS and 1 INDEXES. Below the tabs, there are buttons for Pipeline, Explain, Export, Run, and More Options. The Pipeline section shows a single stage: `$match`. The query is defined as follows:

```

1 /**
2 * query: The query in MQL.
3 */
4 [
5   {
6     "imdb.rating": {$gt: 7.5 },
7     "genres": "Western"
8   }
]
  
```

The output after the `$match` stage is shown as a sample of 10 documents. One document is expanded to show its fields:

```

_id: ObjectId('573a1392f29313caabcbdaef')
plot: "Kent, the unscrupulous boss of Bottleneck has Sheriff Keogh killed when he finds out Kent's been embezzling money from the town's pension fund."
genres: Array
  0: "Comedy"
  1: "Western"
runtime: 94
rated: "APPROVED"
cast: Array
  poster: "https://m.media-amazon.com/images/M/MVSBMTU3NjE5NzYtYT...  
```

Gambar 10. 19 Kondisi di *stage* `$match`

3. Untuk menampilkan beberapa *field* saja dan memberi alias di *field* yang ditampilkan dalam hasil pencarian, tambahkan *stage* `$project` berikut:

The screenshot shows the MongoDB Compass interface for the collection 'sample_mflix.movies'. At the top right, it displays '21.3k DOCUMENTS' and '1 INDEXES'. Below the header, there are tabs for 'Documents', 'Aggregations' (which is selected), 'Schema', 'Explain Plan', 'Indexes', and 'Validation'. Under the 'Aggregations' tab, a pipeline editor is shown with stages: '\$match', '\$project' (highlighted with a red box), '\$sort', and '\$group'. The '\$project' stage has a dropdown menu set to '\$project' and a preview section showing sample documents. One document's output is highlighted with a red box, showing the field '_id: 0'.

Gambar 10. 20 Pemberian nama *field* dengan \$project

- Simpan hasilnya dan beri nama “2. Jumlah penghargaan film”

Tugas 3: Menampilkan top 10 genre film yang dirilis tahun 1990an

- Untuk membuat *pipeline* baru, klik “CREATE NEW” lalu “Confirm”

The screenshot shows the MongoDB Compass interface for the collection 'sample_mflix.movies'. At the top right, it displays '21349 Documents in the collection'. Below the header, there are tabs for 'Documents', 'Aggregations' (selected), 'Schema', 'Explain Plan', 'Indexes', and 'Validation'. Under the 'Aggregations' tab, a pipeline editor is shown with stages: '\$match' (highlighted with a red box), '\$sort', and '\$limit'. The '\$match' stage has a dropdown menu set to '\$match' and a preview section showing sample documents. One document's output is highlighted with a red box, showing the field '_id: 0'.

Gambar 10. 21 Membuat pipeline baru

- Untuk menampilkan film yang dirilis tahun 1990an saja, tambahkan *stage* \$match

The screenshot shows the MongoDB Compass interface for the collection 'sample_mflix.movies'. At the top right, it displays '21349 Documents in the collection'. Below the header, there are tabs for 'Documents', 'Aggregations' (selected), 'Schema', 'Explain Plan', 'Indexes', and 'Validation'. Under the 'Aggregations' tab, a pipeline editor is shown with stages: '\$match' (highlighted with a red box), '\$sort', and '\$group'. The '\$match' stage has a dropdown menu set to '\$match' and a preview section showing sample documents. One document's output is highlighted with a red box, showing the field '_id: 0'.

Gambar 10. 22 Membuat filter berdasarkan tahun

Operator `$gte` digunakan untuk memilih film dengan nilai `year` yang lebih besar atau sama dengan 1990, sedangkan operator `$lt` digunakan untuk memilih film dengan nilai `year` yang kurang dari 2000.

- Genre film disimpan dalam bentuk *array*. Untuk memecah data dari array menjadi dokumen baru per elemen, tambahkan *stage* \$unwind

```

1  /**
2   * path: Path to the array field.
3   * includeArrayIndex: Optional name for index.
4   * preserveNullAndEmptyArrays: Optional
5   *   toggle to unwind null and empty values.
6  */
7  {
8    path: "$genres"
9  }

```

Gambar 10. 23 Memecah dokumen berdasarkan nilai *genres*

Jika satu film memiliki banyak *genre*, misal film “The House of Smiles” awalnya memiliki dua *genre* berikut:

```

▶ _id: ObjectId('573a1398f29313caabcead6a')
  plot: "A septuagenarian couple are attracted to each other in a retirement ho..."
  ▶ genres: Array
    0: "Drama"
    1: "Romance"
  runtime: 110
  ▶ cast: Array
    title: "The House of Smiles"
    fullplot: "Adelina, a former "Miss Smiles" beauty queen, is a resident at a senio..."
  ▶ languages: Array
    released: 1991-03-01T00:00:00.000+00:00
  ▶ directors: Array
  ▶ writers: Array
  ▶ awards: Object
    lastupdated: "2015-08-13 00:06:59.537000000"
    year: 1991

```

Gambar 10. 24 Bentuk awal dokumen (*genres* berupa *array*)

Di stage `$unwind`, dokumen ini akan diubah menjadi dua dokumen yang masing-masing memiliki nilai *genre* tunggal (*single-valued* dan bukan *array*)

```

_id: ObjectId('573a1398f29313caabcead6a')
plot: "A septuagenarian couple are attracted to
each other in a retirement ho..."
genres: "Drama"
runtime: 110
cast: Array
title: "The House of Smiles"
fullplot: "Adelina, a former "Miss Smiles"
beauty queen, is a resident at a

```

```

_id: ObjectId('573a1398f29313caabcead6a')
plot: "A septuagenarian couple are attracted
each other in a retirement ho..."
genres: "Romance"
runtime: 110
▶ cast: Array
title: "The House of Smiles"
fullplot: "Adelina, a former "Miss Smiles"
beauty queen, is a resident at a

```

Gambar 10. 25 Bentuk akhir dokumen (menjadi dua dokumen karena dokumen awal memiliki dua *genre*)

4. Tambahkan stage `$group` untuk mengelompokkan hasil berdasarkan *genre*, lalu hitung jumlah film masing-masing *genre* dengan `$sum`.

```

1 /**
2  * _id: The id of the group.
3  * fieldN: The first field name.
4 */
5 {
6   _id: "$genres",
7   total_film: {
8     $sum: 1
9   }
10

```

Output after \$group stage (Sample of 10 documents)

```

_id: "War"
total_film: 80

```

Gambar 10. 26 Pengelompokan berdasarkan *genre* film dan menghitung jumlah film masing-masing *genre*

Stage ini mengelompokkan dokumen berdasarkan nilai *field genres*

- `_id : "$genres"` mengatur *genre* sebagai kunci pengelompokan
- `jumlah_film: { $sum: 1 }` menghitung jumlah dokumen pada setiap grup dengan menambahkan nilai 1 untuk setiap penambahan dokumen dalam grup

5. Tambahkan *stage* \$sort untuk mengurutkan *genre* berdasarkan film terbanyak

```

1 /**
2  * Provide any number of field/order pairs.
3 */
4 {
5   total_film: -1
6 }

```

Output after \$sort stage (Sample of 10 documents)

```

_id: "Drama"
total_film: 2165

```

Gambar 10. 27 Mengurutkan daftar *genre* berdasarkan jumlah film

Nilai `-1` berarti pengurutan dilakukan dari nilai terbesar ke terkecil.

6. Tambahkan *stage* \$limit untuk mengambil 10 *genre* dengan film terbanyak

```

1 /**
2  * Provide the number of documents to limit.
3 */
4 10

```

Output after \$limit stage (Sample of 10 documents)

```

_id: "Drama"
total_film: 2165

```

Gambar 10. 28 Membatasi jumlah dokumen yang ditampilkan

7. Tambahkan *stage* \$project untuk mengatur *field* apa saja yang ditampilkan dan untuk mengubah nama *field* yang ditampilkan.

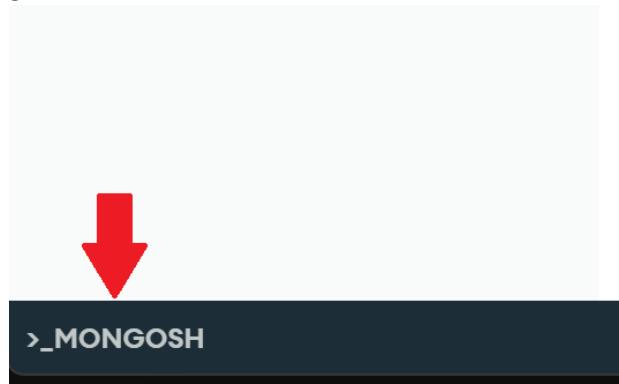
Gambar 10. 29 Mengatur penamaan *field* yang ditampilkan

Stage ini menentukan *field* mana yang ditampilkan di *output*.

- `_id: 0` menyembunyikan field `_id` bawaan MongoDB
 - `genre_film: "$_id"` mengganti nama *field* `_id` hasil pengelompokan menjadi `genre_film`
 - `jumlah_film: 1` menampilkan field `jumlah_film` tanpa perubahan
8. Simpan *pipeline* ini dengan nama “3. Top 10 genre film”

Tugas 4: Membuat *pipeline* agregasi untuk mnghitung jumlah film yang dirilis tahun 90an dengan *shell*

- Buka *shell* dalam MongoDB



Gambar 10. 30 Membuka Mongo *Shell*

- Masukkan `use sample_mflix` lalu tekan Enter untuk memilih *collection*

```
> use sample_mflix
< 'switched to db sample_mflix'
Atlas atlas-110f84-shard-0 [primary] sample_mflix>
```

Gambar 10. 31 Memilih *database* yang akan digunakan

Perintah `use <nama_koleksi>` mengubah *collection/database* aktif yang digunakan.

- Hitung jumlah film yang dirilis tahun 1990an dengan membuat *pipeline* agregasi

```
Pipeline agregasi
db.movies.aggregate([
  { $match: { year: { $gte: 1990, $lt: 2000 } } },
  { $count: "jumlah_film_90an" }
```

])

```
> db.movies.aggregate([
    { $match: { year: { $gte: 1990, $lt: 2000 } } },
    { $count: "jumlah_film_90an" }
])
< {
    jumlah_film_90an: 3558
}
```

Gambar 10. 32 Output dari pipeline agregasi

Tugas 5: Membuat indeks di collection movies

1. Lakukan pencarian film yang dirilis tahun 2005 dengan:

Pencarian tanpa indeks

```
db.movies.find({ "year": 2005 }).explain("executionStats")
```

```
executionStages: {
    isCached: false,
    stage: 'COLLSCAN',
    filter: {
        year: {
            '$eq': 2005
        }
    },
    nReturned: 713,
    executionTimeMillisEstimate: 17,
    works: 21350,
    advanced: 713,
    needTime: 20636,
    needYield: 0,
    saveState: 1,
    restoreState: 1,
    isEOF: 1,
    direction: 'forward',
    docsExamined: 21349
}
```

Gambar 10. 33 Performa pencarian tanpa indeks

Penjelasan:

- stage: "COLLSCAN"** berarti MondoDB melakukan *collection scan*, yaitu membaca dokumen satu per satu, lalu memfilter yang memiliki nilai **year = 2005**.
- nReturned: 713** berarti ada 713 dokumen yang cocok dengan filter
- totalDocsExamined: 21349** artinya seluruh dokumen dalam koleksi dibaca
- executionTimeMillisEstimate: 17** menunjukkan waktu eksekusi dalam milisekon

2. Menambah indeks baru dari *field year*

```
> db.movies.createIndex({year:1})
< 'year_1'
```

Gambar 10. 34 Berhasil menambahkan indeks

3. Ulangi pencarian di langkah 2 dan amati outputnya

```
winningPlan: {
  isCached: false,
  stage: 'FETCH',
  inputStage: {
    stage: 'IXSCAN',
    keyPattern: {
      year: 1
    },
    indexName: 'year_1',
    isMultiKey: false,
    multiKeyPaths: {
      year: []
    },
    isUnique: false,
    isSparse: false,
    isPartial: false,
    indexVersion: 2,
    direction: 'forward',
    indexBounds: {
      year: [
        '[2005, 2005]'
      ]
    }
  },
}
```

Gambar 10. 35 Performa pencarian setelah ada indeks



```

executionStats: {
  executionSuccess: true,
  nReturned: 713,
  executionTimeMillis: 2,
  totalKeysExamined: 713,
  totalDocsExamined: 713,
}

```

Gambar 10. 36 Performa pencarian setelah ada indeks

- Query* memilih Index Scan (IXSCAN) berdasarkan index `year_1`, lalu FETCH dokumen yang sesuai saja
- `indexBounds: [" [2005, 2005] "]` artinya MongoDB hanya membaca index untuk `year = 2005`
- `nReturned: 713` menunjukkan hasil yang sama dengan langkah 2
- `totalKeysExamined: 713` menunjukkan hanya 713 dokumen yang dibaca
- `executionTimeMillis: 2` menunjukkan waktu eksekusi yang jauh lebih cepat dari langkah 2

Tabel 10. 2 Perbandingan performa pencarian sebelum dan sesudah ada indeks

Metriks	Sebelum Indeks	Sesudah Indeks
<code>stage</code>	COLLSCAN	IXSCAN + FETCH
<code>totalDocsExamined</code>	21.349	713
<code>totalKeysExamined</code>	0	713
<code>executionTimeMillis</code>	17 ms	2 ms

Aturan Penilaian (Total Skor: 100):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL 03-KU03-	CPMK 3.3.3	Ikuti langkah-langkah praktikum mulai dari Tugas 1 - 5	Jawaban hasil praktek (Screenshot)	100

10.7 POST-TEST

Jawablah pertanyaan berikut (Total Skor: 100):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor

1.	CPL 03- KU03	CPMK 3.3.3	<p>1. Gunakan <i>collection sales</i> dari <i>database sample_supplies</i>.</p> <p>2. Hitung jumlah total kuantitas per barang yang terjual di setiap lokasi toko</p> <pre><code>_id: "Denver" ▼ penjualan: Array ▼ 0: Object product: "binder" quantity: 7986 ▼ 1: Object product: "pens" quantity: 4188 ▼ 2: Object product: "laptop"</code></pre>	Jawaban hasil praktik disamping (<i>screenshot</i>)	100
----	--------------------	------------	--	---	-----

10.8 HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL 03- KU03	CPMK 3.3.3.	20%		
2.	Praktik	CPL 03- KU03	CPMK 3.3.3	30%		
3.	Post-Test	CPL 03- KU03	CPMK 3.3.3	50%		
Total Nilai						

PRAKTIKUM 11: NoSQL - PEMROGRAMAN DENGAN PYTHON

Pertemuan ke : 11

Total Alokasi Waktu : 180 menit

- Materi : 30 menit
- Pre-Test : 30 menit
- Praktikum : 60 menit
- Post-Test : 60 menit

Total Bobot Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 30 %
- Post-Test : 50 %

Pemenuhan CPL dan CPMK:

CPL 03-KU03	Mempunyai kemampuan dalam mendefinisikan kebutuhan pengguna atau pasar terhadap kinerja (menganalisis, mengevaluasi dan mengembangkan) algoritma/metode berbasis komputer untuk mengambil keputusan secara tepat dalam konteks penyelesaian masalah di bidang Rekayasa Perangkat Lunak serta Data dan Sistem Cerdas maupun bidang lainnya, berdasarkan hasil analisis informasi dan data.
CPMK 3.3.3	Mahasiswa mampu mengimplementasikan query basis data dan lingkungannya

11.2 DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu memahami:

5. Menerapkan konsep teoritis bidang informatika terkait pemrograman basis data NoSQL (khususnya MongoDB) dan pemanfaatan bahasa pemrograman Python untuk memodelkan serta menyelesaikan masalah.
6. Berpikir logis, kritis, sistematis, dan inovatif dalam menggunakan driver *PyMongo* untuk menghubungkan aplikasi Python dengan MongoDB serta melakukan operasi dasar basis data.
7. Memilih, membuat, dan menerapkan teknik pemrograman NoSQL (CRUD, pengolahan data dengan Python, hingga *data modeling*) dengan menggunakan sumber daya dan perangkat modern untuk memecahkan masalah terkait MongoDB.

11.3 INDIKATOR KETERCAPAIAN PEMBELAJARAN

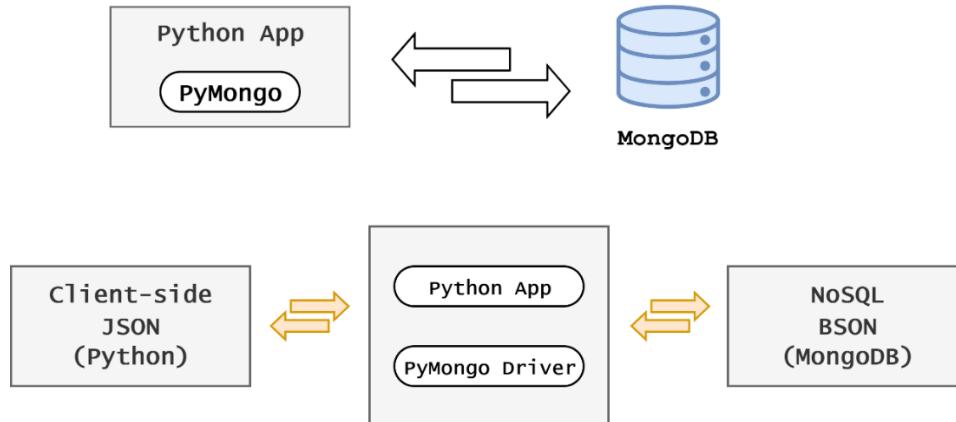
Indikator ketercapaian diukur dengan:

CPL 03-KU03	CPMK 3.3.3	1. Kemampuan mahasiswa meningkat dalam memahami konsep dasar pemrograman NoSQL menggunakan Python, ditunjukkan dengan keberhasilan menghubungkan Python ke MongoDB melalui <i>driver PyMongo</i>
-------------	------------	--

CPL 03- KU03	CPMK 3.3.3	2. Kemampuan mahasiswa meningkat dalam menganalisis, memilih, dan mengimplementasikan operasi CRUD (Create, Read, Update, Delete) pada MongoDB dengan Python
-----------------	---------------	--

11.4 TEORI PENDUKUNG

1. NoSQL Programming dengan Python



Gambar 11. 1 Ilustrasi

Python dapat digunakan untuk berinteraksi dengan MongoDB melalui *library PyMongo*. Pustaka ini berfungsi sebagai *driver* yang menjembatani aplikasi Python dengan MongoDB sehingga memungkinkan dilakukannya berbagai operasi basis data.

Alur komunikasi antara Python dan MongoDB adalah sebagai berikut:

- Python Application: aplikasi yang ditulis menggunakan Python.
- PyMongo: *library* yang berperan sebagai *driver* untuk mengirim perintah dari Python ke MongoDB.
- MongoDB (*Server*): basis data NoSQL yang menyimpan data dalam bentuk dokumen BSON.

Dengan arsitektur ini, Python dapat mengelola data dalam MongoDB seolah-olah bekerja dengan *dictionary* atau struktur data JSON

Untuk dapat menggunakan MongoDB melalui Python diperlukan:

- PyMongo: *library* utama untuk koneksi ke MongoDB.
- DNSPython: *library* tambahan yang dibutuhkan ketika menggunakan MongoDB Atlas dengan koneksi berbasis `URI srv`.

2. Operasi Dasar CRUD

Dalam MongoDB, *database* dan koleksi dibuat secara *lazy*, artinya tidak ada perintah pembuatan *database* dan koleksi (*collection*) yang langsung dieksekusi setelah perintah ditulis. Proses pembuatan baru dilakukan ketika dokumen pertama ditambahkan atau dengan kata lain meskipun perintah pembuatan *database* atau koleksi sudah dieksekusi, MongoDB belum menyimpan perubahannya ke server sampai ada data yang *di-insert* ke sana. Oleh karena itu, biasanya tiga perintah utama untuk membuat *database*, koleksi, dan menambahkan dokumen dijalankan secara bersamaan.



Konsep CRUD adalah inti dari manipulasi data dalam basis data NoSQL:

- a. Create: menambahkan dokumen baru ke dalam koleksi.
- b. Read: membaca dan menampilkan dokumen dari koleksi.
- c. Update: memodifikasi isi dokumen yang sudah ada.
- d. Delete: menghapus dokumen dari koleksi.

MongoDB secara otomatis membuat basis data dan koleksi ketika dokumen pertama disimpan.

3. Representasi Data

Data dalam MongoDB disimpan dalam bentuk BSON yang menyerupai JSON. Di sisi Python, struktur data yang paling mendekati adalah *dictionary*. Kesamaan format ini mempermudah integrasi antara Python dan MongoDB

Contoh:

- a. BSON/JSON di MongoDB → {"nama_item": "iPhooone 13", "harga": 12800000}
- b. Dictionary di Python → {" nama_item ": "iPhooone 13", "harga": 12800000}

11.5 HARDWARE DAN SOFTWARE

Hardware dan software yang digunakan dalam praktikum ini yaitu:

2. MongoDB Compass
3. Visual Studio Code
4. Browser

11.6 PRE-TEST

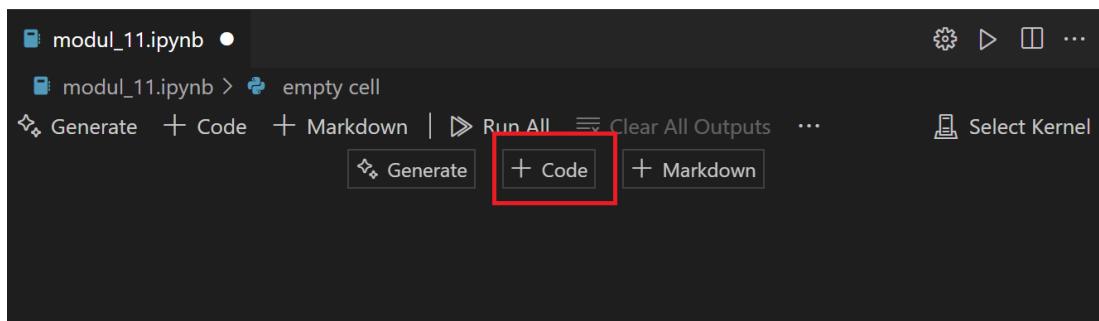
Jawablah pertanyaan berikut (**Total Skor: 100**):

N o	CPL	CPMK	Pertanyaan	Skor
1 .	CPL 03- KU03	CPMK 3.3.3	1. Jelaskan fungsi <i>library</i> PyMongo dalam pemrograman Python untuk mengakses MongoDB!	50
2 .	CPL 03- KU03	CPMK 3.3.3	2. Data di MongoDB disimpan dalam format BSON. Mengapa format ini lebih sesuai dibandingkan format JSON murni dalam konteks basis data NoSQL?	50

11.7 LANGKAH PRAKTIKUM

Tugas 1: Memasang *Library* yang Dibutuhkan

3. Dalam sebuah *folder* (beri nama “NIM_modul_11”, contoh: 123456_modul_11), buat satu *file* bernama modul_11.ipynb. Klik “+ Code” untuk menambah sel baru.



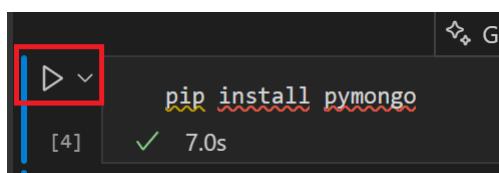
Gambar 11. 2 Menambah sel (*cell*) baru untuk menulis kode

- Untuk menggunakan MongoDB melalui Python, pasang PyMongo dengan perintah berikut:

Memasang PyMongo

```
pip install pymongo
```

Tekan *icon run* untuk menjalankan *cell* atau tekan Ctrl+ENTER



Gambar 11. 3 Menjalankan kode di satu sel

- Untuk menghubungkan Python ke MongoDB Atlas dengan `srv URI`, pasang DNSPython dengan perintah berikut.

Memasang DNSPython

```
pip install dnspython
```

- Cek versi python yang terpasang dengan perintah

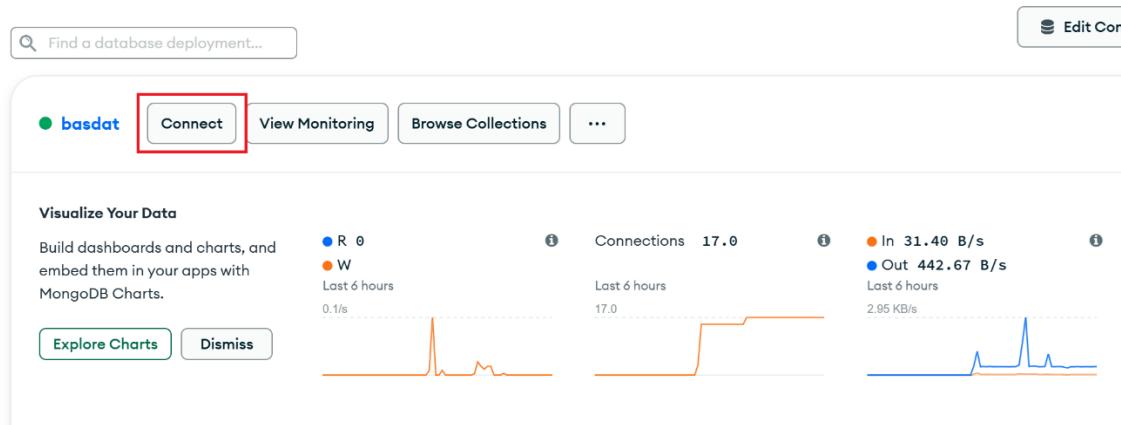
Memeriksa Versi Python DNSPython

```
!python --version
```

Tugas 2: Membangun Koneksi

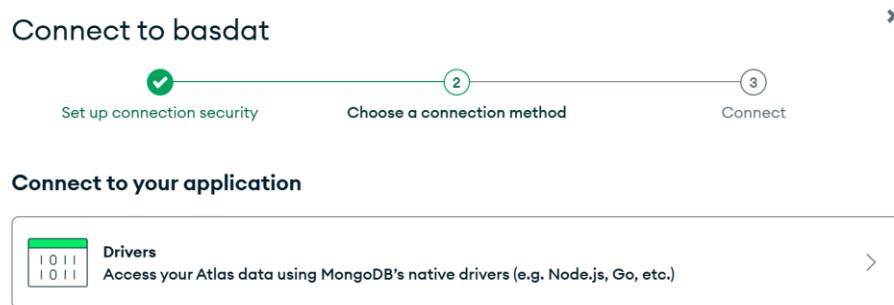
- Login* ke MongoDB Atlas lalu klik *button Connect*

Clusters



Gambar 11. 4 Klik tombol **Connect**

8. Pilih koneksi menggunakan **Drivers**



Gambar 11. 5 Memilih koneksi dengan *driver*

9. Pilih *driver* python dan sesuaikan versinya dengan versi Python yang terpasang dan sudah diperiksa di Tugas 1. Setelah itu, klik *icon copy* untuk menyalin *connection string*

Connecting with MongoDB Driver

1. Select your driver and version

We recommend installing and using the latest driver version.

Driver	Version
Python	3.6 or later

2. Install your driver

Run the following on the command line

Note: Use appropriate Python 3 executable

```
python -m pip install "pymongo[srv]==3.6"
```



[View MongoDB Python Driver installation instructions.](#)

3. Add your connection string into your application code

Use this connection string in your application

View full code sample

```
mongodb+srv://[REDACTED]
```



Replace `<db_password>` with the password for the `basisdata` database user. Ensure any option params are [URL encoded](#).

Gambar 11. 6 Menyalin *connection string*

- Untuk terhubung ke MongoDB dari Python, diperlukan PyMongo. Untuk meng-import driver PyMongo ke Python, gunakan perintah berikut:

Meng-import PyMongo

```
import pymongo
from pymongo import MongoClient
```

- Tes koneksi ke MongoDB Atlas menggunakan `srv URI` yang disalin di langkah 3

Tes Koneksi

```
conn_str = "...=..." #isi dengan connection string dari MongoDB Atlas
client = MongoClient(conn_str, ServerSelectionTimeoutMS=5000)
try:
    print(client.server_info())
except Exception:
    print("Tidak dapat terhubung ke server")
```

Jangan lupa mengganti `<db_password>` dengan *password* yang telah kalian buat di minggu ke-9. Jika berhasil terhubung, keluarannya akan seperti yang tertera pada Gambar 11.7 di bawah ini.



```

conn_str = "mongodb+srv://"
client = MongoClient(conn_str, ServerSelectionTimeoutMS=5000)
try:
    print(client.server_info())
except Exception:
    print("Tidak dapat terhubung ke server")

✓ 0.8s
{'version': '8.0.13', 'gitVersion': '8dc5cd2a30c4524132e2d44bb314544dc477e611', 'modules': |}

```

Gambar 11. 7 Koneksi berhasil dibuat

Tugas 3: Menulis Operasi CRUD Sederhana dengan Python

- Buat database baru bernama `OnlineStore` dan koleksi bernama `ponsel`

Membuat database dan koleksi

```

myDB = client["OnlineStore"]
myCollection = myDB['ponsel']

```

- Buat dokumen baru dari `item_1`

Menambah dokumen

```

item_1 = {
    "_id": "U1IT000007",
    "nama_item": "iPhooone 13",
    "maks_diskon": "10%",
    "no_batch": "RR450020FRG",
    "harga": 12800000,
    "kategori": "Elektronik"
}

myCollection.insert_one(item_1)
print(myCollection.inserted_id)

```

Apabila penambahan dokumen berhasil dilakukan, outputnya adalah sebagai berikut

```

item_1 = {
    "_id": "U1IT000007",
    "nama_item": "iPhooone 13",
    "maks_diskon": "10%",
    "no_batch": "RR450020FRG",
    "harga": 12800000,
    "kategori": "Elektronik"
}
myCollection.insert_one(item_1)
print(myCollection.inserted_id)
✓ 0.0s
Collection(Database(MongoClient(host=['basdat-shard-00-01.

```

Gambar 11. 8 Berhasil menambahkan data

3. Baca dokumen yang baru ditambahkan dengan perintah berikut:

Membaca dokumen

```

import pprint

myCollection = myDB['ponsel']
user1_items = myCollection.find()

for item in user1_items:
    pprint.pprint(item)

```

```

import pprint

myCollection = myDB['ponsel']
user1_items = myCollection.find()

for item in user1_items:
    pprint.pprint(item)
✓ 0.1s
{'_id': 'U1IT000007',
 'harga': 12800000,
 'kategori': 'Elektronik',
 'maks_diskon': '10%',
 'nama_item': 'iPhooone 13',
 'no_batch': 'RR450020FRG'}

```

Gambar 11. 9 Tampilan dokumen yang baru ditambahkan



4. Untuk memasukkan beberapa dokumen sekaligus, gunakan perintah berikut ini:

Menambah beberapa dokumen

```
item_2 = {
    "_id": "U1IT00010",
    "nama_item": "Samsung Note 5",
    "maks_diskon": "5%",
    "no_batch": "RR45054FSR",
    "harga": 4800000,
    "kategori": "Elektronik"
}

item_3 = {
    "_id": "U1IT00020",
    "nama_item": "Nokia X",
    "maks_diskon": "10%",
    "no_batch": "RR450020FRG",
    "harga": 12800000,
    "kategori": "Elektronik"
}

item_4 = {
    "_id": "U1IT00021",
    "nama_item": "Sony Xperia",
    "maks_diskon": "10%",
    "no_batch": "RR450020FRG",
    "harga": 7200000,
    "kategori": "Elektronik"
}

items_baru = [item_2, item_3, item_4]

myCollection.insert_many(items_baru)
```

```
items_baru = [item_2, item_3, item_4]

myCollection.insert_many(items_baru)
✓ 0.3s
<pymongo.results.InsertManyResult at 0x15c36cd2f40>
```

Gambar 11. 10 Berhasil menambah beberapa data sekaligus

5. Untuk membaca semua dokumen yang sudah ditambahkan ke koleksi dan menampilkannya dalam bentuk tabel, gunakan perintah berikut:

Menampilkan semua dokumen dalam bentuk tabel

```
import pandas as pd
```

```
items = myCollection.find()
items_df = pd.DataFrame(items)
items_df
```

Hasilnya adalah sebagai berikut:

	<u>_id</u>	<u>nama_item</u>	<u>maks_diskon</u>	<u>no_batch</u>	<u>harga</u>	<u>kategori</u>
0	U1IT000007	iPhooone 13	10%	RR450020FRG	12800000	Elektronik
1	U1IT000010	Samsong Note 5	5%	RR45054FSR	4800000	Elektronik
2	U1IT000020	Nokya X	10%	RR450020FRG	12800000	Elektronik
3	U1IT000021	Soni Xperia	10%	RR450020FRG	7200000	Elektronik

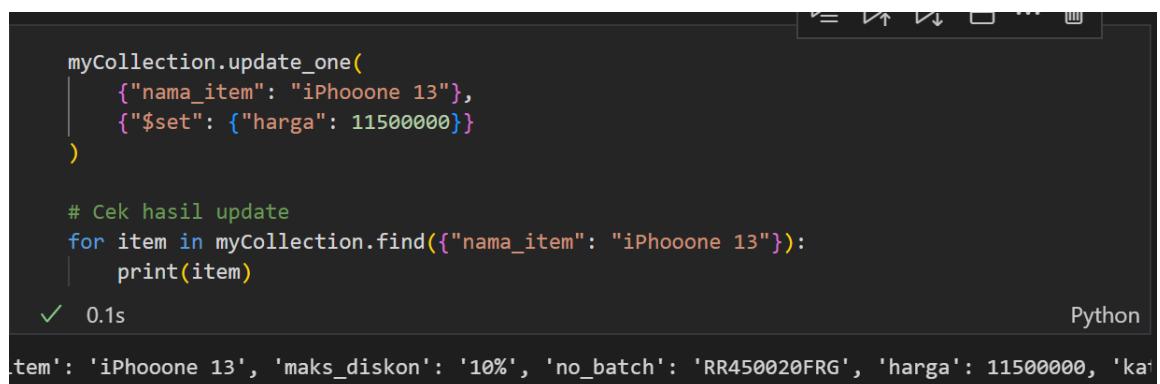
Gambar 11. 11 Tabel hasil *query*

6. Untuk memodifikasi isi dokumen yang sudah ada di koleksi, misal mengganti harga “iPhooone 13” menjadi 11500000, gunakan perintah di bawah ini:

Memodifikasi satu dokumen

```
myCollection.update_one(
    {"nama_item": "iPhooone 13"},
    {"$set": {"harga": 11500000}}
)

# Cek hasil update
for item in myCollection.find({"nama_item": "iPhooone 13"}):
    print(item)
```



```
myCollection.update_one(
    {"nama_item": "iPhooone 13"},
    {"$set": {"harga": 11500000}}
)

# Cek hasil update
for item in myCollection.find({"nama_item": "iPhooone 13"}):
    print(item)
✓ 0.1s
```

Python

{
 '_id': 'iPhooone 13', 'maks_diskon': '10%', 'no_batch': 'RR450020FRG', 'harga': 11500000, 'ka
}

Gambar 11. 12 Hasil *update* data

7. Untuk menghapus dokumen dari koleksi, misal menghapus dokumen dengan *nama_item* = “Soni Xperia”, gunakan perintah berikut ini:

Menghapus satu dokumen

```
myCollection.delete_one({"nama_item": "Soni Xperia"})

# Cek hasil delete
for item in myCollection.find():
```

```
print(item)
```

```

myCollection.delete_one({"nama_item": "Soni Xperia"})

# Cek hasil delete
for item in myCollection.find():
    print(item)

```

Python

```

{'_id': 'U1IT000007', 'nama_item': 'iPhooone 13', 'maks_diskon': '10%', 'no_batch': 'RR4500'}
{'_id': 'U1IT00010', 'nama_item': 'Samsung Note 5', 'maks_diskon': '5%', 'no_batch': 'RR4501'}
{'_id': 'U1IT00020', 'nama_item': 'Nokya X', 'maks_diskon': '10%', 'no_batch': 'RR450020FRG'

```

Gambar 11. 13 Hasil menghapus data

Aturan Penilaian (Total Skor: 100):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL 03- KU03-	CPMK 3.3.3	Praktekkanlah langkah praktikum di Tugas 1-3	Jawaban hasil praktek (screenshot)	100

11.8 POST-TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1 .	CPL 03- KU0 3	CPMK 3.3.3	Dengan Python, buatlah <i>database</i> bernama “filmDB” dan koleksi bernama “film” lalu tambahkan dokumen-dokumen berikut: i. username: GoodGuyGreg firstname: “Good Guy” lastname: “Greg” ii. username: GoodGuyGreg firstname: “Good Guy”	Jawaban hasil praktek disampiring (screenshot)	100

			<p>lastname: "Greg"</p> <p>iii. username: JohnDoe1234</p> <p>firstname: "John"</p> <p>lastname: "Doe"</p> <p>yearjoined: 2005</p> <p>iv. username: SamJohnSings</p> <p>firstname: "Sam John"</p> <p>lastname: "Sings"</p> <p>rating: 4.9</p>		
--	--	--	--	--	--

11.9 HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL 03-KU03	CPMK 3.3.3.	20%		
2.	Praktik	CPL 03-KU03	CPMK 3.3.3	30%		
3.	Post-Test	CPL 03-KU03	CPMK 3.3.3	50%		
Total Nilai						

PRAKTIKUM 12: TREND DBMS

Pertemuan ke : 12

Total Alokasi Waktu : 180 menit

- Materi : 30 menit
- Pre-Test : 30 menit
- Praktikum : 60 menit
- Post-Test : 60 menit

Total Bobot Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 30 %
- Post-Test : 50 %

Pemenuhan CPL dan CPMK:

CPL 02-S02	Menginternalisasi semangat kemandirian, kejuangan, kewirausahaan dan menghargai keanekaragaman budaya, pandangan, agama, kepercayaan, serta pendapat atau temuan orisinal orang lain.
CPMK 3.3.4	Mahasiswa mampu mengimplementasikan basis data dalam kasus sehari-hari

12.1 DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu memahami:

1. Menerapkan konsep tren basis data modern serta relevansinya terhadap kebutuhan analitik data
2. Menggunakan MongoDB (NoSQL) dan Python untuk mengakses, mengolah, serta menganalisis data penjualan pada sebuah studi kasus
3. Menyusun hasil analisis data dalam bentuk tabel dan visualisasi sederhana, kemudian mengompilasinya menjadi laporan dalam format PDF sebagai keluaran akhir praktikum

12.2 INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator Ketercapaian diukur dengan:

CPL 02-S02	CPMK 3.3.4	1. Kemampuan mahasiswa meningkat dalam mengakses koleksi data pada MongoDB Atlas menggunakan Python (PyMongo) tanpa error koneksi
CPL 02-S02	CPMK 3.3.4	2. Kemampuan mahasiswa meningkat dalam menghasilkan visualisasi data (grafik batang) dari hasil <i>query</i>
CPL 02-S02	CPMK 3.3.4	3. Kemampuan mahasiswa meningkat dalam mengintegrasikan tabel dan grafik ke dalam report PDF yang dihasilkan otomatis dengan Python



12.3 TEORI PENDUKUNG

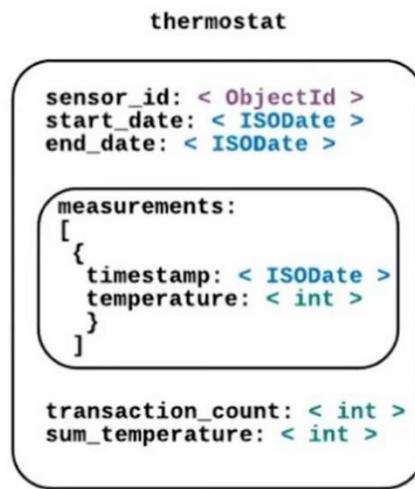
Dalam pengembangan basis data modern, perancangan model data merupakan aspek yang sangat penting untuk menjamin performa, skalabilitas, serta fleksibilitas sistem. Tidak seperti basis data relasional yang menerapkan aturan ketat normalisasi tabel, MongoDB sebagai salah satu basis data NoSQL memberikan kebebasan yang lebih besar dalam menentukan skema dokumen. Kebebasan ini menawarkan fleksibilitas, namun pada saat yang sama juga menuntut adanya pemahaman tentang praktik terbaik dalam perancangan model data. Untuk itu, digunakanlah berbagai *design pattern* yang berfungsi sebagai pedoman agar struktur dokumen yang dirancang dapat mendukung kebutuhan aplikasi.

1. Schema Versioning Pattern

Kebutuhan aplikasi sering kali berkembang sehingga struktur data perlu berubah. Untuk menghadapi perubahan ini, MongoDB mengenal *schema versioning pattern*. Pola ini memungkinkan setiap dokumen menyimpan metadata versi skema sehingga aplikasi dapat mengenali perbedaan struktur antar dokumen.

2. Bucket Pattern

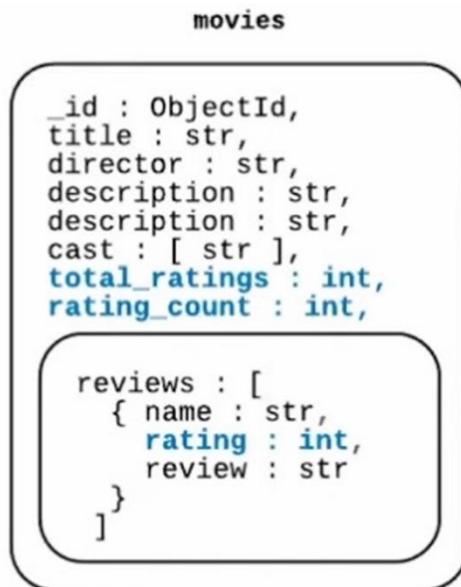
Bucket pattern lazim digunakan untuk menyimpan data berjumlah besar dengan laju pertumbuhan tinggi, misalnya data sensor, *log*, atau transaksi harian. Dibanding menyimpan setiap entri sebagai dokumen terpisah, *bucket pattern* mengelompokkan data ke dalam satu dokumen berdasarkan periode waktu atau kriteria tertentu. Dengan cara ini, jumlah dokumen dapat dikurangi dan efisiensi *query* meningkat.



Gambar 12. 1 Contoh penerapan *bucket pattern*

3. Computed Pattern

Dalam pola ini, nilai hasil perhitungan tertentu, seperti total pembelian pelanggan atau jumlah penjualan suatu produk, disimpan langsung ke dalam dokumen. Tujuannya adalah untuk menghindari perhitungan ulang yang berulang-ulang dan membebani sistem. Dengan menyimpan nilai hasil komputasi, performa baca meningkat karena data sudah tersedia secara langsung. Namun, pendekatan ini menuntut adanya strategi pemeliharaan agar data yang tersimpan tetap konsisten, misalnya dengan melakukan pembaruan secara atomik setiap kali terjadi transaksi



Gambar 12. 2 Contoh penerapan *computed pattern* di field rating_count dan total_ratings

4. Embed vs Reference

Penyimpanan data secara *embed* berarti bahwa subdokumen disimpan langsung di dalam dokumen induk. Pola ini tepat digunakan apabila data tersebut sering diakses bersama dan jarang berubah secara terpisah. Keuntungannya adalah kemudahan akses serta dukungan terhadap transaksi atomik dalam satu dokumen. Namun, pola ini juga dapat menimbulkan duplikasi data dan risiko dokumen yang terlalu besar. Sebaliknya, pola *reference* digunakan ketika subdokumen berukuran besar, sering berubah, atau digunakan secara bersama oleh banyak entitas. Pada pola ini, dokumen induk hanya menyimpan acuan berupa *id*, sedangkan data lengkap disimpan pada koleksi lain. Pendekatan ini lebih efisien dalam pemeliharaan, tetapi membutuhkan mekanisme *join* menggunakan \$lookup atau akses terpisah yang dapat memengaruhi performa.

Apabila suatu koleksi menggunakan pola *embed*, indeks pada *subfield* menjadi penting untuk memastikan *query* tetap efisien. Sementara itu, pola *reference* menuntut adanya indeks pada *foreign key* agar penggabungan antar koleksi dapat dilakukan.

12.4 HARDWARE DAN SOFTWARE

1. MongoDB Compass
2. Visual Studio Code

12.5 PRE-TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL 02-S02	CPMK 3.3.4	1. Sebutkan minimal 3 <i>design patterns</i> !	30
2.	CPL 02-S02	CPMK 3.3.4	2. Jelaskan perbedaan <i>embedded</i> dan <i>reference</i> !	40



3.	CPL 02-S02	CPMK 3.3.4	3. Sebutkan contoh pemakaian <i>bucket pattern</i> !	30
----	------------	---------------	--	----

12.6 LANGKAH PRAKTIKUM

Tugas 1: Memahami *Design Pattern*

4. Eksekusi perintah di bawah ini untuk membuat koneksi dengan MongoDB menggunakan Python dan melihat contoh dokumen dalam koleksi `sales`

Membuat koneksi

```
import pymongo
from pymongo import MongoClient
from pprint import pprint

conn_str = "mongodb+srv://....."
client = MongoClient(conn_str)
```

5. Baca contoh dokumen dalam *collection* menggunakan perintah berikut:

Membaca dokumen

```
db = client["sample_supplies"]
sales = db['sales']

pprint(sales.find_one())
```

```
{
  '_id': ObjectId('5bd761dcae323e45a93ccff2'),
  'couponUsed': False,
  'customer': {'age': 34,
    'email': 'keigutip@vaw.tc',
    'gender': 'F',
    'satisfaction': 4},
  'items': [{['name': 'pens',
    'price': Decimal128('75.21'),
    'quantity': 1,
    'tags': ['writing', 'office', 'school', 'stationary']},
  {'name': 'backpack',
    'price': Decimal128('61.11'),
    'quantity': 5,
    'tags': ['school', 'travel', 'kids']},
  {'name': 'envelopes',
    'price': Decimal128('17.2'),
    'quantity': 8,
    'tags': ['stationary', 'office', 'general']},
  {'name': 'notepad',
    'price': Decimal128('37.41'),
    'quantity': 1,
    'tags': ['office', 'writing', 'school']},
  {'name': 'printer paper',
    'price': Decimal128('43.63'),
    'quantity': 7,
    'tags': ['office', 'stationary']}],
  'purchaseMethod': 'Phone',
  'saleDate': datetime.datetime(2015, 7, 25, 7, 20, 29, 804000),
  'storeLocation': 'Seattle'}
```

Gambar 12.3 Contoh dokumen di `collection sales`

Dari Gambar 12.3, terlihat bahwa *field* item menyimpan data dalam bentuk *array*. *Array* ini digunakan untuk menyimpan daftar item yang dibeli dalam satu transaksi penjualan. Pola seperti ini disebut sebagai pola *embedded*, di mana subdokumen yang berisi data produk disimpan secara langsung dalam sebuah *field* dalam dokumen.

6. Masukkan kode di bawah ini untuk melihat lebih detail struktur dokumen di koleksi `sales` untuk memahami *embedded pattern*.

Mengeksplorasi data

```
doc = sales.find_one()

print("Field utama:")
print(doc.keys())

print("\nField items:")
for item in doc.get("items", []):
    print(item)
```

Dokumen dalam sales memiliki *field* utama, di antaranya `_id`, `saleDate`, `items`, `storeLocation`, `customer`, `couponUsed`, dan `purchaseMethod`. Adapun *field* `items` menyimpan *array* berisi objek-objek.

Gambar 12. 4 Object `items` yang berisi *array of object*

Penggunaan *embedded pattern* didasari oleh data item dalam satu transaksi pembelian yang bersifat statis atau tidak berubah. Selain itu, daftar item yang dibeli dalam satu transaksi juga sering diakses bersama dokumen utama di dalam dokumen transaksi.

7. *Computed pattern* dapat digunakan untuk menyimpan hasil perhitungan yang sering digunakan agar perhitungan tidak perlu dilakukan berulang kali tiap data dibutuhkan. Pola ini dapat diterapkan untuk menghitung jumlah atau total harga *order* dengan cara berikut ini:

Menambah *field* totalOrder

```
pipeline_computed_embed = [
    {"$unwind": "$items"},

    {"$project": {
        "_id": 1,
        "saleDate": 1,
        "customer": 1,
        "purchaseMethod": 1,
        "storeLocation": 1,
        "items": 1,
        "lineTotal": {"$multiply": ["$items.price", "$items.quantity"]}
    }},
    {"$group": {
        "_id": "$_id",
        "saleDate": {"$first": "$saleDate"},
        "customer": {"$first": "$customer"},
        "purchaseMethod": {"$first": "$purchaseMethod"},
        "storeLocation": {"$first": "$storeLocation"},
        "items": {"$push": "$items"},
        "orderTotal": {"$sum": "$lineTotal"}
    }},
    {"$merge": {
        "into": "sales",
        "whenMatched": "merge",
        "whenNotMatched": "discard"
    }}
]

db.sales.aggregate(pipeline_computed_embed)
```

Penjelasan:

- a. `$unwind`: memisahkan setiap anggota dari *array* menjadi dokumen terpisah.
- b. `$project`: memodifikasi *field* yang masih akan digunakan di *stage* berikutnya. Di sini dibuat *field* baru bernama `lineTotal` yang merupakan hasil kali `items.price` dengan `items.quantity`
- c. `$group`: mengelompokkan dokumen berdasarkan nilai yang sama pada `_id`. Nilai yang diambil untuk *field* `saleDate`, `customer`, `purchaseMethod`, `storeLocation` adalah nilai pertama masing-masing *field*. Semua item digabungkan kembali dalam array dengan `$push`, sedangkan `orderTotal` diperoleh dengan menjumlahkan semua nilai `lineTotal`.
- d. `$merge`: menuliskan atau menambahkan hasil agregasi ke koleksi `sales`. `whenMatched: "merge"` digunakan untuk mengatakan agar jika dokumen dengan suatu `_id` sudah ada dalam *collection* sebelum hasil agregasi ditambahkan, maka data akan digabung (overwrite). Adapun `whenNotMatched: "discard"` berfungsi mengatur agar jika `_id` tidak ditemukan, maka dokumen tidak akan ditambahkan.

- Periksa struktur dokumen setelah *computed pattern* diterapkan dengan perintah berikut:

Melihat struktur dokumen

```
pprint(sales.find_one())
```

```
{
  '_id': ObjectId('5bd761dcae323e45a93ccff2'),
  'couponUsed': False,
  'customer': {'age': 34,
    'email': 'keigutip@vaw.tc',
    'gender': 'F',
    'satisfaction': 4},
  'items': [{"name": "pens",
    'price': Decimal128('75.21'),
    'quantity': 1,
    'tags': ['writing', 'office', 'school', 'stationary']},
  {"name": "backpack",
    'price': Decimal128('61.11'),
    'quantity': 5,
    'tags': ['school', 'travel', 'kids']},
  {"name": "envelopes",
    'price': Decimal128('17.2'),
    'quantity': 8,
    'tags': ['stationary', 'office', 'general']},
  {"name": "notepad",
    'price': Decimal128('37.41'),
    'quantity': 1,
    'tags': ['office', 'writing', 'school']},
  {"name": "printer paper",
    'price': Decimal128('43.63'),
    'quantity': 7,
    ...
    'orderTotal': Decimal128('861.18'),
    'purchaseMethod': 'Phone',
    'saleDate': datetime.datetime(2015, 7, 25, 7, 20, 29, 804000),
    'storeLocation': 'Seattle'}
```

Gambar 12. 5 Setelah penambahan *field* `orderTotal`

- Bucket pattern* dapat digunakan untuk membantu proses analisis tren dengan mengelompokkan transaksi berdasarkan waktu. Pola ini dapat diterapkan dengan menambahkan koleksi baru bernama `bucketed_sales` dengan cara berikut ini:

Membuat *collection* penjualan per bulan

```
pipeline = [
  {
    "$addFields": {
      "monthYear": {
```

```

        "$dateToString": {"format": "%Y-%m", "date": "$saleDate"}
    }
},
{
    "$group": {
        "_id": "$monthYear",
        "count": {"$sum": 1},
        "totalSales": {"$sum": "$orderTotal"},
        "orders": {
            "$push": {
                "orderId": "$_id",
                "totalOrder": "$orderTotal"
            }
        }
    }
},
{
    "$sort": {"_id": 1}
},
{
    "$out": "sales_per_month"
}
]

results = sales.aggregate(pipeline)

```

Penjelasan:

- `$addFields`: menambah *field* baru yang menyimpan tahun dan bulan YYYY-MM dari `saleDate`
- `$group`: mengelompokkan dokumen berdasarkan bulan dan tahun. `count` dihitung dengan menambah nilai 1 untuk tiap dokumen yang dikelompokkan, `totalSales` dihitung dengan menjumlahkan semua `orderTotal`, dan `orders` diperoleh dari menambahkan (push) `_id` sebagai `orderId` dan `orderTotal` tiap `_id`.
- `$sort`: mengurutkan berdasarkan bulan dari yang paling awal.
- `$out`: menyimpan hasil agregasi ke koleksi baru bernama `sales_per_month`

Tugas 2: Membuat Visualisasi Data Penjualan Bulanan

- Membuat visualisasi data berupa diagram batang dapat dilakukan dengan bantuan *library* di Python seperti di bawah ini:

```

Membuat bar chart penjualan tahun 2017
from datetime import datetime
import numpy as np

# Ambil data tahun 2017
results = list(sales_per_month.find(
    {"_id": {"$regex": "^2017"}},
```

```

        {"_id": 1, "totalSales": 1}
    ).sort("_id", 1))

# Ekstrak data
months = []
total_sales = []
for r in results:
    date_obj = datetime.strptime(r["_id"], "%Y-%m")
    month_name = date_obj.strftime("%B") #mengambil nama bulan
    months.append(month_name)

    if isinstance(r["totalSales"], Decimal128):
        total_sales.append(float(r["totalSales"].to_decimal()))
    else:
        total_sales.append(float(r["totalSales"]))

# Hitung rata-rata total penjualan semua bulan
average_sales = np.mean(total_sales)

# Buat diagram batang
plt.style.use('ggplot')
fig, ax = plt.subplots(figsize=(12, 6))

bar_color = "#0e5587"
bars = ax.bar(months, total_sales, color=bar_color)

# Garis rata-rata
ax.axhline(average_sales, color='black', linestyle='--', linewidth=1.5,
label=f"Rata-rata: {average_sales:.0f}")

# Label dan judul
ax.set_xlabel("Bulan")
ax.set_ylabel("Total Penjualan ($)")
ax.set_title("Total Penjualan per Bulan Tahun 2017")

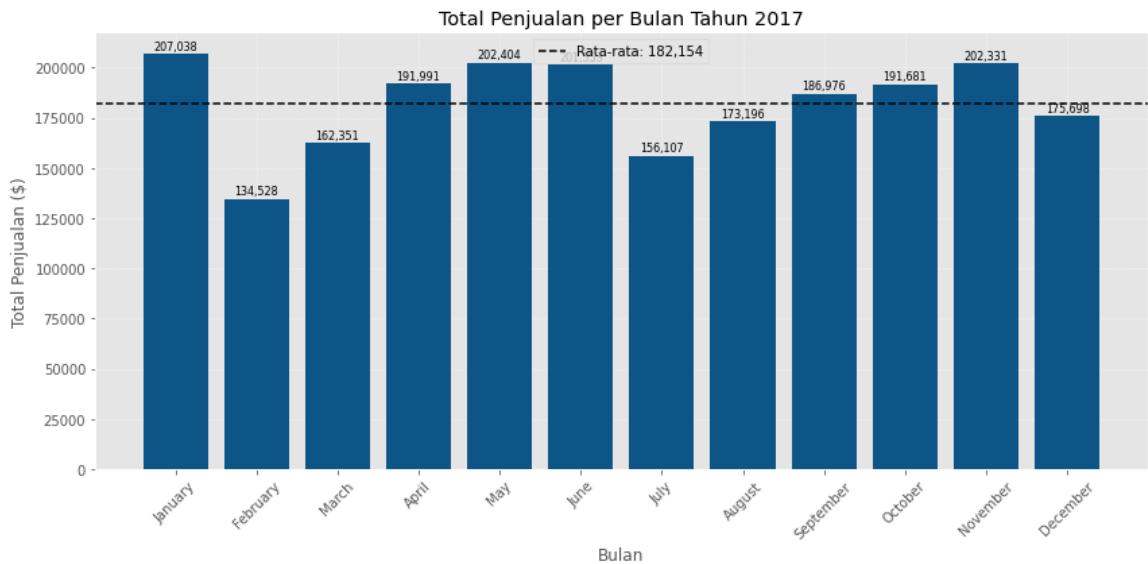
# Grid
ax.grid(True, linestyle='--', linewidth=0.5, alpha=0.8)

# Label angka di atas bar
for i, value in enumerate(total_sales):
    ax.text(i, value + max(total_sales)*0.01, f"{value:,.0f}", ha="center",
    fontsize=8)

# Legend untuk rata-rata
ax.legend()

```

```
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Gambar 12. 6 Diagram batang yang dihasilkan

Tugas 3: Membuat Invoice Otomatis

- Pasang *library reportlab*

Memasang library

```
pip install reportlab
```

- Buat QR-Code dengan perintah berikut

Membuat QR code

```
import qrcode
import gridfs
from io import BytesIO
from pymongo import MongoClient
from bson import ObjectId

fs = gridfs.GridFS(db)
sales_collection = db["sales"]

# Generate QR code
qr = qrcode.make("https://www.mongodb.com/docs/atlas/sample-data/sample-training/") #bisa diganti URL lain
buffer = BytesIO()
qr.save(buffer, format="PNG")

# Simpan ke GridFS
```

```

qr_id = fs.put(buffer.getvalue(), filename="QR.png")

# Update dokumen sales
sales_collection.update_one(
    {"_id": ObjectId("5bd761dcae323e45a93ccff2")},
    {"$set": {"qr_code": qr_id}}
)

```

3. Buat dokumen *invoice* penjualan dari dokumen dengan `_id = 5bd761dcae323e45a93ccff2` dengan perintah berikut:

Membuat collection penjualan per bulan

```

from pymongo import MongoClient
from bson import ObjectId, decimal128
from reportlab.platypus import SimpleDocTemplate, Table, TableStyle,
Paragraph, Spacer
from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
from reportlab.lib.units import mm
from datetime import datetime
from reportlab.platypus import Image
from io import BytesIO
import os
import gridfs

# Mengatur koneksi
conn_str = "mongodb+srv://.../?retryWrites=true&w=majority&appName=basdat"
#ganti dengan connection string kalian
client = MongoClient(conn_str)
db = client["sample_supplies"]
sales_collection = db["sales"]

fs = gridfs.GridFS(db)

# Folder output
output_dir = "invoices" #file PDF akan disimpan di folder dengan nama ini
os.makedirs(output_dir, exist_ok=True)

# Sesuaikan ID
target_id = "5bd761dcae323e45a93ccff2" #harus sesuai ID yang digunakan di Langkah 2

# Mengatur footer
def draw_footer(canvas, doc):
    footer_text = " ** THANK YOU FOR VISITING! ** ".upper()
    canvas.setFont("Courier", 8)
    canvas.drawCentredString(
        doc.pagesize[0] / 2, # X tengah halaman

```

```

        10 * mm,                      # Y 10mm dari bawah
        footer_text
    )

# Mengambil data
try:
    sale = sales_collection.find_one({"_id": ObjectId(target_id)})
except Exception as e:
    print(f"Error: {target_id} bukan ObjectId valid.")
    sale = None

if not sale:
    print(f"Data dengan _id {target_id} tidak ditemukan.")
else:
    # Mengatur style
    styles = getSampleStyleSheet()

    receipt_style = ParagraphStyle(
        "ReceiptStyle",
        fontName="Courier",
        fontSize=8,
        leading=10,
        alignment=1,  # center
        spaceAfter=4,
    )

    title_style = ParagraphStyle(
        "TitleStyle",
        fontName="Courier-Bold",
        fontSize=20,
        leading=18,
        alignment=1,
        spaceAfter=6,
    )

    summary_label_style = ParagraphStyle(
        "SummaryLabelStyle",
        fontName="Courier-Bold",
        fontSize=8,
        alignment=0,  # left
    )

    summary_value_style = ParagraphStyle(
        "SummaryValueStyle",
        fontName="Courier-Bold",

```

```

        fontSize=8,
        alignment=2, # right
    )

# File output
order_id = str(sale.get("_id"))
invoice_file = os.path.join(output_dir, f"invoice_{order_id}.pdf")

doc = SimpleDocTemplate(
    invoice_file,
    pagesize=(90 * mm, 140 * mm),
    rightMargin=20,
    leftMargin=20,
    topMargin=20,
    bottomMargin=20,
)
elements = []

# Mengatur header
elements.append(Paragraph("RECEI-PY.", title_style))
elements.append(Spacer(1, 6))

invoice_number = order_id[:8].upper()
sale_date = sale.get("saleDate")
customer = sale.get("customer", {})
email = customer.get("email", "-").upper()

    sale_date_str = sale_date.strftime("%A, %B %d, %Y").upper() if
isinstance(sale_date, datetime) else "-"
    elements.append(Paragraph(f"ORDER #{{invoice_number}} FOR {{email}}", receipt_style))
    elements.append(Paragraph(sale_date_str, receipt_style))
    elements.append(Spacer(1, 6))
    elements.append(Paragraph("-" * 42, receipt_style))

# Daftar item
data = [["QTY", "DESC", "AMT"]] # Mengatur nama kolom di daftar item
items = sale.get("items", [])

if not items:
    order_total = sale.get("orderTotal", 0)
    if isinstance(order_total, decimal128.Decimal128):
        order_total = float(order_total.to_decimal())
    data.append(["01", "ITEM EXAMPLE".upper(), f"{{order_total:.2f}}"])
else:

```

```

        for idx, item in enumerate(items):
            description = item.get("name", "-").upper()
            qty = item.get("quantity", 1)
            price = item.get("price", 0)

            if isinstance(price, decimal128.Decimal128):
                price = float(price.to_decimal())

            amount = qty * price
            data.append([str(qty), description, f"{amount:.2f}"]) # Data
            yang dimasukkan di baris bawah kolom

        table = Table(data, colWidths=[25, 125, 40]) # Mengatur lebar kolom
        table.setStyle(TableStyle([
            ("FONTPNAME", (0, 0), (-1, 0), "Courier-Bold"),    # header bold
            ("FONTPNAME", (0, 1), (-1, -1), "Courier"),         # isi biasa
            ("FONTSIZE", (0, 0), (-1, -1), 8),
            ("ALIGN", (0, 0), (-1, -1), "LEFT"),
        ]))
        elements.append(table)
        elements.append(Spacer(1, 6))
        elements.append(Paragraph("-" * 42, receipt_style))

        # Menghitung total
        total_amount = sale.get("orderTotal", 0)
        if isinstance(total_amount, decimal128.Decimal128):
            total_amount = float(total_amount.to_decimal())

        summary_data = [
            [
                Paragraph("TOTAL:", summary_label_style),
                Paragraph(f"{total_amount:.2f}", summary_value_style)
            ],
        ]

        summary_table = Table(summary_data, colWidths=[95, 100])
        summary_table.setStyle(TableStyle([
            ("VALIGN", (0, 0), (-1, -1), "TOP"),
        ]))

        elements.append(summary_table)
        elements.append(Paragraph("-" * 42, receipt_style))

qr_id = sale.get("qr_code") # ObjectId dari GridFS

```



```

if qr_id:
    try:
        qr_file = fs.get(qr_id)
        qr_image_data = qr_file.read()

        # Baca data gambar ke BytesIO agar bisa dipakai ReportLab Image
        qr_buffer = BytesIO(qr_image_data)

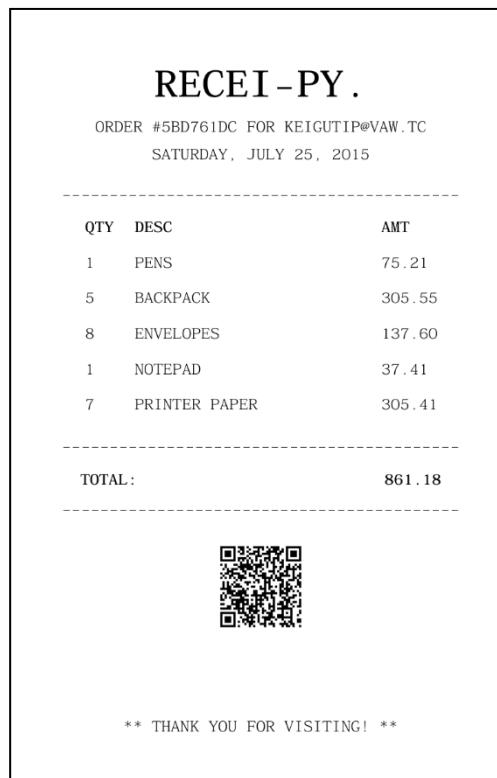
        # Buat Image flowable, atur lebar dan tinggi sesuai kebutuhan
        qr_img = Image(qr_buffer, width=50, height=50) # ukuran dalam
        points (1 point = 1/72 inch)

        # Tambahkan spacer dan gambar ke elements
        elements.appendSpacer(1, 6)
        elements.append(qr_img)
        elements.appendSpacer(1, 12))
    except Exception as e:
        print("Gagal load QR code dari GridFS:", e)

    # Membuat PDF
    doc.build(elements, onFirstPage=draw_footer, onLaterPages=draw_footer)
    print(f"Invoice dibuat: {invoice_file}")

```

File PDF akan dibuat di sebuah folder Bernama “invoice”. Adapun tampilan PDF yang dibuat adalah sebagai berikut:



Gambar 12. 7 Tampilan PDF yang di-generate

Aturan Penilaian (Total Skor: 100):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL 02-S02	CPMK 3.3.4	Praktekkanlah langkah Langkah 1-3	Jawaban hasil praktek (Screenshot)	100

12.7 POST TESTJawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL 02-S02	CPMK 3.3.4	Silahkan lakukan percobaan dengan: <ol style="list-style-type: none"> 1. Melakukan proses tambah 5 data mahasiswa 2. melakukan proses tambah 1 kolom alamat pada tabel mahasiswa 	Jawaban hasil praktek disamping (Screenshot)	100

12.8 HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL 02-S02	CPMK 3.3.4	20%		
2.	Praktik	CPL 02-S02	CPMK 3.3.4	30%		
3.	Post-Test	CPL-03 CPL-04 CPL-05 CPL-07 CPL-08	CPMK-05	50%		
Total Nilai						

DAFTAR PUSTAKA

1. <https://www.mysql.com/why-mysql/> (diakses 10 September 2023)
2. Oracle, 2019, Oracle academy Modul, Database Foundations
3. https://repository.dinus.ac.id/docs/ajar/9.materi_ERD_.pdf (diakses 5 September 2023)
4. <https://ngodingdata.com/tutorial-mongodb-cara-membuat-database-dan-collection/> (diakses 3 September 2023)
5. <https://elektro.um.ac.id/wp-content/uploads/2016/04/Basis-Data-Modul-6-Subquery.pdf> (diakses 7 September 2023)
6. <https://elektro.um.ac.id/wp-content/uploads/2016/04/Basis-Data-Modul-7-Stored-Procedure.pdf> (Diakses 7 September 2023)
7. https://bookdown.org/moh_rosidi2610/panduan_access/relasi.html (diakses 7 September 2023)
8. <https://kuliah.brigidaarie.com/wp-content/uploads/2018/02/Basis-Data-8.pdf> (Diakses 7 September 2023)
9. Silberschatz, A., H.F. Korth, and S. Sudarshan. 2019. Database System Concepts (7th Edition). McGraw-Hill.
10. Elmasri, R., and S. Navathe. 2016. Fundamentals of database systems 7th Edition. PEARSON



UAD | Laboratorium
S1 Informatika



weareuad

