KelasIT.com

MODUL PELATIHAN PROFESIONAL • EDISI DIPERLUAS

Database MySQL Fundamental

Modul pembelajaran lengkap yang membahas konsep, praktik, dan studi kasus RDBMS MySQL — dari instalasi server hingga penulisan Stored Procedure, disusun bertahap untuk pemula hingga menengah.

⏱ 21 Jam Pelatihan | ⌨ 4 Hari Intensif | 🎓 Beginner–Intermediate | 🗄 DDL · DML · DQL · JOIN · VIEW · SP

Program	**Fundamental Database MySQL**
Durasi	**21 Jam — 4 Hari Intensif**
Cakupan	**DDL, DML, DQL, Relasi & JOIN, Index, Administrasi, VIEW, Stored Procedure**
Level	**Pemula hingga Menengah**

Disusun oleh
Hifzon, M.Kom

Daftar Isi

Struktur pembelajaran modul Database MySQL Fundamental

Cara menggunakan modul ini

Setiap hari pelatihan dibuka dengan tujuan pembelajaran, dilanjutkan pembahasan konsep disertai contoh query siap pakai, kotak tips/peringatan untuk hal-hal penting, studi kasus, dan ditutup dengan rangkuman poin-poin kunci. Disarankan mempraktikkan setiap contoh query langsung di MySQL Workbench, DBeaver, atau XAMPP/phpMyAdmin agar pemahaman lebih melekat.

HARI 1 DARI 4

Pengenalan & Dasar SQL

Fondasi konsep basis data relasional, instalasi, dan pembuatan struktur tabel — dasar yang wajib kuat sebelum masuk ke materi query dan relasi.

TUJUAN PEMBELAJARAN HARI 1

- Memahami konsep dasar data, informasi, dan sistem basis data relasional
- Mampu menginstal dan mengonfigurasi MySQL Server beserta tools pendukung
- Mampu membuat database, tabel, dan menerapkan constraint dasar

1 Pengenalan Basis Data & RDBMS

1.1 Apa itu Data, Informasi, dan Basis Data

Data adalah fakta mentah yang belum memiliki konteks — misalnya angka "25" tidak bermakna apa-apa jika berdiri sendiri. Ketika data tersebut diolah dan diberi konteks, ia berubah menjadi **informasi**, misalnya "umur karyawan Budi adalah 25 tahun". Basis data (*database*) adalah kumpulan data yang terorganisir secara sistematis sehingga mudah disimpan, diakses, dikelola, dan diperbarui secara efisien. Sistem perangkat lunak yang digunakan untuk mengelola basis data disebut **DBMS** (Database Management System), dan MySQL adalah salah satu implementasi DBMS yang paling banyak digunakan di dunia industri.

Bayangkan sebuah basis data seperti lemari arsip digital raksasa: setiap laci mewakili sebuah **tabel**, setiap folder di dalam laci mewakili satu baris data (*record*), dan setiap label pada folder mewakili kolom (*field*) seperti nama, tanggal lahir, atau alamat. DBMS bertugas mengatur bagaimana laci-laci ini disusun, dicari, dan dijaga keamanannya.

Mengapa tidak cukup menyimpan data di file Excel/CSV biasa?

File datar seperti spreadsheet cepat menjadi tidak konsisten ketika diakses banyak orang sekaligus, rawan duplikasi data, sulit menjaga relasi antar data, dan tidak memiliki mekanisme keamanan bertingkat. DBMS hadir untuk mengatasi seluruh masalah tersebut secara terstruktur melalui bahasa SQL yang standar.

1.2 RDBMS vs NoSQL

RDBMS (Relational Database Management System) menyimpan data dalam bentuk tabel-tabel yang saling berelasi melalui *key*, dan diakses menggunakan bahasa standar **SQL** (Structured Query Language). Contoh RDBMS populer: MySQL, PostgreSQL, Oracle Database, dan Microsoft SQL Server. Di sisi lain, **NoSQL** menyimpan data dalam format non-tabular seperti dokumen (MongoDB), key-value (Redis), atau graph (Neo4j) — pendekatan ini cocok untuk data yang tidak terstruktur, berubah-ubah bentuknya, atau membutuhkan skalabilitas horizontal yang sangat tinggi.

Aspek	RDBMS	NoSQL
Struktur data	Tabel (baris & kolom)	Dokumen, key-value, graph
Skema	Fixed schema (ketat, terdefinisi di awal)	Fleksibel / schema-less
Relasi antar data	Kuat, dijaga lewat foreign key	Umumnya tanpa relasi eksplisit
Bahasa akses	SQL (standar & seragam)	Bervariasi, API masing-masing produk
Konsistensi data	Sangat konsisten (ACID)	Umumnya eventual consistency
Contoh produk	MySQL, PostgreSQL, Oracle	MongoDB, Redis, Cassandra
Cocok untuk	Data transaksional & terstruktur	Big data, data semi/tidak terstruktur

Istilah ACID: singkatan dari Atomicity, Consistency, Isolation, Durability — empat prinsip yang menjamin setiap transaksi pada RDBMS diproses secara utuh, konsisten, terisolasi dari transaksi lain, dan tersimpan permanen meski terjadi gangguan sistem.

1.3 Mengenal MySQL & Ekosistemnya

MySQL adalah RDBMS open-source paling populer di dunia, banyak digunakan sebagai fondasi aplikasi web modern melalui kombinasi teknologi seperti **LAMP** (Linux, Apache, MySQL, PHP) maupun **LEMP** (Linux, Nginx, MySQL, PHP). Berikut beberapa tools pendukung yang umum digunakan bersama MySQL:

MySQL Workbench

Aplikasi GUI resmi dari Oracle untuk desain skema (ER Diagram), menulis query, dan administrasi server secara visual.

phpMyAdmin

Antarmuka berbasis web yang ringan untuk mengelola MySQL, biasanya sudah tersedia dalam paket XAMPP/Laragon.

DBeaver

Tools GUI universal open-source yang mendukung banyak jenis database sekaligus (MySQL, PostgreSQL, SQLite, dll.).

MariaDB

Fork dari MySQL yang kompatibel secara sintaks, dikembangkan oleh komunitas open-source sebagai alternatif bebas lisensi Oracle.

💡 **Mengapa MySQL Banyak Dipilih?**

Gratis (open-source), performa tinggi untuk aplikasi web skala kecil-menengah, dokumentasi lengkap, dan komunitas yang sangat besar — sehingga mudah mencari solusi ketika menemui masalah. MySQL juga didukung hampir seluruh bahasa pemrograman populer (PHP, Python, Java, Node.js, Go) melalui driver/connector resmi maupun pihak ketiga.

2 Instalasi & Konfigurasi

2.1 Instalasi MySQL Server

MySQL dapat diinstal langsung melalui installer resmi dari `dev.mysql.com` untuk Windows, Linux, maupun macOS, atau melalui paket bundel seperti XAMPP maupun Laragon yang sudah menyertakan Apache, MySQL, dan PHP sekaligus dalam satu paket instalasi — pilihan ini sangat direkomendasikan untuk pemula karena prosesnya sederhana dan minim konfigurasi manual.

- Unduh installer sesuai sistem operasi dari `mysql.com/downloads` atau XAMPP dari `apachefriends.org`.
- Jalankan installer, lalu pilih komponen "MySQL Server" (sertakan MySQL Workbench bila tersedia sebagai paket tambahan).
- Tentukan *root password* pada saat proses konfigurasi berlangsung — simpan dengan aman.
- Pastikan service MySQL berjalan dengan baik (cek melalui XAMPP Control Panel, Services.msc pada Windows, atau `systemctl status mysql` pada Linux).

2.2 Koneksi ke Server

Setelah server aktif, koneksi dapat dilakukan melalui command line client bawaan MySQL atau melalui tools GUI seperti Workbench/DBeaver dengan memasukkan parameter host, port (default **3306**), username, dan password.

```
# Login ke MySQL melalui command line client
mysql -u root -p

# Setelah berhasil login, cek versi server yang terpasang
SELECT VERSION();

# Menampilkan daftar seluruh database yang ada
SHOW DATABASES;
```

> **Tips Koneksi**
>
> Gunakan host `localhost` atau `127.0.0.1` saat MySQL diinstal di komputer sendiri. Simpan password root dengan aman karena akun ini memiliki akses penuh ke seluruh database di server. Untuk lingkungan produksi, sebaiknya buat user terpisah dengan hak akses terbatas dan hindari login langsung sebagai root dari aplikasi.

3 Membuat Database & Tabel (DDL)

3.1 Data Definition Language (DDL)

DDL adalah kelompok perintah SQL yang digunakan untuk mendefinisikan, mengubah, atau menghapus struktur objek database — seperti database itu sendiri, tabel, maupun index. Tiga perintah utamanya adalah **CREATE** (membuat), **ALTER** (mengubah), dan **DROP** (menghapus). Setiap perintah DDL di MySQL bersifat *auto-commit*, artinya perubahan langsung tersimpan permanen begitu perintah dijalankan — berbeda dengan DML yang masih dapat dibatalkan selama belum di-commit.

```
-- Membuat database baru
CREATE DATABASE toko_online;

-- Menggunakan (memilih) database sebagai konteks aktif
USE toko_online;

-- Menampilkan seluruh tabel dalam database aktif
SHOW TABLES;

-- Menghapus database beserta seluruh isinya (hati-hati!)
DROP DATABASE toko_online;
```

3.2 Tipe Data MySQL

Pemilihan tipe data yang tepat sangat memengaruhi efisiensi penyimpanan dan performa query. Tipe data yang terlalu besar memboroskan ruang disk dan memori, sementara tipe data yang terlalu kecil berisiko truncation atau error saat data bertambah besar. Berikut kelompok tipe data yang paling sering digunakan:

Kelompok	Tipe Data	Contoh Penggunaan
Numerik	INT, BIGINT, DECIMAL(p,s), FLOAT, DOUBLE	Jumlah stok, ID, harga (gunakan DECIMAL untuk uang)
Teks	CHAR(n), VARCHAR(n), TEXT	Nama, alamat, deskripsi panjang
Tanggal/Waktu	DATE, DATETIME, TIMESTAMP, TIME, YEAR	Tanggal lahir, waktu transaksi

Lainnya	BOOLEAN, ENUM, JSON	Status aktif/nonaktif, pilihan tetap, data semi-terstruktur

CHAR vs VARCHAR

`CHAR(n)` selalu menggunakan ruang penyimpanan tetap sepanjang n karakter meskipun data yang diisi lebih pendek, sehingga cocok untuk data dengan panjang seragam seperti kode pos atau kode negara. `VARCHAR(n)` menyimpan sesuai panjang data aktual (lebih hemat ruang) dan lebih fleksibel, sehingga menjadi pilihan default untuk sebagian besar kolom teks seperti nama atau alamat.

3.3 Membuat & Mengubah Tabel

```
CREATE TABLE produk (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nama_produk VARCHAR(100) NOT NULL,
  harga DECIMAL(10,2) NOT NULL,
  stok INT DEFAULT 0,
  dibuat_pada DATETIME DEFAULT CURRENT_TIMESTAMP
);

-- Menambah kolom baru ke tabel yang sudah ada
ALTER TABLE produk ADD COLUMN kategori VARCHAR(50);

-- Mengubah tipe/ukuran kolom
ALTER TABLE produk MODIFY COLUMN nama_produk VARCHAR(150);

-- Menghapus kolom
ALTER TABLE produk DROP COLUMN kategori;

-- Menghapus tabel sepenuhnya
DROP TABLE produk;
```

⚠ **Perhatian**

DROP TABLE dan DROP DATABASE bersifat permanen dan tidak dapat dibatalkan (*no undo*). Selalu pastikan sudah memiliki backup sebelum menjalankan perintah ini di lingkungan produksi. Sebagai kebiasaan aman, biasakan menjalankan query SELECT untuk memverifikasi target terlebih dahulu sebelum mengeksekusi perintah yang bersifat merusak.

4 Constraint & Struktur Tabel

Constraint adalah aturan yang diterapkan pada kolom untuk menjaga validitas dan integritas data, sehingga mencegah data yang tidak konsisten masuk ke dalam tabel. Constraint dapat diterapkan pada level kolom (langsung mengikuti definisi kolom) maupun level tabel (dideklarasikan terpisah, umum digunakan untuk constraint yang melibatkan lebih dari satu kolom).

Constraint	Fungsi
PRIMARY KEY	Identitas unik setiap baris; tidak boleh NULL dan tidak boleh duplikat
NOT NULL	Kolom wajib diisi, tidak boleh dikosongkan
UNIQUE	Nilai pada kolom tidak boleh duplikat antar baris
DEFAULT	Nilai otomatis terisi apabila tidak disebutkan saat INSERT
AUTO_INCREMENT	Nilai bertambah otomatis, umumnya dipakai untuk kolom ID
FOREIGN KEY	Menghubungkan kolom dengan primary key di tabel lain (relasi)

▸ Studi Kasus: Tabel Karyawan

```
CREATE TABLE karyawan (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nik VARCHAR(20) NOT NULL UNIQUE,
  nama VARCHAR(100) NOT NULL,
  jabatan VARCHAR(50) DEFAULT 'Staff',
  gaji DECIMAL(12,2) NOT NULL
);
```

Perhatikan bahwa kolom `nik` diberi constraint `UNIQUE` karena setiap karyawan wajib memiliki Nomor Induk Karyawan yang berbeda, sementara kolom `jabatan` diberi nilai `DEFAULT 'Staff'` sehingga jika tidak diisi saat INSERT, sistem otomatis menetapkannya sebagai "Staff".

5 Praktik Mandiri Hari 1

🧩 STUDI KASUS: TOKO ONLINE

Peserta diminta merancang dan membuat skema database sederhana untuk kasus "Toko Online", terdiri dari minimal dua tabel yang berelasi (misalnya produk dan kategori), lengkap dengan pemilihan tipe data dan constraint yang tepat.

- Buat database bernama `toko_online`.
- Buat tabel **kategori** (id, nama_kategori).
- Buat tabel **produk** (id, nama_produk, harga, stok, id_kategori).
- Terapkan PRIMARY KEY dan AUTO_INCREMENT pada masing-masing tabel.
- Coba tambahkan kolom baru menggunakan ALTER TABLE, lalu hapus kembali.

RANGKUMAN HARI 1

- ✓ CREATE DATABASE / DROP DATABASE — mengelola database
- ✓ CREATE TABLE, ALTER TABLE, DROP TABLE — mengelola struktur tabel
- ✓ PRIMARY KEY, NOT NULL, UNIQUE, DEFAULT, AUTO_INCREMENT — constraint dasar
- ✓ Pemilihan tipe data yang sesuai kebutuhan (numerik, teks, tanggal)

HARI 2 DARI 4

Manipulasi & Query Data

Mengelola isi tabel dan menggali informasi menggunakan perintah SQL — dari INSERT dasar hingga fungsi agregasi dan kondisional.

TUJUAN PEMBELAJARAN HARI 2

- Mampu menambah, mengubah, dan menghapus data menggunakan DML
- Mampu menyusun query filtering, sorting, dan agregasi data
- Mampu memanfaatkan fungsi bawaan MySQL untuk mengolah string, tanggal, dan logika kondisional

1 Manipulasi Data (DML)

DML (Data Manipulation Language) digunakan untuk mengelola isi/data di dalam tabel, meliputi tiga perintah utama: **INSERT** (menambah), **UPDATE** (mengubah), dan **DELETE** (menghapus). Berbeda dengan DDL, perubahan dari DML dapat dibatalkan (*rollback*) selama belum di-commit apabila menggunakan transaksi (`START TRANSACTION` ... `COMMIT` / `ROLLBACK`).

```
-- INSERT: menambah satu baris data
INSERT INTO produk (nama_produk, harga, stok)
VALUES ('Keyboard Mechanical', 350000, 25);

-- Menambah beberapa baris sekaligus (lebih efisien)
INSERT INTO produk (nama_produk, harga, stok) VALUES
('Mouse Wireless', 150000, 40),
('Monitor 24 inch', 1800000, 10);

-- UPDATE: mengubah data yang sudah ada
UPDATE produk SET harga = 320000 WHERE nama_produk = 'Keyboard Mechanical';

-- DELETE: menghapus data tertentu
DELETE FROM produk WHERE stok = 0;
```

⚠ Selalu Gunakan WHERE

Pada perintah UPDATE dan DELETE, klausa WHERE menentukan baris mana yang terdampak. Tanpa WHERE, perintah akan berlaku untuk SELURUH baris dalam tabel — kesalahan ini sangat umum terjadi dan berpotensi fatal di lingkungan produksi. Sebagai pengaman tambahan, banyak tim mewajibkan menjalankan SELECT dengan kondisi WHERE yang sama terlebih dahulu untuk memastikan baris yang akan terdampak sudah benar.

2 Query Dasar (DQL)

DQL (Data Query Language) berpusat pada perintah **SELECT** untuk mengambil data dari tabel. SELECT dapat dikombinasikan dengan berbagai klausa untuk menyaring, mengurutkan, dan membatasi hasil.

```
-- SELECT dasar: memilih kolom tertentu
SELECT nama_produk, harga FROM produk;

-- Filtering dengan WHERE
SELECT * FROM produk WHERE harga > 200000;
SELECT * FROM produk WHERE stok = 0 OR harga < 100000;
SELECT * FROM produk WHERE nama_produk LIKE '%Mouse%';
SELECT * FROM produk WHERE harga BETWEEN 100000 AND 500000;
SELECT * FROM produk WHERE id IN (1, 3, 5);

-- Sorting & pembatasan jumlah hasil
SELECT * FROM produk ORDER BY harga DESC;
SELECT * FROM produk ORDER BY harga ASC LIMIT 5;

-- Menghilangkan duplikat
SELECT DISTINCT kategori FROM produk;
```

▸ Operator yang Sering Digunakan

- **Perbandingan:** `=` , `!=` , `<>` , `>` , `<` , `>=` , `<=`
- **Logika:** `AND` , `OR` , `NOT`

- **Rentang & keanggotaan:** `BETWEEN ... AND ...`, `IN (...)`
- **Pola teks:** `LIKE` dengan wildcard `%` (banyak karakter) dan `_` (satu karakter)
- **Nilai kosong:** `IS NULL`, `IS NOT NULL`

3 Fungsi Agregasi & Grouping

Fungsi agregasi digunakan untuk melakukan perhitungan ringkas terhadap sekumpulan baris data, seperti menghitung total, rata-rata, atau nilai maksimum/minimum. Ketika dikombinasikan dengan **GROUP BY**, perhitungan dapat dilakukan per kelompok data — sangat berguna untuk membuat laporan ringkas seperti total penjualan per kategori atau rata-rata gaji per departemen.

```
-- Fungsi agregasi dasar
SELECT COUNT(*) AS total_produk FROM produk;
SELECT SUM(stok) AS total_stok FROM produk;
SELECT AVG(harga) AS rata_rata_harga FROM produk;
SELECT MIN(harga) AS harga_termurah, MAX(harga) AS harga_termahal FROM produk;

-- GROUP BY: mengelompokkan data berdasarkan kolom tertentu
SELECT kategori, COUNT(*) AS jumlah
FROM produk
GROUP BY kategori;

-- HAVING: menyaring hasil setelah proses grouping
SELECT kategori, SUM(stok) AS total_stok
FROM produk
GROUP BY kategori
HAVING total_stok > 50;
```

> 💡 **WHERE vs HAVING**
>
> WHERE menyaring baris SEBELUM data dikelompokkan, sedangkan HAVING menyaring hasil SETELAH proses GROUP BY/agregasi dilakukan. Karena itu, HAVING dapat menggunakan hasil fungsi agregasi (seperti SUM, COUNT), sedangkan WHERE tidak bisa. Urutan eksekusi logis MySQL secara umum: FROM → WHERE → GROUP BY → HAVING → SELECT → ORDER BY → LIMIT.

4 Fungsi Bawaan MySQL

4.1 Fungsi String

```
SELECT CONCAT(nama_produk, ' - Rp', harga) AS info FROM produk;
SELECT UPPER(nama_produk), LOWER(nama_produk) FROM produk;
SELECT SUBSTRING(nama_produk, 1, 5) FROM produk;
SELECT LENGTH(nama_produk) AS panjang_nama FROM produk;
SELECT TRIM(' MySQL ') AS hasil;
```

4.2 Fungsi Tanggal & Waktu

```
SELECT NOW();
SELECT CURDATE(), CURTIME();
SELECT DATE_FORMAT(NOW(), '%d-%m-%Y') AS tanggal_indo;
SELECT DATEDIFF('2026-12-31', NOW()) AS sisa_hari;
SELECT DATE_ADD(NOW(), INTERVAL 7 DAY) AS jatuh_tempo;
```

4.3 Fungsi Kondisional

```
SELECT nama_produk,
  IF(stok = 0, 'Habis', 'Tersedia') AS status_stok
FROM produk;

SELECT nama_produk,
  CASE
    WHEN harga >= 1000000 THEN 'Premium'
    WHEN harga >= 300000 THEN 'Menengah'
    ELSE 'Ekonomis'
  END AS kategori_harga
FROM produk;
```

> 💡 **IF vs CASE**
>
> `IF()` cocok untuk logika sederhana dua kondisi (benar/salah), sedangkan `CASE WHEN` lebih fleksibel untuk banyak kondisi bertingkat dan lebih mudah dibaca ketika logikanya kompleks. Kebiasaan menggunakan CASE WHEN juga memudahkan migrasi ke RDBMS lain karena sintaksnya merupakan standar ANSI SQL.

5 Studi Kasus & Praktik: Laporan Penjualan

Peserta berlatih menyusun query laporan dari tabel transaksi sederhana, menggabungkan filtering, agregasi, dan fungsi kondisional yang telah dipelajari pada sesi-sesi sebelumnya.

🧩 LATIHAN

- Tampilkan total penjualan per kategori produk.
- Tampilkan produk dengan status stok ("Habis"/"Tersedia") menggunakan fungsi IF.
- Urutkan 5 produk dengan harga tertinggi menggunakan ORDER BY dan LIMIT.

Panduan pengerjaan

Gunakan tabel `produk` hasil praktik Hari 1 sebagai basis latihan. Kombinasikan `GROUP BY kategori` dengan `SUM(harga * stok)` untuk memperkirakan nilai total persediaan per kategori, lalu bandingkan hasilnya dengan menambahkan klausa `ORDER BY` agar kategori dengan nilai tertinggi tampil di posisi teratas.

RANGKUMAN HARI 2

- ✓ INSERT, UPDATE, DELETE — memanipulasi isi tabel
- ✓ SELECT, WHERE, ORDER BY, LIMIT, DISTINCT — mengambil dan menyaring data
- ✓ COUNT, SUM, AVG, MIN, MAX, GROUP BY, HAVING — meringkas data
- ✓ Fungsi string, tanggal, dan kondisional (IF/CASE) bawaan MySQL

HARI 3 DARI 4

Relasi Tabel & Administrasi Dasar

Menghubungkan banyak tabel dan mengelola server database secara aman — normalisasi, JOIN, index, backup, dan hak akses user.

TUJUAN PEMBELAJARAN HARI 3

- Memahami normalisasi dan menerapkan relasi antar tabel dengan foreign key
- Mampu menggabungkan data dari beberapa tabel menggunakan JOIN dan subquery
- Mampu melakukan backup/restore serta mengatur hak akses user dasar

1 Relasi Antar Tabel

1.1 Normalisasi (Ringkas)

Normalisasi adalah proses merancang struktur tabel untuk mengurangi redundansi (duplikasi) data dan menjaga konsistensi antar data yang saling berkaitan. Proses ini dilakukan secara bertahap melalui beberapa "bentuk normal" (normal form):

- **1NF (First Normal Form):** setiap kolom berisi nilai atomik — tidak ada nilai berulang/list dalam satu sel.
- **2NF:** memenuhi 1NF, dan seluruh kolom non-key bergantung penuh pada primary key (bukan sebagian saja).
- **3NF:** memenuhi 2NF, dan tidak ada ketergantungan transitif antar kolom non-key (kolom non-key tidak bergantung pada kolom non-key lainnya).

Dalam praktik sehari-hari, mendesain tabel hingga mencapai 3NF sudah cukup memadai untuk sebagian besar aplikasi bisnis — menjaga keseimbangan antara konsistensi data dan kemudahan penulisan query.

1.2 Foreign Key & Referential Integrity

FOREIGN KEY adalah kolom yang merujuk ke PRIMARY KEY di tabel lain, digunakan untuk menjaga integritas relasi antar tabel agar data yang saling berkaitan tetap konsisten — misalnya mencegah sebuah produk merujuk ke kategori yang tidak pernah ada.

```
CREATE TABLE kategori (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nama_kategori VARCHAR(50) NOT NULL
);

CREATE TABLE produk (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nama_produk VARCHAR(100) NOT NULL,
  harga DECIMAL(10,2),
  id_kategori INT,
  FOREIGN KEY (id_kategori) REFERENCES kategori(id)
    ON DELETE SET NULL
    ON UPDATE CASCADE
);
```

ON DELETE / ON UPDATE: menentukan perilaku otomatis pada baris anak ketika baris induk dihapus atau diubah. `CASCADE` akan ikut mengubah/menghapus baris terkait, `SET NULL` mengosongkan foreign key-nya, sedangkan `RESTRICT` (default) akan menolak operasi selama masih ada baris anak yang terkait.

1.3 Jenis Relasi

- **One-to-Many** — satu kategori memiliki banyak produk (relasi paling umum dalam praktik).
- **Many-to-Many** — memerlukan tabel pivot/penghubung, misalnya tabel `siswa_kursus` yang menghubungkan siswa dan kursus.

```
CREATE TABLE siswa_kursus (
  id_siswa INT,
  id_kursus INT,
  PRIMARY KEY (id_siswa, id_kursus),
  FOREIGN KEY (id_siswa) REFERENCES siswa(id),
  FOREIGN KEY (id_kursus) REFERENCES kursus(id)
);
```

2 JOIN Antar Tabel

JOIN digunakan untuk menggabungkan data dari dua tabel atau lebih berdasarkan kolom yang berelasi, sehingga kita bisa menampilkan informasi gabungan dalam satu hasil query.

```
-- INNER JOIN: hanya baris yang cocok di kedua tabel
SELECT p.nama_produk, k.nama_kategori
FROM produk p
INNER JOIN kategori k ON p.id_kategori = k.id;

-- LEFT JOIN: semua baris tabel kiri, meski tak ada pasangan di kanan
SELECT k.nama_kategori, p.nama_produk
FROM kategori k
LEFT JOIN produk p ON p.id_kategori = k.id;

-- RIGHT JOIN: kebalikan dari LEFT JOIN
SELECT p.nama_produk, k.nama_kategori
FROM produk p
RIGHT JOIN kategori k ON p.id_kategori = k.id;
```

> **Gunakan Alias Tabel**
>
> Gunakan alias singkat seperti `p` dan `k` agar query lebih ringkas dan mudah dibaca, terutama ketika melibatkan banyak tabel sekaligus dalam satu query JOIN. Alias juga wajib digunakan ketika dua tabel memiliki nama kolom yang sama, untuk menghindari ambiguitas.

3 Subquery & View

3.1 Subquery

```
-- Subquery pada klausa WHERE
SELECT nama_produk, harga FROM produk
WHERE harga > (SELECT AVG(harga) FROM produk);

-- Subquery pada klausa SELECT
SELECT nama_produk,
  (SELECT nama_kategori FROM kategori k WHERE k.id = p.id_kategori) AS kategori
FROM produk p;
```

3.2 VIEW (Pengenalan)

VIEW adalah tabel virtual hasil dari sebuah query yang disimpan dengan nama tertentu, berguna untuk menyederhanakan query kompleks yang sering dipakai berulang serta membatasi akses ke kolom tertentu. Materi VIEW akan dibahas lebih mendalam pada Hari 4.

```
CREATE VIEW view_produk_lengkap AS
SELECT p.id, p.nama_produk, p.harga, k.nama_kategori
FROM produk p
JOIN kategori k ON p.id_kategori = k.id;

-- Menggunakan VIEW seperti tabel biasa
SELECT * FROM view_produk_lengkap WHERE harga > 500000;
```

4 Index & Optimasi Dasar

Index mempercepat proses pencarian data pada tabel besar dengan cara membuat struktur pencarian tambahan — mirip daftar isi pada sebuah buku yang membantu kita menemukan halaman tertentu tanpa membaca seluruh isi buku. Namun index juga menambah beban penyimpanan dan memperlambat proses INSERT/UPDATE karena struktur index perlu ikut diperbarui.

```
CREATE INDEX idx_nama_produk ON produk(nama_produk);

-- Melihat rencana eksekusi query (query execution plan)
EXPLAIN SELECT * FROM produk WHERE nama_produk = 'Mouse Wireless';

-- Menghapus index
DROP INDEX idx_nama_produk ON produk;
```

- Gunakan index pada kolom yang sering dipakai di klausa WHERE, JOIN, atau ORDER BY.
- Hindari membuat index berlebihan pada tabel yang sering mengalami INSERT/UPDATE dalam volume besar.

5 Administrasi Dasar

5.1 Backup & Restore

```
# Backup database (dijalankan di terminal, bukan di dalam MySQL)
mysqldump -u root -p toko_online > backup_toko_online.sql

# Restore database dari file backup
mysql -u root -p toko_online < backup_toko_online.sql
```

5.2 Manajemen User & Hak Akses

```
-- Membuat user baru
CREATE USER 'staff_gudang'@'localhost' IDENTIFIED BY 'password123';

-- Memberi hak akses tertentu
GRANT SELECT, INSERT ON toko_online.produk TO 'staff_gudang'@'localhost';

-- Mencabut hak akses
REVOKE INSERT ON toko_online.produk FROM 'staff_gudang'@'localhost';
FLUSH PRIVILEGES;
```

Prinsip Least Privilege

Berikan hak akses sesuai kebutuhan minimum setiap user. Staff gudang misalnya hanya memerlukan akses SELECT dan INSERT pada tabel produk, tidak perlu diberi akses DELETE atau DROP. Prinsip ini mengurangi risiko kesalahan maupun penyalahgunaan akses.

6 Studi Kasus Akhir & Evaluasi

PROYEK MINI

Peserta mengerjakan proyek mini merancang skema database end-to-end (misalnya sistem perpustakaan atau sistem penjualan sederhana), mencakup pembuatan tabel, relasi antar tabel, dan minimal lima query (termasuk JOIN dan agregasi), dilanjutkan dengan kuis singkat untuk mengevaluasi pemahaman materi Hari 1 hingga 3.

RANGKUMAN HARI 3

- ✓ Normalisasi (1NF–3NF) dan penerapan FOREIGN KEY
- ✓ INNER JOIN, LEFT JOIN, RIGHT JOIN — menggabungkan data antar tabel
- ✓ Subquery dan pengenalan VIEW
- ✓ CREATE INDEX, EXPLAIN — dasar optimasi query
- ✓ mysqldump, CREATE USER, GRANT, REVOKE — administrasi dasar

HARI 4 DARI 4

Views & Stored Procedure

Mengemas logika query dan proses bisnis menjadi objek database yang dapat digunakan kembali secara efisien dan konsisten.

TUJUAN PEMBELAJARAN HARI 4

- Mampu membuat, mengubah, dan menghapus VIEW untuk kebutuhan pelaporan
- Memahami konsep dan manfaat Stored Procedure dalam pengelolaan logika bisnis
- Mampu membuat Stored Procedure dengan parameter dan kontrol alur (IF, WHILE)

1 VIEW Lanjutan

1.1 Manfaat VIEW

- Menyederhanakan query kompleks/berulang menjadi satu objek yang mudah dipanggil kembali.
- Meningkatkan keamanan dengan membatasi kolom atau baris yang bisa diakses oleh user tertentu.
- Menyediakan lapisan abstraksi, sehingga perubahan struktur tabel dasar tidak selalu memengaruhi aplikasi yang menggunakan VIEW tersebut.

1.2 CREATE, ALTER, DROP VIEW

```
-- Membuat VIEW
CREATE VIEW view_laporan_stok AS
SELECT nama_produk, stok,
  IF(stok = 0, 'Habis', 'Tersedia') AS status
FROM produk;

-- Mengubah definisi VIEW yang sudah ada
ALTER VIEW view_laporan_stok AS
SELECT nama_produk, stok,
  CASE WHEN stok = 0 THEN 'Habis'
       WHEN stok < 10 THEN 'Menipis'
       ELSE 'Tersedia' END AS status
FROM produk;

-- Menghapus VIEW
DROP VIEW view_laporan_stok;
```

1.3 Updatable View vs Read-only View

Sebuah VIEW dapat digunakan untuk INSERT/UPDATE/DELETE (bersifat *updatable*) apabila didefinisikan dari satu tabel tanpa fungsi agregasi, GROUP BY, atau DISTINCT. Sebaliknya, VIEW yang melibatkan JOIN, agregasi, atau subquery kompleks umumnya bersifat *read-only*.

▸ Studi Kasus: VIEW Laporan Gabungan

```
CREATE VIEW view_laporan_penjualan AS
SELECT p.nama_produk, k.nama_kategori, p.harga, p.stok,
  (p.harga * p.stok) AS nilai_stok
FROM produk p
JOIN kategori k ON p.id_kategori = k.id;

SELECT * FROM view_laporan_penjualan ORDER BY nilai_stok DESC;
```

2 Pengenalan Stored Procedure

2.1 Konsep & Manfaat

Stored Procedure (SP) adalah kumpulan perintah SQL yang disimpan di sisi server dan dapat dipanggil sewaktu-waktu menggunakan `CALL`. Manfaatnya antara lain mengurangi duplikasi logika query di banyak tempat, mempercepat eksekusi karena sudah dikompilasi di server, serta menyederhanakan operasi kompleks menjadi satu pemanggilan sederhana dari sisi aplikasi.

2.2 DELIMITER, CREATE PROCEDURE, CALL

Karena isi SP sering mengandung tanda titik-koma (`;`) di setiap barisnya, kita perlu mengganti delimiter sementara agar MySQL tidak salah menafsirkan akhir dari keseluruhan perintah CREATE PROCEDURE.

```
DELIMITER $$
CREATE PROCEDURE tampilkan_semua_produk()
BEGIN
  SELECT * FROM produk;
END $$
DELIMITER ;

-- Memanggil stored procedure
CALL tampilkan_semua_produk();
```

2.3 Variabel & Parameter IN/OUT

```
DELIMITER $$
CREATE PROCEDURE cari_produk_by_kategori(IN kategori_id INT)
BEGIN
  SELECT * FROM produk WHERE id_kategori = kategori_id;
END $$

CREATE PROCEDURE hitung_total_stok(IN kategori_id INT, OUT total INT)
BEGIN
  SELECT SUM(stok) INTO total
  FROM produk WHERE id_kategori = kategori_id;
END $$
DELIMITER ;

CALL cari_produk_by_kategori(1);
CALL hitung_total_stok(1, @hasil);
SELECT @hasil;
```

- **Parameter IN** — nilai dikirim masuk ke dalam SP (perilaku default).
- **Parameter OUT** — nilai dikembalikan/dikeluarkan dari SP ke variabel pemanggil.
- **Parameter INOUT** — dapat menerima nilai sekaligus mengembalikan nilai baru.

3 Stored Procedure Lanjutan & Praktik

3.1 Logika Kondisional (IF...THEN) dalam SP

```
DELIMITER $$
CREATE PROCEDURE cek_status_stok(IN produk_id INT)
BEGIN
  DECLARE jumlah_stok INT;
  SELECT stok INTO jumlah_stok FROM produk WHERE id = produk_id;
  IF jumlah_stok = 0 THEN
    SELECT 'Stok Habis' AS status;
  ELSEIF jumlah_stok < 10 THEN
    SELECT 'Stok Menipis' AS status;
  ELSE
    SELECT 'Stok Aman' AS status;
  END IF;
END $$
DELIMITER ;
```

3.2 Perulangan Dasar (WHILE) dalam SP

```
DELIMITER $$
CREATE PROCEDURE generate_kode_produk(IN jumlah INT)
BEGIN
  DECLARE i INT DEFAULT 1;
  WHILE i <= jumlah DO
    INSERT INTO produk (nama_produk, harga, stok)
    VALUES (CONCAT('Produk-', i), 10000 * i, 0);
    SET i = i + 1;
  END WHILE;
END $$
DELIMITER ;

CALL generate_kode_produk(5);
```

3.3 Mengelola Stored Procedure

```
-- Melihat daftar SP dalam database aktif
SHOW PROCEDURE STATUS WHERE Db = 'toko_online';

-- Melihat isi/source code SP
SHOW CREATE PROCEDURE tampilkan_semua_produk;

-- Menghapus SP
DROP PROCEDURE IF EXISTS tampilkan_semua_produk;
```

▸ Studi Kasus: SP Transaksi & Update Stok Otomatis

Peserta membuat sebuah Stored Procedure yang mencatat transaksi penjualan sekaligus mengurangi stok produk secara otomatis dalam satu pemanggilan, sehingga logika bisnis tetap konsisten dan tidak perlu ditulis berulang-ulang di sisi aplikasi.

```
DELIMITER $$
CREATE PROCEDURE proses_transaksi(
  IN p_id_produk INT,
  IN p_jumlah INT
)
BEGIN
  DECLARE stok_tersedia INT;
  SELECT stok INTO stok_tersedia FROM produk WHERE id = p_id_produk;

  IF stok_tersedia >= p_jumlah THEN
    INSERT INTO transaksi (id_produk, jumlah, tanggal)
    VALUES (p_id_produk, p_jumlah, NOW());
    UPDATE produk SET stok = stok - p_jumlah WHERE id = p_id_produk;
    SELECT 'Transaksi berhasil' AS pesan;
  ELSE
    SELECT 'Stok tidak mencukupi' AS pesan;
  END IF;
END $$
DELIMITER ;

CALL proses_transaksi(1, 3);
```

📌 Catatan Praktik

Latihan ini menggabungkan konsep VIEW dan Stored Procedure dari Hari 4 dengan konsep tabel & relasi dari Hari 1–3, sehingga peserta memahami bagaimana seluruh materi saling terhubung dalam skenario nyata. Perhatikan bahwa validasi stok dilakukan di dalam SP sebelum INSERT/UPDATE dijalankan — pola ini menjaga agar data transaksi tidak pernah tercatat melebihi stok yang tersedia.

RANGKUMAN HARI 4

- ✓ CREATE VIEW, ALTER VIEW, DROP VIEW — objek query tersimpan
- ✓ CREATE PROCEDURE, CALL, DELIMITER — dasar Stored Procedure
- ✓ Parameter IN, OUT, INOUT pada Stored Procedure
- ✓ IF...ELSEIF...ELSE dan WHILE — kontrol alur dalam SP
- ✓ Studi kasus SP transaksi terintegrasi dengan update stok

LAMPIRAN

Referensi Cepat

Rangkuman perintah dan glosarium istilah untuk referensi cepat setelah pelatihan selesai.

Rangkuman Perintah SQL

Kategori	Perintah Kunci
DDL	CREATE, ALTER, DROP
DML	INSERT, UPDATE, DELETE
DQL	SELECT, WHERE, ORDER BY, GROUP BY, HAVING, LIMIT
Relasi	JOIN (INNER/LEFT/RIGHT), FOREIGN KEY
Objek Lanjutan	VIEW, PROCEDURE, INDEX
Administrasi	CREATE USER, GRANT, REVOKE, mysqldump

Glosarium Istilah

Primary Key
Kolom penanda unik setiap baris dalam tabel.

Foreign Key
Kolom yang merujuk ke primary key tabel lain untuk membentuk relasi.

Normalisasi
Proses merancang tabel agar minim duplikasi data.

Index
Struktur tambahan yang mempercepat pencarian data.

View
Tabel virtual hasil query yang disimpan dengan nama tertentu.

Stored Procedure
Kumpulan perintah SQL tersimpan yang dapat dipanggil ulang.

Referensi Lanjutan
Dokumentasi resmi MySQL (dev.mysql.com/doc), W3Schools SQL Tutorial, serta modul praktik dan dataset latihan tambahan yang disediakan oleh trainer.

Tentang Modul Ini
Modul "Database MySQL Fundamental" disusun oleh
Hifzon, M.Kom
sebagai materi pelatihan profesional untuk KelasIT.com. Seluruh isi modul — termasuk contoh query dan studi kasus — dapat dipraktikkan langsung menggunakan MySQL Server versi terbaru.