# LAPORAN TUGAS BESAR

# PROYEK PENGANTAR BASIS DATA LANJUT


**Disusun Oleh Kelompok 3 :**

1. Torang Four Yones Manulang (G1A022052)
2. Rizki Ramadani Dalimunthe (G1A022054)
3. Revo Pratama (G1A022058)
4. Wahyu Ozorah Manurung (G1A022060)
5. Pandu Rizki Mulyanto (G1A022076)

**Nama Asisten Dosen:**

1. Gita Dwi Putriani (G1A019043)
2. Valleryan Virgil Zuliuskandar (G1A020021)
3. Syafrizza Aulia Marizky (G1A020029)
4. Muhadzib Nursaid (G1A020035)
5. Dwinta Septiana (G1A020041)
6. Muhammad Willdhan Arya Putra (G1A020095)
7. Ilham Dio Putra (G1A021024)
8. Rayhan M. Rizki (G1A021083)
9. Kahfi Zairan (G1A021041)
10. Esti Asmareta Ayu (G1A021042)

**Dosen Pengampu:**

1. Mochammad Yusa, S.Kom., M.Kom
2. Endina Putri Purwandari, S.T., M.Kom

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS BENGKULU**


**2024**

## LANDASAN TEORI

Pangkalan data (disebut juga basis data; bahasa Inggris: database) adalah kumpulan data yang terorganisir, yang umumnya disimpan dan diakses secara elektronik dari suatu sistem komputer. Pada saat pangkalan data menjadi semakin kompleks, maka pangkalan data dikembangkan menggunakan teknik perancangan dan pemodelan secara formal. Secara konsep, basis data dan lemari arsip sesungguhnya memiliki prinsip kerja dan tujuan yang sama. Dan tujuan utamanya adalah kemudahan dan kecepatan dalam pengambilan kembali data atau arsip. Perbedaannya hanya terletak pada media penyimpanan yang digunakan. jika lemari arsip menggunakan lemari sebagai media penyimpanan, maka basis data menggunakan media penyimpanan elektronis seperti disk (Flashdisk, harddisk, MicroSD). Yang perlu diingat adalah bahwa tidak semua bentuk penyimpanan data secara elektronis bisa disebut basis data. Yang sangat ditonjolkan dalam basis data adalah pengaturan, pemilahan, pengelompokkan, pengorganisasian data yang akan kita simpan sesuai fungsi atau jenisnya (Andaru, A. (2018)).

Pemilahan/ pengelompokan ini dapat berbentuk sejumlah file atau tabel terpisah atau dalam bentuk pendefinisian kolom- kolom atau field-field data dalam setiap file atau tabel. Perangkat lunak yang dapat digunakan untuk mengelola basis data disebut sistem manajemen basis data (database management sistem) atau disingkat DBMS. DBMS (Database Management System) adalah alat yang ampuh untuk membuat dan mengelola jumlah data yang besar secara efisien dan memungkinkannya bertahan dalam jangka waktu yang lama dengan aman. DBMS adalah kumpulan data yang saling terkait dan seperangkat program untuk mengakses data tersebut.. Tujuan utama DBMS adalah menyediakan cara untuk menyimpan dan mengambil informasi basis data yang nyaman dan efisien. Sistem basis data dirancang untuk mengelola banyak informasi. Manajemen data melibatkan struktur pendefinisian untuk penyimpanan informasi dan menyediakan mekanisme untuk manipulasi informasi. Selain itu, sistem basis data harus memastikan keamanan informasi yang disimpan, meskipun sistem macet atau upaya akses yang tidak sah. Jika data akan dibagikan di antara beberapa pengguna, sistem harus menghindari kemungkinan hasil yang tidak normal (Kadir, A. (2002)).

Dalam penerapannya, terdapat beberapa jenis software DBMS yang sering diaplikasikan untuk mengelola database perusahaan yaitu diantaranya: a. MySQL MySQL sering dipilih oleh banyak perusahaan karena gratis dan cocok untuk bisnis yang sedang tumbuh. Meskipun gratis, keamanan dan kecepatan akses data tetap stabil. Namun, MySQL kurang cocok dengan bahasa pemrograman Foxpro, Visual Basic (VB), dan Delphi, serta tidak optimal untuk data dalam jumlah besar. b. Oracle Oracle adalah perangkat lunak DBMS berbayar yang sangat baik. Fitur fiturnya yang beragam cocok untuk kebutuhan fleksibilitas perusahaan besar. Oracle juga menonjol dalam pemrosesan transaksi yang cepat. Keamanannya tidak diragukan lagi sesuai

dengan standar DBMS. c. Microsoft SQL Server Microsoft SQL Server, selain Oracle, sesuai untuk digunakan dalam jaringan komputer perusahaan besar karena mampu mengelola volume data yang besar (Raissa , A. (2022)).

SQL Server dilengkapi dengan sistem keamanan yang baik serta fitur fitur backup, recovery, dan rollback data. Namun, perangkat ini hanya kompatibel dengan sistem operasi Windows. SQL (Structured Query Language) adalah sebuah bahasa yang digunakan untuk mengakses data dalam basis data relasional. SQL secara de facto merupakan bahasa standar yang digunakan dalam RDBMS (relational database management system). Saat ini hampir semua server basis data yang ada mendukung bahasa SQL untuk melakukan manajemen datanya. SQL merupakan bahasa baku (ANSI/SQL), non prosedural dan berorientasi himpunan (set oriented language) SQL dapat digunakan baik secara interaktif atau ditempelkan (embedded) pada sebuah program aplikasi. SQL dapat digunakan untuk mendefinisikan struktur data, memodifikasi data pada basis data. Secara umum, bahasa SQL memiliki beberapa bagian penting (Mc Graw Hill, 2003).yaitu

a. DDL ( Data Definition Language ) DDL adalah sebuah metode Query SQL yang berguna untuk mendefinisikan data pada sebuah Database, Query yang dimiliki DDL adalah
   • CREATE : Digunakan untuk membuat Database dan Tabel
   • Drop : Digunakan untuk menghapus Tabel dan Database • Alter : Digunakan untuk melakukan perubahan struktur tabel yang telah dibuat, baik menambah Field (Add), mengganti nama Field (Change) ataupun menamakannya kembali (Rename), dan menghapus Field (Drop).
b. DML ( Data Manipulation Language ) DML adalah sebuah metode Query yang dapat digunakan apabila DDL telah terjadi, sehingga fungsi dari Query DML ini untuk melakukan pemanipulasian database yang telah dibuat. Query yang dimiliki DML adalah:
   • INSERT : Digunakan untuk memasukkan data pada Tabel Database
   • UPDATE : Digunakan untuk pengubahan data pada Tabel Database
   • DELETE : Digunakan untuk Penghapusan data pada tabel Database
   • COMMIT : Mentapkan penyimpanan Database
   • ROLLBACK : Membatalkan penyimpanan Database
c. DCL ( Data Control Language ) DCL adalah sebuah metode Query SQL yang digunakan untuk memberikan hak notorisasi mengakses Database, mengalokasikan space, pendefinisian space, dan pengauditan penggunaan database. Query yang dimiliki DCL adalah :

• GRANT : Untuk mengizinkan User mengakses Tabel dalam Database.
• REVOKE : Untuk membatalkan izin hak user, yang ditetapkan oleh perintah GRANT

Ada beberapa manfaat dari penggunaan basis data. Berikut adalah beberapa manfaat penting dari pengembangan basis data (Gayatri, I. S. (2024).).

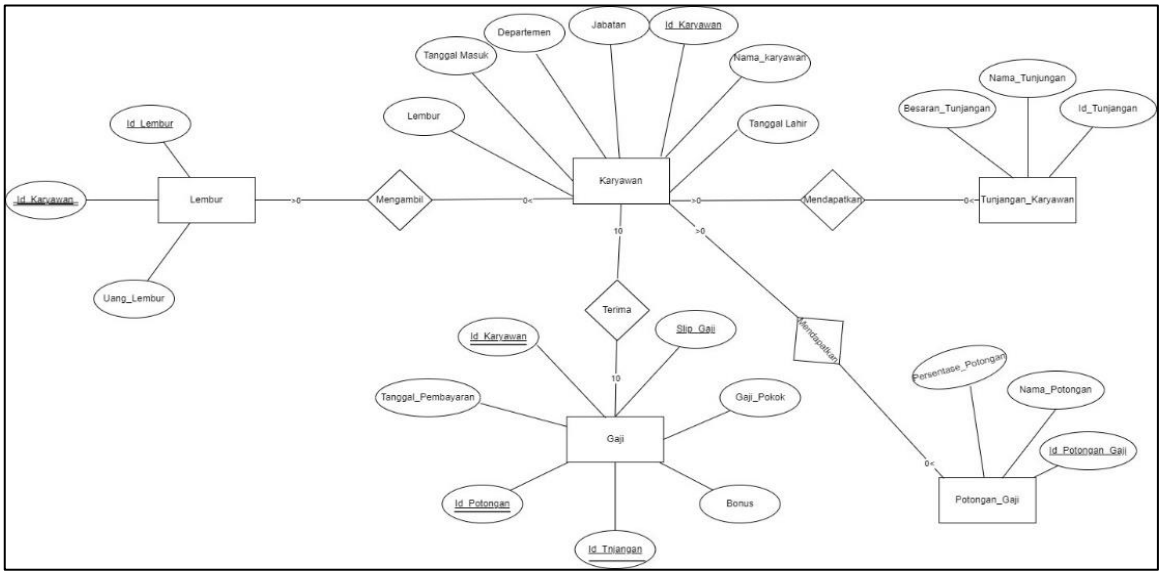
1. Eliminasi Redundansi Data Salah satu manfaat utama dari pengembangan sistem basis data adalah eliminasi redundansi data. Dengan menggunakan Database Management System (DBMS), data dapat dikelola dengan cara yang meminimalkan duplikasi. Ini tidak hanya menghemat ruang penyimpanan, tetapi juga mempercepat waktu akses ke data.
2. Konsistensi Data Terjamin DBMS menjamin konsistensi data. Saat menyimpan data, DBMS memastikan bahwa data disimpan sekali dalam sistem yang sama. Ini mencegah duplikasi di sistem lain dan memastikan konsistensi data. Data juga diperbarui secara berkala untuk memastikan keakuratan.
3. Pembatasan Akses yang Tidak Sah DBMS memungkinkan pembatasan akses ke sistem basis data. Ini membantu mencegah akses yang tidak sah, baik dari pihak internal yang tidak memiliki otoritas atau dari pihak luar yang berusaha mengakses data tanpa izin.
4. Peningkatan Pembagian Data dan Produktivitas Dengan semua data disimpan dalam sistem basis data, siapa pun dengan otoritas yang tepat dapat dengan mudah mengakses data secara keseluruhan. Ini memfasilitasi pembagian data dan dapat meningkatkan produktivitas serta mempercepat pengambilan keputusan berdasarkan data.
5. Akses Mudah ke Banyak Data Sekaligus Dalam pengembangan sistem basis data dengan DBMS, pengguna dapat melihat banyak data sekaligus. Meskipun tampilan DBMS mungkin berbeda untuk setiap pengguna, sebagian besar perangkat lunak DBMS dapat menampilkan banyak data sekaligus.
6. Fasilitas Pencadangan dan Pemulihan Data Kehilangan data adalah masalah besar dalam pengelolaan data. Oleh karena itu, sistem keamanan data yang kuat sangat penting. DBMS memfasilitasi pencadangan data secara berkala, yang sangat penting terutama ketika data berjumlah besar. Selain itu, DBMS juga membantu dalam pemulihan data jika terjadi kehilangan data.

MySQL merupakan database engine atau server database yang mendukung bahasa database SQL sebagai bahasa interaktif dalam mengelola data. MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL atau DBMS yang multithread, multi-user (Saputra, A. (2012)).

MySQL adalah DBMS yang open source dengan dua bentuk lisensi, yaitu Free Software (perangkat lunak bebas) dan Shareware (perangkat lunak berpemilik yang penggunaannya terbatas). MySQL dikembangkan oleh perusahaan Swedia bernama MySQLAB. Jadi MySQL adalah database server yang gratis dengan lisensi GNU General Public License

## SOAL DAN PEMBAHASAN

1. ERD Sistem Manajemen Gaji
   Jawab:




**Gambar 1.** ERD Manajemen Gaji

2. Membuat database, membuat tabel dan memasukkan data

```
2    CREATE DATABASE IF NOT EXISTS perusahaan1;
3    drop database perusahaan1;
4    -- Gunakan database "perusahaan"
5    USE perusahaan1;
```

**Gambar 2**. Membuat dan Menggunakan Database

```
8    CREATE TABLE IF NOT EXISTS Karyawan (
9        ID_Karyawan INT (10) PRIMARY KEY,
10       Nama_Karyawan VARCHAR(50),
11       Departemen VARCHAR(50),
12       Jabatan VARCHAR(50),
13       Tanggal_Lahir DATE,
14       Tanggal_Masuk DATE,
15       Lembur BOOLEAN DEFAULT FALSE
16   );
17   desc karyawan;
18   drop table karyawan;
```

**Gambar 3.**Membuat Tabel Karyawan

```
20   CREATE TABLE IF NOT EXISTS Lembur(
21   ID_lembur INT (10) PRIMARY KEY,
22   ID_Karyawan INT (10) ,
23   Uang_Lembur INT (20),
24   FOREIGN KEY (ID_Karyawan) REFERENCES Karyawan(ID_Karyawan));
25   desc Lembur;
```

**Gambar 4.** Membuat Tabel Lembur

```
28   CREATE TABLE IF NOT EXISTS PotonganGaji (
29       ID_Potongan  INT (10) PRIMARY KEY,
30       Nama_Potongan VARCHAR(50),
31       Persentase_Potongan int (3)
32   );
33   desc PotonganGaji;
```

**Gambar 5.** Membuat Tabel Potongan Gaji

```
36  CREATE TABLE IF NOT EXISTS TunjanganKaryawan (
37      ID_Tunjangan INT (10) PRIMARY KEY,
38      Nama_Tunjangan VARCHAR(50),
39      Besaran_Tunjangan int (10)
40  );
41  desc TunjanganKaryawan;
```

**Gambar 6.** Membuat Tabel TunjanganKaryawan

```
43  -- Tabel Gaji
44  CREATE TABLE IF NOT EXISTS Gaji (
45      slip_gaji INT (10) PRIMARY KEY,
46      ID_Karyawan INT (10),
47      ID_Potongan INT (10),
48      ID_Tunjangan INT (10),
49      GajiPokok int (10),
50      Bonus int(10),
51      TanggalPembayaran DATE,
52      FOREIGN KEY (ID_Karyawan) REFERENCES Karyawan(ID_Karyawan),
53      FOREIGN KEY (ID_Potongan) REFERENCES PotonganGaji(ID_Potongan),
54      FOREIGN KEY (ID_Tunjangan) REFERENCES TunjanganKaryawan(ID_Tunjangan)
```

**Gambar 7.** Membuat Tabel Gaji

Source Code:

```
CREATE TABLE IF NOT EXISTS Karyawan (
   ID_Karyawan INT (10) PRIMARY KEY,
   Nama_Karyawan VARCHAR(50),
   Departemen VARCHAR(50),
   Jabatan VARCHAR(50),
   Tanggal_Lahir DATE,
   Tanggal_Masuk DATE,
   Lembur BOOLEAN DEFAULT FALSE
);
desc karyawan;
drop table karyawan;

CREATE TABLE IF NOT EXISTS Lembur(
ID_lembur INT (10) PRIMARY KEY,
ID_Karyawan INT (10) ,
Uang_Lembur INT (20),
    FOREIGN KEY (ID_Karyawan) REFERENCES Karyawan(ID_Karyawan));

desc Lembur;
```

```
-- Tabel PotonganGaji
CREATE TABLE IF NOT EXISTS PotonganGaji (
    ID_Potongan  INT (10) PRIMARY KEY,
    Nama_Potongan VARCHAR(50),
    Persentase_Potongan int (3)
);
desc PotonganGaji;
-- Tabel TunjanganKaryawan
CREATE TABLE IF NOT EXISTS TunjanganKaryawan (
    ID_Tunjangan INT (10) PRIMARY KEY,
    Nama_Tunjangan VARCHAR(50),
    Besaran_Tunjangan int (10)
);
desc TunjanganKaryawan;

-- Tabel Gaji
CREATE TABLE IF NOT EXISTS Gaji (
      slip_gaji INT (10) PRIMARY KEY,
    ID_Karyawan INT (10),
    ID_Potongan INT (10),
    ID_Tunjangan INT (10),
    GajiPokok int (10),
    Bonus int(10),
    TanggalPembayaran DATE,
    FOREIGN KEY (ID_Karyawan) REFERENCES Karyawan(ID_Karyawan),
    FOREIGN KEY (ID_Potongan) REFERENCES PotonganGaji(ID_Potongan),
    FOREIGN KEY (ID_Tunjangan) REFERENCES TunjanganKaryawan(ID_Tunjangan)
);
      desc gaji;
```

Penjelasan:

Tabel yang telah kami buat terdiri dari beberapa entitas yang saling terkait untuk mengelola informasi terkait karyawan dan penggajian mereka. Pertama, tabel "Karyawan" mencatat informasi dasar tentang setiap karyawan seperti ID, nama, departemen, jabatan, tanggal lahir, tanggal masuk, dan status lembur. Tabel "Lembur" merekam detail lembur,

termasuk ID lembur, ID karyawan yang terlibat, dan jumlah uang lembur. Selanjutnya, tabel "PotonganGaji" menangani potongan gaji karyawan, dengan mencatat ID, nama potongan, dan persentase potongan. Tabel "TunjanganKaryawan" digunakan untuk menyimpan informasi tentang tunjangan karyawan, seperti ID, nama tunjangan, dan besaran tunjangan. Terakhir, tabel "Gaji" merekam detail gaji karyawan, termasuk slip gaji, ID karyawan yang bersangkutan, ID potongan yang diterapkan, ID tunjangan yang diberikan, gaji pokok, bonus, dan tanggal pembayaran. Dengan menggunakan kumpulan tabel ini, kami dapat mengelola data karyawan dan elemen-elemen terkait seperti lembur, potongan gaji, tunjangan, dan pembayaran gaji dengan lebih terstruktur dan terorganisir.

```
51      -- Isi tabel KaryawanDeleteDataKaryawanGajiDeleteDataKaryawanGaji
52 •    INSERT INTO Karyawan (ID_Karyawan, Nama_Karyawan, Departemen, Jabatan, Tanggal_Lahir, Tanggal_Masuk, Lembur)
53      VALUES
54          (1, 'Wahyu Ozorah Manurung','HR', 'Manager', '1980-05-15', '2010-01-01', TRUE),
55          (2, 'Torang Four Yones Manullang', 'CEO', 'Excutive', '1985-09-20', '2015-03-10', TRUE),
56          (3, 'Revo','Kepala bidang', 'Pemasaran', '1980-05-15', '2010-01-01', FALSE),
57          (4, 'Pandu','IT', 'Back-end', '1980-05-15', '2010-01-01', TRUE),
58          (5, 'Rizki', 'Keuangan', 'Akuntan', '1990-12-12', '2018-07-05', FALSE);
59 •    select*from Karyawan;
```

**Gambar 8.** Insert Data Karyawan

```
61      -- Isi tabel Lembur
62 •    INSERT INTO Lembur (ID_Lembur, ID_Karyawan, Uang_Lembur)
63      VALUES
64      (1,1,500000),
65      (2,2,500000);
66 •    select*from Lembur;
```

**Gambar 9.** Insert Data Lembur

```
70      -- Isi tabel PotonganGaji
71 •    INSERT INTO PotonganGaji (ID_Potongan, Nama_Potongan, Persentase_Potongan)
72      VALUES
73          (1, 'Potongan Kehadiran', 2),
74          (2, 'Potongan Telat', 5),
75          (3, 'Potongan Telat', 5),
76          (4, 'Potongan Telat', 5),
77          (5, 'Potongan Telat', 5);
78 •    select * from PotonganGaji;
```

**Gambar 10.** Insert Data PotongGaji

```
80      -- Isi tabel TunjanganKaryawan
81 •    INSERT INTO TunjanganKaryawan (ID_Tunjangan, Nama_Tunjangan, Besaran_Tunjangan)
82      VALUES
83          (111, 'Tunjangan Kesehatan', 300000),
84          (112, 'Tunjangan Transportasi', 20000),
85          (113, 'Tunjangan Transportasi', 20000),
86          (114, 'Tunjangan Pendidikan', 2000000),
87          (115, 'Tunjangan Transportasi', 20000);
88 •    select * from TunjanganKaryawan;
```

**Gambar 11.** Insert Data TunjanganKaryawan

```
90      -- Isi tabel Gaji, termasuk tambahan bonus untuk karyawan yang lembur
91 •    INSERT INTO Gaji (slip_gaji, ID_Karyawan, ID_Potongan, ID_Tunjangan, GajiPokok, Bonus, TanggalPembayaran)
92      VALUES
93          (1234, 1,1,111,5000000, 50000, '2024-03-31'),
94          (1235, 2,2,112, 4500000, 30000,  '2024-03-31'),
95          (1236, 3,2,113, 4800000, 40000,  '2024-03-31'),
96          (1237, 4,1,111,5000000, 50000, '2024-03-31'),
97          (1238, 5,2,112, 4500000, 30000,  '2024-03-31');
98
99 •    select*from Gaji;
```

**Gambar 12.** Insert Data Gaji

Source Code:

```
-- Isi tabel KaryawanDeleteDataKaryawanGajiDeleteDataKaryawanGaji
    INSERT INTO Karyawan (ID_Karyawan, Nama_Karyawan, Departemen, Jabatan, Tanggal_Lahir, Tanggal_Masuk, Lembur)
    VALUES
      (1, 'Wahyu Ozorah Manurung','HR', 'Manager', '1980-05-15', '2010-01-01', TRUE),
      (2, 'Torang Four Yones Manullang', 'CEO', 'Excutive', '1985-09-20', '2015-03-10', TRUE),
      (3, 'Revo','Kepala bidang', 'Pemasaran', '1980-05-15', '2010-01-01', FALSE),
      (4, 'Pandu','IT', 'Back-end', '1980-05-15', '2010-01-01', TRUE),
      (5, 'Rizki', 'Keuangan', 'Akuntan', '1990-12-12', '2018-07-05', FALSE);
    select*from Karyawan;

    -- Isi tabel Lembur
    INSERT INTO Lembur (ID_Lembur, ID_Karyawan, Uang_Lembur)
    VALUES
    (1,1,500000),
    (2,2,500000);
    select*from Lembur;

    -- Isi tabel PotonganGaji
    INSERT INTO PotonganGaji (ID_Potongan, Nama_Potongan, Persentase_Potongan)
    VALUES
      (1, 'Potongan Kehadiran', 2),
      (2, 'Potongan Telat', 5),
      (3, 'Potongan Telat', 5),
      (4, 'Potongan Telat', 5),
      (5, 'Potongan Telat', 5);
    select * from PotonganGaji;
```

```
-- Isi tabel TunjanganKaryawan
INSERT INTO TunjanganKaryawan (ID_Tunjangan, Nama_Tunjangan, Besaran_Tunjangan)
VALUES
    (111, 'Tunjangan Kesehatan', 300000),
    (112, 'Tunjangan Transportasi', 20000),
    (113, 'Tunjangan Transportasi', 20000),
    (114, 'Tunjangan Pendidikan', 2000000),
    (115, 'Tunjangan Transportasi', 20000);
select * from TunjanganKaryawan;

-- Isi tabel Gaji, termasuk tambahan bonus untuk karyawan yang lembur
INSERT INTO Gaji (slip_gaji, ID_Karyawan, ID_Potongan, ID_Tunjangan, GajiPokok, Bonus, TanggalPembayaran)
VALUES
    (1234, 1,1,111,5000000, 50000, '2024-03-31'),
    (1235, 2,2,112, 4500000, 30000,  '2024-03-31'),
    (1236, 3,2,113, 4800000, 40000,  '2024-03-31'),
    (1237, 4,1,111,5000000, 50000, '2024-03-31'),
    (1238, 5,2,112, 4500000, 30000,  '2024-03-31');
```

Penjelasan:

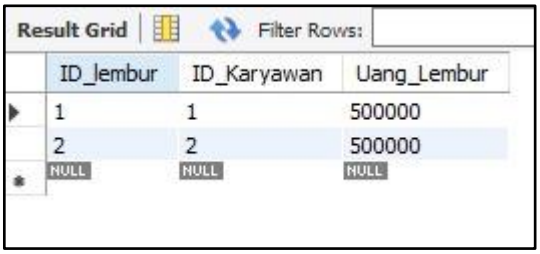
Kami telah mengisi beberapa tabel dalam database dengan data yang relevan. Dalam basis data yang kami buat, dilakukan serangkaian operasi SQL untuk mengelola informasi karyawan dan komponen gaji mereka. Pertama, terdapat tabel Karyawan yang berisi data identitas karyawan seperti ID, nama, departemen, jabatan, tanggal lahir, tanggal masuk, dan status lembur. Data karyawan dimasukkan ke dalam tabel ini menggunakan pernyataan INSERT INTO dengan VALUES yang sesuai. Selanjutnya, informasi lembur karyawan direkam dalam tabel Lembur, yang mencatat ID lembur, ID karyawan, dan jumlah uang lembur yang diterima oleh setiap karyawan. Pernyataan INSERT INTO digunakan untuk memasukkan data lembur ke dalam tabel ini. Tabel PotonganGaji mencatat jenis potongan gaji beserta persentasenya, seperti potongan kehadiran dan potongan telat. Data potongan gaji dimasukkan ke dalam tabel ini dengan pernyataan INSERT INTO yang sesuai. Kemudian, terdapat tabel TunjanganKaryawan yang berisi informasi tentang tunjangan yang diterima oleh setiap karyawan, seperti tunjangan kesehatan, transportasi, dan pendidikan. Data

tunjangan karyawan dimasukkan ke dalam tabel ini menggunakan pernyataan INSERT INTO. Terakhir, tabel Gaji digunakan untuk merekam slip gaji setiap karyawan, dengan detail gaji pokok, bonus, dan tanggal pembayaran. Bonus tambahan diberikan kepada karyawan yang lembur, yang tercermin dalam pernyataan INSERT INTO untuk tabel Gaji.

Melalui serangkaian pernyataan INSERT INTO, data relevan disimpan ke dalam masing-masing tabel sesuai dengan struktur dan kolom yang telah ditentukan sebelumnya. Ini memungkinkan untuk manajemen yang efisien atas informasi karyawan dan pengelolaan gaji mereka, serta memastikan bahwa semua komponen gaji yang relevan tercatat dengan baik dalam basis data.dan elemen-elemen lain yang terkait dengan manajemen sumber daya manusia secara efektif.

| ID_Karyawan | Nama_Karyawan | Departemen | Jabatan | Tanggal_Lahir | Tanggal_Masuk | Lembur |
|---|---|---|---|---|---|---|
| 1 | Wahyu Ozorah Manurung | HR | Manager | 1980-05-15 | 2010-01-01 | 1 |
| 2 | Torang Four Yones Manullang | CEO | Excutive | 1985-09-20 | 2015-03-10 | 1 |
| 3 | Revo | Kepala bidang | Pemasaran | 1980-05-15 | 2010-01-01 | 0 |
| 4 | Pandu | IT | Back-end | 1980-05-15 | 2010-01-01 | 1 |
| 5 | Rizki | Keuangan | Akuntan | 1990-12-12 | 2018-07-05 | 0 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL |

**Gambar 13.** Select Data Karyawan



Result Grid | Filter Rows:

| ID_lembur | ID_Karyawan | Uang_Lembur |
|---|---|---|
| 1 | 1 | 500000 |
| 2 | 2 | 500000 |
| NULL | NULL | NULL |

**Gambar 14.** Select Data Lembur

| ID_Potongan | Nama_Potongan | Persentase_Potongan |
|---|---|---|
| 1 | Potongan Kehadiran | 2 |
| 2 | Potongan Telat | 5 |
| 3 | Potongan Telat | 5 |
| 4 | Potongan Telat | 5 |
| 5 | Potongan Telat | 5 |
| NULL | NULL | NULL |

**Gambar 15.** Select Data Potong Gaji

| ID_Tunjangan | Nama_Tunjangan | Besaran_Tunjangan |
|---|---|---|
| 111 | Tunjangan Kesehatan | 300000 |
| 112 | Tunjangan Transportasi | 20000 |
| 113 | Tunjangan Transportasi | 20000 |
| 114 | Tunjangan Pendidikan | 2000000 |
| 115 | Tunjangan Transportasi | 20000 |
| NULL | NULL | NULL |

**Gambar 16.** Select Data Tunjangan Karyawan

| slip_gaji | ID_Karyawan | ID_Potongan | ID_Tunjangan | GajiPokok | Bonus | TanggalPembayaran |
|---|---|---|---|---|---|---|
| 1234 | 1 | 1 | 111 | 5000000 | 50000 | 2024-03-31 |
| 1235 | 2 | 2 | 112 | 4500000 | 30000 | 2024-03-31 |
| 1236 | 3 | 2 | 113 | 4800000 | 40000 | 2024-03-31 |
| 1237 | 4 | 1 | 111 | 5000000 | 50000 | 2024-03-31 |
| 1238 | 5 | 2 | 112 | 4500000 | 30000 | 2024-03-31 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL |

**Gambar 17.** Select Data Gaji

Penjelasan:

Pada Gambar 13, 14, 15, 16, dan 17 merupakan hasil dari pemanggilan dari masing masing tabel yang telah diisi data kedalamnya. Yang mana ada beberapa kolom yang isinya harus sama karena terhubung langsung dengan tabel lainnya.

3. Implementasi Join dan Nested Query

```
-- Implementasi Join dengan Pendapatan Kotor dan Pendapatan Bersih tanpa desimal --
-- Pendapatan kotor = Gaji pokok + Bonus + Tunjangan + Uang lembur
-- Pendapatan bersih = Pendapatan kotor - Persentase Potongan --
SELECT
    Gaji.slip_gaji AS ID_Slip_Gaji,
    Karyawan.Nama_Karyawan,
    ROUND((Gaji.GajiPokok + Gaji.Bonus + COALESCE(Tunjangan.Besaran_Tunjangan, 0) + COALESCE(Lembur.Uang_Lembur, 0)),
    ROUND((Gaji.GajiPokok + Gaji.Bonus + COALESCE(Tunjangan.Besaran_Tunjangan, 0) + COALESCE(Lembur.Uang_Lembur, 0)) -
FROM
    Gaji
LEFT JOIN Karyawan ON Gaji.ID_Karyawan = Karyawan.ID_Karyawan
LEFT JOIN TunjanganKaryawan AS Tunjangan ON Gaji.ID_Tunjangan = Tunjangan.ID_Tunjangan
LEFT JOIN Lembur ON Gaji.ID_Karyawan = Lembur.ID_Karyawan
LEFT JOIN PotonganGaji ON Gaji.ID_Potongan = PotonganGaji.ID_Potongan;
```

**Gambar 18.** JOIN Pendapatan Kotor

```
-- Pendapatan Kotor paling kecil--
SELECT
    Gaji.slip_gaji AS ID_Slip_Gaji,
    Karyawan.Nama_Karyawan,
    (Gaji.GajiPokok + Gaji.Bonus + COALESCE(Tunjangan.Besaran_Tunjangan, 0) + COALESCE(Lembur.Uang_Lembur, 0)) AS Pendapatan_Kotor
FROM
    Gaji
LEFT JOIN Karyawan ON Gaji.ID_Karyawan = Karyawan.ID_Karyawan
LEFT JOIN TunjanganKaryawan AS Tunjangan ON Gaji.ID_Tunjangan = Tunjangan.ID_Tunjangan
LEFT JOIN Lembur ON Gaji.ID_Karyawan = Lembur.ID_Karyawan
WHERE
    (Gaji.GajiPokok + Gaji.Bonus + COALESCE(Tunjangan.Besaran_Tunjangan, 0) + COALESCE(Lembur.Uang_Lembur, 0)) =
    (SELECT min(Pendapatan_Kotor) FROM (SELECT Gaji.GajiPokok + Gaji.Bonus + COALESCE(Tunjangan.Besaran_Tunjangan, 0) + COALESCE(Lembur.Uang_Lembur, 0) AS Pendapatan_Kotor FROM Gaji
    LEFT JOIN TunjanganKaryawan AS Tunjangan ON Gaji.ID_Tunjangan = Tunjangan.ID_Tunjangan
    LEFT JOIN Lembur ON Gaji.ID_Karyawan = Lembur.ID_Karyawan) AS Subquery);
```

**Gambar 19.** JOIN Pendapatan Kotor Paling Kecil beserta nested query

-- Implementasi Join--

-- Pendapatan kotor --

```
SELECT
    Gaji.slip_gaji AS ID_Slip_Gaji,
    Karyawan.Nama_Karyawan,
    (Gaji.GajiPokok + Gaji.Bonus + COALESCE(Tunjangan.Besaran_Tunjangan, 0) +
COALESCE(Lembur.Uang_Lembur, 0)) AS Pendapatan_Kotor
FROM
    Gaji
LEFT JOIN Karyawan ON Gaji.ID_Karyawan = Karyawan.ID_Karyawan
LEFT JOIN TunjanganKaryawan AS Tunjangan ON Gaji.ID_Tunjangan =
Tunjangan.ID_Tunjangan
LEFT JOIN Lembur ON Gaji.ID_Karyawan = Lembur.ID_Karyawan;
-- Implementasi Join dengan Pendapatan Kotor dan Pendapatan Bersih tanpa desimal --
-- Pendapatan kotor = Gaji pokok + Bonus + Tunjangan + Uang lembur
-- Pendapatan bersih = Pendapatan kotor - Persentase Potongan --
SELECT
    Gaji.slip_gaji AS ID_Slip_Gaji,
    Karyawan.Nama_Karyawan,
    ROUND((Gaji.GajiPokok + Gaji.Bonus + COALESCE(Tunjangan.Besaran_Tunjangan,
0) + COALESCE(Lembur.Uang_Lembur, 0)), 0) AS Pendapatan_Kotor,
    ROUND((Gaji.GajiPokok + Gaji.Bonus + COALESCE(Tunjangan.Besaran_Tunjangan,
0) + COALESCE(Lembur.Uang_Lembur, 0)) - (Persentase_Potongan * (Gaji.GajiPokok +
Gaji.Bonus + COALESCE(Tunjangan.Besaran_Tunjangan, 0) +
COALESCE(Lembur.Uang_Lembur, 0)) / 100), 0) AS Pendapatan_Bersih
FROM
    Gaji
LEFT JOIN Karyawan ON Gaji.ID_Karyawan = Karyawan.ID_Karyawan
LEFT JOIN TunjanganKaryawan AS Tunjangan ON Gaji.ID_Tunjangan =
Tunjangan.ID_Tunjangan
```

```
LEFT JOIN Lembur ON Gaji.ID_Karyawan = Lembur.ID_Karyawan
LEFT JOIN PotonganGaji ON Gaji.ID_Potongan = PotonganGaji.ID_Potongan;
-- Pendapatan Kotor paling kecil--
SELECT
    Gaji.slip_gaji AS ID_Slip_Gaji,
    Karyawan.Nama_Karyawan,
    (Gaji.GajiPokok + Gaji.Bonus + COALESCE(Tunjangan.Besaran_Tunjangan, 0) +
COALESCE(Lembur.Uang_Lembur, 0)) AS Pendapatan_Kotor
FROM
    Gaji
LEFT JOIN Karyawan ON Gaji.ID_Karyawan = Karyawan.ID_Karyawan
LEFT JOIN TunjanganKaryawan AS Tunjangan ON Gaji.ID_Tunjangan =
Tunjangan.ID_Tunjangan
LEFT JOIN Lembur ON Gaji.ID_Karyawan = Lembur.ID_Karyawan
WHERE
    (Gaji.GajiPokok + Gaji.Bonus + COALESCE(Tunjangan.Besaran_Tunjangan, 0) +
COALESCE(Lembur.Uang_Lembur, 0)) =
    (SELECT min(Pendapatan_Kotor) FROM (SELECT Gaji.GajiPokok + Gaji.Bonus +
COALESCE(Tunjangan.Besaran_Tunjangan, 0) + COALESCE(Lembur.Uang_Lembur, 0)
AS Pendapatan_Kotor FROM Gaji
        LEFT JOIN TunjanganKaryawan AS Tunjangan ON Gaji.ID_Tunjangan =
Tunjangan.ID_Tunjangan
        LEFT JOIN Lembur ON Gaji.ID_Karyawan = Lembur.ID_Karyawan) AS Subquery);
```

Penjelasan:

Dalam implementasi join untuk menghitung pendapatan kotor karyawan, kami menggunakan pernyataan SQL yang menggabungkan tabel-tabel terkait. Kami melakukan join antara tabel "Gaji", "Karyawan", "TunjanganKaryawan", dan "Lembur" untuk mendapatkan informasi yang diperlukan. Dengan menggunakan left join, kami memastikan bahwa setiap baris dari tabel "Gaji" dipertahankan, bahkan jika tidak ada entri yang sesuai

dalam tabel lainnya. Dalam hal ini, kami menghitung pendapatan kotor sebagai jumlah dari gaji pokok, bonus, besaran tunjangan, dan uang lembur, dengan mengambil keuntungan dari fungsi COALESCE untuk menangani nilai-nilai NULL.

Kemudian, dalam query yang menggabungkan pendapatan kotor dan pendapatan bersih, kami menambahkan perhitungan untuk pendapatan bersih dengan mengurangkan potongan gaji dari pendapatan kotor. Kami juga memanfaatkan fungsi dari nested query untuk membuat pendapatan bersihnya dan memanfaatkan tabel "PotonganGaji" untuk mengambil nilai persentase potongan yang sesuai dan mengurangkannya dari pendapatan kotor. Kami menggunakan fungsi ROUND untuk memastikan bahwa nilai-nilai pendapatan kotor dan bersih yang dihasilkan tidak memiliki desimal.

Terakhir, dalam query yang menemukan pendapatan kotor terkecil, kami menggunakan subquery untuk menemukan nilai minimum dari pendapatan kotor dalam tabel. Kami melakukan ini dengan mengambil pendapatan kotor dari setiap entri dalam tabel "Gaji", kemudian memilih nilai terkecil di antara mereka. Dengan demikian, kami dapat mengidentifikasi karyawan yang memiliki pendapatan kotor terendah dalam organisasi.

| ID_Slip_Gaji | Nama_Karyawan | Pendapatan_Kotor | Pendapatan_Bersih |
|---|---|---|---|
| 1234 | Wahyu Ozorah Manurung | 5850000 | 5733000 |
| 1235 | Torang Four Yones Manullang | 5050000 | 4797500 |
| 1236 | Revo | 4860000 | 4617000 |
| 1237 | Pandu | 5350000 | 5243000 |
| 1238 | Rizki | 4550000 | 4322500 |

**Gambar 20.** Output Pendapatan Kotor

| ID_Slip_Gaji | Nama_Karyawan | Pendapatan_Kotor |
|---|---|---|
| 1238 | Rizki | 4550000 |

**Gambar 21.** Output Pendapatan Kotor Paling Kecil

```
-- nested query untuk mencetak berdasarkan nama dan bulan --
-- Nested query untuk mencetak tampilan berdasarkan tanggal pemberian gaji --
SELECT
    ID_Slip_Gaji,
    Nama_Karyawan,
    Pendapatan_Kotor,
    Pendapatan_Bersih
FROM
    (SELECT
        Gaji.slip_gaji AS ID_Slip_Gaji,
        Karyawan.Nama_Karyawan,
        ROUND((Gaji.GajiPokok + Gaji.Bonus + COALESCE(Tunjangan.Besaran_Tunjangan, 0) + COALESCE(Lembur.Uang_Lembur, 0)), 0) AS Pendapatan_Kotor,
        ROUND((Gaji.GajiPokok + Gaji.Bonus + COALESCE(Tunjangan.Besaran_Tunjangan, 0) + COALESCE(Lembur.Uang_Lembur, 0)) - (Persentase_Potongan *
        (Gaji.GajiPokok + Gaji.Bonus + COALESCE(Tunjangan.Besaran_Tunjangan, 0) + COALESCE(Lembur.Uang_Lembur, 0)) / 100), 0) AS Pendapatan_Bersih
    FROM
        Gaji
    LEFT JOIN Karyawan ON Gaji.ID_Karyawan = Karyawan.ID_Karyawan
    LEFT JOIN TunjanganKaryawan AS Tunjangan ON Gaji.ID_Tunjangan = Tunjangan.ID_Tunjangan
    LEFT JOIN Lembur ON Gaji.ID_Karyawan = Lembur.ID_Karyawan
    LEFT JOIN PotonganGaji ON Gaji.ID_Potongan = PotonganGaji.ID_Potongan
    WHERE
        Gaji.TanggalPembayaran = '2024-03-31') AS Tanggal_Gaji;
```

**Gambar 22 .** Join dan Nested Query

Source Code:

```
-- nested query untuk mencetak berdasarkan nama dan bulan --
-- Nested query untuk mencetak tampilan berdasarkan tanggal pemberian gaji --
SELECT
    ID_Slip_Gaji,
    Nama_Karyawan,
    Pendapatan_Kotor,
    Pendapatan_Bersih
FROM
    (SELECT
        Gaji.slip_gaji AS ID_Slip_Gaji,
        Karyawan.Nama_Karyawan,
        ROUND((Gaji.GajiPokok + Gaji.Bonus +
COALESCE(Tunjangan.Besaran_Tunjangan, 0) + COALESCE(Lembur.Uang_Lembur,
0)), 0) AS Pendapatan_Kotor,
        ROUND((Gaji.GajiPokok + Gaji.Bonus +
COALESCE(Tunjangan.Besaran_Tunjangan, 0) + COALESCE(Lembur.Uang_Lembur,
0)) - (Persentase_Potongan *
        (Gaji.GajiPokok + Gaji.Bonus + COALESCE(Tunjangan.Besaran_Tunjangan, 0) +
COALESCE(Lembur.Uang_Lembur, 0)) / 100), 0) AS Pendapatan_Bersih
    FROM
        Gaji
    LEFT JOIN Karyawan ON Gaji.ID_Karyawan = Karyawan.ID_Karyawan
    LEFT JOIN TunjanganKaryawan AS Tunjangan ON Gaji.ID_Tunjangan =
Tunjangan.ID_Tunjangan
    LEFT JOIN Lembur ON Gaji.ID_Karyawan = Lembur.ID_Karyawan
    LEFT JOIN PotonganGaji ON Gaji.ID_Potongan = PotonganGaji.ID_Potongan
    WHERE
        Gaji.TanggalPembayaran = '2024-03-31') AS Tanggal_Gaji;
```

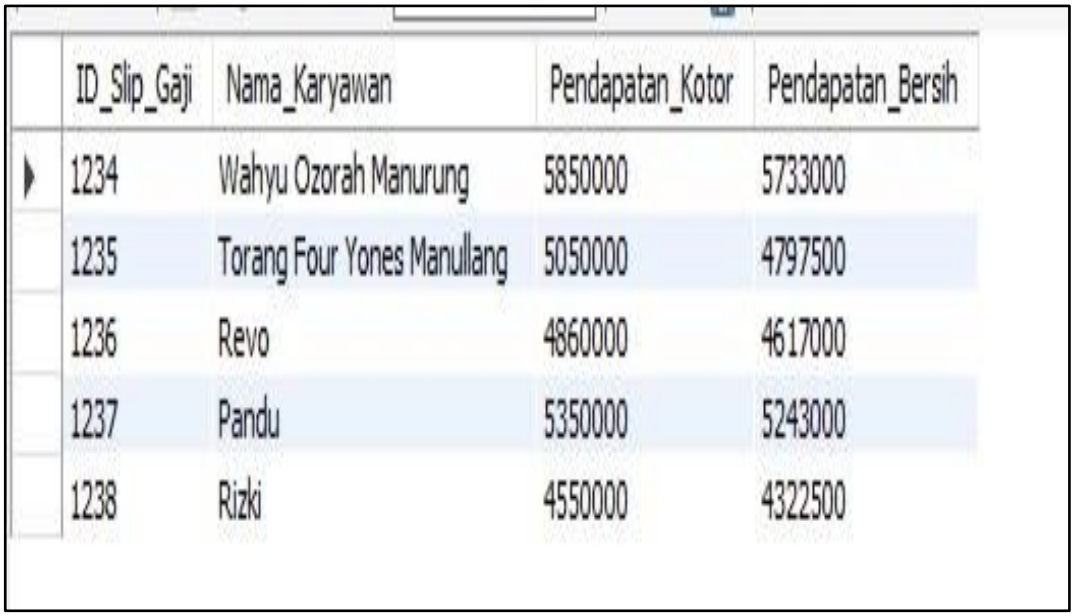
Pnejelasan:

Kode SQL yang diberikan merupakan sebuah kueri kompleks yang dirancang untuk menghasilkan informasi terperinci mengenai slip gaji karyawan pada tanggal pembayaran spesifik, yakni 31 Maret 2024. Kueri ini mengandalkan beberapa operasi penggabungan (join) dan pengolahan data dari berbagai tabel yang saling terkait. Pertama, kueri ini memanfaatkan subkueri (subquery) yang bertugas untuk menggabungkan beberapa tabel

kunci yang relevan dalam hal informasi gaji karyawan. Tabel-tabel tersebut mencakup data gaji dasar (Gaji), informasi karyawan (Karyawan), detail tunjangan karyawan (TunjanganKaryawan), catatan lembur (Lembur), dan jenis-jenis potongan gaji (PotonganGaji). Dengan menggunakan LEFT JOIN, kueri ini memastikan bahwa setiap baris dari tabel gaji (Gaji) akan dipertahankan, bahkan jika tidak ada kecocokan dengan baris dalam tabel-tabel lainnya. Proses ini memungkinkan data gaji tetap terdokumentasi secara lengkap, sementara data tambahan seperti informasi karyawan, tunjangan, lembur, dan potongan gaji diambil dari tabel-tabel terkait. Di dalam subkueri, dilakukan perhitungan untuk menentukan pendapatan kotor karyawan. Pendapatan kotor dihitung dengan menambahkan gaji pokok, bonus, besaran tunjangan (jika ada), dan uang lembur (jika ada).

Selanjutnya, pendapatan bersih dihitung dengan mengurangi persentase potongan gaji dari pendapatan kotor. Semua perhitungan tersebut dilakukan menggunakan fungsi-fungsi bawaan SQL seperti ROUND dan COALESCE untuk memastikan ketepatan hasil dan menangani nilai NULL dengan baik. Setelah data telah diproses dan dihitung dengan benar, kueri utama kemudian memilih kolom-kolom yang diinginkan untuk ditampilkan dalam hasil akhir. Kolom-kolom tersebut meliputi ID Slip Gaji, Nama Karyawan, Pendapatan Kotor, dan Pendapatan Bersih. Informasi ini kemudian diambil dari subkueri yang telah dibentuk sebelumnya. Dengan menggunakan pendekatan ini, kueri SQL tersebut mampu memberikan informasi yang mendalam dan terperinci tentang slip gaji karyawan pada tanggal pembayaran tertentu. Selain itu, kueri ini juga memastikan bahwa semua komponen gaji yang relevan, termasuk tunjangan, lembur, dan potongan gaji, terhitung secara akurat untuk menciptakan laporan gaji yang lengkap dan akurat bagi setiap karyawan.

Output:



| ID_Slip_Gaji | Nama_Karyawan | Pendapatan_Kotor | Pendapatan_Bersih |
|---|---|---|---|
| 1234 | Wahyu Ozorah Manurung | 5850000 | 5733000 |
| 1235 | Torang Four Yones Manullang | 5050000 | 4797500 |
| 1236 | Revo | 4860000 | 4617000 |
| 1237 | Pandu | 5350000 | 5243000 |
| 1238 | Rizki | 4550000 | 4322500 |

**Gambar 23.** Output

4. Implementasi Stored Procedure

```
-- ##################################PROCEDURE CREATE#########################################
DELIMITER $$
CREATE DEFINER=root@localhost PROCEDURE TambahDataKaryawanGaji(
    IN id_karyawan INT,
    IN nama_karyawan VARCHAR(50),
    IN departemen VARCHAR(50),
    IN jabatan VARCHAR(50),
    IN tanggal_lahir DATE,
    IN tanggal_masuk DATE,
    IN lembur BOOLEAN,
    IN id_lembur INT,
    IN uang_lembur INT,
    IN id_potongan INT,
    IN nama_potongan VARCHAR(50),
    IN persentase_potongan INT,
    IN id_tunjangan INT,
    IN nama_tunjangan VARCHAR(50),
    IN besaran_tunjangan INT,
    IN slip_gaji INT,
    IN gaji_pokok INT,
    IN bonus INT,
    IN tanggal_pembayaran DATE


)
BEGIN
    -- Tambah data ke tabel Karyawan
    INSERT INTO Karyawan (ID_Karyawan, Nama_Karyawan, Departemen, Jabatan, Tanggal_Lahir, Tanggal_Masuk, Lembur)
    VALUES (id_karyawan, nama_karyawan, departemen, jabatan, tanggal_lahir, tanggal_masuk, lembur);

    -- Tambah data ke tabel Lembur
    INSERT INTO Lembur (ID_Lembur, ID_Karyawan, Uang_Lembur)
    VALUES (id_lembur, id_karyawan, uang_lembur);

    -- Tambah data ke tabel PotonganGaji
    INSERT INTO PotonganGaji (ID_Potongan, Nama_Potongan, Persentase_Potongan)
    VALUES (id_potongan, nama_potongan, persentase_potongan);

    -- Tambah data ke tabel TunjanganKaryawan
    INSERT INTO TunjanganKaryawan (ID_Tunjangan, Nama_Tunjangan, Besaran_Tunjangan)
    VALUES (id_tunjangan, nama_tunjangan, besaran_tunjangan);

    -- Tambah data ke tabel Gaji
    INSERT INTO Gaji (slip_gaji, ID_Karyawan, ID_Potongan, ID_Tunjangan, GajiPokok, Bonus, TanggalPembayaran)
    VALUES (slip_gaji, id_karyawan, id_potongan, id_tunjangan, gaji_pokok, bonus, tanggal_pembayaran);
END
```

**Gambar 24.** Create Store Procedur

Source Code:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `TambahDataKaryawanGaji`(
    IN id_karyawan INT,
    IN nama_karyawan VARCHAR(50),
    IN departemen VARCHAR(50),
    IN jabatan VARCHAR(50),
```

```
    IN tanggal_lahir DATE,
    IN tanggal_masuk DATE,
    IN lembur BOOLEAN,
    IN id_lembur INT,
    IN uang_lembur INT,
    IN id_potongan INT,
    IN nama_potongan VARCHAR(50),
    IN persentase_potongan INT,
    IN id_tunjangan INT,
    IN nama_tunjangan VARCHAR(50),
    IN besaran_tunjangan INT,
    IN slip_gaji INT,
    IN gaji_pokok INT,
    IN bonus INT,
    IN tanggal_pembayaran DATE
)
BEGIN
    -- Tambah data ke tabel Karyawan
    INSERT INTO Karyawan (ID_Karyawan, Nama_Karyawan, Departemen, Jabatan,
Tanggal_Lahir, Tanggal_Masuk, Lembur)
    VALUES (id_karyawan, nama_karyawan, departemen, jabatan, tanggal_lahir,
tanggal_masuk, lembur);
    -- Tambah data ke tabel Lembur
    INSERT INTO Lembur (ID_Lembur, ID_Karyawan, Uang_Lembur)
    VALUES (id_lembur, id_karyawan, uang_lembur);
    -- Tambah data ke tabel PotonganGaji
    INSERT INTO PotonganGaji (ID_Potongan, Nama_Potongan, Persentase_Potongan)
    VALUES (id_potongan, nama_potongan, persentase_potongan);
    -- Tambah data ke tabel TunjanganKaryawan
    INSERT INTO TunjanganKaryawan (ID_Tunjangan, Nama_Tunjangan,
Besaran_Tunjangan)
    VALUES (id_tunjangan, nama_tunjangan, besaran_tunjangan);
    -- Tambah data ke tabel Gaji
```

```
    INSERT INTO Gaji (slip_gaji, ID_Karyawan, ID_Potongan, ID_Tunjangan,
GajiPokok, Bonus, TanggalPembayaran)
    VALUES (slip_gaji, id_karyawan, id_potongan, id_tunjangan, gaji_pokok, bonus,
tanggal_pembayaran);
END
```

Penjelasan:

Penggunaan kode di atas merupakan contoh dari pembuatan stored procedure MySQL yang bertujuan untuk menambahkan data karyawan beserta informasi gaji mereka ke dalam basis data. Prosedur yang didefinisikan memiliki nama `TambahDataKaryawanGaji` dan menerima sejumlah parameter, antara lain: ID karyawan, nama karyawan, departemen, jabatan, tanggal lahir, tanggal masuk, status lembur, informasi lembur (ID lembur dan uang lembur), informasi potongan gaji (ID potongan, nama potongan, dan persentase potongan), informasi tunjangan karyawan (ID tunjangan, nama tunjangan, dan besaran tunjangan), serta detail gaji (slip gaji, gaji pokok, bonus, dan tanggal pembayaran). Dalam prosedur ini, langkah-langkah yang dilakukan meliputi:

1. Penambahan data karyawan ke dalam tabel Karyawan menggunakan pernyataan INSERT INTO, dimana nilai-nilai parameter yang diterima dari pemanggil prosedur digunakan untuk mengisi kolom-kolom yang sesuai dalam tabel Karyawan.
2. Penambahan data lembur karyawan ke dalam tabel Lembur, dengan memasukkan nilai-nilai parameter yang sesuai.
3. Penambahan data potongan gaji ke dalam tabel PotonganGaji menggunakan nilai-nilai parameter yang diterima dari pemanggil prosedur.
4. Penambahan data tunjangan karyawan ke dalam tabel TunjanganKaryawan, dimana nilai-nilai parameter yang diterima digunakan untuk mengisi kolom-kolom yang sesuai.
5. Penambahan data gaji karyawan ke dalam tabel Gaji menggunakan nilai-nilai parameter yang diterima dari pemanggil prosedur.

Dengan menggunakan stored procedure seperti ini, pengguna dapat memasukkan data karyawan serta informasi gaji mereka dengan lebih efisien, karena prosedur telah terdefinisi dengan langkah-langkah yang spesifik dan dapat dipanggil kapan pun diperlukan tanpa perlu menulis ulang kode yang sama berulang kali. Ini dapat meningkatkan produktivitas dan konsistensi dalam pengelolaan data karyawan dan gaji mereka dalam sistem basis data.


**Gambar 25.** Call Create Store Procedur

Kode `CALL TambahDataKaryawanGaji(...)` merupakan pernyataan yang memanggil stored procedure MySQL yang telah didefinisikan sebelumnya dengan nama `TambahDataKaryawanGaji`. Stored procedure ini bertujuan untuk menambahkan data karyawan beserta informasi gaji mereka ke dalam basis data. Argumen yang diteruskan ke dalam stored procedure ini adalah nilai-nilai yang sesuai dengan parameter yang didefinisikan dalam prosedur tersebut. Secara berurutan, argumen yang diteruskan adalah sebagai berikut: 1. ID karyawan: 6 2. Nama karyawan: 'Manusia' 3. Departemen: 'Finance' 4. Jabatan: 'Koordinator' 5. Tanggal lahir: '1911-03-16' 6. Tanggal masuk: '2021-08-15' 7. Status lembur: FALSE (Tidak melakukan lembur) 8. ID lembur: 6 9. Uang lembur: 300000 10. ID potongan: 6 11. Nama potongan: 'Potongan Kehadiran' 12. Persentase potongan: 6 13. ID tunjangan: 6 14. Nama tunjangan: 'Tunjangan Transportasi' 15. Besaran tunjangan: 20000 16. Slip gaji: 1222 17. Gaji pokok: 5000000 18. Bonus: 50000 19. Tanggal pembayaran: '2024-03-31' Dengan melakukan pemanggilan stored procedure menggunakan kode `CALL`, semua langkah yang didefinisikan dalam prosedur tersebut akan dieksekusi secara berurutan untuk menambahkan data karyawan dan informasi gaji mereka ke dalam basis data. Ini memungkinkan untuk mengelola data karyawan dan gaji mereka dengan lebih efisien dan konsisten dalam sistem basis data.

Output:

| ID_Slip_Gaji | Nama_Karyawan | Pendapatan_Kotor |
|---|---|---|
| 1222 | Manusia | 5370000 |
| 1234 | Wahyu Ozorah Manurung | 5850000 |
| 1235 | Torang Four Yones Manullang | 5050000 |
| 1236 | Revo | 4860000 |
| 1237 | Pandu | 5350000 |
| 1238 | Rizki | 4550000 |

**Gambar 26.** Output

```
235     -- ##########################################################PROCEDURE READ##########################################################################
236     DELIMITER $$
237     CREATE DEFINER=root@localhost PROCEDURE ReadDataKaryawanGaji(IN karyawan_id INT)
238     BEGIN
239         -- Read data Karyawan
240         SELECT * FROM Karyawan WHERE ID_Karyawan = karyawan_id;
241
242         -- Read data Lembur
243         SELECT * FROM Lembur WHERE ID_Karyawan = karyawan_id;
244
245         -- Read data PotonganGaji
246         SELECT * FROM PotonganGaji;
247
248         -- Read data TunjanganKaryawan
249         SELECT * FROM TunjanganKaryawan;
250
251         -- Read data Gaji
252         SELECT * FROM Gaji WHERE ID_Karyawan = karyawan_id;
253     END
254
255     CALL ReadDataKaryawanGaji(13);
```

**Gambar 27.** Stored Procedure Read

Source Code:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `ReadDataKaryawanGaji`(IN karyawan_id INT)
BEGIN
    -- Read data Karyawan
    SELECT * FROM Karyawan WHERE ID_Karyawan = karyawan_id;
    -- Read data Lembur
    SELECT * FROM Lembur WHERE ID_Karyawan = karyawan_id;

    -- Read data PotonganGaji
    SELECT * FROM PotonganGaji;
    -- Read data TunjanganKaryawan
    SELECT * FROM TunjanganKaryawan;
    -- Read data Gaji
    SELECT * FROM Gaji WHERE ID_Karyawan = karyawan_id;
END
```

Penjelasan:

Kode yang diberikan adalah sebuah stored procedure MySQL dengan nama `ReadDataKaryawanGaji`. Prosedur ini bertujuan untuk membaca dan mengambil informasi terkait karyawan dan detail gaji mereka dari beberapa tabel dalam basis data. Langkah-langkah yang dilakukan dalam prosedur ini dimulai dengan membaca data karyawan, di mana pernyataan SELECT digunakan untuk mengambil semua kolom dari tabel Karyawan yang sesuai dengan ID karyawan yang diteruskan sebagai argumen saat pemanggilan stored procedure. Selanjutnya, prosedur membaca data lembur dari tabel Lembur, memberikan informasi apakah karyawan tersebut melakukan lembur atau tidak.

Kemudian, data potongan gaji dibaca dari tabel PotonganGaji, memberikan informasi tentang jenis-jenis potongan gaji yang diterapkan dalam perusahaan. Selanjutnya, prosedur membaca data tunjangan karyawan dari tabel TunjanganKaryawan, memberikan informasi tentang jenis-jenis tunjangan yang mungkin diberikan kepada karyawan. Terakhir, data gaji karyawan dibaca dari tabel Gaji, memberikan detail gaji karyawan seperti gaji pokok, bonus, dan tanggal pembayaran. Dengan menggunakan stored procedure seperti ini, pengguna dapat dengan mudah membaca dan mengambil informasi terperinci tentang karyawan dan gaji mereka dalam basis data hanya dengan memberikan ID karyawan sebagai argumen saat pemanggilan stored procedure. Ini memungkinkan untuk melakukan manajemen data yang lebih efisien dan konsisten dalam sistem basis data.

```
CALL ReadDataKaryawanGaji(1);
```

**Gambar 28.** Call Read

Output:

| ID_Karyawan | Nama_Karyawan | Departemen | Jabatan | Tanggal_Lahir | Tanggal_Masuk | Lembur |
|---|---|---|---|---|---|---|
| 1 | Wahyu Ozorah Manurung | HR | Manager | 1980-05-15 | 2010-01-01 | 1 |

**Gambar 29.** Output Call Read

```
-- ##########################################################PROCEDURE UPDATE#########################################################
DELIMITER $$
CREATE DEFINER=root@localhost PROCEDURE UpdateDataKaryawanGaji(
    IN karyawan_id INT,
    IN nama_karyawan VARCHAR(50),
    IN departemen VARCHAR(50),
    IN jabatan VARCHAR(50),
    IN tanggal_lahir DATE,
    IN tanggal_masuk DATE,
    IN lembur BOOLEAN,
    IN uang_lembur INT,
    IN id_potongan INT,
    IN nama_potongan VARCHAR(50),
    IN persentase_potongan INT,
    IN id_tunjangan INT,
    IN nama_tunjangan VARCHAR(50),
    IN besaran_tunjangan INT,
    IN gaji_pokok INT,
    IN bonus INT,
    IN tanggal_pembayaran DATE
)
BEGIN
    -- Update data di tabel Karyawan
    UPDATE Karyawan
    SET Nama_Karyawan = nama_karyawan,
        Departemen = departemen,
        Jabatan = jabatan,
        Tanggal_Lahir = tanggal_lahir,
        Tanggal_Masuk = tanggal_masuk,
        Lembur = lembur
    WHERE ID_Karyawan = karyawan_id;

    -- Update data di tabel Lembur
    UPDATE Lembur
    SET Uang_Lembur = uang_lembur
    WHERE ID_Karyawan = karyawan_id;

    -- Update data di tabel PotonganGaji
    UPDATE PotonganGaji
    SET Nama_Potongan = nama_potongan,
        Persentase_Potongan = persentase_potongan
    WHERE ID_Potongan = id_potongan;


    -- Update data di tabel TunjanganKaryawan
    UPDATE TunjanganKaryawan
    SET Nama_Tunjangan = nama_tunjangan,
        Besaran_Tunjangan = besaran_tunjangan
    WHERE ID_Tunjangan = id_tunjangan;

    -- Update data di tabel Gaji
    UPDATE Gaji
    SET GajiPokok = gaji_pokok,
        Bonus = bonus,
        TanggalPembayaran = tanggal_pembayaran
    WHERE ID_Karyawan = karyawan_id;
END
```

**Gambar 30.**Stored Procedure Update

Source Code:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `UpdateDataKaryawanGaji`(
    IN karyawan_id INT,
    IN nama_karyawan VARCHAR(50),
    IN departemen VARCHAR(50),
    IN jabatan VARCHAR(50),
    IN tanggal_lahir DATE,
    IN tanggal_masuk DATE,
    IN lembur BOOLEAN,
    IN uang_lembur INT,
    IN id_potongan INT,
    IN nama_potongan VARCHAR(50),
    IN persentase_potongan INT,
    IN id_tunjangan INT,
    IN nama_tunjangan VARCHAR(50),
    IN besaran_tunjangan INT,
    IN gaji_pokok INT,
    IN bonus INT,
    IN tanggal_pembayaran DATE
)
BEGIN
    -- Update data di tabel Karyawan
    UPDATE Karyawan
    SET Nama_Karyawan = nama_karyawan,
        Departemen = departemen,
        Jabatan = jabatan,
        Tanggal_Lahir = tanggal_lahir,
        Tanggal_Masuk = tanggal_masuk,
        Lembur = lembur
    WHERE ID_Karyawan = karyawan_id;

    -- Update data di tabel Lembur
    UPDATE Lembur
    SET Uang_Lembur = uang_lembur
    WHERE ID_Karyawan = karyawan_id;
```

```
    -- Update data di tabel PotonganGaji
    UPDATE PotonganGaji
    SET Nama_Potongan = nama_potongan,
        Persentase_Potongan = persentase_potongan
    WHERE ID_Potongan = id_potongan;

    -- Update data di tabel TunjanganKaryawan
    UPDATE TunjanganKaryawan
    SET Nama_Tunjangan = nama_tunjangan,
        Besaran_Tunjangan = besaran_tunjangan
    WHERE ID_Tunjangan = id_tunjangan;

    -- Update data di tabel Gaji
    UPDATE Gaji
    SET GajiPokok = gaji_pokok,
        Bonus = bonus,
        TanggalPembayaran = tanggal_pembayaran
    WHERE ID_Karyawan = karyawan_id;
END
```

Penjelasan:

Kode yang diberikan adalah sebuah stored procedure MySQL dengan nama `ReadDataKaryawanGaji`. Prosedur ini bertujuan untuk membaca dan mengambil informasi terkait karyawan dan detail gaji mereka dari beberapa tabel dalam basis data. Langkah-langkah yang dilakukan dalam prosedur ini dimulai dengan membaca data karyawan, di mana pernyataan SELECT digunakan untuk mengambil semua kolom dari tabel Karyawan yang sesuai dengan ID karyawan yang diteruskan sebagai argumen saat pemanggilan stored procedure. Selanjutnya, prosedur membaca data lembur dari tabel Lembur, memberikan informasi apakah karyawan tersebut melakukan lembur atau tidak. Kemudian, data potongan gaji dibaca dari tabel PotonganGaji, memberikan informasi tentang jenis-jenis potongan gaji yang diterapkan dalam perusahaan.

Selanjutnya, prosedur membaca data tunjangan karyawan dari tabel TunjanganKaryawan, memberikan informasi tentang jenis-jenis tunjangan yang mungkin diberikan kepada karyawan. Terakhir, data gaji karyawan dibaca dari tabel Gaji, memberikan detail gaji karyawan seperti gaji pokok, bonus, dan tanggal pembayaran.

Dengan menggunakan stored procedure seperti ini, pengguna dapat dengan mudah membaca dan mengambil informasi terperinci tentang karyawan dan gaji mereka dalam basis data hanya dengan memberikan ID karyawan sebagai argumen saat pemanggilan stored procedure. Ini memungkinkan untuk melakukan manajemen data yang lebih efisien dan konsisten dalam sistem basis data.

```
CALL UpdateDataKaryawanGaji(6, 'Wahyu Ozorah Manurung', 'HR', 'Manager', '1980-05-15',
'2010-01-01', TRUE, 250000, 1, 'Potongan Kehadiran', 5, 1,
'Tunjangan Kesehatan', 300000, 5500000,
 600000, '2024-03-31');
```

**Gambar 31.** Call Update

Output:

| ID_Slip_Gaji | Nama_Karyawan | Pendapatan_Kotor |
|---|---|---|
| 1222 | Wahyu Ozorah Manurung | 6650000 |
| 1234 | Wahyu Ozorah Manurung | 5850000 |
| 1235 | Torang Four Yones Manullang | 5330000 |
| 1236 | Revo | 5140000 |
| 1237 | Pandu | 5350000 |
| 1238 | Rizki | 4830000 |

**Gambar 32.** Output

```
317     -- ##########################################################################################PROCEDURE DELETE##########################################################################################
318     DELIMITER $$
319     CREATE DEFINER=root@localhost PROCEDURE DeleteDataKaryawanGaji(
320         IN karyawan_id INT
321     )
322     BEGIN
323         -- Hapus data di tabel Gaji
324         DELETE FROM Gaji WHERE ID_Karyawan = karyawan_id;
325
326         -- Hapus data di tabel TunjanganKaryawan
327         DELETE FROM TunjanganKaryawan WHERE ID_Tunjangan IN (SELECT ID_Tunjangan FROM Gaji WHERE ID_Karyawan = karyawan_id);
328
329         -- Hapus data di tabel PotonganGaji
330         DELETE FROM PotonganGaji WHERE ID_Potongan IN (SELECT ID_Potongan FROM Gaji WHERE ID_Karyawan = karyawan_id);
331
332         -- Hapus data di tabel Lembur
333         DELETE FROM Lembur WHERE ID_Karyawan = karyawan_id;
334
335         -- Hapus data di tabel Karyawan
336         DELETE FROM Karyawan WHERE ID_Karyawan = karyawan_id;
337     END
338
339     CALL DeleteDataKaryawanGaji(10); -- ID karyawan yang ingin dihapus
340     DELIMITER $$
```

**Gambar 33.** Stored Procedure Deletete

Source Code:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `DeleteDataKaryawanGaji`(
    IN karyawan_id INT
)
BEGIN
    -- Hapus data di tabel Gaji
```

```
    DELETE FROM Gaji WHERE ID_Karyawan = karyawan_id;

    -- Hapus data di tabel TunjanganKaryawan
    DELETE FROM TunjanganKaryawan WHERE ID_Tunjangan IN (SELECT ID_Tunjangan FROM Gaji WHERE ID_Karyawan = karyawan_id);

    -- Hapus data di tabel PotonganGaji
    DELETE FROM PotonganGaji WHERE ID_Potongan IN (SELECT ID_Potongan FROM Gaji WHERE ID_Karyawan = karyawan_id);

    -- Hapus data di tabel Lembur
    DELETE FROM Lembur WHERE ID_Karyawan = karyawan_id;

    -- Hapus data di tabel Karyawan
    DELETE FROM Karyawan WHERE ID_Karyawan = karyawan_id;
END
```

Penjelasan:

Kode yang diberikan merupakan sebuah stored procedure MySQL dengan nama `DeleteDataKaryawanGaji`. Tujuannya adalah untuk menghapus data terkait karyawan dan informasi gaji mereka dari beberapa tabel dalam basis data. Prosedur ini hanya memerlukan satu parameter, yaitu ID karyawan yang akan dihapus. Langkah-langkah yang dijalankan dalam prosedur ini adalah sebagai berikut:

1. **Hapus data gaji karyawan**: Dengan menggunakan pernyataan DELETE, prosedur ini menghapus semua entri yang berkaitan dengan ID karyawan yang diteruskan sebagai argumen dari tabel Gaji. Ini termasuk semua catatan gaji pokok, bonus, dan tanggal pembayaran yang terkait dengan karyawan tersebut.
2. **Hapus data tunjangan karyawan**: Selanjutnya, prosedur ini menghapus semua entri dari tabel TunjanganKaryawan dimana ID tunjangan terdapat dalam hasil subkueri yang mengambil ID tunjangan dari tabel Gaji berdasarkan ID karyawan yang diteruskan. Ini memastikan bahwa semua tunjangan yang diberikan kepada karyawan tersebut juga dihapus.
3. **Hapus data potongan gaji**: Data potongan gaji juga dihapus dari tabel PotonganGaji berdasarkan ID potongan yang terkait dengan karyawan yang akan dihapus dari tabel Gaji.
4. **Hapus data lembur karyawan**: Prosedur ini juga menghapus semua entri dari tabel

Lembur dimana ID karyawan sama dengan ID karyawan yang diteruskan sebagai argumen.

5. **Hapus data karyawan**: Terakhir, data karyawan sendiri dihapus dari tabel Karyawan berdasarkan ID karyawan yang diteruskan. Dengan menggunakan stored procedure ini, pengguna dapat dengan mudah dan efisien menghapus seluruh informasi terkait dengan seorang karyawan dari basis data. Ini memastikan bahwa data yang sudah tidak relevan lagi dapat dihapus dengan cepat dan konsisten, mengoptimalkan pengelolaan data dalam sistem basis data.

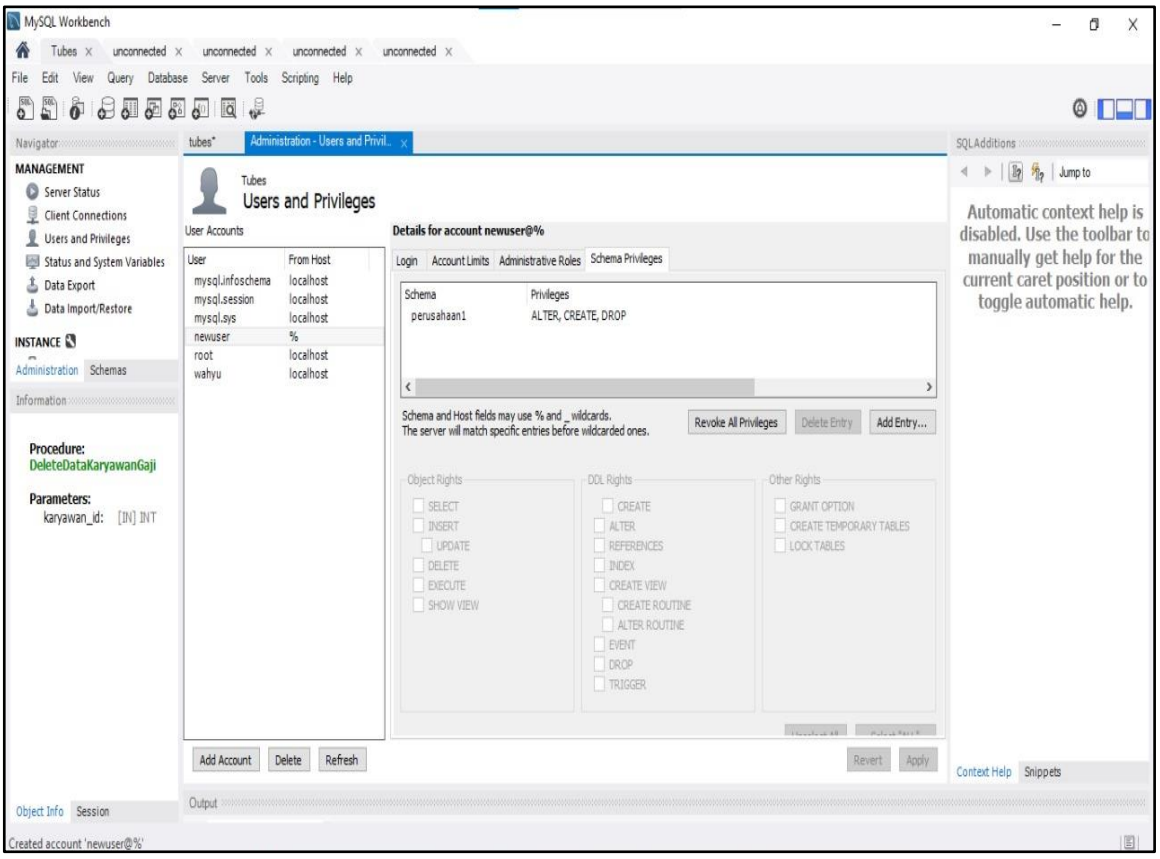
```
CALL DeleteDataKaryawanGaji(6);
```

**Gambar 34.** Output

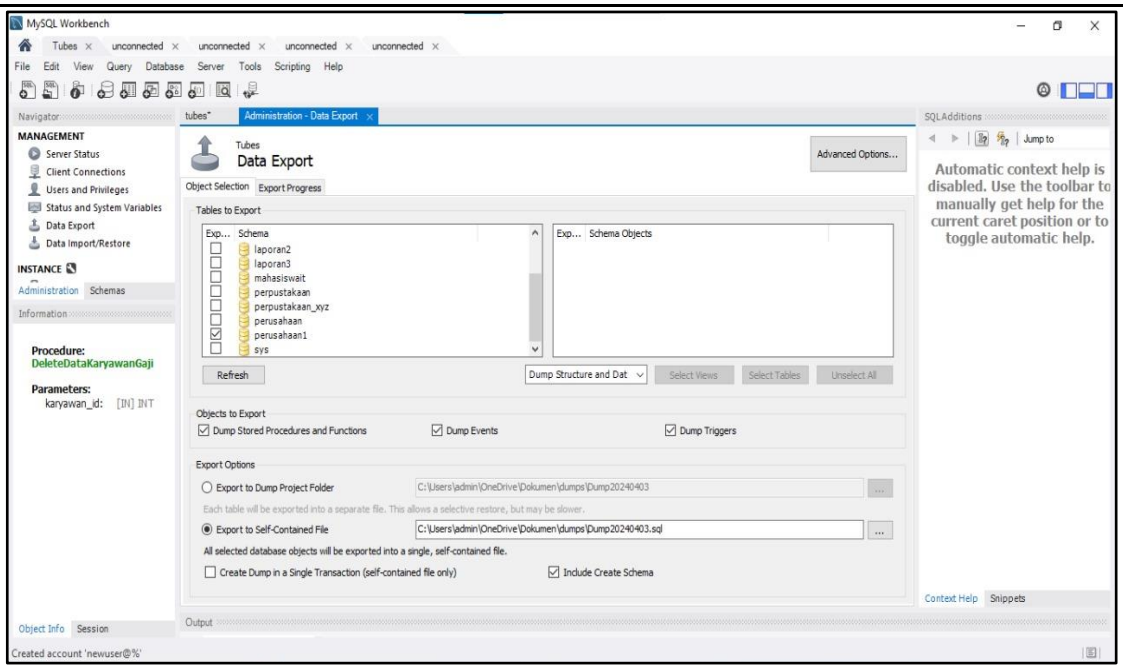
| ID_Slip_Gaji | Nama_Karyawan | Pendapatan_Kotor |
|---|---|---|
| 1234 | Wahyu Ozorah Manurung | 5850000 |
| 1235 | Torang Four Yones Manullang | 5330000 |
| 1236 | Revo | 5140000 |
| 1237 | Pandu | 5350000 |
| 1238 | Rizki | 4830000 |

**Gambar 35.** Output

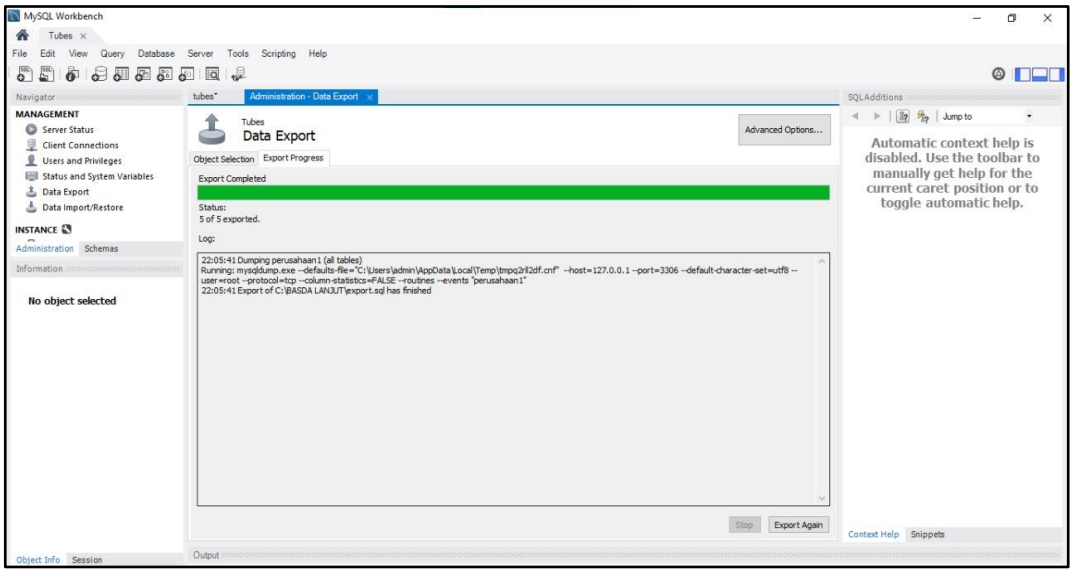
5. Implementasi Data Control Language (DCL)



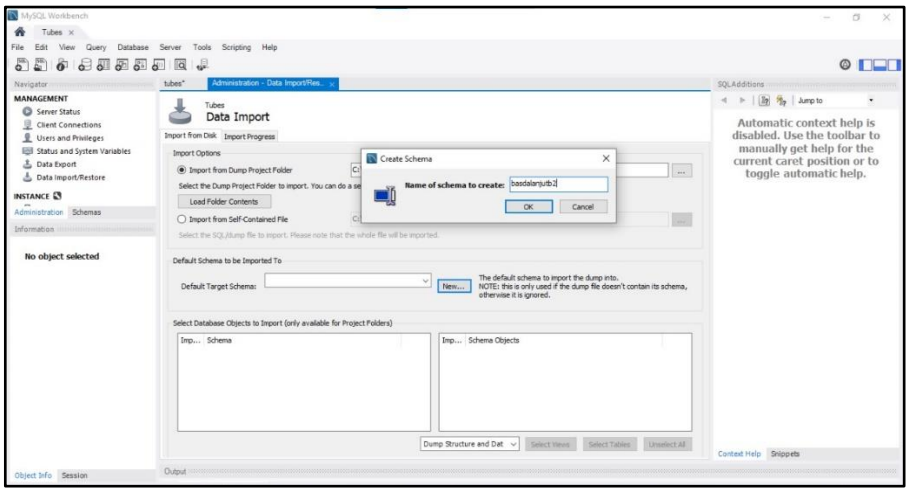

**Gambar 36.** Menambah Account




**Gambar 37.** Data Export

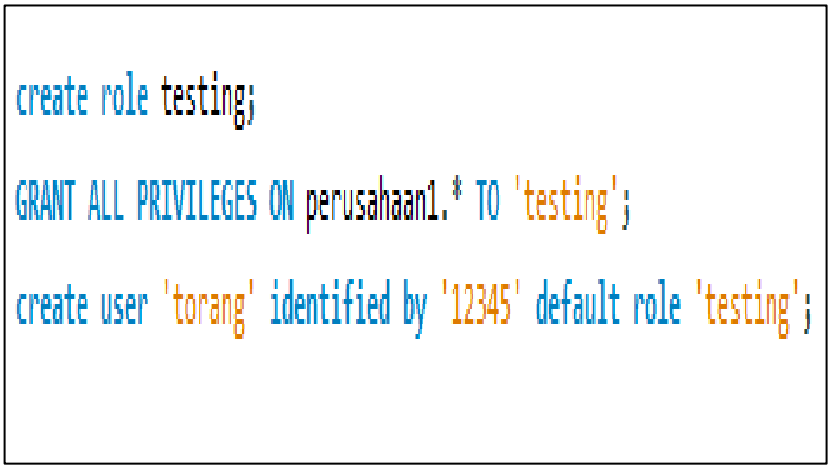



**Gambar 38.** Berhasil Export




**Gambar 39.** Import

6. Grant dan Revoke




**Gambar 40.** Grant

Source Code:

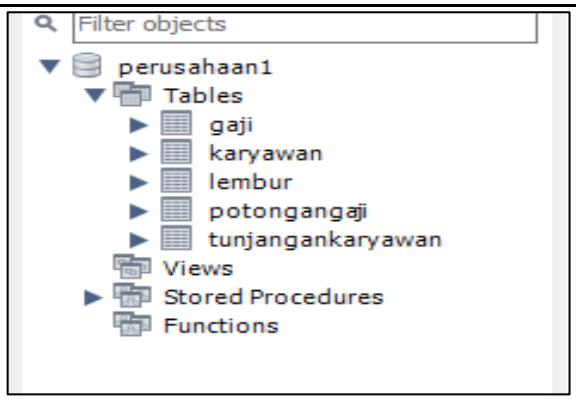
```
create role testing;
GRANT ALL PRIVILEGES ON perusahaan1.* TO 'testing';
create user 'torang' identified by '12345' default role 'testing';
```

Penjelasan:

Dalam perintah SQL di atas, terdapat serangkaian instruksi yang bertujuan untuk mengelola akses pengguna dalam basis data MySQL. Pertama, perintah `CREATE ROLE testing;` digunakan untuk membuat peran baru dengan nama 'testing'. Peran ini akan digunakan untuk mengelola hak akses tertentu di dalam basis data. Selanjutnya, perintah `GRANT ALL PRIVILEGES ON perusahaan1.* TO 'testing';` diberikan untuk memberikan semua hak akses (ALL PRIVILEGES) pada semua objek (dalam hal ini, tabel, prosedur, dll.) yang ada dalam basis data 'perusahaan1' kepada peran 'testing'. Ini berarti peran 'testing' akan memiliki hak akses penuh terhadap semua objek dalam basis data 'perusahaan1'. Terakhir, perintah `CREATE USER 'torang' identified by '12345' default role 'testing';` digunakan untuk membuat pengguna baru dengan nama 'torang' dan kata sandi '12345'.

Pengguna ini secara default diberi peran 'testing', yang berarti pengguna 'torang' akan memiliki hak akses yang sama seperti yang diberikan kepada peran 'testing'. Dengan cara ini, pengguna 'torang' akan dapat mengakses dan mengelola objek dalam basis data 'perusahaan1' sesuai dengan hak akses yang telah ditetapkan untuk peran 'testing'. Dengan menggunakan serangkaian perintah ini, pengguna 'torang' akan memiliki hak akses yang diberikan oleh peran 'testing' ke dalam basis data 'perusahaan1', memungkinkannya untuk melakukan operasi seperti membaca, menulis, dan mengelola data sesuai dengan kebutuhan yang telah ditentukan.

Output:




**Gambar 41.** Output Grant

```
REVOKE ALL PRIVILEGES ON perusahaan1.* FROM testing;
```

**Gambar 42.** Revoke

Source Code:

REVOKE ALL PRIVILEGES ON perusahaan1.* FROM testing;

Penjelasan:

Dalam perintah SQL di atas, terdapat serangkaian instruksi yang bertujuan untuk mengelola akses pengguna dalam basis data MySQL. Pertama, perintah `CREATE ROLE testing;` digunakan untuk membuat peran baru dengan nama 'testing'. Peran ini akan digunakan untuk mengelola hak akses tertentu di dalam basis data. Selanjutnya, perintah `GRANT ALL PRIVILEGES ON perusahaan1.* TO 'testing';` diberikan untuk memberikan semua hak akses (ALL PRIVILEGES) pada semua objek (dalam hal ini, tabel, prosedur, dll.) yang ada dalam basis data 'perusahaan1' kepada peran 'testing'. Ini berarti peran 'testing' akan memiliki hak akses penuh terhadap semua objek dalam basis data 'perusahaan1'. Terakhir, perintah `CREATE USER 'torang' identified by '12345' default role 'testing';` digunakan untuk membuat pengguna baru dengan nama 'torang' dan kata sandi '12345'.

Pengguna ini secara default diberi peran 'testing', yang berarti pengguna 'torang' akan memiliki hak akses yang sama seperti yang diberikan kepada peran 'testing'. Dengan cara ini, pengguna 'torang' akan dapat mengakses dan mengelola objek dalam basis data 'perusahaan1' sesuai dengan hak akses yang telah ditetapkan untuk peran 'testing'. Dengan menggunakan serangkaian perintah ini, pengguna 'torang' akan memiliki hak akses yang diberikan oleh peran 'testing' ke dalam basis data 'perusahaan1', memungkinkannya untuk melakukan operasi seperti membaca, menulis, dan mengelola data sesuai dengan kebutuhan yang telah ditentukan.

# KESIMPULAN DAN SARAN

## A. Kesimpulan

Pada akhirnya dapat disimpulkan bahwasanya pada proyek basis data ini kita dapat membuat sebuah database yang mana database ini akan disesuaikan dengan yang kita inginkan dan padakali ini itu sesuai dengan kasus yang tekah diberikan. saat kita ingin membuat database ada hal yang perlu kita perhatikan dengan baik yaitu kode atau source code nya agar mendapatkan hasil yang maksimal.

Pada database ini kita bisa membuat dengan membuat create database. Pada database kita bisa membuat terlebih dahulu tabel dan atribut-atribut nya yang akan kita buat. Setelah kita membuat atributya lalu kita membuat isinya yaitu dengan INSERT INTO… values yang mana ketika kita sudah membuat nya sesuai dengan urutan

Kemudian juga membuat join adalah untuk menggabungkan bebarapa kolom dari tabel yang ada Dimana ada 4 fungsi join yaitu inner join, left join, right join, dan terakhir adalah union join atau full join. Kemudian ada nested query yang mana menggunkan where dalam operasinya.

Pada basis data juga ada yang namanya procedur yang mana sekumpulan instruksi atau pernyataan SQL yang telah diberi nama dan disimpan di dalam basis data untuk eksekusi berulang kali. Kemudian ada juga DCL yang mana ini adalah untuk memberi akses dalam proses database yang dibuat.

## B. Saran

Pada pratikum ini terdapat beberapa saran yang baik untuk kedepannya.Yang mana terbagi menjadi 2 yaitu untuk orang yang akan melaksanakan pratikum dan untuk asisten dosen.

1. Untuk orang yang akan melakukan pratikum.

   Alangkah lebih baik untuk menginstal terlebih dahulu sebelum pratikum agar dapat mempermudah dalam pelaksanaan pratikumnya. Kemudian untuk pratikumnya alangkah lebih baiknya memahami terlebih dahulu dasar dasar dari membuat tabel, database dan mempelajari mengenai primary serta uniq agar mudah memahaminya dan pahami juga bagaiman mengupdate serta mendelete data.

2. Untuk Asisten Dosen

   Semoga lebih baik lagi dan untuk videonya kadang proses yang dilakukan tidak terlihat.

## DAFTAR PUSTAKA


Andaru, A. (2018). Pengertian database secara umum. *OSF Prepr, 2*..

Enterprise, J. (2014). *MySQL untuk pemula*. Elex Media Komputindo.

Gayatri, I. S. (2024). Analisis Manfaat Penerapan Basis Data dalam Lingkungan Sekolah di Indonesia. *Comit: Communication, Information and Technology Journal*, 2(1), 102-107.

Kadir, A. (2002). Penuntun Praktis Belajar SQL. Andi Yogyakarta.

Raissa , A. (2022). BASIS DATA BUKU AJAR. Bandung: CV. MEDIA SAINS INDONESIA.

Ramakrishnan, Raghu, and Johannes Gehkre. Database Management Systems Third Edition. Mc Graw Hill, 2003.

Saputra, A. (2012). Manajemen basis data mysql pada situs ftp lapan bandung. *Berita Dirgantara*, 13(4).

**KEMENTERIAN RISET, TEKNOLOGI DAN PENDIDIKAN**
**UNIVERSITAS BENGKULU**
**FAKULTAS TEKNIK**
**PROGRAM STUDI INFORMATIKA**
Jl. Wr. Supratman Kandang Limun, Bengkulu
Bengkulu 38371 A Telp: (0736) 344087, 22105-227

# LEMBAR ASISTENSI TUGAS BESAR
# BASIS DATA LANJUT

| | | | |
|---|---|---|---|
| Nama Anggota | : | 1. Torang Four Yones Manulang | (G1A022052) |
| | | 2. Rizki Ramadani Dalimunthe | (G1A022054) |
| | | 3. Revo Pratama | (G1A022058) |
| | | 4. Wahyu Ozorah Manurung | (G1A022060) |
| | | 5. Pandu Rizki Mulyanto | (G1A022076) |
| Dosen | : | 1. Mochammad Yusa, S.Kom.,M.Kom | |
| | | 2. Endina Putri Purwandari, S.T., M.Kom. | |
| Asisten Praktikum | : | Gita Dwi Putriani | (G1A019043) |
| | | Valleryan Virgil Zuliuskandar | (G1A020021) |
| | | Syafrizza Aulia Marizky | (G1A020029) |
| | | Muhadzib Nursaid | (G1A020035) |
| | | Dwinta Septiana | (G1A020041) |
| | | Muhammad Willdhan Arya Putra | (G1A020095) |
| | | Ilham Dio Putra | (G1A021024) |
| | | Kahfi Zairan | (G1A021041) |
| | | Esti Asmareta Ayu | (G1A021042) |
| | | Arief Satrio Budi Prasojo | (G1A021076) |

| Laporan Praktikum | Catatan dan Tanda Tangan |
|---|---|
| Laporan Tugas Besar | |