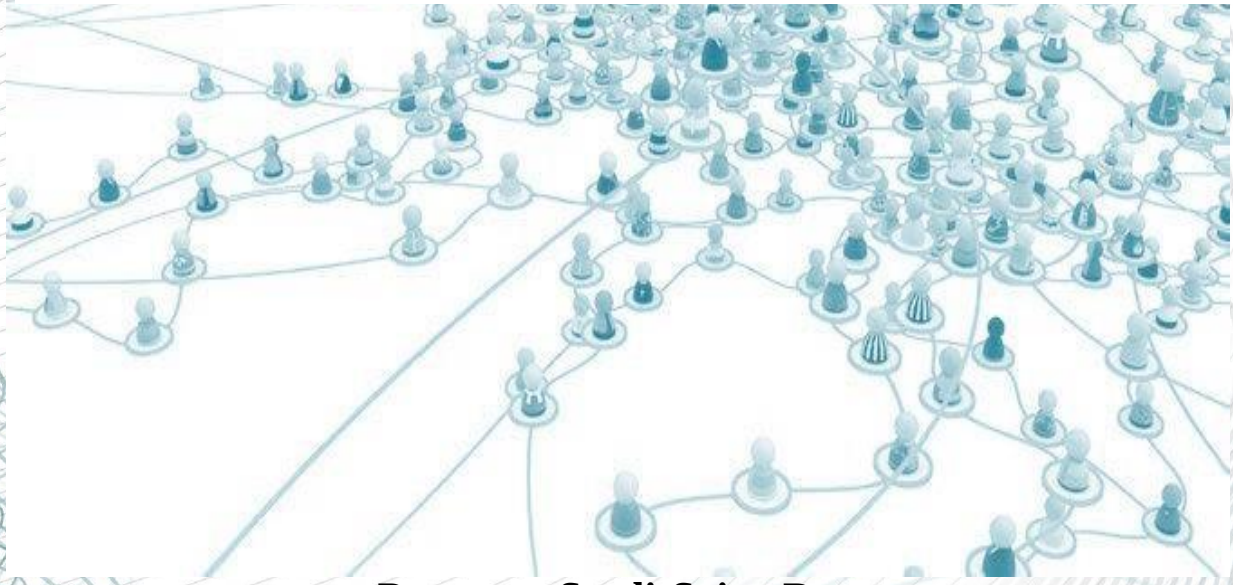
MODUL PRAKTIKUM

SD3203-Teknologi Basis Data



Program Studi Sains Data
Fakultas Sains
Institut Teknologi Sumatera

2025

MODUL 2

Schema Tuning

Schema Tuning

1. Tujuan Praktikum

a. Memahami konsep dasar dan melakukan tuning skema dalam basis data.
b. Mampu menerapkan beberapa metode tuning skema pada basis data.

2. Konsep Dasar

2.1. Tuning Performa

Pada *Database Management System* (DBMS) tuning performa atau kinerja dilakukan untuk meningkatkan kinera sistem agar lebih optimal dan menghemat sumber daya. Tuning performa dilakukan dengan menyesuaikan berbagai parameter serta memilih desain yang sesuai (tidak memberatkan sistem) sehingga performa atau kinerja meningkat. Tuning performa dapat dilakukan pada level *hardware* atau penyimpanannya hingga level yang lebih tinggi seperti skema, indeks, serta kuerinya.

2.2. Tuning Skema dan Metodenya

Tuning skema dilakukan dengan melakukan penyesuaian pada level skema. Penyesuaian dapat dilakukan dengan membagi, menambah, atau menggabungkan tabel-tabel dalam skema. Beberapa cara yang dapat dilakukan diantaranya:

- Splitting atau membagi tabel baik secara horizontal maupun vertikal
- Denormalisasi yang dapat berupa menambah kolom, menambah atribut dari atribut lain, menggabungkan tabel, serta menyalin tabel.
- Mengumpulkan tabel-tabel yang sering diakses ke dalam disk atau ruang penyimpanan yang sama.

3. Database

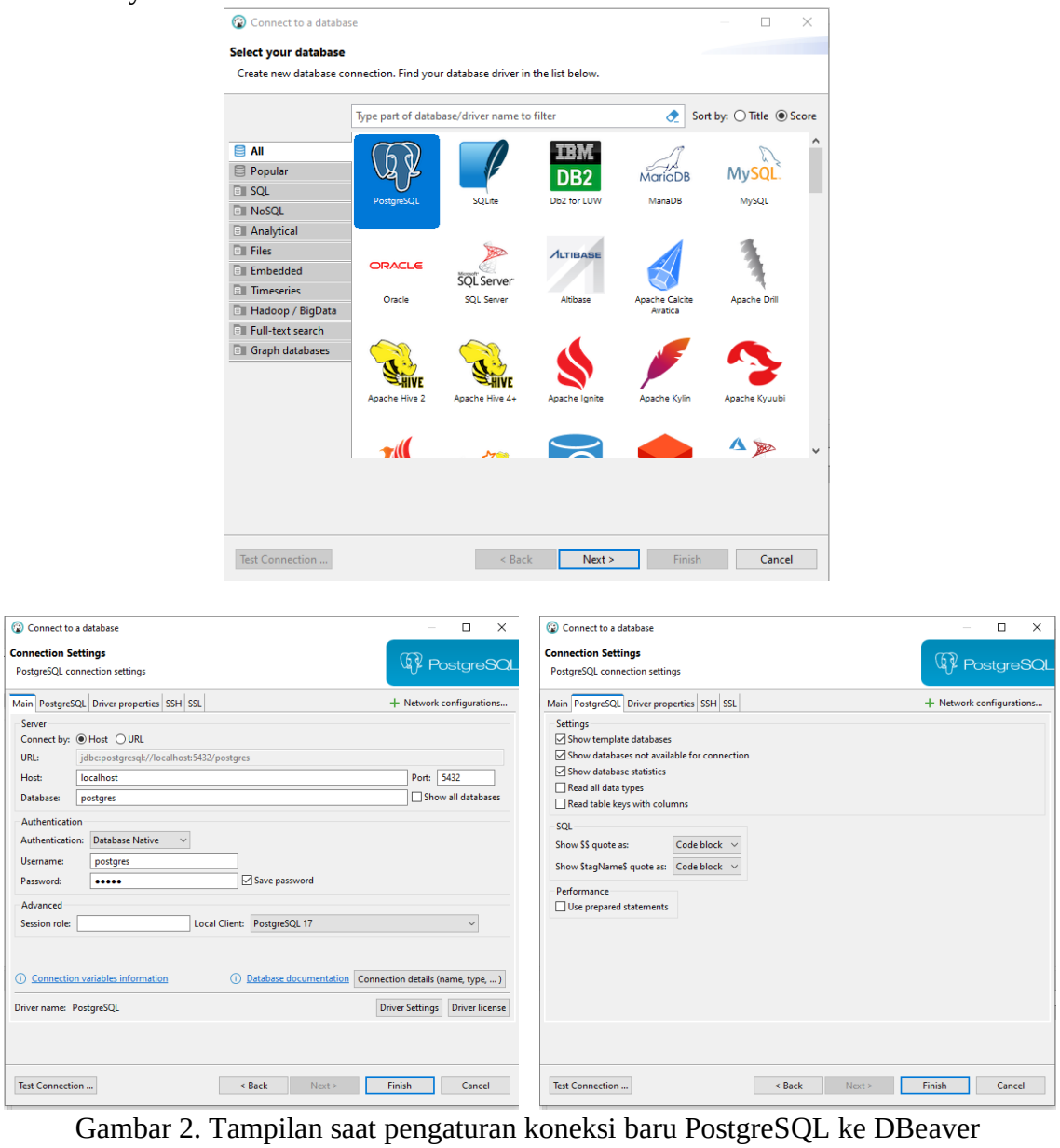
3.1.Koneksi Database

Pada praktikum ini DBMS yang digunakan adalah PostgreSQL sedangkan untuk aplikasi pengelolaan yang digunakan adalah DBeaver. Pastikan PostgreSQL dan DBeaver sudah terinstal pada komputer sebelum memulai praktikum. Jika sudah terpasang, pastikan PostgreSQL telah terkoneksi pada DBeaver.

Gambar 1. PostgreSQL dan DBeaver

Berikut adalah cara melakukan koneksi PostgreSQL pada DBeaver.

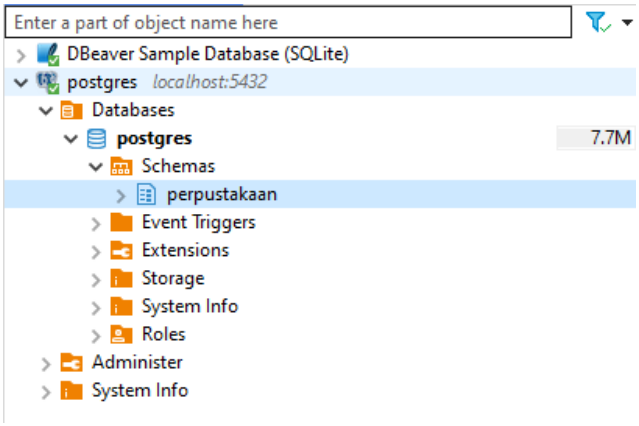
1) Buka aplikasi **DBeaver.**
2) Klik **Database** > **New Database Connection**
3) Pilih PostgreSQL kemudian klik **Next.**
4) Isi password dengan password yang sama saat instalasi PostgreSQL kemudian pada tab **PostgreSQL** ceklis semua pilihan yang ada. Tes koneksi dengan klik **Test Connection** pada bagian bawah. Jika sudah terkoneksi pilih **Finish** untuk menyelesaikan koneksi.



Gambar 2. Tampilan saat pengaturan koneksi baru PostgreSQL ke DBeaver

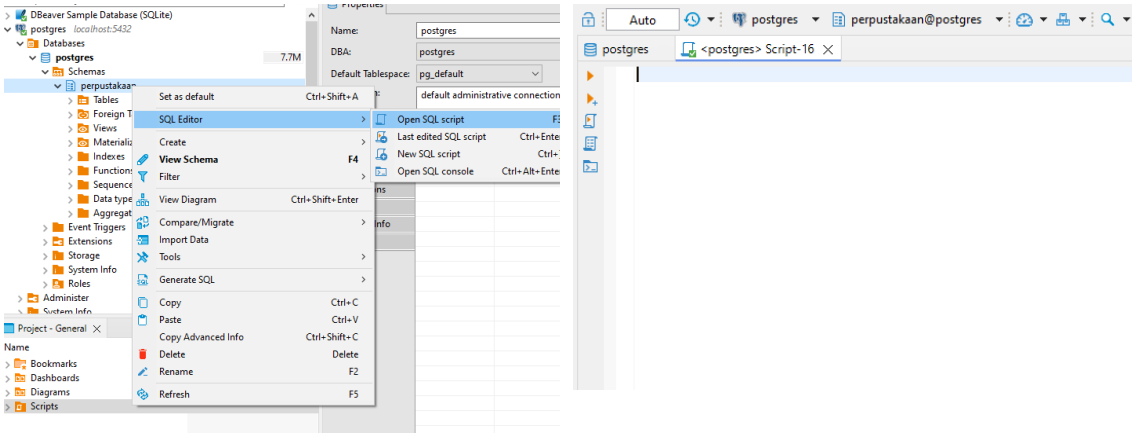
3.2. Pembuatan Database

Pada praktikum skema tuning ini, data yang digunakan adalah data sederhana berupa data peminjaman di perpustakaan. Untuk membuat database terlebih skema baru dibuat terlebih dahulu dengan klik kanan pada **Schemas** kemudian pilih **Create New Schema** kemudian beri nama skema sesuai pilihan – contoh **perpustakaan**. Skema yang telah dibuat akan muncul di bawah tab **Schema** seperti Gambar 3 berikut.



Gambar 3. Skema baru yang berhasil dibuat

Untuk membuat tabel klik kanan pada perpustakaan kemudian pilih **SQL Editor** lalu pilih **Open SQL script** kemudian pilih **New Script** sehingga muncul jendela editor untuk menulis kueri SQL seperti pada Gambar 4.



Gambar 4. Tampilan jendela editor

Untuk membuat tabel database kita dapat melakukan secara manual dengan klik kanan pada **Table** kemudian pilih **Create New Table** atau juga dapat menggunakan kueri dengan kode berikut pada jendela editor untuk membuat tabel **Publisher** kemudian klik Execute untuk mengeksekusi kueri**.**

```
CREATE TABLE Publishers (
    publisher_id SERIAL PRIMARY KEY,
    publisher_name VARCHAR(255) NOT NULL,
    contact_email VARCHAR(100),
    contact_phone VARCHAR(15)
);
```

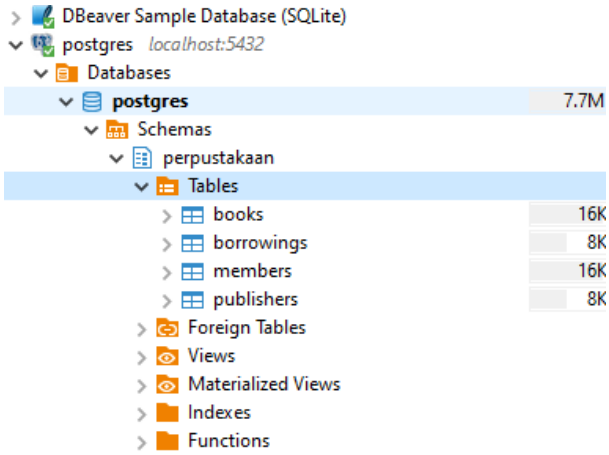
Pada praktikum ini, ada beberapa tabel lagi yang digunakan yaitu Buku, Anggota, dan Peminjaman. Buatlah ketiga tabel tersebut mengikuti langkah-langkah sebelumnya. Kueri dapat dituliskan editor baru atau di bawah kueri sebelumnya dan saat ekekusi, kueri kemudian diblok lalu dieksekusi seperti langkah sebelumnya. Untuk kueri yang digunakan adalah sebagai berikut.

```
CREATE TABLE Books (
    book_id SERIAL PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    author VARCHAR(255),
    publisher_id INT,
    publication_year INT,
    genre VARCHAR(100),
    stock INT DEFAULT 0,
    FOREIGN KEY (publisher_id) REFERENCES Publishers(publisher_id)
);
```

```
CREATE TABLE Members (
    member_id SERIAL PRIMARY KEY,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    email VARCHAR(100) UNIQUE NOT NULL,
    phone_number VARCHAR(15),
    join_date DATE DEFAULT CURRENT_DATE
);
```

```
CREATE TABLE Borrowings (
    borrowing_id SERIAL PRIMARY KEY,
    member_id INT,
    book_id INT,
    borrow_date DATE DEFAULT CURRENT_DATE,
    return_date DATE,
    FOREIGN KEY (member_id) REFERENCES Members(member_id),
    FOREIGN KEY (book_id) REFERENCES Books(book_id)
);
```

Jika tabel-tabel tersebut telah berhasil dibuat, pada tab bagian Table akan muncul nama-nama tabel di bawahnya seperti Gambar 5 berikut.



Gambar 5 Tampilan setelah tabel berhasil di buat

Jika nama-nama tabel yang telah dibuat tidak muncul, klik kanan pada **Schema** atau **Tables** kemudian pilih **Refresh.** Langkah selanjutnya adalah mengisi data pada tabel-tabel tersebut menggunakan kueri pada masing-masing tabel dengan menuliskan kueri seperti pada saat proses pembuatan tabel.

```
INSERT INTO Publishers (publisher_name, contact_email, contact_phone)
VALUES
('Gramedia', 'info@gramedia.com', '0211234567'),
('Erlangga', 'contact@erlangga.co.id', '0212345678'),
('Mizan', 'mizan@mizan.com', '0223456789'),
('HarperCollins', 'info@harpercollins.com', '0214567890'),
('Springer', 'springer@springer.com', '0225678901'),
('Penguin Random House', 'contact@penguinrandomhouse.com', '0216789012'),
('Oxford University Press', 'contact@oup.com', '0217890123'),
('Cambridge University Press', 'info@cambridge.org', '0218901234'),
('Wiley', 'support@wiley.com', '0219012345'),
('McGraw-Hill', 'support@mheducation.com', '0219123456'),
('Pearson', 'contact@pearson.com', '0219234567'),
('Routledge', 'info@routledge.com', '0219345678'),
('Elsevier', 'info@elsevier.com', '0219456789'),
('SAGE Publications', 'contact@sagepub.com', '0219567890'),
('Hachette Livre', 'support@hachette.com', '0219678901'),
('John Wiley & Sons', 'wiley@wiley.com', '0219789012'),
('Blackwell Publishing', 'support@blackwell.com', '0219890123'),
('Harvard University Press', 'contact@harvardpress.com', '0219901234'),
('MIT Press', 'info@mitpress.com', '0219912345'),
('Princeton University Press', 'support@press.princeton.edu', '0219923456'),
('Stanford University Press', 'contact@sup.org', '0219934567'),
('University of Chicago Press', 'info@press.uchicago.edu', '0219945678'),
('Palgrave Macmillan', 'support@palgrave.com', '0219956789'),
('Bloomsbury Publishing', 'contact@bloomsbury.com', '0219967890'),
('Kogan Page', 'info@koganpage.com', '0219978901'),
('Taylor & Francis', 'support@taylorandfrancis.com', '0219989012'),
```

```
('Springer Nature', 'info@springernature.com', '0219990123'),
('University Presses of California', 'support@calpress.edu', '0220001234'),
('Indigo Press', 'info@indigopress.com', '0220012345');
```

```
INSERT INTO Books (title, author, publisher_id, publication_year, genre,
stock)
VALUES
('To Kill a Mockingbird', 'Harper Lee', 1, 1960, 'Fiction', 10),
('1984', 'George Orwell', 2, 1949, 'Dystopian', 15),
('Pride and Prejudice', 'Jane Austen', 3, 1813, 'Romance', 5),
('The Great Gatsby', 'F. Scott Fitzgerald', 4, 1925, 'Fiction', 8),
('Moby Dick', 'Herman Melville', 5, 1851, 'Adventure', 12),
('War and Peace', 'Leo Tolstoy', 6, 1869, 'Historical Fiction', 7),
('Crime and Punishment', 'Fyodor Dostoevsky', 7, 1866, 'Psychological
Fiction', 9),
('The Odyssey', 'Homer', 8, -800, 'Epic', 6),
('The Catcher in the Rye', 'J.D. Salinger', 9, 1951, 'Fiction', 14),
('The Hobbit', 'J.R.R. Tolkien', 10, 1937, 'Fantasy', 20),
('Brave New World', 'Aldous Huxley', 1, 1932, 'Dystopian', 13),
('Frankenstein', 'Mary Shelley', 2, 1818, 'Gothic Fiction', 11),
('Dracula', 'Bram Stoker', 3, 1897, 'Horror', 10),
('Les Misérables', 'Victor Hugo', 4, 1862, 'Historical Fiction', 8),
('The Divine Comedy', 'Dante Alighieri', 5, 1320, 'Epic', 9),
('The Brothers Karamazov', 'Fyodor Dostoevsky', 6, 1880, 'Philosophical
Fiction', 5),
('The Iliad', 'Homer', 7, -750, 'Epic', 6),
('Wuthering Heights', 'Emily Brontë', 8, 1847, 'Gothic Fiction', 12),
('Jane Eyre', 'Charlotte Brontë', 9, 1847, 'Romance', 8),
('The Picture of Dorian Gray', 'Oscar Wilde', 10, 1890, 'Philosophical
Fiction', 7),
('A Tale of Two Cities', 'Charles Dickens', 1, 1859, 'Historical Fiction',
10),
('The Lord of the Rings', 'J.R.R. Tolkien', 2, 1954, 'Fantasy', 25),
('The Chronicles of Narnia', 'C.S. Lewis', 3, 1956, 'Fantasy', 18),
('The Da Vinci Code', 'Dan Brown', 4, 2003, 'Thriller', 30),
('The Shining', 'Stephen King', 5, 1977, 'Horror', 13),
('The Alchemist', 'Paulo Coelho', 6, 1988, 'Adventure', 22),
('Dune', 'Frank Herbert', 7, 1965, 'Science Fiction', 17),
('The Catcher in the Rye', 'J.D. Salinger', 8, 1951, 'Fiction', 19),
('The Hunger Games', 'Suzanne Collins', 9, 2008, 'Dystopian', 24),
('Gone with the Wind', 'Margaret Mitchell', 10, 1936, 'Historical Fiction',
16);
```

```
INSERT INTO Members (first_name, last_name, email, phone_number)
VALUES
('Alice', 'Johnson', 'alice.johnson@example.com', '081234567890'),
('Bob', 'Smith', 'bob.smith@example.com', '081234567891'),
('Charlie', 'Brown', 'charlie.brown@example.com', '081234567892'),
('David', 'Williams', 'david.williams@example.com', '081234567893'),
('Emma', 'Davis', 'emma.davis@example.com', '081234567894'),
('Frank', 'Miller', 'frank.miller@example.com', '081234567895'),
('Grace', 'Wilson', 'grace.wilson@example.com', '081234567896'),
('Hannah', 'Moore', 'hannah.moore@example.com', '081234567897'),
```

```
('Ivy', 'Taylor', 'ivy.taylor@example.com', '081234567898'),
('Jack', 'Anderson', 'jack.anderson@example.com', '081234567899'),
('Kathy', 'Thomas', 'kathy.thomas@example.com', '081234567900'),
('Leo', 'Martinez', 'leo.martinez@example.com', '081234567901'),
('Mona', 'Hernandez', 'mona.hernandez@example.com', '081234567902'),
('Nina', 'Roberts', 'nina.roberts@example.com', '081234567903'),
('Oscar', 'King', 'oscar.king@example.com', '081234567904'),
('Paula', 'Scott', 'paula.scott@example.com', '081234567905'),
('Quinn', 'Adams', 'quinn.adams@example.com', '081234567906'),
('Ryan', 'Baker', 'ryan.baker@example.com', '081234567907'),
('Sara', 'Carter', 'sara.carter@example.com', '081234567908'),
('Tina', 'Gomez', 'tina.gomez@example.com', '081234567909'),
('Uma', 'Evans', 'uma.evans@example.com', '081234567910'),
('Vera', 'Clark', 'vera.clark@example.com', '081234567911'),
('Wendy', 'Lewis', 'wendy.lewis@example.com', '081234567912'),
('Xander', 'Young', 'xander.young@example.com', '081234567913'),
('Yara', 'Walker', 'yara.walker@example.com', '081234567914'),
('Zara', 'Nelson', 'zara.nelson@example.com', '081234567915');
```

```
INSERT INTO Borrowings (member_id, book_id, borrow_date, return_date)
VALUES
(1, 1, '2025-03-01', '2025-03-15'),
(2, 2, '2025-03-02', '2025-03-16'),
(3, 3, '2025-03-03', '2025-03-17'),
(4, 4, '2025-03-04', '2025-03-18'),
(5, 5, '2025-03-05', '2025-03-19'),
(6, 6, '2025-03-06', '2025-03-20'),
(7, 7, '2025-03-07', '2025-03-21'),
(8, 8, '2025-03-08', '2025-03-22'),
(9, 9, '2025-03-09', '2025-03-23'),
(10, 10, '2025-03-10', '2025-03-24'),
(11, 11, '2025-03-11', '2025-03-25'),
(12, 12, '2025-03-12', '2025-03-26'),
(13, 13, '2025-03-13', '2025-03-27'),
(14, 14, '2025-03-14', '2025-03-28'),
(15, 15, '2025-03-15', '2025-03-29'),
(16, 16, '2025-03-16', '2025-03-30'),
(17, 17, '2025-03-17', '2025-03-31'),
(18, 18, '2025-03-18', '2025-04-01'),
(19, 19, '2025-03-19', '2025-04-02'),
(20, 20, '2025-03-20', '2025-04-03'),
(21, 21, '2025-03-21', '2025-04-04'),
(22, 22, '2025-03-22', '2025-04-05'),
(23, 23, '2025-03-23', '2025-04-06'),
(24, 24, '2025-03-24', '2025-04-07'),
(25, 25, '2025-03-25', '2025-04-08'),
(26, 26, '2025-03-26', '2025-04-09')
;
```

Data yang dibuat dapat dilakukan dengan memilih tab **Data** saat nama tabel diklik di bagian kiri. Jika data masih tidak muncul, lakukan **Refresh** pada tiap tabel.

4. Schema Tuning

4.1. Splitting Table

4.1.1. Horizontal Splitting

Horizontal Splitting (*horizontal partitioning*) adalah teknik dalam desain basis data dimana data dalam sebuah tabel dibagi menjadi beberapa bagian berdasarkan baris, bukan kolom. Pada data yang telah dibuat sebelumya terdapat tabel peminjaman dimana terdapat tahun 2025 hingga 2026 dan kita ingin mempartisinya berdasarkan tahun. Berikut adalah kueri untuk membuat partisi tahun 2025.

```
CREATE TABLE Borrowings_2025 (
    borrowing_id SERIAL PRIMARY KEY,
    member_id INT NOT NULL,
    book_id INT NOT NULL,
    borrow_date DATE DEFAULT CURRENT_DATE,
    return_date DATE,
    FOREIGN KEY (member_id) REFERENCES Members(member_id) ON DELETE
CASCADE,
    FOREIGN KEY (book_id) REFERENCES Books(book_id) ON DELETE CASCADE
);
```

```
INSERT INTO Borrowings_2025 (member_id, book_id, borrow_date, return_date)
SELECT member_id, book_id, borrow_date, return_date
FROM Borrowings
WHERE EXTRACT(YEAR FROM borrow_date) = 2025;
```

Untuk partisi pada tahun 2026 lakukan menggunakan kueri sebelumnya, kemudian ganti nama tabel menjadi **Borrowings_2026** dan klausa WHERE diganti menjadi tahun **2026.**

4.1.2. Vertical Splitting

Vertical splitting (*vertical partitioning*) adalah teknik dalam manajemen basis data di mana sebuah tabel dibagi menjadi beberapa tabel yang memiliki subset kolom yang berbeda. Pada praktikum ini, misalkan kita ingin melihat data yang sering diakses, misalkan informasi peminjam buku sehingga kita dapat memisahkan kolom yang sering diakses dengan yang tidak. Berikut adalah kueri yang dapat dilakukan.

```
CREATE TABLE Borrowing_Info (
    borrowing_id SERIAL PRIMARY KEY,
    member_id INT NOT NULL,
    book_id INT NOT NULL,
    FOREIGN KEY (member_id) REFERENCES Members(member_id) ON DELETE
CASCADE,
    FOREIGN KEY (book_id) REFERENCES Books(book_id) ON DELETE CASCADE
);
```

```
CREATE TABLE Borrowing_Dates (
    borrowing_id INT PRIMARY KEY,
    borrow_date DATE DEFAULT CURRENT_DATE,
    return_date DATE,
```

```
    FOREIGN KEY (borrowing_id) REFERENCES Borrowing_Info(borrowing_id) ON
DELETE CASCADE
);
```

Setelah kedua tabel berhasil dibuat, kemudian dilakukan pengisian data pada kedua tabel tersebut. Berikut adalah kueri yang digunakan.

```
INSERT INTO Borrowing_Info (borrowing_id, member_id, book_id)
SELECT borrowing_id, member_id, book_id
FROM Borrowings;
```

```
INSERT INTO Borrowing_Dates (borrowing_id, borrow_date, return_date)
SELECT borrowing_id, borrow_date, return_date
FROM Borrowings;
```

4.2. Denormalisasi

Denormalisasi adalah proses kebalikan dari normalisasi, yaitu menggabungkan tabel-tabel atau menambahkan redundansi data ke dalam database untuk meningkatkan kinerja sistem dalam kondisi tertentu. Denormalisasi dilakukan untuk **mempermudah kueri** atau **meningkatkan kecepatan akses data**, terutama ketika ada kebutuhan untuk mempercepat pembacaan data, meskipun ini bisa menyebabkan **redundansi** (duplikasi data) dan potensi masalah konsistensi.

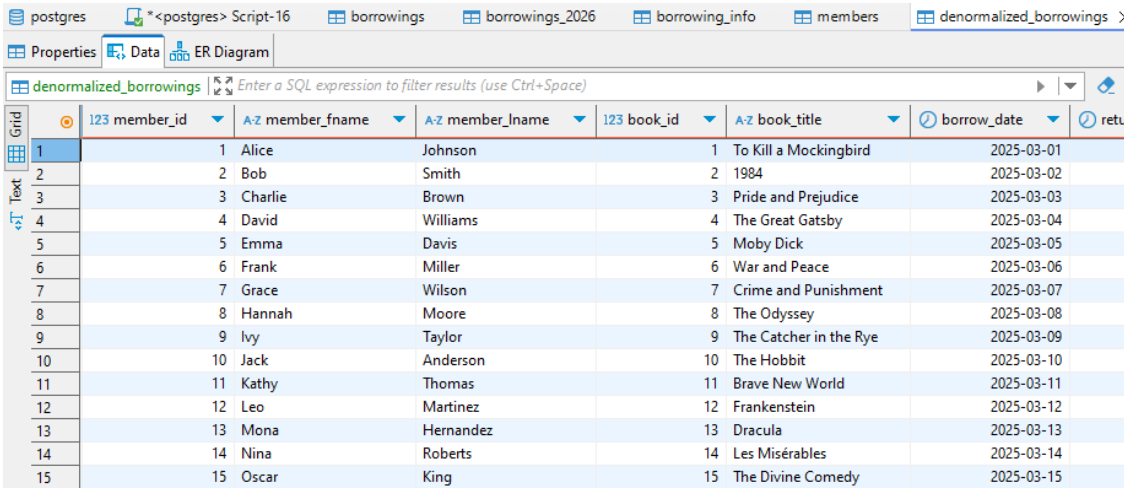
4.2.1. Kolom Redundan (Redundant Column)

Pada praktikum ini misalnya kita sering mengakses data anggota, buku, serta data peminjaman untuk melihat informasi peminjaman. Kita bisa melakukan denormalisasi dengan menggabungkan beberapa kolom pada masing-masing tabel tersebut. Berikut adalah contoh kueri yang digunakan.

```
CREATE TABLE Denormalized_Borrowings AS
SELECT
    b.borrowing_id,
    b.member_id,
    m.first_name AS member_fname,
    m.last_name AS member_lname,
    b.book_id,
    bk.title AS book_title,
    b.borrow_date,
    b.return_date
FROM
    Borrowings b
JOIN
    Members m ON b.member_id = m.member_id
JOIN
    Books bk ON b.book_id = bk.book_id;
```

Pada kueri di atas, denormalisasi dilakukan dengan menggabungkan tabel **Borrowings Members**, dan **Books** menggunakan *key* **member_id**. Hasil kueri tersebut merupakan gabungan ketiga tabel dengan kolom berisi informasi member seperti id, nama, judul

buku, serta tanggal peminjaman dan tanggal pengembalian. Hasil kueri dapat dilihat pada Gambar 6.



postgres | *<postgres> Script-16 | borrowings | borrowings_2026 | borrowing_info | members | denormalized_borrowings

Properties | Data | ER Diagram

denormalized_borrowings | Enter a SQL expression to filter results (use Ctrl+Space)

	member_id	member_fname	member_lname	book_id	book_title	borrow_date	retu
1	1	Alice	Johnson	1	To Kill a Mockingbird	2025-03-01	
2	2	Bob	Smith	2	1984	2025-03-02	
3	3	Charlie	Brown	3	Pride and Prejudice	2025-03-03	
4	4	David	Williams	4	The Great Gatsby	2025-03-04	
5	5	Emma	Davis	5	Moby Dick	2025-03-05	
6	6	Frank	Miller	6	War and Peace	2025-03-06	
7	7	Grace	Wilson	7	Crime and Punishment	2025-03-07	
8	8	Hannah	Moore	8	The Odyssey	2025-03-08	
9	9	Ivy	Taylor	9	The Catcher in the Rye	2025-03-09	
10	10	Jack	Anderson	10	The Hobbit	2025-03-10	
11	11	Kathy	Thomas	11	Brave New World	2025-03-11	
12	12	Leo	Martinez	12	Frankenstein	2025-03-12	
13	13	Mona	Hernandez	13	Dracula	2025-03-13	
14	14	Nina	Roberts	14	Les Misérables	2025-03-14	
15	15	Oscar	King	15	The Divine Comedy	2025-03-15	

Gambar 6 Hasil kueri denormalisasi tiga tabel

4.2.2. Kolom Turunan (Derived Column)

Pada denormalisasi, selain menambahkan kolom redundan dengan menggabungkan beberapa kolom yang berasal dari dua atau lebih tabel, dapat juga dilakukan dengan menambahkan kolom baru yang berasal dari kolom, misalnya menambahkan kolom lama peminjaman yang diturunkan dari kolom tanggal peminjaman dan tanggal pengembalian. Berikut adalah kueri contoh yang bisa dilakukan.

```
CREATE TABLE Loan_Info AS
SELECT
    b.borrowing_id,
    b.member_id,
    m.first_name AS member_fisrtname,
    m.last_name AS member_lastname,
    b.book_id,
    bk.title AS book_title,
    b.borrow_date,
    b.return_date,
    -- Menghitung lama peminjaman dalam satuan hari
    (b.return_date - b.borrow_date) AS loan_duration
FROM
    Borrowings b
JOIN
    Members m ON b.member_id = m.member_id
JOIN
    Books bk ON b.book_id = bk.book_id;
```

Dari kueri di atas, akan dibuat tabel **Loan_Info** yang berisi beberapa kolom yang berasal dari tabel **Borrowings, Members,** dan **Books** serta terdapat kolom tambahan yaitu **loan_duration** yang diturunkan dari kolom **return_date** dan **borrow_date.** Berikut adalah tampilan hasil dari kueri tersebut.

mber_fisrtname	member_lastname	book_id	book_title	borrow_date	return_date	loan_duration
	Johnson	1	To Kill a Mockingbird	2025-03-01	2025-03-15	14
	Smith	2	1984	2025-03-02	2025-03-16	14
	Brown	3	Pride and Prejudice	2025-03-03	2025-03-17	14
	Williams	4	The Great Gatsby	2025-03-04	2025-03-18	14
	Davis	5	Moby Dick	2025-03-05	2025-03-19	14
	Miller	6	War and Peace	2025-03-06	2025-03-20	14
	Wilson	7	Crime and Punishment	2025-03-07	2025-03-21	14
h	Moore	8	The Odyssey	2025-03-08	2025-03-22	14
	Taylor	9	The Catcher in the Rye	2025-03-09	2025-03-23	14
	Anderson	10	The Hobbit	2025-03-10	2025-03-24	14
	Thomas	11	Brave New World	2025-03-11	2025-03-25	14
	Martinez	12	Frankenstein	2025-03-12	2025-03-26	14
	Hernandez	13	Dracula	2025-03-13	2025-03-27	14
	Roberts	14	Les Misérables	2025-03-14	2025-03-28	14
	King	15	The Divine Comedy	2025-03-15	2025-03-29	14
	Scott	16	The Brothers Karamazov	2025-03-16	2025-03-30	14
	Adams	17	The Iliad	2025-03-17	2025-03-31	14

Gambar 7 Tampilan hasil kueri penambahan kolom turunan