

**“Manajemen database perpustakaan  
menggunakan PostgreSQL dan Navicat”**

Mata Kuliah Manajemen Basis Data



***Dosen Pengampu :***

Wiyli Yustanti, S.Si., M.Kom.

**Disusun oleh :**

Kelompok 2

Marsyanda Nur Zahra (22051214119)

Khoirul Ansori (22051214124)

Rajendra Wahyu Aqilla (22051214140)

Sandrak Nababan (22051214141)

**PRODI SISTEM INFORMASI  
FAKULTAS TEKNIK  
UNIVERSITAS NEGERI SURABAYA  
2023**

## **DAFTAR ISI**

- A. Pendahuluan**
- B. Rancangan Basis Data**
  - a. Conceptual Data Model (CDM)**
  - b. Physical Data Model (PDM)**
- C. Rancangan Subquery**
  - a. Subquery buku yang ditulis oleh pengarang tertentu**
  - b. Subquery menampilkan jabatan staf**
  - c. Subquery menghitung jumlah peminjaman buku**
  - d. Subquery banyak anggota meminjam buku**
  - e. Subquery jumlah buku pada rak tertentu**
  - f. Subquery penerbit pada buku dengan judul tertentu**
  - g. Subquery menampilkan pengarang dari judul buku tertentu**
  - h. Subquery menampilkan staf yang melayani transaksi peminjaman**
- D. Rancangan Function ( tujuan, sintak dan output)**
  - a. Function menghitung denda**
  - b. Function mendaftarkan data buku**
  - c. Function menampilkan data buku sesuai pengarang**
  - d. Function menghapus data buku dari daftar buku**
  - e. Function menampilkan data anggota**
  - f. Function menampilkan data buku pada rak tertentu**
  - g. Function menghitung total buku pada rak tertentu**
- E. Rancangan Store Procedure ( tujuan, sintak dan output)**
  - a. Menambahkan data anggota**
  - b. Menambahkan data staf**
  - c. Menghapus data staf**
  - d. Mengupdate data staf**
- F. Rancangan Trigger (tujuan, sintak, output)**

- a. Trigger function anggota
- b. Trigger function buku
- c. Trigger function penerbit
- d. Trigger function pengarang
- e. Trigger function rak

## **G. Kesimpulan**

## **A. PENDAHULUAN**

Dalam upaya memberikan pelayanan yang optimal, perpustakaan memerlukan sistem database yang mampu membantu petugas perpustakaan dalam menyelesaikan tugas - tugasnya. Sistem basis data ini diharapkan dapat memudahkan pengumpulan, pengolahan, penyimpanan dan pengambilan data, sehingga memungkinkan diperolehnya informasi yang akurat dengan kemudahan yang diperlukan. Perancangan sistem informasi ini harus efisien dan efektif agar penggunaannya menjadi lebih praktis dan fleksibel.

Saat ini terdapat banyak sekali sistem informasi berbasis teknologi yang mampu memberikan solusi terhadap permasalahan tersebut dan salah satunya adalah

sistem database PostgreSQL. PostgreSQL adalah sistem database yang kuat dan andal yang telah terbukti mengelola data secara efisien dan akurat. Sebagai alternatif yang ampuh dan fleksibel, PostgreSQL merupakan pilihan yang tepat untuk mengatasi masalah pengelolaan data perpustakaan. PostgreSQL memungkinkan pembuatan sistem yang sesuai dengan kebutuhan dan desain yang dapat disesuaikan dengan preferensi. Fitur seperti tabel, query, dan laporan yang dimiliki PostgreSQL dapat digunakan untuk mengelompokkan buku, mencatat data keanggotaan, serta mencatat peminjaman dan pengembalian perpustakaan.

Selain itu, untuk memudahkan pengelolaan dan pengoperasian database PostgreSQL, laporan ini akan menggunakan perangkat lunak Navicat. Navicat adalah aplikasi manajemen database yang komprehensif dan mudah digunakan yang dapat digunakan untuk mengelola database PostgreSQL dengan mudah. Hal ini memungkinkan pengguna dengan cepat membuat, memodifikasi, dan mengelola tabel, menjalankan kueri, dan menghasilkan laporan yang disajikan dengan menarik.

Dalam konteks ini, laporan ini bertujuan untuk merancang dan mengimplementasikan sistem database perpustakaan menggunakan PostgreSQL dengan bantuan Navicat. Semoga perancangan sistem informasi perpustakaan dengan dukungan Navicat ini dapat menjadi solusi efektif dan meningkatkan kualitas layanan yang diberikan perpustakaan. Oleh karena itu, judul laporan ini adalah “Mengelola database perpustakaan dengan PostgreSQL dan Navicat : Meningkatkan efisiensi dan layanan perpustakaan”.

Dengan menggunakan PostgreSQL dan Navicat, aplikasi ini akan bekerja dengan baik dan memberikan manfaat dalam:

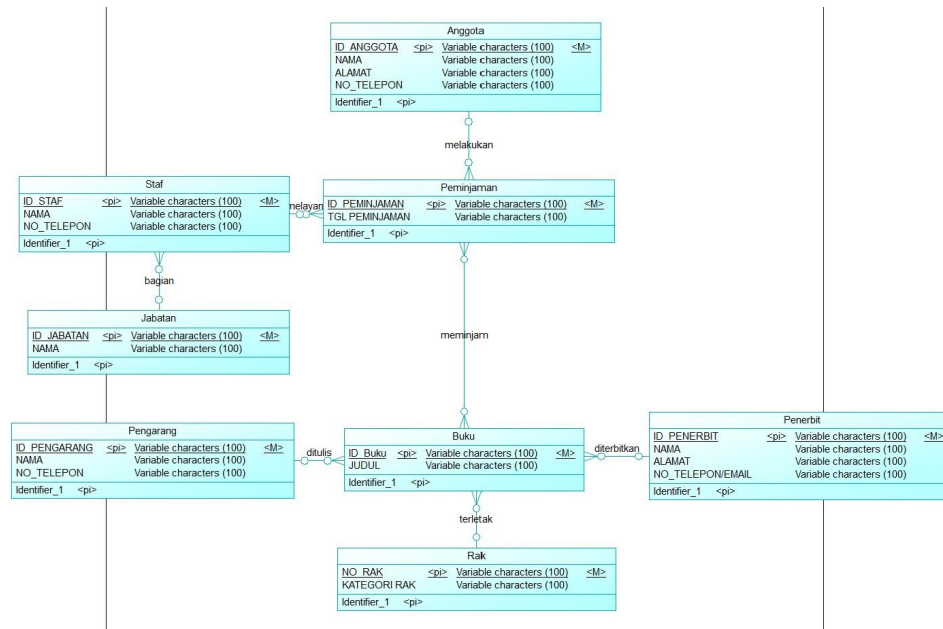
1. Memudahkan akses informasi ketersediaan buku dan laporan tertentu.
2. Meningkatkan efisiensi proses peminjaman dan pengembalian buku.
3. Meningkatkan efisiensi pengelolaan perpustakaan untuk mencapai efisiensi yang lebih besar.
4. Perancangan aplikasi perpustakaan membantu staf perpustakaan dalam mengelola data keanggotaan, buku, dan pengelolaan alur buku dengan lebih mudah dan efektif.
5. Membuat aplikasi yang efisien untuk pengembangan perpustakaan yang lebih baik.

## B. RANCANGAN BASIS DATA PERPUSTAKAAN

### a. CDM

Conceptual Data Model(CDM) merupakan konsep yang berkaitan dengan pandangan pemakai terhadap data yang disimpan dalam basis data. CDM dibuat sudah dalam bentuk tabel-tabel tanpa tipe data yang menggambarkan relasi antar tabel untuk keperluan implementasi ke basis data. CDM merupakan hasil penjabaran lebih lanjut dari ERD.

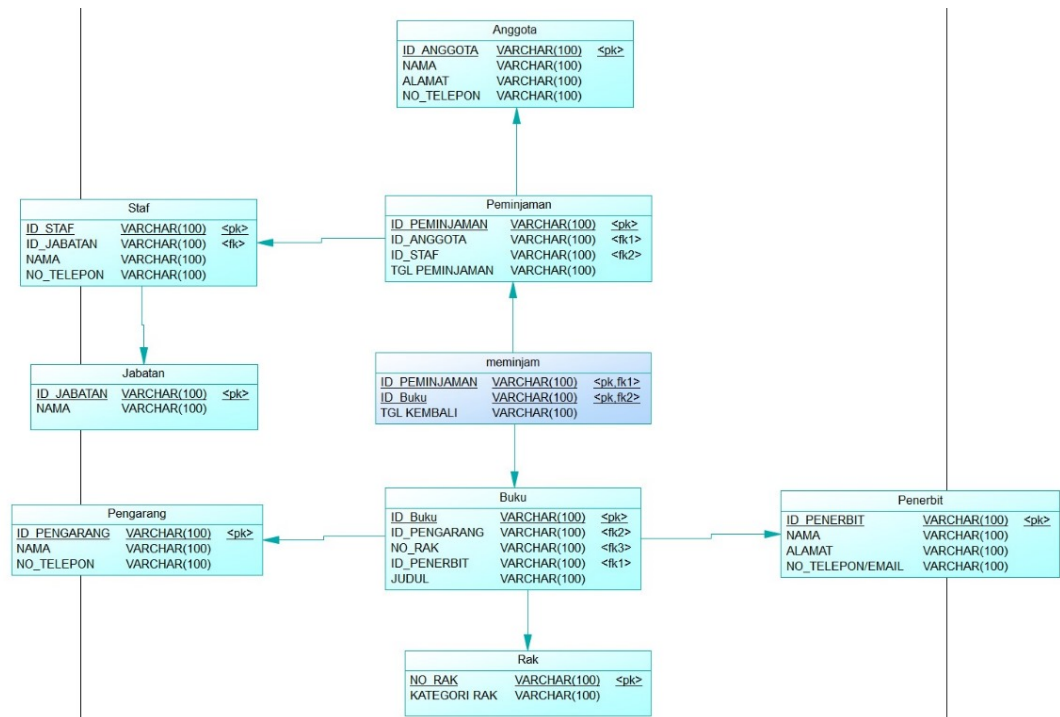
*Berikut CDM Data Base Perpustakaan :*



### b. PDM

Physical Data Model (PDM) adalah model yang menggunakan sejumlah tabel untuk menggambarkan data dan hubungan antar data. Setiap tabel mempunyai sejumlah kolom dimana setiap kolom memiliki nama yang unik beserta tipe datanya. PDM merupakan bentuk desain database fisik yang siap untuk diterapkan dalam DBMS sehingga nama tabel juga merupakan nama row dari tabel yang akan diterapkan dalam DBMS.

*Berikut PDM dari Data Base Perpustakaan :*



### C. RANCANGAN SUBQUERY

#### 1) Menampilkan buku-buku yang ditulis oleh nama pengarang tertentu

*Syntax :*

```

SELECT "BUKU"."ID BUKU", "BUKU"."JUDUL"
FROM "BUKU"
WHERE "BUKU"."ID PENGARANG" = (
    SELECT "ID PENGARANG"
    FROM "PENGARANG"
    WHERE "NAMA" = 'sandrak'
);
  
```

*Output :*

Message	Summary	Result 1
	ID BUKU	JUDUL
▶	8	matematika

#### 2) Menampilkan Jabatan Staf dengan Nama Staf Tertentu

*Syntax :*

```

SELECT "NAMA "
  
```

```
FROM "JABATAN"
WHERE "ID JABATAN" = (
    SELECT "ID JABATAN"
    FROM "STAF"
    WHERE "NAMA" = 'naufal'
);
```

**Output :**

Message	Summary	Result 1
	NAMA	
▶	asisten kepala perpustakaa	

### 3) Menghitung jumlah peminjaman oleh anggota dengan ID tertentu

**Syntax :**

```
SELECT "ID ANGGOTA", "NAMA", (
    SELECT COUNT(*)
    FROM "PEMINJAMAN"
    WHERE "ID ANGGOTA" = "ANGGOTA"."ID ANGGOTA"
) AS "JUMLAH PEMINJAMAN"
FROM "ANGGOTA"
WHERE "ID ANGGOTA" = '1';
```

**Output :**

Message	Summary	Result 1	
	ID ANGGOTA	NAMA	JUMLAH PEMINJAMAN
▶	1	farizlo	2

### 4) Menghitung Jumlah Anggota yang Telah Meminjam Buku

**Syntax :**

```
SELECT COUNT(*)
FROM "ANGGOTA"
WHERE "ID ANGGOTA" IN (
    SELECT DISTINCT "ID ANGGOTA"
```

FROM "PEMINJAMAN"

);

**Output :**

Message	Summary	Result 1
	count	
▶		2

**5) Menghitung Jumlah Buku pada Kategori Rak tertentu**

**Syntax :**

```
SELECT "KATEGORI RAK", (  
    SELECT COUNT(*)  
    FROM "BUKU"  
    WHERE "NO RAK" = "RAK"."NO RAK"  
) AS "JUMLAH BUKU"  
FROM "RAK"  
WHERE "KATEGORI RAK" = 'seni dan budaya';
```

**Output :**

Message	Summary	Result 1	
	KATEGORI RAK	JUMLAH BUKU	
▶	seni dan budaya		4

**6) Menampilkan Nama Penerbit Buku dengan Judul Tertentu**

**Syntax :**

```
SELECT "NAMA"  
FROM "PENERBIT"  
WHERE "ID PENERBIT" = (  
    SELECT "ID PENERBIT"  
    FROM "BUKU"  
    WHERE "JUDUL" = 'melayu'  
);
```

**Output :**



Message	Summary	Result 1
	NAMA	
▶	ddd	

7) Menampilkan nama pengarang dari judul buku tertentu

*Syntax :*

```
SELECT "NAMA"
FROM "PENGARANG"
WHERE "ID PENGARANG" = (
    SELECT "ID PENGARANG"
    FROM "BUKU"
    WHERE "JUDUL" = 'TIK'
);
```

*Output :*

Message	Summary	Result 1
	NAMA	
▶	rajendra	

8) Menampilkan staf yang melayani transaksi peminjaman tertentu

*Syntax :*

```
SELECT "NAMA"
FROM "STAF"
WHERE "ID STAF" = (
    SELECT "ID STAF"
    FROM "PEMINJAMAN"
    WHERE "ID PEMINJAMAN" = '1'
);
```

*Output :*

Message	Summary	Result 1
NAMA		
▶ hans		

#### D. RANCANGAN FUNCTION

- 1) Menghitung denda dari id peminjaman tertentu, rumusnya adalah selisih (tanggal kembali - tanggal kembali realisasi) \* 5000

*Syntax :*

CREATE OR REPLACE FUNCTION

"public"."hitung\_denda"("id\_peminjaman" varchar)

RETURNS TABLE("ID PEMINJAMAN" varchar, "DENDA" numeric) AS

\$BODY\$

DECLARE

denda NUMERIC;

tgl\_kembali DATE;

tgl\_kembali\_realisasi DATE;

BEGIN

SELECT "TGL KEMBALI", "TGL KEMBALI REALISASI" INTO

tgl\_kembali, tgl\_kembali\_realisasi

FROM "MEMINJAM"

WHERE "MEMINJAM"."ID PEMINJAMAN" = id\_peminjaman;

denda := (tgl\_kembali\_realisasi - tgl\_kembali) \* 5000;

IF denda < 0 THEN

denda := 0;

END IF;

RETURN QUERY SELECT id\_peminjaman, denda;

END;

\$BODY\$

LANGUAGE plpgsql VOLATILE

COST 100

ROWS 1000

**Output:**

```
SELECT * FROM hitung_denda('1');
```

Data Output

Messages

Notifications

	<div><div>ID PEMINJAMAN</div><div>character varying</div></div> <div></div>	<div><div>DENDA</div><div>numeric</div></div> <div></div>	
1	1	10000	

**2) Menambahkan daftar buku**

**Syntax :**

```
CREATE OR REPLACE FUNCTION "public"."menambah_buku"("id_buku"
varchar, "judul" varchar, "id_pengarang" varchar, "id_penerbit" varchar,
"no_rak" varchar)
RETURNS "pg_catalog"."void" AS $BODY$
BEGIN
    INSERT INTO "BUKU"("ID BUKU", "JUDUL", "ID PENGARANG",
"ID PENERBIT", "NO RAK")
VALUES (id_buku, judul, id_pengarang, id_penerbit, no_rak);
END;
$BODY$
LANGUAGE plpgsql VOLATILE
COST 100
```

**Output :**

```
SELECT "public"."menambah_buku"('38', 'SERBA SERIBU', '1', '1', '3');
select * from "BUKU";
```

	ID BUKU [PK] character varying (255)	JUDUL character varying (255)	ID PENGARANG character varying (255)	ID PENERBIT character varying (255)	NO RAK character varying (255)
20	33	ererer	3	4	5
21	1	aku apa diaa	2	3	1
22	36	aku seorang superstar	3	4	5
23	37	pasar	1	1	3
24	38	SERBA SERIBU	1	1	3

### 3) Menampilkan data buku sesuai input ID Pengarang tertentu

**Syntax :**

CREATE OR REPLACE FUNCTION

"public"."daftar\_buku\_per\_pengarang"("id\_pengarang" varchar)

RETURNS TABLE("ID BUKU" varchar, "JUDUL" varchar, "ID  
PENGARANG" varchar, "ID PENERBIT" varchar) AS \$BODY\$

BEGIN

RETURN QUERY

SELECT "BUKU"."ID BUKU", "BUKU"."JUDUL", "BUKU"."ID  
PENGARANG", "BUKU"."ID PENERBIT"

FROM "BUKU"

WHERE "BUKU"."ID PENGARANG" = id\_pengarang;

END;

\$BODY\$

LANGUAGE plpgsql VOLATILE

COST 100

ROWS 1000

**Output :**

SELECT \* FROM "public"."daftar\_buku\_per\_pengarang"('4');

	ID BUKU character varying	JUDUL character varying	ID PENGARANG character varying	ID PENERBIT character varying
1	4	jawa	4	3
2	5	batak	4	3
3	6	melayu	4	3
4	7	papua	4	4
5	20	gerakan senam	4	3

Total rows: 5 of 5    Query complete 00:00:00.104

#### 4) Menghapus data buku dari daftar buku berdasarkan ID Buku

**Syntax :**

```
CREATE OR REPLACE FUNCTION "public"."menghapus_buku"("id_buku"
varchar)
```

```
RETURNS "pg_catalog"."void" AS $BODY$
```

```
BEGIN
```

```
DELETE FROM "BUKU" WHERE "ID BUKU" = id_buku;
```

```
END;
```

```
$BODY$
```

```
LANGUAGE plpgsql VOLATILE
```

```
COST 100
```

**Output :**

```
SELECT "public"."menghapus_buku"('3');
```

```
select * from "BUKU";
```

	ID BUKU [PK] character varying (255)	JUDUL character varying (255)	ID PENGARANG character varying (255)	ID PENERBIT character varying (255)	NO RAK character varying (255)
1	2	cyber security	2	2	1
2	4	jawa	4	3	2
3	5	batak	4	3	2
4	6	melayu	4	3	2
5	7	papua	4	4	2

Total rows: 23 of 23    Query complete 00:00:00.152    Ln 3, Col 23

#### 5) Mencari Anggota perpustakaan berdasarkan ID Anggota atau nama Anggota

**Syntax :**

```
CREATE OR REPLACE FUNCTION "public"."cari_anggota"("kriteria"
varchar)
RETURNS TABLE("ID ANGGOTA" varchar, "NAMA" varchar,
"ALAMAT" varchar, "NO TELEPON" varchar) AS $BODY$
BEGIN
RETURN QUERY
SELECT "ANGGOTA"."ID ANGGOTA"::character varying,
"ANGGOTA"."NAMA"::character varying,
"ANGGOTA"."ALAMAT"::character varying, "ANGGOTA"."NO
TELEPON"::character varying
FROM "ANGGOTA"
WHERE "ANGGOTA"."ID ANGGOTA" = kriteria OR
"ANGGOTA"."NAMA" ILIKE '% ' || kriteria || '%';
END;
$BODY$
LANGUAGE plpgsql VOLATILE
COST 100
ROWS 1000
```

**Output :**

```
SELECT * FROM "public"."cari_anggota"('6');
```

	ID ANGGOTA character varying	NAMA character varying	ALAMAT character varying	NO TELEPON character varying
1	6	kalian	madura	08004

**6) Mengambil dan menampilkan data buku pada rak tertentu**

**Syntax:**

```
CREATE OR REPLACE FUNCTION "public"."buku_pada_rak"("no_rak"
varchar)
```




```

RETURNS TABLE("ID BUKU" varchar, "JUDUL" varchar, "ID
PENGARANG" varchar, "ID PENERBIT" varchar) AS $BODY$
BEGIN
    RETURN QUERY
        SELECT b."ID BUKU", b."JUDUL", b."ID PENGARANG", b."ID
PENERBIT"
        FROM "BUKU" AS b
        WHERE b."NO RAK" = no_rak;
END;
$BODY$
LANGUAGE plpgsql VOLATILE
COST 100
ROWS 1000

```

**Output :**

```
SELECT * FROM "public"."buku_pada_rak"('5');
```

	ID BUKU character varying 	JUDUL character varying 	ID PENGARANG character varying 	ID PENERBIT character varying 
1	14	ekonomi pembangunan	7	5
2	15	manajemen perekonomian	8	4
3	16	ekonomi islam	7	3
4	32	REREE	3	4
5	33	ererer	3	4
6	36	aku seorang superstar	3	4
Total rows: 6 of 6		Query complete 00:00:00.120		

**7) Menghitung total buku pada rak tertentu**

**Syntax:**

```

CREATE OR REPLACE FUNCTION
"public"."total_buku_pada_rak"("no_rak" varchar)
RETURNS "pg_catalog"."int4" AS $BODY$
DECLARE
    total_buku integer;
BEGIN

```

```

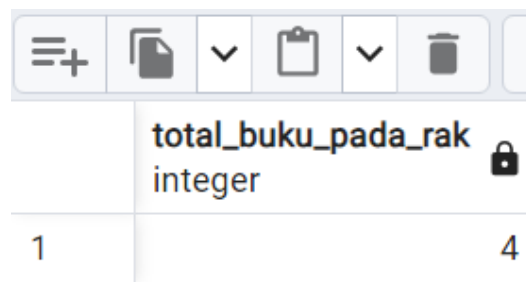
SELECT COUNT(*) INTO total_buku
FROM "BUKU"
WHERE "NO RAK" = no_rak;

RETURN total_buku;
END;
$BODY$
LANGUAGE plpgsql VOLATILE
COST 100

```

**output:**

```
SELECT "public"."total_buku_pada_rak"('2');
```



	total_buku_pada_rak integer
1	4

## E. RANCANGAN STORE PROCEDURE

### 1) Menambah data anggota

**Syntax:**

```

CREATE OR REPLACE PROCEDURE
"public"."add_data_anggota"("id_anggota" varchar, "nama" varchar, "alamat"
varchar, "no_telepon" varchar)
AS $BODY$
BEGIN
    INSERT INTO "ANGGOTA" ("ID ANGGOTA", "NAMA", "ALAMAT",
"NO TELEPON")
    VALUES (id_anggota, nama, alamat, no_telepon);
END;

```



\$BODY\$

LANGUAGE plpgsql

**Output :**

*CALL add\_data\_anggota('99','Sugi','Sidomoro','09876567800');*  
*select \* from "ANGGOTA";*

Data Output Messages Notifications					
	ID ANGGOTA [PK] text	NAMA text	ALAMAT text	NO TELEPON text	
6	7	silaban	surabaya	08003	
7	8	sihombing	surabaya	08002	
8	9	ginting	surabaya	08001	
9	1	farizloo	palembang	080808	
10	10	silalahii	malang	08010	
11	99	Sugi	Sidomoro	09876567800	

**2) Mengupdate data anggota**

**Syntax :**

CREATE OR REPLACE PROCEDURE public.update\_data\_anggota(

    IN id\_anggota character varying,

    IN nama character varying,

    IN alamat character varying,

    IN no\_telepon character varying)

LANGUAGE 'plpgsql'

AS \$BODY\$

BEGIN

    UPDATE "ANGGOTA"

    SET

        "NAMA" = nama,

        "ALAMAT" = alamat,

        "NO TELEPON" = no\_telepon

    WHERE

        "ID ANGGOTA" = id\_anggota;

END;

\$BODY\$;

*Output :*

call update\_data\_anggota('2','Rakan','Gresik','08769876988');

SELECT \* FROM "ANGGOTA";

Data Output   Messages   Notifications				
	ID ANGGOTA [PK] text	NAMA text	ALAMAT text	NO TELEPON text
6	8	sihombing	surabaya	08002
7	9	ginting	surabaya	08001
8	1	farizloo	palembang	080808
9	10	silalahii	malang	08010
10	99	Sugi	Sidomoro	09876567800
11	2	Rakan	Gresik	08769876988

### 3) Menambahkan data staf

*Syntax :*

CREATE OR REPLACE PROCEDURE public.tambah\_staf(

    IN p\_id\_staf character varying,

    IN p\_nama character varying,

    IN p\_no\_telpon character varying,

    IN p\_id\_jabatan character varying)

LANGUAGE 'plpgsql'

AS \$BODY\$

BEGIN

    -- Memasukkan data staf ke dalam tabel STAF

    INSERT INTO "STAF" ("ID STAF", "NAMA", "NO TELEPON", "ID  
JABATAN")

        VALUES (p\_id\_staf, p\_nama, p\_no\_telpon, p\_id\_jabatan);

    -- Commit transaksi

    COMMIT;

```
END;

$BODY$;
```

**Output :**

```
CALL tambah_staf('33','Sumanto','0567898765','1');
Select * from "STAF";
```

Data Output   Messages   Notifications				
	ID STAF [PK] character varying (255)	NAMA character varying (255)	NO TELEPON character varying (255)	ID JABATAN character varying (255)
4	6	vika	08444	5
5	7	tiora	08333	5
6	8	siregar	08222	5
7	2	ardiansyahh	08999	2
8	1	farizzz	08888	1
9	33	Sumanto	0567898765	1

#### 4) Menghapus data staf berdasarkan ID Staf

**Syntax :**

```
CREATE OR REPLACE PROCEDURE public.hapus_staf(
    IN id_staff character varying)
LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
    -- Menghapus data staf dari tabel STAF berdasarkan ID staf
    DELETE FROM "STAF" WHERE "ID STAF" = id_staff;

    -- Commit transaksi
    COMMIT;
END;

$BODY$;
```

**Output :**

```
CALL hapus_staf('33');
Select * from "STAF";
```

	ID STAF [PK] character varying (255)	NAMA character varying (255)	NO TELEPON character varying (255)	ID JABATAN character varying (255)
3	5	hans	08555	5
4	6	vika	08444	5
5	7	tiora	08333	5
6	8	siregar	08222	5
7	2	ardiansyahh	08999	2
8	1	farizzz	08888	1

## 5) Mengupdate data staf

**Syntax :**

```
CREATE OR REPLACE PROCEDURE public.update_staf(
    IN id_staff character varying,
    IN namaa character varying,
    IN no_telpon character varying,
    IN idjabatan character varying)
LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
    -- Mengupdate data staf di dalam tabel STAF berdasarkan ID staf
    UPDATE "STAF"
    SET "NAMA" = namaa, "NO TELEPON" = no_telpon, "ID JABATAN" =
idjabatan
    WHERE "ID STAF" = id_staff;

    -- Commit transaksi
    COMMIT;
END;
$BODY$
```

**Output :**

```
CALL update_staf('1','Suryono','08576789076','5');
select * from "STAF";
```

≡+

▼

▼

	ID STAF [PK] character varying (255)	NAMA character varying (255)	NO TELEPON character varying (255)	ID JABATAN character varying (255)
3	5	hans	08555	5
4	6	vika	08444	5
5	7	tiora	08333	5
6	8	siregar	08222	5
7	2	ardiansyahh	08999	2
8	1	Suryono	08576789076	5

## F. RANCANGAN TRIGGER

### 1) Function ANGGOTA

Fungsi ini mencatat perubahan data dari tabel "ANGGOTA" ke "ANGGOTA\_UPDATE" saat INSERT, UPDATE, atau DELETE terjadi di "ANGGOTA." Data baru/diperbarui dicatat dengan timestamp dan pengguna, sedangkan data yang dihapus dicatat dengan timestamp dan pengguna di "ANGGOTA\_UPDATE."

#### *Sintax :*

```
-- FUNCTION: public.update_anggota()
-- DROP FUNCTION IF EXISTS public.update_anggota();

CREATE OR REPLACE FUNCTION public.update_anggota()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
BEGIN
    IF TG_OP = 'INSERT' THEN
        INSERT INTO public."ANGGOTA_UPDATE" ("ID ANGGOTA", "NAMA",
"ALAMAT", "NO TELEPON", "LAST_UPDATE", "LAST_USER")
        VALUES (NEW."ID ANGGOTA", NEW."NAMA", NEW."ALAMAT",
NEW."NO TELEPON", current_timestamp, 'postgres');
        RETURN NEW;
    ELSIF TG_OP = 'UPDATE' THEN
        INSERT INTO public."ANGGOTA_UPDATE" ("ID ANGGOTA", "NAMA",
"ALAMAT", "NO TELEPON", "LAST_UPDATE", "LAST_USER")
        VALUES (NEW."ID ANGGOTA", NEW."NAMA", NEW."ALAMAT",
NEW."NO TELEPON", current_timestamp, 'postgres');
        RETURN NEW;
    ELSIF TG_OP = 'DELETE' THEN
        INSERT INTO public."ANGGOTA_UPDATE" ("ID ANGGOTA",
"LAST_UPDATE", "LAST_USER")
        VALUES (OLD."ID ANGGOTA", current_timestamp, 'postgres');
        RETURN OLD;
    END IF;
END;
$BODY$;

ALTER FUNCTION public.update_anggota()
    OWNER TO postgres;

-- Trigger: anggota_update_trigger

-- DROP TRIGGER IF EXISTS anggota_update_trigger ON public."ANGGOTA";
CREATE OR REPLACE TRIGGER anggota_update_trigger
    AFTER INSERT OR DELETE OR UPDATE
    ON public."ANGGOTA"
```

```
FOR EACH ROW
EXECUTE FUNCTION public.update_anggota();
```

### Output :

Data Output Messages Notifications						
	ID ANGGOTA [PK] text	NAMA text	ALAMAT text	NO TELEPON text	LAST_UPDATE timestamp without time zone	LAST_USER text
1	1	farizloo	palembang	080808	2023-10-16 12:18:50.479146	postgres
2	10	silalahii	malang	08010	2023-10-16 12:19:25.663033	postgres

## 2) Function BUKU

Fungsi trigger "buku\_change\_trigger" dan tiga trigger yang terhubung mencatat perubahan data di tabel "BUKU" saat ada operasi INSERT, UPDATE, atau DELETE, lalu menyimpan data perubahan tersebut di tabel "BUKU\_UPDATE."

### Sintax:

```
-- FUNCTION: public.buku_change_trigger()
-- DROP FUNCTION IF EXISTS public.buku_change_trigger();

CREATE OR REPLACE FUNCTION public.buku_change_trigger()
  RETURNS trigger
  LANGUAGE 'plpgsql'
  COST 100
  VOLATILE NOT LEAKPROOF
  AS $BODY$
BEGIN
  IF TG_OP = 'INSERT' THEN
    INSERT INTO public."BUKU_UPDATE" ("ID BUKU", "JUDUL", "ID
PENGARANG", "ID PENERBIT", "NO RAK", "LAST_UPDATE",
"USER_UPDATE")
      VALUES (NEW."ID BUKU", NEW."JUDUL", NEW."ID PENGARANG",
NEW."ID PENERBIT", NEW."NO RAK", now(), 'postgres');
  ELSIF TG_OP = 'UPDATE' THEN
    INSERT INTO public."BUKU_UPDATE" ("ID BUKU", "JUDUL", "ID
PENGARANG", "ID PENERBIT", "NO RAK", "LAST_UPDATE",
"USER_UPDATE")
      VALUES (NEW."ID BUKU", NEW."JUDUL", NEW."ID PENGARANG",
NEW."ID PENERBIT", NEW."NO RAK", now(), 'postgres');
  ELSIF TG_OP = 'DELETE' THEN
    INSERT INTO public."BUKU_UPDATE" ("ID BUKU", "JUDUL", "ID
PENGARANG", "ID PENERBIT", "NO RAK", "LAST_UPDATE",
"USER_UPDATE")
```

```
VALUES (OLD."ID BUKU", OLD."JUDUL", OLD."ID PENGARANG",
OLD."ID PENERBIT", OLD."NO RAK", now(), 'postgres');
END IF;
RETURN NEW;
END;
$BODY$;
```

```
ALTER FUNCTION public.buku_change_trigger()
OWNER TO postgres;
```

```
-- Trigger: buku_delete_trigger
```

```
-- DROP TRIGGER IF EXISTS buku_delete_trigger ON public."BUKU";
```

```
CREATE OR REPLACE TRIGGER buku_delete_trigger
AFTER DELETE
ON public."BUKU"
FOR EACH ROW
EXECUTE FUNCTION public.buku_change_trigger();
```

```
-- Trigger: buku_insert_trigger
```

```
-- DROP TRIGGER IF EXISTS buku_insert_trigger ON public."BUKU";
```

```
CREATE OR REPLACE TRIGGER buku_insert_trigger
AFTER INSERT
ON public."BUKU"
FOR EACH ROW
EXECUTE FUNCTION public.buku_change_trigger();
```

```
-- Trigger: buku_update_trigger
```

```
-- DROP TRIGGER IF EXISTS buku_update_trigger ON public."BUKU";
```

```
CREATE OR REPLACE TRIGGER buku_update_trigger
AFTER UPDATE
ON public."BUKU"
FOR EACH ROW
EXECUTE FUNCTION public.buku_change_trigger();
```

**Output :**

Data Output Messages Notifications							
	ID BUKU [PK] character varying (255)	JUDUL character varying (255)	ID PENGARANG character varying (255)	ID PENERBIT character varying (255)	NO RAK character varying (255)	LAST_UPDATE timestamp without time zone	USER_UPDATE text
1	3	TIK	3	2	1	2023-10-16 13:44:59.999831	postgres
2	36	aku seorang superstar	3	4	5	2023-10-16 13:28:55.432019	postgres
3	37	pasar	1	1	3	2023-10-16 13:32:09.290804	postgres
4	38	SERBA SERIBU	1	1	3	2023-10-16 13:33:49.656402	postgres



### 3) Function PENERBIT

Function trigger ini mencatat perubahan data pada tabel "PENERBIT" ke dalam tabel "PENERBIT\_UPDATE" saat terjadi operasi INSERT, UPDATE, atau DELETE. Ini memungkinkan pencatatan perubahan data beserta informasi waktu dan pengguna yang melakukan perubahan untuk keperluan pelacakan dan audit.

#### *Syntax:*

```
-- FUNCTION: public.penerbit_change_trigger_function()
-- DROP FUNCTION IF EXISTS public.penerbit_change_trigger_function();

CREATE OR REPLACE FUNCTION public.penerbit_change_trigger_function()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
BEGIN
    IF TG_OP = 'INSERT' THEN
        INSERT INTO public."PENERBIT_UPDATE" ("ID PENERBIT", "NAMA",
"ALAMAT", "NO TELEPON/EMAIL", "LAST_UPDATE", "USER_UPDATE")
        VALUES (NEW."ID PENERBIT", NEW."NAMA", NEW."ALAMAT", NEW."NO
TELEPON/EMAIL", NOW(), 'postgres');
        RETURN NEW;
    ELSIF TG_OP = 'UPDATE' THEN
        INSERT INTO public."PENERBIT_UPDATE" ("ID PENERBIT", "NAMA",
"ALAMAT", "NO TELEPON/EMAIL", "LAST_UPDATE", "USER_UPDATE")
        VALUES (NEW."ID PENERBIT", NEW."NAMA", NEW."ALAMAT", NEW."NO
TELEPON/EMAIL", NOW(), 'postgres');
        RETURN NEW;
    ELSIF TG_OP = 'DELETE' THEN
        INSERT INTO public."PENERBIT_UPDATE" ("ID PENERBIT", "LAST_UPDATE",
"USER_UPDATE")
        VALUES (OLD."ID PENERBIT", NOW(), 'postgres');
        RETURN OLD;
    END IF;
END;
$BODY$;

ALTER FUNCTION public.penerbit_change_trigger_function()
    OWNER TO postgres;

-- Trigger: penerbit_change_trigger
-- DROP TRIGGER IF EXISTS penerbit_change_trigger ON public."PENERBIT";

CREATE OR REPLACE TRIGGER penerbit_change_trigger
    AFTER INSERT OR DELETE OR UPDATE
    ON public."PENERBIT"
    FOR EACH ROW
    EXECUTE FUNCTION public.penerbit_change_trigger_function();
```

## Output :

Data Output Messages Notifications						
	ID PENERBIT character varying (255)	NAMA character varying (255)	ALAMAT character varying (255)	NO TELEPON/EMAIL character varying (255)	LAST_UPDATE timestamp without time zone	USER_UPDATE character varying (255)
1	1	sss	surabaya	sss@gmail.com	2023-10-16 11:53:28.882568	postgres

### 4) Function PENGARANG

Fungsi trigger ini mencatat perubahan data pengarang, termasuk ID, nama, dan nomor telepon, saat ada operasi INSERT, UPDATE, atau DELETE pada tabel "PENGARANG," untuk keperluan audit dan pemantauan data.

#### Sintax:

```
-- FUNCTION: public.pengarang_audit_function()
-- DROP FUNCTION IF EXISTS public.pengarang_audit_function();

CREATE OR REPLACE FUNCTION public.pengarang_audit_function()
  RETURNS trigger
  LANGUAGE 'plpgsql'
  COST 100
  VOLATILE NOT LEAKPROOF
AS $BODY$
BEGIN
  IF TG_OP = 'INSERT' THEN
    INSERT INTO "PENGARANG_UPDATE" ("ID PENGARANG", "NAMA", "NO TELEPON",
"LAST_UPDATE", "LAST_USER")
      VALUES (NEW."ID PENGARANG", NEW."NAMA", NEW."NO TELEPON", now(), 'postgres');
    RETURN NEW;
  ELSIF TG_OP = 'UPDATE' THEN
    INSERT INTO "PENGARANG_UPDATE" ("ID PENGARANG", "NAMA", "NO TELEPON",
"LAST_UPDATE", "LAST_USER")
      VALUES (NEW."ID PENGARANG", NEW."NAMA", NEW."NO TELEPON", now(), 'postgres');
    RETURN NEW;
  ELSIF TG_OP = 'DELETE' THEN
    INSERT INTO "PENGARANG_UPDATE" ("ID PENGARANG", "NAMA", "NO TELEPON",
"LAST_UPDATE", "LAST_USER")
      VALUES (OLD."ID PENGARANG", OLD."NAMA", OLD."NO TELEPON", now(), 'postgres');
    RETURN OLD;
  END IF;
END;
$BODY$;

ALTER FUNCTION public.pengarang_audit_function()
  OWNER TO postgres;

-- Trigger: pengarang_audit_trigger

-- DROP TRIGGER IF EXISTS pengarang_audit_trigger ON public."PENGARANG";

CREATE OR REPLACE TRIGGER pengarang_audit_trigger
```

```

AFTER INSERT OR DELETE OR UPDATE
ON public."PENGARANG"
FOR EACH ROW
EXECUTE FUNCTION public.pengarang_audit_function();

```

### Output :

Data Output Messages Notifications					
	ID PENGARANG [PK] character varying (255)	NAMA character varying (255)	NO TELEPON character varying (255)	LAST_UPDATE timestamp without time zone	LAST_USER text
1	1	sandrak	082111	2023-10-16 10:39:05.955508	postgres

## 5) Function RAK

### • Tujuan :

Fungsi trigger ini digunakan untuk mencatat perubahan data pada tabel "RAK" ke dalam tabel "RAK\_UPDATE" saat operasi INSERT, UPDATE, atau DELETE terjadi pada "RAK." Tujuannya adalah untuk melacak dan mencatat perubahan data beserta informasi waktu dan pengguna yang melakukan perubahan.

### • Sintak :

```

-- FUNCTION: public.update_rak_trigger_function()

-- DROP FUNCTION IF EXISTS public.update_rak_trigger_function();

CREATE OR REPLACE FUNCTION public.update_rak_trigger_function()
  RETURNS trigger
  LANGUAGE 'plpgsql'
  COST 100
  VOLATILE NOT LEAKPROOF
  AS $BODY$
BEGIN
  IF TG_OP = 'INSERT' THEN
    INSERT INTO public."RAK_UPDATE" ("NO RAK", "KATEGORI RAK",
"LAST_UPDATE", "USER_UPDATE")
      VALUES (NEW."NO RAK", NEW."KATEGORI RAK", NOW(), 'postgres');
  ELSIF TG_OP = 'UPDATE' THEN
    INSERT INTO public."RAK_UPDATE" ("NO RAK", "KATEGORI RAK",
"LAST_UPDATE", "USER_UPDATE")
      VALUES (NEW."NO RAK", NEW."KATEGORI RAK", NOW(), 'postgres');
  ELSIF TG_OP = 'DELETE' THEN
    INSERT INTO public."RAK_UPDATE" ("NO RAK", "KATEGORI RAK",
"LAST_UPDATE", "USER_UPDATE")
      VALUES (OLD."NO RAK", OLD."KATEGORI RAK", NOW(), 'postgres');
  END IF;
  RETURN NULL;
END;
$BODY$;

```

```
ALTER FUNCTION public.update_rak_trigger_function()  
  OWNER TO postgres;
```

```
-- Trigger: rak_update_trigger
```

```
-- DROP TRIGGER IF EXISTS rak_update_trigger ON public."RAK";
```

```
CREATE OR REPLACE TRIGGER rak_update_trigger  
  AFTER INSERT OR DELETE OR UPDATE  
  ON public."RAK"  
  FOR EACH ROW  
  EXECUTE FUNCTION public.update_rak_trigger_function();
```

- **Output :**

Data Output   Messages   Notifications				
	NO RAK character varying (255) 🔒	KATEGORI RAK character varying (255) 🔒	LAST_UPDATE timestamp without time zone 🔒	USER_UPDATE character varying (255) 🔒
1	1	informasi teknologi	2023-10-16 11:52:28.714288	postgres

## **KESIMPULAN**

Laporan ini membahas penggunaan sistem database PostgreSQL dan perangkat lunak Navicat untuk mengelola data perpustakaan. Perpustakaan memerlukan sistem informasi yang efisien dan andal untuk mengumpulkan, memproses, menyimpan, dan mengakses data secara akurat.

PostgreSQL memberikan fleksibilitas dalam merancang dan mengelola database perpustakaan, memungkinkan pengelolaan berbagai jenis data dan pembuatan laporan yang kaya informasi. Perangkat lunak Navicat menyediakan antarmuka yang ramah pengguna untuk mengelola database dengan cepat dan efisien.

Tujuan dari laporan ini adalah untuk meningkatkan efisiensi dan layanan perpustakaan melalui penggunaan PostgreSQL dan Navicat. Dengan sistem database yang sesuai, informasi ketersediaan buku, peminjaman, pengembalian dan data keanggotaan dapat diakses dengan mudah, sehingga perpustakaan dapat meningkatkan efisiensi administrasi dan pemberian layanan yang lebih baik.

Oleh karena itu, penerapan “Manajemen database perpustakaan menggunakan PostgreSQL dan Navicat” merupakan langkah positif dalam meningkatkan efisiensi dan layanan perpustakaan, mendukung kebutuhan pengguna, dan mencapai tujuan kemajuan umum perpustakaan.

## **DAFTAR PUSTAKA**

*<https://medium.com/@muhammadnizamuddin.19072/planning-cdm-and-pdm-71bdd3f265c8>*