**DBMS Project Report**

**PES University**

**Database Management Systems**

**UE18CS252**

Submitted By

|  |  |  |
| --- | --- | --- |
| **SRN :**  **PES1201800389** | **Name :**    **N Sanketh Reddy** | **Evaluation date and time:**  **29 - 5 - 2020** |

Functional Dependencies

[2](#_tyjcwt)

Identifying Keys based on FDs [2](#_tyjcwt)

[Normalization](#_tyjcwt) & testing for lossless join property 2+[2](#_tyjcwt)

[DDL](https://docs.google.com/document/d/1gcMs7iQNpn-eMEc0NX79AJ4yqihNphj8iBxuiP4T9XM/edit#heading=h.3dy6vkm): Table creation with all constraints 2+[2](https://docs.google.com/document/d/1gcMs7iQNpn-eMEc0NX79AJ4yqihNphj8iBxuiP4T9XM/edit#heading=h.3dy6vkm)

[Triggers](https://docs.google.com/document/d/1gcMs7iQNpn-eMEc0NX79AJ4yqihNphj8iBxuiP4T9XM/edit#heading=h.1t3h5sf) [2](https://docs.google.com/document/d/1gcMs7iQNpn-eMEc0NX79AJ4yqihNphj8iBxuiP4T9XM/edit#heading=h.1t3h5sf)

[SQL Queries](https://docs.google.com/document/d/1gcMs7iQNpn-eMEc0NX79AJ4yqihNphj8iBxuiP4T9XM/edit#heading=h.4d34og8) 2

Viva / modifications(Unit III/ IV concepts)2+2

***Criminal Database management Systems***

This is a project which helps in visualizing the police database and also has a python front-end for appending the data into the database and updating the data in the database.

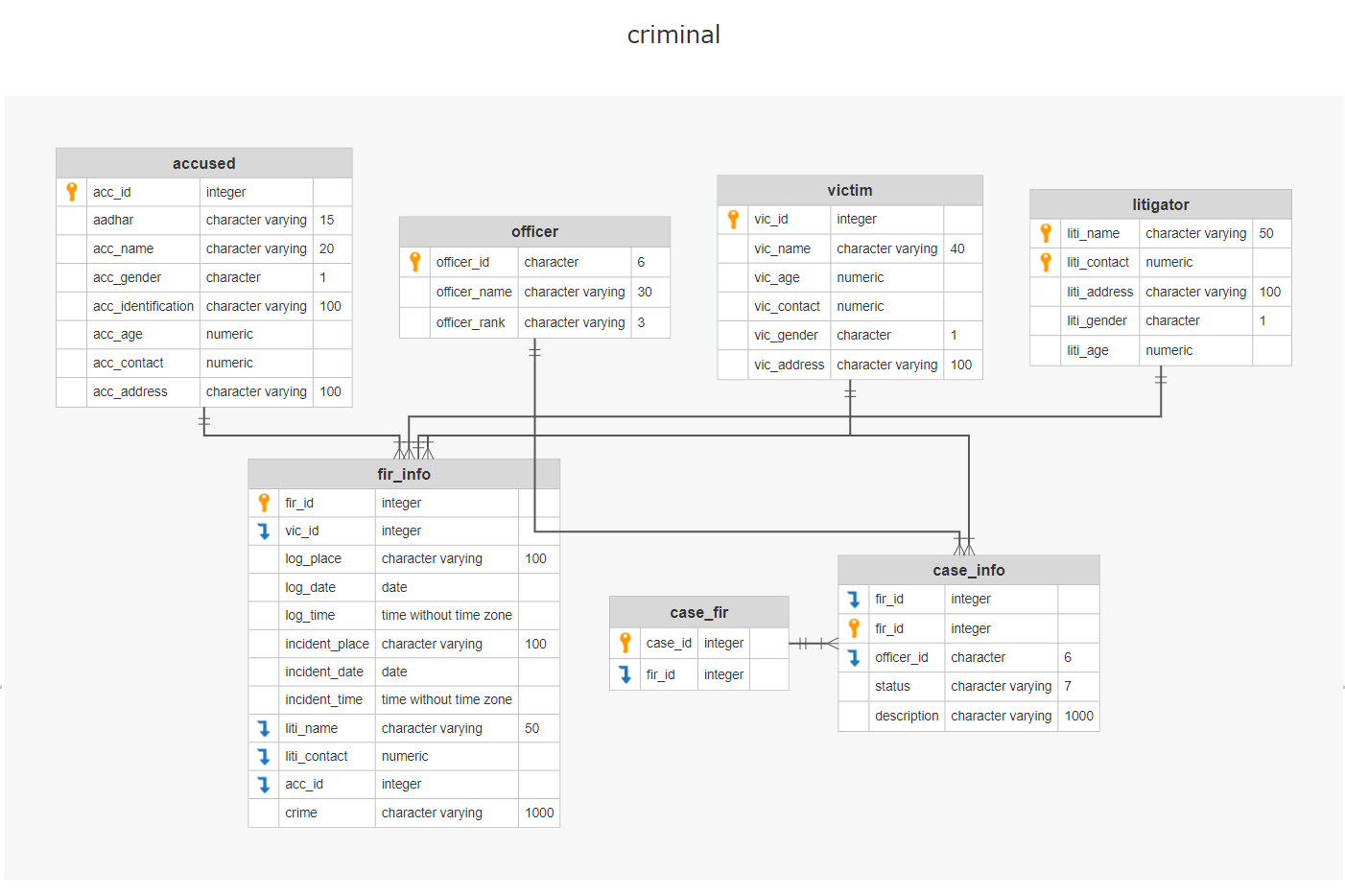
The database has 6 main tables :

* Victim table: Stores the information of the victims.
* Accused table: Stores the information of the accused.
* Officer table: Stores the information of the police officers.
* Litigator table: Stores the information related to the Person filing the FIR.
* FIR\_info: Stores the information of the FIRs
* Case\_info: Stores the information related to the case.

After normalizing we get 7 tables because we had to split the case\_info table into case\_info and case\_fir to satisfy the normalizing conditions.

The final list of tables:

* Victim
* Accused
* Officer
* Litigator
* Fir\_info
* Case\_info
* Case\_fir

**Final database schema.**

**Normalization**:

* **1NF** : Removes repeating groups from the table. Create a seperate table for each set of related data. Identify each set of related data with a primary key.

We had no repeating values so the INF is satisfied.

* **2NF** : The tables in the database should not contain partial dependencies.(In case your primary key is single-valued, you can ignore this NF. In case it is multi-valued, Each non prime attribute has to depend on every value.) .

The litigator table has a composite primary key and every other value in the table fully depends on both the attributes that make up the primary key, so the 2NF is satisfied.

* **3NF** : There should be no transitive dependency for non-prime attributes.

The case\_info table had a transitive relation:

Case\_id -> fir\_id

Fir\_id -> all the other attributes of case\_info

Case\_id -> all the other attributes of case\_info

So we split the table into 2 tables i.e case\_fir and case\_info.So the 3NF is satisfied.

* **3.5NF(BCNF)** : Every functional dependency A->B ,then A must be the Super Key of that particular table.

The 3.5NF is already satisfied.

* **4NF** : The given relation may not contain more than one multi-valued attribute. The multi-valued dependency X→Y holds in a relation R if whenever we have two tuples of R that are same in all the attributes of X, then we can swap their Y components and get two new tuples that are also in R.

We have no multivalued attributes, so 4NF is already satisfied.

* **5NF** : 5NF is satisfied when all tables are broken into as many tables as possible in order to avoid redundancy. Once it is in fifth normal form it cannot be broken into smaller relations without changing the facts or the meaning.

5NF is already satisfied since it has passed all the previous normal forms.

## **Functional dependencies:**

* **case\_info** : fir\_id -> {officer\_id, status, description}
* **case\_fir** : case\_id -> fir\_id
* **fir\_info** : fir\_id -> { vic\_id,log\_place, log\_date, log\_time, incident\_place, incident\_date, incident\_time, liti\_name, liti\_contact, acc\_id, crime}
* **accused** : acc\_id -> {aadhar, acc\_name, acc\_gender, acc\_identification, acc\_age, acc\_contact, acc\_address}
* **litigator** : {liti\_name , liti\_contact} -> {liti\_address, liti\_gender, liti\_age }
* **officer** : officer\_id -> {officer\_name , officer\_rank}
* **victim** : vic\_id ->{vic\_name, vic\_age, vic\_contact, vic\_gender, vic\_adress}

**Checks and Constraints:**

* **officer**
  + CHECK(officer\_id LIKE 'OF\_\_\_\_')
  + PRIMARY KEY (officer\_id)
* **litigator**
  + CHECK (liti\_gender in ('M','F','m','f'))
  + CONSTRAINT litigator\_pkey PRIMARY KEY (liti\_name ,liti\_contact)
* **victim**
  + vic\_id integer NOT null
  + CHECK (vic\_gender in ('M','F','m','f'))
  + CONSTRAINT victim\_pkey PRIMARY KEY (v\_id)
* **accused**
  + acc\_id serial not null
  + check(aadhar ~ '^[0-9]\*$')
  + CHECK (acc\_gender in ('M','F','m','f'))
  + CONSTRAINT accused\_pkey PRIMARY KEY (acc\_id)
* **fir\_info**
  + fir\_id SERIAL PRIMARY KEY
  + log\_date date DEFAULT current\_date
  + log\_time time default current\_time
  + constraint fir\_acc\_fk FOREIGN KEY(acc\_id) references accused(acc\_id)
  + constraint fir\_liti\_fk FOREIGN KEY(liti\_name,liti\_contact) references litigator(liti\_name,liti\_contact)
  + constraint fir\_vic\_fk FOREIGN KEY(vic\_id) references victim(vic\_id) )
* **case\_info**
  + CHECK(status in ('OPEN','CLOSED'))
  + CONSTRAINT cd\_fk1 FOREIGN KEY(fir\_id) references fir\_info(fir\_id)
  + CONSTRAINT cd\_fk2 FOREIGN KEY(officer\_id) references officer(officer\_id) )
* **case\_fir**
  + case\_id SERIAL PRIMARY KEY
  + constraint case\_fir\_fk FOREIGN KEY(fir\_id) references fir\_vicid(fir\_id))

**DDL and Triggers:**

**Officer :**

CREATE TABLE public.officer (

officer\_id bpchar(6) NOT NULL,

officer\_name varchar(30) NULL,

officer\_rank varchar(3) NULL,

CONSTRAINT officer\_officer\_id\_check CHECK ((officer\_id ~~ 'OF\_\_\_\_'::text)),

CONSTRAINT officer\_pk PRIMARY KEY (officer\_id)

);

**Victim :**

-- Drop table

-- DROP TABLE public.victim;

CREATE TABLE public.victim (

vic\_id serial NOT NULL,

vic\_name varchar(40) NULL,

vic\_age numeric(2) NULL,

vic\_contact numeric(10) NULL,

vic\_gender bpchar(1) NULL,

vic\_address varchar(100) NULL,

CONSTRAINT victim\_pkey PRIMARY KEY (vic\_id),

CONSTRAINT victim\_vic\_gender\_check CHECK ((vic\_gender = ANY (ARRAY['M'::bpchar, 'F'::bpchar, 'm'::bpchar, 'f'::bpchar])))

);

-- Table Triggers

-- DROP TRIGGER vic\_triggger ON public.victim;

create trigger vic\_triggger before

insert

or

update

on

public.victim for each row execute function vic\_trig();

**Accused :**

-- Drop table

-- DROP TABLE public.accused;

CREATE TABLE public.accused (

acc\_id serial NOT NULL,

aadhar varchar(15) NULL,

acc\_name varchar(20) NULL,

acc\_gender bpchar(1) NULL,

acc\_identification varchar(100) NULL,

acc\_age numeric(2) NULL,

acc\_contact numeric(10) NULL,

acc\_address varchar(100) NULL,

CONSTRAINT accused\_aadhar\_check CHECK (((aadhar)::text ~ '^[0-9]\*$'::text)),

CONSTRAINT accused\_acc\_gender\_check CHECK ((acc\_gender = ANY (ARRAY['M'::bpchar, 'F'::bpchar, 'm'::bpchar, 'f'::bpchar]))),

CONSTRAINT accused\_pkey PRIMARY KEY (acc\_id)

);

-- Table Triggers

-- DROP TRIGGER acc\_trigger ON public.accused;

create trigger acc\_trigger before

insert

or

update

on

public.accused for each row execute function acc\_trig();

**Litigator:**

-- Drop table

-- DROP TABLE public.litigator;

CREATE TABLE public.litigator (

liti\_name varchar(50) NOT NULL,

liti\_contact numeric(10) NOT NULL,

liti\_address varchar(100) NULL,

liti\_gender bpchar(1) NULL,

liti\_age numeric(2) NULL,

CONSTRAINT litigator\_liti\_gender\_check CHECK ((liti\_gender = ANY (ARRAY['M'::bpchar, 'F'::bpchar, 'm'::bpchar, 'f'::bpchar]))),

CONSTRAINT litigator\_pkey PRIMARY KEY (liti\_name, liti\_contact)

);

**fir\_info:**

-- Drop table

-- DROP TABLE public.fir\_info;

CREATE TABLE public.fir\_info (

fir\_id serial NOT NULL,

vic\_id int4 NULL,

log\_place varchar(100) NULL,

log\_date date NULL DEFAULT CURRENT\_DATE,

log\_time time NULL DEFAULT CURRENT\_TIME,

incident\_place varchar(100) NULL,

incident\_date date NULL,

incident\_time time NULL,

liti\_name varchar(50) NULL,

liti\_contact numeric(10) NULL,

acc\_id int4 NULL,

crime varchar(1000) NULL,

CONSTRAINT fir\_info\_pkey PRIMARY KEY (fir\_id),

CONSTRAINT fir\_acc\_fk FOREIGN KEY (acc\_id) REFERENCES accused(acc\_id),

CONSTRAINT fir\_liti\_fk FOREIGN KEY (liti\_name, liti\_contact) REFERENCES litigator(liti\_name, liti\_contact),

CONSTRAINT fir\_vic\_fk FOREIGN KEY (vic\_id) REFERENCES victim(vic\_id) ON DELETE CASCADE

);

**Case\_fir:**

-- Drop table

-- DROP TABLE public.case\_fir;

CREATE TABLE public.case\_fir (

case\_id serial NOT NULL,

fir\_id int4 NULL,

CONSTRAINT case\_fir\_pkey PRIMARY KEY (case\_id),

CONSTRAINT case\_fir\_fk FOREIGN KEY (fir\_id) REFERENCES case\_info(fir\_id)

);

**case\_info:**

-- Drop table

-- DROP TABLE public.case\_info;

CREATE TABLE public.case\_info (

fir\_id int4 NOT NULL,

officer\_id bpchar(6) NULL,

status varchar(7) NULL,

description varchar(1000) NULL,

CONSTRAINT case\_info\_pkey PRIMARY KEY (fir\_id),

CONSTRAINT case\_info\_status\_check CHECK (((status)::text = ANY ((ARRAY['OPEN'::character varying, 'CLOSED'::character varying])::text[]))),

CONSTRAINT cd\_fk1 FOREIGN KEY (fir\_id) REFERENCES fir\_info(fir\_id),

CONSTRAINT cd\_fk2 FOREIGN KEY (officer\_id) REFERENCES officer(officer\_id)

);

**Functions for triggers:**

* **vic\_trig() -> To avoid duplicate rows in Victim table**

CREATE OR REPLACE FUNCTION public.vic\_trig()

RETURNS trigger

LANGUAGE plpgsql

AS $function$

DECLARE cnt\_vic integer;

BEGIN

IF EXISTS (

select table\_name from information\_schema.tables where table\_name='vic\_temp')

THEN

DROP TABLE vic\_temp;

END IF;

CREATE TEMP TABLE vic\_temp AS

**SELECT vic\_id from victim WHERE (NEW.vic\_name = victim.vic\_name AND NEW.vic\_contact=victim.vic\_contact) ;**

SELECT count(\*) INTO cnt\_vic from vic\_temp;

if cnt\_vic>=1

THEN

RAISE EXCEPTION 'Victim details already exists';

RETURN NULL;

END IF;

IF cnt\_vic=0

THEN

RETURN NEW;

END IF;

RETURN NULL;

END;

$function$

;

* **fir\_trig() -> To avoid duplicate FIRs**

CREATE OR REPLACE FUNCTION public.fir\_trig()

RETURNS trigger

LANGUAGE plpgsql

AS $function$

DECLARE cnt\_fir integer;

BEGIN

IF EXISTS (

select table\_name from information\_schema.tables where table\_name='fir\_temp')

THEN

DROP TABLE fir\_temp;

END IF;

CREATE TEMP TABLE FIR\_table AS

**SELECT fir\_id from fir\_info WHERE (NEW.crime = fir\_info.crime AND NEW.vic\_id=fir\_info.vic\_id AND NEW.incident\_place=fir\_info.incident\_place AND NEW.incident\_date=fir\_info.incident\_date ) ;**

SELECT count(\*) INTO cnt\_fir from fir\_temp;

if cnt\_fir>=1

THEN

RAISE EXCEPTION 'FIR already filed on this crime';

RETURN NULL;

END IF;

IF cnt\_fir=0

THEN

RETURN NEW;

END IF;

RETURN NULL;

END;

$function$

;

* **acc\_trig() -> To avoid duplicate Accused details**

CREATE OR REPLACE FUNCTION public.acc\_trig()

RETURNS trigger

LANGUAGE plpgsql

AS $function$

declare cnt integer;

BEGIN

if exists(

select table\_name from information\_schema.tables where table\_name = 'acc\_temp')

then

drop table acc\_temp;

end if;

create temp table acc\_temp as

**select acc\_id from accused where new.aadhar = accused.aadhar;**

SELECT count(\*) INTO cnt from acc\_temp;

if cnt>=1

THEN

RAISE EXCEPTION 'Accused details already exists';

RETURN NULL;

END IF;

IF cnt=0

THEN

RETURN NEW;

END IF;

RETURN NULL;

END;

$function$

;

**Complex SQL Queries:**

* **Retrive the officer's name who is incharge of murder cases**

*select officer\_name from officer where officer\_id in (select officer\_id from case\_info where fir\_id in (select fir\_id from fir\_info where (crime = 'MURDER') or (crime = 'murder')))*

* **Retrieve accused names who were accused by litigators who are men and are of aged above 20 years.**

*select acc\_name from accused where acc\_id in (select acc\_id from fir\_info where liti\_name in (select liti\_name from litigator where liti\_gender = 'M' and liti\_age > 20 ) )*

* **Retrieve the accused name whose age is above average accuRetrivesed age.**

*select acc\_name from accused where acc\_age > (select avg(acc\_age) from accused)*

* **Retrieve the count of male and female victims**

*select vic\_gender, count( vic\_gender) from victim group by vic\_gender*

* **Retrieve the the aadhar number of the accused for the case\_id = 8**

*select aadhar from (case\_fir CF join fir\_info F on CF.fir\_id=F.fir\_id join accused AC on AC.acc\_id = F.acc\_id )*

*where case\_id=8*

* **Retrieve the minimum age of the accused , based on the gender, who have been included in the incidents after the year 2000**

*select acc\_gender, min(acc\_age) from accused where acc\_id in (select acc\_id from fir\_info where incident\_date > '2000-1-1') and acc\_age > 0 group by acc\_gender*

**Python Frontend:**

**insert.py/ipynb :**

This python script helps in filing the fir the litigator needs to enter needs to enter some details about himself and the incident the rest of the fields are automatically filled.

And if all the fir details are typed in correctly ,the program continues and creates a case and allocates a police officer to deal with the case.

The program also creates an empty row in the accused and victim table related to the fir.

**update.py/ipynb :**

This python script helps in updating the victim and accused information which was earlier created ,if the victim or the accused already exists in the database its not gonna create a new value instead it is going to delete the new empty value that was created during the execution of insert.py and update the vic\_id or acc\_id of the already existing person in the case and fir information tables.