CSE414

DATABASES

PROJECT REPORT

ALP EMİR BİLEK

161044049

Introduction

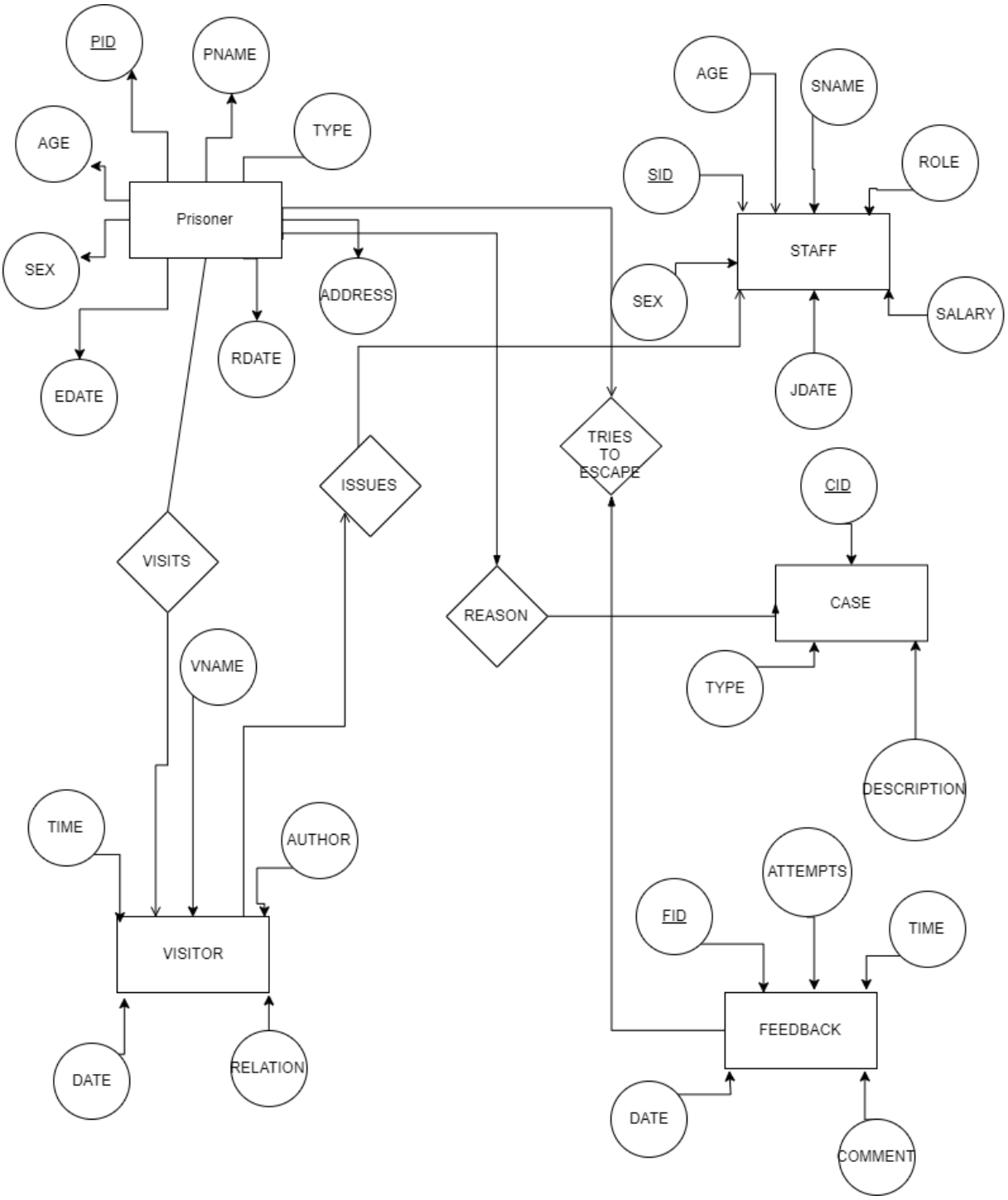
In this project, I aimed to program a prison management system. I designed the database and frontend with the help of MySQL and php. 17 tables were created, including 12 databases and 5 view tables.

Functions have been added where you can view prisoners, employees and visitors and update their information.

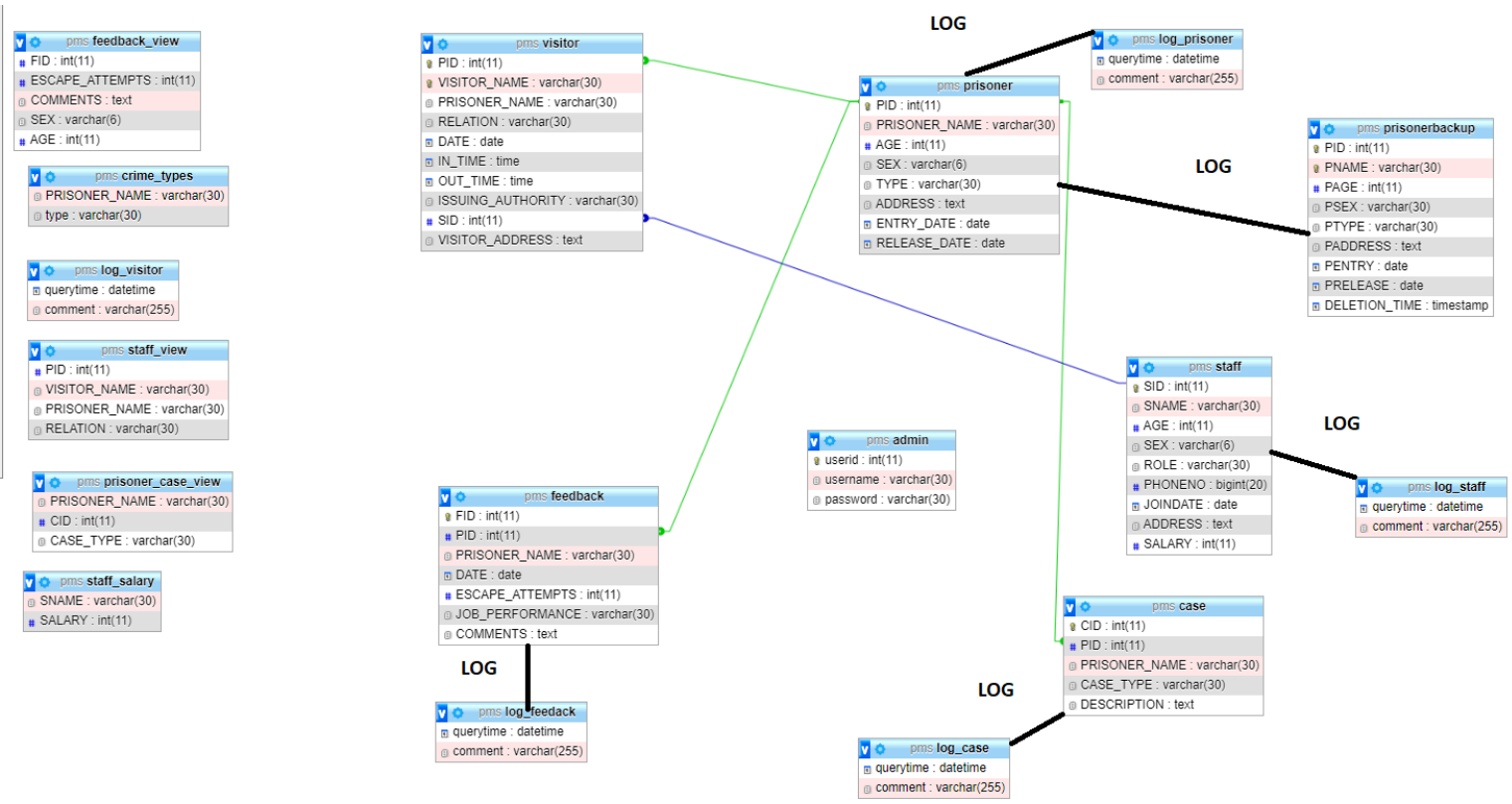
User Requirements

- User can add/delete prisoner.
- User can add/delete visitor.
- User can add admin.
- The user can view/update the information of the prisoners.
- User can add/delete staff.
- User can view/update staffs information.
- Staffs take care of prisoners' visitors.
- User can view/update prisoners' crimes.
- User can view/update visitors' information.
- Staffs can give feedback on prisoners.
- User can view/update these feedbacks.
- The user can view the reason for entering the prisoners along with their names in a view.
- The user can view the joined information of the feedbacks through a view.

E-R DIAGRAM


PID
PNAME
AGE
TYPE
Prisoner
SEX
ADDRESS
RDATE
EDATE
AGE
SNAME
SID
ROLE
STAFF
SEX
SALARY
JDATE
TRIES TO ESCAPE
ISSUES
CID
VISITS
REASON
CASE
VNAME
TYPE
DESCRIPTION
TIME
AUTHOR
ATTEMPTS
FID
TIME
VISITOR
FEEDBACK
DATE
RELATION
DATE
COMMENT

RELATIONAL SCHEMA



NORMALIZATION

Database normalization is a process used to organize a database into tables and columns. There are three main forms: first normal form , second normal form, and third normal form. The main idea is each table should be about a specific topic and only supporting topics included. Take a spreadsheet containing the information as an example, where the data contains salespeople and customers serving several purposes:

- Identify salespeople in your organization
- List all customers your company calls upon to sell a product
- Identify which salespeople call on specific customers.

By limiting a table to one purpose you reduce the number of duplicate data contained within your database. This eliminates some issues stemming from database modifications.

TABLES AND FUNCTIONAL DEPENDENCIES

Admin

Adı	Türü	Karşılaştırma	Öznitelikler	Boş	Varsayılan	Açıklamalar	Ekstra
userid	int(11)			Hayır	*Yok*		AUTO_INCREMENT
username	varchar(30)	latin1_swedish_ci		Hayır	*Yok*		
password	varchar(30)	latin1_swedish_ci		Hayır	*Yok*		

This table holds the data for login to the system. The table has 3 columns. These columns consist of the admin's username and password.

Primary Key = userid

Functional Dependencies;

Userid → username

Userid → password

Case

Adı	Türü	Karşılaştırma	Öznitelikler	Boş	Varsayılan	Açıklamalar	Ekstra
CID	int(11)			Hayır	*Yok*		AUTO_INCREMENT
PID	int(11)			Hayır	*Yok*		
PRISONER_NAME	varchar(30)	latin1_swedish_ci		Hayır	*Yok*		
CASE_TYPE	varchar(30)	latin1_swedish_ci		Hayır	*Yok*		
DESCRIPTION	text	latin1_swedish_ci		Hayır	*Yok*		

This table is there to keep the crimes of the prisoners. There are 5 columns in the table.

Primary Key = CID

Functional Dependencies;

CID → PRISONER_NAME

CID → CASE_TYPE

CID → DESCRIPTION

Foreign Key = PID

Feedback

Adı	Türü	Karşılaştırma	Öznitelikler	Boş	Varsayılan	Açıklamalar	Ekstra
FID	int(11)			Hayır	*Yok*		AUTO_INCREMENT
PID	int(11)			Hayır	*Yok*		
PRISONER_NAME	varchar(30)	latin1_swedish_ci		Hayır	*Yok*		
DATE	date			Hayır	*Yok*		
ESCAPE_ATTEMPTS	int(11)			Hayır	*Yok*		
JOB_PERFORMANCE	varchar(30)	latin1_swedish_ci		Hayır	*Yok*		
COMMENTS	text	latin1_swedish_ci		Hayır	*Yok*		

This table is for the staff to give feedback and write comments in the escape attempts of the prisoners. There are 7 columns in the table.

Primary Key = FID

Functional Dependencies;

FID → PRISONER_NAME

FID → DATE

FID → ESCAPE_ATTEMPTS

FID → JOB_PERFORMANCE

FID → COMMENTS

Foreign Key = PID

Prisoner

Adı	Türü	Karşılaştırma	Öznitelikler	Boş	Varsayılan	Açıklamalar	Ekstra
PID	int(11)			Hayır	*Yok*		AUTO_INCREMENT
PRISONER_NAME	varchar(30)	latin1_swedish_ci		Hayır	*Yok*		
AGE	int(11)			Hayır	*Yok*		
SEX	varchar(6)	latin1_swedish_ci		Hayır	*Yok*		
TYPE	varchar(30)	latin1_swedish_ci		Hayır	*Yok*		
ADDRESS	text	latin1_swedish_ci		Hayır	*Yok*		
ENTRY_DATE	date			Hayır	*Yok*		
RELEASE_DATE	date			Hayır	*Yok*		

This table exists to keep the information of the prisoners. There are 8 columns in the table.

Primary Key = PID

Functional Dependencies;

PID → PRISONER_NAME

PID → AGE

PID → SEX

PID → TYPE

PID → ADDRESS

PID → ENTRY_DATE

PID → RELEASE_DATE

Staff

Adı	Türü	Karşılaştırma	Öznitelikler	Boş	Varsayılan	Açıklamalar	Ekstra
SID	int(11)			Hayır	*Yok*		AUTO_INCREMENT
SNAME	varchar(30)	latin1_swedish_ci		Hayır	*Yok*		
AGE	int(11)			Hayır	*Yok*		
SEX	varchar(6)	latin1_swedish_ci		Hayır	*Yok*		
ROLE	varchar(30)	latin1_swedish_ci		Hayır	*Yok*		
PHONENO	bigint(20)			Hayır	*Yok*		
JOINDATE	date			Hayır	*Yok*		
ADDRESS	text	latin1_swedish_ci		Hayır	*Yok*		
SALARY	int(11)			Hayır	*Yok*		

This table is there to keep the information of the staff and to deal with the visitors. There are 9 columns in the table.

Primary Key = SID

Functional Dependencies;

SID → SNAME

SID → SEX

SID → ROLE

SID → AGE

SID → PHONENO

SID → JOINDATE

SID → SALARY

SID → ADDRESS

Visitor

Adı	Türü	Karşılaştırma	Öznitelikler	Boş	Varsayılan	Açıklamalar	Ekstra
PID	int(11)			Hayır	*Yok*		
VISITOR_NAME	varchar(30)	latin1_swedish_ci		Hayır	*Yok*		
PRISONER_NAME	varchar(30)	latin1_swedish_ci		Hayır	*Yok*		
RELATION	varchar(30)	latin1_swedish_ci		Hayır	*Yok*		
DATE	date			Hayır	*Yok*		
IN_TIME	time			Hayır	*Yok*		
OUT_TIME	time			Hayır	*Yok*		
ISSUING_AUTHORITY	varchar(30)	latin1_swedish_ci		Hayır	*Yok*		
SID	int(11)			Hayır	*Yok*		
VISITOR_ADDRESS	text	latin1_swedish_ci		Hayır	*Yok*		

This table is there to keep the information of the visitors and which staff will take care of it.

There are 10 columns in the table.

Primary Key = VISITOR_NAME

Functional Dependecies;

VISITOR_NAME → PRISONER_NAME

VISITOR_NAME → RELATION

VISITOR_NAME → DATE

VISITOR_NAME → IN_TIME

VISITOR_NAME → OUT_TIME

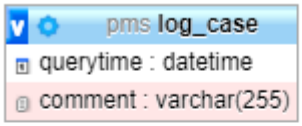
VISITOR_NAME → VISITOR_ADDRESS

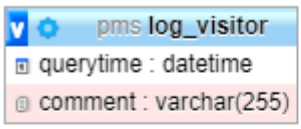
VISITOR_NAME → ISSUING_AUTHORITY

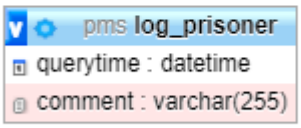
Foreign Key = PID , SID

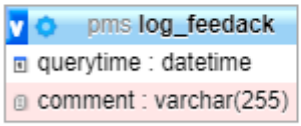
Log Tables

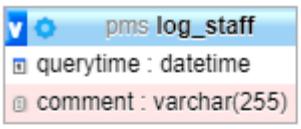
There are 6 log tables. These tables add which data to the log tables when there are changes on the case , prisoner , visitor , feedback and staff tables.

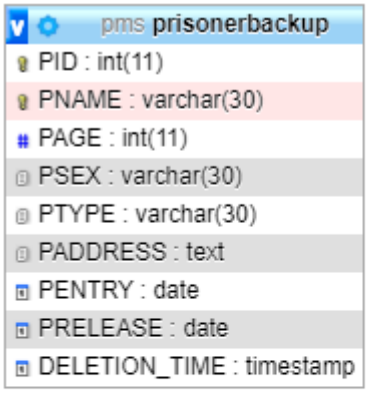












Tables are simply created as in the photo below.

```
CREATE TABLE `prisoner` (
  `PID` int(11) NOT NULL,
  `PRISONER_NAME` varchar(30) NOT NULL,
  `AGE` int(11) NOT NULL,
  `SEX` varchar(6) NOT NULL,
  `TYPE` varchar(30) NOT NULL,
  `ADDRESS` text NOT NULL,
  `ENTRY_DATE` date NOT NULL,
  `RELEASE_DATE` date NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

TRIGGERS

```
DELIMITER $$
CREATE TRIGGER `createhistory` BEFORE DELETE ON `prisoner` FOR EACH ROW INSERT INTO `prisonerbackup`
SET `PID`= OLD.PID,`pname`= OLD.PRISONER_NAME,`PAGE`= OLD.AGE,`PSEX`= OLD.SEX,`PTYPE`= OLD.TYPE,
`PADDRESS`= OLD.ADDRESS,`PENTRY`= OLD.ENTRY_DATE,`PRELEASE`= OLD.RELEASE_DATE
$$
DELIMITER ;
```

This trigger is added to the prisonerbackup table with the date of deletion before one of the prisoners is deleted.

```
DELIMITER $$
create trigger PRISONER_UPDATE_LOG
after update
on `prisoner`
for each row
begin
insert into log_prisoner values
(now(),concat( "[",cast(new.`PRISONER_NAME` as char),"]"," this prisoner is updated."));
end
$$
DELIMITER ;
```

This trigger is added to the log_prisoner table whenever there is a change in the information of the prisoners.

```
DELIMITER $$
create trigger VISITOR_UPDATE_LOG
after update
on `visitor`
for each row
begin
insert into log_visitor values
(now(),concat( "[",cast(new.`VISITOR_NAME` as char),"]"," this visitor is updated."));
end
$$
DELIMITER ;
```

This trigger is added to the log_visitor table whenever there is a change in the information of the visitors.

```
create table log_staff (
  querytime datetime,
  comment varchar(255)
);
DELIMITER $$
create trigger STAFF_UPDATE_LOG
after update
on `staff`
for each row
begin
insert into log_staff values
(now(),concat( "[",cast(new.`SNAME` as char),"]"," this staff is updated."));
end
$$
DELIMITER ;
```

This trigger is added to the log_staff table whenever there is a change in the information of the staffs.

```
DELIMITER $$
create trigger FEEDBACK_UPDATE_LOG
after update
on `feedback`
for each row
begin
insert into log_feedack values
(now(),concat( "[",cast(new.`FID` as char),"]"," this feedback is updated."));
end
$$
DELIMITER ;
```

This trigger is added to the log_feedback table whenever there is a change in the information of the feedbacks.

```
DELIMITER $$
create trigger CASE_UPDATE_LOG
after update
on `case`
for each row
begin
insert into log_case values
(now(),concat( "[",cast(new.`CID` as char),"]"," this case is updated."));
end
$$
DELIMITER ;
```

This trigger is added to the log_case table whenever there is a change in the information of the cases.

VIEWS

```
CREATE VIEW crime_types AS SELECT
`PRISONER_NAME`,`type` FROM `prisoner`;
```

This view selects PRISONER_NAME and type columns and adds them to crime_types.

```
CREATE VIEW staff_salary AS SELECT
`SNAME`,`SALARY` FROM `staff` WHERE staff.SALARY>3000;
```

This view adds the names and salaries of those with salaries greater than 3000 in the staff table to the crime_types.

```
CREATE VIEW feedback_view AS
SELECT `FID`,`ESCAPE_ATTEMPTS`,`COMMENTS`,prisoner.`SEX`,prisoner.`AGE`
FROM `prisoner`
RIGHT JOIN `feedback`
ON `prisoner`.`PID` = `feedback`.`PID`;
```

This view adds the data to the feedback_view by joining the same PIDs in the prisoner and feedback tables.

```
CREATE VIEW staff_view AS
SELECT `PID`,`VISITOR_NAME`,`PRISONER_NAME`,`RELATION`
FROM `staff`
LEFT JOIN `visitor`
ON `staff`.`SID` = `visitor`.`SID`;
```

This view adds the data to the staff_view by joining the same PIDs in the staff and visitor tables.

```
CREATE VIEW prisoner_case_view AS
SELECT prisoner.`PRISONER_NAME`, `CID`,`CASE_TYPE` FROM prisoner
LEFT JOIN `case` ON prisoner.`PID` = `case`.`PID`
UNION
SELECT prisoner.`PRISONER_NAME`, `CID`,`CASE_TYPE` FROM prisoner
RIGHT JOIN `case` ON prisoner.`PID` = `case`.`PID`;
```

This view joins right and left when the PID of the prisoners is equal to the PID in the case table and adds it to the prisoner_case_view.