Database Management Systems Final Report

Nicholas Prashad, Larry Moreno, and Jobin John

Adelphi University

CSC 263: Database Management Systems

Professor Sukun Li

December 21, 2021

Case Study:
Computer Security Incident Response Teams (CSIRTs) track incidents as they occur. When an incident is declared, it is assigned a unique identifier and it is recorded in a database. For each incident, the incident number, a type of incident (chosen from a dynamically changing list), the date it was created, an incident state ('open', 'closed', 'stalled'), and a list of free-form comments is maintained. Associated with each comment is the name of the handler who wrote it.

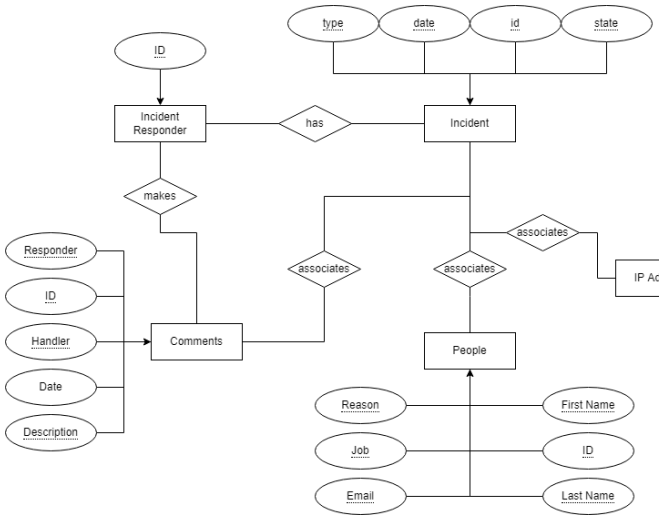
In addition, associated with each incident could be any number of people (last name, first name, job title, email address), and any number of IP addresses. With each IP address or each person, the handler can record a reason for the address association.

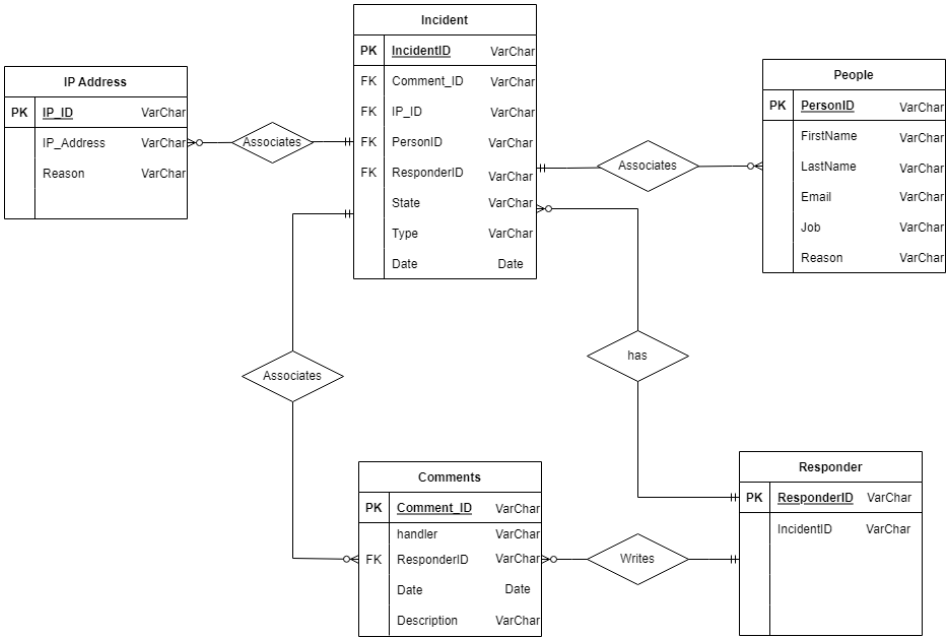
Incident responders must be able to query the database by incident number and receive a report containing the full history of a given incident. Free form comments must be sorted from most recent on top, to oldest on the bottom. In addition, incident responders must be able to record new incidents, change the state of incidents, add comments, and/or add and remove people and IP addresses.

To facilitate information sharing, the incident tracking system must be able to export each incident in a standard incident exchange format and send it via email. Likewise, other teams may send incident-related information via email to the CSIRT operating this application. When that happens, each report received will trigger a new incident being recorded.

All updates to an incident must be recorded as free-form comments to the incident. All use of the system is subject to authentication via an external single sign-on system.

ER Diagram:





Explanation:

The ER Diagram displays the relationships between each entity available to us based on the case study. It helps us to identify attributes and requirements for building a proper database. The second diagram expands upon the basic ER diagram with labels for the foreign and primary keys. The addition of the labeled keys combined with the relationship shown between the entities help visualize the connection between the two. The Incident entity has attributes IncidentID, Comment_ID, IP_ID, PersonID, ResponderID, State, Type Date and IncidentID is the primary key which is used to search in the database. The IncidentID acts as the unique identifier that can be used to gather all other data linked to it with a single query. This works because of the connections between the Incident entity and every other entity. Each incident has comments which are written by responders who have their own ResponderID. Each incident also has an IP Address and Person associated with it represented by the People and IP Address entities.

Relational Database and Normalization:

UNF:

- A table that contains one or more repeating groups.

INCIDENT (IncidentID, IncidentType, IncidentState, IncidentDate, ResponderID, CommentID, Handler, Date, Description, IP_ID, IP_Address, IP_Reason, PersonID, FirstName, LastName, Email, Job, Person_Reason, IncidentNum)

INCIDENT

IncidentID	IncidentType	IncidentState	IncidentDate	ResponderID	CommentID	handler	date	description

IP_ID	IP_Address	PersonID	FirstName	LastName	Email	Job	Person_Reason	IncidentNum

---

1NF:

- UNF - 1NF
  - Repeating groups are removed
  - Primary key is established

INCIDENT (IncidentID, IncidentType, IncidentState, IncidentDate, ResponderID, CommentID, Handler, Date, Description, IP_ID, IP_Address, IP_Reason, PersonID, FirstName, LastName, Email, Job, Person_Reason, IncidentNum)

INCIDENT

IncidentID (PK)	IncidentType	IncidentState	IncidentDate	ResponderID	CommentID	handler	date	description

IP_ID	IP_Address	PersonID	FirstName	LastName	Email	Job	Person_Reason	IncidentNum

---

2NF:

- 1NF- 2NF
  - Is in First Normal Form

- Non key attributes are dependant on primary key (No partial dependency)

INCIDENT (IncidentID, IncidentType, IncidentState, IncidentDate, ResponderID, CommentID, IP_ID, PersonID)
PEOPLE (PersonID, FirstName, LastName, Email, Job, Reason)
COMMENTS (CommentID, Handler, Responder, Date, Description)
IP_ADDRESS (IP_ID, IP_Address, Reason)
RESPONDER (ResponderID, IncidentNum)

INCIDENT

IncidentID (PK)	IncidentType	IncidentState	IncidentDate	ResponderID	CommentID	IP_ID	PersonID

PEOPLE

PersonID (PK)	FirstName	LastName	Email	Job	Reason

COMMENTS

CommentID (PK)	Handler	ResponderID	Date	Description

IP_ADDRESS

IP_ID (PK)	IP_Address	Reason

RESPONDER

ResponderID (PK)	IncidentNum

3NF:

- 2NF- 3NF
  - Is in Second Normal Form
  - No transitive dependencies on primary key (A → B, B → C, A → C)

INCIDENT (IncidentID, IncidentType, IncidentState, IncidentDate, *ResponderID*, *CommentID*, *IP_ID*, *PersonID*)
Foreign Keys:

INCIDENT(CommentID) REFERENCES COMMENT(CommentID)
INCIDENT(ResponderID) REFERENCES RESPONDER(ResponderID)
INCIDENT(PersonID) REFERENCES PEOPLE(PersonID)
INCIDENT(IP_ID) REFERENCES IP_ADDRESS(IP_ID)

PEOPLE (PersonID, FirstName, LastName, Email, Job, Description)

COMMENTS (CommentID, handler, *ResponderID*, CommentDate, description)

Foreign Keys:

INCIDENT(ResponderID) REFERENCES RESPONDER(ResponderID)

IP_ADDRESS (IP_ID, IP_Address, IP_Reason)

RESPONDER (ResponderID, IncidentNum)

INCIDENT

IncidentID	IncidentType	IncidentState	IncidentDate	*ResponderID* (FK)	*CommentID*(FK)	*IP_ID* (FK)	*PersonID* (FK)

PEOPLE

PersonID	FirstName	LastName	Email	Job	Reason

COMMENTS

CommentID	Handler	*ResponderID* (FK)	Date	Description

IP_ADDRESS

IP_ID	IP_Address	Reason

RESPONDER

ResponderID	IncidentNum

BCNF:

- 3NF- BCNF
  - Is in Third Normal Form
  - Every determinant key is a candidate key

INCIDENT (IncidentID, IncidentType, IncidentState, IncidentDate, *ResponderID*, *CommentID*, *IP_ID*, *PersonID*)

Foreign Keys:

INCIDENT(CommentID) REFERENCES COMMENT(CommentID)
INCIDENT(ResponderID) REFERENCES RESPONDER(ResponderID)
INCIDENT(PersonID) REFERENCES PEOPLE(PersonID)
INCIDENT(IP_ID) REFERENCES IP_ADDRESS(IP_ID)

PEOPLE (PersonID, FirstName, LastName, Email, Job, Description)

COMMENTS (CommentID, handler, *ResponderID*, CommentDate, description)

Foreign Keys:

INCIDENT(ResponderID) REFERENCES RESPONDER(ResponderID)

IP_ADDRESS (IP_ID, IP_Address, IP_Reason)

RESPONDER (ResponderID, IncidentNum)

INCIDENT

IncidentID	IncidentType	IncidentState	IncidentDate	*ResponderID* (FK)	*CommentID*(FK)	*IP_ID* (FK)	*PersonID* (FK)

PEOPLE

PersonID	FirstName	LastName	Email	Job	Reason

COMMENTS

CommentID	Handler	Description	Date	*ResponderID* (FK)

IP_ADDRESS

IP_ID	IP_Address	Reason

RESPONDER

ResponderID	IncidentNum

Schema:

Creating The Database and Tables

```
Create DATABASE db_263_project;
Use db_263_project;

CREATE TABLE People(
 personID CHAR(4) NOT NULL,
 firstName CHAR(25) NOT NULL,
 lastName CHAR(25) NOT NULL,
 email VARCHAR(100) NULL,
 job CHAR(25) NULL,
 description VARCHAR(255) NULL,
 CONSTRAINT PK_People PRIMARY KEY (personID)
);

CREATE TABLE Ip_Address(
 ipID VARCHAR(10) NOT NULL,
 ipaddress INT NULL,
 reason VARCHAR(255) NULL,
 CONSTRAINT PK_Ip_Address PRIMARY KEY (ipID)
);

CREATE TABLE Responder(
responderID CHAR(4) NOT NULL,
incidentNum CHAR(4) NOT NULL,
 CONSTRAINT PK_Responder PRIMARY KEY (responderID)
);

CREATE TABLE Comment(
commenID CHAR(4) NOT NULL,
handler CHAR(25) NULL,
description VARCHAR(255) NOT NULL,
date DATE,
responderID CHAR(4) NULL,
CONSTRAINT PK_Comment PRIMARY KEY (commenID),
CONSTRAINT FK_Responder FOREIGN KEY (responderID) REFERENCES Responder
(responderID)
);
```

```
CREATE TABLE Incident(
incidentID CHAR(4) NOT NULL,
type VARCHAR(25) NOT NULL,
state VARCHAR(25) NOT NULL,
date DATE,
responderID CHAR(4) NULL,
commenID CHAR(4) NULL,
ipID VARCHAR(10) NULL,
personID CHAR(4) NULL,
CONSTRAINT PK_Incident PRIMARY KEY (incidentID),
CONSTRAINT FK_Responded FOREIGN KEY (responderID) REFERENCES Responder
(responderID),
CONSTRAINT FK_Comment FOREIGN KEY (commenID) REFERENCES Comment
(commenID),
CONSTRAINT FK_Ip_Address FOREIGN KEY (ipID) REFERENCES Ip_Address (ipID),
CONSTRAINT FK_Person FOREIGN KEY (personID) REFERENCES People (personID),
CONSTRAINT CHK_State CHECK (state = 'open' OR state = 'closed' OR state = 'stalled')
);
```

Insert Data Into Table

```
INSERT INTO People VALUES('P001', 'Larry', 'Moreno', 'larrymoreno@mail.adelphi.edu',
'VicePresdient', 'Bystander');
INSERT INTO People VALUES('P002', 'Nicholas', 'Prashad',
'nicholasprashad@mail.adelphi.edu',  'President', 'Witness');
INSERT INTO People VALUES('P003', 'Jobin', 'John', 'jobinjohn@mail.adelphi.edu',
'Secretary', 'Injured');

INSERT INTO Ip_Address VALUES('I001','123.456','Connected Computer Center');
INSERT INTO Ip_Address VALUES('I002','456.789','President Phone IP Address');
INSERT INTO Ip_Address VALUES('I003','789.123','Camera #3');

INSERT INTO Responder VALUES('R001', 'N001');
INSERT INTO Responder VALUES('R002', 'N002');
INSERT INTO Responder VALUES('R003', 'N003');

INSERT INTO Comment VALUES('C001', 'John', 'Microwave Fire', '2021-12-01 0:00:00',
'R001');
INSERT INTO Comment VALUES('C002', 'Peter', 'Leaking Pipe', '2021-12-05 0:00:00', 'R002');
```

INSERT INTO Comment VALUES('C003', 'Henry', 'Loose Elevator Cable', '2021-12-08 0:00:00', 'R003');

INSERT INTO Incident VALUES('N001', 'Microwave Fire', 'open', '2021-12-01 0:00:00', 'R001', 'C001', 'I001', 'P001');
INSERT INTO Incident VALUES('N002', 'Leaking Pipe', 'closed', '2021-12-05 0:00:00', 'R002', 'C002', 'I002', 'P002');
INSERT INTO Incident VALUES('N003', 'Loose Elevator Cable', 'stalled', '2021-12-08 0:00:00', 'R003', 'C003', 'I003', 'P003');

HTML/PHP/CSS:

Dbconnect.php:

- File is used to establish connection to the mysql database
- Hostname, Database username, Database password, and Database name are all left blank to allow the current user to input their information.
- The connection is made using the provided information and then returned for use in other files.

```
<!DOCTYPE html>
<html>
<head>
    <title>PHP Connect MySQL Database</title>
</head>
<body>
    <p><?php
        function connect_db()
        {
            $db_host = "localhost";   // Hostname
            $db_user = "root";  // Database username
            $db_pass = "Pr@shAd3lpH1";  // Database Password
            $db_name = "db_263_project";  // Database name

            $conn = new mysqli($db_host, $db_user, $db_pass, $db_name) or die("Connection failed: %s\n" . $conn->error);

            return $conn;
        }
    ?></p>
</body>
</html>
```

Main.php:

- This is a mage page that will act as an intermediate between all other pages.
- This home page will allow a viewer to access any page needed.

```
<html>
  <title> Computer Security Incident Response Team </title>
  <body style="background-color:white">
  <center>
    <h1> Computer Security Incident Response Team </h1>
    <h3>  </h3>
    <p> Click on the action you would like to perform: </p>

    <p> <h4> <a href='addIncident.php'> Add New Incident Information </a> </h4> </p>

    <p> <h4> <a href='updateInfo.php'> Update Incident Information </a> </h4> </p>

    <p> <h4> <a href='RemoveInfo.php'> View Incident Information </a> </h4> </p>

    <p> <h4> <a href='viewInfo.php'> View Incident Information </a> </h4> </p>

  </center>
  </body>
</html>
```

AddInfo.php:

- This pages allows users to create new incidents and add them to the database

```php
<?php
// Import dbconnect script here to use database connection
require_once __DIR__ . '/dbconnect.php';
// Database connection object to access mysql functions
$con = connect_db();

// Check if submit button is clicked
if (isset($_GET['addIncident'])) {
  // Get data from request
  $IncidentId = $_GET['IncidentID'];
  $IncidentType = $_GET['IncidentType'];
  $IncidenState = $_GET['IncidentState'];
  $Date = $_GET['Date'];

  // Add incident to database
  $sql = $con->query("INSERT INTO INCIDENT(incidentID, type, state, date) VALUES ('$IncidentId', '$IncidentType', '$IncidenState', '$Date')");

  // Check if query ran successfully
  if ($sql) {
    // Query ran successfully show success message
    echo "<script>alert('Successfully added!')</script>";
  } else {
    // Show Error
    echo "<script>alert('Some error occurred please check your data!')</script>";
  }
}

?>

<!DOCTYPE html>
<html>

<head>
    <title>Add New Incident Information</title>
    <!-- CSS for basic styling -->
    <link rel="stylesheet" href="styles2.css">
</head>

<body>
    <div class="header">
        <h2>Add New Incident Information</h2> <br>
        <a href="viewInfo.php">View Incident Information</a>
        <a href="addInfo.php">Add New Incidents</a>
        <a href="updateInfo.php">Update Incident Information</a>
        <a href="removeInfo.php">Remove Incident Information</a>
    </div>

    <div class="container">
        <!-- Show a form to user to update incident information-->
        <form method="get">
            <input type="VarChar" class="input" name="IncidentID" placeholder="Enter the Incident ID." required> <br>
            <input type="VarChar" class="input" name="IncidentType" placeholder="Enter the Incident Type" required> <br>
            <input type="VarChar" class="input" name="IncidentState" placeholder="Enter the Incident State" required> <br>
            <input type="Date" class="input" name="Date" placeholder="Enter the date of Incident" required> <br>
            <button type="submit" class="btn" name="addIncident">Submit</button>
        </form>
    </div>
</body>

</html>
```

UpdateInfo.php:

- This page allows users to edit information of current or past incidents
- Users can add new comments or change the state of the incident

```php
<?php
// Import dbconnect script here to use database connection
require_once __DIR__ . '/dbconnect.php';
// Database connection object to access mysql functions
$con = connect_db();

// Check if submit button is clicked
if (isset($_GET['updateState'])) {
  // Get data from request
  $IncidentID = $_GET['incidentID'];
  $IncidentState = $_GET['state'];

  // Update state in database
  $sql = $con->query("UPDATE INCIDENT SET state = '$IncidentState' WHERE incidentID = '$IncidentID'");

  // Check if query ran successfully
  if ($sql) {
    // Query ran successfully show success message
    echo "<script>alert('Successfully updated!')</script>";
  } else {
    // Show Error
    echo "<script>alert('Some error occurred please check your data!')</script>";
  }
}

?>

<!DOCTYPE html>
<html>

<head>
    <title>Update Existing Incident Information</title>
    <!-- CSS for basic styling -->
    <link rel="stylesheet" href="styles2.css">
</head>

<body>
    <div class="header">
        <h2>Update Existing Incident Information</h2> <br>
        <a href="viewInfo.php">View Incident Information</a>
        <a href="addInfo.php">Add New Incidents</a>
        <a href="updateInfo.php">Update Incident Information</a>
        <a href="removeInfo.php">Remove Incident Information</a>
    </div>

    <div class="container">
        <!-- Show a form to user to update incident information-->
        <form method="get">
            <input type="VarChar" class="input" name="incidentID" placeholder="Enter the Incident ID." required> <br>
            <input type="VarChar" class="input" name="state" placeholder="Enter the Incident State" required> <br>
            <button type="submit" class="btn" name="updateState">Submit</button>
        </form>
    </div>
</body>

</html>
```

RemoveInfo.php:

- This page allows users to remove information from a specific incident.

```php
<?php
// Import dbconnect script here to use database connection
require_once __DIR__ . '/dbconnect.php';
// Database connection object to access mysql functions
$con = connect_db();

// Check if submit button is clicked
if (isset($_GET['removeIncident'])) {
  // Get data from request
  $IncidentId = $_GET['IncidentId'];

  // remove incident to database
  $sql = $con->query("DELETE FROM incident WHERE incidentID = '$IncidentId'");

  // Check if query ran successfully
  if ($sql) {
    // Query ran successfully show success message
    echo "<script>alert('Successfully removed!')</script>";
  } else {
    // Show Error
    echo "<script>alert('Some error occurred please check your data!')</script>";
  }
}


?>

<!DOCTYPE html>
<html>

<head>
    <title>Remove Incident Information</title>
    <!-- CSS for basic styling -->
    <link rel="stylesheet" href="styles2.css">
</head>

<body>
    <div class="header">
        <h2>Remove Incident Information</h2> <br>
        <a href="viewInfo.php">View Incident Information</a>
        <a href="addInfo.php">Add New Incidents</a>
        <a href="updateInfo.php">Update Incident Information</a>
        <a href="removeInfo.php">Remove Incident Information</a>
    </div>

    <div class="container">
        <!-- Show a form to user to remove incident information-->
        <form method="get">
            <input type="VarChar" class="input" name="IncidentId" placeholder="Enter the Incident ID." required> <br>
            <button type="submit" class="btn" name="removeIncident">Submit</button>
        </form>
    </div>
</body>

</html>
```

ViewInfo.php

- The page allows users to query the database and retrieve all data around the chosen incident

```php
<?php
// Import dbconnect script here to use database connection
require_once __DIR__ . '/dbconnect.php';
// Database connection object to access mysql functions
$con = connect_db();

?>
<!DOCTYPE html>
<html lang="en">

<head>
  <title>View Incident Information</title>
  <!-- CSS for basic styling -->
  <link rel="stylesheet" href="styles2.css">
</head>

<body>
  <div class="header">
    <h2>Incident Information</h2> <br>
    <a href="viewInfo.php">View Incident Information</a>
    <a href="addInfo.php">Add New Incidents</a>
    <a href="updateInfo.php">Update Incident Information</a>
    <a href="removeInfo.php">Remove Incident Information</a>
  </div>

  <div class="container">
    <!-- Show a form to get Incidentid and if incidentID matches one in teh database then show incident information -->
    <?php if (empty($_GET)) : ?>

      <form method="get">
        <input type="VarChar" class="input" name="incidentID" id="incidentID" placeholder="Enter the incidentID." required>
        <button type="submit" class="btn" name="checkIncident">Submit</button>
      </form>

    <?php else : ?>
      <div class="card">
        <?php
        // Check if user submitted the form
        if (isset($_GET['checkIncident'])) {
          // Get incident id
          $incidentID = $_GET['incidentID'];

          // Check for incident in database
          // Create a query to fetch data from incident table
          $sql1 = $con->query("SELECT * FROM Incident WHERE incidentID = '$incidentID'");

          // If number of rows in result is zero then print error
          if ($sql1->num_rows == 0) {
        ?>
            <div class="error">
              Incident not found.
            </div>
          <?php
          } else {
            // Get incident info from incident table and print incident information
            $sql = $con->query("SELECT * FROM INCIDENT WHERE incidentID = '$incidentID'");
            $Incident = $sql->fetch_array();

            // Print details
          ?>
            <b>Incident ID: </b> <?= $Incident['incidentID'] ?> <br>
            <b>Incident Type: </b> <?= $Incident['type'] ?> <br>
            <b>Incident State: </b> <?= $Incident['state'] ?> <br>
            <b>Date of Incident: </b> <?= $Incident['date'] ?> <br>
            <b>Associated People: </b> <?= $Incident['personID'] ?> <br>
            <b>Associated IP: </b> <?= $Incident['ipID'] ?> <br>

            <hr>

          <?php
            }
          }
          ?>
      </div>
    <?php endif ?>
  </div>
</body>

</html>
```

Styles.css

- Provides the colors and font for all parts of the website

```css
body {
  margin: 0px;
  font-family: Helvetica;
}

.header {
  background-color: gray;
  padding: 8px;
  color: black;
  align-self: center;
  overflow: hidden;
  text-align: center;
}

.container {
  width: 80%;
  margin: auto;
  padding: 20px;
  text-align: center;
}

.card {
  margin: auto;
  border: 1px gray;
  border-radius: 4px;
  padding: 20px 10px;
  text-align: center;
}

.input {
  margin-top: 4px;
  padding: 8px;
}

.btn {
  margin-top: 4px;
  padding: 8px;
  background-color: gray;
  color: black;
  border: 2px black;
  border-radius: 4px;
  cursor: pointer;
}

table {
  margin-left: auto;
  margin-right: auto;
}
```