Cairo University - Faculty of Engineering

Computer Engineering Department

Advanced Database Systems

Project Phase one

Mohamed Shawky Zaky
SEC:2, BN:15

Remonda Talaat Eskarous
SEC:1, BN:19

Mohamed Ahmed Mohamed Ahmed
SEC:2, BN:10

Mohamed Ramzy Helmy
SEC:2, BN:13

Contents

List of Figures

1 System Description

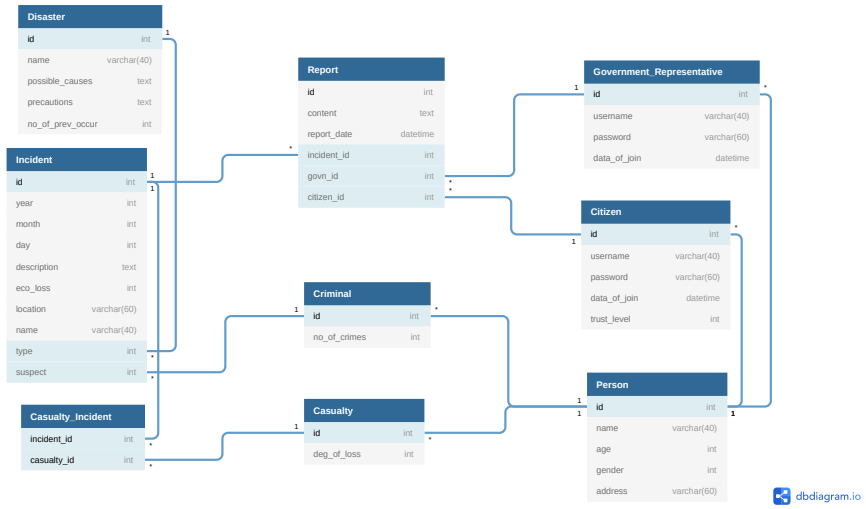

Figure 1: Database Schema

1.1 Schema Illustration

The chosen system consists of a database that stores *natural* and *man-made disasters* for creating an *incident report website*. The database consists of 9 relations, which are described as follows :

1. **Disaster Relation :** contains the *names* of the main natural and man-made disasters, for example: *the names of famous hurricanes and floods*, and their *causes* and *precautions*.

2. **Incident Relation :** contains the information of *specific incidents* of the disasters, like their *dates, locations and descriptions*.

3. **Person Relation :** an abstract relation of *all types of persons* that can exist in the database, contains the meta information of any person *(name, age, gender and address)*.

4. **Citizen Relation :** contains information of a citizen, *which is the person that can report an incident on the website*. This information includes *username, password, date of join and trust level of the citizen (used to weight the submitted report)*.

5. **Government Representative Relation :** contains information of a government representative, *which is the person that can review incident reports on the website.* This information includes *username, password and date of join.*

6. **Casualty Relation :** contains the information of a certain casualty in an incident, *which is basically the degree of loss.*

7. **Criminal Relation :** contains the information of a certain criminal that committed an incident, *which is basically the number of crimes committed before.*

8. **Report Relation :** contains the details of a submitted report, *such as its content and date.* Also, it refers to *a specific incident, a specific citizen that submitted the report and a specific government representative that will review the report.*

9. **Casualty_Incident Relation :** this is basically a relation to specify, *which casualties were in a specific incident (M:N relationship).*

1.2 Hardware Specifications

- **Operating System :** Ubuntu 20.04
- **CPU :** Intel $i5$ 6600k
- **Utilized RAM Capacity :** 10GB.
- **Utilized Hard Disk Storage :** 200GB *(Current Database Size: 250MB).*

2 ER Diagram

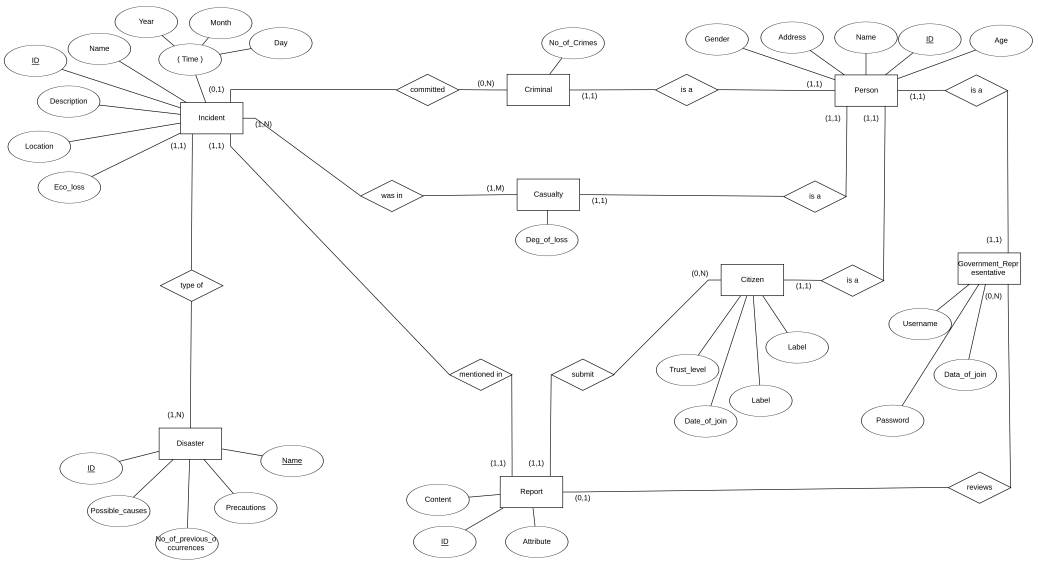

Figure 2: Database ER Diagram

Figure 2 shows the entity relationship *(ER)* diagram of the proposed system. We can see that the proposed system covers most of the database features :

- All relationship cardinalities.
- Single and multiple inheritance.
- Single and composite keys.
- Various relationships between entities.
- Different record sizes for different relations.

3 Database Filling Statistics

Table Name	Row Count	Main Key	Indexes	FK
Disaster	200000	YES	2	0
Incident	199998	YES	3	2
Person	200000	YES	1	0
Citizen	50000	YES	1	1
Government Representative	50000	YES	1	1
Casualty	49999	YES	1	1
Criminal	50000	YES	1	1
Report	199997	YES	4	3
Casualty_Incident	199996	YES *(Composite)*	2	2

Table 1: Database Filling Report Part 1

Table Name	Identity Column	Max Row Size (Bytes)
Disaster	YES	131118
Incident	YES	65663
Person	YES	109
Citizen	NO	116
Government Representative	NO	112
Casualty	NO	8
Criminal	NO	8
Report	YES	65559
Casualty_Incident	NO	8

Table 2: Database Filling Report Part 2

4 NoSQL Implementation

For *NoSQL* implementation, we use *MongoDB*. The 9 relations are represented by 2 collection as follows :

1. **Reports** collection : This collection contains documents that describe the report information and the related incident. *Incident* and *Disaster* tables are embedded inside it. Also, it links to **Persons** collection using *person id*.

2. **Persons** collection : This collection contains documents that describe the person information for each of *citizen*, *criminal*, *government representative* and *casualty*. All of them share some basic attributes and differ in others.

The database is designed in such way, in order to optimize both performance *(number of queries)* and storage *(redundant data and document size)*. We have only 2 collections, which significantly reduces the number of queries *(disk accesses)*, meanwhile the only redundant data that can exist is that of the *disaster* table, which is relatively small in a *NoSQL DBMS*. Also, we utilize the power of *NoSQL*, so that all person types are combined into a single collection, even if they can have some different fields *(which is feasible in NoSQL)*.