World Crisis Database

Phase 1 Technical Report

University of Texas at Austin
Computer Science
CS373 Software Engineering

Team Crisix

Tom Torres
Ejenio Capetillo
Jonathan Velasquez
Enam Ayivi
Matt Thurner
Dalton Schmidt

Introduction

Problem

In today's age, things come and go in the blink of an eye. Technology has allowed us to communicate effectively, and it seems as if a new crisis blazes through the media every day. Whether it is an environmental disaster, terrorist attack, or a nation in revolt, the spotlight quickly turns to the crisis at hand but abruptly changes focus as a new crisis spawns. It's easy to forget about the chaotic things that have happened in our world, and the Crisix Database is a tool designed to help us help ourselves and remember the events that have shaped the society we live in.

Mission

Our goal is to provide unbiased, yet informative information about crises that have occurred globally. Among this information, we will provide a synopsis of each crisis, impact on society, along with organizations and people involved. This information will be easily accessible on the web with each item (crisis, person, or organization) having a unique page. Each page will consist of images, videos, and external links, providing the user a wealth of resources related to a crisis. Our mission will be considered successful if we provide a perennial framework that gives users a hub to explore various aspects of a crisis, along with pointing them to resources that can provide relief to the victims.

Use Case

We've outlined a typical use case via Alistair Cockburn that follows the typical UML "actor" paradigm:

Title: Research a crisis
Goal: Search WCDB for information about a particular crisis
Actor: A computer user with Internet access
Scope: Have a successful query that is in the database
Level: User-goal, sea-level, return a successful query
Story: The user goes to the WCDB and queries a crisis, person, or organization. On a successful search, the user will be directed to a page containing the relevant data. The data will be item specific but will relate to some crisis. From here, the user can click on links that contain information on a related item, or he can click on links to external sites, or even multimedia.

We hope to provide users with relevant information that will keep them informed, but this is only a secondary concern. Our main goal is to direct our users so that they have the information and opportunity to help.

Design

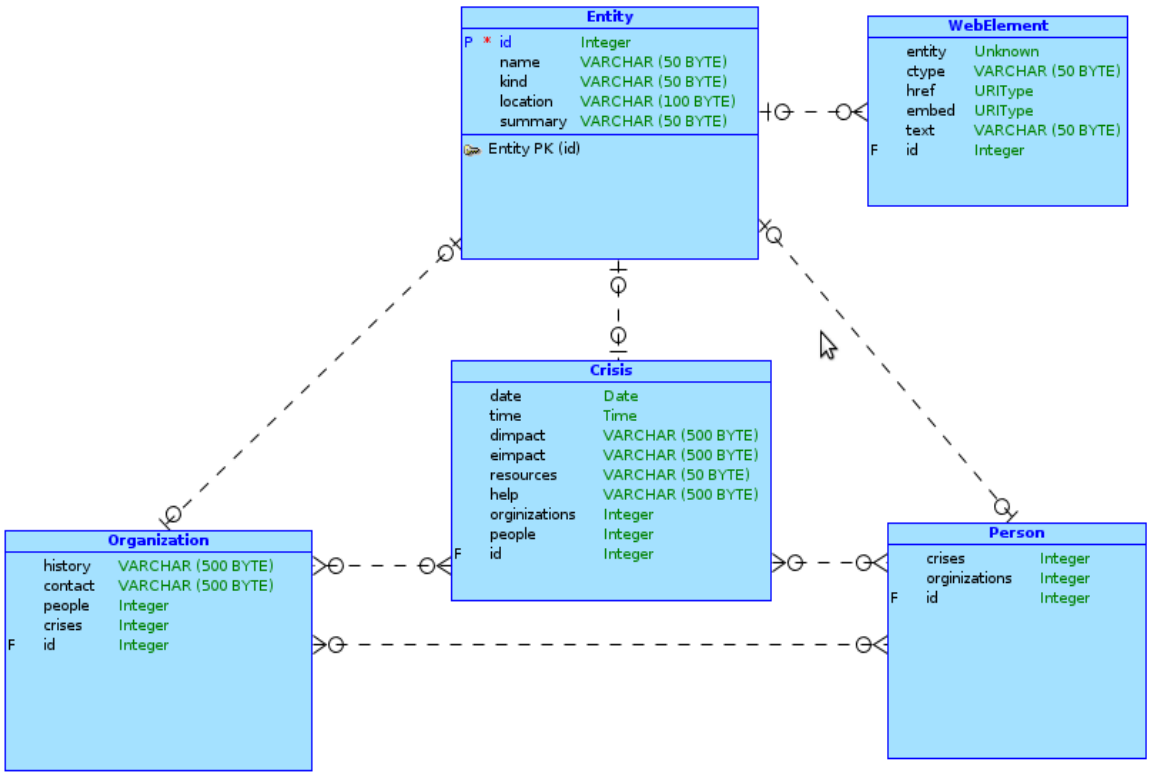
XML Schema

The XSD schema we used to design our XML file is the same schema created by fellow classmates. The schema uses ID/IDREF to avoid data duplication. The schema is written so that all crises, people and organizations are grouped together. Attributes for all three include name, location, kind, citations, and external links. Attributes specific to crises include date and time, human impact, economic impact, resources needed, and ways to help. Organizations maintain history and contact information. Apart from these attributes, there are also common elements (embedded data) such as images, videos, maps, and social network feeds.

Models

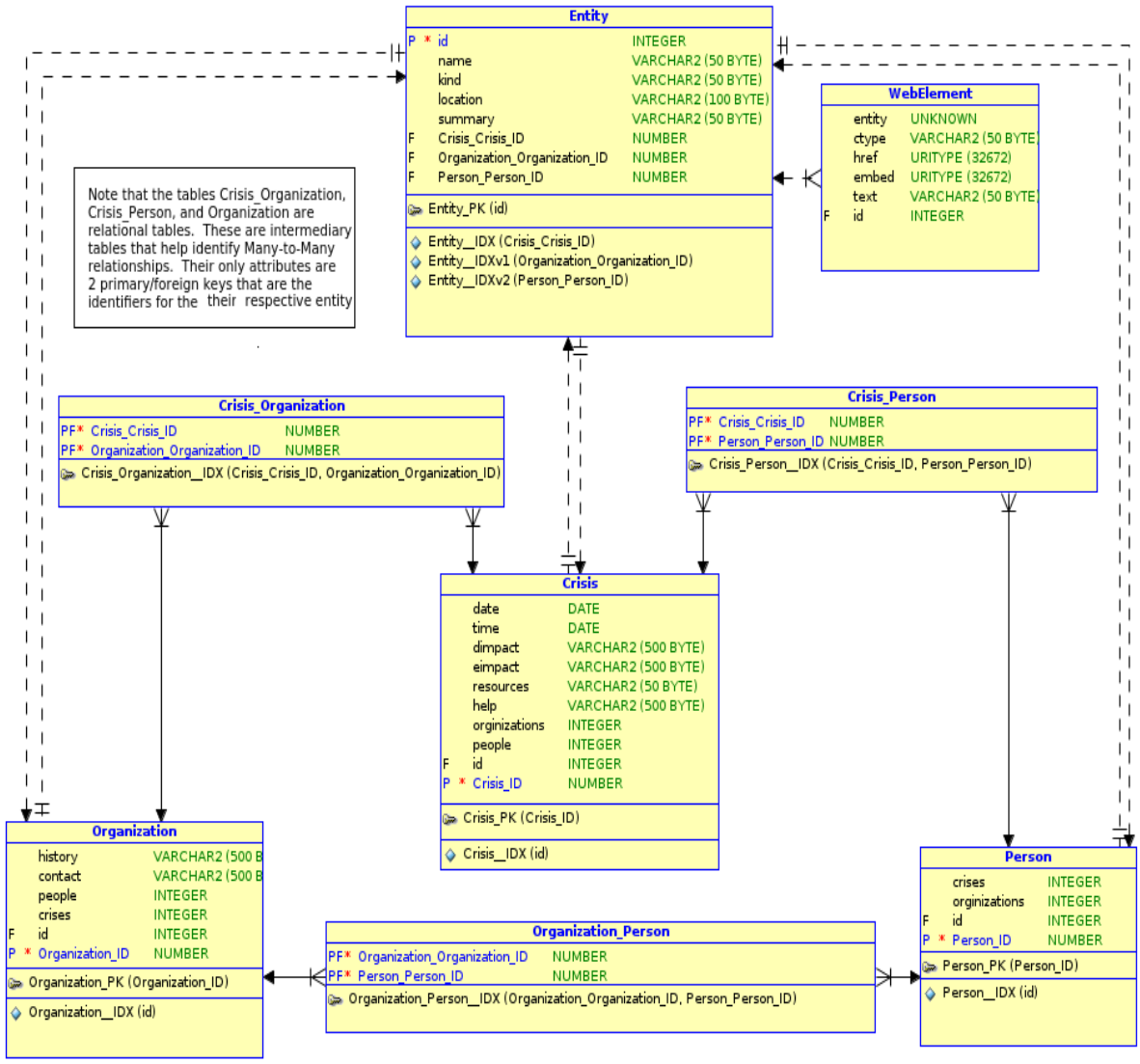
We created a set of Django models to represent our data. In an object-orient programming sense, the models (classes) exhibit inheritance. We have a parent class, named Entity, that contains all of the generic attributes. The direct children of the Entity class are the Crisis, Organization, and Person classes. The end result is just what you'd assume: when constructing a Crisis, Organization, or Person, an Entity is also implicitly constructed. The way this translates into the database is similar.

Crisix Logical Model



A

fourth table, Entity, is created. It is entirely separate and relates to its children via one-to-one unique relationships.



Implementation

Environment

Views

Import/Export

Testing

Unit Tests