Data Analytics (Data Warehouse and Visualization) project

author: Joanna Binek

1. Description of a dataset

Chosen dataset - Global Terrorism Database

The Global Terrorism Database (GTD) is an open-source database including information on terrorist attacks around the world from 1970 through 2017 (except 1993*)*. The GTD includes systematic data on domestic as well as international terrorist incidents that have occurred during this time period and now includes the exact number of 181 691 attacks. Incidents from the GTD follow a 12-digit Event ID system:

- first 8 numbers – date recorded "yyyymmdd"
- last 4 numbers – sequential case number for the given day (0001, 0002 etc). This is "0001" unless there is more than one case occurring on the same date

The *.csv* file with the dataset contains 130 columns which provides numerous characteristics about terrorist attacks, which can be roughly grouped as follows:

- information about time and date,
- information about location,
- information about tactics,
- information about type of attack,
- information about perpetrators,
- information about targets,
- information about outcomes

2. Preprocessing and cleaning data

Preprocessing and cleaning data tasks are very important part of the project as it is a set of operations performed on the dataset to modify ambiguous data which can be a bottleneck to analytical results. It also helps in converting this raw data into a more meaningful, focused, interpretable and readable format.
The GTD dataset is incomplete, inconsistent, contains some errors, often missing attributes values, and outliers. Below data preprocessing tasks helped to resolve these discrepancies.

a) Manual analysis of the dataset and removing useless and irrelevant columns (analysis based on the official codebook)

   After the lecture of the official codebook dedicated to the GTD dataset and understanding the meaning of all attributes, some of them have been removed from the *.csv* file as they were considered not useful in further analysis and achieving final goals of the project. Some of them were removed also due to almost complete lack of data in particular columns (f.e. *aproxdate* where 95% of data are nulls).

b) Calculating percentage of missing values for each column and removing columns for which calculated value is equal or greater than 80% (dimensionality reduction)

In the dataset, there are numerous fields like 'nhours', 'attacktype2_txt', 'ndays', 'nreleased' which are missing either due to information not available or that the field was not relevant for that specific event. Hence such fields were removed to reduce complexity and to avoid reducing the representativeness of the data.

```
def null_vals(df):
    df_null = pd.DataFrame()
    for col in df.columns:
        df_null.loc[col, 'name'] = df[col].name
        df_null.loc[col, 'null values [%]'] =
round(df[col].isnull().sum()*100/ len(df),2)
        df_null.loc[col, 'dtype'] = df[col].dtype

    return df_null.sort_values(ascending = False, by = 'null values [%]')
```

Figure 1. Function to determine percentage of missing values

As a result following attributes were removed:

['weaptype4_txt', 'attacktype3_txt', 'ransompaid', 'targsubtype3_txt', 'natlty3_txt', 'targtype3_txt', 'target3', 'ransomamt', 'weaptype3_txt', 'kidhijcountry', 'nhours', 'attacktype2_txt', 'ndays', 'nreleased', 'targsubtype2_txt', 'natlty2_txt', 'hostkidoutcome_txt', 'target2', 'targtype2_txt', 'weaptype2_txt', 'nhostkidus', 'nhostkid', 'claimmode_txt', 'alternative_txt'].

In this step the data integration part was also done as possible conflicts among data are resolved. F.e. different representations of the same data such as multiple subcategories of weapon type (weapsubtype1, weapsubtype2, weapsubtype3) should be put together to avoid confusion and duplications, but as their content were mostly empty, they have been removed.

c) Analysis of empty values and their conversion

In order to make the empty values consistent they were converted into one applicable form. As their previous values differed from each other in different attributes which caused inconsistency and noise, all data which were empty or stated for lack of information were converted based on the type of the column (string - “Unknown”, float - “-99”).

```
# converting NaN values in columns based on their data types:
for col in cols_missing:
  if GTD_data[col].dtype == np.float64:
    GTD_data[col] = GTD_data[col].fillna(-99)
  else:
    GTD_data[col] = GTD_data[col].fillna("Unknown")
```

Figure 2. Converting NaN values in columns based on their data types

d) Dimensionality reduction caused by geographical changes

Taking into consideration that from 1970 geopolitical situation has changed and some countries don’t exist or have transformed into other countries, some changes were necessary. The cleaning process was about:

- dimensionality reduction in the "country" column (f.e. West Germany and East Germany -> Germany)
- dimensionality reduction in the "city" column (f.e. Rome district -> Rome)

All preprocessing and cleaning data tasks were performed by using Python language (numpy and pandas libraries) and Google Sheets.

3. Relational database

a) Creation of a relational schema

First step was about dividing the original .csv file into smaller ones in order to create a relational database from them by using SQLite software. This part was done programmatically with Python language. Based on existing attributes and their relationships, I created a relational schema for the database:

attack_table (**attack_id INTEGER PRIMARY KEY**, extended INTEGER, nperps INTEGER, nperpcap INTEGER, claimed BOOLEAN, nkill INTEGER, nkillus INTEGER, nkillter INTEGER, nwound INTEGER, nwoundus INTEGER, nwoundte INTEGER, is_property_damage BOOLEAN, ishostkid BOOLEAN, ransom INTEGER, INT_LOG BOOLEAN, INT_IDEO BOOLEAN,
id_time_table CHAR REFERENCES time_table (id), *id_location_table* CHAR REFERENCES location_table (id),
id_category_table CHAR REFERENCES category_table (id), *id_attack_type_table* CHAR REFERENCES attack_type_table (id),
id_victim_table CHAR REFERENCES victim_table (id), *id_perpetrator_table* CHAR REFERENCES perpetrator_table (id),
id_weapon_table CHAR REFERENCES weapon_table (id), *id_property_damage_table* CHAR REFERENCES property_damage_table (id));

attack_type_table (**id CHAR PRIMARY KEY**, attacktype CHAR);

category_table (**id CHAR PRIMARY KEY**, category1 BOOLEAN, category2 BOOLEAN, category3 BOOLEAN);

location_table (**id CHAR PRIMARY KEY**, country CHAR, region CHAR, provstate CHAR, city CHAR, latitude DOUBLE, longitude DOUBLE);

perpetrator_table (**id CHAR PRIMARY KEY**, gname CHAR);

property_damage_table (**id CHAR PRIMARY KEY**, property_damage CHAR);

time_table (**id CHAR PRIMARY KEY**, year INTEGER, month INTEGER, day INTEGER);

victim_table (**id CHAR PRIMARY KEY**, targtype CHAR, targsubtype CHAR, target_name CHAR, target_natlty CHAR);

weapon_table (**id CHAR PRIMARY KEY**, weaptype CHAR);

b) Creation of relation database and ERD

During the next step I imported smaller .csv files into SQLite as tables and created a relational database. I established all relations between tables which let me create an Entity Relationship Diagram which illustrates how entities relate to each other within a database.

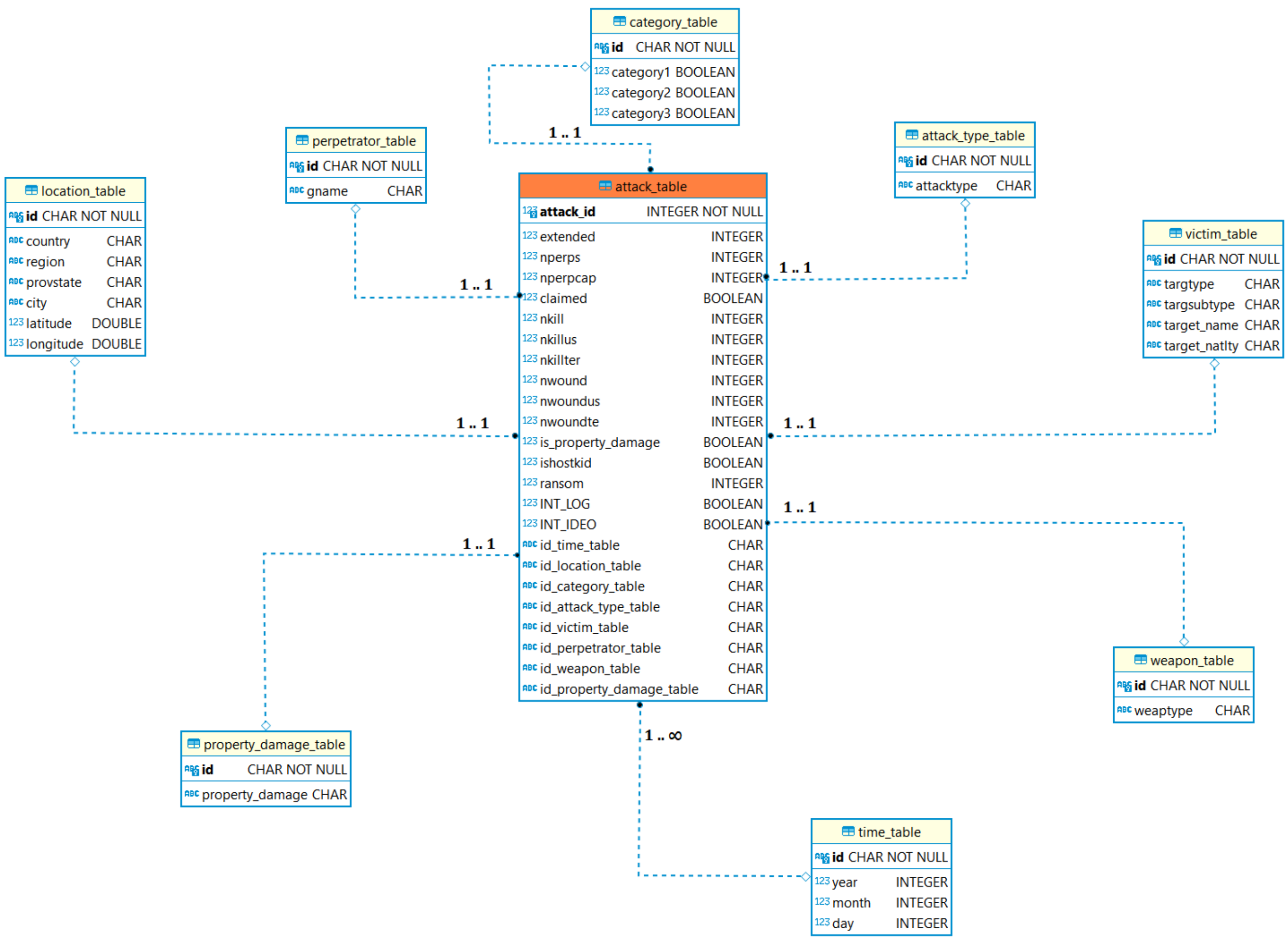


Figure 3. Entity Relationship Diagram

c) Attribute tree

Having a Entity Relationship Diagram I created an Attribute tree in a following way:

- The primary key of the main entity becomes the root (ATTACK = attack_id)
- Each attribute and identifier within the ERD schema becomes a node (a node may represent a fact, a fact attribute (key figure), a dimension, a dimension attribute or a non-dimension attribute of the following fact schema)
- If a node corresponds to a primary key of an entity in the ERD schema, all other nodes representing attributes of this entity are attached to it
- Circles marked with green color represent dimensions
- Circles marked with white color represent fact attributes
- Black circle represent root

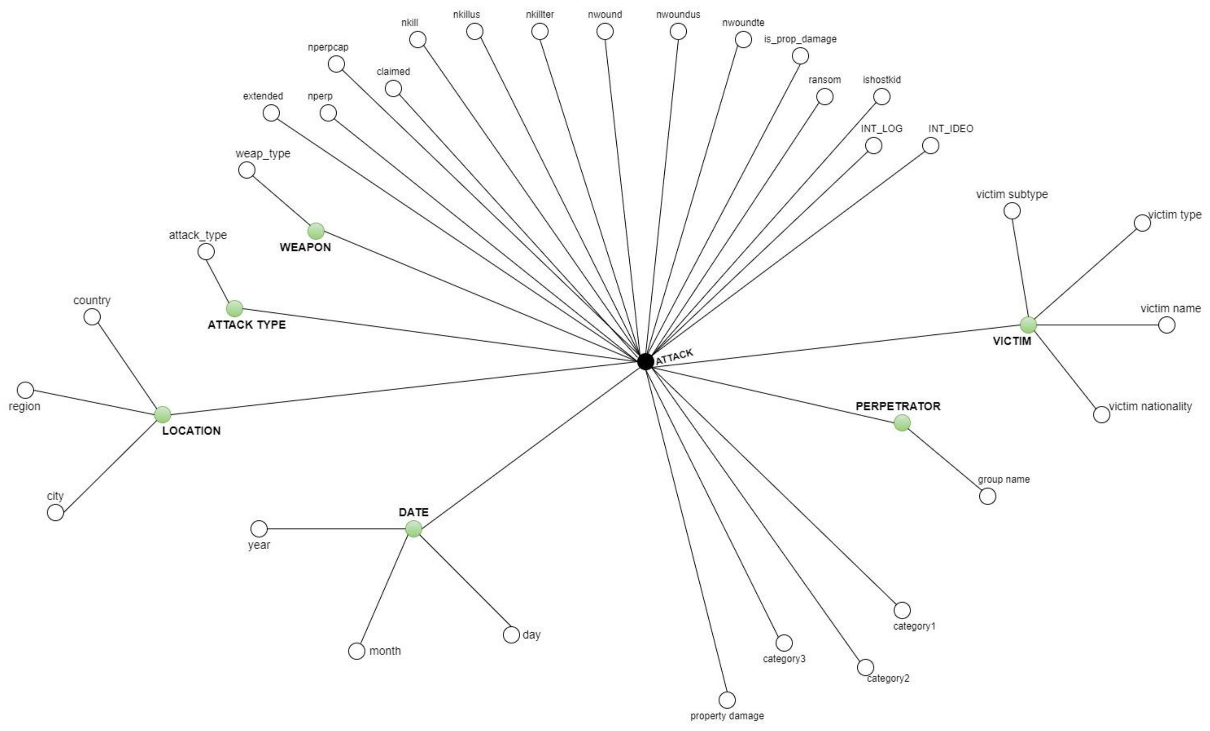


Figure 4. Attribute tree

d) Reduction of the Attribute Tree

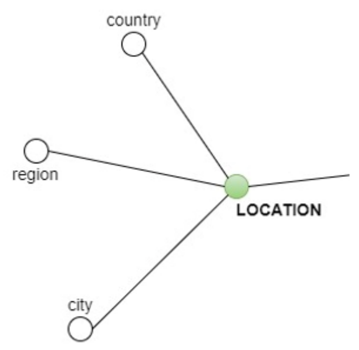
Not all of the tree's attributes (nodes) were interesting or necessary for the data warehouse project, so the next step was eliminating unwanted or unnecessary information from the attribute tree.

I decided to leave LOCATION, WEAPON, ATTACK_TYPE, DATE, VICTIM and PERPETRATOR dimensions as they were and reduce some of the nodes.

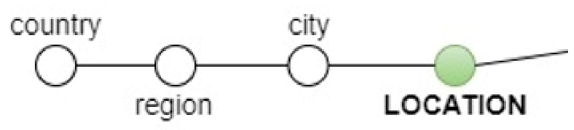
Interesting and valuable from the point of view of future visualizations were is_prop_damage, category1, category2, category3, nkill, nwound, nprep, nprepcap, ransom and INT_LOG.

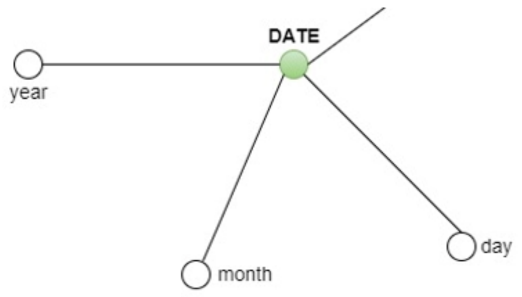
The rest of nodes (removed ones) contained not useful information that wouldn't be helpful during planned visualizations.

e) Defining hierarchies
In this step I defined the hierarchies for the dimensions identified previously. Using the attribute tree I had a meaningful basis for the organization of the hierarchies.

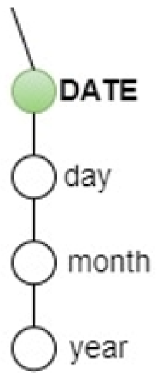


→





→



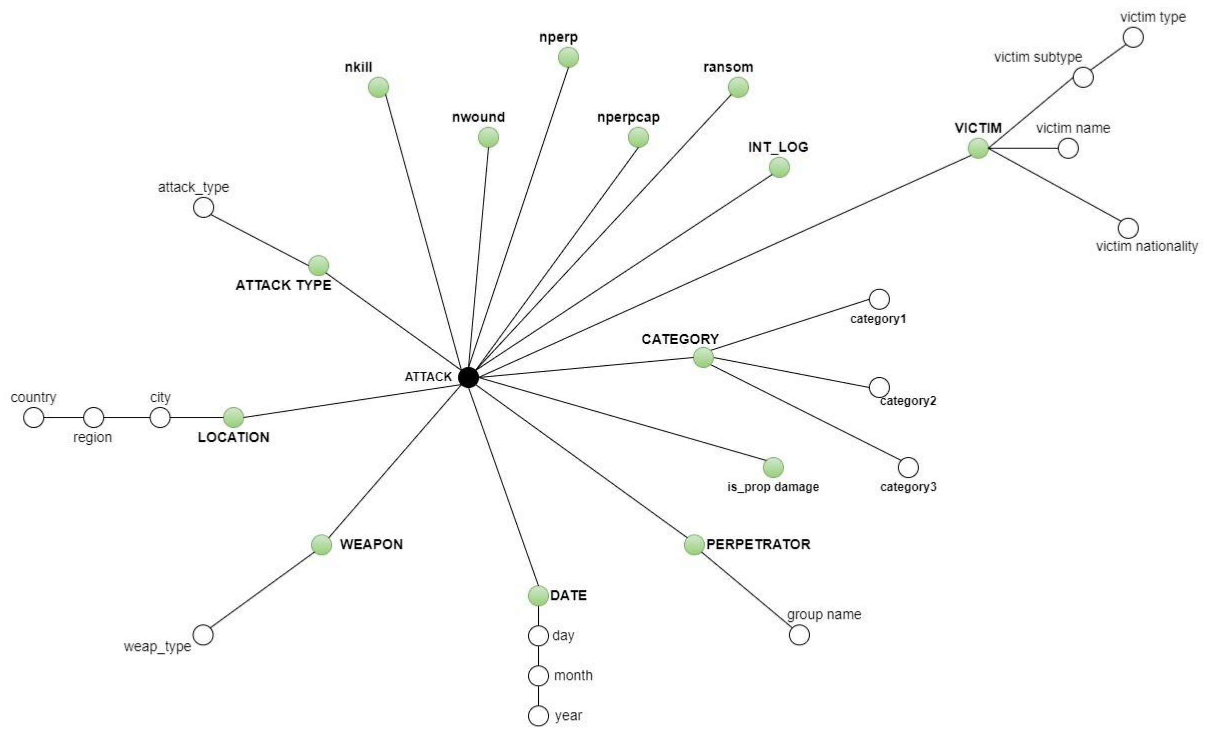


Figure 5. New structure of attribute tree

f) Fact schema

As a result of above modifications and operations on the Attribute Tree I created a final structure of a Fact Schema.

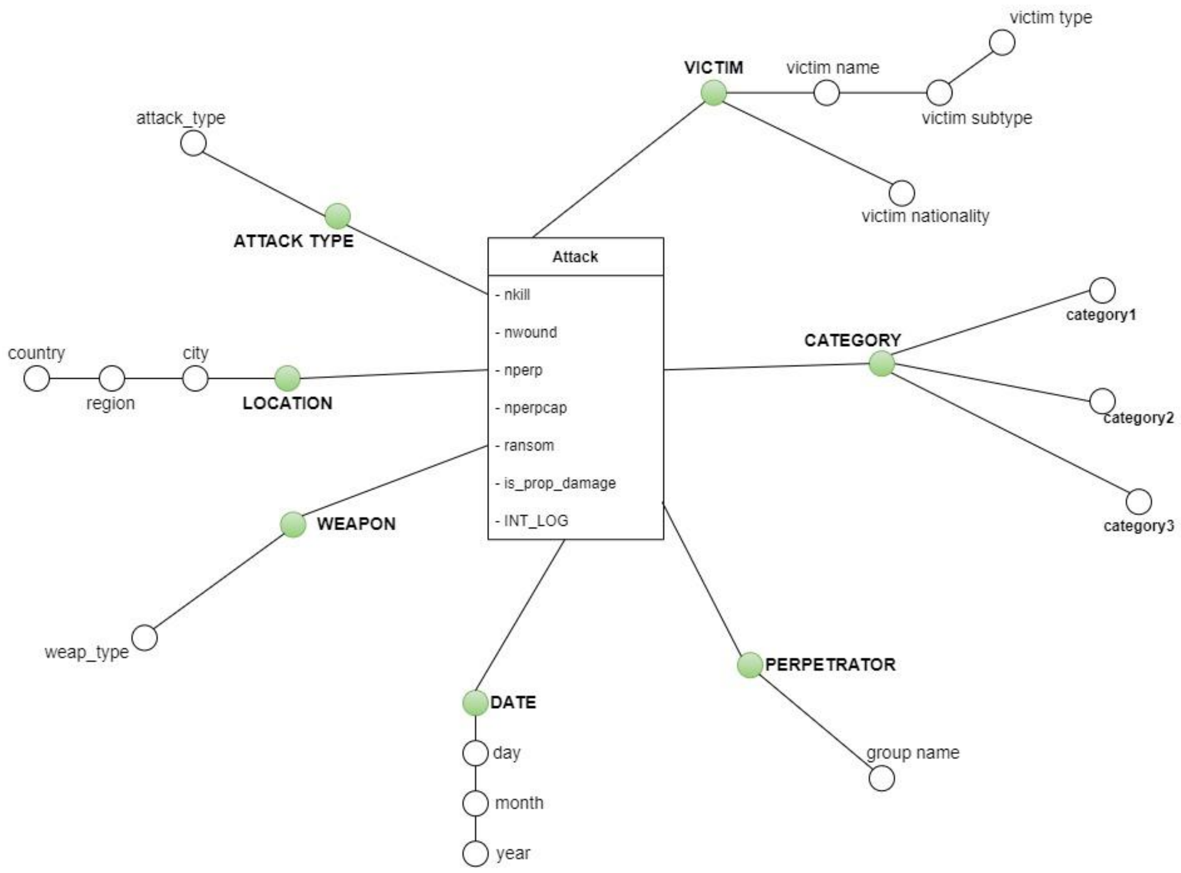


Figure 6. Fact schema

The fact table is called “Attack” and it contains measures which will be used later for calculations for visualizations.

4. Logical design

Having created the above fact schema, the next step of the project was about logical design. A DW logical data model describes the model in more detail compared to the DW conceptual data model. A data warehouse logical data model describes the data in as much detail as possible, this model does not describe how the model is implemented.

Based on previous conceptual models of the data, the best fitting schema for the project is the star schema. The center of the star has one fact table (attack) and around it there is a number of associated dimension tables (attack_type, location, weapon, date, perpetrator, category,victim). Every dimension in a star schema is represented with the only one-dimension table. The dimension tables are joined to the fact table using a foreign keys and they are not joined to each other.

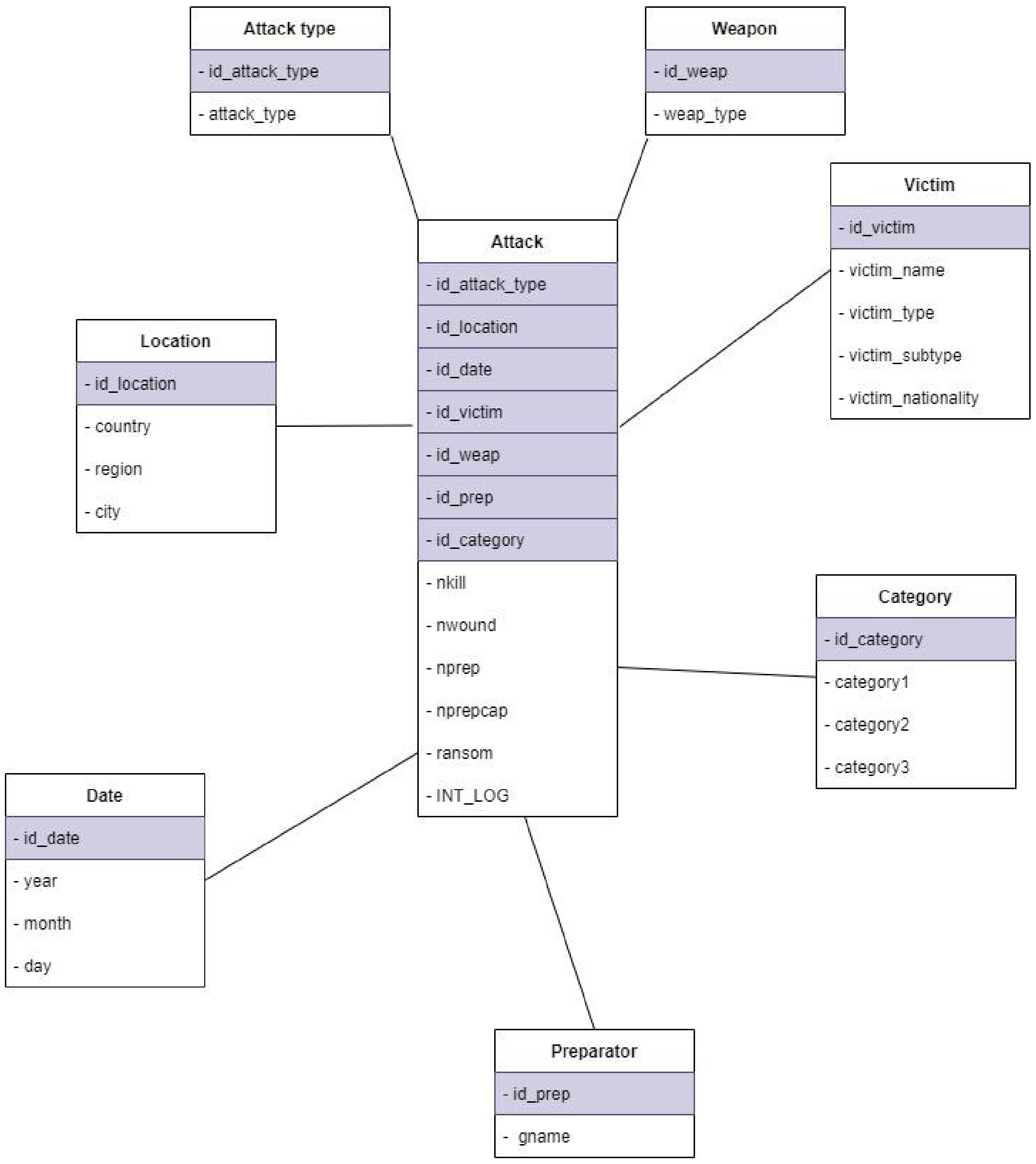


Figure 6. Star schema

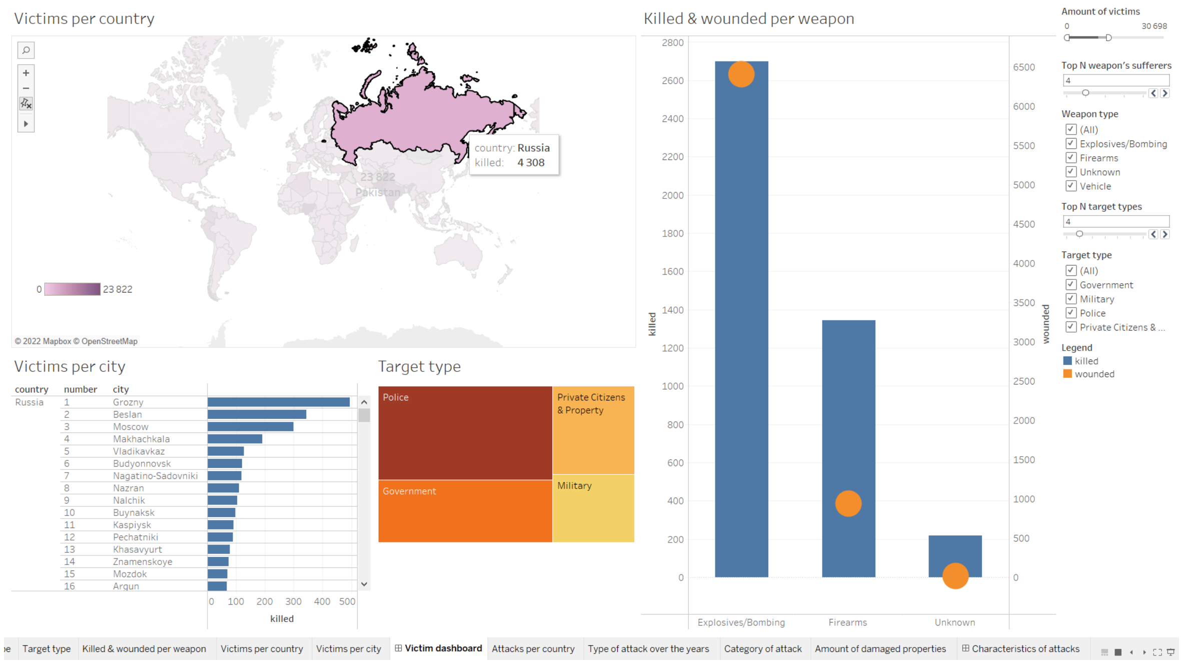
5. Visualization

As a last step of the project I used Tableau software to prepare visualizations from previously modified data.
I decided to create two dashboards, each of them containing four diagrams.

a) Victim characteristics dashboard

First dashboard is focused on victims that were killed or wounded during terrorist attack all over the world.

It allows searching through countries based on the specified number of victims. Then after choosing a particular country, it allows to check how many victims were in cities of this country (in descending order), how is the ratio of the different types of victims and how many victims were killed and wounded by specific type or types of weapon.



country	number	city
Russia	1	Grozny
	2	Beslan
	3	Moscow
	4	Makhachkala
	5	Vladikavkaz
	6	Budyonnovsk
	7	Nagatino-Sadovniki
	8	Nazran
	9	Nalchik
	10	Buynaksk
	11	Kaspiysk
	12	Pechatniki
	13	Khasavyurt
	14	Znamenskoye
	15	Mozdok
	16	Argun

Amount of victims: 0 – 30 698

Top N weapon's sufferers: 4

Weapon type: ☑ (All) ☑ Explosives/Bombing ☑ Firearms ☑ Unknown ☑ Vehicle

Top N target types: 4

Target type: ☑ (All) ☑ Government ☑ Military ☑ Police ☑ Private Citizens & ...

Figure 7. Victim characteristics dashboard

b) Characteristics of attacks dashboard

Second dashboard is focused on the event of the attack itself.
It allows us to analyze how a particular type or types of attack has been changing over years in a specified country (or in general over the world if nothing is specified).
In addition it allows us to check the ratio of the different categories of attacks and how many properties were damaged during an attack in a specified country.
It also allows to perform above analysis only in a particular range of years or in particular range of number of attacks.

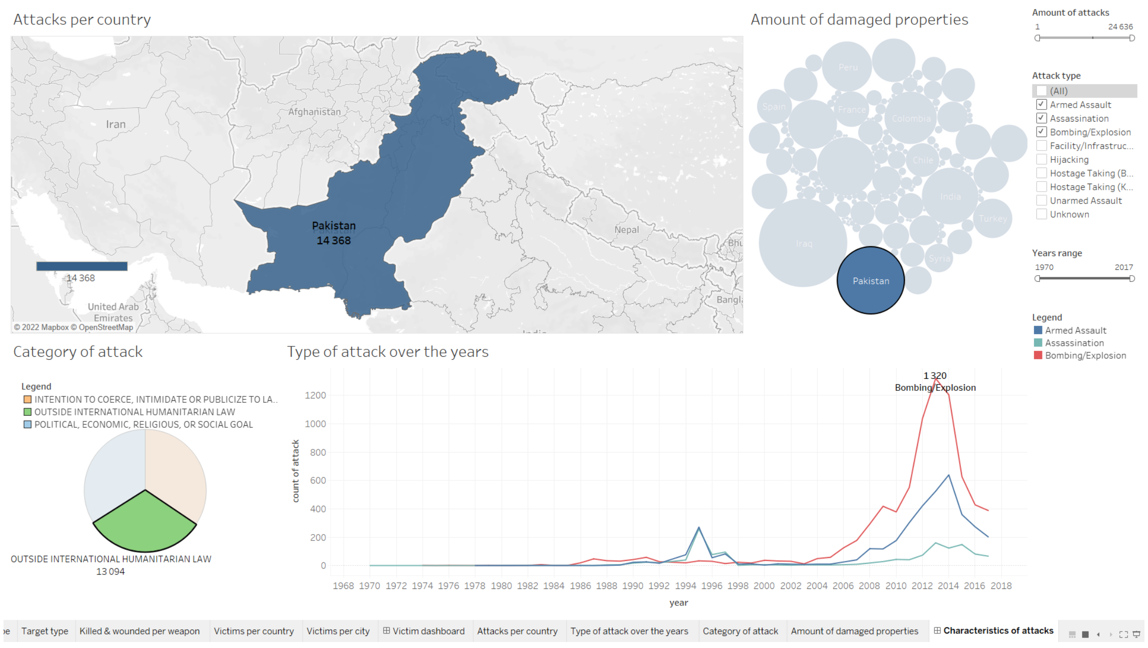


Figure 8. Characteristics of attacks dashboard