Safe Path Recommender based on Crime Statistics using Distributed Database

Ishank Agarwal, Parmeet Kaur and Chetna Dabbas

[ishankagarwal62@gmail.com](mailto:ishankagarwal62@gmail.com), [parmeet.kaur@jiit.ac.in](mailto:parmeet.kaur@jiit.ac.in), [chetna.dabas@jiit.ac.in](mailto:chetna.dabas@jiit.ac.in)

***Abstract*—This research aims to implement safe path using crime statistics of Delhi Ncr on a Google map through the combined use of distributed database concepts as well as using some machine learning algorithms. The application can be deployed as a web application and it is crucial for prediction and depiction of safe route using the information entered by the user in the source and destination fields.**

***Index Terms*—Distributed Database, MongoDb, Machine Learning, K-Means Clustering, Google Maps Api, Fault tolerance, NFS, SSH, MPICH2, Sharding**

1. **INTRODUCTION**

We are going to address the issue of safety in our project named ‘Safe Path Recommender’.

Delhi, national capital of India, is a hub for murder, kidnapping and abduction, juveniles in conflict and economic offences. Delhi tops the list of IPC crimes according to the National Crimes Records Bureau (NCRB). Among the top 19 cities, Delhi accounts for 38.8% of the crimes.

People walk on the streets but they are not safe, back from work, parties, late night dinners, parks, etc. and even coffee shops and different shopping or grocery stores during the day, children walk back from school and tuitions. Like any metropolitan city might have some spots as safe and others not, Delhi has them too. People should be able to judge whether a specific area or neighborhood is safe from possible crimes or not at a certain time of day. We have done this research on a web app which tells the safest route using crime data of Delhi Ncr to reach the desired destination SAFE!

**Our solution to the problem:**

Google Maps app give the fastest route between two points without even considering security but our web app in the research finds the safest route by considering the danger index of the possible paths.

1. **DATASETS RELATED WORK**

Dataset is obtained from the site "delhistats.com" and "datagov.in"(Open Government Data (OGD) Platform India). Out of the two datasets one contains data of Crimes in India, whereas the other contains the data of the Crimes in Delhi.

The reason we used a separate and "more detailed" dataset of Delhi is because of Delhi is also called the Crime Capital of India apart from being the National Capital. So, in this dataset we have crime data of all the sub-regions in Delhi like "Nehru Place, Karkardooma, etc...".

The **"Crimes in India"** contains :

* Crime data from 2001-2015 with 10,678 tuples and 33 attributes:
* Attributes of the dataset are:

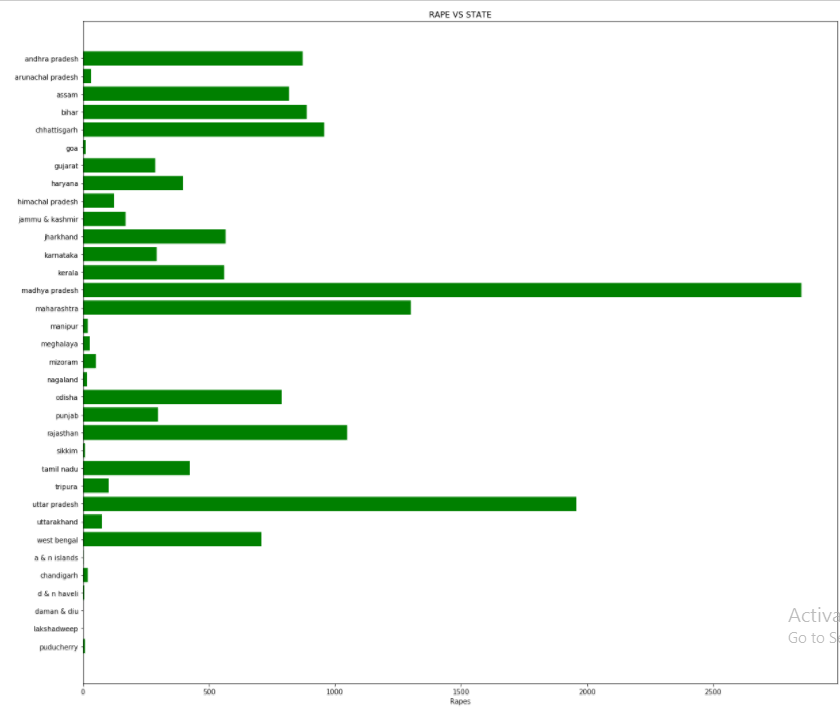
1. State/UT: Andhra Pradesh, Arunachal Pradesh, Assam, Bihar, Chhattisgarh, Goa, Gujarat, Haryana, Himachal Pradesh, J & K, Jharkhand, Karnataka, Kerala, Madhya Pradesh, Maharashtra, Manipur, Meghalaya, Mizoram, Nagaland, Odisha, Punjab, Rajasthan, Sikkim, Tamil Nadu, Tripura, Uttar Pradesh, Uttarakhand, West Bengal, A & N Islands, Chandigarh, D & N Haveli, Daman & Diu, Delhi UT, Lakshadweep, Puducherry.
2. District: Containing particular districts of where crimes were committed.
3. Year: containing year in which crimes were committed.
4. Murder: Murders committed.
5. Attempt to Murder
6. Culpable homicide not amounting to murder
7. Rape: Rapes committed.
8. Custodial Rape
9. Other Rape
10. Kidnapping & Abduction.
11. Kidnapping & Abduction of Girl & Women.
12. Kidnapping & Abduction of Others.
13. Dacoity
14. Preparation & Assembly for Dacoity
15. Robbery
16. Burglary
17. Theft
18. Auto Theft
19. Other Theft
20. Riots
21. Criminal Breach of Trust
22. Cheating
23. Counterfeiting
24. Arson
25. Hurt/Grievous Hurt
26. Dowry Deaths
27. Assault on women with intent to outrage her modesty
28. Insult to modesty of women
29. Cruelty by Husband or his Relatives
30. Importation of Girls From Foreign countries
31. Causing Death by Negligence
32. Other IPC Crimes
33. Total IPC Crimes: Mean: 5,310, Max: 2,72,423 crimes

in the year 2014.

The **"Crime in Delhi"** dataset contains:

* Crime Data of Delhi with 166 tuples and 14 attributes.
* Attributes are:

1. nm\_pol: name of region in Delhi.
2. murder
3. rape
4. gangrape
5. robbery
6. theft
7. assault Murders
8. totarea
9. totalcrime
10. long: longitude of the place of crime.
11. lat: latitude of the place of crime.
12. crime/area: ratio of the number of crimes occurred to the area affected ie. "region".
13. area: area of the particular region.



* **Fig. 1.** Graph showing distribution of Rapes committed vs States Clearly shows Madhya Pradesh being the highest victim of crimes.

To test the app, we further created “**Dummy**” Crime data for **Noida** and **Gurgaon** as well. Now, the research is done considering whole India and Delhi, Gurgaon and Noida in specific where crime rates have been alarmingly high, as compared to the other states.

1. **SYSTEM SETUP**

* For the research a web application was deployed on **Cloud 9,** provided by AWS- amazon web services. We have made a beautiful front-end mainly using bootstrap 3.7 and jQuery. For the backend, we mainly used **Nodejs** and dependencies like **express** and **Ejs** using npm (node package manager). Everything from the front to backend, including server , was done on Cloud 9.
* Distributed Database**: MongoDB** is an open-source program which has a documented-oriented database. It use json-like documents with schemas.
* Connection to Database: Mongoose
* Firing Queries with : PyMongo
* We have divided the backend into a distributed database into 4 nodes representing 4 zones of Indian map - "East, West, North, South", with "**North**" being the "**Master Node**" and rest of the regions being the "**Slave Nodes**".
* Using 7 virtual machines and ubuntu as a platform, we created a cluster comprising of **3 configuration servers** and **1 master node** and generated **3 replica sets**. The 3 replica sets acting as slave nodes for the master node are responsible for holding the database for the cluster. The 3 configuration servers are responsible for sharding. Thus, we generated **7 OS.**



* **Fig. 2.** Figure showing Master (top-left), 3Slaves(left below master) and 3 Config Servers(right) making 7 OS running on Windows virtually.
* Network configuration between the masters, slaves, and configuration servers:

(Master): “/etc/network/interfaces” file changes

auto lo

iface lo inet loopback

auto enp0s3

iface enp0s3 inet dhcp

auto enp0s8

iface enp0s8 inet static

address 192.168.100.100

netmask 255.255.255.0

network 192.168.100.0

broadcast 192.168.100.255

gateway 10.0.2.15

(Slaves): address 192.168.100.101, 102, 103 respectively.

(Configuration Servers): address 192.168.100.104, 105, 106 respectively.

* All Config Servers are running on port no. 27019 using mongod –configsvr command.
* Master is running on port no. 27020 in connection with the config servers using mongos --configdb 192.168.100.104:27019, 192.168.100.105:27019, 192.168.100.106:27019 --port 27020 command.
* Slaves are running on port no. 27018 using mongod –shardsvr command.
* The whole setup involves a crucial authentication feature which is **SSH** passwordless login in order to use Linux to automate the tasks. Therefore we need an automatic login from host A / user a to Host B / user b. We don’t want to enter any passwords, because we want to call ssh from a within a shell script.
* We also used shared **NFS** folder(named /mirror) for **fault tolerance** in replica sets thus changing the default file system of NTFS to NFS. So, if in any case, any node fails such like system breakdown, then we can access any process using replica sets i.e. slave nodes.
* /etc/exports file configuration in ‘Master node’:

/mirror \*(rw,sync)

* NFS kernel server was started using /etc/init.d/nfs-kernel-server start command
* All the slaves were mounted to the Shared Folder using Mount 192.168.100.100:/mirror /mirror command.
* To verify the same we created a file in the master node in the shared folder, as expected the same file was present in all the Slave Shared Folders.
* **MPICH** (high performance and widely portable implementation of the Message Passing Interface (MPI) standard), is used to optimize our shared NFS Folder.

/mirror folder configurations:

* ./configure --prefix=mirror/mpich2 --disable-f77 --disable-fc
* export PATH=/mirror/mpich2/bin:$PATH
* export LD\_LIBRARY\_PATH=”/mirror/mpich2/lib:$LD\_LIBRARY\_PATH”
* /etc/environment file configuration:

add “/mirror/mpich2/bin:”

* Hosts file configuration:

192.168.100.100

192.168.100.101

192.168.100.102

192.168.100.103

1. **METHODOLOGY & FEATURES**

**K-Means Clustering**: k-means clustering is a type of unsupervised learning which involves vector quantization, for cluster analysis in data mining. This algorithm aims to group data with the number of groups represented by the variable k by partitioning n observations into k clusters. Each observation belongs to the cluster with the nearest mean, which is the prototype of the cluster based on the features which defines the resulting groups

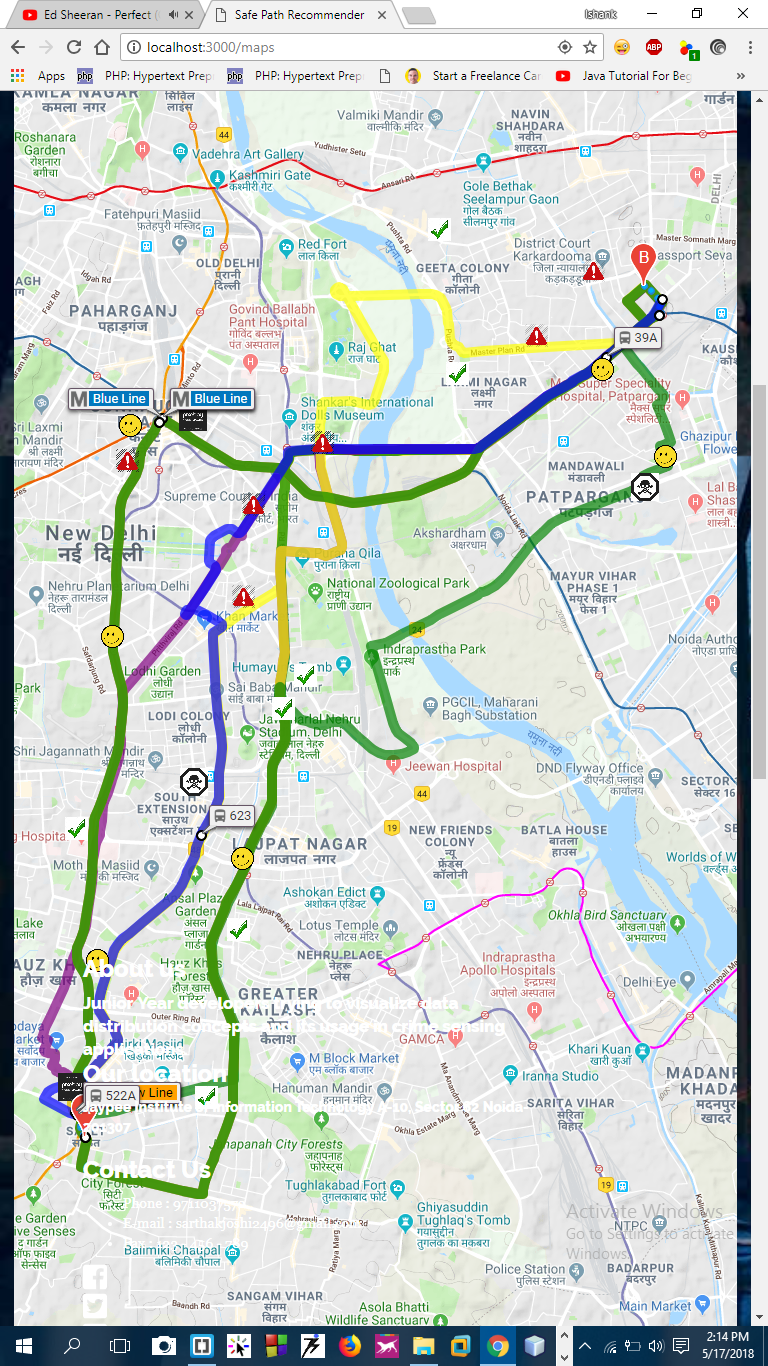
* We have applied above Unsupervised Machine Learning Algorithm, K-means clustering to find danger index of the many possible paths between two places (starting and destination locations) assigning a magnitude of criminal activities on the map of Delhi Ncr (Delhi, Noida & Gurgaon) in range of **0 to 4**. We used Google map and Google Places Apis to display all possible routes between any two locations A & B.

1. Firstly, Danger Index of the different possible paths to a destination has been calculated to compare the safest paths out of the total paths considered.
2. An autocomplete feature in the search boxes of the starting and destination locations is introduced so that user does not have to type the whole address and the drop down list emerges as other alternative suggestions for a user to pick from a varied list of places.
3. The functionality of the choice for the user to pick from different mode of travel has been introduced ie. Walking, Transit and Driving.
4. Markers have been placed on all the possible paths to indicate the Danger Index associated with the path ie. Smiley: safest, Green Tick: moderately safe, Exclamation Mark Triangle: be careful, Skull: moderately dangerous and Cross: extremely dangerous.
5. Statistics of the Safety of paths in the form of detailed information is output below the map so that user also knows what all things he has to consider while making his choice to make smart decisions like Distance, Time Duration, Danger Index, Route Number, and Color of the Route.
6. We have divided the backend using MongoDB into a distributed database into 4 nodes representing 4 zones of Indian map - East, West, North and South.
7. Having a distributed database gives us the advantage of fault tolerance over regular databases, in case of failure of a node, other nodes can still access the crime data which is possible because of replica sets.
8. In the “Crime in Delhi” dataset we have used **Google Maps Reverse Geocoding Api** using the longitudes and lattitudes from the dataset to be more precise. In the “Crime in India” dataset we have used **Google Maps Api** using the names of the districts. Google Apis used:

* Google Maps JavaScript API
* Google Maps Embed API
* Google Maps Directions API
* Google Maps Geolocation API
* Google Places API Web Service:

1. **RESULTS**

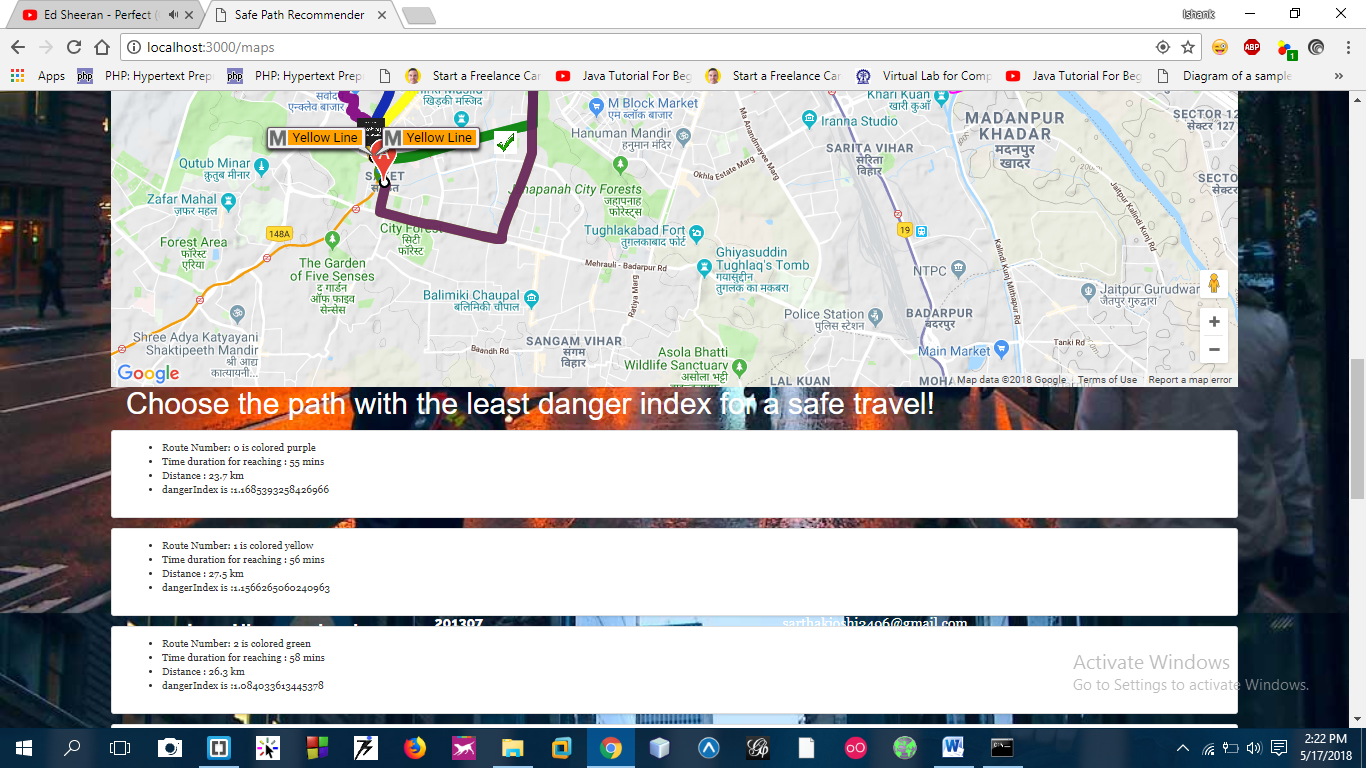
* Web App is able to display different possible paths with different colors and markers indicating danger level of a place which falls on that route.
* The figure below is showing the path shown between “Saket, New Delhi” and “Karkardooma, New Delhi” via Driving mode.



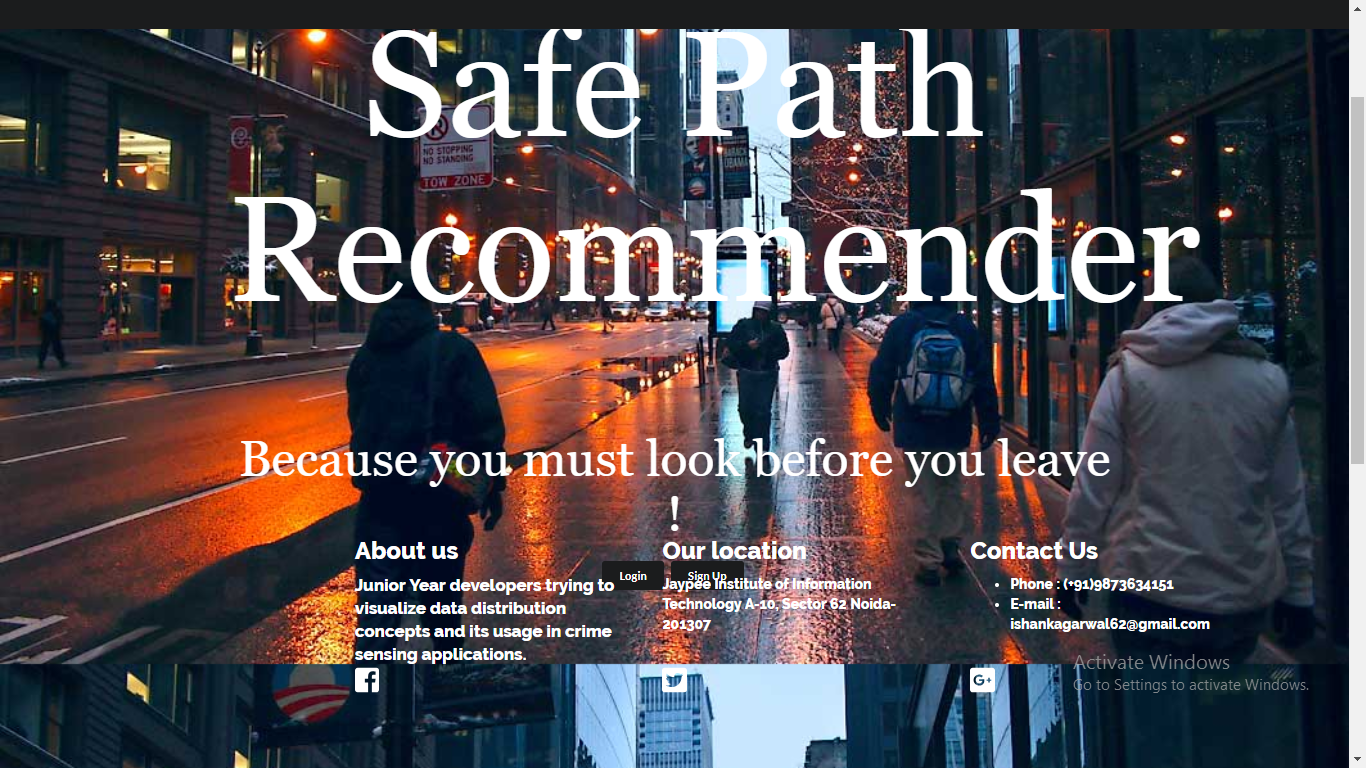
* **Fig. 4.** Web App Showing Safe Path between “Saket, New Delhi, India” to “Karkardooma, New Delhi, India” with the different markers (Skull, Exclamation Mark Triangle, Smiley, Green Ticks, etc) and different routed each colored differently.



* **Fig. 5.** This is how the Web App looks like while recommending Safe Paths.
* The k-means algorithm assigns 0-4 magnitude of crime level to all locations in Delhi Ncr (Delhi, Noida and Gurgaon).
* The Danger Index of ‘0’ implies that the place considered is relatively much safe with less crime records in past while an index of 4 means that the place has high crime records in the past.
* The figure below is showing detailed information of the Safety of paths below the map so that user also knows what all things he has to consider while making his choice to make smart decisions like Distance, Time Duration, Danger Index, Route Number, and Color of the Route.



* **Fig. 6.** Figure showing the statistics of the recommended 3 Safe Paths like its Color, Time Duration for the respective path, Distance of the path, Danger Index (according to which the priority of listing of paths is set).



* **Fig. 7.** Homepage of the app.

1. **FUTURE SCOPE**

For the final version of the research, we are planning to work further on the distributed database part by optimizing it to handle even live streaming data, like we will also try to add a feature in the app so as to get **live crime data from social media sites** and **inform the user about the incidents occurring around him while travelling** in the form of “in-app notifications”. We will aim on improving the machine learning algorithms we have used to optimize the safe route web application.

**VIII.REFERENCES**

[1] “Crimes chicago - data portal.” [Online]. Available: https://data. cityofchicago.org/public-safety/crimes-2001-topresent/ijzp-q8t2.

[2] “Crimes philadelphia-opendata,” 2.On- line]. Available: http://www.opendataphilly.org/opendata/resource/215/philadelphiapolice-part-one-crime-incidents.

[3] “Openstreetmap.” [Online]. Available: http://www.openstreetmap.org.

[4] “Osm metro extracts.” [Online]. Available: http://metro.teczno.com.

[5] [Online]. Available: https://github.com/tristramg/osm4routing.

[6] S. J. Ratcliffe, “Predicting crime using analytics and big data.” Wiley, 2005. [Online]. Available: http://tinyurl.com/pjg3v68.

[7] S. Chainey, L. Tompson, and undefined S. Uhlig, “The utility of hotspot mapping for predicting spatial patterns of crime,” In Security Journal, vol. 21, pp. 4–28, 2008.

[8] Haroun Rababaah, “Distributed Databases Fundamentals And Research”, Advanced Database – B561. Spring 2005.

[9] Kaiji Chen, Yongluan Zhou, and Yu Cao, “Online Data Partitioning in Distributed Database Systems”.

[10] Umut Tosun, “Distributed Database Design: A Case Study”.