CS4221/CS5421 Fake Data Generator

Demirkirkan Ege Liu Zimu Yang Jiyu
Mathivathana Paramanthan May Htet Htet Hlaing

April 2022

Abstract

Data is crucial in many aspects of technology and commercial operations, however owing to privacy concerns, certain types of data will not be given for testing or training when specific apps are deployed. In this project, relationship types, probability distribution types, and cardinality restrictions are used to generate fictitious but realistic data from an entity and it's relationship given in the form of User Commands. The output data may be utilized in either sql or csv formats.

1 Introduction

Fake Data Generator is a tool to generate realistic random data. The primary goal of Fake Data Generator is to not only generate realistic data, but also focus on the relationships between tables in terms of PRIMARY KEYs, FOREIGN KEYs, participation and distribution constraints. All of the aforementioned aspects are addressed in this project.

Motivation. There are many use cases for real data; for instance, students training in a Database course, app developers that can only test their applications on realistic data or even management presentations. However, as privacy is of utmost importance, certain types of data will unlikely be released for training or testing purposes. This types of data could be people's personal IDs, home addresses and even credit card numbers. Hence, this brings about the need for fake but realistic data. Fake data is credible and consistent to real data. It must also be able to mimic certain distributions and relationships found in real data.

Currently there are tools available online, but are lacking in certain areas. This will be discussed further below. Therefore, we have embarked on this project to create a tool to help us generate realistic data.

Problem Statement. Design and implementation of a tool that generates realistic random data for an entity-relationship design considering participation constraints, probability distributions, and joint probability distributions.

Outline of Solution. The system begins its work by collecting user requirements from command line and converts them into schema. Based on the schema,

data is retrieved from Mockaroo and applied probability distribution when required. After performing foreign key constraints and joint probability distribution on data, random realistic data is generated as output for the user.

Summary of Contribution. We have created a tool to generate fake data. The main features are as follows:

1. Fake but realistic data.
2. Data generation that can adhere to participation and cardinality constraints as defined by the user.
3. Data generation that can adhere to uniform, normal, exponential and binomial distributions on single attribute or multiple attributes as defined by the user.
4. Data generation that adheres to the user-defined joint probability distribution.

Plan of Paper. The remainder of this article is structured as follows. Section 2 presents the necessary background on the participation constraints, probability distributions and joint probability distributions. Section 3 discusses some of the existing and related tools online currently. Section 4 presents our methodology and solution to the problem. Section 5 presents our proven results.Section 6 narrates our conclusion.

2 Background

This section presents the background (with references to textbook and other sources) necessary to understand the report.

1. Probability Distributions:
    - Uniform distribution
      The uniform distribution describes an experiment where the distribution support is equally probable withing the bounds of its parameters, a and b [2] as seen in 15 and 16.
    - Gaussian/Normal Distribution
      The parameter μ is the mean or expectation of the distribution (and its median and mode), while the parameter α is its standard deviation [4] as seen in 17 and 18.
    - Exponential distribution
      Exponential distribution is the probability distribution of time between the events in a Poisson point process with mean λ. The Poisson processes are events which occur continuously and independently [4] as seen in 19 and 20.
    - Binomial distribution
      Binomial distribution is the discrete probability of the number of successes in a sequence of n independent trials, each with its own success/failure outcome with probability p of success [3] as seen in 21 and 22.

- Geometric distribution
  Geometric distribution is a discrete probability distribution that represents the probability of the number of successive failures before a success is obtained in a success/failure trial [5].

2. Joint distributions:
   Probability distribution of two random variables on the same probability space [3]. An example is the drawing of a ball from 2 different urns. The probability of drawing a red ball is 2/3 and a 1/3 for a blue ball for both urns. The joint probability distribution is given as:

	A=Red	A=Blue	P(B)
B=Red	(2/3)(2/3)=4/9	(1/3)(2/3)=2/9	4/9+2/9=2/3
B=Blue	(2/3)(1/3)=2/9	(1/3)(1/3)=1/9	2/9+1/9=1/3
P(A)	4/9+2/9=2/3	2/9+1/9=1/3	

3. Multiple distribution column:
   A single column has multiple distributions based on a separate key. For instance, we can have a salary column dependent on the job column. Each salary will correspond to a different distribution based on their job title.

4. Participation and cardinality constraints:

   - Total Participation constraint
     The entity must participate in the relationship to exist. An example is that a student can also exist when it is enrolled in a school.
   - Partial participation constraint
     The entity can exist without participating in a relationship with another entity. An example is that a course can exist even without students' participation.
   - Cardinality constraints refer to the minimum and maximum number of instances an entity can be associated with another entity. Some of the cardinalities can be generalised to One to one, Many to one/ one to many and Many to many.

3 Related Work

During the course of our research, we came across multiple data generation tools online. Their pros and cons will be discussed in this chapter.

1. Mockaroo
   Pros
   Mockaroo allows users to generate limited amounts of data daily through different tools [6].

   - Provides a broad range of data types including cities, credit card numbers, phone numbers.
   - Ability to mock Normal and Poisson distributions.
   - Ability to mock same data across 2 tables.

Cons

- Inability to mock participation and cardinality constraints.
- Inability to mock foreign key constraints.

2. DATPROF
   Pros
   DATPROF provides many more services than those listed here but it is a paid service [1].

   - Provides a broad range of data types including cities, names, emails.
   - Can generate data from scratch or generate more data from an existing database.
   - All existing relationships between tables remain unchanged, including foreign key constraints.
   - Ability to mock foreign key constraints.

   Cons

   - Paid service.

3. snowflake
   Pros
   snowflake also provides more services but is a paid service as well [7].

   - Provides a broad range of data types including cities, names, emails.
   - Ability to mock Normal and uniform distributions.

   Cons

   - Paid service.
   - Inability to mock foreign key constraints.

4 Methodology

The methodology that was used is depicted in the flow diagram Figure 1 as mentioned below.

We acquired information for an ER Diagram that the user wishes to portray via a command-line interface. In the first phase, the User will provide information such as TABLE NAME, ATTRIBUTE NAME, TYPE, PRIMARY KEY, PARTICIPATION , CARDINALITY CONSTRAINT, PROBABILITY DISTRIBUTION AND JOINT PROBABILITY DISTRIBUTION. Then We handle the command line through a series of steps after receiving confirmation from the user that their input procedure is completed.Following the User's approval, the next step is to map the underlying commands to a table structure using the information provided.

Then, using the details about columns and their types from the table schema generated, integration has been made with Mockaroo API using JAVA .We chose Mockaroo because of its extensive support for different kinds of column.Data can be generated in the same way that it is generated on the website using the

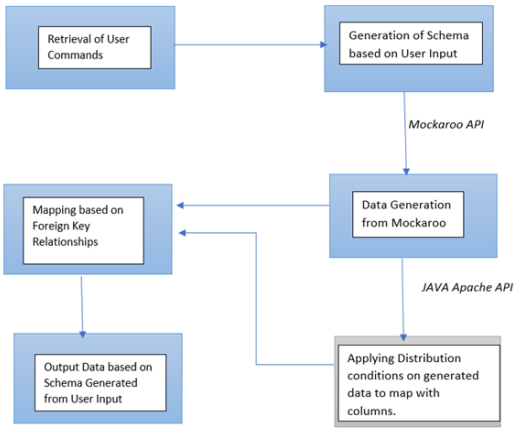


Figure 1: Flow diagram of Fake Data Generator

Mockaroo API.Certain validations have been performed in the User Commands to ensure a successful integration with Mockaroo. As Mockaroo supports certain list of column types, the corresponding column attribute type, such as "First Name," "Full Name," "Gender," "IBAN," "IP Address V4"...must be matched with the User stated attribute given at the beginning.

From the Mockaroo end, random data has been generated with the specified count based on the data passed regarding column and types in the form of JSON Array along with the row count.

Following the Mockaroo section, it proceeds to Distribution Computation or mapping foreign keys with Joint Probability Distribution (for tables with relation) or immediately to data file production (for tables without relation), as shown in the flow chart.The distribution that the user required for the given column has been applied in the Distribution Part with the JAVA APACHE API.This stage dealt with the Uniform, Normal, Exponential, Binomial, and Geometric Distributions.We chose the JAVA APACHE API since it supports a diverse spectrum of distributions. As a result of the distribution supplied, the column data has been updated and will be combined with the final table data.The output file will then be generated in both .sql and.csv formats for user convenience.

5 Case or Evaluation

5.1 Set-up

Phase 1[User Phase] Retrieval of User Commands from the Command Line Interface is the subject of this phrase. The input must contain the following data based on the User need to construct,

- one independent entity: PRIMARY KEY,ATTRIBUTE NAME,TYPE,TABLE NAME (for single table)
  - SYNTAX_PROVIDED :

        * CREATE ENTITY teacher WITH t_id Row_Number, t_name First_Name, t_salary Number PRIMARY KEY t_id

- multiple independent entities: PRIMARY KEY,ATTRIBUTE NAME,TYPE,TABLE NAME (for multiple tables based on User Input)
  - SYNTAX_PROVIDED :
    * CREATE ENTITY teacher WITH t_id Row_Number, t_name First_Name, t_salary Number PRIMARY KEY t_id
    * CREATE ENTITY student WITH s_id Row_Number, s_name First_Name, s_lname Last_Name PRIMARY KEY s_id

- multiple entities with Relationship:
  - PRIMARY KEY,ATTRIBUTE NAME,TYPE,TABLE NAME(TABLE1).
  - PRIMARY KEY,ATTRIBUTE NAME,TYPE,TABLE NAME(TABLE2).
  - PARTICIPATION AND CARDINALITY CONSTRAINT.
    * SYNTAX_PROVIDED :
      · CREATE ENTITY teacher WITH t_id Row_Number, t_name First_Name, t_salary Number PRIMARY KEY t_id
      · CREATE ENTITY student WITH s_id Row_Number, s_name First_Name, s_lname Last_Name PRIMARY KEY s_id
      · CREATE RELATION student-teacher FOR student WITH PARTICIPATION MIN 0 MAX * AND teacher WITH PARTICIPATION MIN 1 MAX * ATTRIBUTES date Datetime

- one independent entity with distribution:
  - PRIMARY KEY,ATTRIBUTE NAME,TYPE,TABLE NAME(TABLE1).
  - DISTRIBUTION WITH PARAMETERS.
    * SYNTAX_PROVIDED :
      · CREATE ENTITY teacher WITH t_id Row_Number, t_name First_Name, t_salary Number PRIMARY KEY t_id
      · CREATE UNIFORM_INTEGER DISTRIBUTION WITH PARAM1 2000 PARAM2 5000 FOR ATTRIBUTE salary IN TABLE teacher

- multiple entities with joint probability relation:
  - PRIMARY KEY,ATTRIBUTE NAME,TYPE,TABLE NAME(TABLE1).
  - PRIMARY KEY,ATTRIBUTE NAME,TYPE,TABLE NAME(TABLE2).
  - PARTICIPATION AND CARDINALITY CONSTRAINT.
  - JOINT PROBABILITY DISTRIBUTION CONSTRAINT.
    * SYNTAX_PROVIDED :
      · CREATE ENTITY teacher WITH t_id Row_Number, t_fname First_Name, t_lname Last_Name PRIMARY KEY t_id
      · CREATE ENTITY student WITH s_id Row_Number, s_name First_Name, s_lname Last_Name PRIMARY KEY s_id

· CREATE RELATION student-teacher FOR student WITH PARTICIPATION MIN 0 MAX 1 AND teacher WITH PARTICIPATION MIN 0 MAX * ATTRIBUTES date Datetime
· CREATE JOINT PROBABILITY DISTRIBUTION WITH MEAN 4 SD 1 FOR TABLE teacher

- one independent entity with multiple column distribution
  - PRIMARY KEY,ATTRIBUTE NAME,TYPE,TABLE NAME(TABLE1).
  - MULTIPLE COLUMN DISTRIBUTION WITH PARAMETERS
    * SYNTAX_PROVIDED
      · CREATE ENTITY person WITH id Row_Number, level Number, name First_Name, salary Number PRIMARY KEY id
      · CREATE UNIFORM_INTEGER DISTRIBUTION WITH PARAM1 1 PARAM2 4 FOR ATTRIBUTE level IN TABLE person
      · CREATE MULTIPLE COLUMN NORMAL DISTRIBUTION WITH PARAM1 10000 PARAM2 2000 WHERE ATTRIBUTE level 1 PARAM1 5000 PARAM2 1000 WHERE ATTRIBUTE level 2 PARAM1 2000 PARAM2 500 WHERE ATTRIBUTE level 3 PARAM1 1000 PARAM2 200 FOR ATTRIBUTE salary IN

We additionally include the following commands for user convenience:

- help - to list all the available commands
- SHOW ENTITIES - to list all the Entities created in TABLE Format
- SHOW RELATIONS - to list all the Relations given in TABLE Format by mentioning their respective Participation constraint
- SHOW DISTRIBUTIONS - to list all the Distributions given in TABLE Format by mentioning their respective Column and Table Name
- REMOVE ENTITY entity_name - to allow the User to remove the Entity
- REMOVE RELATION relation_name - to allow the User to remove the Relationship given
- GENERATE DATA ROW_COUNT - to allow the User to confirm that he has done with all his inputs.

5.2 Results

Phase 2 [Output Phase]

Phase 2.1 [Output From Command Line Interface] This phase takes place prior to the user's confirmation (i.e. the Generate Data Command).This phase assists the user in cross-verifying their formed entity, its participation, and the distribution offered, as well as removing any entities found to be incorrectly distributed.

Commands and Respective Outputs : In the Command Line Interface, the application begins by showing the following command to the user.

```
Enter a command, all available commands can be seen using 'help' command
```

Figure 2: Initiative Command Displayed on Command Line Interface

- help - to list all the available commands

```
Enter a command, all available commands can be seen using 'help' command
help
```

Figure 3: Help Command

- CREATE ENTITY - to create entity with the table name,attribute with its type and primary key

```
Enter a command, all available commands can be seen using 'help' command
CREATE ENTITY teacher WITH t_id Row_Number, t_name First_Name, t_salary Number PRIMARY KEY t_id
Entity is successfully created
```

Figure 4: Create Entity Command

- CREATE RELATION - to describe the relation between entities in terms of Participation and Cardinality Constraint .

  Constraint Comprises of,

  - One - One = (1,1)
  - One - Many
    - Total - Partial = (1,1),(0,*)
    - Partial - Partial = (0,1),(0,*)
  - Many - One
    - Partial - Total = (0,*),(1,1)
    - Partial - Partial = (0,*),(0,1)
  - Many - Many
    - Total - Total = (1,*),(1,*)
    - Total - Partial = (1,*),(0,*)
    - Partial - Total = (0,*),(1,*)
    - Partial - Partial = (0,*),(0,*)

  In Command Line, the Participation and Cardinality constraint has been mentioned by means of MIN and MAX Keyword.

```
Enter a command, all available commands can be seen using 'help' command
CREATE RELATION student-teacher FOR student WITH PARTICIPATION MIN 0 MAX * AND teacher WITH PARTICIPATION MIN 1 MAX * ATTRIBUTES date Datetime
Relation is successfully created
```

Figure 5: Create Many-Many Relation Command

```
Enter a command, all available commands can be seen using 'help' command
CREATE RELATION student-teacher FOR student WITH PARTICIPATION MIN 0 MAX 1 AND teacher WITH PARTICIPATION MIN 1 MAX * ATTRIBUTES date Datetime
Relation is successfully created
```

Figure 6: Create One-Many Relation Command

- SHOW ENTITIES - to list the Entities Created in Tabular Schema with Row , Column Structure
- SHOW RELATIONS - to list the Relations Created in Tabular Schema with Row , Column Structure

```
Enter a command, all available commands can be seen using 'help' command
show entities
+------------------------------+--------------------------+--------------------------+
|                                student (ENTITY)                                    |
+------------------------------+--------------------------+--------------------------+
|    s_id (Row_Number) (PK)    |    s_name (First_Name)   |    s_lname (Last_Name)   |
+------------------------------+--------------------------+--------------------------+

+------------------------------+--------------------------+-------------------------+
|                               teacher (ENTITY)                                     |
+------------------------------+--------------------------+-------------------------+
|    t_id (Row_Number) (PK)    |    t_name (First_Name)   |    t_salary (Number)    |
+------------------------------+--------------------------+-------------------------+


Enter a command, all available commands can be seen using 'help' command
show relations
+------------------------------+----------------------------------+--------------------------+
|      student (ENTITY)        |    student-teacher (RELATION)    |     teacher (ENTITY)     |
+------------------------------+----------------------------------+--------------------------+
|    Partial Participation     |             One-Many             |    Total Participation   |
+------------------------------+----------------------------------+--------------------------+
|                                  date (Datetime)                                        |
+------------------------------+----------------------------------+--------------------------+
```

Figure 7: Show Entities and Show Relations Command

- GENERATE DATA ROW_COUNT - to get confirmation that User has done with his/her input and to retrieve the no. of rows from User .

```
Enter a command, all available commands can be seen using 'help' command
generate data 100
Your csv files are generated. You can find them on the output folder
Goodbye...
```

Figure 8: Generate Data Command

Phase 2.2 [Output From .sql and .csv file] For the Random Data Generation and Applying Distribution in the needed columns, the Generate Data Command has pushed the data from Phase 2.1 to Phase 2.2.

- INPUT [**SINGLE ENTITY**] -
  CREATE ENTITY student WITH s_id Row_Number, s_name First_Name, s_lname Last_Name PRIMARY KEY s_id
- OUTPUT :
  Output result is shown in Figure 9 and 10.

```sql
insert into  (s_id, s_name, s_lname) values (1, 'Marysa', 'Dayton');
insert into  (s_id, s_name, s_lname) values (2, 'Lynnet', 'Lebourn');
insert into  (s_id, s_name, s_lname) values (3, 'Joella', 'Alsopp');
insert into  (s_id, s_name, s_lname) values (4, 'Freda', 'Dooney');
insert into  (s_id, s_name, s_lname) values (5, 'Skipper', 'Curbishley');
insert into  (s_id, s_name, s_lname) values (6, 'Deva', 'Ballin');
insert into  (s_id, s_name, s_lname) values (7, 'Hyacinthie', 'Sabater');
insert into  (s_id, s_name, s_lname) values (8, 'Sabina', 'Clemintoni');
insert into  (s_id, s_name, s_lname) values (9, 'Kerrill', 'Cricket');
insert into  (s_id, s_name, s_lname) values (10, 'Dot', 'Rawles');
```

Figure 9: .sql file with Random Data Generated for Single Entity

```csv
s_id,s_lname,s_name
1,Colby,Trent
2,Muddicliffe,Gilbert
3,Lacroutz,Kori
4,Askham,Doe
5,Darlington,Sadella
6,Mackneis,Kalil
7,Call,Fidela
8,Silwood,Corilla
9,Baiyle,Charla
10,Victor,Kelby
```

Figure 10: .csv file with Random Data Generated for Single Entity

- INPUT [**MULTIPLE ENTITY WITH RELATION - Many to Many (Partial-Total)**] -

CREATE ENTITY student WITH s_id Row_Number, s_name First_Name, s_lname Last_Name PRIMARY KEY s_id

CREATE ENTITY teacher WITH t_id Row_Number, t_name First_Name, t_salary Number PRIMARY KEY t_id

CREATE RELATION student-teacher FOR student WITH PARTICIPATION MIN 0 MAX * AND teacher WITH PARTICIPATION MIN 1 MAX *

- OUTPUT :
  As the given Input has Many-Many Condition , hence the third table has been generated with the two primary columns of corresponding tables.As Student table has Partial Involvement than Teacher Table, hence in the third table id's are mapped accordingly with all the id's from Teacher Table and Partial id's from Student table. The result is show in Figure 11.

Figure 11: .csv file with Random Data Generated for Many-Many (Partial-Total)

- INPUT [**MULTIPLE ENTITY WITH RELATION - One to One (Total-Total)**] -

  CREATE ENTITY student WITH s_id Row_Number, s_name First_Name, s_lname Last_Name PRIMARY KEY s_id

  CREATE ENTITY teacher WITH t_id Row_Number, t_name First_Name,

t_salary Number PRIMARY KEY t_id

CREATE RELATION student-teacher FOR student WITH PARTICIPATION MIN 1 MAX 1 AND teacher WITH PARTICIPATION MIN 1 MAX 1 ATTRIBUTES date Datetime

- OUTPUT :
  As One-One requires one common table to get generated.Hence here we had merged the entities given with One-One Relation.The columns and respective data has been given into the third table with either of it's Primary Key as the Key.The file get generated with the name of the relation mentioned by User as shown in Figure 12.

Figure 12: .csv file with Random Data Generated for One-One

- INPUT [**MULTIPLE ENTITY WITH RELATION - One to Many (Partial-Partial)**] -

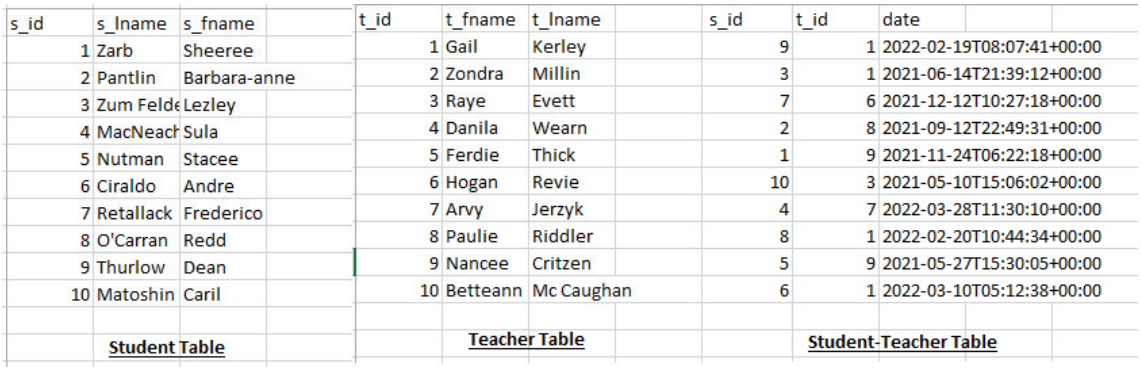
  CREATE ENTITY student WITH s_id Row_Number, s_name First_Name, s_lname Last_Name PRIMARY KEY s_id

  CREATE ENTITY teacher WITH t_id Row_Number, t_fname First_Name, t_lname Last_Name PRIMARY KEY t_id

  CREATE RELATION student-teacher FOR student WITH PARTICIPATION MIN 0 MAX 1 AND teacher WITH PARTICIPATION MIN 0 MAX * ATTRIBUTES date Datetime

- OUTPUT :
  Since the relation has One to Many (partial-partial) constraint, the primary key of left table that is student table serves as primary key of the

third table (relation table). As joint probability distribution is not applied in this relation, foreign keys are linked randomly. The result is depicted in Figure 13.



s_id	s_lname	s_fname
1	Zarb	Sheeree
2	Pantlin	Barbara-anne
3	Zum Felde	Lezley
4	MacNeach	Sula
5	Nutman	Stacee
6	Ciraldo	Andre
7	Retallack	Frederico
8	O'Carran	Redd
9	Thurlow	Dean
10	Matoshin	Caril

Student Table

t_id	t_fname	t_lname
1	Gail	Kerley
2	Zondra	Millin
3	Raye	Evett
4	Danila	Wearn
5	Ferdie	Thick
6	Hogan	Revie
7	Arvy	Jerzyk
8	Paulie	Riddler
9	Nancee	Critzen
10	Betteann	Mc Caughan

Teacher Table

s_id	t_id	date
9	1	2022-02-19T08:07:41+00:00
3	1	2021-06-14T21:39:12+00:00
7	6	2021-12-12T10:27:18+00:00
2	8	2021-09-12T22:49:31+00:00
1	9	2021-11-24T06:22:18+00:00
10	3	2021-05-10T15:06:02+00:00
4	7	2022-03-28T11:30:10+00:00
8	1	2022-02-20T10:44:34+00:00
5	9	2021-05-27T15:30:05+00:00
6	1	2022-03-10T05:12:38+00:00

Student-Teacher Table

Figure 13: .csv file with Random Data Generated for One-Many (Partial-Partial)

- INPUT [**MULTIPLE ENTITY WITH RELATION - Many to One (Partial-Total) and APPLY JOINT PROBABILITY DISTRIBUTION CONSTRAINT**] -

  Commands for creating student entity and teacher entity is the same as One to Many relation

  CREATE RELATION student-teacher FOR student WITH PARTICIPATION MIN 0 MAX * AND teacher WITH PARTICIPATION MIN 1 MAX 1 ATTRIBUTES date Datetime
  CREATE JOINT PROBABILITY DISTRIBUTION WITH MEAN 4 SD 1 FOR TABLE student

- OUTPUT :
  For Many to One (Partial-Total) relation, there are two tables. The left table (student table) is an independent table. Since right entity must participate exactly once, right table that is teacher table is combined with relation table. Foreign keys (s_id) are linked according to joint probability distribution defined by user. A Gaussian probability distribution with a user-defined mean and standard deviation is used to compute the joint probability distribution. Each estimated probability distribution is used to determine how many times each foreign key is connected to the table. Output result is displayed in Figure 14.

s_id	s_lname	s_fname
1	McCaughr	Zelig
2	Beaglehol	Randie
3	McSporrin	Elyn
4	Vant	Jonell
5	Decourt	Royall
6	Maunders	Thaddus
7	O'Cahey	Claus
8	Duggen	Dee dee
9	Altoft	Gaye
10	Frearson	Abbe

Student Table

t_id	date	t_fname	t_lname	s_id
1	2021-05-2	Ailbert	Sondland	8
2	2021-08-0	Taylor	Spraging	1
3	2021-05-2	Zsazsa	Larcier	2
4	2022-02-2	Ibbie	Storms	7
5	2022-03-1	Lewiss	Tudor	10
6	2021-09-0	Albie	Peyzer	7
7	2021-10-1	Maxy	Northam	5
8	2022-01-1	Genovera	Soane	9
9	2022-01-2	Mildrid	Chalmers	9
10	2021-09-1	Hughie	Trussman	4

Teacher Table

Figure 14: .csv file with Random Data Generated for Many-One (Partial-Total)

- INPUT [**MULTIPLE COLUMN DISTRIBUTION CONSTRAINT**]
  -
  CREATE ENTITY person WITH id Row_Number, level Number, name First_Name, salary Number PRIMARY KEY id

  CREATE UNIFORM_INTEGER DISTRIBUTION WITH PARAM1 1 PARAM2 4 FOR ATTRIBUTE level IN TABLE person

  CREATE MULTIPLE COLUMN NORMAL DISTRIBUTION WITH PARAM1 10000 PARAM2 2000 WHERE ATTRIBUTE level 1 PARAM1 5000 PARAM2 1000 WHERE ATTRIBUTE level 2 PARAM1 2000 PARAM2 500 WHERE ATTRIBUTE level 3 PARAM1 1000 PARAM2 200 FOR ATTRIBUTE salary IN TABLE person

- OUTPUT :
  For multiple column distribution, the distribution of values for one column is dependent on the values another column. In this case, all values of salary column in person table follows normal distribution. However, the distribution parameters of salary is different for different values of title column. This describes the scenario that people of different job title have different ranges of salaries.Output result is displayed in Figure 23.

6 Conclusion

Sample realistic data is required for testing and deployment in many applications as data becomes a significant instrument in technical and business operations. There are several tools for creating sample realistic data, however the most of them are solely for a table. Fake Data Generator generates realistic data for all tables transformed from a user-defined entity relationship model, taking into account relationship, participation and cardinality constraints, probability distributions, and joint probability distributions. To utilize in database or other analysis processes, the output can be created in either sql or csv format. As of now, Fake Data Generator has only been able to create tables for simple relations with two entities and one relation. This project may be expanded by examining an entity relationship diagram with several entities, relations and complicated relationship types .

References

[1] DATPROF. Generation. `https://docs.datprof.com/privacy/4.11/Generation.4785283.html`, 2021.

[2] Dekking, and Michel. *A modern introduction to probability and statistics : understanding why and how*. London : Springer, 2005.

[3] Feller, W. *An Introduction to Probability Theory and Its Applications*, 3 ed., vol. 1. 1957.

[4] Feller, W. *An Introduction to Probability Theory and Its Applications*, 2 ed., vol. 2. January 1991.

[5] Holmes, A., Illowsky, B., , and Dean, S. *Introductory Business Statistics*. Pressbooks, November 2017. An optional note.

[6] Mockaroo. Mockaroo apis. `https://www.mockaroo.com/docs`.

[7] snowflake. Data generation functions. `https://docs.snowflake.com/en/sql-reference/functions-data-generation.html`.

7 Appendix

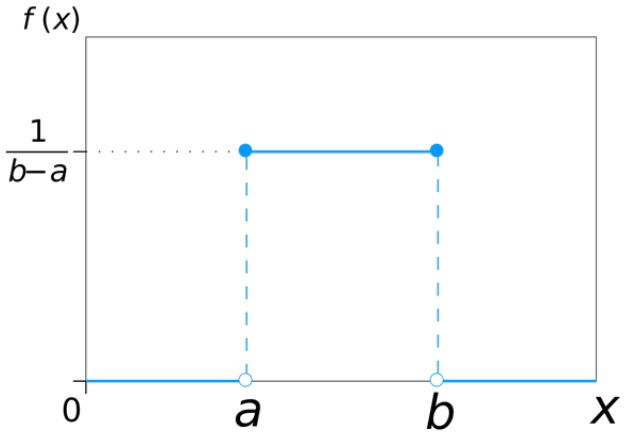


Figure 15: Probability density graph for uniform distribution

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } a \leq x \leq b, \\ 0 & \text{for } x < a \text{ or } x > b \end{cases}$$

Figure 16: Uniform distribution formula

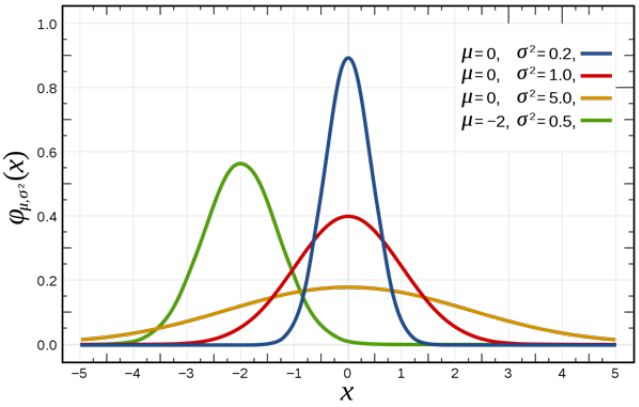


Figure 17: Probability density graph for normal distribution

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Figure 18: Normal distribution formula

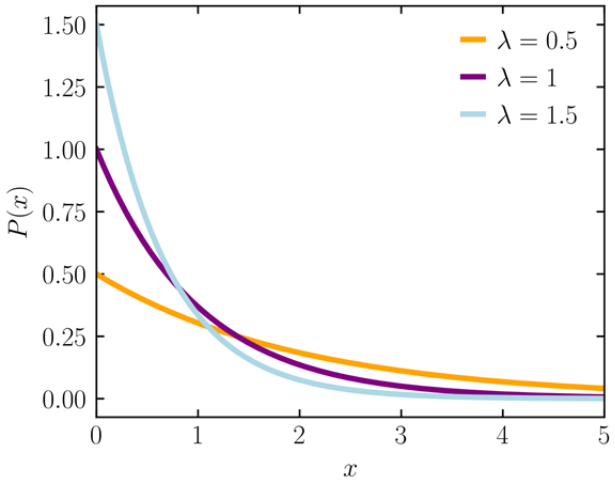


Figure 19: Probability density graph for exponential distribution

$$f(x;\lambda) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0, \\ 0 & x < 0. \end{cases}$$

Figure 20: Exponential distribution formula

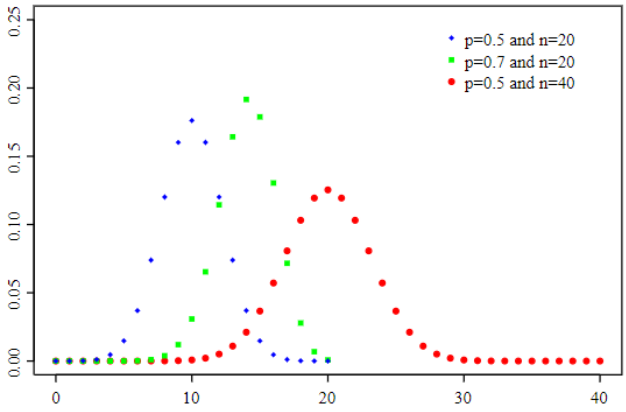


Figure 21: Probability mass function for binomial distribution

$$f(k,n,p) = \Pr(k;n,p) = \Pr(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

Figure 22: Binomial distribution probability mass function formula

level	name	id	salary
1	Evyn	2	6346.611887
1	Kahlil	11	11489.67878
1	Ada	14	9136.265762
1	Kit	16	6460.686668
2	Raffarty	7	3675.799625
2	Devin	8	4812.174789
2	Hermy	9	4718.647543
2	Seward	12	3369.425691
2	Courtnay	13	3162.823232
2	Pammie	15	3011.978259
2	Randi	18	5865.874532
2	Mic	19	6107.041375
2	Verge	20	5475.769686
3	Ransom	5	1432.743912
3	Marcelline	6	1942.748689
3	Ronny	10	2057.488707
3	Gerald	17	1669.012379
4	Carl	1	835.0078996
4	Adrianne	3	1029.163594
4	Suzy	4	1050.976704

Figure 23: .csv file with Random Data Generated for Multiple Column Distribution