Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

Студенты: Лозовска Карина и Сарварова Мария

Группы: ИУ9-51Б, ИУ9И-54Б

Преподаватель: А.Н. Непейвода

2022 г.

СОДЕРЖАНИЕ

Техническое задание

Условие задания состоит из нескольких пунктов:

1. Составление грамматики для описания объекта
2. Составление документации: как задавать пользовательский синтаксис, а также как записывать требуемый объект (или параметры для генерации)
3. Считывание значений параметров из отдельного файла "syntax.txt"
4. Построение объекта по данным, учитывая синтаксис
5. Обобщение полученных результатов и навыков, составление отчёта по практике

1 Постановка задачи

— Необходимо предложить грамматику описания сущностей, атрибутов и связей между ними с учётом кардинальностей. Параметризованными токенами грамматиками могут выступать, например:
    1. обозначения кардинальностей и типов ключей
    2. способы группировки атрибутов, относящихся к объектам (например, синтаксис скобочной структуры, ограничивающей список атрибутов, относящихся к одному и тому же объекту)

— В заданном синтаксисе из входного файла читается описание ER-диаграммы

— Результатом должна быть ER-диаграмма и реляционная диаграмма, полученная из неё посредством процедуры преобразования. Имена порождённых при преобразовании новых сущностей должны генерироваться автоматически

Также в лабораторной работе автоматически извлекается таблица кардинальностей из описания модели.

2 Индивидуальное задание

2.1 Входная грамматика

Входные данные: текстовый файл, в котором прописаны сущности. У них указаны названия, primary key, foreign key, атрибуты и возможные отношения ребёнка, родителя или что-то иное. Входную грамматику можно увидеть ниже:
$[Grammar] \rightarrow [Entities]$
$[Entities] \rightarrow [Entity] \mid [Entity];\ [Entity]$
$[Entity] \rightarrow [Name] :: [PK]; [FK]; [Attributes]; [Parent]; [Child]; [OtherRelations]$
$[Name] \rightarrow [a - zA - Z]\ [Str]$
$[PK] \rightarrow [a - zA - Z]\ [Str],\ [PK] \mid [a - zA - Z]\ [Str]$
$[FK] \rightarrow [a - zA - Z]\ [Str], [FK] \mid [a - zA - Z]\ [Str]$
$[Attributes] \rightarrow [Attribute],\ [Attributes] \mid \epsilon$
$[Attribute] \rightarrow [a - zA - Z]\ [Str]$
$[Parent] \rightarrow [Name]\ ([Cardinality]),\ [Parent] \mid \epsilon$
$[Child] \rightarrow [Name]\ ([Cardinality]),\ [Child] \mid \epsilon$
$[OtherRelations] \rightarrow [Name]([Cardinality]),\ [OtherRelations] \mid \epsilon$
$[Cardinality] \rightarrow 1 \mid 0 - M \mid 1 - M$
$[Str] \rightarrow [a - zA - Z0 - 9\_]\ [Str] \mid \epsilon$

2.2 Выходные данные

ER-диаграмма, реляционная диаграмма, полученная из неё посредством преобразования, таблица кардинальности из описания модели

Тестовые данные: Пример рандомной модели

2.3 Листинг

```
class Entity:
def __init__(self, Name):
  self.Name = Name
  self.PK = []
  self.FK = []
  self.Attributes = []
  self.Parent = {}
  self.Child = {}
  self.OtherRelations = {}
```

Листинг 1: Класс сущностей

Данный отрывок кода содержит функцию парсинга исходного файла.

Определение. Парсинг – индексирования информации с последующей конвертацией ее в иной формат или даже иной тип данных.

```
x = x.split('::') # Имя сущности
for i, y in enumerate(x):
  x[i] = x[i].strip(' ').split(';')
Ent = Entity(x[0][0])

for y in (x[1][0].split(",")): # Primary Key
  Ent.PK.append(y.strip(' '))

for y in (x[1][1].split(',')):  # Foreign Key
  if y.find('NULL') != -1: break
  Ent.FK.append(y.strip(' '))

for y in (x[1][2].split(',')): # Attributes
  Ent.Attributes.append(y.strip(' '))

for y in (x[1][3].replace('Parent : ', '').split(',')):  # Parent
  if y.find('NULL') != -1: break
  y = y.strip(' ').replace('(', '').replace(')', '')
  Ent.Parent[(y.split(' '))[0]] = (y.split(' '))[1]

for y in (x[1][4].replace('Child : ', '').split(',')):  # Child
  if y.find('NULL') != -1: break
  y = y.strip(' ').replace('(', '').replace(')', '')
  Ent.Child[(y.split(' '))[0]] = (y.split(' '))[1]
```

```
25
26    for y in (x[1][5].replace(' OtherRelations : ', '').split(',')):  # Other Relations
27      if y.find('NULL') != -1: break
28      y = y.strip(' ').replace('(', '').replace(')', '')
29      Ent.OtherRelations[(y.split(' '))[0]] = (y.split(' '))[1]
30
31    return Ent
```

В лабораторной работе была использована библиотека $JSON$, с помощью которой мы преобразовали наши входные данные в формат $JSON$. А далее уже из этого файла с помощью $rdot$ получаем графики er-модели и реляционной модели.

Далее заполняем информацию об аттрибутных сущностей и об отношениях между сущностями (Листинг 3). Но сначала преобразуем обозначения кардинальности к нужному виду (Листинг 2):

$0 - M \rightarrow *$

$1 - M \rightarrow +$

$1 \rightarrow 1$

```
1   for j, i in enumerate(Entities):
2     for k, v in i.OtherRelations.items():
3       if v == "1-M": Entities[j].OtherRelations[k] = "+"
4       elif v == "0-M": Entities[j].OtherRelations[k] = "*"
5
6   for j, i in enumerate(Entities):
7     for k, v in i.Parent.items():
8       if v == "1-M": Entities[j].Parent[k] = "+"
9       elif v == "0-M": Entities[j].Parent[k] = "*"
10
11  for j, i in enumerate(Entities):
12    for k, v in i.Child.items():
13      if v == "1-M": Entities[j].Child[k] = "+"
14      elif v == "0-M": Entities[j].Child[k] = "*"
```

Листинг 2: Преобразование обозначения кардинальности к нужному виду

```
        for num, ent in enumerate(Entities):
    for k, v in ent.Parent.items():
      for i in Entities:
        if i.Name == k:
          for k1, v1 in i.Child.items():
            if k1 == ent.Name:
              result = v1
              dic['relations'].append(str(k) + ":" + i.PK[0] + " " + str(v) + "--" +
                  str(result) + " " + ent.Name + ":" + ent.FK[0])
              break

    for k, v in ent.OtherRelations.items():
      for num1, i in enumerate(Entities):
        if (i.Name == k) & (num1 > num):
          for k1, v1 in i.OtherRelations.items():
            if k1 == ent.Name:
              result = v1
              dic['relations'].append(ent.Name + ":" + ent.PK[0] + " " + result + "--" +
                  v + " " + k + ":" + i.PK[0])
              break
```

Листинг 3: Заполнение информации

```
    dic = to_json_file(Entities, 1)
  with open("diagram1.json", "w") as write_file:
    json.dump(dic, write_file, indent=4)
  command = "python3 --version"

  os.system("erdot diagram1.json")
  os.system("dot diagram1.dot -Tpng -o diagram1.png")
```

Листинг 4: Модель сущность-связь

Для преобразования ER-модели в реляционную мы выделяем обозначение FK и PK, а также добавляем новые сущности. Для этого проверяем сущности на связь многие ко многим, а затем добавляем связи с новой сущностью в изначальные.

```
            for i, e in enumerate(Entities):
        if e.Name == z1[0]:
          if y[3] == '+': Entities[i].Child[ent.Name] = '1-M'
          if y[3] == '*': Entities[i].Child[ent.Name] = '0-M'
          e.OtherRelations.pop(z2[0])
          continue
        if e.Name == z2[0]:
          if y[0] == '+': Entities[i].Child[ent.Name] = '1-M'
          if y[0] == '*': Entities[i].Child[ent.Name] = '0-M'
          e.OtherRelations.pop(z1[0])
          continue
```

Листинг 5: Добавление связей в реляционной модели

Для дополнительного задания, построения таблицы кардинальности, была использована библиотека *tabulate*. В два цикла мы сначала проходим по отношениям ребёнка - родителя, а потом уже по остальным сущностям. И сразу заполяем таблицу нужными данными для её построения.

```
            if i.Child[ent.Name] == '1':
              if ent.Parent[i.Name] == '1':
                newdata.append('1:1')
                newdata.append('M-O')
              if ent.Parent[i.Name] == '0-M':
                newdata.append('1:N')
                newdata.append('M-O')
              if ent.Parent[i.Name] == '1-M':
                newdata.append('N:1')
                newdata.append('M-M')
            if i.Child[ent.Name] == '0-M':
              if ent.Parent[i.Name] == '1':
                newdata.append('N:1')
                newdata.append('O-M')
              if ent.Parent[i.Name] == '0-M':
                newdata.append('M:N')
                newdata.append('O-O')
              if ent.Parent[i.Name] == '1-M':
                newdata.append('M:N')
                newdata.append('O-M')
            if i.Child[ent.Name] == '1-M':
              if ent.Parent[i.Name] == '1':
                newdata.append('1:N')
                newdata.append('M-M')
              if ent.Parent[i.Name] == '0-M':
                newdata.append('M:N')
                newdata.append('M-O')
              if ent.Parent[i.Name] == '1-M':
                newdata.append('M:N')
                newdata.append('M-M')
            data.append(newdata)
```

Листинг 6: Заполнение отношений родитель – ребёнок

```
                if i.OtherRelations[ent.Name] == '1':
                  if ent.OtherRelations[i.Name] == '1':
                    newdata.append('1:1')
                    newdata.append('M-O')
                  if ent.OtherRelations[i.Name] == '0-M':
                    newdata.append('1:N')
                    newdata.append('M-O')
                  if ent.OtherRelations[i.Name] == '1-M':
                    newdata.append('1:N')
                    newdata.append('M-M')
                if i.OtherRelations[ent.Name] == '0-M':
                  if ent.OtherRelations[i.Name] == '1':
                    newdata.append('1:N')
                    newdata.append('O-M')
                  if ent.OtherRelations[i.Name] == '0-M':
                    newdata.append('M:N')
                    newdata.append('O-O')
                  if ent.OtherRelations[i.Name] == '1-M':
                    newdata.append('M:N')
                    newdata.append('O-M')
                if i.OtherRelations[ent.Name] == '1-M':
                  if ent.OtherRelations[i.Name] == '1':
                    newdata.append('1:N')
                    newdata.append('M-M')
                  if ent.OtherRelations[i.Name] == '0-M':
                    newdata.append('M:N')
                    newdata.append('M-O')
                  if ent.OtherRelations[i.Name] == '1-M':
                    newdata.append('M:N')
                    newdata.append('M-M')
                data.append(newdata)
```

Листинг 7: Заполнение других отношений между сущностями

3 Пример работы программы на тестовых данных

Тестовые входные данные:

Customer :: Phone; NULL; FirstName, LastName, Email, City; Parent : NULL;
Child : NULL; OtherRelations : Order (0-M)
BookStore :: BookstoreName; NULL; Email, URL, Phone; Parent : NULL;
Child : NULL; OtherRelations : Order (0-M)
Order :: OrderNumber; NULL; PaymentType, MakingOrderDate, DeliveryDate;
Parent : NULL; Child : OrderLineBook (1-M); OtherRelations : Customer (1), BookStore (1)
OrderLineBook :: LineNumber; OrderNumber, ISBN; Quantity, ExtendedPrice;
Parent : Order (1), Book (1); Child : NULL; OtherRelations : Book (1)
Book :: ISBN; NULL; Title, Genre, PublishingHouse, PublishingYear, Price;
Parent : NULL; Child : OrderLineBook (1-M); OtherRelations : Author (1-M)
Author :: AuthorID; NULL; FirstName, LastName, DeathYear, Country; Parent : NULL;
Child : NULL; OtherRelations : Book (0-M)

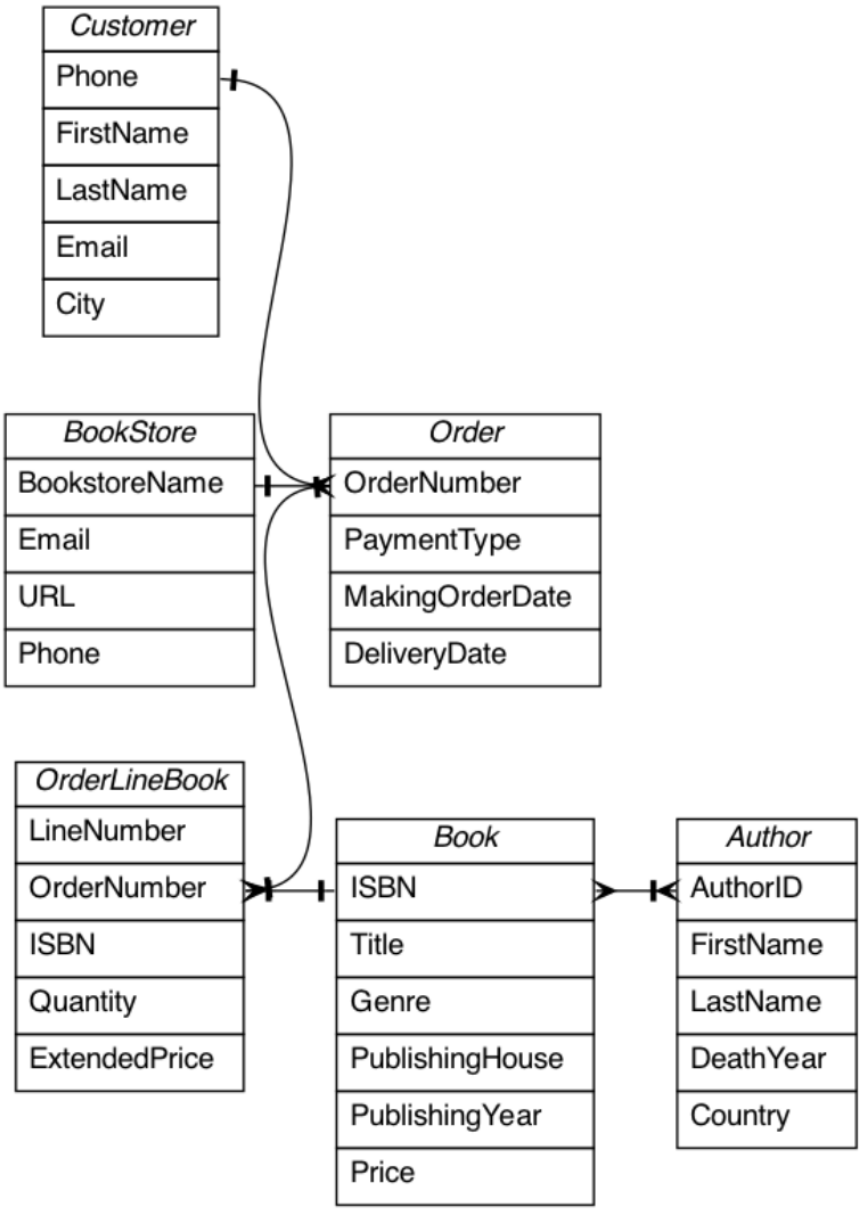

Рисунок 1 — ER-модель

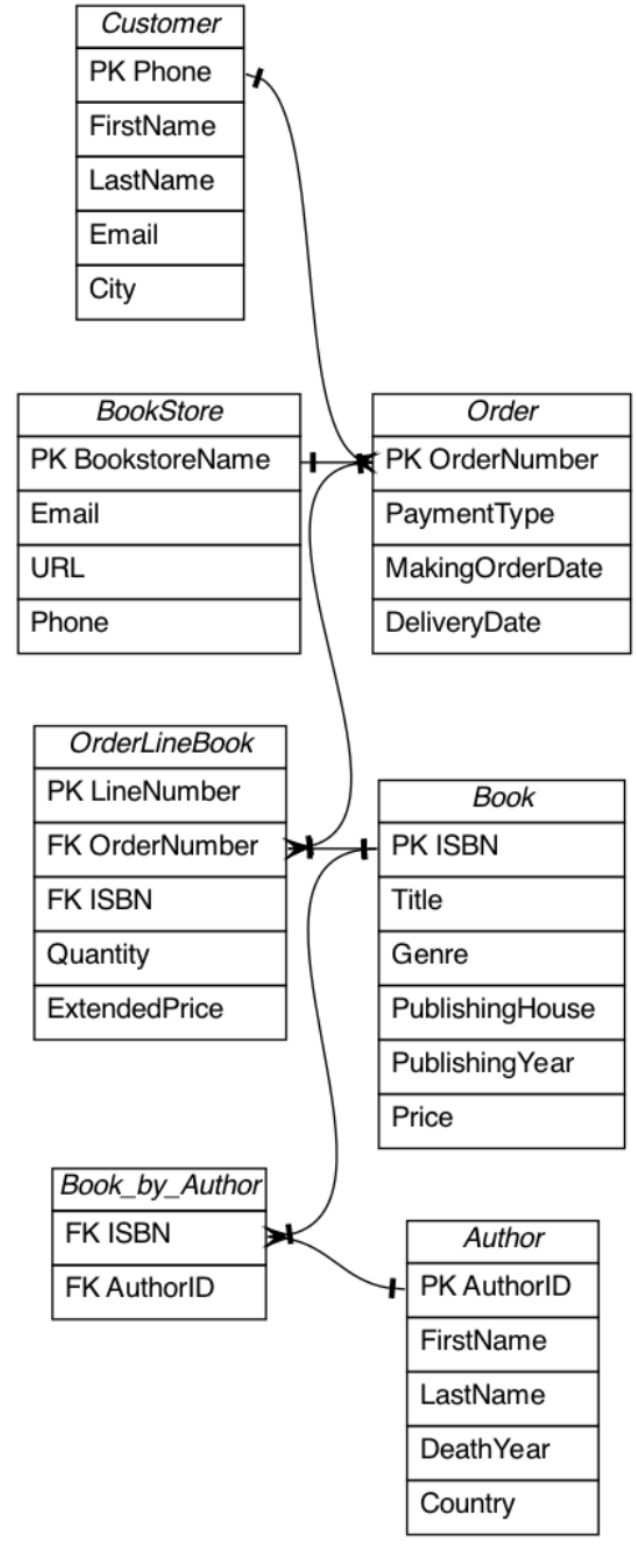

Рисунок 2 — Реляционная модель

	Parent	Child	MAX	MIN
0	Customer	Order	1:N	M-O
1	BookStore	Order	1:N	M-O
2	Order	OrderLineBook	1:N	M-M
3	Book	OrderLineBook	1:N	M-M
4	Book	Author	M:N	O-M

Рисунок 3 — Таблица кардинальностей

4 Заключение

Во время лабораторной работы было выполнено данное по вариантам задание. Результатом нашей деятельности является построение ER-диаграммы, реляционной диаграммы и извлечение таблицы кардинальностей по входным данным.