データベース中間レポート

学籍番号　: 22266505

氏名　　: 岩井 政樹

学科　　: 知能情報システム工学科

目次

1. 本レポートの概要

本レポートでは定食屋の商品、注文を管理、分析するためのデータベースを作成した。

はじめに作成したデータベースのER図を示し、ER図を基にMySQLを用いて実際にデータベースを作成、注文テーブルに対するデータの追加・削除、ユーザデータの変更、注文配達データの作成を行い、その結果を示した。

2. データベースの目的・概要

本レポートで想定する定食屋の概要を以下に示す。

- 客は主食（ご飯）、汁物、副菜（サラダ）、主菜（メイン）をそれぞれ、メニューから一つずつ選び注文する
- 複数の支店が存在する
- 電話やモバイルアプリを用いたデリバリーでの注文も承っている

上のような定食屋の商品、注文を管理するためのデータベースを作成した。
次項3でデータベースの関係図を示し、説明を加える。

3. 関係図

3.1. ER図

以下に作成したデータベースのER図を示す。

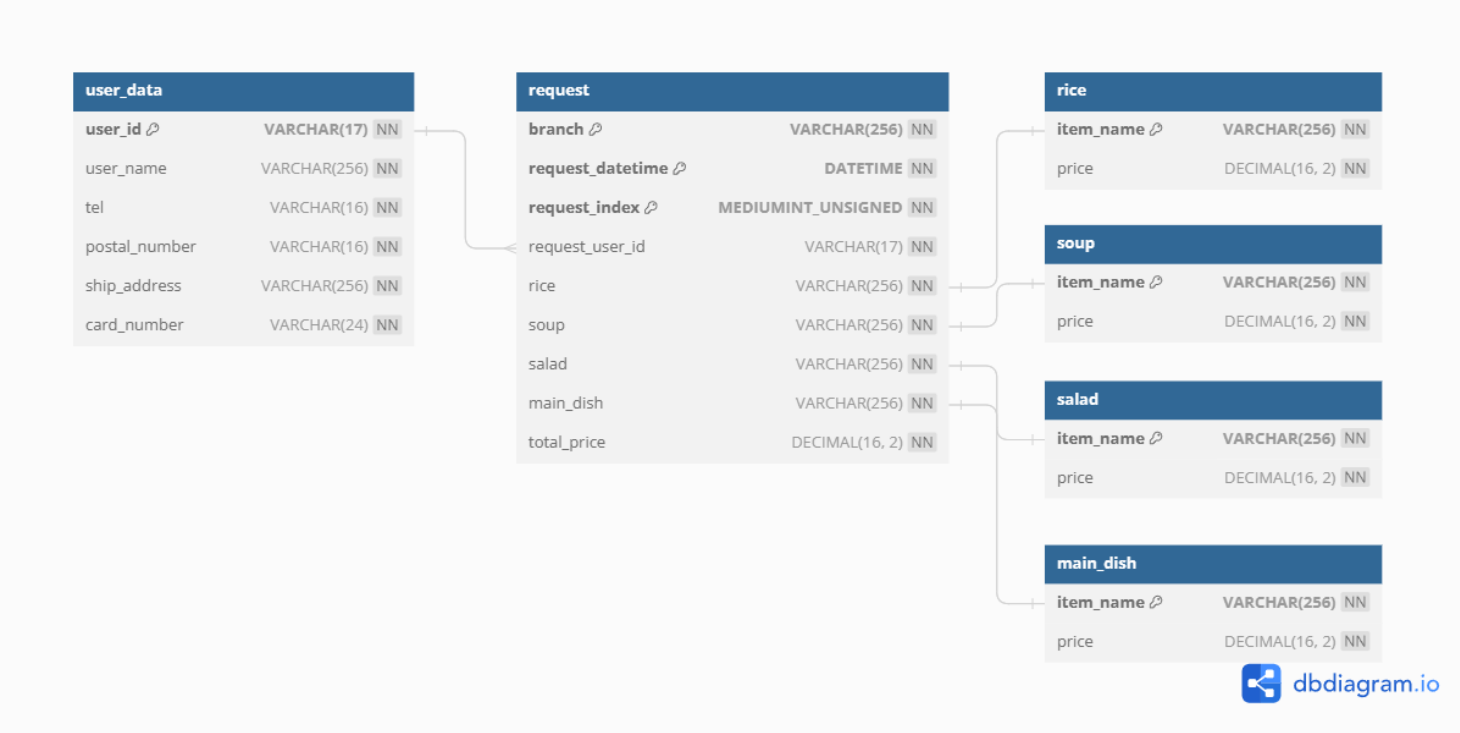

図1. 定食屋データベースのER図

user_dataテーブルにはユーザの識別と配達、決済に必要な情報を保存する。
user_idはユーザーごとに16桁のunique IDを割り当てている。
また、モバイルアプリの使用率をデータベースから算出するためモバイルアプリのuser_idのprefixにMを、支店のシステムのuser_idのprefixにはSをつけることとした。

requestテーブルには注文を受けた支店と日時、その支店での注文番号の注文を一意に識別するための情報と、注文をしたユーザーを識別するためにユーザーID、注文内容と合計金額を保存する。
各支店での注文番号は毎日０時に１にリセットされ、注文が入る度にインクリメントされていくものとした。これは、注文にユニークなIDを割り当てるといずれID空間を埋め尽くしてしまう可能性があるためである。

商品は主食（ご飯）、汁物、副菜（サラダ）、主菜（メイン）のそれぞれの分類ごとにテーブルを作成した。
商品テーブルには商品名と商品価格を保存する。

3.2. テーブル(表)

それぞれのテーブルについてデータ例を以下に示す。

request								
branch	request_datetime	request_index	request_user_id	rice	soup	salad	main_dish	total_price
三鷹支店	2023-01-01 05:43:25	151	M2258555050055487	白米 大	豚汁	ほうれん草のごま和え	豚の生姜焼き	1150.00
三鷹支店	2023-01-01 08:28:14	148	M1837026021070863	白米 大	コンソメスープ	ワカメのサラダ	鮭の塩焼き	1200.00
三鷹支店	2023-01-01 09:56:15	150	M6991501146827906	白米 中	コンソメスープ	ワカメのサラダ	鯖の塩焼き	1200.00

図2. request テーブル

user_data					
user_id	user_name	tel	postal_number	ship_address	card_number
M0043036203350933	竹内棟上	09053097678	192-0004	東京都八王子市加住町2-2-12	1093085779586950
M0075290594937843	谷本啓三	08038071971	153-0042	東京都目黒区青葉台3-12-1青葉台コーポ305	9224079771269144
S0000000000000001	小金井支店	0424721835	184-0002	東京都小金井市梶尾町5丁目1-1	

図3. user_data テーブル

rice	
item_name	price
白米 中	200.00
白米 大	250.00
白米 小	150.00

図4. rice テーブル

soup	
item_name	price
コンソメスープ	100.00
豆腐の味噌汁	100.00
豚汁	200.00

図5. soup テーブル

salad	
item_name	price
チキンのサラダ	400.00
ほうれん草のごま和え	200.00
ワカメのサラダ	300.00

図6. salad テーブル

main_dish	
item_name	price
煮込みハンバーグ	600.00
豚の生姜焼き	500.00
鮭の塩焼き	550.00
鯖の塩焼き	600.00

図7. main_dish テーブル

4. 演算

ユーザーに商品を配達するために必要なデータを取り出すための演算、射影、選択、結合をそれぞれ考える。

注文のデータが与えられたときを考える。

ユーザーに商品を配達するために必要なデータを考える。
配達に必要なデータはそのユーザーが注文した、注文データとカード番号以外のユーザーデータである。

4.1. 結合

配達のためには射影で取り出したデータに加えて注文した内容も必要であるためユーザーIDをキーとして、注文テーブルとユーザーテーブルを結合したデータが必要となる。 そのための結合の演算は以下のようになる

```
結合 request user_data
```

4.2. 選択

結合した表から、配達する注文のデータを一意に特定する必要がある。 これには、支店名、注文日時、注文インデックス、ユーザーデータをキーとして、結合した表の中からデータを選択すればよい。
以下に示す演算でデータを選択し、配達のための注文データを一意に特定し取り出すことができる。

```
選択 branch 三鷹支店 and request_datetime 2023-01-01 02:43:01 and request_index 1 and user_id M4316471149704861
```

4.3. 射影

配達のために必要なデータは注文内容と合計金額、ユーザー名、電話番号、郵便番号、住所である。
上記の射影を得る演算は以下のようになる。

```
射影 user_name, tel, postal_number, ship_address, rice, soup, main_dish, total_price
```

以上、3つの演算を組み合わせることで配達のために必要なデータを得ることができる。

5. 実験

MySQLを用いて実際にデータベースを作成し、データベースの操作を行う。
行う操作の内容を次項4.1に示す。

5.1. 実験環境

Windows11上のDocker for Windows using WSL2 backendにMySQL環境を構築した。
イメージはDocker hubのMySQL公式イメージを使用した。
MySQLのバージョンは8.2.0, distributionはOracle Linuxである。

databaseの操作はホストOS上のpythonスクリプトから行う。
database操作はMySQL公式から配布されているモジュールmysql-connector-pythonを用いた。

5.1.1. 初期データ

また、ユーザーデータの初期データは一部疑似個人情報データ生成サービスを用いて生成した。
注文データの初期データはユーザーデータと商品データの初期データから生成した。

商品データは手動で作成したものを主食（ご飯）、汁物、副菜（サラダ）、主菜（メイン）それぞれのカテゴリで3つずつ、
ユーザーデータは手動で作成したもの3件を含めた101件、
注文データはその他の初期データからランダム生成した5000件
を初期データとして用いる。

5.2. 実験内容

5.2.1. 注文データの追加・削除

以下に示す関数を用いて、10件の注文データをランダム生成し、データ挿入・削除の実行時間の計測と、データが挿入・削除されているかの検証を行った。
関数内で利用しているユーザー定義関数はデータの挿入・削除に関して本質的な関係がないと判断したものはここに記述しないが、本レポートの末尾に付録として、実験に使用したスクリプトをすべて記載した。

```
def writeRequestData(tableName, tableData):
  with open(f"./table_data/{tableName}.csv", mode="w", encoding='utf-8') as f:
    f.write("branch, request_datetime, request_index, request_user_id, rice, soup, salad, main_dish, total_price\n")
    for tpl in tableData:
      f.write(f"{tpl[0]}, {tpl[1]}, {tpl[2]}, {tpl[3]}, {tpl[4]}, {tpl[5]}, {tpl[6]}, {tpl[7]}, {tpl[8]}\n")

def insertRequestData(
    cursor: cursor.MySQLCursor,
    value : dict[str, str]
):
  query = ("INSERT INTO request "
           "(branch, "
           "request_datetime, request_index, request_user_id, "
           "rice, soup, salad, main_dish, "
           "total_price) "
           "VALUES (%(branch)s, "
           "%(request_datetime)s, %(request_index)s, %(request_user_id)s, "
           "%(rice)s, %(soup)s, %(salad)s, %(main_dish)s, "
           "%(total_price)s)")

  cursor.execute(query, value)

def deleteRequestData(
  cursor: cursor.MySQLCursor,
  branch: str,
  request_datetime: str,
  request_index: str
):
  query = ("DELETE FROM request "
             "WHERE branch = '{}' AND request_datetime = '{}' AND request_index = '{}' "
             .format(branch, request_datetime.replace('T', ' '), request_index))
  cursor.execute(query)

def insertRandomRequestData(
  cnx: mysql.connector.MySQLConnection,
  cursor: cursor.MySQLCursor,
  count: int
):
  rice     = fetchTableData(cursor, 'rice')
  salad    = fetchTableData(cursor, 'salad')
  soup     = fetchTableData(cursor, 'soup')
  mainDish = fetchTableData(cursor, 'main_dish')
  userData = fetchTableData(cursor, 'user_data')

  requestData = [
    createRandomRequestData(
        rice, salad, soup, mainDish, userData
    ) for _ in range(count)
    ]

  writeRequestData('insert_request_data', [list(req.values()) for req in requestData])

  with open('insert_request.txt', mode="w", encoding='utf-8') as f:
    f.write(f"実験開始時刻: {datetime.datetime.now().isoformat()}\n")
    start = time.perf_counter_ns()
    for req in requestData:
      insertRequestData(cursor, req)
    cnx.commit()
    end = time.perf_counter_ns()
    f.write(f"計測実行時間: {(end-start) / 1000000} [ms]")

  selected = []
  for req in requestData:
    query = ("SELECT * FROM request "
             "WHERE branch='{}' AND request_datetime = '{}' AND request_index='{}' "
             .format(req['branch'], req['request_datetime'], req['request_index']))
    cursor.execute(query)
    selected.append(cursor.fetchall()[0])
```

```
writeRequestData('insert_request_result', selected)

with open('delete_request.txt', mode="w", encoding='utf-8') as f:
  f.write(f"実験開始時刻: {datetime.datetime.now().isoformat()}\n")
  start = time.perf_counter_ns()
  for req in requestData:
    deleteRequestData(cursor, req['branch'], req['request_datetime'], req['request_index'])
  cnx.commit()
  end = time.perf_counter_ns()
  f.write(f"計測実行時間: {(end-start) / 1000000} [ms]")

selected = []
for req in requestData:
  query = ("SELECT * FROM request "
           "WHERE branch='{}' AND request_datetime = '{}' AND request_index='{}' "
           .format(req['branch'], req['request_datetime'], req['request_index']))
  cursor.execute(query)
  selected.append(cursor.fetchall()[0])

writeRequestData('delete_request_result', selected)
```

プログラム1. データ挿入・削除を行うスクリプト

5.2.2. ユーザーデータの変更

以下のスクリプトを用いて、配達先住所の変更を行い、データ変更の実行速度の計測と、データが変更されているかの検証を行った。

```
def updateUserAddress(
  cnx: mysql.connector.MySQLConnection,
  cursor: cursor.MySQLCursor,
  user_id: str,
  postal_number: str,
  ship_adress: str
):
  query = ("UPDATE user_data "
           "SET postal_number='{}', ship_address='{}' "
           "WHERE user_id='{}'".format(postal_number, ship_adress, user_id))

  select_query = ("SELECT * FROM user_data WHERE user_id='{}'".format(user_id))

  with open('update_request.txt', mode="w", encoding='utf-8') as f:
    f.write(f"実験開始時刻: {datetime.datetime.now().isoformat()}\n")
    start = time.perf_counter_ns()
    cursor.execute(query)
    cnx.commit()
    end = time.perf_counter_ns()
    f.write(f"計測実行時間: {(end-start) / 1000000} [ms]")

    cursor.execute(select_query)
    result = cursor.fetchall()[0]
    f.write(", ".join(result))
```

プログラム2. データ変更を行うスクリプト

5.2.3. 配達データの作成

以下のスクリプトを用いて、配達に必要なデータを取り出す演算を行い、その実行速度の計測と実行結果の検証を行った。

```
def createShippingData(
  cursor: cursor.MySQLCursor,
  user_id: str,
  branch: str,
  request_datetime: str,
  request_index: str
):
  query = ("SELECT user_name, tel, postal_number, ship_address, rice, soup, salad, main_dish, total_price "
               "FROM user_data INNER JOIN request ON user_data.user_id = request.request_user_id "
               "WHERE user_id='{}' AND branch='{}' AND request_datetime = '{}' AND request_index='{}'"
               .format(user_id, branch, request_datetime, request_index))

  with open('select_request.txt', mode="w", encoding='utf-8') as f:
    f.write(f"実験開始時刻: {datetime.datetime.now().isoformat()}\n")
    start = time.perf_counter_ns()
    cursor.execute(query)
    end = time.perf_counter_ns()
    f.write(f"計測実行時間: {(end-start) / 1000000} [ms]\n")

    tpl = cursor.fetchall()[0]
    f.write(f"{tpl[0]}, {tpl[1]}, {tpl[2]}, {tpl[3]}, {tpl[4]}, {tpl[5]}, {tpl[6]}, {tpl[7]}, {tpl[8]}\n")
```

プログラム3. データ結合と選択、射影を行うスクリプト

5.3. 実験結果

5.3.1. 注文データの追加・削除

自動生成されたデータは以下のようになった。

```
branch, request_datetime, request_index, request_user_id, rice, soup, salad, main_dish, total_price
国分寺支店, 2023-04-20T09:03:31, 5, M0959979499165585, 白米 中, 豆腐の味噌汁, ワカメのサラダ, 豚の生姜焼き, 1100.00
三鷹支店, 2023-07-26T11:41:34, 4, M3764404782998205, 白米 大, 豆腐の味噌汁, ワカメのサラダ, 鯖の塩焼き, 1250.00
三鷹支店, 2023-03-19T13:58:27, 5, M6363338569666435, 白米 大, コンソメスープ, ほうれん草のごま和え, 豚の生姜焼き, 1050.00
小金井支店, 2023-12-26T21:46:18, 4, M8149505322780065, 白米 中, 豚汁, ほうれん草のごま和え, 鯖の塩焼き, 1200.00
国分寺支店, 2023-07-04T12:58:02, 11, M5980822666877486, 白米 大, コンソメスープ, ワカメのサラダ, 鯖の塩焼き, 1250.00
三鷹支店, 2023-04-28T23:08:07, 4, M0514976918844333, 白米 小, 豚汁, チキンのサラダ, 豚の生姜焼き, 1250.00
国分寺支店, 2023-08-15T05:50:37, 3, M1525342523725107, 白米 大, コンソメスープ, ワカメのサラダ, 鯖の塩焼き, 1250.00
三鷹支店, 2023-12-04T11:21:28, 4, M5628732464656951, 白米 大, コンソメスープ, チキンのサラダ, 鯖の塩焼き, 1350.00
小金井支店, 2023-03-07T00:59:08, 8, M0851225857326843, 白米 大, コンソメスープ, ほうれん草のごま和え, 鯖の塩焼き, 1150.00
小金井支店, 2023-04-09T01:05:03, 5, M1742213958426250, 白米 小, コンソメスープ, ワカメのサラダ, 鯖の塩焼き, 1150.00
```

プログラム4. 自動生成されたデータ

データの追加

- 実験開始時刻: 2023-12-01T21:47:24.514500
- 計測実行時間: 31.1841 [ms]

データ挿入後、挿入されたデータを検索したところ、すべてのデータが正常に得られた。
データは正常に挿入されたと判断できる。

データの削除

- 実験開始時刻: 2023-12-01T21:47:24.565307
- 計測実行時間: 23.5708 [ms]

データ削除後、削除されたデータを検索したところ、すべてのデータは得られなかった。
データは正常に削除されたと判断できる。

5.3.2. ユーザーデータの変更

上のデータを下のデータに変更した。

- `M0514976918844333, 北梅吉, 09022631866, 193-0841, 東京都八王子市裏高尾町3-11-20コンフォート裏高尾町108, 5968600999006945`
- `M0514976918844333, 北梅吉, 09022631866, 106-0045, 東京都港区麻布十番3-11-8, 5968600999006945`

実験結果は以下のようになった。

- 実験開始時刻: 2023-12-01T21:55:47.651606
- 計測実行時間: 10.8684 [ms]
- `M0514976918844333, 北梅吉, 09022631866, 106-0045, 東京都港区麻布十番3-11-8, 5968600999006945`

変更後のデータが得られており、データは正常に変更されたと判断できる。

5.3.3. 配達データの作成

以下の注文、ユーザーデータを結合した配達データを作成した。

- 三鷹支店, 2023-01-02 08:33:27, 1, M3493785522067783, 白米 小, 豆腐の味噌汁, チキンのサラダ, 鯖の塩焼き, 1250.00
- M3493785522067783, 根岸葵日夏, 09013167145, 106-0045, 東京都港区麻布十番3-5-11シティ麻布十番308, 5643387009288408

結果は以下のようになった。

- 実験開始時刻: 2023-12-01T22:24:58.907206
- 計測実行時間: 1.9426 [ms]
- 根岸葵日夏, 09013167145, 106-0045, 東京都港区麻布十番3-5-11シティ麻布十番308, 白米 小, 豆腐の味噌汁, チキンのサラダ, 鯖の塩焼き, 1250.00

第4項で説明した目的のデータのみが得られており、演算は正常に行われたと判断できる。

6. 総括・考察

定食屋の商品、注文を管理、分析するためのデータベースを設計し、MySQLを用いて実際にデータベースを作成、注文テーブルに対するデータの追加・削除、ユーザーデータの変更、注文配達データの作成を行うことができた。
論理的に設計したデータベースが、物理的にも正常に動作することが確認できた。

今回設計したデータベースでは、ユーザーの配達先がひとつしか設定できないなど、実際のサービスを運用するには設計が甘い部分がある。
また、物理的な操作に関してもSQLインジェクションの対策をしていないなど、実際のサービスを運用するには甘い部分が多い。

しかし、基本的なデータベース設計、操作に関して具体例を示すことはできたと考える。
今後、余裕があれば設計、操作に関して甘い部分を修正したものを作成したい。

7. 参考文献

MySQLの操作に関して以下のサイトを参考にした。

- MySQL公式ドキュメント version 8.2, (https://dev.mysql.com/doc/refman/8.2/en/)
- MySQL Connector/Python Developer Guide, (https://dev.mysql.com/doc/connector-python/en/connector-python-example-connecting.html)

8. 付録

実験に使用したスクリプトを次ページ以降に示す。

```
import mysql.connector
from mysql.connector import errorcode
from mysql.connector import cursor
from decimal import *
from random import randrange
import datetime
import time

def connectDB():
  try:
    cnx = mysql.connector.connect(
      user     = 'root',
      password = 'root',
      host     = '127.0.0.1',
      database = 'restaurant'
    )
  except mysql.connector.Error as err:
    if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
      print("Something is wrong with your user name or password")
    elif err.errno == errorcode.ER_BAD_DB_ERROR:
      print("Database does not exist")
    else:
      print(err)

    return None
  else:
    print('connect sucess')
    return cnx

def fetchTableData(cursor: cursor.MySQLCursor, table: str):
  query = "SELECT * FROM {}".format(table)
  cursor.execute(query)

  return cursor.fetchall()

BRANCH = ["小金井支店", "三鷹支店", "国分寺支店"]
requestIndexes = {
  "小金井支店": dict[str, str](),
  "三鷹支店"  : dict[str, str](),
  "国分寺支店": dict[str, str]()
}
def setUpRequestIndexes(cursor: cursor.MySQLCursor):
  request = fetchTableData(cursor, 'request')
  for req in request:
    day = req[1].replace(hour=0, minute=0, second=0).isoformat()
    requestIndexes[req[0]][day] = 0

  for req in request:
    day = req[1].replace(hour=0, minute=0, second=0).isoformat()
    requestIndexes[req[0]][day] = max(int(req[2]), requestIndexes[req[0]][day])

def createRandomRequestData(riceData, soupData, saladData, mainDishData, userData):
  branchIndex        = randrange(0, len(BRANCH))
  userDataIndex      = randrange(0, len(userData))
  riceIndex          = randrange(0, len(riceData))
  soupIndex          = randrange(0, len(soupData))
  saladIndex         = randrange(0, len(saladData))
  mainDishIndex      = randrange(0, len(mainDishData))
  totalPrice:Decimal = (riceData[riceIndex][1] +
                        saladData[saladIndex][1] +
                        soupData[soupIndex][1] +
                        mainDishData[mainDishIndex][1])

  dateTime = datetime.datetime(
```

```
    year   = 2023,
    month  = randrange(1,13),
    day    = randrange(1,29)
  )

  dateStr = dateTime.isoformat()
  if not dateStr in requestIndexes[BRANCH[branchIndex]]:
    requestIndexes[BRANCH[branchIndex]][dateStr] = 1
  else:
    requestIndexes[BRANCH[branchIndex]][dateStr] = requestIndexes[BRANCH[branchIndex]][dateStr] + 1

  dateTime = dateTime.replace(
    hour   = randrange(0,24),
    minute = randrange(0,60),
    second = randrange(0, 60)).isoformat()

  value = {
    'branch'          : BRANCH[branchIndex],
    'request_datetime': dateTime,
    'request_index'   : requestIndexes[BRANCH[branchIndex]][dateStr],
    'request_user_id' : userData[userDataIndex][0],
    'rice'            : riceData[riceIndex][0],
    'soup'            : saladData[saladIndex][0],
    'salad'           : soupData[soupIndex][0],
    'main_dish'       : mainDishData[mainDishIndex][0],
    'total_price'     : totalPrice,
  }

  return value

def writeRequestData(tableName, tableData):
  with open(f"./table_data/{tableName}.csv", mode="w", encoding='utf-8') as f:
    f.write("branch, request_datetime, request_index, request_user_id, rice, soup, salad, main_dish, total_price\n")
    for tpl in tableData:
      f.write(f"{tpl[0]}, {tpl[1]}, {tpl[2]}, {tpl[3]}, {tpl[4]}, {tpl[5]}, {tpl[6]}, {tpl[7]}, {tpl[8]}\n")

def insertRequestData(
    cursor: cursor.MySQLCursor,
    value : dict[str, str]
):
  query = ("INSERT INTO request "
           "(branch, "
           "request_datetime, request_index, request_user_id, "
           "rice, soup, salad, main_dish, "
           "total_price) "
           "VALUES (%(branch)s, "
           "%(request_datetime)s, %(request_index)s, %(request_user_id)s, "
           "%(rice)s, %(soup)s, %(salad)s, %(main_dish)s, "
           "%(total_price)s)")

  cursor.execute(query, value)

def insertRandomRequestData(
  cnx: mysql.connector.MySQLConnection,
  cursor: cursor.MySQLCursor,
  count: int
):
  rice     = fetchTableData(cursor, 'rice')
  salad    = fetchTableData(cursor, 'salad')
  soup     = fetchTableData(cursor, 'soup')
  mainDish = fetchTableData(cursor, 'main_dish')
  userData = fetchTableData(cursor, 'user_data')

  requestData = [
    createRandomRequestData(
```

```
        rice, salad, soup, mainDish, userData
      ) for _ in range(count)
    ]

  writeRequestData('insert_request_data', [list(req.values()) for req in requestData])

  with open('insert_request.txt', mode="w", encoding='utf-8') as f:
    f.write(f"実験開始時刻: {datetime.datetime.now().isoformat()}\n")
    start = time.perf_counter_ns()
    for req in requestData:
      insertRequestData(cursor, req)
    cnx.commit()
    end = time.perf_counter_ns()
    f.write(f"計測実行時間: {(end-start) / 1000000} [ms]")

  selected = []
  for req in requestData:
    query = ("SELECT * FROM request "
             "WHERE branch='{}' AND request_datetime = '{}' AND request_index='{}' "
             .format(req['branch'], req['request_datetime'], req['request_index']))
    cursor.execute(query)
    selected.append(cursor.fetchall()[0])

  for req in requestData:
    query = ("DELETE FROM request "
             "WHERE branch='{}' AND request_datetime = '{}' AND request_index='{}' "
             .format(req['branch'], req['request_datetime'], req['request_index']))
    cursor.execute(query)
  cnx.commit()

  writeRequestData('insert_request_result', selected)

def deleteRequestData(
  cnx: mysql.connector.MySQLConnection,
  cursor: cursor.MySQLCursor,
  branch: str,
  request_datetime: str,
  request_index: str
):
  query = ("DELETE FROM request "
           "WHERE branch = '{}' AND request_datetime = '{}' AND request_index = '{}' "
           .format(branch, request_datetime.replace('T', ' '), request_index))
  cursor.execute(query)
  cnx.commit()



def main():
  cnx      = connectDB()
  cursor   = cnx.cursor()

  rice     = fetchTableData(cursor, 'rice')
  salad    = fetchTableData(cursor, 'salad')
  soup     = fetchTableData(cursor, 'soup')
  mainDish = fetchTableData(cursor, 'main_dish')
  userData = fetchTableData(cursor, 'user_data')
  req      = fetchTableData(cursor, 'request')

  setUpRequestIndexes(cursor)
  insertRandomRequestData(cnx, cursor, 10)

  #with open('table_data/insert_request_data.csv', mode='r', encoding='utf-8') as f:
  #  for line in f:
  #    row = line.split(', ')
  #    if row[0] == 'branch':
  #      pass
  #    else:
  #      deleteRequestData(cnx, cursor, row[0], row[1], row[2])
```

```
  cursor.close()
  cnx.close()

if __name__ == "__main__":
  main()

def writeItemData(tableName, tableData):
  with open(f"./table_data/{tableName}.csv", mode="w", encoding='utf-8') as f:
    f.write("item_name, price\n")
    for tpl in tableData:
      f.write(f"{tpl[0]}, {tpl[1]}\n")

def writeUserData(tableName, tableData):
  with open(f"./table_data/{tableName}.csv", mode="w", encoding='utf-8') as f:
    f.write("user_id, user_name, tel, postal_number, ship_address, card_number\n")
    for tpl in tableData:
      f.write(f"{tpl[0]}, {tpl[1]}, {tpl[2]}, {tpl[3]}, {tpl[4]}, {tpl[5]}\n")

def writeDBcontents(cursor: cursor.MySQLCursor):
  rice     = fetchTableData(cursor, 'rice')
  salad    = fetchTableData(cursor, 'salad')
  soup     = fetchTableData(cursor, 'soup')
  mainDish = fetchTableData(cursor, 'main_dish')
  userData = fetchTableData(cursor, 'user_data')
  req      = fetchTableData(cursor, 'request')

  writeItemData('rice', rice)
  writeItemData('salad', salad)
  writeItemData('soup', soup)
  writeItemData('main_dish', mainDish)
  writeUserData('user_data', userData)
  writeRequestData('request', req)

def createRandomUserDataFromCSV(filename: str):
  # csvファイルから疑似個人情報データ生成サービスを用いて生成した
  # 氏名, 電話番号, 郵便番号, 住所 を読み込む
  userdata = []
  with open(f"./dummy_data/{filename}", mode="r", encoding='utf-8') as f:
    for line in f:
      userID     = 'M' + str(randrange(0, 10000000000000000)).zfill(16)
      cardNumber = str(randrange(0, 10000000000000000)).zfill(16)
      data       = f"{userID},{line[:len(line)-1]},{cardNumber}"
      userdata.append(data)

  with open("./dummy_data/user_data.csv", mode="w", encoding='utf-8') as f:
    for data in userdata:
      f.write(data + '\n')

def insertInitialUserData(cnx: mysql.connector.MySQLConnection, cursor: cursor.MySQLCursor):
  query = ("INSERT INTO user_data "
           "(user_id, user_name, tel, postal_number, ship_address, card_number) "
           "VALUES (%s, %s, %s, %s, %s, %s)")

  with open("./dummy_data/user_data.csv", mode="r", encoding='utf-8') as f:
    for line in f:
      data    = line.split(',')
      data[5] = data[5].replace('\n', '')
      cursor.execute(query, data)

  cnx.commit()

def dropRequestTable(
  cnx: mysql.connector.MySQLConnection, cursor: cursor.MySQLCursor
):
  query = "DROP TABLE request"
  cursor.execute(query)
  cnx.commit()
```

```
def createRequestTable(
  cnx: mysql.connector.MySQLConnection, cursor: cursor.MySQLCursor
):
  query = ("CREATE TABLE request("
  "branch VARCHAR(256) NOT NULL,"
  "request_datetime DATETIME NOT NULL,"
  "request_index MEDIUMINT UNSIGNED NOT NULL,"
  "request_user_id VARCHAR(17) NOT NULL,"
  "rice VARCHAR(256) NOT NULL,"
  "soup VARCHAR(256) NOT NULL,"
  "salad VARCHAR(256) NOT NULL,"
  "main_dish VARCHAR(256) NOT NULL,"
  "total_price DECIMAL(16, 2) NOT NULL,"
  "PRIMARY KEY(branch, request_datetime, request_index))"
  )

  cursor.execute(query)
  cnx.commit()
```

プログラム5. 実験に使用したスクリプト