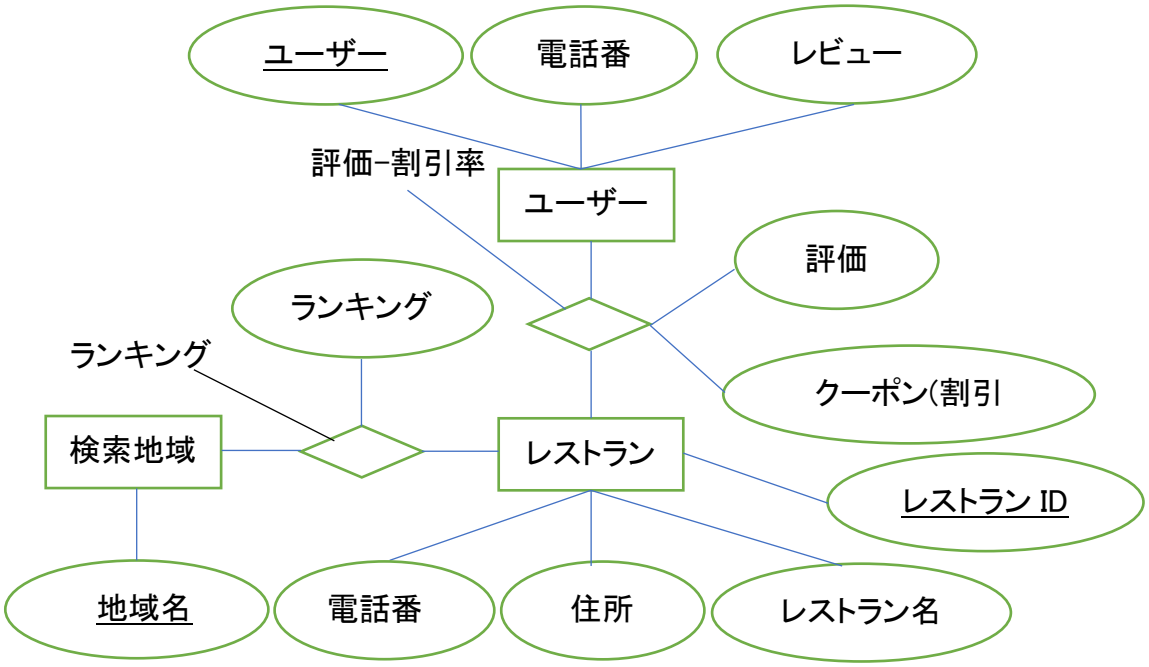
計算機科学実験及演習 4(データベース)課題 5

1029-33-0786
松井 玲

1. 設計した ER 図

ユーザーは名前と電話番号を持ち、ユーザー名は登録時に他ユーザーとの重複を許さないので、キー属性となる。
ユーザーは各レストランに対し評価（点数）を行い、さらにレストランはユーザーに対しクーポン（割引率）をもつ関連集合が存在する。評価点は 0 から 100 の間の整数とする。
レストランは ID、名前、住所、そして電話番号を持ち、ID によって一意に定まる。
レストランは料理に応じて地域別にまとめられ、検索範囲の地域ごとのランキングを関連集合にもつ。ランキングは自然数である。



2. 設計した関係スキーマ

関係：(ユーザー(ユーザー名、電話番号、レビュー数)，Σ (ユーザー))
Σ(ユーザー)は次の一貫性制約を含む
σ1：属性（ユーザー名）に含まれる値に重複はない
σ2：属性（電話番号）に含まれる値に重複はない
σ3：dom(ユーザー名) = String
σ4：dom(電話番号) = String
σ5：dom(レビュー数) = Integer
関数従属性集合 F={ユーザー名→電話番号、レビュー数
電話番号→ユーザー名、レビュー数}

ユーザー名は各ユーザーに一つで、重複はない。また、電話番号も同様の性質を持つ。

関係:(レストラン(<u>レストラン ID</u>、レストラン名、電話番号、住所),Σ(レストラン))
Σ(レストラン)は次の一貫性制約を含む
σ1:属性集合{レストラン ID}が主キーである。
σ2:dom(レストラン ID) = Integer (1≦レストラン ID)
σ3:dom(レストラン名) = String
σ4:dom(電話番号) = String
σ5:dom(住所) = String

関数従属性集合 F={<u>レストラン ID</u>→レストラン名、電話番号、住所
レストラン名、住所→<u>レストラン ID</u>、電話番号
電話番号→<u>レストラン ID</u>、レストラン名、住所)

レストラン ID は各ユーザーに一つで、重複はない。また電話番号、レストラン名と住所の組も同様の性質を持つ。

関係:(評価-割引率(<u>ユーザー名</u>、<u>レストラン ID</u>、評価、割引率),Σ(評価-割引率))
Σ(評価-割引率)は次の一貫性制約を含む
σ1:属性集合{ユーザー名、レストラン ID}が主キーである。
σ2:dom(ユーザー名) = String
σ3:dom(評価) = Integer (0≦評価≦100)
σ4:dom(割引率) = Integer (0≦割引率≦100)
σ5:dom(レストラン ID) = Integer (1≦レストラン ID)
関数従属性集合 F={<u>ユーザー名</u>、<u>レストラン ID</u>→評価、割引率}

ユーザー名とレストラン名の組が決まれば、そのユーザーがレストランに対して書いた評価も一つに定まる。

関係:(ランキング(<u>レストラン ID</u>、<u>地域名</u>、ランキング),Σ(ランキング))
Σ(ランキング)は次の一貫性制約を含む
σ1:属性集合{レストラン ID、地域名}が主キーである。
σ2:dom(レストラン ID) = Integer (1≦レストラン ID)
σ3:dom(地域名) = String

σ4：dom(ランキング) = Integer (1≦ランキング)

関数従属性集合 F={レストラン ID、地域名→ランキング}
レストラン ID と検索範囲の地域名が決まれば、その地域内でのランキングも一つに定まる。

これらの関係スキーマにおいて多値従属性は存在しない。

3. 関係スキーマの正規化

3-1. 関係：(ユーザー(ユーザー名、電話番号、レビュー数)，Σ (ユーザー))

Σ(ユーザー)は次の一貫性制約を含む
σ1：属性(ユーザー名)に含まれる値に重複はない
σ2：属性(電話番号)に含まれる値に重複はない
σ3：dom(ユーザー名) = String
σ4：dom(電話番号) = String
σ5：dom(レビュー数) = Integer
関数従属性集合 F={ユーザー名→電話番号、レビュー数
電話番号→ユーザー名、レビュー数}

この関係の候補キーは属性(ユーザー名),(電話番号})である。
これは部分関数従属を持たず、データは値であるため第 2 正規形である。
決定項はともに候補キーであるのでボイスコッド正規形である。

3-2. 関係：(レストラン(レストラン ID、レストラン名、電話番号、住所)，Σ (レストラン))

Σ(レストラン)は次の一貫性制約を含む
σ1：属性集合{レストラン ID}が主キーである。
σ2：dom(レストラン ID) = Integer (1≦レストラン ID)
σ3：dom(レストラン名) = String
σ4：dom(電話番号) = String
σ5：dom(住所) = String

関数従属性集合 F={レストラン ID→レストラン名、電話番号、住所
レストラン名、住所→レストラン ID、電話番号
電話番号→レストラン ID、レストラン名、住所)}

この関係の候補キーは属性{レストラン ID}、{電話番号}、{レストラン名、住所}である。
よってこの関係は部分関数従属を持たず、決定項は候補キーであるのでボイスコッド正規形である。

3-3. 関係:(評価-割引率(ユーザー名、レストラン ID、評価、割引率), Σ(評価-割引率))

Σ(評価-割引率)は次の一貫性制約を含む
σ1:属性集合{ユーザー名、レストラン ID}が主キーである。
σ2:dom(ユーザー名) = String
σ3:dom(評価) = Integer (0≦評価≦100)
σ4:dom(割引率) = Integer (0≦割引率≦100)
σ5:dom(レストラン ID) = Integer (1≦レストラン ID)
関数従属性集合 F={ユーザー名、レストラン ID→評価、割引率}

この関係の決定項は主キーであるので、これはボイスコッド正規形を満たしている。

3-4. 関係:(ランキング(レストラン ID、地域名、ランキング), Σ(ランキング))

Σ(ランキング)は次の一貫性制約を含む
σ1:属性集合{レストラン ID、地域名}が主キーである。
σ2:dom(レストラン ID) = Integer (1≦レストラン ID)
σ3:dom(地域名) = String
σ4:dom(ランキング) = Integer (1≦ランキング)

関数従属性集合 F={レストラン ID、地域名→ランキング}
この決定項は主キーであるのでボイスコッド正規形を満たしている。

4. 正規化の手法

3NF や BCNF でない設計から BCNF に変換するためには、以下のステップとアプローチが考えられる。

1. **候補キーの確認:**
    - **BCNF に変換するためには、まずテーブル内の候補キーを確認する。候補キーは、一意性制約を満たす属性または属性の組み合わせである。**

2. 関数従属性の特定:
    - テーブル内の関数従属性を特定する。
3. 部分関数従属性の解決:
    - テーブル内に部分関数従属性が存在する場合、それらを解決する。これを行うためには、新しいテーブルを作成し、部分関数従属性の属性を移動する。
4. 関数従属性の分解:
    - BCNF に変換するために、関数従属性を分解する、つまり関数従属性を満たす新しいテーブルを作成する。
    - 次の条件を満たす属性集合 X,Y,Z($\subseteq V$)を選ぶ
        1. XYZ = V
        2. $\Sigma \vDash v$ X→Y
        3. 各 A∈Z に対して $\Sigma \nvDash v$ X→A
    - 属性集合を(XY),(XZ)に分割し、R 中に W⊆W' なる二つの関係スキーマが存在すれば W を削除する

この手法を用い続ければ BCNF に変換することができる。

5. テーブルの定義とデータの挿入

① 表の定義において，キーの指定により保持できる関数従属性や正規形について考察しなさい．表を正規化していく（例えば分解法や合成法）中で，注目するキーや関数従属性で結果がどのように変化するか（どの正規形になるか，どの関数従属性が保持されるか）を考察しなさい．

まず、分解法は上記 4 で述べた正規化の手法であり、BCNF まで正規化することができるが、テーブルの分解の過程で関数従属性が保持されない場合がある。X→Y が存在するにもかかわらず X と Y のテーブルを分割してしまう場合があるからである。一方で合成法は各関数従属性の両辺を関係スキーマとし、元の関係スキーマのキーを含む関係があれば終了する。この手法は関数従属性を保持できるが第3正規形までの合成しか保証できない。

② 課題3で設計した関係スキーマに基づいて関係表を定義しなさい．定義するための SQL 文を示しなさい．なお，各関数従属性が保持されることを文章で説明しなさい。

関係：(ユーザー(ユーザー名、電話番号、レビュー数)，Σ (ユーザー))
関数従属性集合 F={ユーザー名→電話番号、レビュー数
電話番号→ユーザー名、レビュー数}

```
CREATE TABLE Users (
 UserName TEXT NOT NULL,
 PhoneNumber TEXT NOT NULL,
ReviewCount INTEGER,
PRIMARY KEY (UserName, PhoneNumber),
 UNIQUE (UserName),
UNIQUE (PhoneNumber)
);
```

図 1：Users テーブルの定義文

関係：(レストラン(レストラン ID、レストラン名、電話番号、住所)，Σ (レストラン))
関数従属性集合 F={レストラン ID→レストラン名、電話番号、住所
レストラン名、住所→レストラン ID、電話番号
電話番号→レストラン ID、レストラン名、住所)}

```
CREATE TABLE Restaurants (
    RestaurantID INTEGER PRIMARY KEY,
    RestaurantName TEXT,
    PhoneNumber TEXT,
    Address TEXT
);
```

図 2：Restaurants テーブルの定義文

関係：(評価-割引率(ユーザー名、レストラン ID、評価、割引率)，Σ (評価-割引率))
関数従属性集合 F={ユーザー名、レストラン ID→評価、割引率}

```
CREATE TABLE RatingsDiscounts (
    UserName TEXT,
    RestaurantID INTEGER,
    Rating INTEGER,
```

```
    DiscountRate INTEGER,
    PRIMARY KEY (UserName, RestaurantID),
    FOREIGN KEY (UserName) REFERENCES Users(UserName),
    FOREIGN KEY (RestaurantID) REFERENCES Restaurants(RestaurantID)
);
```

図3：RatingDiscounts テーブルの定義文

関係：(ランキング(レストラン ID、地域名、ランキング)，Σ (ランキング))
関数従属性集合 F={レストラン ID、地域名→ランキング}

```
CREATE TABLE Rankings (
    RestaurantID INTEGER,
    RegionName TEXT,
    Ranking INTEGER,
    PRIMARY KEY (RestaurantID, RegionName),
    FOREIGN KEY (RestaurantID) REFERENCES Restaurants(RestaurantID)
);
```

図4：Rankings テーブルの定義文

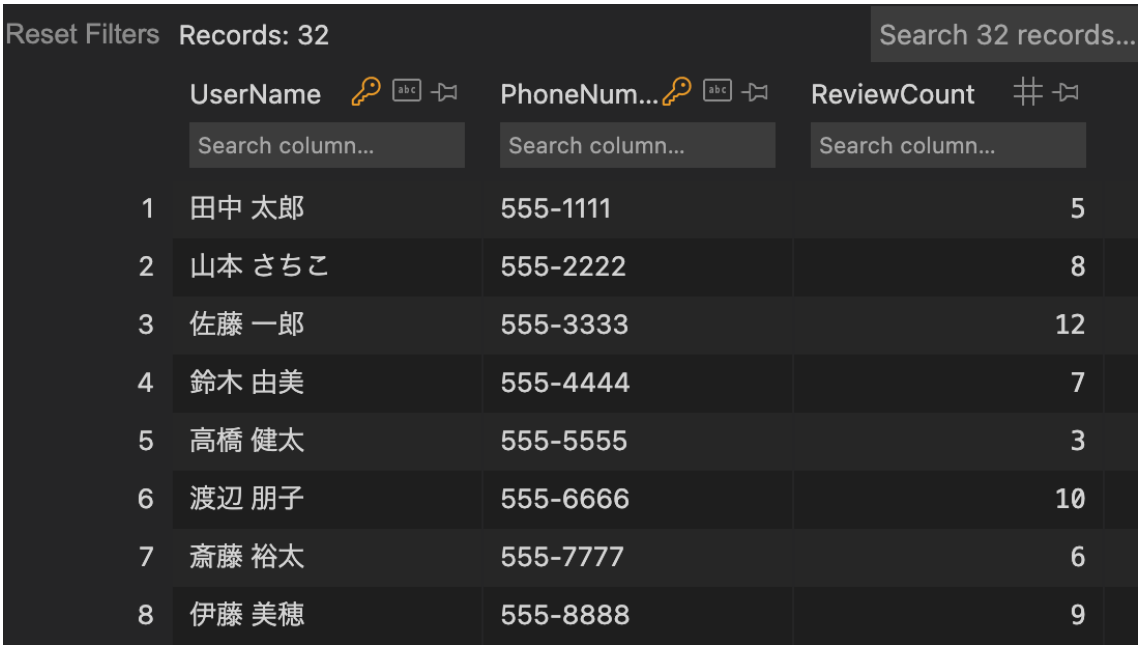
これらの SQL 文で関係表を定義することができる。
また、関係従属性集合に含まれる関数従属性は分割されていないため、保存されている。

③　データを作成して，上記の表に挿入しなさい．データを挿入するための SQL 文を示しなさい．また，データを挿入した表の出力（先頭の一部で結構）を示しなさい．

以下にデータ挿入文と挿入結果（の一部）を示す

```
INSERT INTO Users (UserName, PhoneNumber, ReviewCount)
VALUES
    ('田中 太郎', '555-1111', 5),
    ('山本 さちこ', '555-2222', 8),
    ('佐藤 一郎', '555-3333', 12),
    ('鈴木 由美', '555-4444', 7),
    ('高橋 健太', '555-5555', 3),
    ('渡辺 朋子', '555-6666', 10),
    ('斎藤 裕太', '555-7777', 6),
    ('伊藤 美穂', '555-8888', 9),
    ('中村 雄大', '555-9999', 15),
```

図 5：Users テーブルにデータ挿入文



Reset Filters　Records: 32　Search 32 records...

	UserName	PhoneNum...	ReviewCount
	Search column...	Search column...	Search column...
1	田中 太郎	555-1111	5
2	山本 さちこ	555-2222	8
3	佐藤 一郎	555-3333	12
4	鈴木 由美	555-4444	7
5	高橋 健太	555-5555	3
6	渡辺 朋子	555-6666	10
7	斎藤 裕太	555-7777	6
8	伊藤 美穂	555-8888	9

図 6：Users テーブルの挿入結果

```
INSERT INTO Restaurants (RestaurantName, PhoneNumber, Address)
VALUES
    ('居酒屋 たんか', '03-1111-2222', '東京都新宿区 1-1-1'),
    ('寿司 まぐろや', '06-2222-3333', '大阪府大阪市中央区 2-2-2'),
    ('焼肉 こだわり家', '075-3333-4444', '京都府京都市東山区 3-3-3'),
    ('ラーメン 一八', '092-4444-5555', '福岡県福岡市博多区 4-4-4'),
    ('カフェ ゆり', '052-5555-6666', '愛知県名古屋市中村区 5-5-5'),
    ('蕎麦 かずのこ', '011-6666-7777', '北海道札幌市中央区 6-6-6'),
    ('うどん やまと', '082-7777-8888', '広島県広島市中区 7-7-7'),
```

図 7：Restaurants テーブルのデータ挿入文

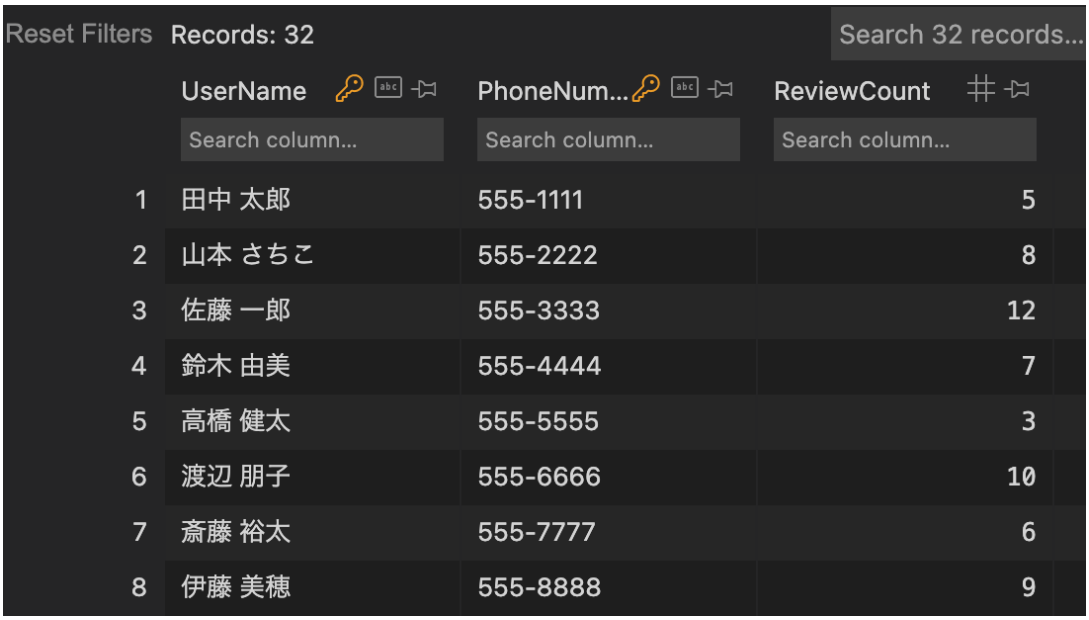


図 8：Restaurants テーブルの挿入結果

```sql
INSERT INTO RatingsDiscounts (UserName, RestaurantID, Rating, DiscountRate)
VALUES
    ('田中 太郎', 1, 90, 10),
    ('山本 さちこ', 2, 85, 15),
    ('佐藤 一郎', 3, 78, 12),
    ('鈴木 由美', 4, 92, 8),
```

図 9：RatingDiscounts テーブルのデータ挿入文

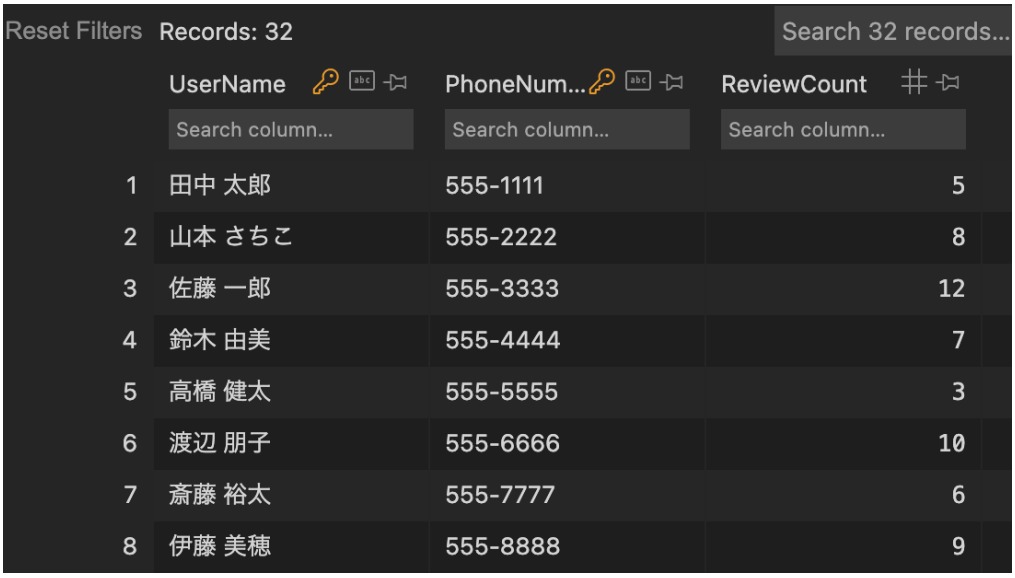


図 10：RatingDiscounts テーブルの挿入結果

```sql
INSERT INTO Rankings (RestaurantID, RegionName, Ranking) VALUES
    (1, '関東', 1),
    (11, '関東', 3),
    (2, '関西', 2),
    (12, '関西', 1),
    (3, '関西', 3
```

図 11：Rankings テーブルのデータ挿入文

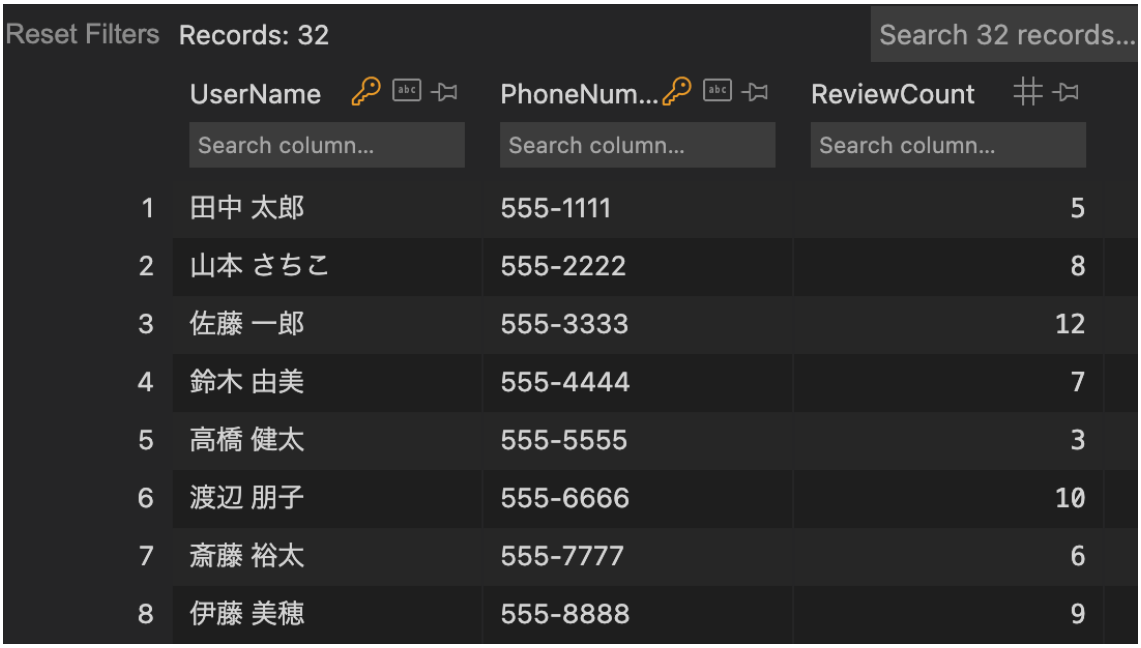


図 12：Rankings テーブルの挿入結果

6. 課題 5 について

① 関係代数の射影および選択に対応する SQL 文

```sql
SELECT UserName, PhoneNumber, ReviewCount
FROM Users
WHERE ReviewCount >= 10;
```

図 13：レビュー数が 10 以上のユーザーを選択する文

これは Users テーブルからレビュー数が 10 以上のユーザーを選ぶ選択文であり、より評価を行なっている信頼性の高いユーザーを見つける時に使えると考えられる。

以下は実行結果である。

UserName	PhoneNumber	ReviewCount
佐藤　一郎	555-3333	12
渡辺　朋子	555-6666	10
中村　雄大	555-9999	15
林　陽子	555-2020	11
木村　浩二	555-5050	10
井上　剛	555-7070	14
田村　勇樹	555-9090	13
岡田　拓也	555-2323	12
河野　啓介	555-6767	11
村上　真紀	555-1011	10
杉山　彩	555-5566	12

図14：選択の実行結果

② 関係代数の自然結合に対応する SQL 文

```
SELECT *
FROM Restaurants
NATURAL JOIN RatingsDiscounts;
```

図15：レストランと評価割引率の自然結合

上記は Restaurants テーブルと RatingDiscount テーブルの自然結合であり、各レストランとユーザーの組に対してその評価と付与される割引率が一目でわかるという点で有用である。
以下はその実行結果である。

RestaurantID	RestaurantName	PhoneNumber	Address	UserName	Rating	DiscountRate
1	居酒屋 たんか	03-1111-2222	東京都新宿区1-1-1	田中 太郎	90	10
2	寿司 まぐろや	06-2222-3333	大阪府大阪市中央区2-2-2	山本 さちこ	85	15
3	焼肉 こだわり家	075-3333-4444	京都府京都市東山区3-3-3	佐藤 一郎	78	12
4	ラーメン 一八	092-4444-5555	福岡県福岡市博多区4-4-4	鈴木 由美	92	8
5	カフェ ゆり	052-5555-6666	愛知県名古屋市中村区5-5-5	高橋 健太	88	12
6	蕎麦 かずのこ	011-6666-7777	北海道札幌市中央区6-6-6	渡辺 朋子	95	5
7	うどん やまと	082-7777-8888	広島県広島市中区7-7-7	斎藤 裕太	80	20
8	焼き鳥 やまもと	045-8888-9999	神奈川県横浜市西区8-8-8	伊藤 美穂	86	14
9	イタリアン オリーブ	089-9999-1010	愛媛県松山市九州町9-9-9	中村 雄大	94	6
10	寿司 さくら	098-1010-1111	沖縄県那覇市10-10-10	小林 未来	75	25
4	ラーメン 一八	092-4444-5555	福岡県福岡市博多区4-4-4	藤田 和也	91	10
4	ラーメン 一八	092-4444-5555	福岡県福岡市博多区4-4-4	岸 朋美	84	15

図16：自然結合の実行結果

③ UNION を含む SQL 文

```
SELECT UserName
FROM RatingsDiscounts
WHERE Rating >= 95
UNION
SELECT UserName
FROM RatingsDiscounts
WHERE Rating <= 20;
```

これは UNION 文を用いて極端に評価が高いか低いユーザーの名前を表示する。

UserName
中島 優子
中村 雄大
井上 剛
伊藤 美穂
佐々木 里香
加藤 美晴
吉田 美香
山口 春美
山田 正人
岡田 拓也

以上は実行結果である。

④　EXCEPT 文を含む SQL 文

```
SELECT UserName
FROM RatingsDiscounts
EXCEPT
SELECT UserName
FROM RatingsDiscounts
WHERE Rating <= 20;
```

上記は 20 点以下の極端に低い評価をつけたユーザー名を排除して表示する SQL 文であり、荒らし目的のユーザーを排除する際に有用であると考えられる。
以下は実行結果である。

UserName
中島　優子
中村　雄大
伊藤　美穂
佐久間　大輔
佐藤　一郎
吉田　美香
大野　千鶴
小林　未来
山本　さちこ
岸　朋美
斎藤　裕太
杉山　彩
桜井　慎一
森田　美奈
河野　啓介
渡辺　朋子

⑤　DISTIINCT を含む SQL 文

```
SELECT DISTINCT UserName
FROM RatingsDiscounts
```

上記は DISTINCT 文を用いて評価を行なったユーザー名を重複なしで取り出すことができ、評価を行なったユーザーを一目で見ることができる。
以下は実行結果である。

UserName
中島　優子
中村　雄大
井上　剛
伊藤　美穂
佐々木　里香
佐久間　大輔
佐藤　一郎
加藤　美晴
吉田　美香
大野　千鶴
小林　未来
山口　春美
山本　さちこ

⑥　AVG を用いた SQL 文

```
SELECT AVG(Rating),RestaurantName
FROM (RatingsDiscounts NATURAL JOIN Restaurants)
WHERE RestaurantID = 5
```

上記は RestaurantID が 5 のレストランの名前と平均の評価を表示する SQL 文である。
以下は実行結果である。

AVG(Rating)	RestaurantName
80.9230769230769	カフェ ゆり

⑦ 副質問を含む SQL 文

```
SELECT RestaurantName
FROM Restaurants
WHERE RestaurantID = (
    SELECT RestaurantID
    FROM (
        SELECT RestaurantID, AVG(Rating) AS AvgRating
        FROM RatingsDiscounts
        GROUP BY RestaurantID
    ) AS Subquery
    WHERE AvgRating = (
        SELECT MAX(AvgRating)
        FROM (
            SELECT AVG(Rating) AS AvgRating
            FROM RatingsDiscounts
            GROUP BY RestaurantID
        ) AS MaxSubquery
    )
);
```

これは副質問を使い平均の評価が最も高いレストラン名を表示する SQL 文である。まず内部のサブクエリを使用して、各レストランの平均評価を計算する。次に、もう一つのサブクエリを使用して、最も高い平均評価を持つ **RestaurantID** を見つけ、最後に、その **RestaurantID** に対応するレストラン名を取得する。
以下は実行結果である。

RestaurantName
居酒屋 たけし

⑧　UPDATE を含む SQL 文

```
UPDATE Users
SET PhoneNumber = '555-2929'
WHERE UserName= '田中 太郎';
```

上記はユーザーの電話番号を変更する UPDATE 文であ。
以下は実行結果である。

```
1    田中 太郎              555-2929
```

⑨　ORDER BY を含む SQL 文

```
SELECT Restaurants.RestaurantName, AVG(RatingsDiscounts.Rating) AS
AverageRating
FROM Restaurants
JOIN RatingsDiscounts ON Restaurants.RestaurantID =
RatingsDiscounts.RestaurantID
GROUP BY Restaurants.RestaurantName
ORDER BY AverageRating DESC;
```

上記はレストランごとの平均評価を高い順に表示する ORDER BY を用いた文である。
以下はその実行結果である。

RestaurantName	AverageRating
居酒屋 たけし	95.1666666666667
イタリアン オリーブ	89.1
ラーメン 一八	88.7142857142857
居酒屋 たんか	86.6
カフェ ゆり	80.9230769230769
天ぷら かわもと	79.875
寿司 さくら	76.0
カフェ ほんだ	75.1111111111111
焼き鳥 やまもと	72.1111111111111
寿司 まぐろや	72.0
焼肉 こだわり家	69.3636363636364
おでん やすだ	69.2
焼き肉 こだわり	65.375
蕎麦 かずのこ	62.2222222222222
寿司 やまさ	60.3
居酒屋 わかば	54.6
寿司 まつもと	49.5
ラーメン やすもと	39.0
うどん やまと	15.9090909090909

⑩　CREATE VIEW を含む SQL 文

```
CREATE VIEW AverageRatings AS
SELECT Restaurants.RestaurantName, AVG(RatingsDiscounts.Rating) AS
AverageRating
FROM Restaurants
JOIN RatingsDiscounts ON Restaurants.RestaurantID =
RatingsDiscounts.RestaurantID
GROUP BY Restaurants.RestaurantName;
```

上記は平均評価を AverageRatings として表にする SQL 文である。これを用いると今までの評価を用いる文が書きやすくなる。
以下は実行結果である。

RestaurantName	AverageRating
うどん やまと	15.9090909090909
おでん やすだ	69.2
イタリアン オリーブ	89.1
カフェ ほんだ	75.1111111111111
カフェ ゆり	80.9230769230769
ラーメン やすもと	39.0
ラーメン 一八	88.7142857142857
天ぷら かわもと	79.875
寿司 さくら	76.0
寿司 まぐろや	72.0
寿司 まつもと	49.5
寿司 やまさ	60.3
居酒屋 たけし	95.1666666666667
居酒屋 たんか	86.6
居酒屋 わかば	54.6
焼き肉 こだわり	65.375
焼き鳥 やまもと	72.1111111111111
焼肉 こだわり家	69.3636363636364
蕎麦 かずのこ	62.2222222222222