計算機科学実験及演習4（データベース）課題6

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。

検索地域

評価

ユーザー

レストラン

評価-割引率

ランキング

1. **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（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）

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（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}

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（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}

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。

1. **関係スキーマの正規化**
   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。

* 1. **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。

* 1. **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。

* 1. **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。

1. **索引による高速化**

各テーブルに対して有効であると考えられる索引について考察する。またそのSQL文を挿入する。

4-1 Userテーブル

属性ユーザー名は検索数が多く、また更新頻度も多くない。そのため、主キーに対する主キーインデックスを作成することが有効である。これにより、データの一意性が保たれ、検索パフォーマンスが向上すると考えられる。

電話番号、レビュー数は更新頻度が多く、また検索頻度もそこまで多くないと考えられるため索引を作成することはあまり有効でないと言える。

* -- 主キーであるRestaurantIDにクラスタ化インデックスを作成
* CREATE UNIQUE CLUSTERED INDEX idx\_Restaurant\_RestaurantID ON Restaurant (RestaurantID);
* -- PhoneNumberとRestaurantNameに非クラスタ化B-Treeインデックスを作成
* CREATE INDEX idx\_Restaurant\_PhoneNumber ON Restaurant (PhoneNumber);
* CREATE INDEX idx\_Restaurant\_RestaurantName ON Restaurant (RestaurantName);

4-2 Restaurantテーブル

* **RestaurantID** が主キーであり、一意である。クラスタ化インデックスが作成されるのが一般的である。
* **PhoneNumber** や **RestaurantName** はよく検索されるため、それぞれに非クラスタ化インデックスを追加で構築することが有効である。それらは特定の検索が行われるため、B-Treeインデックスで実装することが有効である。

-- 主キーであるUserNameとRestaurantIDにクラスタ化インデックスを作成

CREATE UNIQUE CLUSTERED INDEX idx\_RatingDiscounts\_UserName\_RestaurantID ON RatingDiscounts (UserName, RestaurantID);

-- RatingとDiscountRateに非クラスタ化B-Treeインデックスを作成

CREATE INDEX idx\_RatingDiscounts\_Rating ON RatingDiscounts (Rating);

CREATE INDEX idx\_RatingDiscounts\_DiscountRate ON RatingDiscounts (DiscountRate);

-- 外部キー制約によるインデックス（ユーザー名とレストランIDに関連する外部キー制約インデックス）

4-3 RatingDisocountsテーブル

* **PRIMARY KEY (UserName, RestaurantID)** は主キークラスタ化インデックスとして成立する。
* **Rating** または **DiscountRate** は一度設定されると更新される可能性は高くないが、検索頻度は高い。よってB-Tree検索で特定のユーザーの評価を検索する性能を向上させることができる。
* 外部キー制約は、データの整合性を維持するために重要ですが、通常、デフォルトで外部キー制約に関連するインデックスを作成することによって、ユーザー名またはレストランIDを用いた評価検索の性能が向上するため非常に有用である。その場合範囲検索を用いることもあるのでR-Treeインデックスを使うことが考えられる。
* 主キーであるUserNameとRestaurantIDにクラスタ化インデックスを作成
* CREATE UNIQUE CLUSTERED INDEX idx\_RatingDiscounts\_UserName\_RestaurantID ON RatingDiscounts (UserName, RestaurantID);
* -- RatingとDiscountRateに非クラスタ化B-Treeインデックスを作成
* CREATE INDEX idx\_RatingDiscounts\_Rating ON RatingDiscounts (Rating);
* CREATE INDEX idx\_RatingDiscounts\_DiscountRate ON RatingDiscounts (DiscountRate);
* -- 外部キー制約によるインデックス（ユーザー名とレストランIDに関連する外部キー制約インデックス）
  1. Rankingsテーブル

**・ PRIMARY KEY (RegionName, RestaurantID)**は主キークラスタ化インデックスとして成立する。

* 属性Rankings検索頻度が高くさらに範囲検索の可能性が高い。よってR-Treeインデックスを設定して検索性能を向上することができる。
* -- 主キーであるRegionNameとRestaurantIDにクラスタ化インデックスを作成
* CREATE UNIQUE CLUSTERED INDEX idx\_Rankings\_RegionName\_RestaurantID ON Rankings (RegionName, RestaurantID);
* -- RankingsテーブルにR-Treeインデックスを設定