

1. 基礎理論

1.1. 第1正規形関連

1.1.1. 第1正規形の条件

- 繰り返し項目がない
- 繰り返しグループがない
- 非単純定義域がない

1.1.2. 第1正規形だが第2正規形ではない理由

部分関数従属の存在を指摘

- 候補キー【 】の一部である【 】に非キー属性の【 】が部分関数従属するため。
- 候補キー【 】に非キー属性の【 】が完全関数従属しないため。
お好み & 字数に余裕があれば第1正規形の条件を満たすことも書く。

1.1.3. 部分関数従属の更新時異常の指摘

$R(\underline{A}, \underline{B}, C, D)$

$\{A, B\} \rightarrow D, B \rightarrow C$ の場合

- 既に $\{B, C\}$ の組合せを登録していても、重複登録しなければならない。
- $\{B, C\}$ の組合せが複数のタプルにあるとき、 B, C を重複更新しなければならない。
- A の値が決まっていない (NULL) ため、事前に B, C を登録できない。

1.2. 第2正規形関連

1.2.1. 第2正規形の条件

- 部分関数従属性がない
- 第1正規形かつ非キー属性が候補キーに完全関数従属する

1.2.2. 推移的関数従属の指摘

- $\{ \text{候補キー} \} \rightarrow \{ \text{候補キー以外} \} \rightarrow \{ \text{非キー属性} \}$ を 指摘する。

1.2.3. 推移的関数従属の更新時異常

$R(\underline{A}, \underline{B}, C, D)$

$\{A, B\} \rightarrow C \rightarrow D$ の場合

- 既に $\{C, D\}$ の組合せを登録していても、重複登録しなければならない。
- $\{C, D\}$ の組合せが複数のタプルにあるとき、 C, D を重複更新しなければならない。
- $\{A, B\}$ の値が決まっていないので、事前に C, D を登録することができない。

1.3. 第3正規形関連

1.3.1. 第3正規形の条件

- 第2正規形の条件を満たし、かつ推移関数従属性がない
- 第2正規形かつ非キー属性がどの候補キーにも推移関数従属しない

1.3.2. 第3正規形である理由

- 部分関数従属が存在せず、非キー属性がどの候補キーにも推移関数従属しないため

1.4. ボイスコード正規形関連

1.4.1. ボイスコード正規形の条件

- 第3正規形の条件を満たしている。
- すべての属性が候補キーに完全関数従属している。

1.5. 第4正規形関連

1.5.1. 第4正規形の条件

$R(X, Y)$

- $X \rightarrow Y$ は自明な多値従属である。
- X は関係 R のスーパーキーである。

1.5.2. ボイスコード正規形だが第4正規形ではない理由

$A \rightarrow X \mid \{Y, Z\}$ の場合

- 自明でない多値従属性が存在し、かつ、 A は候補キー【 】のスーパーキーではない。

第4正規形への分割： $R_1(\underline{A}, \underline{X}), R_2(\underline{A}, \underline{Y}, \underline{Z})$

1.6. 基礎理論の見直しポイント

- 図の表題に「主な」関数従属性、「未完成」がある場合は、再度、属性の意味と制約を確認する。
- 一意に、毎、別などのキーワードを見落としていないか確認する。
- 非キー属性の指摘時に、属性の列記に{ }はいらない。
- 非キー属性を例示する必要があつてかつ非キー属性が多い場合、後の記述でAA~DDもしくはAA等と省略できる。
- 承認者コード1, 承認者コード2, 承認者コード3, 承認者コード4とかをまとめて、承認者コード1~4と省略できる。
- 当該スキーマの属性で、複数の属性を囲っている四角は、候補キーかもしれない。疑うべし。
- 関数従属性の図がない場合は、とりあえず余白に書いて確認すべし。
- 第 正規形の「第」を忘れない。(解答用紙に書いてあるかも)

《 午後 解答にあたって 》

1. 関係スキーマに主キー, 外部キーを明示するのか?
 - 図と同様な関係スキーマの形式で 図に合せる
 - 図のように関係スキーマの形式で 図に合せる
 - 関係スキーマについて・・・適切な関係に分割した結果を示せ 図に合せる
 - 関係スキーマで示せ 関係スキーマと題した図があれば図に合せる。(無ければ明示する)
2. 候補キーと主キーの使い分け
 - 基礎理論だったら候補キー
 - データベース設計だったら主キー

2. SQL

2.1. 外結合 (ON)

2.1.1. H15午後I問2 設問1(1)

```
SELECT 部門.部門コード, 部門.部門名, COUNT(社員.社員番号),
       COUNT(受講集計.社員番号),
       SUM(年間受講ポイント数 + 繰越ポイント数),
       SUM(使用ポイント数)
FROM 部門, 社員 LEFT JOIN
     (SELECT 社員番号, SUM(受講ポイント数) AS 使用ポイント数
      FROM 受講, コース
      WHERE 受講.コースコード = コース.コースコード
      AND 受講開始年月日 BETWEEN '2002-04-02' AND '2002-09-30'
      GROUP BY 社員番号 ) 受講集計
     ON 社員.社員番号 = 受講集計.社員番号
WHERE 部門.部門コード = 社員.部門コード
GROUP BY 部門.部門コード, 部門.部門名
ORDER BY 部門.部門コード
```

```
FROM A, B, C LEFT JOIN
     (SELECT X, ... (省略) ... GROUP BY X ) AS R ON C.X = R.X
FROM の最後の表名は結合されるテーブルだということを忘れるな。
```

2.2. IN演算子

2.2.1. H15午後 問2 設問1(2)

```
SELECT コース.コースコード, コース.コース名
FROM コース, 必修コース, 社員
WHERE 社員番号 = 123456
AND 必修コース.コースコード = コース.コースコード
AND (必修コース.担当職務コード = 社員.担当職務コード1 OR
     必修コース.担当職務コード = 社員.担当職務コード2)
AND コース.コースコード NOT IN
     (SELECT コースコード
      FROM 受講
      WHERE 社員番号 = 123456)
```

基本は、WHERE 列名A NOT IN (SELECT 列名A FROM ~)

2.3. EXIST演算子

2.3.1. H12午後 問3設問3

```
SELECT * FROM 施策
```

```
WHERE EXISTS
```

```
(SELECT * FROM 社員, 課, 施策グループ
```

```
WHERE :USER = 社員.ユーザID
```

```
AND 社員.所属課コード = 課.課コード
```

```
AND 課.所属部コード = 施策グループ.参画部コード
```

```
AND 施策.施策ID = 施策グループ.施策ID )
```

SELECT * が標準。存在の有無を条件としているため、列名は無関係。

2.4. ANY演算子

x = ANY(副問合せ)と x IN(副問合せ),

x <> ANY(副問合せ)と x NOT IN(副問合せ)は同じ意味。

2.5. 権限 (GRANT)

2.5.1. H12午後 問3設問2 (2)

```
GRANT SELECT ON ユーザ管理 TO PUBLIC
```

GRANT 許可する操作 ON テーブル名 TO 誰に

許可する操作 : ALL : 全ての操作

誰に : TO PUBLIC : みんなに

2.6. GROUP BYとHAVING

```
SELECT A, B, SUM( C ) AS X
```

(省略)

```
GROUP BY A, B
```

```
HAVING SUM( C ) > 5
```

SELECT と GROUP BY の整合に注意。 HAVING に別名は使えない。

2.7. ORDER BY

```
ORDER BY [列名/別名/整数] [ASC/DESC], ...
```

□ 降順は DESC。ド忘れしないように。

□ ORDER BY で指定した列名は SELECT でも指定しなければならない。

2.8. その他レアっぽいこと

□ CREATE VIEW (A, B, C,...) AS SELECT ...

□ UPDATE SET A = X WHERE ...

2.9. SQL解答 (見直し) のポイントとポイント

□ a, b, c と順番に埋めていっても良いが、順番に拘らず、分かる所から埋めていって全体の雰囲気をつかんだ方が時間短縮できる。

□ 条件で『 [] '日付' AND '日付' 』, BETWEEN が入るのは言うまでもないが、何の BETWEEN か列名を忘れないように。

□ SELECT と 列名の間は「DISTINCT」。

□ GROUP BY 句は SELECT の列名にあわせること。

□ 必要なテーブル名の修飾を忘れずに。(全部修飾するという解答もあり?)

□ 『副問合せ AS 相関名(別名)』があったら 相関名はどこかで参照されている。

□ 定数で数値以外は ' ' でくること。(コード系は数字でもくるべし)

□ 解答欄で書いた結合条件が解答欄以外で既に指定されていないか確認すること。

□ FROM 句のテーブル名に過不足がないか見直すこと。

□ WHERE の条件文の AND, OR, <, > を再度確認すること。

3. データベース設計

3.1. 定番問題

□ テーブルの正規化 (分割)

□ 不足している列名の補完

□ 新規要件に対応するためのテーブルの追加

・ 組合せの管理

・ 時系列性の保持

□ 人事異動・組織変更が問題になる。

□ 多対多を管理するテーブル (連関テーブル) の追加

□ スーパータイプ/サブタイプ/カテゴリ識別子

□ 集計テーブルの追加

□ 要件で提示されている情報でテーブルを一意に特定できるかも注意。

□ 正規化崩しの理由

□ 履歴管理は他のテーブルの思想と合せる。

(他のテーブルと思想を合せる。開始だけ/開始と終了の両方)

3.2. 見直しポイント

□ テーブル名, 列名は問題文から引用すべし。問題文中に異音同意語がないか確認すること。

□ 「 ごと」の は主キーに含まれているか?

□ 外部キーの明示が必要かどうか再確認すること。

□ 列名「 番号」はすべて外部キーとは限らない。