データベースシステム
最終レポート

2024 年 1 月 10 日 (水)

6321120

横溝　尚也

1　課題 1

1.1　課題内容

LETUS のような講義管理システムを考える．
必要な項目として学生（学籍番号と名前，etc），履修，講義，担当教員等の情報がある．
これらを管理する上で異状がないテーブルを作成し，作成途中の正規化（第 1，2，3）の流れを説明しなさい．
※ 項目は上記以外にも自由に追加して良い．（学科，実施年，講義の部屋など）

1.2　正規化前のデータベースについて

授業スライド内の履修システムのデータベース項目にいくつかの項目を加え、今回自分が設定したデータベース項目は以下のようなデータを使用した。

学生名	学籍番号	学科 (学科コード)	科目コード (科目名)	教員番号	教員名	評点

表 1: データベースの項目

講義管理システムの前提として以下の状況下でのデータベース運用を考える。

- 学生は必ず一つの学科に所属する
- 学生は複数の科目を受講する可能性がある
- 科目は一人の教員が担当
- 教員は複数科目担当する可能性がある

既存のデータは以下を用意した。わかりやすいデータベース作成のために教員名 (敬称略), 学科 ID, 学籍番号などは理科大 (情報計算科学科) に関する名前を使用した。

学生名	学籍番号	学科 (学科コード)	科目コード (科目名)	教員番号	教員名	評点
佐藤	6321000	数学科 (MA)	1(データベースシステム)	100	松澤	60
			2(機械学習)	101	桂田	69
鈴木	6321001	物理学科 (PH)	1(データベースシステム)	100	桂田	62
			2(機械学習)	101	桂田	70
斎藤	6321002	物理学科 (PH)	1(データベースシステム)	100	松澤	62
高橋	6321003	情報科学科 (IS)	1(データベースシステム)	100	松澤	70
田中	6321004	建築学科 (AR)	2(機械学習)	101	桂田	70
伊藤	6321105	機械工学科 (ME)	2(機械学習)	101	桂田	80

表 2: データベースの内容

上記の設定の下、正規化 1〜3 を行っていく。

1.3 第一正規化

第 1 正規化で行うことは以下である。

- 直積集合の排除
- べき集合の排除
- 主キーの選出

1.3.1 直積集合の排除

各セルに一つの値のみ格納する形にする。以下が直積集合を排除した後のデータベースである。

学生名	学籍番号	学科	学科コード	科目コード	科目名	教員番号	教員名	評点
佐藤	632100	数学科	MA	1	データベースシステム	100	松澤	60
				2	機械学習	101	桂田	69
鈴木	6321001	物理学科	PH	1	データベースシステム	100	桂田	62
				2	機械学習	101	桂田	70
斎藤	6321002	物理学科	PH	1	データベースシステム	100	松澤	62
高橋	6321003	情報科学科	IS	1	データベースシステム	100	松澤	70
田中	6321004	建築学科	AR	2	機械学習	101	桂田	70
伊藤	6321105	機械工学科	ME	2	機械学習	101	桂田	80

表 3: 直積集合を排除したデータベースの項目

1.3.2 べき集合の排除

繰り返し項目に関してセルを分離する。以下がべき集合を削除した後のデータベースである。

学生名	学籍番号	学科	学科コード	科目コード	科目名	教員番号	教員名	評点
佐藤	6321000	数学科	MA	1	データベースシステム	100	松澤	60
佐藤	6321000	数学科	MA	2	機械学習	101	桂田	69
鈴木	6321001	物理学科	PH	1	データベースシステム	100	桂田	62
鈴木	6321001	物理学科	PH	2	機械学習	101	桂田	70
斎藤	6321002	物理学科	PH	1	データベースシステム	100	松澤	62
高橋	6321003	情報科学科	IS	1	データベースシステム	100	松澤	70
田中	6321004	建築学科	AR	2	機械学習	101	桂田	70
伊藤	6321105	機械工学科	ME	2	機械学習	101	桂田	80

表 4: べき集合を排除したデータベースの項目

1.3.3 主キーの選択

初めに候補キー (タプルが一位に定まる列の組み合わせ) を選ぶ。以下が候補キーの組み合わせの一覧である。
学生名は氏名が同じ人が存在しないとも限らないため一意に定まらないと判断した。

- (学籍番号, 科目コード)
- (学籍番号, 科目名, 評定)
- (学籍番号, 教員名, 評定) など

この中から今回は (学籍番号,科目コード) を主キーとする。
第一正規化のみではデータベースへの追加・削除・修正を行った際にデータの不整合が起きてしまう。そのため第２・３正規化を行う。

1.4 第二正規化

主キーの一部に従属する「部分関数従属性」を抜き出す。以下が部分関数従属の関係になっている項目である。

- 学籍番号 → 学生名
- 学籍番号 → 学科
- 学籍番号 → 学科コード
- 科目コード → 科目名
- 科目コード → 教員番号
- 科目コード → 教員名

以上を考慮して第２正規化を施した後のデータベースは以下である。

学籍番号	学生名	学科	学科コード
6321000	佐藤	数学科	MA
6321001	鈴木	物理学科	PH
6321002	斎藤	物理学科	PH
6321002	高橋	情報科学科	IS
6321003	田中	建築学科	AR
6321104	伊藤	機械工学科	ME

表 5: 学生テーブル

科目コード	科目名	教員番号	教員名
1	データベースシステム	100	松澤
2	機械学習	101	桂田

表 6: 科目テーブル

学籍番号	科目コード	評定
6321000	1	60
6321000	2	69
6321001	1	62
6321001	2	70
6321002	1	62
6321003	1	70
6321004	2	70
6321005	2	80

表 7: 履修テーブル

この正規化によって授業名の変更、授業担当者の変更、学生の学部の変更 (現実的にあまりないですが) が発生したときに一部のタプルを変更するだけで整合性を保ちながらデータベースの修正が可能となる。

1.5 第三正規化

主キー以外の属性（または属性の組）に従属する推移的関数従属性を抜き出す。以下が推移的関数従属の項目である。

- 学籍番号 → 学科コード → 学科
- 科目コード → 教員番号 → 教員名

これらに対して第 3 正規化を施したデータベースが以下である。

学籍番号	学生名	学科コード
6321000	佐藤	MA
6321001	鈴木	PH
6321002	斎藤	PH
6321002	高橋	IS
6321003	田中	AR
6321104	伊藤	ME

表 8: 学生テーブル

学科コード	学科名
MA	数学科
PH	物理学科
IS	情報科学科
AR	建築学科
ME	機械工学科

表 9: 学科テーブル

科目コード	科目名	教員番号
1	データベースシステム	100
2	機械学習	101

表 10: 科目テーブル

教員番号	教員名
100	松澤
101	桂田

表 11: 教員テーブル

学籍番号	科目コード	評定
6321000	1	60
6321000	2	69
6321001	1	62
6321001	2	70
6321002	1	62
6321003	1	70
6321004	2	70
6321005	2	80

表 12: 履修テーブル

これにより新たな教員や、新学科設立などが発生したときにも一部のテーブルを変更するだけでデータベースが矛盾なく更新が可能となった。

2 課題 2

2.1 課題内容

自身の趣味や取り組んでいること（学業以外）で，データベース管理が必要な情報を抜き出し，以下の設問 1〜3 に答えなさい．

※ リレーショナルデータベースで管理すべき題材にすること．

※ 可能な限り他の学生さんと被らないユニークな対象が望ましい．

1. 正規化したテーブルを作成しなさい．

2. 今回の題材を選んだ理由について説明しなさい．

3. 上記で設計したデータベースで想定される一番多い検索を説明し，その SQL(Select) 文を説明しなさい．

2.2 正規化したテーブルの作成

背番号 uniformNum	試合コード matchCode	得点数 PT	3P 試投数 try_3P	3P 成功数 suc_3P	2P 試投数 try_2P	2P 成功数 suc_2P	出場時間 MIN
4	1	10	5	1	6	3	17:22
5	1	6	0	0	5	3	22:20
6	1	2	0	0	3	0	18:16
4	2	5	0	0	10	2	18:52
5	2	8	1	0	7	4	17:35
6	2	5	2	0	9	2	16:58

表 13: match_stats

試合コード matchCode	試合日 matchDate	対戦校コード 1 school1	対戦校コード 2 school2
1	8 月 2 日	1	2
2	11 月 15 日	1	3

表 14: school

学校コード shoolCode	学校名 schoolName
1	県立春日部高校
2	A 高校
3	B 高校

表 15: schoolCode

背番号 uniformNum	選手名 name
4	横溝 尚也
5	A さん
6	B さん

表 16: player

2.3 選んだ題材理由

今回自分が採用したデータベースはバスケのスタッツ (試合結果) である。スポーツの戦略を立てるとき、過去の相手チームのデータから弱点を見つけ、戦術を立てることがある。特に現代では統計データの分析結果から戦略を立てることがプロの世界でよく行われている。そこで自分は小中高バスケをしており、バスケの試合結果をデータベースとして扱い、データ解析を行うためのデータベース作成をしようと考えた。
高校バスケの大会であるウィンターカップ (野球で言う甲子園) における全国大会決勝のスタッツ (試合結果) をデータベースにしようと試みたが、公式サイトに転載・複製禁止とあったため、この課題で使うことができなかった。そこで代わりに自分の高校時代のスタッツを今回の題材とした。２試合での３名の試合データをデータベースとし、自分以外の人の名前や、相手高校名は伏せた。実際にあった試合結果を参考にしているが、付加情報として試合日 (正確にはわからないため設定した) などの情報も含めた。

2.4 想定されるデータベース検索

バスケでは得点をどこから決めているか (3P か、2P) を分析することがある。そのため、チームの 3P の成功率を出力する SQL 文を記述する。ただしテーブル名は表に記載されているキャプション名 result とする。

プログラム 1: rateOf3p.sql

```
SELECT
  SUM("try_3P") AS 総試投数,
  SUM("suc_3P") AS 総成功数,
  SUM("suc_3P") * 100 / SUM("try_3P") AS 成功率
FROM
  result;
```

もう一つよく使用されると考える SQL 文を考える。大会 MVP 選手の選定を考える。MVP 選手の算出方法は、その大会における総得点数の一番高い人とする。その時の MVP 選手の出力は以下のような文となる。ただし、得点数や背番号の変数は上記の項目に示した通り、point,uniformNum とする。

プログラム 2: mvp.sql

```
WITH point_rank AS (
  SELECT
    uniformNum,
    SUM(PT) AS sum_points,
    RANK() OVER (ORDER BY SUM(PT) DESC) AS ranking
  FROM
    result
  GROUP BY
    uniformNum
)
SELECT
  uniformNum,
  sum_points
```

```
14 FROM
15     point_rank
16 WHERE
17   ranking = 1;
```

with 句を使用して総得点数を計算し、一時的に情報を保持している。その後ランキングが 1 位の人を where 句の条件とし、背番号と総得点数を出力している。

3 課題 3

3.1 課題内容

トランザクション処理が満たすべき要件として ACID 特性がある．それぞれの特性が満たされない場合の不具合を，具体例を用いて説明しなさい．

3.2 Atomicity(原子性)

原子性 [1] とはある更新処理トランザクション (回復の単位) が正常に終了したときのみ処理を反映し、異常が発生したときは何もなかった状態に戻すこと。つまりある処理単位トランザクションの実行が 0 か 100 で行うこと。これはコミットとロールバックで実現される。以下が原子性が満たされない場合の不具合の例である。

銀行口座間の送金途中での障害
授業で紹介されていた例と同じになりますが一番わかりやすいと感じたのでこれにしました。具体的な設定として残高 10 万円の銀行口座 A から残高 10 万円の銀行口座 B へ 2 万円送金することを考える。このとき一連の処理 (トランザクション) は以下の順番で行われる。

1. 銀行口座 A から 2 万円引き落とす
2. 銀行口座 B に 2 万円振り込む

このトランザクション 1 と 2 の途中で振り込みシステムに障害が発生したとする。このとき原子性が満たされていないとき、トランザクション 1 のみが実行されてしまい、合計の残高が 18 万円となってしまう。この矛盾が大規模な銀行システムなどで障害が発生したときには多くの利用者が被害を受けることになる。そこで原子性が満たされていれば、トランザクション 1 と 2 の途中で障害が発生したときにはトランザクション 1 がロールバックされ、送金処理が一切行われていない段階まで戻る。その後もう一度送金を試みれば正常に送金されることになる。

3.3 Consistency(一貫性)

一貫性 [1] とはトランザクション処理においてデータベース内のデータに矛盾が起きないこと。データの整合性が常に保たれていること。以下が一貫性が保たれない場合の具体例である。

送金金額と残高の矛盾残高 1 万円の銀行口座 A から残高 1 万円の銀行口座 B への 2 万円の送金を試みる。対面で人と人とがお金の受け渡しをするときには、2 万円を”あげるお金がない”ことに簡単に気づくことができる。しかし、送金システムのアルゴリズムは

1. (銀行口座 A の残高)　= (銀行口座 A の残高) - (2 万円)
2. (銀行口座 B の残高) = (銀行口座 B の残高) + (2 万円)

である。このとき一貫性を満たさなければ (銀行口座 A の残高)=-1 万円,(銀行口座 B の残高)=3 万円となっ

てしまい、残高が負というあってはならない状況が生まれてしまう。これを防ぐために残高が負になってしまうような送金に対してはエラーにするなどを行う必要がある。また、SQL 演習で行った”一意制約”や、”非 NULL 制約”もこの一貫性の仕組みである [2]。

3.4 Isolation(隔離性)

隔離性 [1] とは複数のトランザクションが同時に実行した場合と順に実行した場合で処理結果が同じになること。以下が隔離性が保たれない場合の具体例である。

複数の送金が同時に行われたときの不整合
銀行口座 A,B,C の残高がそれぞれ 2 万円とする。銀行口座 A から C へ 5000 円の送金、銀行口座 B から C へ 5000 円の送金が同時に行われたとする。これを人と人とが直接やり取りした場合には A,B,C の残高はそれぞれ 1 万 5000,1 万 5000,3 万円になるはずである。送金システムにおいてのトランザクションは以下の手順で行われるとする。

1. 銀行口座 A から C への送金
2. 銀行口座 B から C への送金

このとき隔離性を満たしていないと、以下の順番で二つのトランザクションが実行されてしまう。

1. 銀行口座 A から 5000 円引き落とす
2. トランザクション 2 を行うためにデータベースを参照
3. C への送金
4. 銀行口座 B から C への送金

トランザクション 1 が行われている最中、つまり銀行口座 A から 5000 円が引かれ、銀行口座 C に今から 5000 円振り込む途中にトランザクション 2 がデータベースを参照する。このときの銀行口座 A,B,C の残高はそれぞれ 1 万 5000,2 万,2 万である。その後トランザクション 2 が行われたときの最終的な残高は 1 万 5000,1 万 5000,2 万 5000 となってしまう。これではデータの整合性が保たれていない。このため隔離性ではトランザクション 1 が完了するまでデータベースを外部から参照できないようにする必要がある。

3.5 Durability(持続性)

持続性 [1] とは一度正常終了 (Commit) したトランザクションは障害が発生したとしてもデータベースから消失しないこと。以下が持続性が保たれない場合の具体例である。

トランザクション完了後の障害の発生
銀行口座間での送金が完了した (コミットした) のちに何らかの障害が発生したとする。原子性はコミットされていないトランザクションについてロールバックすることでデータベースの整合性を保つものであった。しかし、完全にコミットされたトランザクションまではロールバックせずにあらかじめ保存してあるバックアップのようなもので復元するべきである。もしこの持続性が満たされていないとき、どこまでロールバックされたのか不明確となり、再試行するトランザクションがどれなのか曖昧になってしまう。例に挙げた銀行口座間

の送金に関してもこの送金がロールバックした後に実行された状態かわからなければこれは大問題である。

4　課題 4

4.1　課題内容

下図の二つのテーブルを自然結合させたテーブルを作成しなさい．

※ `MySQL` の出力結果でもよい．

Name	Grade
Alice	3
Bob	4
Charlie	2

表 17: 学年テーブル

Name	Dep
Bob	MA
Charlie	PH
Dave	IS

表 18: 学科テーブル

4.2　MySQL による実行結果

二つのテーブルを作成し、それを授業で学習した自然結合の構文を用いて結合した。出力結果は以下である。

```
mysql> select * from grade;
+---------+-------+
| name    | grade |
+---------+-------+
| Alice   |     3 |
| Bob     |     4 |
| Charlie |     2 |
+---------+-------+
3 rows in set (0.00 sec)

mysql> select * from department;
+---------+-----+
| name    | dep |
+---------+-----+
| Bob     | MA  |
| Charlie | PH  |
| Dave    | IS  |
+---------+-----+
3 rows in set (0.00 sec)

mysql> select * from grade natural join department;
+---------+-------+-----+
| name    | grade | dep |
+---------+-------+-----+
| Bob     |     4 | MA  |
| Charlie |     2 | PH  |
+---------+-------+-----+
2 rows in set (0.00 sec)
```

図 1: 二つのテーブル内容とその自然結合

5　参考文献

参考文献

[1] 大滝みや子. 令和 5 年応用情報技術者合格教本

[2] ACID 特性とは
https://wa3.i-3-i.info/word110350.html