工学実験実習IV レポート

SQL データベース

実験日1　2020年7月15日 1〜2コマ目
実験日2　2020年7月21日 1〜2コマ目
実験日3　2020年7月22日 1〜2コマ目
実験日4　2020年7月29日 1〜2コマ目
実験日5　2020年8月05日 1〜2コマ目
提出日　令和2年8月6日

組番号　408
学籍番号　17406

氏名　金澤雄大

1　目的

多くのデータを扱いたいとき, ファイルにデータを保存し読み込むことで多くのデータを扱うことがある. しかしファイルを扱うと, 複数人が同時に編集を扱うことや膨大な数のデータの検索や更新を行うことが難しい. そこでデータベースを利用する. このデータベースとやりとりする言語が SQL(Structured Query Language) である. 本実験では SQL の使い方を学習することを目的とする. また, データベースを効率よく格納し, 拡張性を考慮したテーブルの定義として正規化がある. この正規化について理解し, 実際に適用することを目的とする. さらに, 国や地方公共団体が提供している実際のデータ (オープンデータ) を利用しデータベースを作成することを目的とする. また作成したデータベースの利用方法や有用性を考察することも目的とする.

2　理論・原理

本章では, データベースの概要, および MySQL の基礎, および正規化と ER 図について述べる.

2.1　データベースの概要

「データベース」とはデータが保管されているものという意味である. コンピュータシステムにおける「データベース」はデータベースマネージメントシステム (DBMS) を指すことが多い.DBMS とはデータベースの管理, およびデータの抽出を代表とする読み書きを行うソフトウェアのことである. イメージとしては大量の本がある図書館がデータベースであり, 本の検索や管理を行う図書館司書が DBMS である.DBMS には様々なものがあるが本実験では MySQL を用いる.

MySQL は命令 (クリエという) を発行すると,DBMS を介してデータベースにアクセスし, クエリの要求に答える出力を行う. クエリは大別すると次のような種類のものがある.

- データの格納
- データの検索
- データの操作 (追加, 削除, 更新)
- データの定義と関連付け

2.2　MySQL の基礎

MySQL の基礎を説明するために用語の説明を行う. SQL においてデータを保存するための表を「テーブル」という. 図 1 はテーブルの例を示したものである. 図 1 ではテーブル A とテーブル B の 2 つがあることがわかる. 図 1 に示した, データの項目をカラムという. 図 1 のテーブル A におけるカラムは,ID, 学籍番号, 名前の 3 つである. また,1 行分のデータをレコードという. 図 1 のテーブル A およびテーブル B は 2 つのレコードがあることが読み取れる. 複数のテーブルを集めたものをデータベースと呼ぶ. 図 1 のデータベース D は, テーブル A とテーブル B の 2 つのテーブルをもっているデータベースであることがわかる.

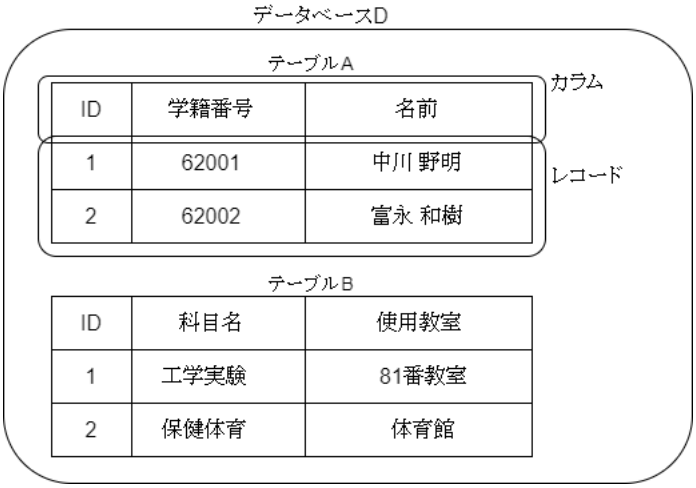

図 1: データベースとテーブル

本実験ではデータベースにおけるデータの格納方法として「リレーショナルデータベース」を用いる. リレーショナルデータベースとはテーブルにカラムが設定され, レコードが記録されたテーブルが相互に関係をとることで, 要求にあったデータを抽出することができるような仕組みを持つデータベースのことである.

また SQL は C 言語や Java のように格納するデータの型が存在する. データ型は数値型, 日付型, 文字列型の 3 つがある. 表 1, および表 2, および表 3 に各型の代表的なデータ型を示す. また, 表 1 に示した数値型の他に, 浮動小数点を扱う FLOAT 型および DOUBLE 型が存在する.

表 1: 数値型の代表例

型	ストレージ (バイト)	最小値	最大値
		符号付き/符号なし	符号付き/符号なし
TINYINT	1	-128	127
		0	255
SMALLINT	2	-32768	32767
		0	65535
MEDIUMINT	3	-8388608	8388607
		0	16777215
INT	4	-2147483648	2147483647
		0	4294967295
BIGINT	8	-9223372036854775808	9223372036854775807
		0	18446744073709551615

表 2: 日付型の代表例

型	内容	型/範囲
DATE	日付部分 (時間部分は含まない)	'YYYY-MM-DD' '1000-01-01'〜'9999-12-31'
DATETIME	日付と時間の両方	'YYYY-MM-DD HH:MM:SS' '1000-01-01 00:00:00'〜'9999-12-31 23:59:59'
TIMESTAMP	日付と時間の両方	'1970-01-01 00:00:01' UTC 〜'2038-01-19 03:14:07' UTC

表 3: 文字列型の代表例

型	内容	範囲
CHAR	固定文字列	0 から 255 までの任意の値
VARCHAR	可変長文字列	0 から 65535 までの値

データベースの構築を行うときに用いる用語について説明する. 図 1 に示したように, 各表のテーブル名およびカラム名が登場する項目そのままの名前で表されているモデルを論理モデルという. 実際にデータベースを構築することを考えると, テーブル名およびカラム名が日本語であると文字コードのトラブルが起きる可能性がある. そこでテーブル名およびカラム名を英数字のみの構成にする. このようなモデルを物理モデルと呼ぶ. 例として, 図 1 のデータベースでは,「名前」を「name」,「科目名」を「subject」に置き換える.

2.3 正規化と ER 図

正規化とはデータベースにデータを効率よく格納し, 拡張性を考慮した, データベースの定義方法のことである. 本実験では第一正規化, 第二正規化, 第三正規化と呼ばれる 3 つの正規化を行う.

まず, 第一正規化について説明する. 表 4 に第一正規化を行う前のデータを示す.

表 4: 正規化する前のデータ

学籍番号	学年	名前	学科	科目	担当者	点数
22421	4	田中 次郎	電子情報工学科	データベース	中村 剛	90
				電子回路	和田 知子	100
22310	4	中村 花子	電子制御工学科	制御工学	江田 純也	87
				材料工学	藤枝 明彦	68
22315	4	成田 翼	電子制御工学科	材料工学	藤枝 明彦	89
21504	3	木村 剛	環境都市工学科	都市計画	内川 早苗	89

第一正規化はレコードをまたいで定義される値がないように 1 レコードごとに分離する. 表 5 に表 4 を第一正規化したテーブルを示す. 表 4 においてレコードをまたいで定義されている学籍番号, 学年, 名前, 学科の 4 つのカラムが, レコードをまたがないように分離されていることが読み取れる.

表 5: 第一正規化したデータ

学籍番号	学年	名前	学科	科目	担当者	点数
22421	4	田中 次郎	電子情報工学科	データベース	中村 剛	90
22421	4	田中 次郎	電子情報工学科	電子回路	和田 知子	100
22310	4	中村 花子	電子制御工学科	制御工学	江田 純也	87
22310	4	中村 花子	電子制御工学科	材料工学	藤枝 明彦	68
22315	4	成田 翼	電子制御工学科	材料工学	藤枝 明彦	89
21504	3	木村 剛	環境都市工学科	都市計画	内川 早苗	89

第一正規化したデータをもとに第二正規化を行う. 第二正規化は主となるカラムとそれに従属するカラムを分離する. ここで, 主となるカラムを主キー (PRIMARY KEY) という. 主キーは重複することを許さない. 例えば, 学籍番号を主として, 学年, 名前, 学科の 3 つが定まる. また, 科目を主として担当者が決まる. さらに, 学籍番号を主と科目に応じて科目の点数が決まる. すなわち, 学生に関するテーブル, 科目に関するテーブル, 点数に関するテーブルの 3 つができあがる. 表 6, および表 7 および, 表 8 にこの 3 つのテーブルを示す.

表 6: 学生に関するテーブル (第二正規化)

学籍番号	学年	名前	学科
22421	4	田中 次郎	電子情報工学科
22310	4	中村 花子	電子制御工学科
22315	4	成田 翼	電子制御工学科
21504	3	木村 剛	環境都市工学科

表 7: 科目に関するテーブル (第二正規化)

科目名	担当者
データベース	中村 剛
電子回路	和田 知子
制御工学	江田 純也
材料力学	藤枝 明彦
都市計画	内川 早苗

表 8: 点数に関するテーブル (第二正規化)

学籍番号	科目名	点数
22421	データベース	90
22421	電子回路	100
22310	制御工学	87
22310	材料力学	68
22315	材料力学	89
21504	都市計画	89

第三正規化は, 第二正規化されたテーブルで従属関係を分離できていない部分を分離する. また, 性別や学科のように選択肢から選ぶ項目を別のテーブルに抜き出す (LGBT の観点から性別を分けようとすると非常に複雑になってしまうためここでは男/女とする). 表 9〜表 12 に第二正規化されたテーブルを第三正規化したテーブルを示す. 第二正規化からの変更点は, 表 10 に示すように学科に関するテーブルを設けて, 学生に関するテーブル (表 9) からは学科 ID で取り扱う. ただしここでの学科は本校の学科を暗に仮定している. こうすることで人間による表記ゆれ (例えば「電子 情報工学科」,「電子情報 工学科」) を防ぐことができる. また, 科目に関するテーブル (表 11) を設け点数に関するテーブル (表 12) からは科目を科目 ID で扱う.

表 9: 学生に関するテーブル (第三正規化)

学籍番号	学年	名前	学科
22421	4	田中 次郎	4
22310	4	中村 花子	3
22315	4	成田 翼	3
21504	3	木村 剛	5

表 10: 学科に関するテーブル (第三正規化)

学科 ID	学科名
1	機械工学科
2	電気電子工学科
3	電子制御工学科
4	電子情報工学科
5	環境都市工学科

表 11: 科目に関するテーブル (第三正規化)

科目 ID	科目名	担当者
10001	データベース	中村 剛
10002	電子回路	和田 知子
10003	制御工学	江田 純也
10004	材料力学	藤枝 明彦
10005	都市計画	内川 早苗

表 12: 点数に関するテーブル (第三正規化)

学籍番号	科目名	点数
22421	10001	90
22421	10002	100
22310	10003	87
22310	10004	68
22315	10004	89
21504	10005	89

第一正規化から第三正規化を行い,4 つのテーブルができた.4 つ程度のテーブルで構成されるデータベースであれば, テーブルの関係がわかるかもしれないが, 実際に扱うテーブルはそれよりも多いと考えられる. テーブルが増えると, テーブルの関係がより複雑になり, 簡単に把握することが難しくなる. そこで,ER 図というものを用いてテーブル同士の関係を図で表すことでテーブルの関係を明快にすることができる. 図 2 および図 3 に第三正規化した 4 つのテーブルを ER 図にしたものを示す. 図 2 は論理モデルにおける ER 図, 図 3 は物理モデルにおける ER 図を示している.ER 図の見方は, 囲いの外側に書かれている文字がテーブル名である. 囲いの上から一段目に書かれているカラムが主キー, 二段目に書かれているカラムが主キーに従属するキーである. また (FK) とは FOREIGN KEY(外部キー) のことで入力を別のテーブルの特定のカラムのデータに制限する. またテーブル間の線はテーブルに依存関係があるか否かを示している.

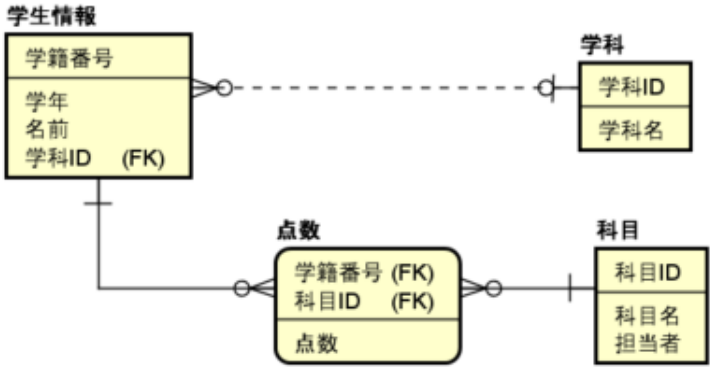

図 2: ER 図の例 (論理モデル)

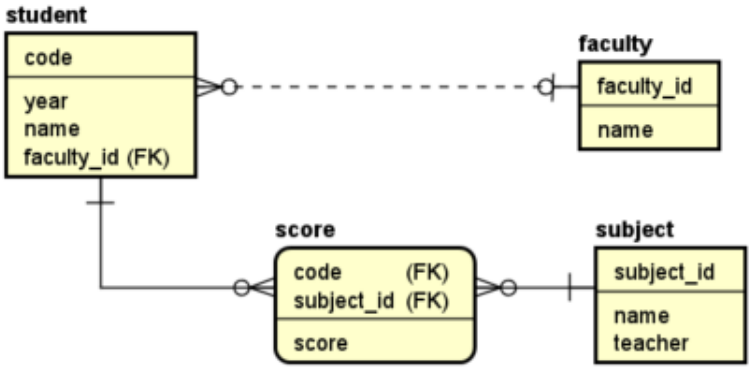

図 3: ER 図の例 (物理モデル)

astah で ER 図を作成する場合, 作成した ER 図からテーブルを作成するクエリを発行することができる. 本実験ではこの機能を利用して MySQL のテーブル定義を行う.

3 実験方法

本章では実験環境, および実験の手順および内容について述べる. 本実験では ER 図の作成に astah を用いる. また,MySQL のソースコードにおいて, 命令は大文字で表記するものとする.

3.1 実験環境

表 13: 実験環境

実験環境を表 13 に示す.

CPU	Intel Core i7-6500 2.50Ghz
メモリ	16.0GB DDR3
OS	Windows 10 Home
MySQL	version8.0.19 Community Server - GPL
astah professional	version 8.2.0/b743f7
文字コード	UTF-8

3.2 実験の手順および内容

実験として授業中に与えられた課題を 4 つ行う. 本節では 4 つの課題の内容について述べる.

3.2.1 課題 1

表 14 に示すテーブルについて第一正規化〜第三正規化を行い ER 図を作成する. ただし条件は次の通りである.

- 担当コードおよび学籍番号は 1 人につき 1 つ付与されていて, ユニークな値である.
- 同じ科目名の科目は複数存在しない.
- 1 つの科目の担当者は必ず 1 名である.
- 成績の入力日は入力した日付とする.
- 学科および校舎は本校の場合を仮定する [1]. 学科の一覧は表 10 に示したものを利用する. 校舎の一覧は表 15 に示す.
- 性別は男/女のどちらかとする.
- 判定は表 16 に示す方法で方法で決定する.

表 14: 学生成績表

学籍番号	学生名	年齢	性別	学科	校舎	科目名	点数	判定	教員コード	教員名	入力日
26401	山川 一郎	18	男	電子情報工学科	電子情報工学科棟	国語 II	65	可	G002	長松次郎	2018-06-10
						物理 I	42	不可	G001	長松 遥	2018-06-11
						現代社会	79	良	G005	高田 真由	2018-06-14
26205	谷川 太郎	19	男	電気電子工学科	電気電子・機械工学科棟	物理 I	76	良	G001	長松 遥	2018-06-11
						電気工学	66	可	E012	三浦 裕	2018-06-12
						信号処理	90	優	E012	三浦 裕	2018-06-12
26132	海土 花子	20	女	機械工学科	電気電子・機械工学科棟	物理 I	87	優	G001	長松 遥	2018-06-10
						機械工学	78	良	M301	上条 雄三	2018-06-11
						計測工学	100	優	M533	西岡 剛	2018-06-12

表 15: 校舎の一覧

校舎 ID	校舎名
1	管理・一般校舎
2	電子制御工学科棟
3	電気電子・機械工学科棟
4	電子情報工学科棟
5	環境都市工学科棟
6	情報教育センター
7	図書館センター
8	体育館

表 16: 成績の判定方法

判定	最小値	最大値
不可	0	59
可	60	69
良	70	79
優	80	100

3.2.2 課題 2

課題 1 で正規化したテーブルおよび ER 図を用いて,MySQL にデータベースおよびテーブルを定義する. また定義したテーブルにレコードを格納するクエリを発行し, 各テーブルに登録したデータの内容を表示する. カラムの物理名, 物理, 型, 必須/非必須 (データが NULL であることを許すか) は表 17 に従うとする.

表 17: カラムの設定

項目	カラム (論理名)	カラム (物理名)	型	必須/非必須
学籍番号	学籍番号	code	CHAR(5)	必須
学生名	名前	name	VARCHAR(40)	必須
年齢	年齢	age	INT	非必須
性別	性別	gender	CHAR(4)	非必須
学科	学科	faculty	VARCHAR(40)	必須
校舎	校舎	place	VARCHAR(40)	必須
科目名	科目名	subject	CVARHAR(40)	必須
教員コード	教員コード	teacher_id	CHAR(4)	必須
教員名	名前	name	VARCHAR(40)	必須
点数	点数	score	INT	必須
判定	判定	result	VARCHAR(6)	必須
入力日	入力日	date	DATE	必須

3.2.3 課題 3

課題 2 で登録したデータに対し, 次に示す 5 つのクエリを発行し, 実行する.

1. 第一正規化したときの表を表示するクエリ.
2. いずれかの科目の判定が「優」である学生の学籍番号, 氏名, 学科を問い合わせるクエリ.
3. 担当者が「高田」, または「三浦」である科目を履修している学生について, 学生名, 科目名, 点数を問い合わせるクエリ.
4. 科目名に「学」が含まれる科目について, 履修している学生名, 科目名, 点数を問い合わせるクエリ.
5. 担当が「三浦」の科目の低近点を問い合わせるクエリ.

3.2.4　課題 4

オープンデータについて, 課題 1 から 3 と同様の処理を行う. すなわちオープンデータについて第三正規化までを行い,ER 図を作成する. さらにデータベースおよびテーブル定義を行うクエリを発行し, 実行する. 最後に課題 3 のようにデータを問い合わせるクエリを発行し, 実行する.

4　課題 1

本章では 3 つの正規化の過程および結果について述べる.

4.1　第一正規化の結果

表 14 に示すデータを第一正規化する. 第一正規化とはレコードをまたいで定義されるカラムをレコードをまたがないように分離する操作であった. 表 14 のデータでは, 学生名, 年齢, 性別, 学科, 校舎の 5 つのカラムがレコードをまたいでいることがわかる. これより, これらのカラムについて, レコードをまたがないように変更する. 表 18 に表 14 のデータを正規化したデータを示す. 表 18 ではレコードをまたいで定義されるカラムがないことがわかる.

表 18: 第一正規化の結果

学籍番号	学生名	年齢	性別	学科	校舎	科目名	点数	判定	教員コード	教員名	入力日
26401	山川一郎	18	男	電子情報工学科	電子情報工学科棟	国語 II	65	可	G002	長松次郎	2018/6/10
26401	山川一郎	18	男	電子情報工学科	電子情報工学科棟	物理 I	42	不可	G001	長松遥	2018/6/11
26401	山川一郎	18	男	電子情報工学科	電子情報工学科棟	現代社会	79	良	G005	高田真由	2018/6/14
26205	谷泉太郎	19	男	電気電子工学科	電気電子・機械工学科棟	物理 I	76	良	G001	長松遥	2018/6/11
26205	谷泉太郎	19	男	電気電子工学科	電気電子・機械工学科棟	電気工学	66	可	E012	三浦裕	2018/6/12
26205	谷泉太郎	19	男	電気電子工学科	電気電子・機械工学科棟	信号処理	90	優	E012	三浦裕	2018/6/12
26132	海土花子	20	女	機械工学科	電気電子・機械工学科棟	物理 I	87	優	G001	長松遥	2018/6/10
26132	海土花子	20	女	機械工学科	電気電子・機械工学科棟	機械力学	78	良	M301	上条雄三	2018/6/11
26132	海土花子	20	女	機械工学科	電気電子・機械工学科棟	計測工学	100	優	M553	西岡剛	2018/6/12

4.2　第二正規化の結果

表 18 のデータを第二正規化する. 第二正規化とは主となるカラムと, それに従属するカラムにデータを分離することであった. 表 18 を見ると, 学籍番号を主として, 学生名, 年齢, 性別, 学科, 校舎の 5 つが定まることがわかる. また科目名を主として, 教員コードと教員名が定まる. 同様に, 学籍番号と科目名の 2 つを主として, 点数, 判定, 入力日の 3 つが定まる. すなわち, 第二正規化によって 3 つのテーブルが出来上がる. 説明のために, それぞれのテーブルを学生テーブル, 科目テーブル, 成績テーブルと呼ぶことにする. 表 19〜表 21 に第二正規化によってできる 3 つのテーブルを示す.

表 19: 学生テーブル (第二正規化)

学籍番号	学生名	年齢	性別	学科	校舎
26401	山川一郎	18	男	電子情報工学科	電子情報工学科棟
26205	谷泉太郎	19	男	電気電子工学科	電気電子・機械工学科棟
26132	海土花子	20	女	機械工学科	電気電子・機械工学科棟

表 20: 科目テーブル (第二正規化)

科目名	教員コード	教員名
国語 II	G002	長松次郎
物理 I	G001	長松遥
現代社会	G005	高田真由
電気工学	E012	三浦裕
信号処理	E012	三浦裕
機械力学	M301	上条雄三
計測工学	M553	西岡剛

表 21: 成績テーブル (第二正規化)

学籍番号	科目名	点数	判定	入力日
26401	国語 II	65	可	2018/6/10
26401	物理 I	42	不可	2018/6/11
26401	現代社会	79	良	2018/6/14
26205	物理 I	76	良	2018/6/11
26205	電気工学	66	可	2018/6/12
26205	信号処理	90	優	2018/6/12
26132	物理 I	87	優	2018/6/10
26132	機械力学	78	良	2018/6/11
26132	計測工学	100	優	2018/6/12

4.3 第三正規化の結果

第二正規化した結果を元に第三正規化を行う. 第三正規化とは, 第二正規化で分離できていない従属関係を分離し, 選択肢から選ぶ項目を抜き出す操作であった. 学生テーブル (表 19) では, 性別と学科は選択肢から選ぶ項目であるから抜き出してよいと考える. また校舎は学科によって定まるものであるため, 校舎は学科から参照する形に変更する. 科目テーブル (表 20) では, 教員名は教員コードで一意に定まるものであるから, 教員名を教員コードから参照する形に変更する. 成績テーブル (表 21) では, 判定は点数によって定まるから, 判定を成績テーブルから分離する. 点数から判定を決定する方法は上限, 下限を用いて SQL のクエリから計算を行うようにする. これより, 第三正規化によって 8 つのテーブルができることがわかる. 表 22〜29 に 8 つのテーブルを示す.

表 22: 学生テーブル (第三正規化)

学籍番号	学生名	年齢	性別	学科
26401	山川一郎	18	0	4
26205	谷泉太郎	19	0	2
26132	海土花子	20	1	1

表 23: 性別テーブル (第三正規化)

性別 ID	カテゴリ
1	男
2	女

表 24: 学科テーブル (第三正規化)

学科 ID	学科	校舎 ID
1	機械工学科	3
2	電気電子工学科	3
3	電子制御工学科	2
4	電子情報工学科	4
5	環境都市工学科	5

表 25: 校舎テーブル (第三正規化)

校舎 ID	校舎名
1	管理・一般校舎
2	電子制御工学科棟
3	電気電子・機械工学科棟
4	電子情報工学科棟
5	環境都市工学科棟
6	情報教育センター
7	図書館センター
8	体育館

表 26: 科目テーブル (第三正規化)

科目 ID	科目名	教員コード
10001	国語 II	G002
10002	物理 I	G001
10003	現代社会	G005
10004	電気工学	E012
10005	信号処理	E012
10006	機械力学	M301
10007	計測工学	M553

表 27: 教員テーブル (第三正規化)

教員コード	教員名
G002	長松次郎
G001	長松遥
G005	高田真由
E012	三浦裕
M301	上条雄三
M553	西岡剛

表 28: 成績テーブル (第三正規化)

学籍番号	科目 ID	点数	判定	入力日
26401	10001	65	3	2018/6/10
26401	10002	42	4	2018/6/11
26401	10003	79	2	2018/6/14
26205	10002	76	2	2018/6/11
26205	10004	66	3	2018/6/12
26205	10005	90	1	2018/6/12
26132	10002	87	1	2018/6/10
26132	10006	78	2	2018/6/11
26132	10007	100	1	2018/6/12

表 29: 判定テーブル (第三正規化)

判定 ID	判定名	上限	下限
1	優	100	80
2	良	79	70
3	可	60	69
4	不可	59	0

4.4 ER 図

第三正規化の結果を元に ER 図を作成する. 図 4 に ER 図を示す. 図 4 からテーブルが 8 つあり, それぞれのテーブルを学生テーブルが第三正規化によってできたテーブルに対応していることがわかる.

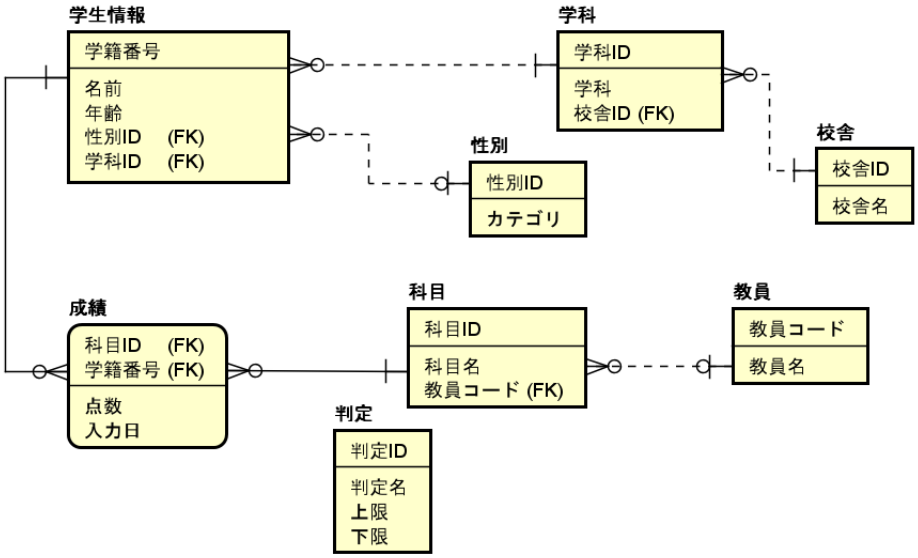

図 4: 課題 1 の ER 図

5 課題 2

本章では課題 2 におけるプログラムの説明および実行結果について述べる.

5.1 プログラムの説明

まず,MySQL 上にデータベースを定義し接続する. 課題 2 および課題 3 では「grade」というデータベースを用いる. リスト 1 にデータベース「grade」を定義し接続するプログラムを示す. リスト 1 において,1 行目のクエリがデータベース grade を定義するクエリである.2 行目はコメントアウトしてあるがデータベース grade ができているか確認することができる.3 行目のクエリはデータベース grade に接続するクエリである.

リスト 1: データベースの定義および接続

```
1 CREATE DATABASE grade;
2 -- SHOW DATABASES;
3 USE grade;
```

次にテーブルを定義する. テーブルを定義するクエリは,astah で ER 図 (図 4) を作成すると自動生成される. このため, テーブル定義を行うクエリ全体を説明することは省略する. ここでは学生テーブルの定義クエリを見て, どのような命令でテーブル定義を行っているか説明する. リスト 2 に学生テーブルの定義クエリを示す. リスト 2 では「CREATE TABLE テーブル名 (物理名)」という命令を用いてテーブルを定義していることが読み取れる. 内部では各カラムについて,「カラム名, 型, 条件」というようにカラム定義を行っていることが読み取れる. また 8,9 行目では外部キーの設定を行っていることが読み取れる. 他のテーブルについても「CREATE TABLE」命令を用いてテーブル定義を行っている.

リスト 2: 学生テーブルの定義

```
1 CREATE TABLE student (
2  code CHAR(5) NOT NULL PRIMARY KEY,
3  name VARCHAR(40) NOT NULL,
4  age INT,
5  gender_id INT,
```

```
 faculty_id INT NOT NULL,

 FOREIGN KEY (gender_id) REFERENCES gender (gender_id),
 FOREIGN KEY (faculty_id) REFERENCES faculty (faculty_id)
);
```

テーブル定義ができたから, テーブルにレコードを格納する. レコード格納を行うクエリ全体は非常に長いため, ここでも学生テーブルを例として説明する. リスト 3 に学生テーブルにレコード格納を行うクエリを示す. リスト 3 では「INSERT INTO テーブル名 (カラム) VALUES」という命令を用いてテーブルにレコードを格納している. 他のテーブルについても同様に INSERT 命令を用いてレコードを格納している.

リスト 3: 学生テーブルのレコード格納

```
INSERT INTO student(code,name,age,gender_id,faculty_id) VALUES
("26401","山川一郎",18,1,4),
("26205","谷泉太郎",19,1,2),
("26132","海土花子",20,2,1);
```

最後にテーブルの定義およびレコード格納が成功しているか確認する. リスト 4 に全テーブルからレコードを取得するクエリを示す. レコードを取得する命令は SELECT である. 全レコードを取得する場合は「SELECT * FROM テーブル名」というクエリを発行するとテーブル内の全レコードを取得できる.

リスト 4: 全テーブルからレコードを取得するクエリ

```
SELECT * FROM faculty;
SELECT * FROM gender;
SELECT * FROM place;
SELECT * FROM result;
SELECT * FROM score;
SELECT * FROM student;
SELECT * FROM subject;
SELECT * FROM teacher;
```

5.2 実行結果

実行結果としてリスト 4 の実行結果を示す. しかし, リスト 4 の実行結果は非常に長いので学生テーブルと成績テーブルについて確認する. 図 5 および図 6 に, リスト 4 の実行結果を示す. 図 5 は学生テーブルを取得したときの実行結果, 図 6 は成績テーブルを取得したときの実行結果である. 表 22 と図 5 が同じであることから学生テーブルは正常にテーブル定義およびレコード格納が行われていることがわかる. 同様に表 26 と図 6 が同じであるから成績テーブルも正常であることがわかる.

```
mysql> SELECT * FROM student;
+-------+--------------+------+-----------+------------+
| code  | name         | age  | gender_id | faculty_id |
+-------+--------------+------+-----------+------------+
| 26132 | 海土花子     |   20 |         2 |          1 |
| 26205 | 谷泉太郎     |   19 |         1 |          2 |
| 26401 | 山川一郎     |   18 |         1 |          4 |
+-------+--------------+------+-----------+------------+
3 rows in set (0.00 sec)
```

図 5: リスト 4 の実行結果 1

```
mysql> SELECT * FROM score;
+------------+-------+-------+------------+
| subject_id | code  | score | date       |
+------------+-------+-------+------------+
|      10001 | 26401 |    65 | 2018-06-10 |
|      10002 | 26132 |    87 | 2018-06-10 |
|      10002 | 26205 |    76 | 2018-06-11 |
|      10002 | 26401 |    42 | 2018-06-11 |
|      10003 | 26401 |    79 | 2018-06-14 |
|      10004 | 26205 |    66 | 2018-06-12 |
|      10005 | 26205 |    90 | 2018-06-12 |
|      10006 | 26132 |    78 | 2018-06-11 |
|      10007 | 26132 |   100 | 2018-06-12 |
+------------+-------+-------+------------+
9 rows in set (0.00 sec)
```

図 6: リスト 4 の実行結果 2

6　課題 3

本章では課題 3 で設けられている 5 つの課題についてクエリの説明および実行結果について述べる.

6.1　クエリの説明

本節では問題 1〜5 のクエリについて述べる.

6.1.1　問題 1 のクエリ

問題 1 は第一正規化したときの表を取得するクエリを発行することである. 問題を言い換えると,MySQL の命令を用いて表 18 の出力をするクエリを作成することである. リスト 5 に第一正規化した表を取得するクエリを示す. リスト 5 のクエリでは SELECT 文を用いて student.code を代表とする必要な項目を出力するように設定する. また, 内部結合 (INNER JOIN) を用いて必要なテーブルを学生テーブルに結合する. 内部結合文は「INNER JOIN 結合するテーブル ON 条件」になっている. 例えば, リスト 5 の 4 行目では成績テーブルを成績テーブルの学籍番号と学生テーブルの学籍番号が一致するように結合する. また 11 行目の ORDER 文は表示結果をソートする命令である.「ORDER BY カラム DESC または ASC」というように記述する. ASC は昇順,DESC は降順のことである. 今回は, 表 18 に合わせて, 結果を学生の年齢で昇順にソートしている.

リスト 5: 問題 1 のクエリ

```
1  SELECT student.code,student.name,student.age,gender.gender,faculty.faculty,place.place,
2  subject.name,score.score,result.result,teacher.teacher_id,teacher.name,score.date
3  FROM student
4  INNER JOIN score ON score.code = student.code
5  INNER JOIN gender ON student.gender_id = gender.gender_id
6  INNER JOIN faculty ON student.faculty_id = faculty.faculty_id
7  INNER JOIN place ON faculty.place_id = place.place_id
8  INNER JOIN subject ON subject.subject_id = score.subject_id
9  INNER JOIN teacher ON teacher.teacher_id = subject.teacher_id
10 INNER JOIN result ON score.score <= result.max AND score.score >= result.min
11 ORDER BY student.age ASC
12 ;
```

6.1.2 問題 2 のクエリ

問題 2 は, いずれかの科目の判定が「優」である学生の学籍番号, 氏名, 学科を問い合わせるクエリを発行し, 実行することである. リスト 6 に問題 2 のクエリを示す. リスト 6 では SELECT 文と内部結合を用いて必要なテーブルを結合したのちに, WHERE 文で条件にあったレコードのみを取得する.WHERE 文は「WHERE 条件」という文法で用いる. ここでは成績テーブルの成績が「優」のレコードのみを取得している. また「DISTINCT」とは取得結果において, 重複するレコードを重複がなくなるようにする命令のことである

リスト 6: 問題 2 のクエリ

```
SELECT DISTINCT student.code,student.name,faculty.faculty
FROM student INNER JOIN faculty ON faculty.faculty_id = student.faculty_id
INNER JOIN score ON score.code = student.code
INNER JOIN result ON score.score <= result.max AND score.score >= result.min
WHERE result.result = "優"
;
```

6.1.3 問題 3 のクエリ

問題 3 は, 担当者が「高田」, または「三浦」である科目を履修している学生について, 学生名, 科目名, 点数を問い合わせるクエリを発行し, 実行することである. リスト 7 に問題 3 のクエリを示す. リスト 7 では SELECT 文と内部結合を用いて必要なテーブルを結合したのちに, WHERE 文を用いて条件にあったレコードを取得する. しかし WHERE 文の条件が課題 2 に比べて複雑になっている. WHERE 文の条件がどのようになっているか説明する. まず条件は大きく分けて「OR」という演算子の前と後に分けられる. 前半部分では「teacher.name LIKE 三浦%」という条件になっている. これは LIKE 句を用いて担当教師の名前が条件にあう文字列のときに TRUE になる.LIKE 句は文字列の検索を行う命令で,「LIKE 検索文字列」という文法である. 検索文字列はワイルドカードを使用して, 部分検索を行うことができる. 今考えている条件は「三浦%」である. これは「三浦」の後に 0 文字以上の任意の文字列を含む文字列という意味である. これによって, 担当者が「三浦○○」である科目のレコードを取得している. 命令の後半部分も同様で, 担当者が「高田○○」である科目のレコードを取得する. この 2 つの命令が OR(論理和) になっているから, 問題の「担当者が「高田」, または「三浦」である科目」を取得することができる.

リスト 7: 問題 3 のクエリ

```
SELECT student.name,subject.name,teacher.name,result.result
FROM student INNER JOIN score ON score.code = student.code
INNER JOIN subject ON subject.subject_id = score.subject_id
INNER JOIN teacher ON teacher.teacher_id = subject.teacher_id
INNER JOIN result ON score.score <= result.max AND score.score >= result.min
WHERE teacher.name LIKE "三浦%" OR teacher.name LIKE "高田%"
ORDER BY student.age ASC
;
```

6.1.4 問題 4 のクエリ

問題 4 は, 科目名に「学」が含まれる科目について, 履修している学生名, 科目名, 点数を問い合わせるクエリを発行し, 実行することであった. リスト 8 に問題 4 のクエリを示す. リスト 8 では SELECT 文と内部結合を用いて必要なテーブルを結合したのちに, WHERE 文を用いて条件にあったレコードを取得する.WHERE 文の条件は, 科目テーブルの科目名が LIKE 句以下の条件に当てはまるときである. LIKE 句の条件は「%学%」である.%は課題 3 のクエリで説明したように 0 文字以上の任意の文字列である. 「%学%」にすることで「統計学」のように一番後ろの文字が「学」である場合も「力学 I」のように途中に「学」が入っている場合も検出できる.

リスト 8: 問題 4 のクエリ

```
SELECT student.name,subject.name,score.score
FROM student INNER JOIN score ON score.code = student.code
INNER JOIN subject ON subject.subject_id = score.subject_id
WHERE subject.name LIKE "%学%"
ORDER BY student.age ASC
;
```

6.1.5 問題 5 のクエリ

問題 5 は担当が「三浦」の科目の平均点を問い合わせるクエリを発行し, 実行することである. リスト 9 に問題 5 のクエリを示す. リスト 9 では SELECT 文で「AVG(score.score)」を画面出力する.AVG とは average(平均) のことである. ここでの平均は算術平均をことである. リスト 9 では科目の平均点を問い合わせるので成績テーブルの成績の平均を出力している. また,WHERE 文の条件としては問題 3 のクエリ (リスト 7) を流用している.

リスト 9: 問題 5 のクエリ

```
SELECT AVG(score.score)
FROM student INNER JOIN score ON score.code = student.code
INNER JOIN subject ON subject.subject_id = score.subject_id
INNER JOIN teacher ON teacher.teacher_id = subject.teacher_id
WHERE teacher.name LIKE "三浦%"
;
```

6.2 実行結果

本節では問題 1〜問題 5 のクエリの実行結果について述べる.

6.2.1 問題 1 の実行結果

課題 1 のクエリ (リスト 5) の実行結果の図は横に長いため, 掲載を省略する. 実行結果は表 18 の第一正規化の結果と一致していることが確認できた. これよりリスト 5 のクエリは課題 1 の題意は満たせたと言える.

6.2.2 問題 2 の実行結果

課題 2 のクエリ (リスト 6) の実行結果を図 7 に示す. 表 18 から, 判定に「優」がある学生は「谷川太郎」と「海土花子」であることがわかる. すなわち, 問題 2 のクエリの正しい実行結果として, この 2 人の学籍番号, 氏名, 学科が取得できればよい. 図 7 を見ると, この 2 人の学籍番号, 氏名, 学科が取得できていることがわかる. これよりリスト 6 のクエリは課題 2 の題意は満たせたと言える.

```
mysql>  SELECT DISTINCT student.code,student.name,faculty.faculty
    ->  FROM student INNER JOIN faculty ON faculty.faculty_id = student.faculty_id
    ->  INNER JOIN score ON score.code = student.code
    ->  INNER JOIN result ON score.score <= result.max AND score.score >= result.min
    ->  WHERE result.result = "優"
    ->  ;
+-------+--------------+------------------------+
| code  | name         | faculty                |
+-------+--------------+------------------------+
| 26132 | 海土花子     | 機械工学科             |
| 26205 | 谷泉太郎     | 電気電子工学科         |
+-------+--------------+------------------------+
2 rows in set (0.08 sec)
```

図 7: リスト 6 の実行結果

6.2.3 問題 3 の実行結果

課題 3 のクエリ (リスト 7) の実行結果を図 8 に示す. 表 18 から, 担当者が「高田」, または「三浦」である科目を履修している学生とその科目は「山川一郎」が「現代社会」,「谷川太郎」が「信号処理」,「電気工学」であることがわかる. すなわち, 課題 3 のクエリの正しい実行結果として, この 3 つの科目において, 履修学生の氏名, 科目名, 点数が取得できればよい. またここでは確認のために担当者の名前も表示する. 図 7 を見ると, この 3 つの科目の履修学生の氏名, 科目名, 点数が取得できていることがわかる. これよりリスト 7 のクエリは課題 3 の題意は満たせたと言える.

```
mysql>  SELECT student.name,subject.name,teacher.name,result.result
    ->  FROM student INNER JOIN score ON score.code = student.code
    ->  INNER JOIN subject ON subject.subject_id = score.subject_id
    ->  INNER JOIN teacher ON teacher.teacher_id = subject.teacher_id
    ->  INNER JOIN result ON score.score <= result.max AND score.score >= result.min
    ->  WHERE teacher.name LIKE "三浦%" OR teacher.name LIKE "高田%"
    ->  ORDER BY student.age ASC
    ->  ;
+--------------+--------------+--------------+--------+
| name         | name         | name         | result |
+--------------+--------------+--------------+--------+
| 山川一郎     | 現代社会     | 高田真由     | 良     |
| 谷泉太郎     | 信号処理     | 三浦裕       | 優     |
| 谷泉太郎     | 電気工学     | 三浦裕       | 可     |
+--------------+--------------+--------------+--------+
3 rows in set (0.02 sec)
```

図 8: リスト 7 の実行結果

6.2.4 問題 4 の実行結果

課題 4 のクエリ (リスト 8) の実行結果を図 9 に示す. 表 18 から,「学」が含まれる科目は「電気工学」,「機械工学」,「計測工学」の 3 つであることがわかる. すなわち, 課題 4 のクエリの正しい実行結果として, この 3 つの科目を履修している学生名, 科目名, 点数が取得できればよい. 図 9 を見ると, この 3 つの科目を履修している科目の学生名, 科目名, 点数が取得できていることがわかる. これよりリスト 8 のクエリは課題 4 の題意は満たせたと言える.

```
mysql>  SELECT student.name,subject.name,score.score
    ->  FROM student INNER JOIN score ON score.code = student.code
    ->  INNER JOIN subject ON subject.subject_id = score.subject_id
    ->  WHERE subject.name LIKE "%学%"
    ->  ORDER BY student.age ASC
    ->  ;
+--------------+--------------+-------+
| name         | name         | score |
+--------------+--------------+-------+
| 谷泉太郎     | 電気工学     |    66 |
| 海土花子     | 機械力学     |    78 |
| 海土花子     | 計測工学     |   100 |
+--------------+--------------+-------+
3 rows in set (0.00 sec)
```

図 9: リスト 8 の実行結果

6.2.5 問題 5 の実行結果

課題 5 のクエリ (リスト 9) の実行結果を図 10 に示す. 表 18 から, 担当者が「三浦」である科目は 2 つあり, それぞれ 66 点と 99 点であることがわかる. すなわち平均点は 78 点である. これが実行結果として取得できればよい. 図 10 を見ると, 平均点として 78.0 点が取得できていることがわかる. これよりリスト 9 のクエリは課題 5 の題意は満たせたと言える.

```
mysql>  SELECT AVG(score.score)
    ->  FROM student INNER JOIN score ON score.code = student.code
    ->  INNER JOIN subject ON subject.subject_id = score.subject_id
    ->  INNER JOIN teacher ON teacher.teacher_id = subject.teacher_id
    ->  WHERE teacher.name LIKE "三浦%"
    ->  ;
+------------------+
| AVG(score.score) |
+------------------+
|          78.0000 |
+------------------+
1 row in set (0.01 sec)
```

図 10: リスト 9 の実行結果

7　課題 4

本章では課題 4 として次に示す項目について述べる.

1. データの概要および目的
2. データの正規化と ER 図
3. MySQL による実装
4. 実装結果
5. テスト項目
6. テスト項目のプログラムの説明
7. テスト項目の実行結果

7.1　データの概要および目的

課題 4 として長野市のオープンデータである「避難場所の一覧」を用いる [2]. このオープンデータを用いる目的として, 近年の自然災害の多さから, 避難場所を検索するデータベースを作成し, 付近の避難場所を検索を行うことを目的とする. オープンデータ「避難場所の一覧」は 2020 年 8 月 4 日現在,2020 年 4 月 7 日に最終更新されたデータである.
レコード数は 202 でカラムの一覧は次に示す通りである. ここで適否とは避難に適しているか否かの情報であるから〇もしくは×のどちらかである . また指定避難所, 広域避難場所は指定されている場合〇, それ以外の場合はハイフンまたは空欄になっている. 洪水, 土砂災害, 地震は留意事項の項目があるが, 大規模な火事には留意事項がないことも注意が必要である.

1. ID
2. 指定緊急避難場所の名称 (以下, 避難場所名)
3. 名称かな
4. 住所
5. 所在地区
6. 緯度
7. 経度

8. 洪水等の適否

9. 洪水等の適否 (留意事項)

10. 土砂災害の適否

11. 土砂災害の適否 (留意事項)

12. 地震の適否

13. 地震の適否 (留意事項)

14. 大規模な火事の適否

15. 指定避難所

16. 広域避難場所

7.2 データの正規化と ER 図

データの正規化を行う. 元のデータはカラム数が多いため掲載は省略する. また, 元のデータはレコードをまたいで定義される値がない. このため第一正規化はすでに完了しているのと同義である.

第二正規化を行う. ここでは 3 つのテーブルに分離する.1 つ目は ID を主として, 避難場所名, 名称かな, 住所, 所在地区, 緯度, 経度の 6 カラムを従属するカラムとする (避難場所テーブル).2 つ目は ID を主として, 洪水, 土砂災害, 地震, 大規模な火事の適否とその留意事項の合計 7 カラムを従属するカラムとする (避難適否テーブル).3 つ目は ID を主として指定避難所, 広域避難場所を従属するカラムとする (避難場所属性テーブル). これらを表で表すと表 30〜表 32 のようになる. 表 30〜表 32 では各カラムについて最初の 3 行を表示している. ただし, 避難場所テーブル (表 30) の「名称かな」は長いため省略する.

表 30: 避難場所テーブル

ID	避難場所名	名称かな	住所	所在地区	緯度	経度
1	長野市立加茂小学校	(省略)	長野県長野市大字西長野 185 − 6	01 第一	36.65751141	138.1764597
2	信州大学教育学部グラウンド	(省略)	長野県長野市西長野 6 の口	01 第一	36.65736035	138.1798011
3	ひまわり公園	(省略)	長野県長野市大字長野旭町 1108-10	01 第一	36.65470572	138.1824138
⋮	⋮	⋮	⋮	⋮	⋮	⋮

表 31: 避難適否テーブル

ID	洪水等の適否	洪水等の適否（留意事項）	土砂災害の適否	土砂災害の適否（留意事項）	地震の適否	大規模な火事の適否
1	○	グラウンド除く	○		○	×
2	○		○	北西の一部除く	○	×
3	○		○		○	×
⋮	⋮	⋮	⋮	⋮	⋮	⋮

表 32: 避難場所属性テーブル

ID	指定避難所	広域避難場所
1	○	
2	-	
3	-	
⋮	⋮	⋮

次に第三正規化を行う. 避難場所テーブル (表 30) において所在地区は一定の選択肢から選ぶ項目になっているため, 分離する. また, 避難適否テーブル (表 31) において留意事項も一定の選択肢から選ぶか, 空欄となっているから分離する. 避難場所属性テーブル (表 32) は○,-, 空白の扱いが入り乱れているため, ○もしくは×に統一する . 表 33〜表 37 に第三正規化したテーブルを示す. 土砂災害, 地震, 火災の適否テーブルおよび留意事項テーブルは洪水テーブルと同様のものであるから省略する.

表 33: 避難場所テーブル

ID	避難場所名	名称かな	住所	所在地区	緯度	経度
1	長野市立加茂小学校	(省略)	長野県長野市大字西長野 185 − 6	1	36.65751141	138.1764597
2	信州大学教育学部グラウンド	(省略)	長野県長野市西長野 6 の口	1	36.65736035	138.1798011
3	ひまわり公園	(省略)	長野県長野市大字長野旭町 1108-10	1	36.65470572	138.1824138
⋮	⋮	⋮	⋮	⋮	⋮	⋮

表 34: 所在地区テーブル

所在地区 ID	所在地区名
1	01 第一
2	02 第二
3	03 第三
⋮	⋮
4	44 大岡
5	45 信州新町
6	46 中条

表 35: 避難場所の属性テーブル

ID	指定避難所	広域避難場所
1	○	×
2	×	×
3	×	×
⋮	⋮	⋮

表 36: 洪水テーブル

ID	避難の適否	留意事項
1	○	1
2	○	0
3	○	0
⋮	⋮	⋮

表 37: 洪水留意事項テーブル

留意事項 ID	留意事項
0	なし
1	グラウンド除く
2	体育館除く
3	2 階以上
4	2 階部分の観客席及び 内部・外部コンコース

第三正規化ができたから,ER 図の作成を行う. 図 11 に ER 図を示す.

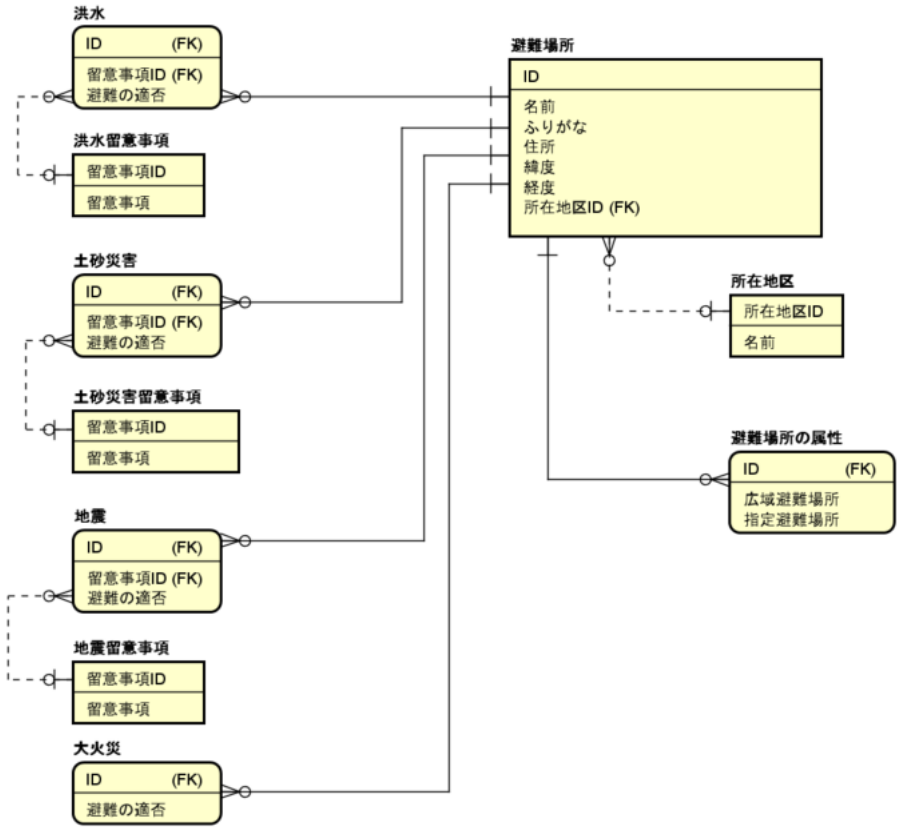

図 11: 課題 4 の ER 図

7.3 MySQL による実装

MySQL で避難場所データベースを実装する. 表 38 にカラムの設定を示す.

表 38: カラムの設定

項目	カラム (論理名)	カラム (物理名)	型	必須/非必須
ID	ID	code	INT	必須
避難所名	名前	name	VARCHAR(100)	必須
名称かな	ふりがな	kana	VARCHAR(100)	必須
住所	住所	address	VARCHAR(50)	必須
緯度	緯度	latitude	FLOAT(10)	必須
経度	経度	longitude	FLOAT(10)	必須
所在地区 ID	所在地区 ID	district_id	INT	必須
所在地区	所在地区名	name	VARCHAR(10)	必須
広域避難場所	広域避難場所	wide_area	CHAR(2)	非必須
指定避難所	指定避難所	designation	CHAR(2)	非必須
避難の適否	避難の適否	suitability	CHAR(2)	必須
留意事項	留意事項	note	VARCHAR(50)	非必須

避難場所データベースをデータベース名を「evacuation」として定義する. また, 表 38 に示す条件で, 避難場所データベースのテーブル定義クエリを astah で生成し定義する. さらに, テーブルにレコードを追加するクエリを発行して, 実行する. データベース定義クエリおよびレコード追加クエリは非常に長いため省略する.

7.4 実装結果

テーブルが正しく生成できているか確認する. ここでは,SELECT 文を用いて避難場所テーブルおよび洪水テーブルが正しく生成できているか確認する. リスト 10 避難場所テーブルおよび洪水テーブルの全レコードを取得するクエリを示す.

リスト 10: 実装結果を確認するためのクエリ

```
1 SELECT * FROM place;
2 SELECT * FROM flood;
```

避難場所テーブルの場合の実行結果は非常に長いため省略するが, 全レコードを問題なく取得できた. 図 12 に洪水テーブルにおける実行結果の抜粋を示す. 図 12 は表 36 と同じであるから正しく実装できていると言える.

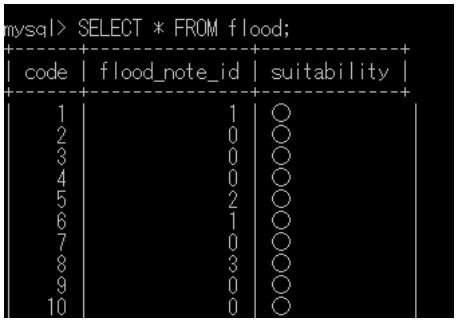

図 12: 洪水レコードの実装の確認

7.5 テスト項目

課題 3 と同様に次の 5 つのクエリを発行し, 実行する.

1. 第一正規化したときの表を取得するクエリ.
2. 広域避難場所に指定されている避難場所の番号, 名前, 所在地区を問い合わせるクエリ.
3. 所在地区が「朝陽」,「豊野」の避難場所の名前, 所在地区, 各災害の避難の適否を問い合わせるクエリ.
4. 避難場所の名前に「学校」が含まれている避難場所について避難場所の名前, 所在地区, 避難所の属性を問い合わせるクエリ.
5. 現在地 (緯度, 経度) から最も近い避難場所の名前, 住所, 緯度, 経度を問い合わせるクエリ.

7.6 クエリの説明

本節では問題 1〜5 のクエリについて述べる.

7.6.1 問題 1 のクエリ

問題 1 は第一正規化したときの表を取得するクエリを発行し, 実行することであった. リスト 11 に問題 1 のクエリを示す. リスト 11 ではすべてのテーブルを内部結合し,SELECT 文を用いて表示している.

リスト 11: 問題 1 のクエリ

```
SELECT place.code,place.name,place.kana,place.address,
place.latitude,place.longitude,district.name,
attribute.designation,attribute.wide_area,
flood.suitability,flood_note.note,
sediment_disaster.suitability,sediment_disaster_note.note,
earthquake.suitability,earthquake_note.note,
fire.suitability
FROM place
INNER JOIN district ON district.district_id = place.district_id
INNER JOIN attribute ON attribute.code = place.code
INNER JOIN flood ON flood.code = place.code
INNER JOIN sediment_disaster ON sediment_disaster.code = place.code
INNER JOIN earthquake ON earthquake.code = place.code
INNER JOIN fire ON fire.code = place.code
INNER JOIN flood_note ON flood_note.flood_note_id = flood.flood_note_id
INNER JOIN sediment_disaster_note ON sediment_disaster_note.sediment_disaster_notes_id
= sediment_disaster.sediment_disaster_notes_id
INNER JOIN earthquake_note ON earthquake_note.earthquake_note_id
= earthquake.earthquake_note_id
;
```

7.6.2 問題 2 のクエリ

問題 2 は広域避難場所に指定されている避難場所の番号, 名前, 所在地区を問い合わせるクエリを発行し, 実行することであった. リスト 12 に問題 2 のクエリを示す. リスト 12 では SELECT 文を用いて必要なテーブルを結合したのち, WHERE 文を用いて広域避難場所に指定されているレコードのみを取得している.

リスト 12: 問題 2 のクエリ

```
SELECT place.code,place.name,district.name
FROM place
INNER JOIN district ON district.district_id = place.district_id
INNER JOIN attribute ON attribute.code = place.code
WHERE attribute.wide_area = "○"
;
```

7.6.3 問題 3 のクエリ

問題 3 は所在地区が「朝陽」,「豊野」の避難場所の名前, 所在地区, 各災害の避難の適否を問い合わせるクエリを発行し, 実行することであった. リスト 13 に問題 3 のクエリを示す. リスト 13 えは SELECT 文を用いて必要なテーブルを結合したのち, WHERE 文を用いて所在地区が「朝陽」,「豊野」であるレコードを取得している.

リスト 13: 問題 3 のクエリ

```
SELECT place.name,district.name,flood.suitability,
sediment_disaster.suitability,earthquake.suitability,fire.suitability
FROM place
INNER JOIN district ON district.district_id = place.district_id
INNER JOIN flood ON flood.code = place.code
INNER JOIN sediment_disaster ON sediment_disaster.code = place.code
INNER JOIN earthquake ON earthquake.code = place.code
INNER JOIN fire ON fire.code = place.code
WHERE district.name = "朝陽" OR district.name = "豊野"
;
```

7.6.4 問題 4 のクエリ

問題 4 は避難場所の名前に「学校」が含まれている避難場所について避難場所の名前, 所在地区, 避難所の属性を問い合わせるクエリを発行し, 実行することであった. リスト 14 に問題 4 のクエリを示す. リスト 14 では SELECT 文を用いて必要なテーブルを結合したのち,WHERE 文と LIKE 句を用いて「学校」を含むレコードを取得している.

リスト 14: 問題 4 のクエリ

```
SELECT place.name,district.name,attribute.designation,attribute.wide_area
FROM place
INNER JOIN district ON district.district_id = place.district_id
INNER JOIN attribute ON attribute.code = place.code
WHERE place.name LIKE "%学校%"
;
```

7.6.5 問題 5 のクエリ

問題 5 は現在地 (緯度経度) から最も近い避難場所の名前, 住所, 緯度, 経度を問い合わせるクエリを発行し, 実行することであった. リスト 15 に問題 5 のクエリを示す. リスト 15 では SELECT 文を用いて, 必要なテーブルを結合したのちに, 緯度, 経度の情報から, 避難場所までの距離を計算し, 近くにある避難場所を取得する. 点 P(緯度, 経度) と表すことにすると,2 点 A(x_1, y_2),B(x_1, y_2) の距離 d は式 (1) で表される [3]. ただし, 赤道半径 $r = 6378km$ とする. 距離の計算結果を LIMIT 文を用いて距離が近い順に, 上位 5 件取得する.

リスト 15: 問題 5 のクエリ

```
SELECT place.name,place.address,place.latitude,place.longitude,(
    6371 * ACOS(
      COS(RADIANS(現在地(緯度))) * COS(RADIANS(place.LATITUDE)) *
      COS(RADIANS(place.LONGITUDE) - RADIANS(現在地(経度)))
       + SIN(RADIANS(現在地(緯度))) * SIN(RADIANS(place.LATITUDE))
    )
  ) AS DISTANCE
FROM place
ORDER BY DISTANCE
LIMIT 5
;
```

$$d = r \cos^{-1}(\cos y_1 \cos y_2 \cos(x_2 - x_1) + \sin y_1 \sin y_2) \tag{1}$$

7.7 実行結果

本節では問題 1〜5 の実行結果について述べる.

7.7.1 問題 1 の実行結果

問題 1 の実行結果は非常に長いため省略するが, 第一正規化の表を取得することができた. これより問題 1 の題意は満たせたと言える.

7.7.2 問題 2 の実行結果

図 13 に問題 2 のクエリ (リスト 12) の実行結果を示す. 長野市の広域避難場所は長野市のホームページの「避難場所・避難所・福祉避難所・応急救護所・防災備蓄倉庫について」[4] より図 13 の実行結果で取得した 5 件であることがわかる. これより, 問題 2 の題意を満たせたと言える.

```
mysql> SELECT place.code,place.name,district.name
    -> FROM place
    -> INNER JOIN district ON district.district_id = place.district_id
    -> INNER JOIN attribute ON attribute.code = place.code
    -> WHERE attribute.wide_area = "○"
    -> ;
+------+--------------------------------+-----------+
| code | name                           | name      |
+------+--------------------------------+-----------+
|    9 | 城山公園                       | 第二      |
|   42 | 長野運動公園                   | 吉田      |
|   99 | 南長野運動公園                 | 篠ノ井    |
|  104 | 長野市立松代中学校             | 松代      |
|  133 | 川中島古戦場史跡公園           | 更北      |
+------+--------------------------------+-----------+
5 rows in set (0.01 sec)
```

図 13: 問題 2 の実行結果

7.7.3 問題 3 の実行結果

図 14 に問題 3 のクエリ (リスト 13) の実行結果を示す. 第一正規化したデータと図 14 の実行結果を比較すると, 所在地区が「朝陽」,「豊野」のレコードを正しく取得できていると言える. これより問題 3 の題意は満たせたと言える.

```
mysql> SELECT place.name,district.name,flood.suitability,sediment_disaster.suitability,earthquake.suitability,fire.suitability
    -> FROM place
    -> INNER JOIN district ON district.district_id = place.district_id
    -> INNER JOIN flood ON flood.code = place.code
    -> INNER JOIN sediment_disaster ON sediment_disaster.code = place.code
    -> INNER JOIN earthquake ON earthquake.code = place.code
    -> INNER JOIN fire ON fire.code = place.code
    -> WHERE district.name = "朝陽" OR district.name = "豊野"
    -> ;
+--------------------------------------------------------------+--------+-------------+-------------+-------------+-------------+
| name                                                         | name   | suitability | suitability | suitability | suitability |
+--------------------------------------------------------------+--------+-------------+-------------+-------------+-------------+
| 長野市立朝陽小学校                                           | 朝陽   | ×           | ×           | ○           | ×           |
| 長野市オリンピック記念アリーナ（エムウェーブ）               | 朝陽   | ○           | ○           | ○           | ×           |
| 長野県長野盲学校                                             | 朝陽   | ○           | ○           | ○           | ×           |
| 信州大学教育学部附属長野中学校                               | 朝陽   | ×           | ×           | ○           | ×           |
| 信州大学教育学部附属長野小学校                               | 朝陽   | ×           | ×           | ○           | ×           |
| 長野市立豊野西小学校                                         | 豊野   | ×           | ×           | ○           | ×           |
| 長野市立豊野公民館                                           | 豊野   | ×           | ×           | ○           | ×           |
| 長野市立豊野中学校                                           | 豊野   | ×           | ×           | ○           | ×           |
| 豊野体育館                                                   | 豊野   | ×           | ×           | ○           | ×           |
| 豊野東部地区集会所                                           | 豊野   | ×           | ×           | ○           | ×           |
| 長野市立豊野東小学校                                         | 豊野   | ○           | ○           | ○           | ×           |
+--------------------------------------------------------------+--------+-------------+-------------+-------------+-------------+
11 rows in set (0.01 sec)
```

図 14: 問題 3 の実行結果

7.7.4 問題 4 の実行結果

図 15 に問題 4 のクエリ (リスト 14) の実行結果の冒頭を示す. 図 15 から, 避難場所名に「学校」を含むレコードを取得できていることがわかる. これより, 課題 4 の題意は満たせたと言える.

```
mysql> SELECT place.name,district.name,attribute.designation,attribute.wide_area
    -> FROM place
    -> INNER JOIN district ON district.district_id = place.district_id
    -> INNER JOIN attribute ON attribute.code = place.code
    -> WHERE place.name LIKE "%学校%"
    -> ;
+--------------------------------------------------------+-------------+-------------+-----------+
| name                                                   | name        | designation | wide_area |
+--------------------------------------------------------+-------------+-------------+-----------+
| 長野市立加茂小学校                                     | 第一        | ○           | ×         |
| 長野市立西部中学校                                     | 第一        | ○           | ×         |
| 長野市立城山小学校                                     | 第二        | ○           | ×         |
| 長野市立柳町中学校                                     | 第二        | ○           | ×         |
| 長野県長野高等学校グラウンド                           | 第二        | ×           | ×         |
| 長野市立湯谷小学校                                     | 第二        | ○           | ×         |
| 長野県長野西高等学校                                   | 第二        | ×           | ×         |
| 長野市立城東小学校                                     | 第二        | ○           | ×         |
| 長野市立鍋屋田小学校                                   | 第三        | ○           | ×         |
| 長野県長野商業高等学校                                 | 第四        | ×           | ×         |
| 長野市立山王小学校                                     | 第五        | ○           | ×         |
| 長野市立裾花小学校                                     | 第五        | ○           | ×         |
| 長野市立犀陵中学校                                     | 芹田        | ○           | ×         |
| 文化学園長野中学・高等学校グラウンド                   | 芹田        | ×           | ×         |
```

図 15: 問題 4 の実行結果

7.7.5 問題 5 の実行結果

図 16 および図 17 に問題 4 のクエリ (リスト 15) の実行結果を示す. 図 16 は長野高専の緯度, 経度の場合の実行結果, 図 17 は長野駅の緯度, 経度の場合の実行結果である. 長野高専の場合, 長野高専のグラウンド, 市立長野高校, 徳間小学校が取得されている. これは正しい結果であると言える. また, 長野駅の場合, 長野駅東口の公園, 長野市立鍋屋田小学校が取得されている. これも正しい結果であると言える. これらより, 課題 5 の題意は満たせたと言える.

```
mysql> SELECT place.name,place.address,place.latitude,place.longitude,(
    ->     6371 * ACOS(
    ->       COS(RADIANS(36.678011)) * COS(RADIANS(place.LATITUDE)) * COS(RADIANS(place.LONGITUDE) - RADIANS(138.234185))
    ->       + SIN(RADIANS(36.678011)) * SIN(RADIANS(place.LATITUDE))
    ->     )
    ->   ) AS DISTANCE
    -> FROM place
    -> ORDER BY DISTANCE
    -> LIMIT 5
    -> ;
+-----------------------------------------+-----------------------------------+----------+-----------+--------------------+
| name                                    | address                           | latitude | longitude | DISTANCE           |
+-----------------------------------------+-----------------------------------+----------+-----------+--------------------+
| 長野工業高等専門学校グラウンド          | 長野県長野市大字徳間716           |  36.6769 |   138.234 | 0.1257877293380131 |
| 長野市立長野中学校・高等学校            | 長野県長野市大字徳間1133          |  36.6804 |   138.235 | 0.2717919489516784 |
| 長野市立徳間小学校                      | 長野県長野市大字徳間570           |  36.6764 |   138.228 | 0.5956571514199579 |
| 昭和の森公園                            | 長野県長野市上野2-120-13          |  36.6853 |   138.235 | 0.8187180252196682 |
| 昭和の森公園フィットネスセンター        | 長野県長野市上野2-120-13          |  36.6854 |   138.235 | 0.8210940370837945 |
+-----------------------------------------+-----------------------------------+----------+-----------+--------------------+
5 rows in set (0.02 sec)

mysql>
```

図 16: 問題 5 の実行結果 (座標:長野高専)

```
mysql> SELECT place.name,place.address,place.latitude,place.longitude,(
    ->     6371 * ACOS(
    ->       COS(RADIANS(36.643133)) * COS(RADIANS(place.LATITUDE)) * COS(RADIANS(place.LONGITUDE) - RADIANS(138.188674))
    ->       + SIN(RADIANS(36.643133)) * SIN(RADIANS(place.LATITUDE))
    ->     )
    ->   ) AS DISTANCE
    -> FROM place
    -> ORDER BY DISTANCE
    -> LIMIT 5
    -> ;
+-----------------------------+-----------------------------------------------+----------+-----------+---------------------+
| name                        | address                                       | latitude | longitude | DISTANCE            |
+-----------------------------+-----------------------------------------------+----------+-----------+---------------------+
| 長野駅東口公園              | 長野県長野市大字栗田973番地1                  |  36.6413 |   138.192 | 0.37530458095815825 |
| 長野市立鍋屋田小学校        | 長野県長野市大字鶴賀上千歳町1365-2            |    36.65 |    138.19 |  0.7719269177376873 |
| 長野市立山王小学校          | 長野県長野市大字中御所30-1                    |   36.647 |   138.181 |  0.7967363426600262 |
| 多目的（防災）広場          | 長野県長野市大字鶴賀緑町1613                  |  36.6486 |   138.195 |  0.8214425701857968 |
| 長野市芸術館                | 長野県長野市大字鶴賀緑町1613                  |  36.6486 |   138.195 |  0.8214425701857968 |
+-----------------------------+-----------------------------------------------+----------+-----------+---------------------+
5 rows in set (0.01 sec)
```

図 17: 問題 5 の実行結果 (座標:長野駅)

8　考察

課題 1〜 3 について, 簡単なデータで正規化や ER 図,MySQL での実装を学習することができた. これより, 正規化や SQL の使い方について学習するという目的を達成できたと言える. 課題 4 については, また, 課題 4 について, 長野市のオープンデータを用いてデータベースを作成できた. しかし第二正規化, 第三正規化が非常に簡素であり, 正規化の有効性や拡張性を理解ためのデータとしてはあまりふさわしくないと考える. しかし問題 5 で作成したクエリは任意の座標 (緯度, 経度) から付近の避難所を検索することができるため, 実用性があると考える. このことから長野市のオープンデータを用いて, 実際に役に立つデータベースやクエリを作成することができたと考える.

参考文献

[1] 国立高専機構長野高専,http://www.nagano-nct.ac.jp/ , 閲覧日 2020 年 8 月 5 日

[2] 長野市オープンデータサイト,https://www.city.nagano.nagano.jp/site/opendata/ , 閲覧日 2020 年 8 月 5 日

[3] 三浦英俊,”緯度経度を用いた 3 つの距離計算方法”,http://www.orsj.or.jp/archive2/or60-12/or60_12_701.pdf, 閲覧日 2020 年 8 月 5 日

[4] 避難場所・避難所・福祉避難所・応急救護所・防災備蓄倉庫について,https://www.city.nagano.nagano.jp/soshiki/kikibousai/2530.html , 閲覧日 2020 年 8 月 5 日