SQL に関する試験の自動採点システムの試作

Experimental Implementation of Automatic Grading System for Examination of SQL

田辺　良則

TANABE Yoshinori

概　要

データベース言語 SQL を対象とした試験問題の自動採点システムを試作した．システムは 2 つのモジュールからなる．最初のモジュールは，問題に対して作成された SQL 文が正しいかどうかを判定するものである．ソフトウェアテスティングで一般的に行われている技法を用いて，入力と正解を比較することで判定を行う．もう ひとつのモジュールは，受験者が答案の提出を行うものである．これを用いることで，期末試験のような，インターネットへの接続を制限したい状況でも自動採点が可能になる．本システムを，実際の授業の小テストで用いる試行を行い，いくつかの問題点を抽出した．その結果も踏まえて，今後のシステムの拡張方向について検討を行った．

1.　はじめに

著者は，鶴見大学文学部ドキュメンテーション学科で，2 年生の必修科目「情報基礎演習 III（データベース)」を担当している．この科目では，教科書として文献 [1] を用いてデータベース問い合わせ言語 SQL の初歩を学習する．SQL 言語は一種のコンピュータ言語，ないし，ドメイン固有言語 (Domain Specific Language) であると考えられる．プログラミング言語などの他のコンピュータ言語と同様に，SQL 言語の習得のためには，基礎となる理論を学ぶだけでなく，多くの練習問題を解く必要がある．

しかし，初学者にとって練習問題を解いた時に，自分の解答が正しいかどうか判断するのは容易ではない．あまり習得に熱意の無い学習者は，とりあえず解答を作れたということに満足してしまい，その正しさを十分に検討することなく提出してしまう傾向がある．

これらの問題に対処するために，自動採点システムを構築することが有効であると考えられる．

1.1 出題形式の比較

SQL の理解を評価する状況のひとつとして，WHERE 句での比較の方法を理解しているかどうかを評価したい，という例を考えよう．この評価のための出題の方法はいくつか考えられる．典型的な 2 つの方法として，選択式と記述式を比較することにする．各々の例をあげる．

選択式　テーブル table1 から，カラム price の値が 1000 以上 2000 未満のレコードを抽出する SQL 文として，もっとも適切なものを選べ．（選択肢は図 1 を参照）

記述式　テーブル table1 から，カラム price の値が 1000 以上 2000 未満のレコードを抽出する SQL 文を書け．

選択式の明らかな問題点として，「まぐれあたり」の可能性がある．解答者が SQL について全く知らなかったとしても，1/5 の確率で正解をひきあててしまう．さらに，「抽出は SELECT，更新は UPDATE」という程度の漠然とした理解があるだけで，この確率は 1/3 に上昇する．正解を答えた解答者をどの程度評価して良いのか難しい．

また，選択式では，問題に答えさせる際に実システムへのアクセスを禁止せざるを得ない．実システムが使えれば，5 つの SQL 文を実際に実行することで，どれが正解か簡単に判別できてしまうからである．しかし，評価したいのは，実システムにアクセスする環境にあって，必要な SQL 文を作成する能力である．たとえば (やや強引な例であるが) ある解答者が，SQL において 3 つの数値 *a, b, c* がこの順で小さいこ

```
1.  SELECT price FROM table1 WHERE price < 1000 OR 2000 > price
2.  UPDATE table1 SET price WHERE price < 1000 OR 2000 > price
3.  SELECT * FROM table1 WHERE price >= 1000 AND price < 2000
4.  SELECT * FROM table1 WHERE 1000 <= price < 2000
5.  UPDATE table1 SET price WHERE 1000 <= price < 2000
```

図 1 選択式問題の選択肢

とを記述するために（Python のように）`a < b < c` と書くことができるかどうか覚えておらず，しかも連言を表す論理演算子が（Python のように）`AND` であるか（C のように）`&&` であるかを覚えていなかったとしよう．すると，解答者がなにも参照できない場合には，選択式問題を誤って答える可能性がある．一方，実システムへのアクセスが可能であれば，このような細部は試してみることで確認することができる．もちろん，このような知識も無いより有る方が良いのは間違いないのであるが，たとえば試験であれば，「試してみる」ことに要する時間が，ちょうどその分のペナルティになると考えることができる．

記述式問題においては，このような問題はなく，受講者の実力をより正確に反映することができると考えられる．このように，受験者の評価という観点では，記述式の方が問題形式として優れている．一方で，記述式は，採点に要するコストが高いという問題点がある．

あらかじめ用意された正解と比較して採点するという方法をとることは難しい．同一の問題でも非常に多くの正解が存在しうるからである．たとえばスペースの数にしても，スペースを置いても置かなくても良いところは多数存在する．1 個以上のスペースはほぼ常に同一視できるだろうが，それも文字列リテラルの中は例外になり，これを判断するためにはやはりパーザが必要になる．そもそも複雑な問題であれば，本質的に異なる正解を列挙するのは困難である．

したがって，通常は出題者が目視によって正解か否かを判断することになるが，これは出題者に多大な負荷をかけることになる．このことから，頻繁に小テストを行ったり多くの宿題を出したりすることが現実的に困難になっている．

この問題を解決するために，記述式問題に対して受講者が提出した答案 (SQL 文) が正しいかどうかを自動的に判定するシステムを開発する．

2. システム

2.1 アイディア

システムの基本的なアイディアは単純である．受講者が提出した SQL 文を実際に実行する．文法エラーや実行時エラーが生じれば，もちろんその解答は不正解である．

エラーが無かったときには，実行結果を正解の実行結果と比較する．これらが異なっていたときには，解答は不正解である．

実行結果が一致している場合であるが，理論的には，提出された答案を正解と断じることはできない．実行を行ったデータセット（テーブルの内容）が，たまたま結果が一致するようなデータであって，別のデータセットを用いると実行結果が異なることになる，という可能性が排除できないからである．たとえば，上述の記述式の問題の例で，データセットに price の値がちょうど 2000 であるデータが無かったとしたら，「2000 未満」でなく「2000 以下」と書かれた SQL 文と，正解となる SQL 文との結果は一致してしまう．

これは本手法の持つ問題点ではあるが，ソフトウェア品質保証において，テスティングという手法が持つ本質的な問題点 [2] と同一である．そこで，境界値分析，同値分析などのソフトウェアテスティングの標準的な手法 [3] を参照して，このような事態をできるだけ避けるのが現実的な選択となる．

2.2 データ

データセットの準備方法を，問題の種類毎に説明する．

2.2.1 参照

SELECT 文の実行結果を比較する．

例として，次の問題を考える : テーブル table1 から，カラム price の値が 1000 以上 2000 未満であるか，またはカラム startdate の値が 2019 年 4 月 3 日より前のレコードの，カラム id, price, startdate の値を，startdate が新しい順に表示せよ．

想定する解答は以下の通りである :

```
SELECT id, price, startdate FROM table1
  WHERE (1000 <= price AND price < 2000)
     OR (startdate < '2019-04-03')
  ORDER BY startdate DESC
```

この場合, 境界値および中間の値として, 以下のデータを入れるようにすべきである.

- price の値が，812, 1000, 1001, 1342, 1999, 2000, 2198
- startdate の値が，’2007-08-15’, ’2019-04-02’, ’2019-04-03’, ’2025-01-13’

整列が正しく指定されていることを確認するために，主キー id の順序と startdate の降順での順序が異なっているレコードを用意する必要がある.

以上の考察の下，具体的には表 1 のデータを準備する．なお, 指定されていないカラムの値は任意であり, たとえばすべて NULL を指定しても構わない.

表 1　SELECT 文例題用のデータ

id	price	startdate
1	812	’2025-01-13’
2	1000	’2025-01-13’
3	1001	’2025-01-13’
4	1999	’2025-01-13’
5	2000	’2025-01-13’
6	2198	’2025-01-13’
7	812	’2007-08-15’
8	812	’2019-04-02’
9	812	’2019-04-03’

2.2.2　更新

この節では，UPDATE 文 , INSERT 文，DELETE 文を扱う.

SELECT 文の場合には，単純に提出プログラムを実行して，結果を正解と比較したのだが，更新の場合には，採点プログラムは，各提出につき，以下の処理を行う.

1. 解答に関係するテーブルをクリアする.
2. データをクリアしたテーブル内に生成する
3. 提出された SQL 文を実行する.
4. 結果確認用に，全テーブルのレコードを表示する SQL 文を実行し，実行結果を正解と比較する.

次の例題を考える :

table1 の startdate の値が 2019 年 1 月 1 日より前であるレコードについて，price の値から 100 を減ぜよ.

正解例には，次の SQL 文がある :

```
UPDATE table1
  SET price = price - 100
  WHERE startdate < ’2019-01-01’
```

前節と同様の考察を行うことによって，表 2 のデータを用意することとする.

この例題では，UPDATE 文を扱ったが，INSERT 文や DELETE 文でも考え方は同じである．ただし，INSERT 文では，ステップ 2 は不要である.

2.3　実装

自動採点を行うプログラムは，Python (version3) のスクリプトとして実装した．用いたソフトウェアは表 3 の通りである.

表 2　UPDATE 文例題用のデータ

id	price	startdate
1	1500	’2018-07-13’
2	1000	’2018-12-31’
3	3400	’2019-01-01’
4	2900	’2020-05-10’

表 3　試行に用いたソフトウェア

項目	記述
OS	Microsoft Windows 10
DBMS	Maria DB (XAMPP 7.2.23)
Python	version 3.7
IF	mysqlclient 1.4.2

```
-----#examId01#--
-----#begin#--
-----#sid#--
2924999
-----#name#--
山田太郎
-----#q1#--
SELECT *
  FROM table1
  WHERE id = 10

-----#q2#--
DELETE FROM table1
  WHERE price IS NULL

-----#end#--
```

図 2　提出ファイルの例

受講者が提出するファイルの例を図 2 に示す．このように，複数の問題に対する解答を，1 つのテキストファイルに記述させる．行セパレータは，正規表現“^.*---#(\w+)#---\s*$”にマッチする行である．採点プログラムは，各問題の識別子を保有しており，この例では，q1とq2である．特別な識別子 sid と name は，それぞれ，受講者の識別番号（学籍番号）と氏名を示す.

これだけの情報があれば, 提出されたファイルから, 各受講者ごとの解答を抽出することが可能である.

3. 問題配布・答案回収システム

この SQL 評価システムは, 授業内の小テストおよび学期末試験で使用することを想定している. 問題の配布と答案の回収をどのように行うかを検討する必要がある. 特に, 学期末試験に関しては, 以下の点を考える必要がある.

- 試験開始時刻に問題を配付する. それ以前に学生には問題は見えてはいけない.
- 試験終了時刻に解答を終了させる.
- 不正行為を容易には実行できなくさせる.

当学科では, 学生には入学時にノート型の PC を一人一台貸与している. OS は, Microsoft Windows である. バージョンは年によって異なるが, 今年度に本科目を受講した大部分の学生の PC は, Windows 10 Pro が動作している. 通常の授業では, 多くの学生はインターネットに接続している無線ネットワークに PC を接続した状態で授業を受けている.

授業の際に配付する講義資料などはこの PC に保管する学生がほとんどであるし, 演習もこの PC を用いて実施している. 試験の際にはこの PC 内に保存してある資料はすべて参照可ということにしている. これは伝統的な試験の考え方とは異なるかもしれない. しかし, 試験において測るべき能力は, たとえば SQL コマンドを暗記できる力ではなく, 如何に SQL を運用するか, である. 今日では, インターネットから簡単に情報を入手できるのだから, 検索すれば入手できる情報を記憶しているかどうかを試験で問うてもあまり意味はなく, その状況の下でタスクを実行できる能力を測るべきである.

この議論を敷衍すれば, 試験時においてもインターネットアクセスを認めるべきである. しかしながら, 現実問題としてそれは困難である. インターネット上に待機している友人に問題を転送して代わりに解いてもらい, 答だけを受取る, といった不正行為が, 比較的容易に実行できてしまうからである.

このような考察の下, 以下のようなシステムを構築した.

- インターネットとは切り離した独立なネットワークを用意し, このネットワーク上に, 無線 LAN ルータを置く. アクセスポイントが有効となる.
- 学生の PC は, 上記の無線アクセスポイントに接続させる.
- Windows PC を 1 台用意し, これをサーバとしてネットワーク上に配置し, ルータに有線で接続する. このマシン上で Apache ウェブサーバを運用し, 試験用のコンテンツを配置する.
- 試験用のトップページに受講者がアクセスし, ページをブラウザ上で開いている間は, ブラウザからウェブサーバに対して一定間隔（20 秒程度）でポーリングを行うようにする. サーバ側はポーリングによるアクセスを監視し続ける. ある受験者から, 起こるべきアクセスが無い場合には, 試

テーブル st1 に関して, 以下を実施する SQL 文を書け.

(1) フィールド fE の値が 2018 年 3 月 3 日以前であるレコードを表示する.
(2) フィールド fD の値が red でないレコードについて, fB の値を-99 に変更する.
　(「でない」に注意すること).
(3) フィールド fB の値が-10 より小さいレコードを削除する.
(4) 次に示すレコードを 1 つ挿入する:
　fA の値は 91, fB の値は-20, fC は文字列 rabbit, fD は文字列 purple,
　fE は 2017 年 10 月 9 日

図 3　問題例

```
CREATE TABLE st1 (
  fA int NOT NULL,
  fB int NOT NULL,
  fC varchar(10) COLLATE utf8_bin,
  fD varchar(20) COLLATE utf8_bin,
  fE date NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
```

図 4　使用したテーブルの定義

験監督者にその受験者名を警告する．ポーリングアクセスは，トップページに記述する JavaScript コードによって実現している．このため，受験者には，試験時間中は，トップページをブラウザにロードしたままにするように指示する．（必ずしも表示させる必要はなく，ブラウザのどこかのタブに置いておけばよい）

- 試験開始時刻になるまでは，問題にはアクセスできない．試験開始時刻に，試験監督者が監督者用のページから操作を行うと，問題が受講者に提示される．なお，現在の実装では，受講者はトップページの再読込を行う必要がある．
- 試験終了時刻になるまでは，受講者は何度でも答案ファイルをアップロードすることができる．複数回のアップロードを行った場合には，最後のファイルが有効になる．試験終了時刻を過ぎると，アップロードできなくなる．なお，原理としては，最後のファイルを除いて消去して良いのだが，最後に誤操作によって関係ないファイルをアップロードしてしまうなどの事故の際の救済措置のために，すべてのアップロードファイルを保存するようにしている．

ブラウザからのポーリングを行う理由は，インターネットへの接続を禁ずるためである．受験者が，試験用のネットワークから切断してインターネットに接続を行うと，ポーリングが途絶える．このときに試験監督者に対して学生を特定して警告が行われるので，該当の学生に注意を与えることができる．

このしくみは，完全に不正行為を防ぐものではない．複数のネットワークインタフェースを用いれば，試験用のネットワークとインターネットの双方に接続することは技術的には可能であり，これを防止するには机間巡視などの人間に頼る方法が必要である．しかし，もともと，たとえば携帯電話などを用いた不正行為防止のためにはこれは必要であり，当システムを導入した故の脆弱性と考えることはできない．

4. 試行

試作したシステムを実際に授業で利用して小テストを実施し，受講生に解答を行わせる試行を行った．

4.1 試行の状況

数回実施した小テストのうち，ある回の問題を（図 3）に示す．試験問題中で参照しているテーブル（図 4）と，このテーブルにインスタンスを作成するスクリプトは事前に受講生に配布してある．なおかつ，授業中で，このテーブルを利用して SQL 文を書く練習問題も実施しており，受講生はこのテーブル自体には習熟していると考えられる．

4.2 フォーマットの問題

試行の結果明らかになったのは，ファイルを正しく作成できない学生が一定数いるということである．

```
-----#examId01#--
-----#begin#--
-----#sid#--

-----#name#--

-----#q1#--

-----#q2#--

-----#end#--
```

図 5 初回に配布した解答用シート

初回に配布した解答用シートは，図 5 のような形のものであった．「-- で始まる行を改変してはならず，記入はその次の行以下に行う」ということを，文書でも示し，口頭でも注意したにもかかわらず，行セパレータと同一行から解答を書き始めた者が相当数存在した．

このため，やむを得ず解答用シートを図 6 に示すように修正した．その結果，誤ったフォーマットで提出する学生は減少した．

テキストファイルを指示されたフォーマットで作成するのはコンピュータのスキルとして非常に基本的なものであるので，本論文の主題とは別に，指導法の検討を要するところである．

5. システム拡張の方向

本節では，今回試作した自動採点システムを踏まえ，今後の拡張方向について検討する．

5.1 提出フォーマット

前述のように，現在のユーザインタフェースは，テキストファイルを提出させる方式である．この方式は，教育的な見地からは望ましい面もある一方で，本来集中すべき SQL に関する問題以外にも，解答者に負荷を与えているという側面は否めない．

これに対して，問題ごとに解答領域を用意し，そこにコピー・ペーストによって解答を貼り付けさせるようにする方式も考えられる．このような方式にすれば，解答者にフォーマットに関して余計な考慮を要求することはなくなる．一方で，従来方式であれば，複

```
-- 小テスト 0523 解答用ファイル ----------#sExam0523#----

-- スペースが足りない場合には，自由に行を追加して良い．
-- ただし，「--」で始まっている行の内容を変更してはならない．

-- この行もしくはこの行より上に何かを書き込んではならない．----#begin#-------
-- 学籍番号を下の行に記述せよ．この行を変更してはならない．----#sid#----------

-- 氏名を下の行に記述せよ．この行を変更してはならない．--------#name#----------

-- (1) の解答を下の行に記述せよ．この行を変更してはならない．----#1#-----------

-- (2) の解答を下の行に記述せよ．この行を変更してはならない．----#2#----------

-- (3) の解答を下の行に記述せよ．この行を変更してはならない．----#3#----------

-- (4) の解答を下の行に記述せよ．この行を変更してはならない．----#4#----------

-- この行もしくはこの行より下に何かを書き込んではならない．------#end#--------
```

図 6　改変した解答用シート

数の問題に対して単一のファイルを用意すれば，一度のアップロードによって問題の提出が完了していたのに比較して，何度もコピー・ペーストを実行しなければならなくなるという問題が生じる．

しかし，総合的に見れば，問題ごとの解答領域を用意する手法の方が優れていると考えられ，今後の開発を図りたい．

5.2　Moodle との連携

Moodle [4] は，大学等の高等教育機関で最も広く用いられている講義管理システム (LMS=Learning Management System) の一つである．

今回開発した提出システムでは，独自に受講生を認証している．これは，インターネットに接続していない単独のネットワークで行う以上やむを得ない選択である．

しかし，期末試験ほどの厳密性を要しない状況も考えられる．たとえば，本システムを用いて学生に復習をさせるような場合に，ある問題セットに完答した場合にはインセンティブを与えるような場合が該当する．このような状況では，Moodle の認証システムと採点プログラムを連動させられれば，提出と採点結果の記録が Moodle 内で行えることになり，担当教員にとっての利便性が高まると考えられる．

5.3　Python のテストフレームワークの利用

今回の実装では，採点プログラムは単独のプログラムとして作成している．本プログラムの機能はある意味で，ソフトウェアテスティングに近いと考えられる．プログラムを決められた入力に対して実行して，あらかじめ用意されている正解と比較するという流れだからである．

このような作業の自動化を支援する仕組みとして，今回実装に用いた Python 言語では，テストフレームワークとして，Robot, Pytest, PyUnit などのしくみが用いられている．今後は，これらのフレームワークの利用を検討することで，開発の効率化を図りたい．

5.4　テストデータの自動生成

2.2 で検討したように，問題に応じて適切なテストデータを生成する必要がある．プログラミング言語に関するテストデータ作成についてはさまざまな研究があり，たとえば記号実行を用いてテストの網羅性を上げる [5] など，さまざまな手法が開発されている．

SQL は，手続き的な言語ではないので，上述の手法が直ちに適用できるわけではない．しかし，対象とする構文を限定することによって，自動的にテストデータを生成する余地はあると考えられる．

6.　関連研究

C や Java などの一般のプログラミング言語によるプログラムを自動採点することについては，さまざまな研究がある．提出されたプログラムを静的に解析するものと動的に実行するものに大別される．

静的な解析のもっとも単純な方法は，提出されたプログラムを，正解プログラムとテキストとして比較するものであるが，同じ意味内容を持つプログラムにもたくさんのテキストが存在するので，単純な比較では

妥当な評価を与えることはできない．

この問題に対して，なんらかの類似度を用いて評価を行う手法が提案されている．たとえば小川らによる [6] では，構文木を用いている．提出されたプログラムと正解プログラムの抽象構文木に対して適用する類似度関数が提案されており，この関数の値にもとづいた評価を行っている．Wang らによる研究 [7] では，意味論的な類似度が提案されている．プログラムの意味論を，System dependence graph (SDG) と呼ばれるグラフで表現している．同じ振る舞いをするプログラムが同一の SDG に対応づけられるわけではないが，標準形を定義することができて，プログラムは，その振る舞いを変えずに標準形に変形できることが示されており，これにもとづいて類似度の評価が行われる．

プログラムの動的な実行にもとづいて評価を行う方法も多数提案されている．Virtual Programming Lab [8] のように，Moodle などの LMS のプラグインの形で実現されているシステムもあり，また，Eclipse などの統合開発環境 (IDE) との連携を重視するツールもある [9].

関連して，特にアルゴリズムを重視するプログラミング課題に対応する自動採点システムとして，オンラインジャッジと呼ばれるシステム [10] が，世界各地のサイトで運用されている . 国内でも，最大規模である AtCoder [*1] のほか，会津大学の Aizu Online Judge (AOJ) [11] などがある．本研究の手法は，これらのオンラインジャッジシステムを参考にしている．

一方で，SQL に関する自動採点の研究はあまり行われていない．坂口らは文献 [12] で，学生が書いた SQL を自動採点する手法を提案している．しかしこの提案手法は，正解となる SQL 文と提出された SQL 文とを比較する方式であり，1.1 節で指摘した問題点がそのまま残っている．

7. おわりに

本論文では，データベース操作言語 SQL の初学者に対する教育を行う授業において，授業内小テストや宿題などを想定して，受講者の作成する解答を自動的に採点するシステムを試作し，その評価を行った．また，その評価に基づいて，今後の拡張方向についてさまざまな角度から検討を行った．

今後はこの検討に基づき，さらに効果的な採点システムをめざし，システムの改良をつづけたい．また，同様のレベルの授業を行っている他の科目 (Excel を教える「情報基礎演習 II（アプリケーション）」や，本科目の次の段階に位置づけている「データベース演習 I」などの科目) においても，同様の自動採点システムを構築することを検討していきたい．

註

*1 https://atcoder.jp

参考文献

[1] 中山清喬, 飯田理恵子. スッキリわかる SQL 入門. インプレス, 2013.

[2] Edsger W. Dijkstra. The humble programmer. *Communications of the ACM,* Vol. 15, No.10, pp. 859-866, 1972.

[3] Glenford J. Myers, Corey Sandler, and Tom Badgett. *The Art of Software Testing, 3rd Edition.* Wiley, 2011.

[4] M. Dougiamas and P. C Taylor. Moodle: Using learning communities to create an open source course management system. In *Proceedings of the EDMEDIA 2003 Conference, Honolulu, Hawaii,* 2003.

[5] Corina S. Păsăreanu and Neha Rungta. Symbolic pathfinder: Symbolic execution of java bytecode. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering,* ASE '10, pp. 179-180, New York, NY, USA, 2010. ACM.

[6] 小川弘迪, 小林亜樹. プログラミング課題の自動採点に向けた構文木上のカーネル法による類似度関数の提案. 第 80 回全国大会講演論文集, Vol. 2018, No. 1, pp. 661-662, Mar 2018.

[7] Tiantian Wang, Xiaohong Su, Yuying Wang, and Peijun Ma. Semantic similarity-based grading of student programs. *Information and Software Technology,* Vol. 49, No. 2, pp. 99-107, 2007.

[8] Juan Carlos Rodríguez del Pino, Enrique Rubio Royo, and Zenón José Hernández Figueroa. A virtual programming lab for moodle with automatic assessment and anti-plagiarism features. 2012.

[9] Ricardo Alexandre Peixoto Queirós and José Paulo Leal. Petcha: A programming exercises teaching assistant. In *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education,* ITiCSE '12, pp. 192-197, New York, NY, USA, 2012. ACM.

[10] Szymon Wasik, Maciej Antczak, Jan Badura, Artur Laskowski, and Tomasz Sternal. A survey on online judge systems and their applications. *ACM Comput. Surv.,* Vol. 51, No. 1, pp. 3:1-3:34, January 2018.

[11] 渡部有隆. オンラインジャッジの開発と運用 -Aizu Online Judge-. 情報処理, Vol. 56, No. 10, pp. 998-1005, Sep 2015.

[12] 坂口龍一, 中田大智, 津曲悠貴, 水谷泰治. SQL 演習における select 文の自動採点システムの提案. 電子情報通信学会総合大会講演論文集, Vol. 2016, No. 1, p. 215, 2016.