8. 行の挿入，SQL問い合わせのまとめ

（データベース演習）

URL: https://www.kkaneko.jp/cc/de/index.html

金子邦彦

全体まとめ

- **リレーショナルデータベース管理システム**には，**さまざまな種類**がある

  Access, MySQL, SQLite, Oracle など

- MySQL, SQLite などは，**オンラインで利用**できる

- **SQL** は、**リレーショナルデータベースシステム**のさまざまな機能を使える言語.
- **SQL は世界標準**なので，SQL のスキルは，さまざまなリレーショナルデータベース管理システムで通用する
- データベース操作（**行の挿入・削除，更新**）も**SQL で可能**

アウトライン

番号	項目	説明時間の目安
8-1	SQL の特徴	4分
8-2	SQL による行の挿入・削除，更新	6分
8-3	オンラインのプログラム開発環境	8分
8-4	オンラインで MySQL を利用できる Paiza.IO の紹介	4分
8-5	さまざまな問い合わせ（クエリ）	19分
8-6	テーブルの分解と結合	6分

8-1 は復習である
8-5, 8-6 は SQL 全般を復習するための演習である

各自、資料を読み返したり、課題に取り組んだりも行う

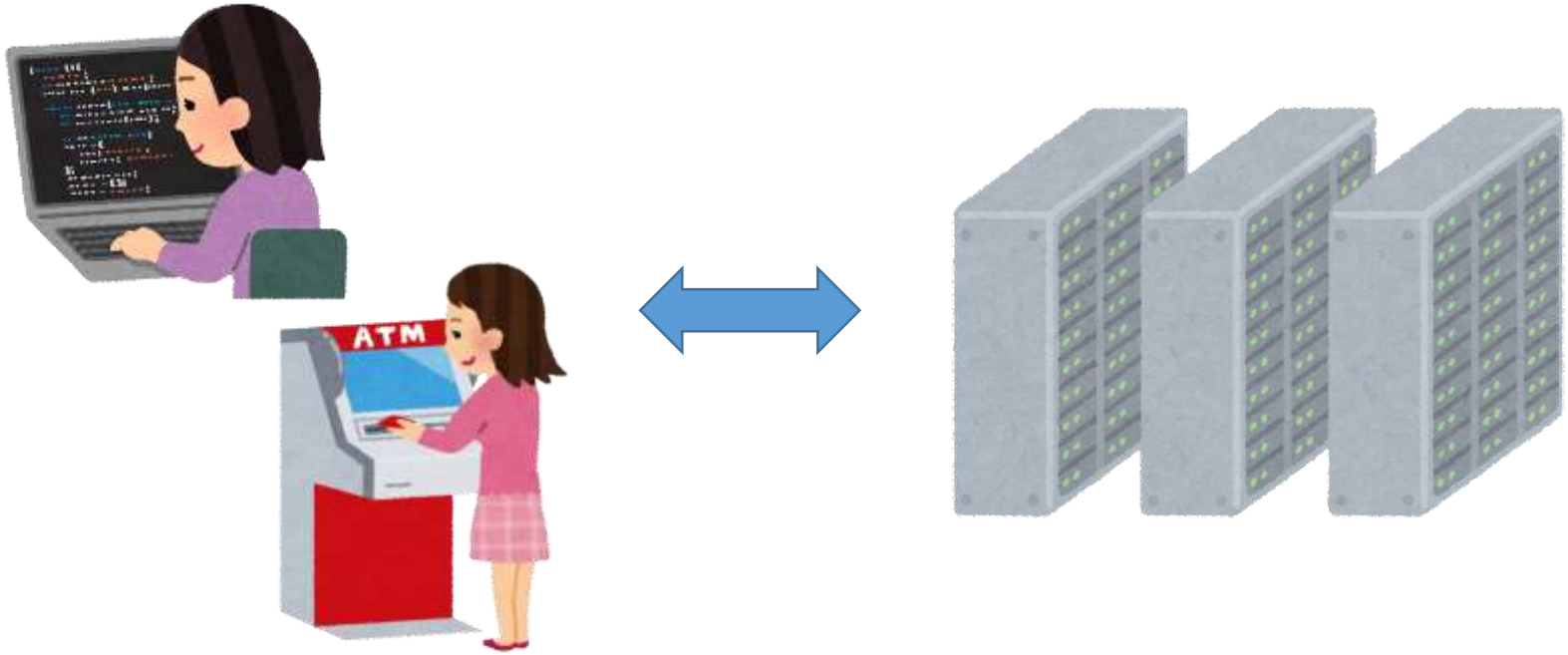
8-1. SQL の特徴

- **SQL** は、**リレーショナルデータベースシステム**のさまざまな機能を使える言語

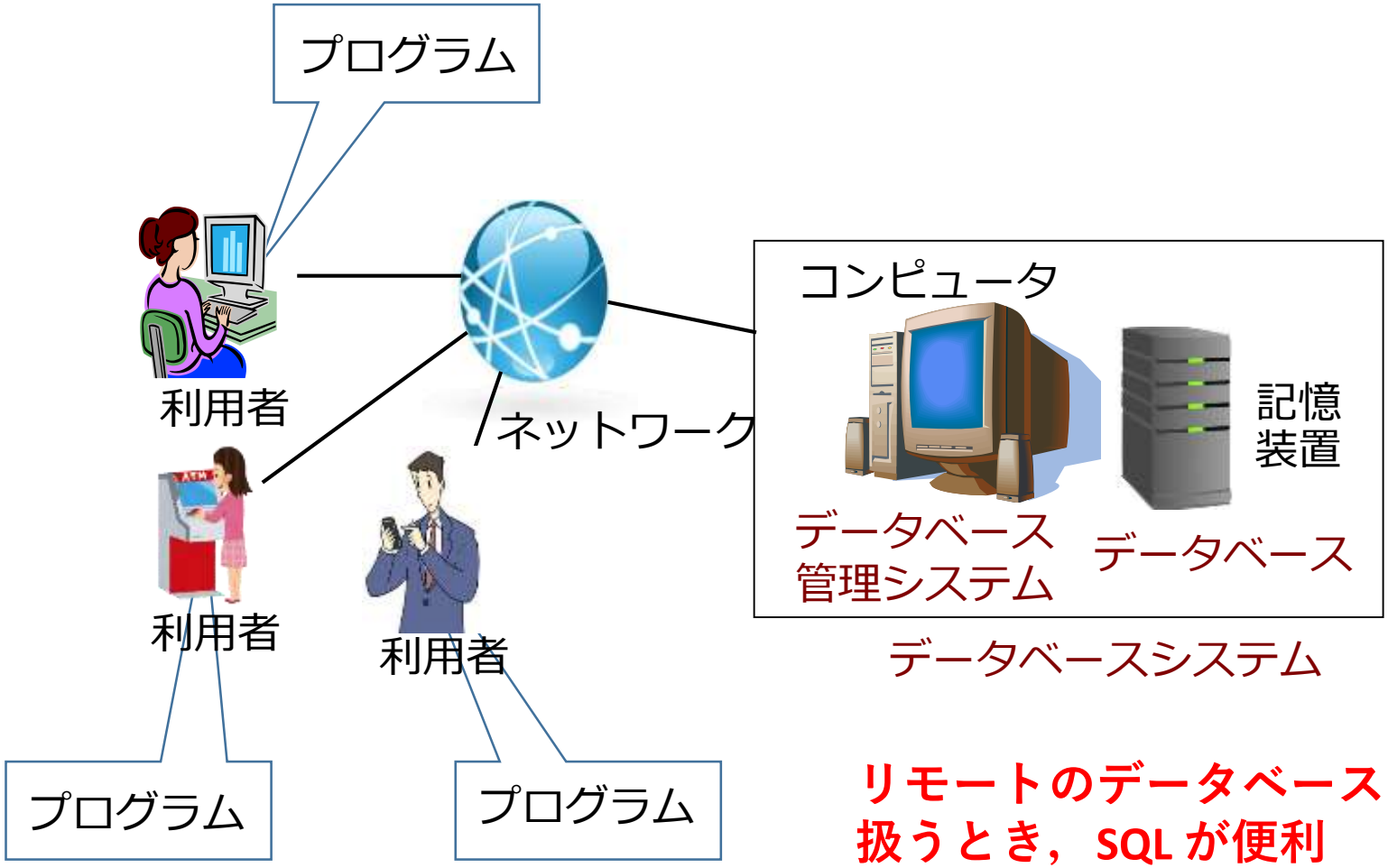
  問い合わせ（クエリ）、

  テーブル定義、

  その他の操作



コンピュータどうしがつながるときも「SQL が便利」という場合も



リモートのデータベースシステムを扱うとき，SQL が便利
（アプリの中に SQL のプログラムが入っているということも）

SQL による問い合わせ（クエリ）の例

① **SELECT * FROM** 商品;

② **SELECT** 名前, 単価 **FROM** 商品;

③ **SELECT** 名前, 単価 **FROM** 商品 **WHERE** 単価 > 80;

④ **SELECT** 受講者, **COUNT(*) FROM** 成績 **GROUP BY** 受講者;

⑤ **SELECT * FROM** 米国成人調査データ **ORDER BY** 年齢;

⑥ **SELECT * FROM T, S;**

⑦ **SELECT * FROM T, S WHERE a = b;**

ツール，SQL の両方が大切

ツール

マイクロソフトAccess の**クエリのデザインビュー**（Access だけで動くツール）など

マウスと
キーボード

ＳＱＬ（SQLは**世界標準**）

```
SELECT ID, 商品, 単価
FROM 商品;
```

コマンド言語

SQL の良いところ

- **ＳＱＬ**は、すべての**リレーショナルデータベース管理システム**で通用する**共通**言語

  リレーショナルデータベース管理システムの例

  Access, SQL Server, Oracle, MySQL, PostgreSQL, SQLite, Firebird, ・・・

- **ＳＱＬ**はコマンド言語なので、自動実行が簡単.

テーブル定義

リレーショナルデータベースの**テーブル定義**は,

・**テーブル名**

・各属性の**属性名**

・各属性の**データ型**

<u>など</u>を定義すること. , 参照整合性制約などの**制約**の指定も行う

属性のデータ型

Access の主なデータ型	SQL のキーワード	
	NULL	空値
短いテキスト	**char**	文字列
長いテキスト	**text**	文字列
数値	**integer, real**	整数や浮動小数点数
日付／時刻	**datetime**	日付や時刻など
Yes／No	**bit, bool**	ブール値

※ **整数**は integer, **浮動小数点数**（小数付きの数）は real
※ **短いテキスト**は**半角 255文字分**までが目安
それ以上になる可能性があるときは**長いテキスト**

8-2. SQL による行の挿入・削除，更新

SQL によるデータベース操作

操作の種類	SQL のキーワード
新しい行の挿入	**INSERT INTO**
条件に合致する行の削除	**DELETE FROM WHERE**
既存のデータの更新	**UPDATE SET WHERE**

挿入

テーブル T

名前	昼食	料金

空

テーブル T

名前	昼食	料金
A	そば	250
B	カレーライス	400
C	カレーライス	400
D	うどん	250

○ 実行例（テーブル定義，３行の挿入，確認）

```
create table T(名前 text, 昼食 text, 料金 integer);
insert into T values('A', 'そば', 250);
insert into T values('B', 'カレーライス', 400);
insert into T values('C', 'カレーライス', 400);
insert into T values('D', 'うどん', 250);
select * from T;
```

```
名前    昼食    料金
A       そば    250
B       カレーライス    400
C       カレーライス    400
D       うどん  250
```

更新

テーブル T

名前	昼食	料金
A	そば	250
B	カレーライス	400
C	カレーライス	400
D	**うどん**	**250**

テーブル T

名前	昼食	料金
A	そば	250
B	カレーライス	400
C	カレーライス	400
D	**うどん**	**300**

○ **演習**として次を行う

```
create table T(名前 text, 昼食 text, 料金 integer);
insert into T values('A', 'そば', 250);
insert into T values('B', 'カレーライス', 400);
insert into T values('C', 'カレーライス', 400);
insert into T values('D', 'うどん', 250);
update T set 料金 = 300 where 昼食 = 'うどん';
select * from T;
```

更新を行う SQL

```
名前    昼食    料金
A       そば    250
B       カレーライス    400
C       カレーライス    400
D       うどん  300
```

削除

テーブル T

名前	昼食	料金
A	そば	250
B	カレーライス	400
C	カレーライス	400
D	うどん	300

テーブル T

名前	昼食	料金
B	カレーライス	400
C	カレーライス	400
D	うどん	300

○ **演習**として次を行う

```
1  create table T(名前 text, 昼食 text, 料金 integer);
2  insert into T values('A', 'そば', 250);
3  insert into T values('B', 'カレーライス', 400);
4  insert into T values('C', 'カレーライス', 400);
5  insert into T values('D', 'うどん', 250);
6  update T set 料金 = 300 where 昼食 = 'うどん';
7  delete from T where 名前 = 'A';
8  select * from T;
```

削除を行う SQL

```
名前      昼食      料金
B         カレーライス      400
C         カレーライス      400
D         うどん    300
```

SQL によるデータベース操作

操作の種類	SQL の例
新しい行の挿入	**insert into T values('A', 'そば', 250);**
条件に合致する行の削除	**delete from T where 名前 = 'A';**
既存のデータの更新	**update T set 料金 = 300 where 昼食 = 'うどん';**

8-3. オンラインの
プログラム開発環境

プログラム開発環境

プログラム開発環境は，**プログラミング**におけるさまざまなことを支援する機能をもった**プログラム**

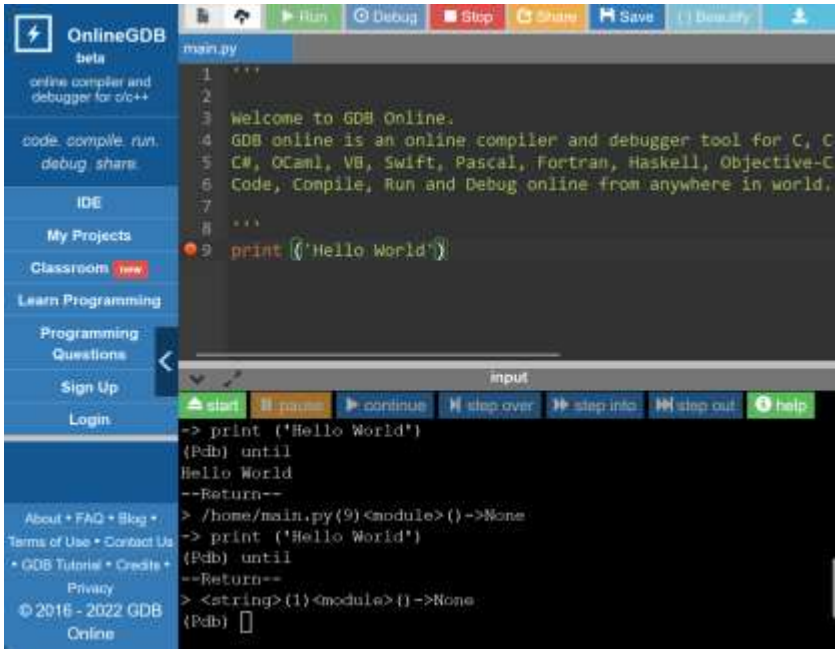
- プログラムの作成，編集（**エディタ**）
- プログラム中の誤り（**バグ**）の発見やテストの支援（**デバッガ**）
- プログラムの実行
- マニュアルの表示
- プログラムが扱うファイルのブラウズ
- プログラムの配布（**パッケージ機能**など），共有，共同編集
- バックアップ，バージョン管理

これらが簡単に行えるようになる

オンラインのプログラム開発環境

- **プログラム開発環境**の操作は，ウエブブラウザでできる
- 自分のパソコンに，特別なソフトをインストールする**必要がない**
- 機能制限がある場合が多い
- 利用登録の有無と内容，利用条件，料金については，利用者で確認のこと

プログラム作成ができるウエブサービス（オンラインの開発環境）の例 ②

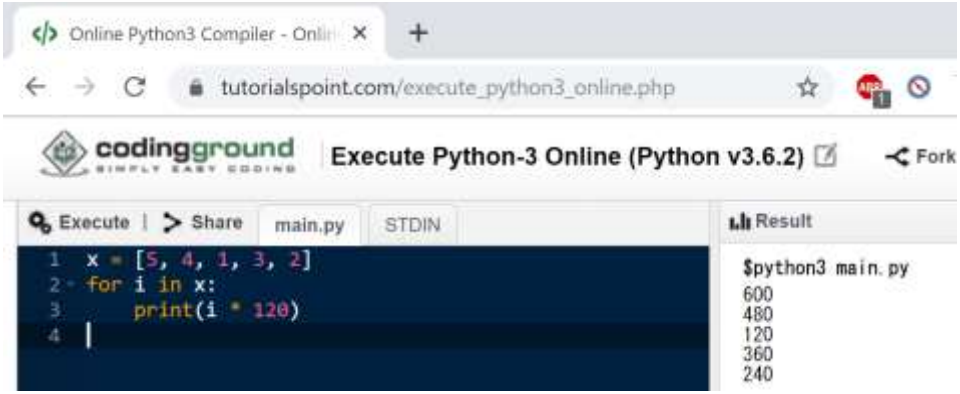


GDB online

C, C++, Java, Python, PHP, C#, OCam, VB, HTML, Ruby, Perl, Pascal, R, Fortran, Haskell, アセンブリ, Objective C, **SQLite**, Javascript, Prolog, Swift, Rust, Go, Bash

デバッガの機能あり

https://www.onlinegdb.com/



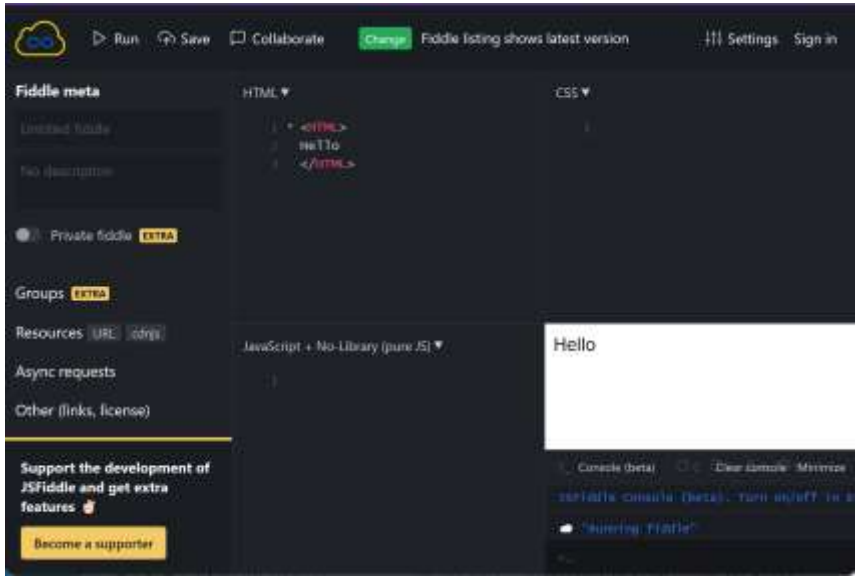
Coding Ground

Python, C, Java, JavaScript, R, Octave/MATLAB, SQL, bash, アセンブリ, **MySQL, SQLite,** その他多数

ファイル作成，ファイル読み書き，複数プログラムファイルの組み合わせ可能

https://www.tutorialspoint.com/codingground.htm

プログラム作成ができるウエブサービス（オンラインの開発環境）の例 ③



JSFiddle

HTML, CSS, JavaScript
見た目をオンラインで確認

https://jsfiddle.net/

Paiza.IO

Python, C, Java, JavaScript, R, **MySQL**
など多数
表示は日本語.
一定の条件下でファイル操作も可能

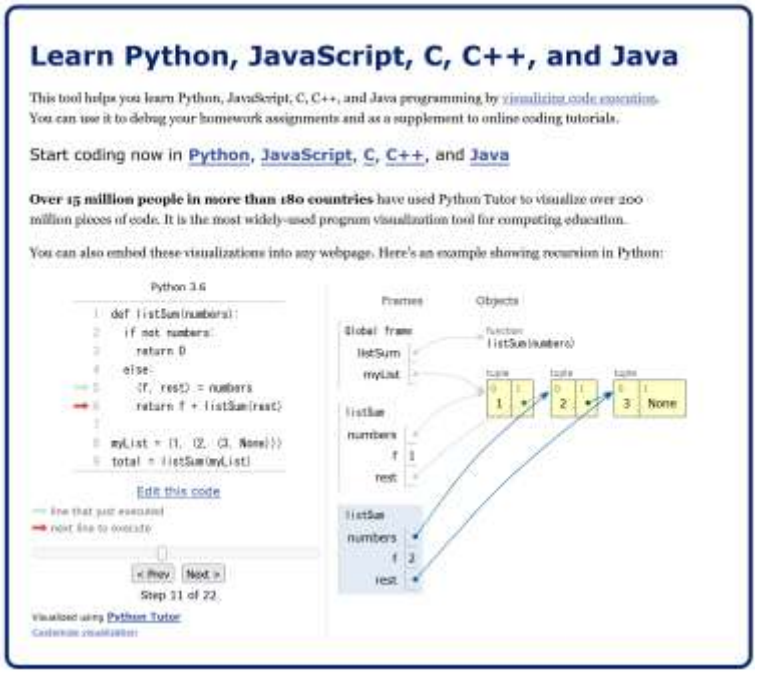
https://paiza.io/

プログラム作成ができるウエブサービス（オンラインの開発環境）の例 ①

Google Colaboratory

Python の開発環境
多数のパッケージがインストール済み
ノートブックにより、記録が簡単に残せる．ビジュアルな表示も簡単に可能
プログラムの共有も簡単

https://colab.research.google.com/



Python Tutor

Python, JavaScript, C, C++, Java
ステップ実行、オブジェクトの表示がビジュアルに

https://pythontutor.com/

8-4. オンラインで MySQL を利用できる Paiza.IO の紹介

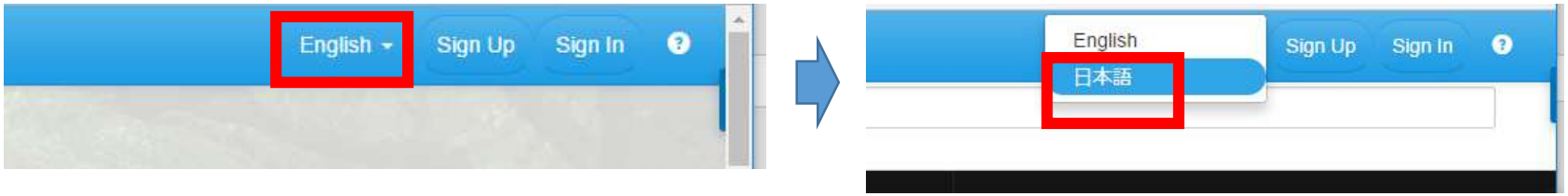
Paiza.IO の使い方

① ウェブブラウザを起動する

② 次の URL を開く

https://paiza.io/

③ もし，表示が英語になっていたら，**日本語**に切り替える

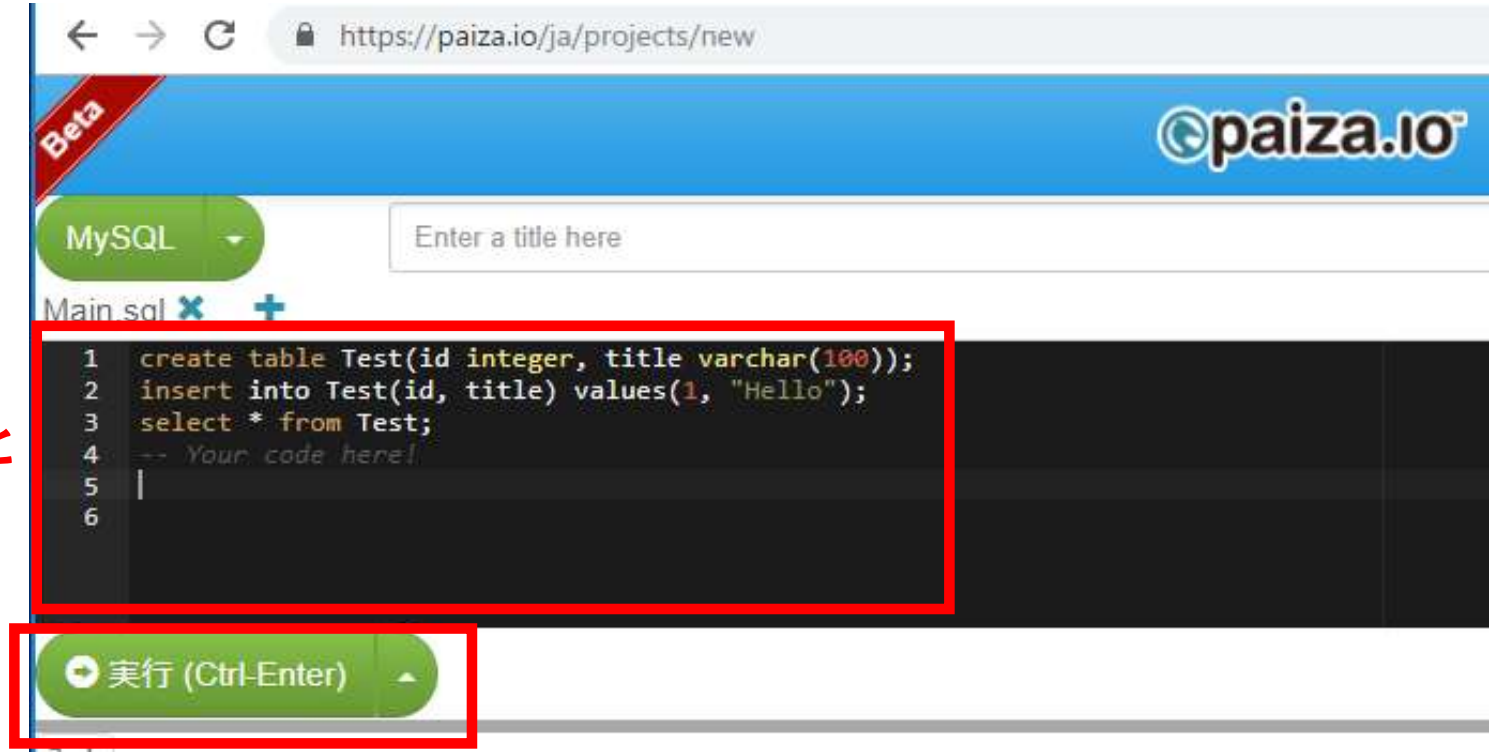


④ 「コード作成を試してみる」をクリック

⑤ 「**MySQL**」を選ぶ（**左上のボタンをクリック**するとメニューが出る）

プログラムの
編集画面

プログラムを
書き換えることができる



実行ボタン

編集画面を確認する。
すでに、**SQL が入っている**が、使わないので**消す**。

```
1  create table Test(id integer, title varchar(100));
2  insert into Test(id, title) values(1, "Hello");
3  select * from Test;
4  -- Your code here!
5
6
```

8-5. さまざまな問い合わせ（クエリ）

使用するテーブル

テーブル products

id	name	price
1	orange	50
2	apple	100
3	melon	500

テーブル sales

id	customer	pid	num
1	X	1	2
2	Y	1	3
3	X	3	1
4	Y	2	4

ここで行うこと

- **テーブル定義**

  CREATE TABLE ...

- **問い合わせ（クエリ）**

  SELECT ... FROM ...

  SELECT ... FROM ... WHERE ...

- **行の挿入**

  INSERT INTO ...

- **問い合わせ結果の保存**

  CREATE TABLE AS ...

テーブル定義 products

１から5行目に、次の **SQL を入れ**、「実行」をクリック。
エラーメッセージが出ないことを確認
※　エラーメッセージが出たときは、SQLを修正してから、
　　再度実行する

```
1  create table products (
2    id integer,
3    name text,
4    price integer
5  );
```

データ型
テキスト（文字列） **text**
数値 **integer**, **real**

ここで，補足説明

SQL プログラムは**消さず**に，
下に書き加えるようにしてください

資料のスクリーンショットでは「行番号」
も付けているので，参考にしてください

テーブル定義 sales

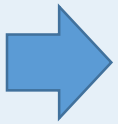
6から 11行目に、次の **SQL を入れ**、「実行」をクリック。
エラーメッセージが
出ないことを確認

```
6   create table sales (
7     id integer,
8     customer text,
9     pid integer,
10    num integer
11  );
```

ここで，補足説明

行の挿入

ID	商品名	単価
1	みかん	50
2	りんご	100
3	メロン	500



ID	商品名	単価
1	みかん	50
2	りんご	100
3	メロン	500
4	レモン	80

```
insert into 商品 values(4, 'レモン', 80);
```

テーブル名

値の並び．半角のカンマ「,」で区切る
※　文字列は半角の「'」で囲む

新しい行の挿入

12から 20行目に、次の **SQL を入れ**、「実行」をクリック。確認

```
12  insert into products values(1, 'orange', 50);
13  insert into products values(2, 'apple', 100);
14  insert into products values(3, 'melon', 500);
15  insert into sales values(1, 'X', 1, 2);
16  insert into sales values(2, 'Y', 1, 3);
17  insert into sales values(3, 'X', 3, 1);
18  insert into sales values(4, 'Y', 2, 4);
19  select * from products;
20  select * from sales;
```

id	name	price
1	orange	50
2	apple	100
3	melon	500

id	customer		pid	num
1	X	1	2	
2	Y	1	3	
3	X	3	1	
4	Y	2	4	

結合（2つのテーブルのすべてのペア）

① 次の SQL 問い合わせを実行し確認

```
21  select * from products, sales;
```

id	name	price	id	customer		pid	num
3	melon	500	1	X	1	2	
2	apple	100	1	X	1	2	
1	orange	50	1	X	1	2	
3	melon	500	2	Y	1	3	
2	apple	100	2	Y	1	3	
1	orange	50	2	Y	1	3	
3	melon	500	3	X	3	1	
2	apple	100	3	X	3	1	
1	orange	50	3	X	3	1	
3	melon	500	4	Y	2	4	
2	apple	100	4	Y	2	4	
1	orange	50	4	Y	2	4	

行（行）の順序が違っている場合がある

結合（結合条件あり）

② 次の SQL 問い合わせを実行し確認

```
23  select * from products, sales where products.id = sales.pid;
```

id	name	price	id	customer		pid	num
1	orange	50	1	X	1	2	
1	orange	50	2	Y	1	3	
3	melon	500	3	X	3	1	
2	apple	100	4	Y	2	4	

並べ替え（ソート）

③ 次の SQL 問い合わせを実行し確認．昇順.

```
24  select * from products order by price;
```

id	name	price
1	orange	50
2	apple	100
3	melon	500

並べ替え（ソート）

④ 次の SQL 問い合わせを実行し確認．降順.

```
25  select * from products order by price desc;
```

id	name	price
3	melon	500
2	apple	100
1	orange	50

数え上げ

⑤ 次の SQL 問い合わせを実行し確認

```
26  select customer, count(*) from sales group by customer;
```

customer	count(*)
X	2
Y	2

範囲指定

⑥ 次の SQL 問い合わせを実行し確認

```
27  select * from products where price between 50 and 200;
```

id	name	price
1	orange	50
2	apple	100

重複除去

⑦ 次の SQL 問い合わせを実行し確認

```
28  select distinct customer from sales;
```

customer
X
Y

ここで，補足説明

次のSQL は，SQL 問い合わせ（クエリ）の結果を，**新しいテーブルに保存**する

```
create table T as select name from products where name = 'orange';
```

【書き方】

create table ＜テーブル名＞ as ＜SQL 問い合わせ＞

問い合わせの結果をテーブルに保存

⑧ 次の SQL 問い合わせを実行し確認

```
create table T as select name from products where name = 'orange';
select * from T;
```

name
orange

ここで使用した SQL

- **テーブル定義**

  CREATE TABLE ...

- **問い合わせ（クエリ）**

  SELECT ... FROM ...

  SELECT ... FROM ... WHERE ...

- **行の挿入**

  INSERT INTO ...

- **問い合わせ結果の保存**

  CREATE TABLE AS ...

8-6. テーブルの分解と結合

- テーブルの分解と結合の演習

- 今まで入れた SQL プログラムは，すべて消しても問題ありません

① テーブル定義

次の **SQL を入れる.**

```
create table scores (
  id integer,
  name text,
  teacher_name text,
  student_name text,
  score integer
);
```

データ型

テキスト（文字列） **text**

数値 **integer**, **real**

② 実行

エラーメッセージが出ないことを確認. 表示はない.

※　エラーメッセージが出たときは、SQLを修正してから、**再度**実行する

③ 行の挿入

次の **SQL** を**書き加える**.

```
insert into scores values(1, 'database', 'k', 'kk', 85);
insert into scores values(2, 'database', 'k', 'aa', 75);
insert into scores values(3, 'database', 'k', 'nn', 90);
insert into scores values(4, 'programming', 'a', 'kk', 85);
insert into scores values(5, 'programming', 'a', 'nn', 75);
```

④ 実行

エラーメッセージが出ないことを確認．表示はない．

※　エラーメッセージが出たときは、SQLを修正してから、**再度**実行する

⑤ 問い合わせ

次の **SQL** を**書き加える**.

```
13  select * from scores;
```

⑥ 実行

確認.

※　エラーメッセージが出たときは、SQLを修正してから、**再度**実行する

id	name	teacher_name	student_name	score
1	database	k	kk	85
2	database	k	aa	75
3	database	k	nn	90
4	programming	a	kk	85
5	programming	a	nn	75

⑦ 問い合わせ

次の **SQL** を**書き加える**.

```
14  select name, teacher_name from scores;
```

⑧ 実行

確認.

※　エラーメッセージが出たときは、SQLを修正してから、**再度**実行する

name	teacher_name
database	k
database	k
database	k
programming	a
programming	a

⑨ 重複行除去

次の **SQL** を**書き加える**.

```
14  select distinct name, teacher_name from scores;
```

⑩ 実行

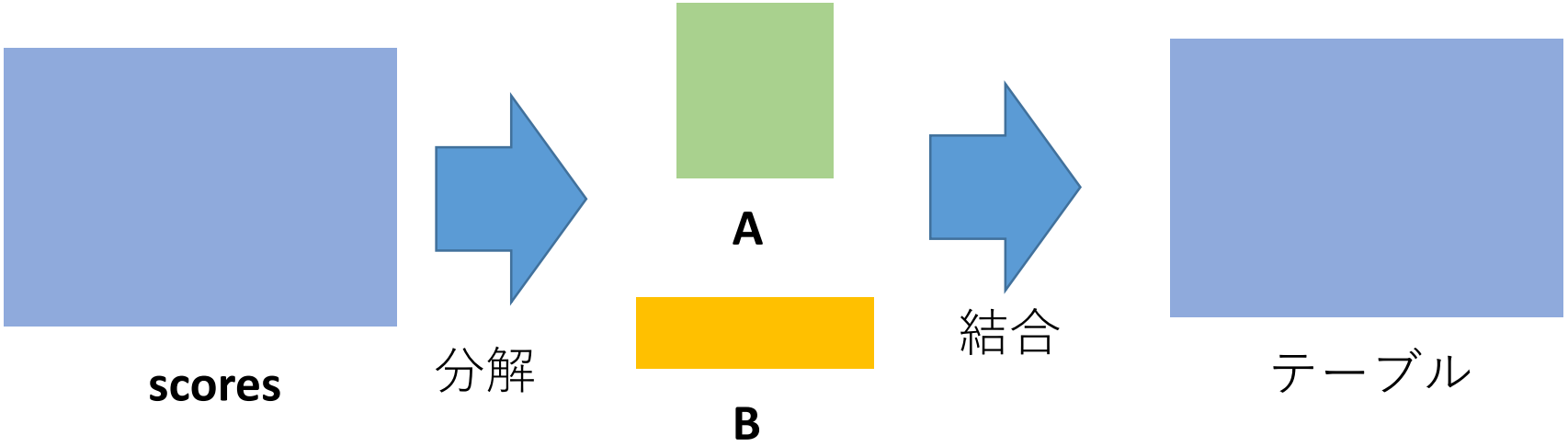
確認.

※　エラーメッセージが出たときは、SQLを修正してから、**再度**実行する

name	teacher_name
database	k
programming	a

テーブルの分解

- いまから，テーブル scores を，テーブル A, B に分解する



問い合わせの結果を、
テーブルとして保存

テーブルへの保存の方法

Access: INTO
その他のシステム(世界標準): create table … as

⑪ テーブルの分解のため，テーブル A の作成

次の **SQL** を**書き加える**.

```
create table A as select distinct name, teacher_name from scores;
select * from A;
```

⑫ 実行

確認.

※　エラーメッセージが出たときは、SQLを修正してから、**再度**実行する

name	teacher_name
database	k
programming	a

⑬ テーブルの分解のため，テーブル B の作成

次の **SQL** を**書き加える**.

```
create table B as select distinct id, name, student_name, score from scores;
select * from B;
```

⑭ 実行

確認.

※　エラーメッセージが出たときは、SQLを修正してから、**再度**実行する

id	name	student_name	score
1	database	kk	85
2	database	aa	75
3	database	nn	90
4	programming	kk	85
5	programming	nn	75

⑮ テーブル A, B の結合

次の **SQL** を**書き加える**.

```
select B.id, A.name, A.teacher_name, B.student_name, B.score
from A, B
where A.name = B.name;
```

⑯ 実行

確認.

※　エラーメッセージが出たときは、SQLを修正してから、**再度**実行する

id	name	teacher_name	student_name	score
1	database	k	kk	85
2	database	k	aa	75
3	database	k	nn	90
4	programming	a	kk	85
5	programming	a	nn	75

全体まとめ

- **リレーショナルデータベース管理システム**には，**さまざまな種類**がある

  Access, MySQL, SQLite, Oracle など

- MySQL, SQLite などは，**オンラインで利用**できる

- **SQL** は、**リレーショナルデータベースシステム**のさまざまな機能を使える言語.
- **SQL は世界標準**なので，SQL のスキルは，さまざまなリレーショナルデータベース管理システムで通用する
- データベース操作（**行の挿入・削除，更新**）も**SQL で可能**