8. SQLにおける副問い合わせと論理演算子（AND, OR）の基礎

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

金子邦彦

謝辞：この資料では「いらすとや」のイラストを使用しています

アドバイス

- アクティブな学習を実践しよう

SQLを学ぶ際に、**プログラムを変更した結果を実際に見る**ことも心がけましょう。実際のデータベース操作を通じて学習を深めます。

- 簡単なスタートから

初めはシンプルなものからスタートしましょう。反復練習しましょう。

- ステップ・バイ・ステップで応用に進む

SQLスキルを向上させるために、**少しずつ難易度を上げ**、今まで自分ができなかったことにも**チャレンジ**しましょう。

アウトライン

1. イントロダクション
2. IN
3. 副問い合わせ
4. 論理演算、AND、OR
5. 演習

SQLFiddle のサイトにアクセス

Webブラウザを使用

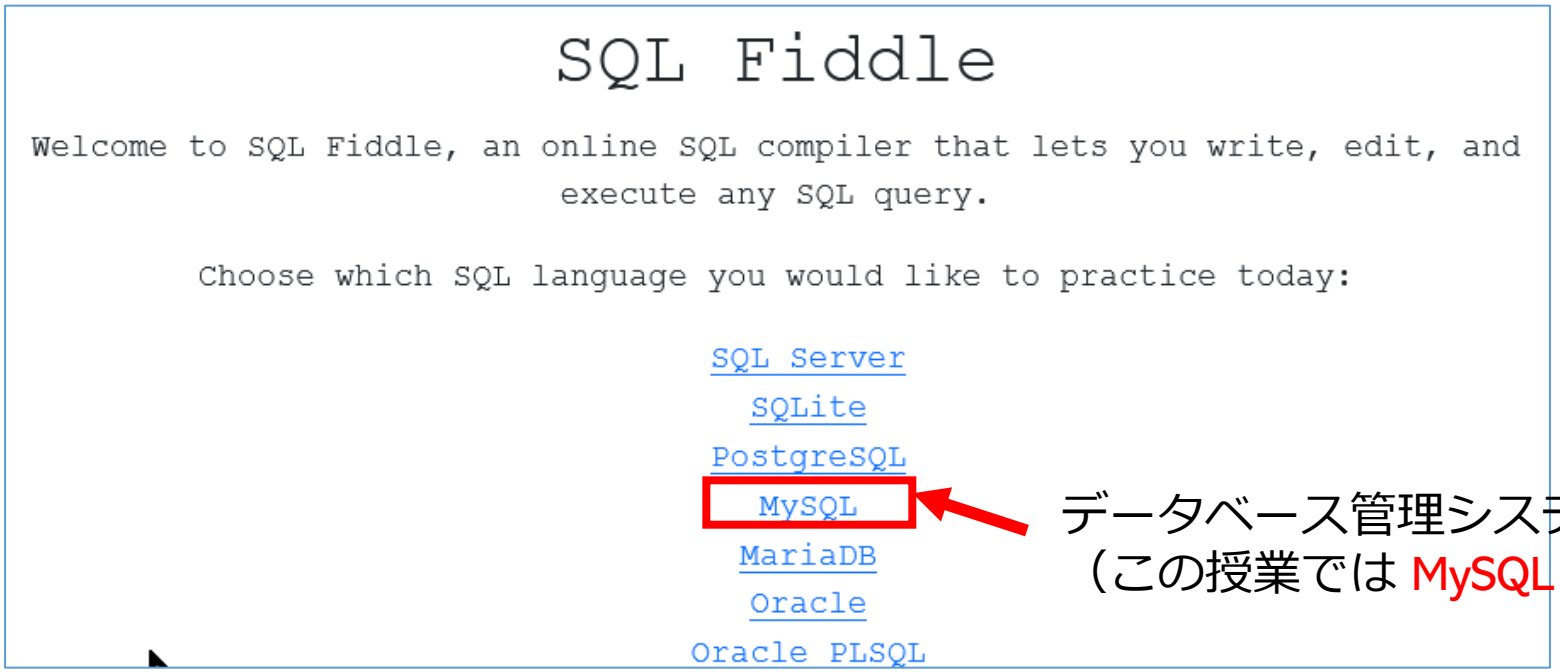
1. ウェブブラウザを開く
2. アドレスバーにSQLFiddleのURLを入力

http://sqlfiddle.com/

3. **MySQL を選ぶ**

URLが分からないときは、Googleなどの**検索エンジン**を利用。
「**SQLFiddle**」と**検索**し、表示された結果からSQLFiddleの
ウェブサイトをクリック。

SQLFiddle でのデータベース管理システムの選択



データベース管理システムの選択
（この授業では MySQL を使用）

SQLFiddle の画面

上のパネル: SQLの入力（複数可能）

- **テーブル定義** CREATE TABLE
- **データの追加** INSERT INTO
- **SQL問い合わせ**。SELECT, FROM, WHERE など

8-1. イントロダクション

リレーショナルデータベースの仕組み

- データを**テーブル**と呼ばれる**表形式で保存**
- **テーブル間**は**関連**で結ばれる
- 複雑な構造を持ったデータを効率的に管理することを可能

商品

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

関連

購入

購入者	商品番号
X	1
X	3
Y	2

SQL 理解のための前提知識

○ テーブル

データを**テーブル**と呼ばれる**表形式で保存**

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

購入者	商品番号
X	1
X	3
Y	2

○ 問い合わせ（クエリ）

- **問い合わせ（クエリ）**は、**データベース**から**必要なデータを検索、加工するための指令**
- SELECT, FROM, WHERE など、**多様**なコマンドが存在。
- **結合、集計、ソート、副問い合わせ**など、高度な操作も可能

SQL によるテーブル定義

- **テーブル名**：**成績**
- **属性名**：**科目、受講者、得点**
- **属性のデータ型**：**テキスト、テキスト、数値**
- データの整合性を保つための**制約**：**なし**

```
CREATE TABLE 成績 (
        科目 TEXT,
        受講者 TEXT,
        得点 INTEGER);
```

データ追加のSQL

成績

科目	受講者	得点
国語	A	85
国語	B	90
算数	A	90
算数	B	96
理科	A	95

```
INSERT INTO 成績 VALUES('国語', 'A', 85);
INSERT INTO 成績 VALUES('国語', 'B', 90);
INSERT INTO 成績 VALUES('算数', 'A', 90);
INSERT INTO 成績 VALUES('算数', 'B', 96);
INSERT INTO 成績 VALUES('理科', 'A', 95);
```

範囲指定の方法

- **AND を用いる範囲指定**

複数の条件「COST >= 10」，「COST <= 100 」をつなげる

```
select ID, COST
from ORDERS
where COST >= 10 and COST <= 100;
```

10以上 100以下

- **BETWEEN を用いる範囲指定**

```
select ID, COST
from ORDERS
where COST between 10 and 100;
```

10以上 100以下

「**where COST >= 10 and COST <= 100**」と
「**where COST between 10 and 100**」は、同じ結果

（10 以上 100以下）を得ることができる

演習１．テーブル定義とデータの追加

【トピックス】

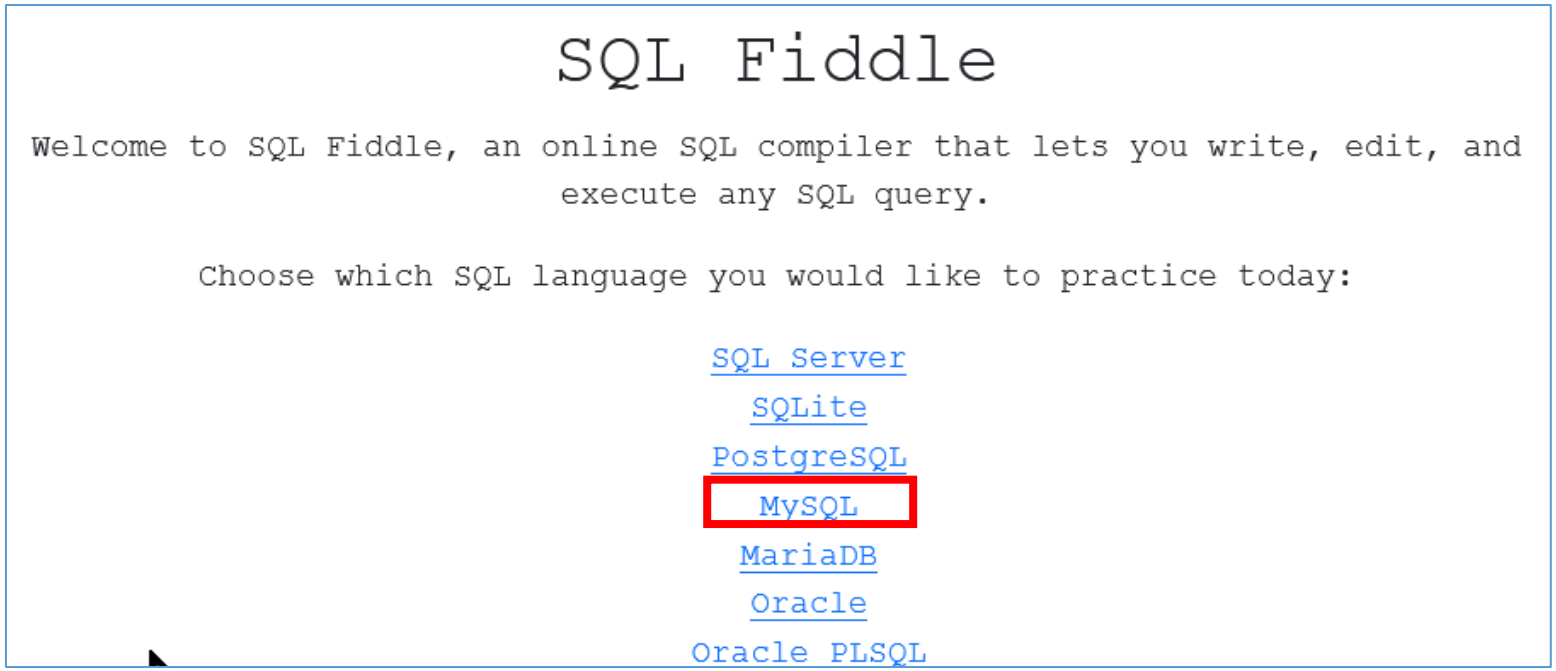
1. **SQL によるテーブル定義**
2. **SQL によるデータの追加**
3. **問い合わせ（クエリ）による確認**

Webブラウザを使用

① アドレスバーにSQLFiddleのURLを入力

http://sqlfiddle.com/

② 「**MySQL**」を選択



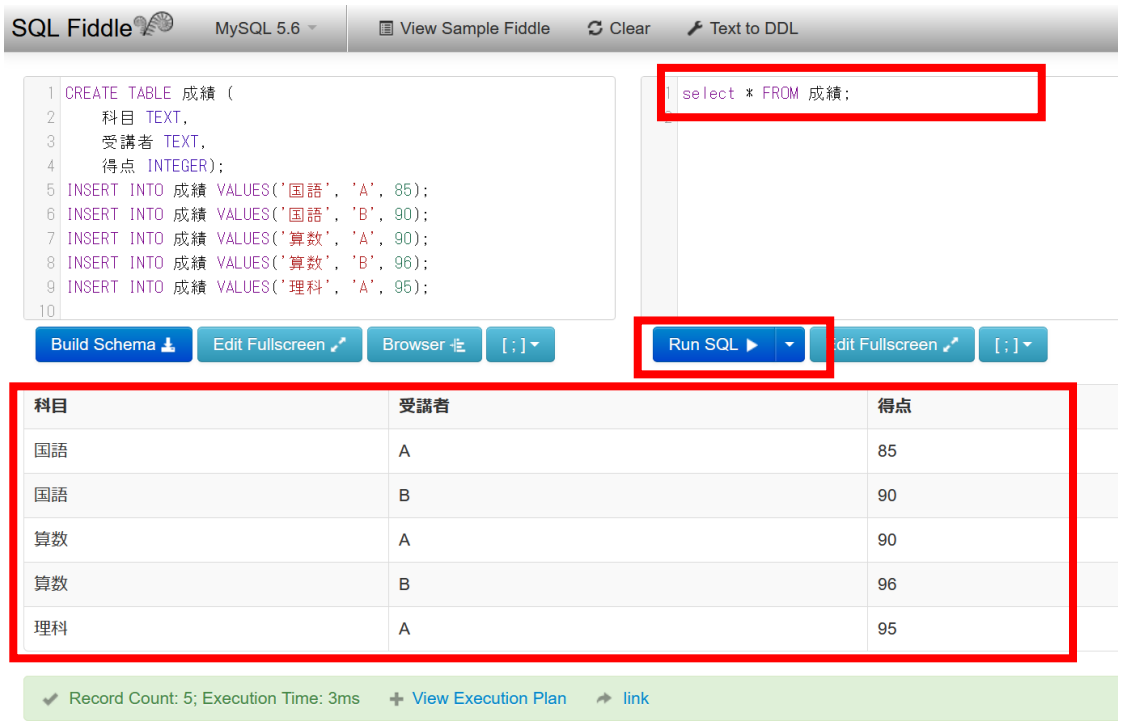
③ **上のパネル**に、**テーブル定義**と**データの追加と問い合わせ**を行う SQL を入れる。（SQLFiddleで、最初に出てくるSQLは不要なので消す）。

```
CREATE TABLE 成績 (
科目 TEXT,
受講者 TEXT,
得点 INTEGER);
INSERT INTO 成績 VALUES('国語', 'A', 85);
INSERT INTO 成績 VALUES('国語', 'B', 90);
INSERT INTO 成績 VALUES('算数', 'A', 90);
INSERT INTO 成績 VALUES('算数', 'B', 96);
INSERT INTO 成績 VALUES('理科', 'A', 95);
select * FROM 成績;
```

④ 「**Execute**」をクリック

SQL 文が**実行**され、結果が表示される。

⑤ 下のパネルで、**結果を確認。**



演習２．範囲指定

【トピックス】

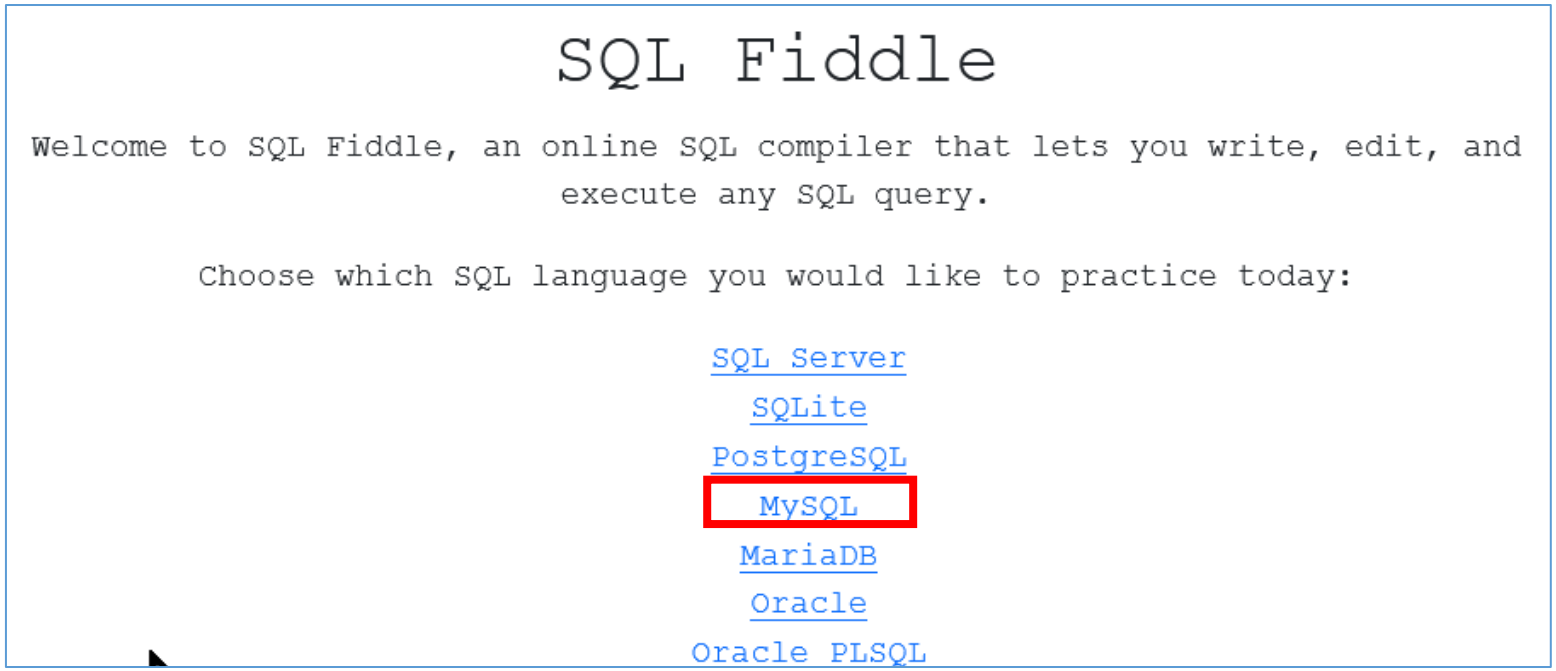
1. **複数の条件を AND で連結することによる範囲指定**
2. **BETWEEN と AND による範囲指定**

Webブラウザを使用

① アドレスバーにSQLFiddleのURLを入力

http://sqlfiddle.com/

② 「**MySQL**」を選択



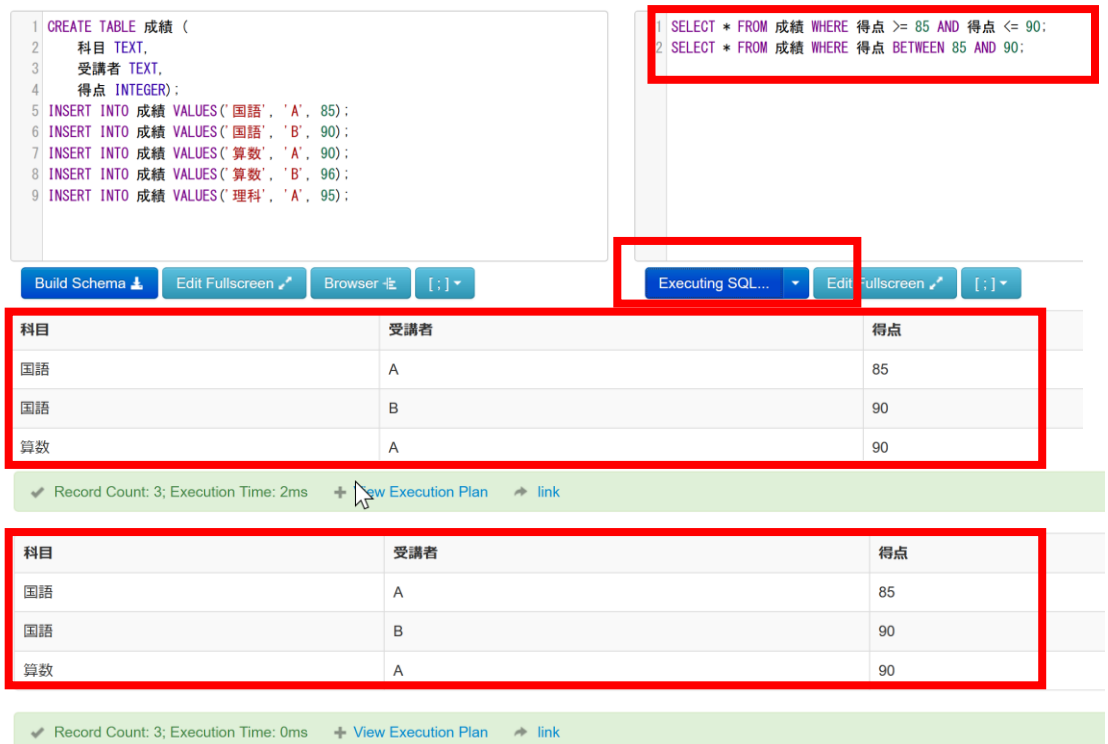
③　**上のパネル**に、**テーブル定義**と**データの追加**と**問い合わせ**を行う SQL を入れ実行。（**以前の SQL は不要なので消す**）

```
CREATE TABLE 成績 (
科目 TEXT,
受講者 TEXT,
得点 INTEGER);
INSERT INTO 成績 VALUES('国語', 'A', 85);
INSERT INTO 成績 VALUES('国語', 'B', 90);
INSERT INTO 成績 VALUES('算数', 'A', 90);
INSERT INTO 成績 VALUES('算数', 'B', 96);
INSERT INTO 成績 VALUES('理科', 'A', 95);
SELECT * FROM 成績 WHERE 得点 >= 85 AND 得点 <= 90;
SELECT * FROM 成績 WHERE 得点 BETWEEN 85 AND 90;
```

④ 「**Execute**」をクリック

SQL 文が**実行**され、結果が表示される。

⑤ 下のパネルで、**結果を確認。**（**同じ結果が２つ**）



発展演習１．特定の得点範囲の検索

目的: ANDを使用して、特定の得点範囲内の成績を検索する。練習のため、BETWEENは使わない。

指示: **得点が 90以上 100以下の成績を検索するSQL 文を書いてください。BETWEENは使わないでください。**

ヒント: WHERE でANDを使用して、得点の下限と上限を指定する。

発展演習２．特定の得点範囲の検索

目的: ANDを使用して、特定の得点範囲内の成績を検索する。今度はBETWEENを使う。

指示: **得点が 90以上 100以下の成績を検索するSQL 文を書いてください。**BETWEENとANDを**使ってください**。

ヒント: WHERE で BETWEEN と ANDを使用して、得点の下限と上限を指定する。

発展演習３．複数条件を用いた検索

目的: ANDとBETWEENを組み合わせて、特定の条件を満たす成績を検索する

指示: **国語の得点が 90以上 100以下の成績を検索するSQL文を書いてください。**BETWEENとANDを使ってください。

ヒント: WHERE で ANDを使用して、複数の条件を連結する

解答例

発展演習１：

SELECT * FROM 成績 WHERE 得点 >= 90 AND 得点 <= 100

発展演習２：

SELECT * FROM 成績 WHERE 得点 BETWEEN 90 AND 100;

発展演習３：

SELECT * FROM 成績 WHERE 科目 = '国語' AND 得点 BETWEEN 90 AND 100;

ここまでのまとめ

範囲指定（AND、BETWEEN）

- **AND**：複数の条件を連結
- **BETWEEN**：特定の範囲内の値を指定

範囲指定のクエリ例

得点が85以上90以下のデータを選択

```
SELECT * FROM 成績 WHERE 得点 >= 85 AND 得点 <= 90;
```

```
SELECT * FROM 成績 WHERE 得点 BETWEEN 85 AND 90;
```

8-2. SQL の IN

IN 演算子の基本

- **SQL の IN 演算子は，複数の値のいずれか**に**一致するか**どうかをテストする.

  例：WHERE 科目 IN ('国語', '算数');

- OR 演算子で，複数の値を並べるよりも簡潔

  例：WHERE 科目 = '国語' OR 科目 = '算数';

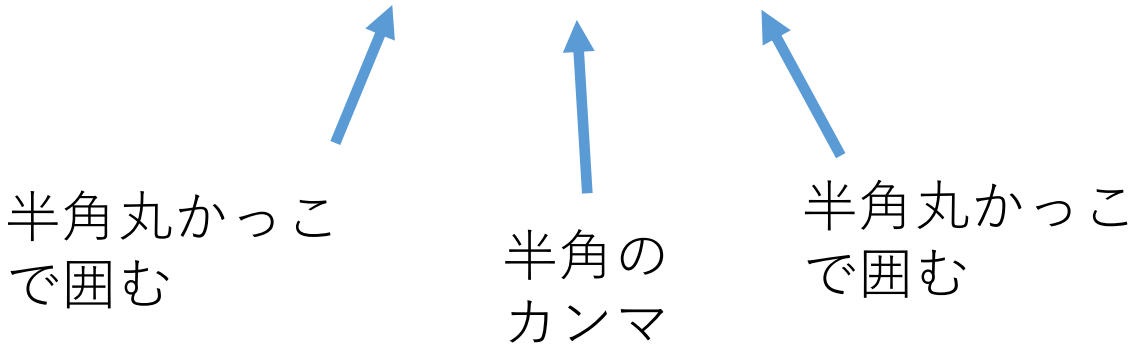
IN を用いた SQL の例　科目が国語または算数に一致

```
SELECT *
FROM 成績
WHERE 科目 IN ('国語', '算数');
```

IN 演算子の構文

- **IN演算子**で「**複数の値のいずれか**に**一致するか**」を指定する際は，**半角の丸かっこ**で**全体を囲み，値と値の間は半角のカンマで区切る**.

```
SELECT *
FROM 成績
WHERE 科目 IN ('国語', '算数');
```



演習３．SQL の IN

【トピックス】

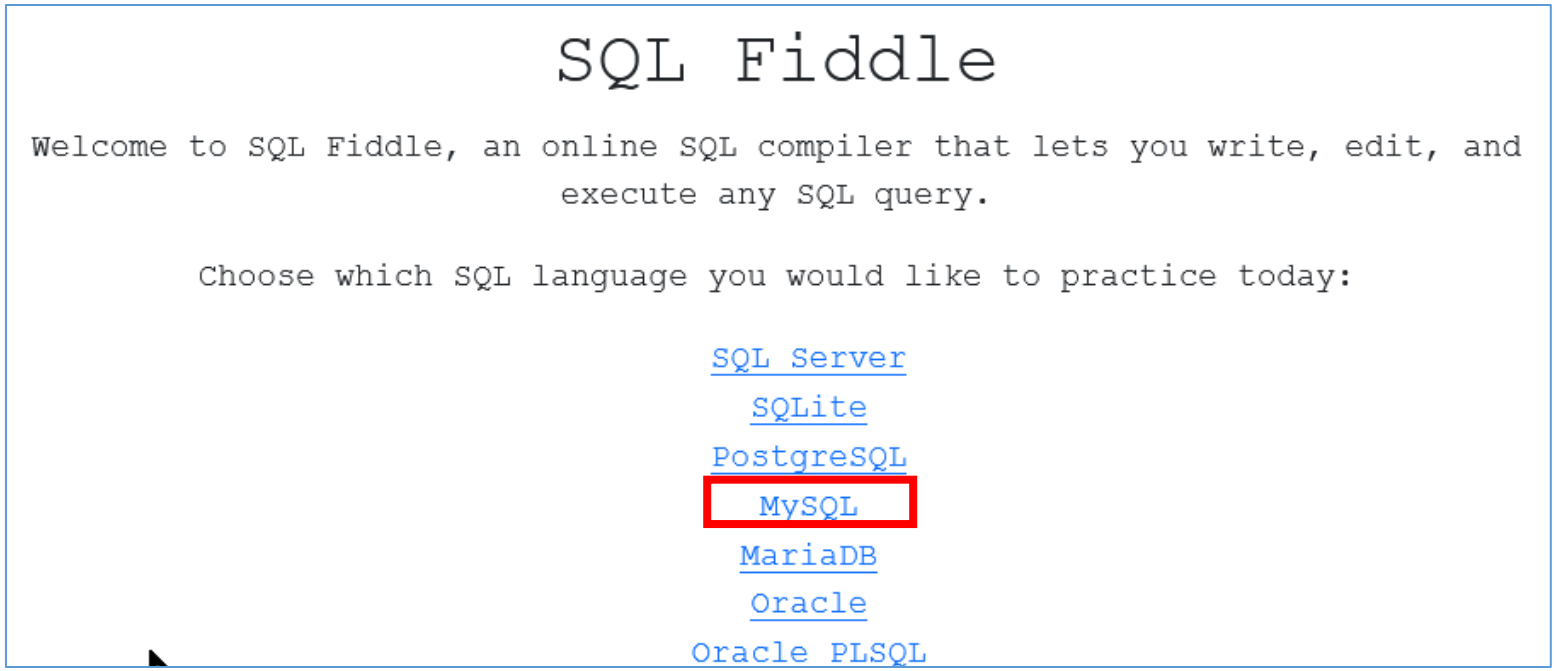
1. **複数の値のいずれかに一致するかテスト**
2. **IN**

Webブラウザを使用

① アドレスバーにSQLFiddleのURLを入力

http://sqlfiddle.com/

② 「**MySQL**」を選択



③　**上のパネル**に、**テーブル定義**と**データの追加**と**問い合わせ**を行う SQL を入れ実行。（**以前の SQL は不要なので消す**）

```
CREATE TABLE 成績 (
科目 TEXT,
受講者 TEXT,
得点 INTEGER);
INSERT INTO 成績 VALUES('国語', 'A', 85);
INSERT INTO 成績 VALUES('国語', 'B', 90);
INSERT INTO 成績 VALUES('算数', 'A', 90);
INSERT INTO 成績 VALUES('算数', 'B', 96);
INSERT INTO 成績 VALUES('理科', 'A', 95);
SELECT * FROM 成績 WHERE 科目 IN ('国語','算数');
```

④ 「**Execute**」をクリック

SQL 文が**実行**され、結果が表示される。

⑤ 下のパネルで、**結果を確認。**

発展演習４．複数の値のいずれかに一致するかを条件とする検索

目的: IN を用いて、特定の点数を得た学生のみを検索する

指示: **得点が 80 点であるか 90点である成績を検索するSQL文を書いてください。**IN を使ってください。

発展演習５．複数の値のいずれかに一致するかを条件とする検索

目的: IN を用いて、特定の点数を得た学生のみを検索する

指示: **得点が 80 点であるか、85点であるか、90点である成績を検索するSQL 文を書いてください。**IN を使ってください。

解答例

発展演習４：

SELECT * FROM 成績 WHERE 得点 IN (80, 90);

発展演習５：

SELECT * FROM 成績 WHERE 得点 IN (80, 85, 90);

8-3. 副問い合わせ

副問い合わせ

副問い合わせは，**別のSQL問い合わせ（クエリ）**内に**埋め込まれた**SQL問い合わせ（クエリ）である.

```
SELECT 受講者 FROM 成績 WHERE 得点 =
(SELECT MAX(得点) FROM 成績);
```

副問い合わせ

複数 SQL の組み合わせ

副問い合わせを使用することで，**複数のSQLを組み合わせる**ことができる.

例：**成績テーブル**から**最高得点**の**受講者**を検索する場合，**MAX関数による副問い合わせ**と**主問い合わせを組み合わせて**使用する.

成績

科目	受講者	得点
国語	A	85
国語	B	90
算数	A	90
算数	B	96
理科	A	95

```
SELECT 受講者 FROM 成績 WHERE 得点 =
(SELECT MAX(得点) FROM 成績);
```

問い合わせの種類

○ 単一行副問い合わせ

- **副問い合わせは，必ず，一つの行のみを返す.**
- **比較演算子（=, <, > など）**と共に使用。

例：SELECT * FROM 従業員 WHERE 給与 **=** (SELECT MAX(給与) FROM 従業員);

○ 複数行副問い合わせ

- **副問い合わせは，複数の行を返す可能性がありえる**
- **IN** などと共に使用。

SELECT * FROM 従業員 WHERE 部署ID **IN** (SELECT 部署ID FROM 部署 WHERE 場所 = '東京');

副問い合わせの重要性

- **副問い合わせ**は、**複雑なデータ抽出**を可能にする
- 一つの問い合わせの**条件**を**別の問い合わせから得る**ことができる
- 常に**最新のデータベース内のデータ**に基づいて**条件**を設定することが可能になる
- データの集計や比較をより柔軟に行うことができる

SQL の例①

成績

科目	受講者	得点
国語	A	85
国語	B	90
算数	A	90
算数	B	96
理科	A	95

最高得点は？

```
SELECT MAX(得点) FROM 成績;
```

SQL の例②

成績

科目	受講者	得点
国語	A	85
国語	B	90
算数	A	90
算数	B	96
理科	A	95

96点の得点の受講者は？

```
SELECT 受講者 FROM 成績 WHERE 得点 = 96;
```

複数の SQL の組み合わせ

成績

科目	受講者	得点
国語	A	85
国語	B	90
算数	A	90
算数	B	96
理科	A	95

```
SELECT MAX(得点) FROM 成績 WHERE 科目;
```

```
SELECT 受講者 FROM 成績 WHERE 得点 = 96;
```

組み合わせる。
かっこと = を使用

```
SELECT 受講者 FROM 成績 WHERE 得点 =
(SELECT MAX(得点) FROM 成績);
```

演習４．副問い合わせ

【トピックス】

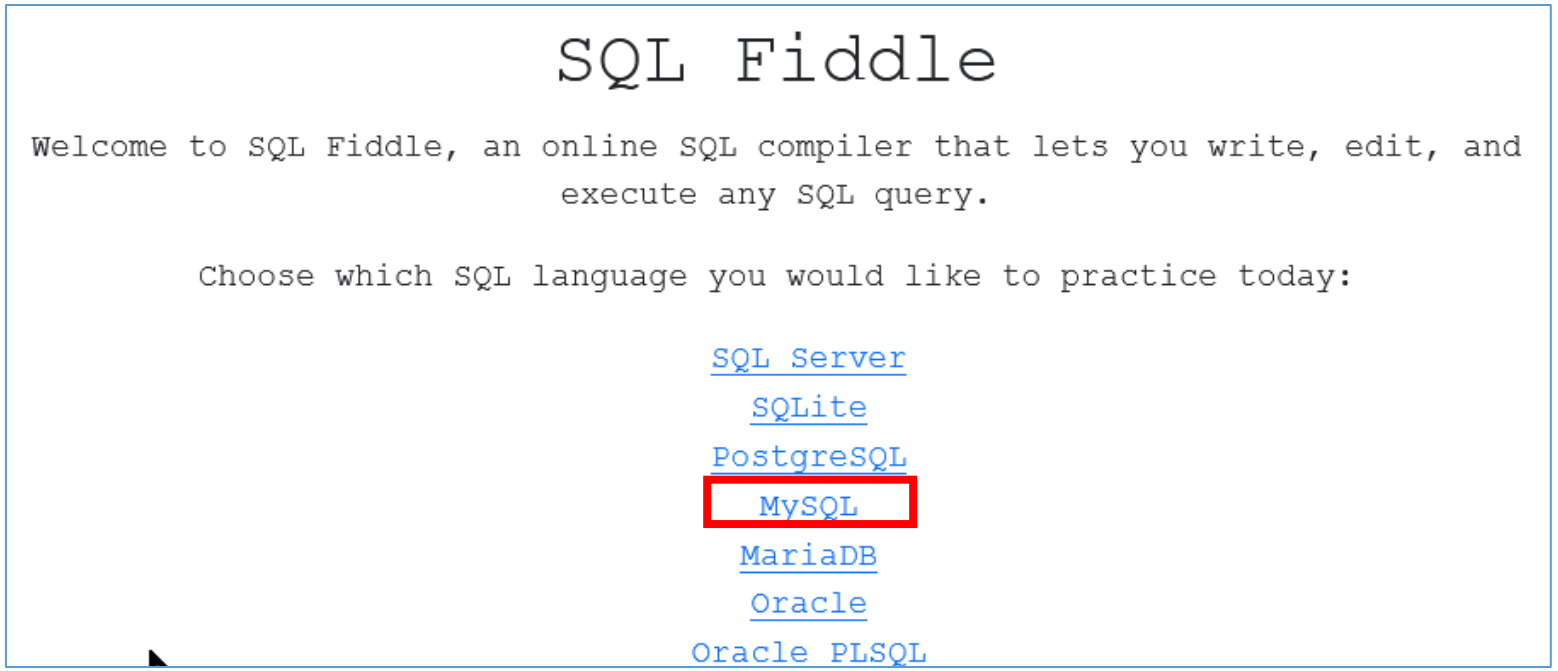
1. **副問い合わせ**

Webブラウザを使用

① アドレスバーにSQLFiddleのURLを入力

http://sqlfiddle.com/

② 「**MySQL**」を選択



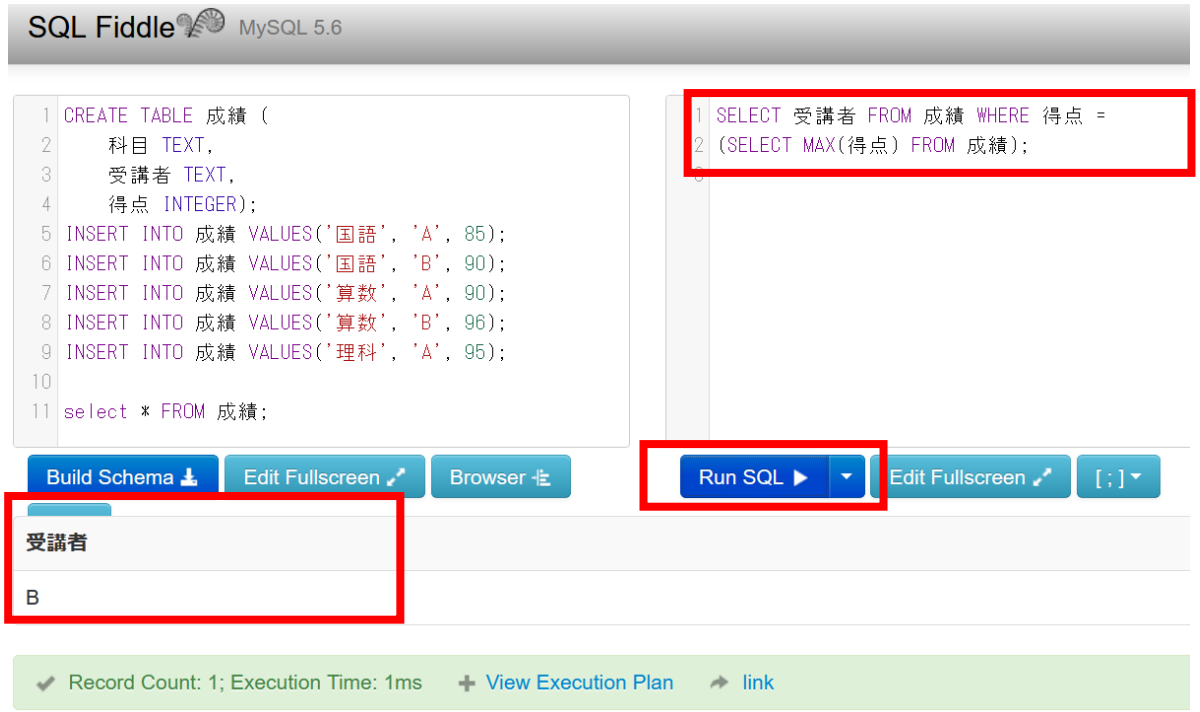
③　**上のパネル**に、**テーブル定義**と**データの追加**と**問い合わせ**を行う SQL を入れ実行。（**以前の SQL は不要なので消す**）

```
CREATE TABLE 成績 (
科目 TEXT,
受講者 TEXT,
得点 INTEGER);
INSERT INTO 成績 VALUES('国語', 'A', 85);
INSERT INTO 成績 VALUES('国語', 'B', 90);
INSERT INTO 成績 VALUES('算数', 'A', 90);
INSERT INTO 成績 VALUES('算数', 'B', 96);
INSERT INTO 成績 VALUES('理科', 'A', 95);
SELECT 受講者 FROM 成績 WHERE 得点 =
(SELECT MAX(得点) FROM 成績);
```

④ 「**Execute**」をクリック

SQL 文が**実行**され、結果が表示される。

⑤ 下のパネルで、**結果を確認。**



発展演習６．平均得点よりも高いことを条件とする検索

目的: 全科目の平均点よりも高い得点である行のみを選択

指示: **成績テーブルから、全科目の平均点よりも高い得点である行を選択してください。**

ヒント：全科目の平均得点は副問い合わせで計算。AVG を使用。「よりも高い」という条件にも注意。

次のような結果になる

科目	受講者	得点
算数	B	96
理科	A	95

発展演習 6

```
SELECT * FROM 成績 WHERE 得点 >
(SELECT AVG(得点) FROM 成績);
```

8-4. AND と OR

AND，OR 演算子の使用

- SQLでは，**AND演算子**は**両方の条件が成立する場合**を，**OR演算子**は**いずれかの条件が成立する場合**を選択する.
- ORは「どちらか一方を選ぶ」という意味ではなく，両方の条件が成立する場合も含まれる

AND

【条件A】 AND 【条件B】

条件Aと条件Bの**両方の条件が成立**

OR

【条件A】 OR 【条件B】

条件A、条件Bの**いずれの条件が成立（両方が成立する場合を含む）**

二進数は 0 または 1

右手が上がって**いない**

右手が上がって**いる**

二通り

二進数は 0 または 1

右手が上がって**いない**

FALSE

右手が上がって**いる**

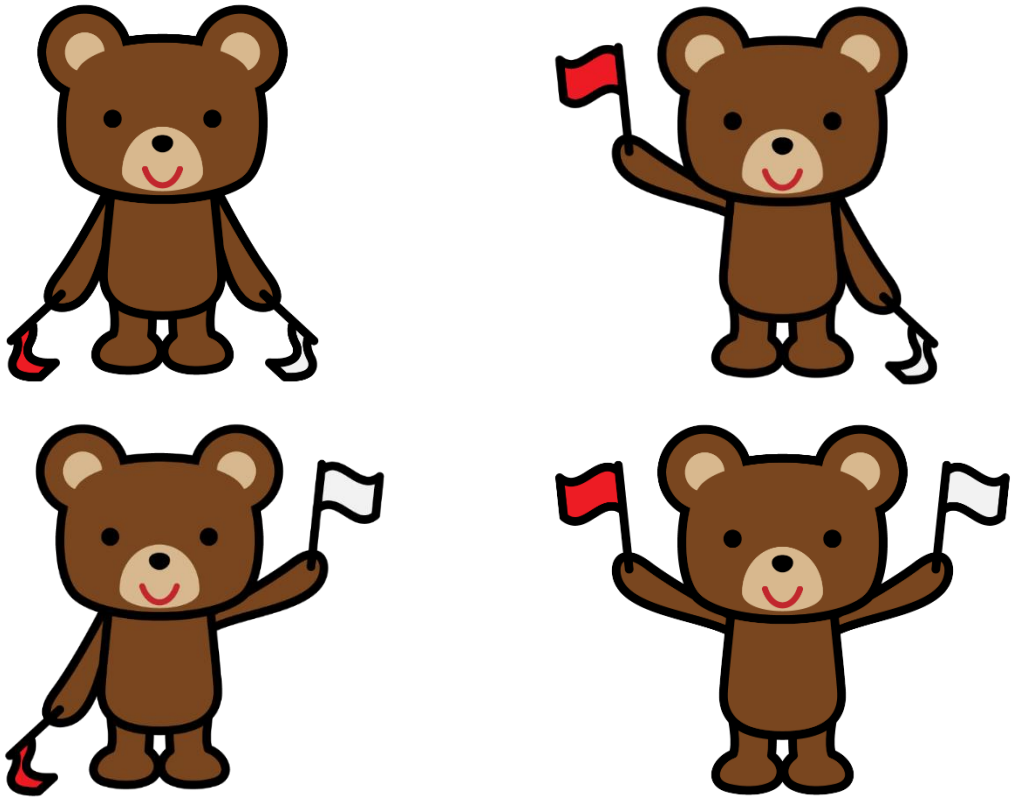
TRUE

変数が２つ

右手と左手の
両方を考えると

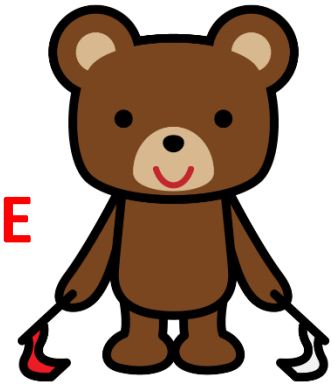
↓

４通り



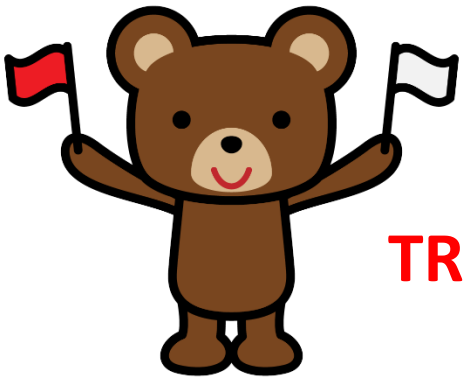
変数が２つ

FALSE と FALSE



TRUE と FALSE

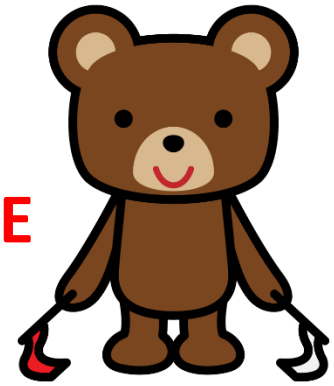
FALSE と TRUE



TRUE と TRUE

AND

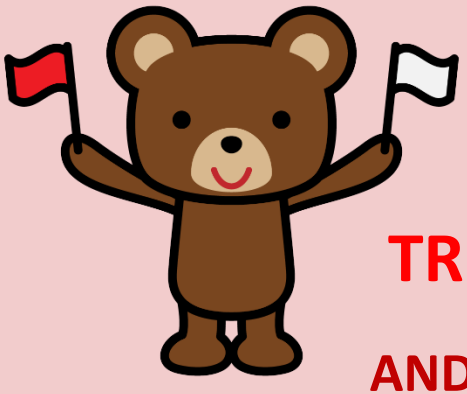
FALSE と FALSE



TRUE と FALSE

FALSE と TRUE

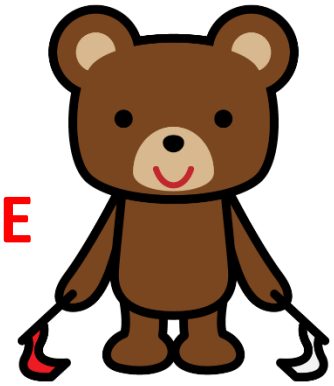
TRUE と TRUE



AND は両方とも 1

OR

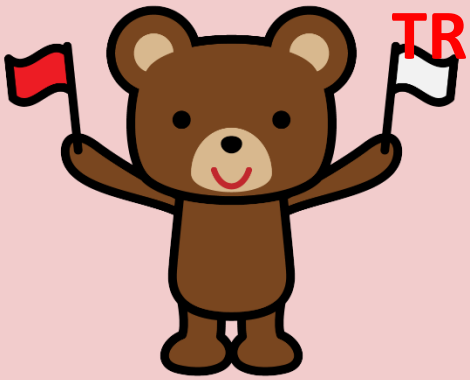
FALSE と FALSE



TRUE と FALSE

FALSE と TRUE

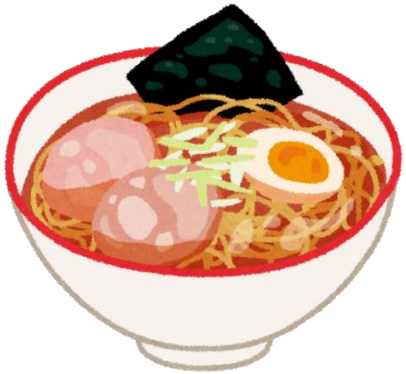
TRUE と TRUE



OR は
少なくとも
片方には
１がある

OR は「どちらかを選びなさい」という意味ではない

とんこつラーメン



醤油ラーメン

醤油ラーメン
食べない：FALSE

食べる：TRUE

とんこつラーメン
食べない：FALSE　食べる：TRUE

どちらか食べても、両方食べても OK!
（これが **OR** の鉄則）

※　「どちらかを選べ」
という意味では***ない***

8-5. 演習

演習の目的

- SQL の理解、SQL のスキルについて能力を高める
- テーブル定義、データの追加、問い合わせの実行を通じて、リレーショナルデータベースの操作に慣れる

演習の概要

テーブルの作成とデータ追加

- 従業員テーブルの定義
- 24行のデータ追加
- 属性は、従業員ID、名前、部署ID、給与

SQLサンプルの実行

- 部署IDに基づくデータの選択
- 給与の平均以上を稼ぐ従業員の特定
- 特定の部署IDを持つ従業員の選択
- 特定の条件を満たす従業員の選択

様々な問い合わせの例を提供。

発展演習問題

- 実践的な問題を解決するための発展演習問題

演習．SQL の演習

【トピックス】

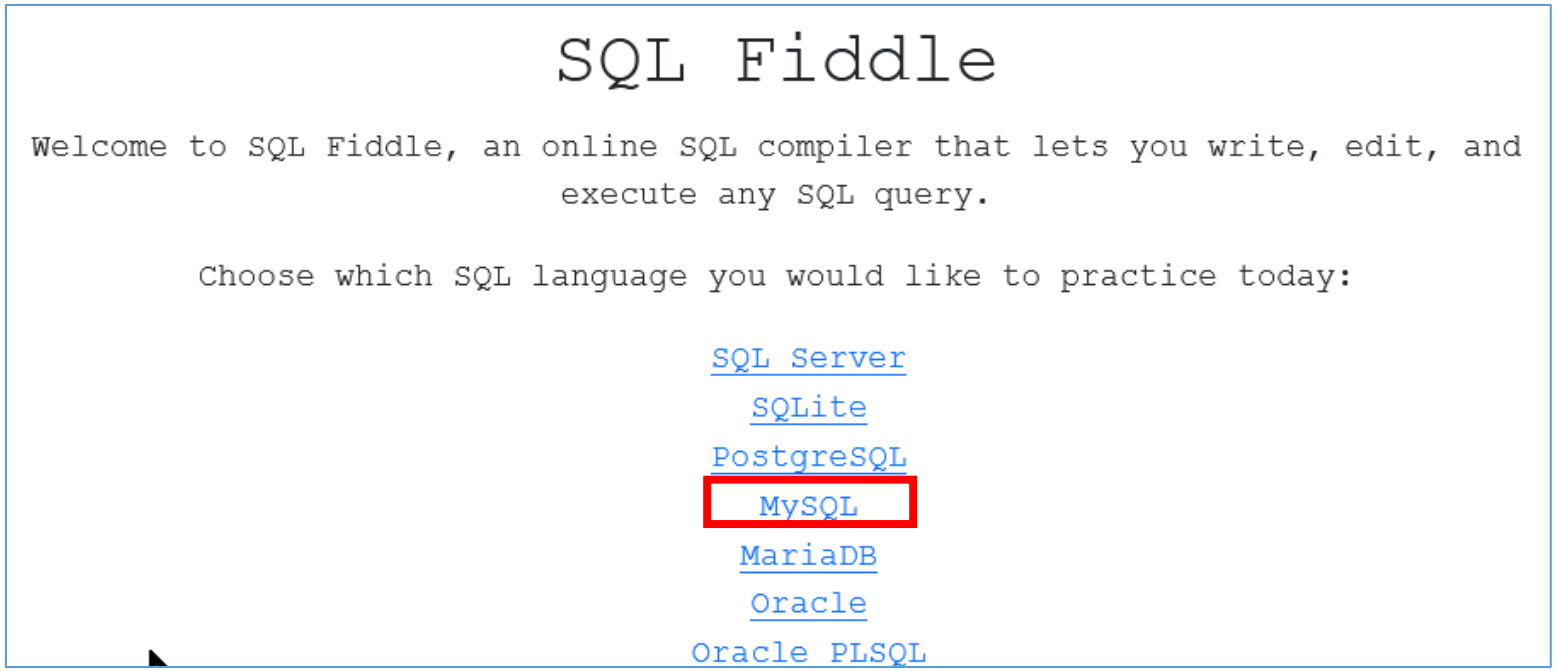
1. テーブル定義
2. データの追加
3. 問い合わせ
4. 副問い合わせ
5. IN
6. AND

Webブラウザを使用

① アドレスバーにSQLFiddleのURLを入力

http://sqlfiddle.com/

② 「**MySQL**」を選択



③　上のパネルに、テーブル定義とデータの追加と問い合わせを行う SQL を入れ実行。（以前の SQL は不要なので消す）

```
CREATE TABLE 従業員 (
    従業員ID INTEGER,
    名前 TEXT,
    部署ID INTEGER,
    給与 INTEGER
);
INSERT INTO 従業員 VALUES(5, '伊藤', 104, 280000);
INSERT INTO 従業員 VALUES(6, '渡辺', 101, 320000);
INSERT INTO 従業員 VALUES(7, '小林', 102, 270000);
INSERT INTO 従業員 VALUES(8, '加藤', 103, 290000);
INSERT INTO 従業員 VALUES(9, '吉田', 104, 310000);
INSERT INTO 従業員 VALUES(10, '中村', 101, 330000);
INSERT INTO 従業員 VALUES(11, '小川', 102, 260000);
INSERT INTO 従業員 VALUES(12, '高橋', 103, 340000);
INSERT INTO 従業員 VALUES(13, '山本', 104, 300000);
INSERT INTO 従業員 VALUES(14, '石川', 101, 350000);
INSERT INTO 従業員 VALUES(15, '中島', 102, 280000);
INSERT INTO 従業員 VALUES(16, '佐々木', 103, 360000);
INSERT INTO 従業員 VALUES(17, '山口', 104, 290000);
INSERT INTO 従業員 VALUES(18, '松本', 101, 370000);
INSERT INTO 従業員 VALUES(19, '井上', 102, 310000);
INSERT INTO 従業員 VALUES(20, '木村', 103, 280000);
INSERT INTO 従業員 VALUES(21, '林', 104, 320000);
INSERT INTO 従業員 VALUES(22, '清水', 101, 330000);
INSERT INTO 従業員 VALUES(23, '山崎', 102, 340000);
INSERT INTO 従業員 VALUES(24, '中田', 103, 300000);
SELECT 名前, 給与 FROM 従業員 WHERE 部署ID = 101;
```

部署ID 101 の従業員の名前と給与を選択:

④ 「**Execute**」をクリック

SQL 文が**実行**され、結果が表示される。

⑤ 下のパネルで、**結果を確認。**

名前	給与
渡辺	320000
中村	330000
石川	350000
松本	370000
清水	330000

⑥ 上のパネルに、問い合わせ（クエリ）を行う SQL を追加する。

```
SELECT * FROM 従業員 WHERE 給与 >= (SELECT AVG(給与) FROM 従業員);
```

給与が平均以上の従業員を選択

⑦ 「**Execute**」をクリック

SQL 文が**実行**され、結果が表示される。

⑧ 下のパネルで、**結果を確認。**

従業員ID	名前	部署ID	給与
6	渡辺	101	320000
10	中村	101	330000
12	高橋	103	340000
14	石川	101	350000
16	佐々木	103	360000
18	松本	101	370000
21	林	104	320000
22	清水	101	330000
23	山崎	102	340000

⑨ **上のパネルに、問い合わせ（クエリ）を行う SQL を追加する。**

```
SELECT * FROM 従業員 WHERE 部署ID IN (101, 102);
```

部署ID 101 または 102 に所属する従業員を選択

⑩ 「**Execute**」をクリック

SQL 文が**実行**され、結果が表示される。

⑪ 下のパネルで、**結果を確認。**

従業員ID	名前	部署ID	給与
6	渡辺	101	320000
7	小林	102	270000
10	中村	101	330000
11	小川	102	260000
14	石川	101	350000
15	中島	102	280000
18	松本	101	370000
19	井上	102	310000
22	清水	101	330000
23	山崎	102	340000

⑫ **上のパネルに、問い合わせ（クエリ）を行う SQL を追加する。**

```
SELECT * FROM 従業員 WHERE 給与 >= 300000 AND 部署ID = 102;
```

給与が300000以上かつ部署IDが102の従業員を選択

⑬ 「**Execute**」をクリック

SQL 文が**実行**され、結果が表示される。

⑭ 下のパネルで、**結果を確認。**

従業員ID	名前	部署ID	給与
19	井上	102	310000
23	山崎	102	340000

発展演習７．最高給与を受け取る従業員の名前を特定

目的：最高給与を受け取る従業員の名前を副問い合わせを用いて特定

従業員テーブルから最高給与を受け取る従業員の名前を選択するSQL文を書いてください。

ヒント: 最高給与はMAXを使って副問い合わせで求める

発展演習８．特定の部署に所属し、特定の給与範囲にある従業員を特定

目的：**部署ID 102 あるいは 103 に所属し、給与が300000以上の従業員**を特定

従業員テーブルから、「目的」で指定された条件を満たす従業員を選択するSQL文を書いてください

ヒント: ANDを用いて、部署IDと給与の条件を組み合わせる。

発展演習 7

```
SELECT 名前 FROM 従業員 WHERE 給与 = (SELECT MAX(給与) FROM 従業員);
```

発展演習 8

```
SELECT 名前, 給与 FROM 従業員 WHERE 部署ID IN (102, 103) AND 給与 >= 300000;
```

基本的な SQL 文のまとめ

SELECT *	**テーブルの全表示**
SELECT 属性名あるいは属性名リスト	**特定属性の表示**
DISTINCT	**重複除去**
WHERE	**条件指定**
LIKE	**パターンマッチ**
GROUP BY	**グループ化**
COUNT	**行数のカウント**
SUM, AVG, MAX, MIN	**合計，平均，最大，最小**
BETWEEN	**範囲指定**
IN	**値の一致**
AND	**両方の条件が成立**
OR	**いずれかの条件が成立**

まとめ

- **副問い合わせ**
  別のSQL問い合わせ内に埋め込まれた問い合わせ（クエリ）．複雑な条件を実現する.
- **IN演算子**
  値の一致についての判定を行う演算子.
- **AND/OR演算子**
  ANDは両条件の成立，ORはいずれかの条件成立を選択する基本的な論理演算子
- **BETWEEN演算子**
  値の範囲指定を簡潔に記述できる演算子.