

# 12. 中間まとめ：効率的なデータ管理と分析の実践<

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



# アウトライン

1. イントロダクション
2. NULL
3. テーブル定義、問い合わせ
4. データ操作

# 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

## SQL Fiddle

Welcome to SQL Fiddle, an online SQL compiler that lets you write, edit, and execute any SQL query.

Choose which SQL language you would like to practice today:

[SQL Server](#)

[SQLite](#)

[PostgreSQL](#)

[MySQL](#)

[MariaDB](#)

[Oracle](#)


[Oracle PLSQL](#)

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



```
1 -- INIT database
2 CREATE TABLE Product (
3   ProductID INT AUTO_INCREMENT KEY,
4   Name VARCHAR(100),
5   Description VARCHAR(255)
6 );
7
8 INSERT INTO Product(Name, Description) VALUES ('Entity Framework Extensions', 'Use
9 INSERT INTO Product(Name, Description) VALUES ('Dapper Plus', 'Use <a href="https
10 INSERT INTO Product(Name, Description) VALUES ('C# Eval Expression', 'Use <a href
11
12 -- QUERY database
13 SELECT * FROM Product;
14 SELECT * FROM Product WHERE ProductID = 1;
15
```

実行ボタン


Execute

< Share

MySQL

結果ウィンドウ

Results



| ProductID | Name                        | Description   |
|-----------|-----------------------------|---|
| 1         | Entity Framework Extensions | Use <a href="#">Entity Framework Extensions</a> to extend your DbContext with high-performance bulk operations. |
| 2         | Dapper Plus                 | Use <a href="#">Dapper Plus</a> to extend your IDbConnection with high-performance bulk operations.             |

# 12-1. 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に。

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    |

# 商品テーブルと購入テーブル

## 商品

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

## 購入

| 購入者 | 商品番号 |
|-----|------|
| X   | 1    |
| X   | 3    |
| Y   | 2    |

関連

Xさんは、**1**の**みかん**と、  
**3**の**メロン**を買った  
Yさんは、**2**の**りんご**を買った

購入テーブルの情報      商品テーブルの情報



# リレーショナルデータベースの重要性

1. **データの整合性:** リレーショナルデータベースは、**データの整合性を保持するための機能**を有する。これにより、誤ったデータや矛盾したデータが保存されるのを防ぐことができる。
2. **柔軟な問い合わせ（クエリ）能力:** リレーショナルデータベースの**SQL**（Structured Query Language）の使用により、**複雑な検索やデータの抽出**が可能になる。
3. **トランザクションの機能:** 一連の操作全体を一つの単位として取り扱うことができる機能。これにより、**データの一貫性と信頼性が向上**する。
4. **セキュリティ:** **アクセス権限の設定**などにより、セキュリティを確保。

データの安全な保管、効率的なデータ検索・操作、ビジネスや研究の意思決定をサポート。

# 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 問い合わせの全体像①

- **select**: データの検索・加工や射影

例 : **SELECT \* FROM employees**

- **from**: 問い合わせ対象テーブルの指定

例 : **FROM employees**

- **where**: 条件に一致する行を選択

例 : **WHERE age > 30**

- **join, on**: 結合、結合条件

例 : **JOIN B ON A.b\_id = B.id**

- **insert into**: 新しい行の追加（挿入）

- **update, set**: 条件に一致する行を更新

- **delete from**: 条件に一致する行を削除

## SQL 問い合わせの全体像②

- **distinct**: 重複行の除去

例 : **SELECT DISTINCT age FROM employees**

- **count**: 行数のカウント

例 : **SELECT COUNT(\*) FROM employees**

- **avg, max, min, sum**: 平均、最大、最小、合計の計算

例 : **AVG(salary), MAX(salary)**

- **group by**: 属性でグループ化

例 : **GROUP BY department\_id**

- **order by**: 並べ替え (ソート)

例 : **ORDER BY age**

- 副問い合わせ: SQL文の中に別のSQL文を埋め込む。

例 : **WHERE salary > (SELECT AVG(salary) FROM employees)**

# SQL による結合の基本

## 商品

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

## 購入

| 購入者 | 商品番号 |
|-----|------|
| X   | 1    |
| X   | 3    |
| Y   | 2    |

関連



## 結合のためのSQL

**SELECT \* FROM** 商品

**INNER JOIN** 購入

**ON** 商品.ID = 購入.商品番号;

} **結合条件**

| ID | 商品名 | 単価  | 購入者 | 商品番号 |
|----|-----|-----|-----|------|
| 1  | みかん | 50  | X   | 1    |
| 3  | メロン | 500 | X   | 3    |
| 2  | りんご | 100 | Y   | 2    |

**結合条件**に基づいて、  
両テーブルのデータが  
結合される。

# 集約の方法

**AVG, MAX, MIN, SUM** : 平均値, 最大値, 最小値, 合計値を算出

**COUNT** : 行数を計算

記録テーブル

| 名前   | 得点 | 居室 |
|------|----|----|
| 徳川家康 | 85 | 1階 |
| 源義経  | 78 | 2階 |
| 西郷隆盛 | 90 | 3階 |
| 豊臣秀吉 | 82 | 1階 |
| 織田信長 | 75 | 2階 |

**SELECT AVG(得点) FROM** 記録;

結果: 82

**SELECT MAX(得点) FROM** 記録;

結果: 90

**SELECT MIN(得点) FROM** 記録;

結果: 75

**SELECT SUM(得点) FROM** 記録;

結果: 410

**SELECT COUNT(\*) FROM** 記録;

結果: 5

# グループ化

グループ化は、同じ属性値を共有するデータを集めるプロセス。

例：科目の「国語」、「算数」、「理科」でグループ化

| 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 |



| 科目 | 受講者 | 得点 |
|----|-----|----|
| 国語 | A   | 85 |
| 国語 | B   | 90 |

| 科目 | 受講者 | 得点 |
|----|-----|----|
| 理科 | A   | 95 |

| 科目 | 受講者 | 得点 |
|----|-----|----|
| 算数 | A   | 90 |
| 算数 | B   | 96 |

例：受講者の「A」、「B」でグループ化

| 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 |



| 科目 | 受講者 | 得点 |
|----|-----|----|
| 国語 | A   | 85 |
| 算数 | A   | 90 |
| 理科 | A   | 95 |

| 科目 | 受講者 | 得点 |
|----|-----|----|
| 国語 | B   | 90 |
| 算数 | B   | 96 |

それぞれの値ごとにグループに分けることで、データの分析が容易になる

# グループ化と集約

- **グループ化**は、**同じ属性値を共有するデータを集めるプロセス**。

例：「科目」や「受講者」ごとにデータを分類できる

- こうして形成された**グループを集約（行数、平均値、合計値）**し、元のデータの概要を把握することが可能になる。

| 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 |



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



概要

得点の平均

|    |      |
|----|------|
| 国語 | 87.5 |
| 算数 | 93   |
| 理科 | 95   |

グループ化

集約



# GROUP BY の役割と書き方

- **SQL 問い合わせ「SELECT ...」の中で、GROUP BY を使用してデータをグループ化する**
- **1つ以上の属性を GROUP BY に指定してグループ化の基準とする。**

すべての科目ごとに、受講者の数を計算

**SELECT 科目, COUNT(\*) FROM 成績 GROUP BY 科目;**

| 科目 | COUNT(*) |
|----|----------|
| 国語 | 2        |
| 理科 | 1        |
| 算数 | 2        |

# テーブルと属性

- **テーブル**：データを**表形式で保存**する構造
- **属性（列）**：データの種類に対応（例：ID, 商品名, 単価）
- **行**：属性（列）に基づいた具体的なデータの集まり

**テーブル**

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

「ID」と「商品名」と  
「単価」の**属性**

# 属性のデータ型

## 【主なデータ型】

- **整数 (INTEGER)** : ID, 単価など
- **テキスト (TEXT)** : 商品名など
- **日付／時刻 (DATETIME)**
- **Yes／No (BIT, BOOL)** : ブール値

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



整数で  
オートナンバー

テキスト

整数

※**オートナンバー**  
は、自動で 1,2,3  
のように**通し番号**  
が付くもの

# 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       |
| 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**テキスト**

# SQL によるテーブル定義

## テーブル名：商品

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

数値・            テキスト            半角の数値  
オートナンバー

```
create table 商品 (  
    ID autoincrement,  
    商品名 text,  
    単価 integer  
);
```

区切りの半角カンマ

## SQL文

※ Access では

「integer autoincrement」と  
書かずに「autoincrement」

# 主キー

- **主キー**は、**テーブルの各行を識別するためのキー**

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

**ID属性は主キーである**

# 主キー制約と PRIMARY KEY

**PRIMARY KEY** はテーブル定義時に使用し, 「**主キー制約**」を示す

```
CREATE TABLE テーブル名 (  
  列名1 データ型 PRIMARY KEY,  
  列名2 データ型,  
  ...  
);
```

SQL の書き方

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

```
CREATE TABLE 商品 (  
  ID INTEGER PRIMARY KEY,  
  商品名 TEXT,  
  単価 INTEGER);
```

主キー

# 外部キー

外部キーは、他のテーブルの主キーを参照するキー

購入テーブル

外部キー

| ID | 購入者 | 商品ID | 数量 |
|----|-----|------|----|
| 1  | X   | 1    | 10 |
| 2  | Y   | 2    | 5  |



購入テーブルの外部キー「商品ID」は、購入テーブルの主キー「ID」を参照

商品テーブル

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

主キー



# SQL によるテーブル定義

- テーブル名 : **購入**
- 属性名 : **ID、購入者、商品ID、数量**
- 属性のデータ型 : **数値、テキスト、数値、数値**
- データの整合性を保つための制約 : **主キー制約、参照整合性制約**

```
CREATE TABLE 購入 (  
  ID INTEGER PRIMARY KEY,  
  購入者 TEXT,  
  商品ID INTEGER,  
  数量 INTEGER,  
  FOREIGN KEY (商品ID) REFERENCES 商品(ID));
```

# FOREIGN KEY ... REFERENCES

**PRIMARY KEY ... REFERENCES** はテーブル定義時に使用し、あるテーブルの**外部キー**が別のテーブルの**主キー**を**参照**する「**参照整合性制約**」を示す

```
CREATE TABLE 購入 (  
  ID INTEGER PRIMARY KEY,  
  購入者 TEXT,  
  商品ID INTEGER,  
  数量 INTEGER,  
  FOREIGN KEY (商品ID) REFERENCES 商品(ID));
```

き方

外部キー

| ID | 購入者 | 商品ID | 数量 |
|----|-----|------|----|
| 1  | X   | 1    | 10 |
| 2  | Y   | 2    | 5  |

主キー

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

主キー



# データベース操作の種類

## **INSERT（追加）**

- テーブルに新しい行を追加する操作

## **SELECT（問い合わせ）**

- 必要なデータを検索・加工する操作

## **DELETE（削除）**

- テーブルから条件に合致する行をすべて削除する操作

## **UPDATE（更新）**

- 条件に合致する部分について、値を変更する操作

# INSERT の基本 (データの追加)

## INSERT の基本形式

**INSERT INTO テーブル名 VALUES (値1, 値2, ...);**

| 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 |

テーブル T

```
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);
```



## 演習 1. テーブル定義とデータの追加、主キー制約

### 【トピックス】

1. SQL によるテーブル定義
2. 主キー制約 PRIMARY KEY
3. SQL によるデータの追加
4. 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 を入れ実行。（以前の SQL は不要なので消す）

```
CREATE TABLE メニュー (  
ID INTEGER PRIMARY KEY,  
商品名 TEXT,  
単価 INTEGER);  
INSERT INTO メニュー VALUES (1, 'かき氷', 400);  
INSERT INTO メニュー VALUES (2, 'カレーライス', 400);  
INSERT INTO メニュー VALUES (3, 'サイダー', 200);  
SELECT * FROM メニュー;
```

④ 「**Execute**」をクリック

SQL 文が**実行**され、結果が表示される。

⑤ 下側のウィンドウで、**結果を確認**。

|                     |        |     |  |
|---------------------|--------|-----|--|
| +-----+-----+-----+ |        |     |  |
| ID                  | 商品名    | 単価  |  |
| +-----+-----+-----+ |        |     |  |
| 1                   | かき氷    | 400 |  |
| 2                   | カレーライス | 400 |  |
| 3                   | サイダー   | 200 |  |



## 12-2. NULL

# データベースの NULL

- データ化できないものは、**NULL**

**未定**      (将来決まるかも)

**未知**      (将来分かるかも)

**存在しない**      (そもそも存在しない)

自由席の切符に座席番号は存在しない

試合をしていないとき勝率は存在しない

**NULL** は「**ヌル**」あるいは「**ナル**」と読む

## NULL を使う例（1）

- カレーライスの値段が、まだ決まっていない（未定）

| 商品     | 価格   |
|--------|------|
| かき氷    | 400  |
| カレーライス | NULL |
| サイダー   | 200  |

値段は、必ず決まるはずだが、  
まだ決まっていない

## NULL を使う例（2）

- 試合をしていないので，勝率は**存在しない**

| チーム名 | 試合数 | 勝ち数 | 勝率   |
|------|-----|-----|------|
| A    | 10  | 6   | 0.6  |
| B    | 4   | 3   | 0.75 |
| C    | 0   | 0   | NULL |

試合をしていないので，  
勝率は存在しない

# NULL、IS NULL、IS NOT NULL

- IS NULL

NULLであることを条件

**select \* from T where 価格 IS NULL;**

- IS NOT NULL

NULLでないことを条件

**select \* from T where 価格 IS NOT NULL;**

# NULLと「0」は違う

| 商品     | 価格   |
|--------|------|
| かき氷    | 400  |
| カレーライス | NULL |
| サイダー   | 200  |



| 商品     | 価格  |
|--------|-----|
| かき氷    | 400 |
| カレーライス | 0   |
| サイダー   | 200 |

値段は、まだ決まっていない  
あるいは、知らない

カレーライスは無料

NULL を使い、未定，未知，非存在であることを  
を正しく記録

## 演習 2

| 商品     | 価格   |
|--------|------|
| かき氷    | 400  |
| カレーライス | NULL |
| サイダー   | 200  |

```
SELECT * FROM メニュー  
WHERE 単価 IS NULL;
```

| 商品     | 価格   |
|--------|------|
| カレーライス | NULL |

```
SELECT * FROM メニュー  
WHERE 単価 >= 0;
```

| 商品   | 価格  |
|------|-----|
| かき氷  | 400 |
| サイダー | 200 |



## 演習 2. NULL

### 【トピックス】

1. NULL
2. IS NULL

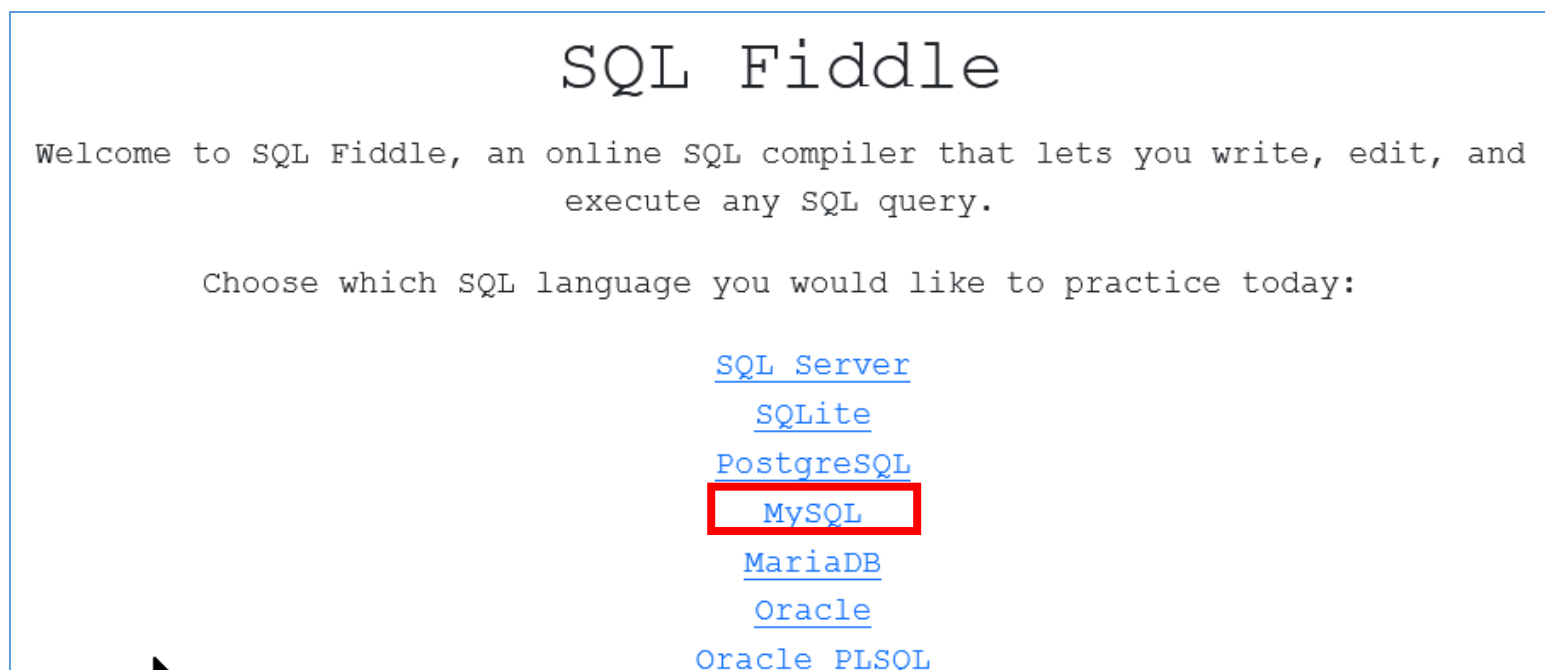


Webブラウザを使用

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 メニュー (  
ID INTEGER PRIMARY KEY,  
商品名 TEXT,  
単価 INTEGER);  
INSERT INTO メニュー VALUES (1, 'かき氷', 400);  
INSERT INTO メニュー VALUES (2, 'カレーライス', NULL);  
INSERT INTO メニュー VALUES (3, 'サイダー', 200);  
SELECT * FROM メニュー;
```

④ 「**Execute**」をクリック

SQL 文が**実行**され、結果が表示される。

⑤ 下側のウィンドウで、**結果を確認**。

```
+-----+-----+-----+-----+-----+-----+-----+
| ID | 商品名 | 単価 |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | かき氷 | 400 |
| 2 | カレーライス | NULL |
| 3 | サイダー | 200 |
```

⑥ 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 メニュー (  
ID INTEGER PRIMARY KEY,  
商品名 TEXT,  
単価 INTEGER);  
INSERT INTO メニュー VALUES (1, 'かき氷', 400);  
INSERT INTO メニュー VALUES (2, 'カレーライス', NULL);  
INSERT INTO メニュー VALUES (3, 'サイダー', 200);  
SELECT * FROM メニュー WHERE 単価 IS NULL;
```

⑦ 「Execute」をクリック

SQL 文が**実行**され、結果が表示される。

⑧ 下側のウィンドウで、**結果を確認。**

| ID | 商品名    | 単価   |
|----|--------|------|
| 2  | カレーライス | NULL |

⑨ 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 メニュー (  
ID INTEGER PRIMARY KEY,  
商品名 TEXT,  
単価 INTEGER);  
INSERT INTO メニュー VALUES (1, 'かき氷', 400);  
INSERT INTO メニュー VALUES (2, 'カレーライス', NULL);  
INSERT INTO メニュー VALUES (3, 'サイダー', 200);  
SELECT * FROM メニュー WHERE 単価 >= 0;
```

⑩ 「Execute」をクリック

SQL 文が**実行**され、結果が表示される。

⑪ 下側のウィンドウで、**結果を確認。**

| ID | 商品名  | 単価  |
|----|------|-----|
| 1  | かき氷  | 400 |
| 3  | サイダー | 200 |

## 12-3. テーブル定義、問い合わせ



# MySQL の AUTO\_INCREMENT

## MySQL の AUTO\_INCREMENT : 自動の通し番号

(MySQL の固有機能)

- **データ追加**のたびに 1, 2, 3, . . . の**通し番号**が自動で設定される
- **この機能を有効にするために、INSERT でのデータ追加では NULL を使用**

```
INSERT INTO Products VALUES (NULL, '商品A', 100);
```

```
INSERT INTO Products VALUES (NULL, '商品B', 200);
```

```
INSERT INTO Products VALUES (NULL, '商品C', 150);
```

|   |     |     |
|---|-----|-----|
| 1 | 商品A | 100 |
| 2 | 商品B | 200 |
| 3 | 商品C | 150 |

自動の通し番号



## 演習 3. テーブル定義、データの追加、問い合わせ

### 【トピックス】

1. SQL によるテーブル定義
2. 主キー制約 PRIMARY KEY
3. SQL によるデータの追加
4. 問い合わせ（クエリ）による確認
5. MySQL の AUTO\_INCREMENT と NULL

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



The screenshot shows the SQL Fiddle website. At the top, it says "SQL Fiddle". Below that, a welcome message reads: "Welcome to SQL Fiddle, an online SQL compiler that lets you write, edit, and execute any SQL query." Underneath, it asks the user to "Choose which SQL language you would like to practice today:". A list of database engines is provided as links: "SQL Server", "SQLite", "PostgreSQL", "MySQL", "MariaDB", "Oracle", and "Oracle PLSQL". The "MySQL" link is highlighted with a red rectangular box.

SQL Fiddle

Welcome to SQL Fiddle, an online SQL compiler that lets you write, edit, and execute any SQL query.

Choose which SQL language you would like to practice today:

- [SQL Server](#)
- [SQLite](#)
- [PostgreSQL](#)
- [MySQL](#)
- [MariaDB](#)
- [Oracle](#)
- [Oracle PLSQL](#)

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 商品 (  
    id INTEGER AUTO_INCREMENT PRIMARY KEY,  
    商品名 TEXT,  
    単価 INTEGER  
);  
  
CREATE TABLE 申し込み (  
    id INTEGER AUTO_INCREMENT PRIMARY KEY,  
    日時 DATETIME,  
    氏名 TEXT,  
    商品番号 INTEGER,  
    個数 INTEGER,  
    FOREIGN KEY (商品番号) REFERENCES 商品(id)  
);  
  
INSERT INTO 商品 VALUES (NULL, '商品A', 100);  
INSERT INTO 商品 VALUES (NULL, '商品B', 200);  
INSERT INTO 商品 VALUES (NULL, '商品C', 150);  
INSERT INTO 申し込み VALUES (NULL, NOW(), 'X', 1, 1);  
INSERT INTO 申し込み VALUES (NULL, NOW(), 'X', 2, 10);  
INSERT INTO 申し込み VALUES (NULL, NOW(), 'Y', 2, 5);  
INSERT INTO 申し込み VALUES (NULL, NOW(), 'X', 1, 1);  
SELECT * FROM 商品;
```

④ 「**Execute**」をクリック

SQL 文が**実行**され、結果が表示される。

⑤ 下側のウィンドウで、**結果を確認**。

|                    |     |     |  |
|--------------------|-----|-----|--|
| +-----+-----+----- |     |     |  |
| id                 | 商品名 | 単価  |  |
| +-----+-----+----- |     |     |  |
| 1                  | 商品A | 100 |  |
| 2                  | 商品B | 200 |  |
| 3                  | 商品C | 150 |  |
| +-----+-----+----- |     |     |  |

⑥ 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 商品 (  
    id INTEGER AUTO_INCREMENT PRIMARY KEY,  
    商品名 TEXT,  
    単価 INTEGER  
);  
  
CREATE TABLE 申し込み (  
    id INTEGER AUTO_INCREMENT PRIMARY KEY,  
    日時 DATETIME,  
    氏名 TEXT,  
    商品番号 INTEGER,  
    個数 INTEGER,  
    FOREIGN KEY (商品番号) REFERENCES 商品(id)  
);  
  
INSERT INTO 商品 VALUES (NULL, '商品A', 100);  
INSERT INTO 商品 VALUES (NULL, '商品B', 200);  
INSERT INTO 商品 VALUES (NULL, '商品C', 150);  
INSERT INTO 申し込み VALUES (NULL, NOW(), 'X', 1, 1);  
INSERT INTO 申し込み VALUES (NULL, NOW(), 'X', 2, 10);  
INSERT INTO 申し込み VALUES (NULL, NOW(), 'Y', 2, 5);  
INSERT INTO 申し込み VALUES (NULL, NOW(), 'X', 1, 1);  
SELECT * FROM 申し込み;
```

⑦ 「**Execute**」をクリック

SQL 文が**実行**され、結果が表示される。

⑧ 下側のウィンドウで、**結果を確認**。

---

| id | 日時                  | 氏名 | 商品番号 | 個数 |
|----|---------------------|----|------|----|
| 1  | 2024-12-05 10:29:55 | X  | 1    | 1  |
| 2  | 2024-12-05 10:29:55 | X  | 2    | 10 |
| 3  | 2024-12-05 10:29:55 | Y  | 2    | 5  |
| 4  | 2024-12-05 10:29:55 | X  | 1    | 1  |

⑨ 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 商品 (  
    id INTEGER AUTO_INCREMENT PRIMARY KEY,  
    商品名 TEXT,  
    単価 INTEGER  
);  
  
CREATE TABLE 申し込み (  
    id INTEGER AUTO_INCREMENT PRIMARY KEY,  
    日時 DATETIME,  
    氏名 TEXT,  
    商品番号 INTEGER,  
    個数 INTEGER,  
    FOREIGN KEY (商品番号) REFERENCES 商品(id)  
);  
  
INSERT INTO 商品 VALUES (NULL, '商品A', 100);  
INSERT INTO 商品 VALUES (NULL, '商品B', 200);  
INSERT INTO 商品 VALUES (NULL, '商品C', 150);  
INSERT INTO 申し込み VALUES (NULL, NOW(), 'X', 1, 1);  
INSERT INTO 申し込み VALUES (NULL, NOW(), 'X', 2, 10);  
INSERT INTO 申し込み VALUES (NULL, NOW(), 'Y', 2, 5);  
INSERT INTO 申し込み VALUES (NULL, NOW(), 'X', 1, 1);  
SELECT * FROM 申し込み  
JOIN 商品 ON 申し込み.商品番号 = 商品.id;
```



⑩ 「**Execute**」をクリック

SQL 文が**実行**され、結果が表示される。

⑪ 下側のウィンドウで、**結果を確認**。

| id | 日時                  | 氏名 | 商品番号 | 個数 | id | 商品名 | 単価  |
|----|---------------------|----|------|----|----|-----|-----|
| 1  | 2024-12-05 10:31:23 | X  | 1    | 1  | 1  | 商品A | 100 |
| 4  | 2024-12-05 10:31:23 | X  | 1    | 1  | 1  | 商品A | 100 |
| 2  | 2024-12-05 10:31:23 | X  | 2    | 10 | 2  | 商品B | 200 |
| 3  | 2024-12-05 10:31:23 | Y  | 2    | 5  | 2  | 商品B | 200 |

⑫ 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 商品 (  
    id INTEGER AUTO_INCREMENT PRIMARY KEY,  
    商品名 TEXT,  
    単価 INTEGER  
);  
  
CREATE TABLE 申し込み (  
    id INTEGER AUTO_INCREMENT PRIMARY KEY,  
    日時 DATETIME,  
    氏名 TEXT,  
    商品番号 INTEGER,  
    個数 INTEGER,  
    FOREIGN KEY (商品番号) REFERENCES 商品(id)  
);  
  
INSERT INTO 商品 VALUES (NULL, '商品A', 100);  
INSERT INTO 商品 VALUES (NULL, '商品B', 200);  
INSERT INTO 商品 VALUES (NULL, '商品C', 150);  
INSERT INTO 申し込み VALUES (NULL, NOW(), 'X', 1, 1);  
INSERT INTO 申し込み VALUES (NULL, NOW(), 'X', 2, 10);  
INSERT INTO 申し込み VALUES (NULL, NOW(), 'Y', 2, 5);  
INSERT INTO 申し込み VALUES (NULL, NOW(), 'X', 1, 1);  
SELECT 申し込み.日時,申し込み.氏名,申し込み.個数 * 商品.単価  
FROM 申し込み  
JOIN 商品 ON 申し込み.商品番号 = 商品.id;
```

⑬ 「**Execute**」をクリック

SQL 文が**実行**され、結果が表示される。

⑭ 下側のウィンドウで、**結果を確認**。

| +-----+-----+-----  |    |           |       |
|---------------------|----|-----------|-------|
| 日時                  | 氏名 | 申し込み.個数 * | 商品.単価 |
| +-----+-----+-----  |    |           |       |
| 2024-12-05 10:33:10 | X  | 100       |       |
| 2024-12-05 10:33:10 | X  | 100       |       |
| 2024-12-05 10:33:10 | X  | 2000      |       |
| 2024-12-05 10:33:10 | Y  | 1000      |       |
| +-----+-----+-----  |    |           |       |

⑮ 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 商品 (  
    id INTEGER AUTO_INCREMENT PRIMARY KEY,  
    商品名 TEXT,  
    単価 INTEGER  
);  
  
CREATE TABLE 申し込み (  
    id INTEGER AUTO_INCREMENT PRIMARY KEY,  
    日時 DATETIME,  
    氏名 TEXT,  
    商品番号 INTEGER,  
    個数 INTEGER,  
    FOREIGN KEY (商品番号) REFERENCES 商品(id)  
);  
  
INSERT INTO 商品 VALUES (NULL, '商品A', 100);  
INSERT INTO 商品 VALUES (NULL, '商品B', 200);  
INSERT INTO 商品 VALUES (NULL, '商品C', 150);  
INSERT INTO 申し込み VALUES (NULL, NOW(), 'X', 1, 1);  
INSERT INTO 申し込み VALUES (NULL, NOW(), 'X', 2, 10);  
INSERT INTO 申し込み VALUES (NULL, NOW(), 'Y', 2, 5);  
INSERT INTO 申し込み VALUES (NULL, NOW(), 'X', 1, 1);  
SELECT 氏名, SUM(個数 * 商品.単価)  
FROM 申し込み  
JOIN 商品 ON 申し込み.商品番号 = 商品.id  
GROUP BY 氏名;
```

2つのテーブルを使い、  
氏名ごとに申し込みの  
合計金額を求める

⑩ 「**Execute**」をクリック

SQL 文が**実行**され、結果が表示される。

⑪ 下側のウィンドウで、**結果を確認**。

| 氏名 | SUM(個数 * 商品.単価) |
|----|-----------------|
| X  | 2200            |
| Y  | 1000            |

⑮ 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 商品 (  
    id INTEGER AUTO_INCREMENT PRIMARY KEY,  
    商品名 TEXT,  
    単価 INTEGER  
);  
  
CREATE TABLE 申し込み (  
    id INTEGER AUTO_INCREMENT PRIMARY KEY,  
    日時 DATETIME,  
    氏名 TEXT,  
    商品番号 INTEGER,  
    個数 INTEGER,  
    FOREIGN KEY (商品番号) REFERENCES 商品(id)  
);  
  
INSERT INTO 商品 VALUES (NULL, '商品A', 100);  
INSERT INTO 商品 VALUES (NULL, '商品B', 200);  
INSERT INTO 商品 VALUES (NULL, '商品C', 150);  
INSERT INTO 申し込み VALUES (NULL, NOW(), 'X', 1, 1);  
INSERT INTO 申し込み VALUES (NULL, NOW(), 'X', 2, 10);  
INSERT INTO 申し込み VALUES (NULL, NOW(), 'Y', 2, 5);  
INSERT INTO 申し込み VALUES (NULL, NOW(), 'X', 1, 1);  
  
SELECT *  
FROM 商品  
WHERE 単価 > (SELECT AVG(単価) FROM 商品);
```

単価の平均を求める。  
そして、単価の平均より  
も高い単価を持つ  
商品の情報を得る

①⑨ 「**Execute**」をクリック

SQL 文が**実行**され、結果が表示される。

②⑩ 下側のウィンドウで、**結果を確認**。

|    |     |     |  |
|----|-----|-----|--|
|    |     |     |  |
| id | 商品名 | 単価  |  |
| 2  | 商品B | 200 |  |

## 発展演習 1 . 商品の単価の最大値を得るSQL

商品テーブルを使用して、商品の単価の最大値を得るSQLを得る SQL を作成しなさい

| MAX(単価) |
|---------|
| 200     |

ヒント : MAX を使用



## 発展演習 2. Xによる申し込み

氏名が 'X' のすべての申し込み情報を得るSQLを作成しなさい

| id | 日時                   | 氏名 | 商品番号 | 個数 |
|----|----------------------|----|------|----|
| 1  | 2023-12-15T01:12:45Z | X  | 1    | 1  |
| 2  | 2023-12-15T01:12:45Z | X  | 2    | 10 |
| 4  | 2023-12-15T01:12:45Z | X  | 1    | 1  |

(注意) 日時の値は、データを追加した日時になるので、人によって結果が異なる

ヒント : SELECT を使用

## 発展演習 3 . 商品Aを申し込んだ人の取得

商品名が '商品A' である商品を申し込んだすべての人の氏名を得るSQLを作成しなさい。 DISTINCT による重複行の除去も行うこと。

| 氏名 |
|----|
| X  |

ヒント : SELECT、DISTINCT、JOIN、WHERE を使用

## 発展演習 4．氏名別の申し込み数の計算

目的：氏名ごとに、申し込みの回数を得る

申し込みテーブルを使用して、氏名ごとに、申し込みの回数  
を得る SQL を作成しなさい。

| 氏名 | 申し込み数 |
|----|-------|
| X  | 3     |
| Y  | 1     |

ヒント：COUNT と GROUP BY を使用

## 発展演習 5．商品Bに対する申し込み総数の計算

JOINを使用して商品テーブルと申し込みテーブルを結合し、「**商品B**」に対する申し込みの総数を数を得る SQL を作成しなさい

| SUM(申し込み.個数) |
|--------------|
| 15           |

ヒント：WHEREで「商品B」を選択。SUMで合計を求める。

## 解答例

### 発展演習 1 .

```
SELECT MAX(単価)  
FROM 商品;
```

MAX(単価)

200

### 発展演習 2 .

```
SELECT *  
FROM 申し込み  
WHERE 氏名 = 'X';
```

| id | 日時                   | 氏名 | 商品番号 | 個数 |
|----|----------------------|----|------|----|
| 1  | 2023-12-15T01:12:45Z | X  | 1    | 1  |
| 2  | 2023-12-15T01:12:45Z | X  | 2    | 10 |
| 4  | 2023-12-15T01:12:45Z | X  | 1    | 1  |

### 発展演習 3 .

```
SELECT DISTINCT(申し込み.氏名  
)FROM 申し込み  
JOIN 商品 ON 申し込み.商品番号 = 商品.id  
WHERE 商品.商品名 = '商品A';
```

氏名

X

## 解答例

### 発展演習 4.

```
SELECT 氏名, COUNT(*) AS 申し込み数  
FROM 申し込み  
GROUP BY 氏名;
```

| 氏名 | 申し込み数 |
|----|-------|
| X  | 3     |
| Y  | 1     |

### 発展演習 5.

```
SELECT SUM(申し込み.個数)  
FROM 申し込み  
JOIN 商品 ON 申し込み.商品番号 = 商品.id  
WHERE 商品.商品名 = '商品B';
```

| SUM(申し込み.個数) |
|--------------|
| 15           |

## 12-4. データベース操作



## 演習 4 . データベース操作

### 【トピックス】

1. UPDATE ... SET
2. DELETE FROM

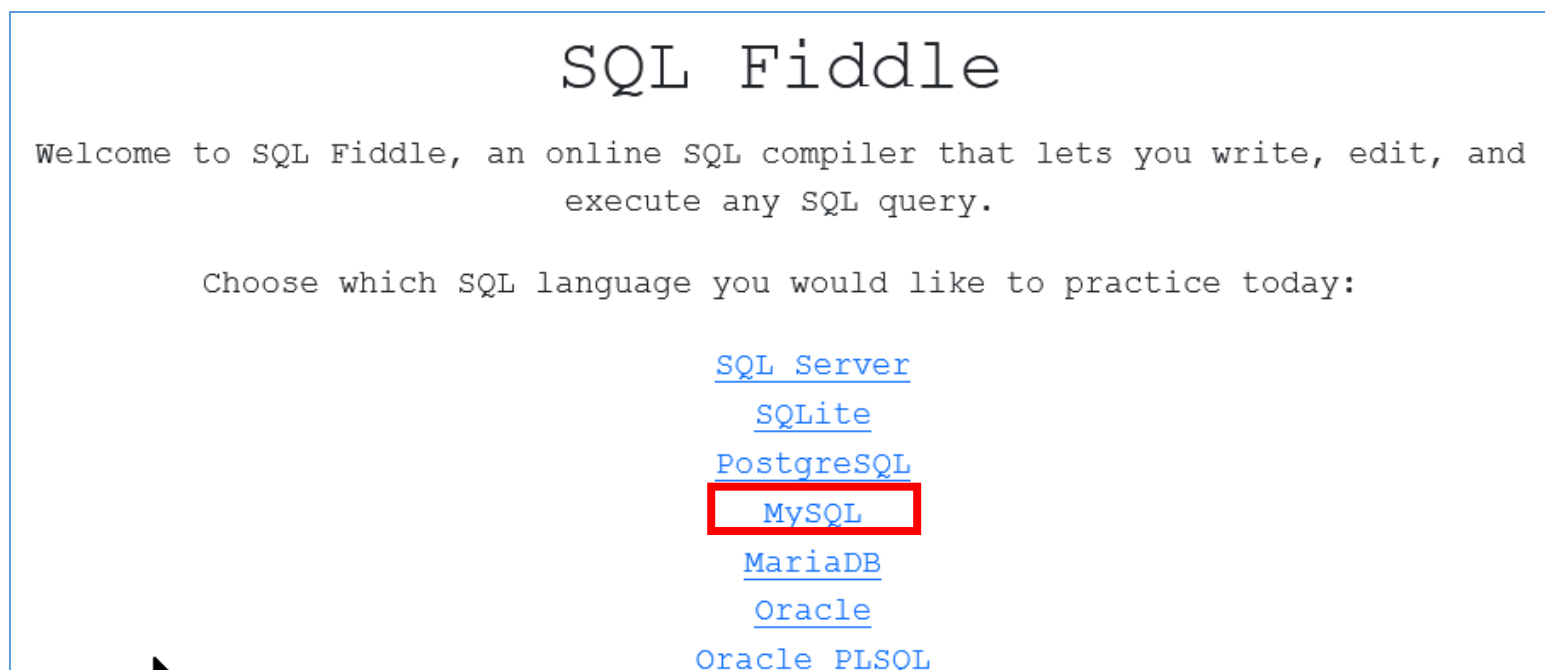


Webブラウザを使用

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 商品 (  
    id INTEGER AUTO_INCREMENT PRIMARY KEY,  
    商品名 TEXT,  
    単価 INTEGER  
);  
INSERT INTO 商品 VALUES (NULL, '商品A', 100);  
INSERT INTO 商品 VALUES (NULL, '商品B', 200);  
INSERT INTO 商品 VALUES (NULL, '商品C', 150);  
UPDATE 商品  
SET 単価 = 120  
WHERE 商品名 = '商品A';  
SELECT * FROM 商品;
```

④ 「**Execute**」をクリック

SQL 文が**実行**され、結果が表示される。

⑤ 下側のウィンドウで、**結果を確認**。

|                     |     |     |  |
|---------------------|-----|-----|--|
| +-----+-----+-----+ |     |     |  |
| id                  | 商品名 | 単価  |  |
| +-----+-----+-----+ |     |     |  |
| 1                   | 商品A | 120 |  |
| 2                   | 商品B | 200 |  |
| 3                   | 商品C | 150 |  |

⑥ 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 商品 (  
    id INTEGER AUTO_INCREMENT PRIMARY KEY,  
    商品名 TEXT,  
    単価 INTEGER  
);  
INSERT INTO 商品 VALUES (NULL, '商品A', 100);  
INSERT INTO 商品 VALUES (NULL, '商品B', 200);  
INSERT INTO 商品 VALUES (NULL, '商品C', 150);  
DELETE FROM 商品  
WHERE 商品名 = '商品C';  
SELECT * FROM 商品;
```

⑨ 「**Execute**」をクリック

SQL 文が**実行**され、結果が表示される。

⑩ 下側のウィンドウで、**結果を確認**。

| id | 商品名 | 単価  |
|----|-----|-----|
| 1  | 商品A | 100 |
| 2  | 商品B | 200 |

## 発展演習 6 . 商品価格の更新

目的：特定の商品の単価を更新する

商品テーブルで、商品名が'**商品B**'の単価を **1000** に更新してください。

| id | 商品名 | 単価   |
|----|-----|------|
| 1  | 商品A | 100  |
| 2  | 商品B | 1000 |

ヒント：UPDATE、SET、WHERE を使用

解答例

発展演習 6.

UPDATE 商品

SET 単価 = 1000

WHERE 商品名 = '商品B';

| id | 商品名 | 単価   |
|----|-----|------|
| 1  | 商品A | 100  |
| 2  | 商品B | 1000 |