# 15-1 SQLite 3 のインストール，データベース新規作成，テーブル定義，テーブルの削除

# アウトライン

1. **SQLite 3 のインストールと基本操作**
2. **テーブル定義，データ型，主キー，SQL 問い合わせ**
3. **結合**

**SQL の基本**を，SQlite 3 を用いて演習する.

URL: https://www.kkaneko.jp/data/sqlite3/index.html

# 第１回のアウトライン

- SQLite 3 のインストール
- SQLite 3 データベースの新規作成
- SQLite 3 コマンドラインシェルの終了

# SQLite 3

- SQLite 3は，**リレーショナルデータベース管理システム**.
- SQLite 3 の特徴
  - アカウント（ユーザ名やパスワード）の機能がない
  - 設定なしで動く
  - 並行処理制御，リモートとの通信の機能は SQLite 3 にない
- SQLite 3 の URL: **https://www.sqlite.org**
- SQLite 3 のコピーライト: **https://www.sqlite.org/copyright.html**
- Windows での SQLite 3 のインストール: **https://www.kkaneko.jp/data/sqlite3/sqlite3.html**

# SQLite 3 のインストール

① SQLite の Web ページを開く.

https://www.sqlite.org/

# SQLite 3 のインストール

## ②「Download」をクリック


## ③ Windows 版のダウンロード

# SQLite 3 のインストール

④ ダウンロードした .zip ファイルを展開（解凍）

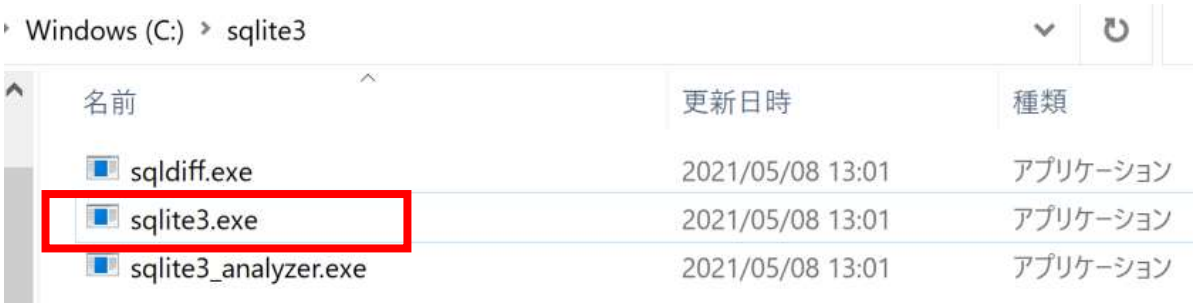
Windows (C:) › sqlite3

| 名前 | 更新日時 | 種類 |
| --- | --- | --- |
| sqldiff.exe | 2021/05/08 13:01 | アプリケーション |
| sqlite3.exe | 2021/05/08 13:01 | アプリケーション |
| sqlite3_analyzer.exe | 2021/05/08 13:01 | アプリケーション |

# SQLite 3 の起動

- 起動

  sqlite3.exe を使う



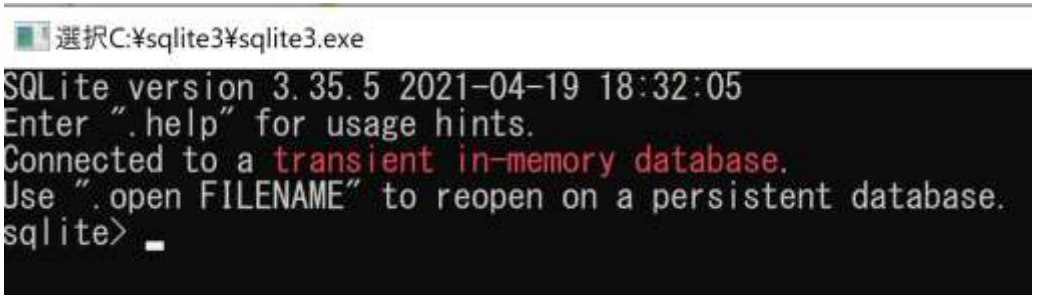
# SQLite 3 データベースの新規作成

データベースファイル名: **C:¥sqlite3¥mydb**
**で，SQLite 3 データベースの新規作成**

① 前もって Windows で **C:¥sqlite3 のような名前のデータベースディレクトリ**を作成しておく

② sqlite3.exe を実行（SQLite 3 の起動）


③ 新しい画面が開くので確認



# SQLite 3 データベースの新規作成

データベースファイル名: **C:¥sqlite3¥mydb**
**で，SQLite 3 データベースの新規作成**

④ 「**.open --new**」で，データベースの**新規作成**

**.open --new C:¥¥sqlite3¥¥mydb**

```
sqlite> .open --new C:¥¥sqlite3¥¥mydb
sqlite>
```

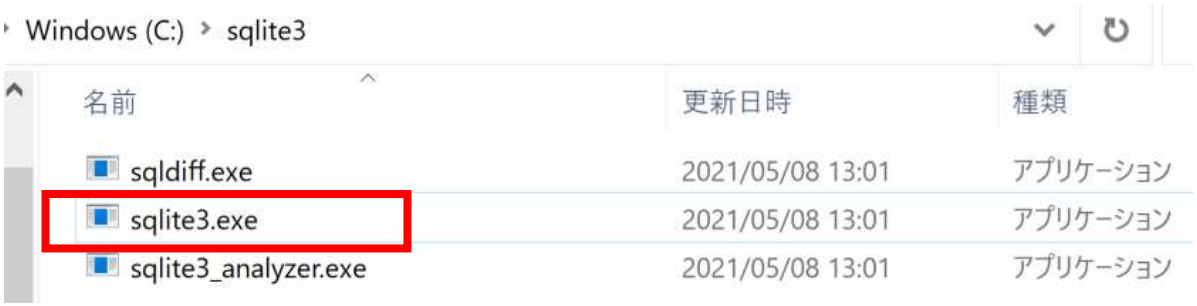
⑤ 「**.exit**」で，SQLite 3 を**終了**

```
sqlite> .exit
```

# SQLite 3 の起動と終了

- **起動**

  sqlite3.exe を使う




- **終了**

  次のコマンドを実行

```
.exit
```

# データベースオープン

SQLite 3 で，次のコマンドを実行

| | |
|---|---|
| **.open --new　<データベースファイル名>** | **新規作成**して**オープン** |
| **.open <データベースファイル名>** | **既存のもの**を**オープン** |

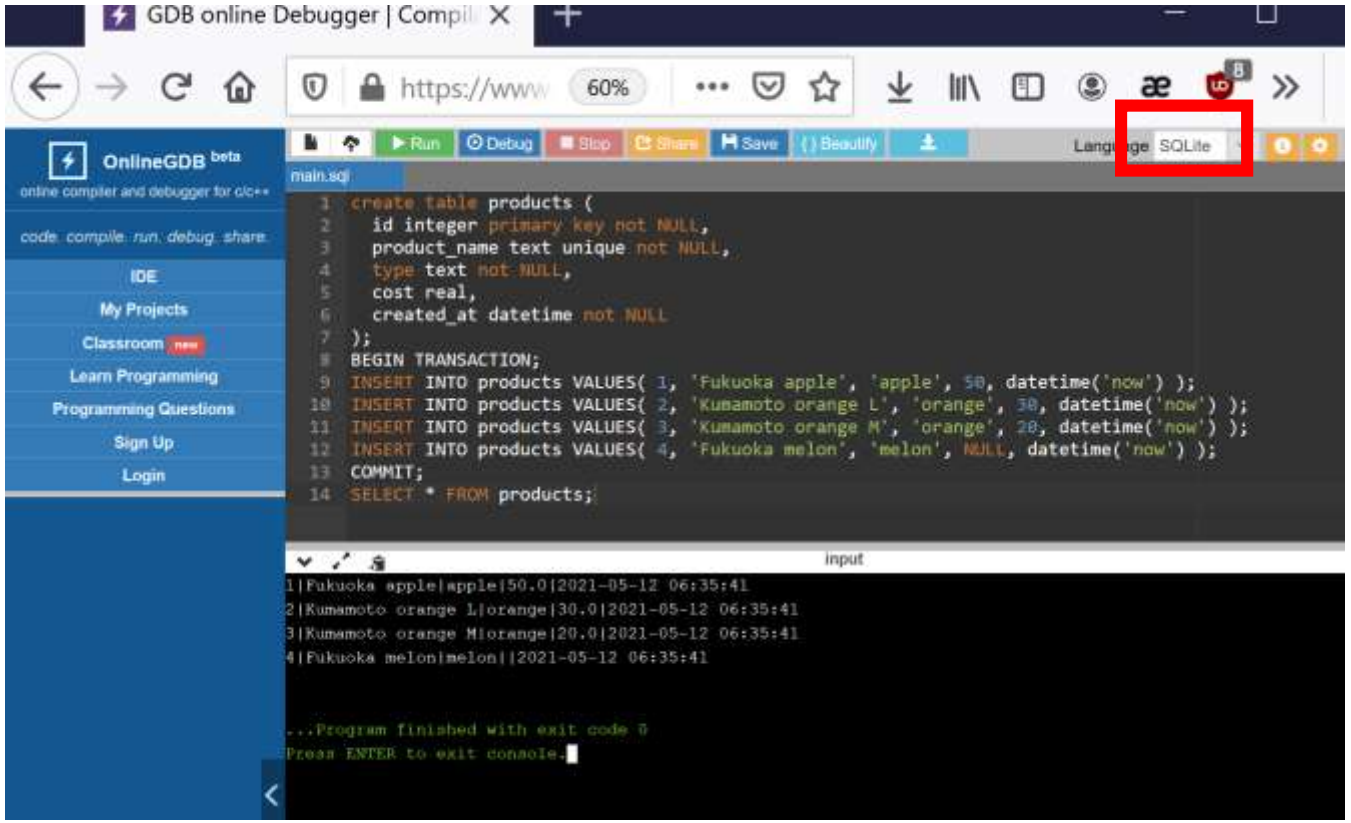
※ データベースの新規作成について

- 指定したファイルが**すでに存在するとき**は，**削除**され，**空のデータベースファイル**が新規作成される
- 指定したファイルが**存在しないとき**は，**空のデータベースファイル**が**新規作成**される

# SQLite 3 を実行できるオンラインサイトの例

- GDB online Debugger のサイト

URL: https://www.onlinegdb.com/




Language で **SQLite** を選ぶ

# 15-2 SQLite3 でテーブルの作成

# アウトライン

- テーブル
- テーブル定義
- データ型
- 主キー
- NULL
- テーブルへのレコードの挿入
- SQL 問い合わせ

# テーブルの例

**テーブル名: products**

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

# テーブル定義

**テーブル名: products**

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

**テーブル定義**では,

- **テーブル名**
- **属性**の**属性名**
- **属性**の**データ型**

**など**を設定して，**テーブル**を**定義**する

```
CREATE TABLE products (
  id INTEGER PRIMARY KEY NOT NULL,
  name TEXT NOT NULL,
  price REAL);
```

# 属性のデータ型

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

**属性名**

**テーブル**の本体

**整数**
**INTEGER**

**長いテキスト**
**TEXT**

**浮動小数点数**
**REAL**

← **SQL のキーワード**

それぞれの**属性**の**データ型**

# 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**

# 主キー

通し番号，学生番号のように，１つのテーブルの中で**同じ値が２回以上出ない**と前もって分かっている**属性**

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

**主キー**

# リレーショナルデータベースの NULL

- **NULL** は「ヌル」あるいは「ナル」と読む
- **リレーショナルデータベース**で **NULL** は，次の場合に使う
  1. **未定，未知，不明（**分からない場合）
  2. **非存在**（もともと存在しない場合）

# テーブル定義と一貫性制約

**【SQL プログラム】**

```
CREATE TABLE products (
  id INTEGER PRIMARY KEY NOT NULL,
  name TEXT NOT NULL,
  price REAL);
```

id:
**主キー** (PRIMARY KEY),
**NULL になることはない** (NOT NULL)
name:
**NULL になることはない** (NOT NULL)

テーブルの制約について記述.
データベースの一貫性を維持するのに役立つ.

# テーブル定義 products

**【SQL プログラム】**

```
CREATE TABLE products (
  id INTEGER PRIMARY KEY NOT NULL,
  name TEXT NOT NULL,
  price REAL);
```

選択C:¥sqlite3¥sqlite3.exe

```
SQLite version 3.35.5 2021-04-19 18:32:05
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> CREATE TABLE products (
   ...>   id INTEGER PRIMARY KEY NOT NULL,
   ...>   name TEXT NOT NULL,
   ...>   price REAL);
sqlite> _
```

# 新しいレコードの挿入

**テーブル名: products**

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

| id | name | price |
|---|---|---|
| **1** | orange | 50 |
| **2** | apple | 100 |
| **3** | melon | 500 |
| **4** | apple | 150 |

**INSERT INTO** products **VALUES**(4, 'apple', 150);

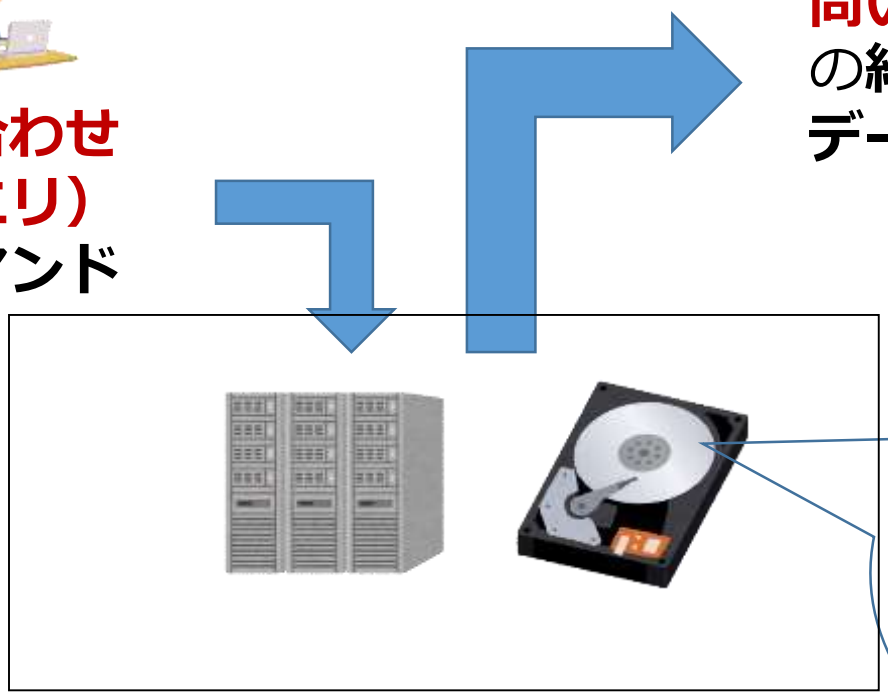
テーブル名　　値の並び．半角のカンマ「,」で区切る

※　文字列は半角の「'」で囲む

# 問い合わせ（クエリ）の仕組み

**問い合わせ（クエリ）**
の**コマンド**

**問い合わせ（クエリ）**
の**結果**は，**テーブル形式の**
**データ**



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

| | | what | at |
|---|---|---|---|
| 赤 | XX | 貸出 | 2021-05-11 13:30:18 |
| 赤 | XX | 返却 | 2021-05-11 13:30:18 |
| 青 | YY | 貸出 | 2021-05-11 13:30:18 |
| 緑 | ZZ | 貸出 | 2021-05-11 13:30:18 |

データの種類ごとに分かれた，たくさんの**テーブル**

**リレーショナル**
**データベースシステム**

# レコードの挿入，SQL 問い合わせ

【SQL プログラム】

```
INSERT INTO products VALUES( 1, 'orange', 50 );
INSERT INTO products VALUES( 2, 'apple', 100 );
INSERT INTO products VALUES( 3, 'melon', 500 );
SELECT * FROM products;
```

```
sqlite> INSERT INTO products VALUES( 1, 'orange', 50 );
sqlite> INSERT INTO products VALUES( 2, 'apple', 100 );
sqlite> INSERT INTO products VALUES( 3, 'melon', 500 );
sqlite> SELECT * FROM products;
1|orange|50.0
2|apple|100.0
3|melon|500.0
sqlite>
```

# SQL 問い合わせ

**【SQL プログラム】**

**SELECT * FROM** products **WHERE** price > 90;

```
sqlite> SELECT * FROM products WHERE  price > 90;
2|apple|100.0
3|melon|500.0
sqlite>
```

# SQL を用いたテーブルの削除

**テーブル products の削除**

**【SQL プログラム】**

**drop table** products;

```
sqlite> drop table products;
sqlite>
```

# ここで使用した SQL

- テーブル定義

  **CREATE TABLE** ...

- 問い合わせ

  **SELECT** ... **FROM** ...

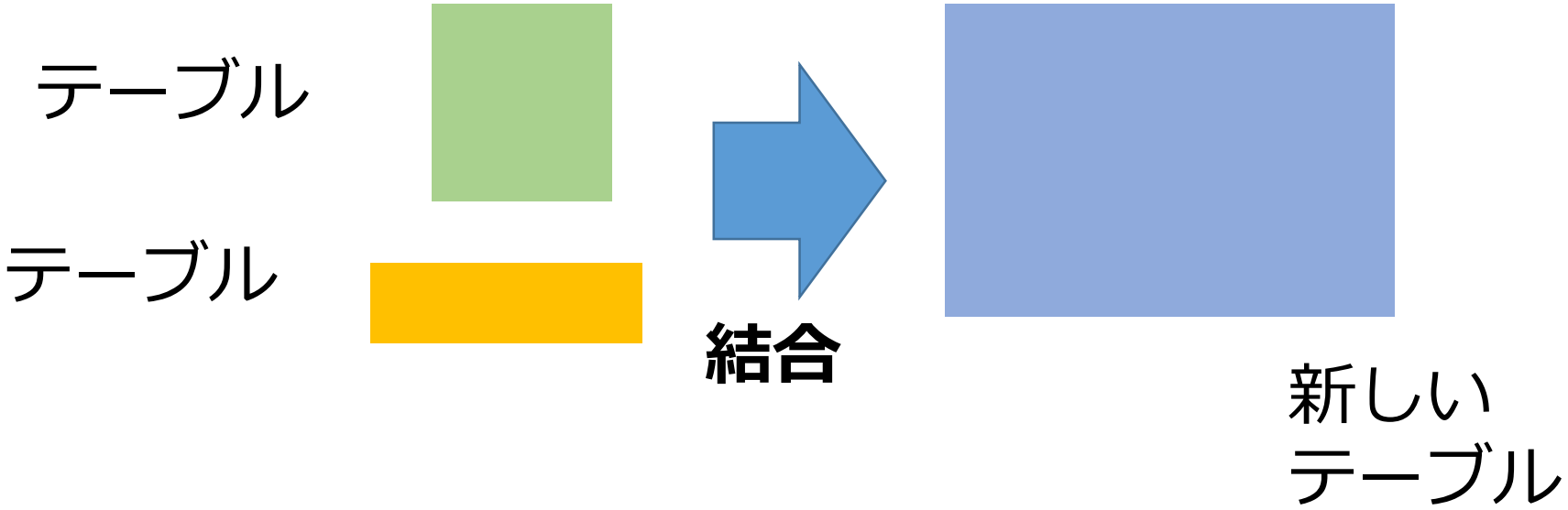
  **SELECT** ... **FROM** ... **WHERE** ...

- レコードの挿入

  **INSERT INTO** ...

# 15-3 SQLによる結合

# 結合



# テーブル定義 teacher

【SQL プログラム】

```
CREATE TABLE teacher (
  id INTEGER PRIMARY KEY NOT NULL,
  name TEXT NOT NULL,
  teacher_name TEXT NOT NULL);
```

選択C:¥sqlite3¥sqlite3.exe

```
SQLite version 3.35.5 2021-04-19 18:32:05
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> CREATE TABLE teacher (
   ...>   id INTEGER PRIMARY KEY NOT NULL,
   ...>   name TEXT NOT NULL,
   ...>   teacher_name TEXT NOT NULL);
sqlite> _
```

# テーブル定義 student

**【SQL プログラム】**

```
CREATE TABLE student (
  id INTEGER PRIMARY KEY NOT NULL,
  student_name TEXT NOT NULL,
  tid INTEGER NOT NULL,
  score INTEGER);
```

```
sqlite> CREATE TABLE student (
   ...>   id INTEGER PRIMARY KEY NOT NULL,
   ...>   student_name TEXT NOT NULL,
   ...>   tid INTEGER NOT NULL,
   ...>   score INTEGER);
sqlite> _
```

# レコードの挿入と確認

**【SQL プログラム】**

```
INSERT INTO teacher VALUES(1, 'db', 'k');
INSERT INTO teacher VALUES(2, 'python', 'a');
SELECT * FROM teacher;
```

```
sqlite> INSERT INTO teacher VALUES(1, 'db', 'k');
sqlite> INSERT INTO teacher VALUES(2, 'python', 'a');
sqlite> SELECT * FROM teacher;
1|db|k
2|python|a
sqlite>
```

# レコードの挿入と確認

**【SQL プログラム】**

```
INSERT INTO student VALUES(1, 'kk', 1, 85);
INSERT INTO student VALUES(2, 'aa', 1, 75);
INSERT INTO student VALUES(3, 'nn', 1, 90);
INSERT INTO student VALUES(4, 'kk', 2, 85);
INSERT INTO student VALUES(5, 'nn', 2, 75);
SELECT * FROM student;
```

```
sqlite> INSERT INTO student VALUES(1, 'kk', 1, 85);
sqlite> INSERT INTO student VALUES(2, 'aa', 1, 75);
sqlite> INSERT INTO student VALUES(3, 'nn', 1, 90);
sqlite> INSERT INTO student VALUES(4, 'kk', 2, 85);
sqlite> INSERT INTO student VALUES(5, 'nn', 2, 75);
sqlite> SELECT * FROM student;
1|kk|1|85
2|aa|1|75
3|nn|1|90
4|kk|2|85
5|nn|2|75
sqlite>
```

# 結合

## 【SQL プログラム】

```
SELECT * FROM teacher, student
WHERE teacher.id = student.tid;
```

```
sqlite> SELECT * FROM teacher, student
   ...> WHERE teacher.id = student.tid;
1|db|k|1|kk|1|85
1|db|k|2|aa|1|75
1|db|k|3|nn|1|90
2|python|a|4|kk|2|85
2|python|a|5|nn|2|75
sqlite>
```