10. データベース設計の基礎：主キー・外部キーを用いたテーブル間の関連付けとSQL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. 主キー
3. 外部キーと参照整合性制約
4. データベース設計

SQLFiddle のサイトにアクセス

Webブラウザを使用

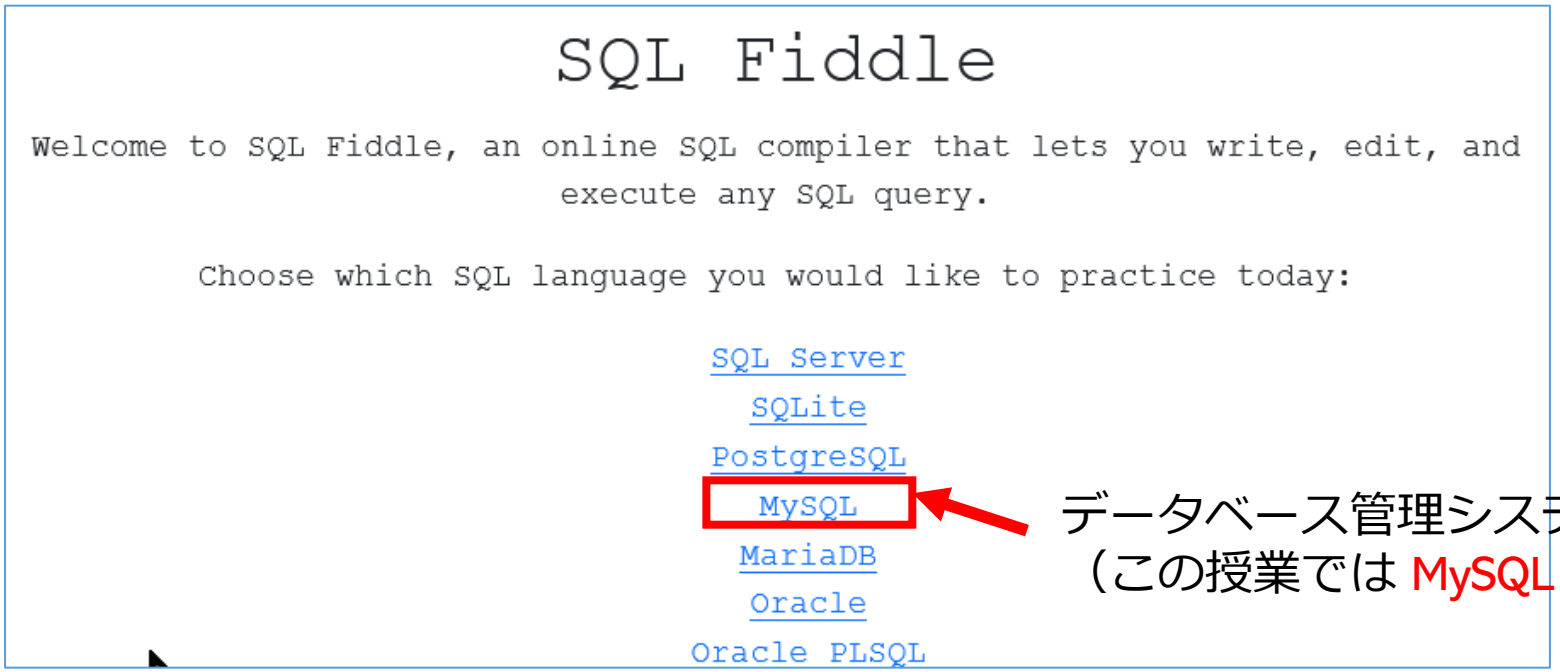
1. ウェブブラウザを開く
2. アドレスバーにSQLFiddleのURLを入力

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

10-1. 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

関連

購入

購入者	商品番号
X	1
X	3
Y	2

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。

10-2. 主キー

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

主キーの役割

- **商品**テーブルで、**すべての商品は一意の ID を持つ**
- 同じ ID をもつ商品は**2つ以上ない**
- 商品の行を**正確に特定するとき**に便利

  例： ID = 2 である行　→　1 つに定まる

ID	商品名	単価
1	**みかん**	**50**
2	**りんご**	100
3	メロン	500
4	**みかん**	30
5	**りんご**	**50**

主キー

主キーの選択基準

- 一意である：　**主キー**は**必ず違う値**になる．（同じ主キーの値が重複することはない）
- 不変である：　値が変更されることはない

ID	商品名	単価
1	**みかん**	**50**
2	**りんご**	100
3	メロン	500
4	**みかん**	30
5	**りんご**	**50**

主キー

PRIMARY KEY の効果

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

主キー

ID	商品名	単価
1	みかん	50
2	りんご	100
3	メロン	500
3	いちご	1000

「PRIMARY KEY」の使用により、
同じID の商品を複数登録することはできなくなる。

学生ID	科目ID	得点
1	1001	90
1	1002	100
2	1001	85
2	1002	90
2	1003	95

主キー

学生ID	科目ID	得点
1	1001	90
1	1002	100
2	1001	85
2	1002	90
2	1003	95
1	1001	95

「PRIMARY KEY」の使用により、
同じ学生が同じ科目に複数回登録することはできなくなる

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

主キー

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

複数属性を主キーとして指定

但し、**主キー**は1つの属性でも良いし、**複数の属性の組み合わせ**でもよい（書き方は下のようになる）

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

SQL の書き方

学生ID	科目ID	得点
1	1001	90
1	1002	100
2	1001	85
2	1002	90
2	1003	95

主キー

```
CREATE TABLE 成績 (
   学生ID INTEGER,
   科目ID INTEGHER,
   得点 INTEGER,
   PRIMARY KEY(学生ID, 科目ID));
```

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

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

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**

SQL によるテーブル定義

- **テーブル名**：**商品**
- **属性名**：**ID、商品名、単価**
- **属性のデータ型**：**数値、テキスト、数値**
- データの整合性を保つための**制約**：**主キー制約**

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

データ追加のSQL

商品

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

```
INSERT INTO 商品 VALUES(1, 'みかん', 50);
INSERT INTO 商品 VALUES(2, 'りんご', 100);
INSERT INTO 商品 VALUES(3, 'メロン', 500);
```

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】

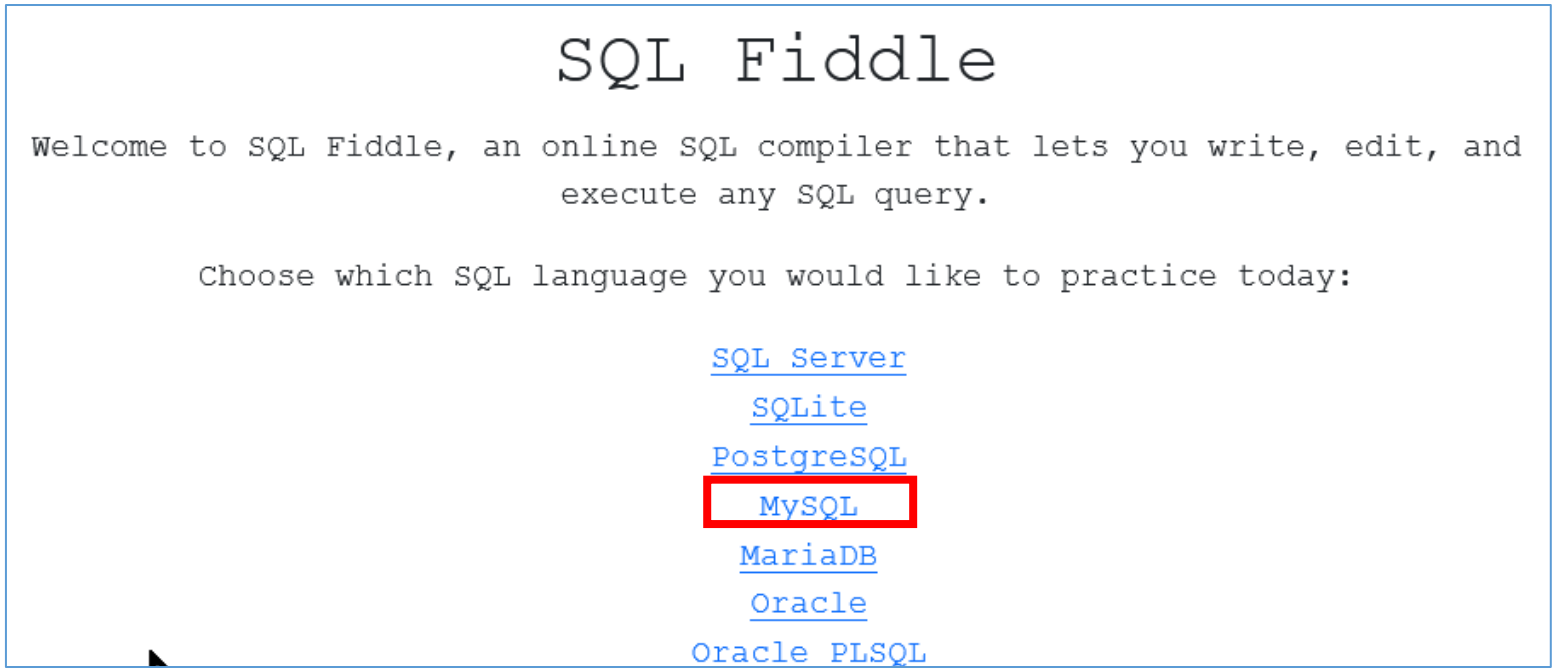
1. **SQL によるテーブル定義**
2. **主キー制約 PRIMARY KEY**
3. **SQL によるデータの追加**
4. **問い合わせ（クエリ）による確認**

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 を入れる。（SQLFiddleで、最初に出てくるSQLは不要なので消す）。

```
CREATE TABLE 商品 (
     ID INTEGER PRIMARY KEY,
     商品名 TEXT,
     単価 INTEGER);
INSERT INTO 商品 VALUES(1, 'みかん', 50);
INSERT INTO 商品 VALUES(2, 'りんご', 100);
INSERT INTO 商品 VALUES(3, 'メロン', 500);
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 下のパネルで、**結果を確認**。

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

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 PRIMARY KEY,
     商品名 TEXT,
     単価 INTEGER);
INSERT INTO 商品 VALUES(1, 'みかん', 50);
INSERT INTO 商品 VALUES(2, 'りんご', 100);
INSERT INTO 商品 VALUES(3, 'メロン', 500);
INSERT INTO 商品 VALUES(1, 'いちご', 1000);
```

⑦ 「**Execute**」をクリック

⑧ 下のパネルで、**結果を確認。**

「主キー制約」に違反している旨の**エラーメッセージ**が出て実行できないことを確認する

これは、**制約違反のチェック機能が働いたため、正常動作**

```
ERROR 1062 (23000) at line 8: Duplicate entry '1' for key '商品.PRIMARY'
```

発展演習１．複数属性の主キー制約
主キー制約の SQL での書き方について理解を深める

次の SQL を実行しなさい
上のパネル

```
CREATE TABLE 成績 (
  学生ID INTEGER,
  科目ID INTEGER,
  得点 INTEGER,
  PRIMARY KEY(学生ID, 科目ID));

INSERT INTO 成績 VALUES(1, 1001, 90);
INSERT INTO 成績 VALUES(1, 1002, 100);
INSERT INTO 成績 VALUES(2, 1001, 85);
INSERT INTO 成績 VALUES(2, 1002, 90);
INSERT INTO 成績 VALUES(2, 1003, 95);
select * FROM 成績;
```

発展演習２．主キー制約の違反

主キー制約に違反する場合、実行できないことを確認する

次の SQL を実行しなさい（**発展演習１に１行増やしている**）

上のパネル

```
CREATE TABLE 成績 (
  学生ID INTEGER,
  科目ID INTEGER,
  得点 INTEGER,
  PRIMARY KEY(学生ID, 科目ID));

INSERT INTO 成績 VALUES(1, 1001, 90);
INSERT INTO 成績 VALUES(1, 1002, 100);
INSERT INTO 成績 VALUES(2, 1001, 85);
INSERT INTO 成績 VALUES(2, 1002, 90);
INSERT INTO 成績 VALUES(2, 1003, 95);
INSERT INTO 成績 VALUES(1, 1001, 95);
select * FROM 成績;
```

- **発展演習 1**

```
| 学生ID | 科目ID | 得点 |
+----------+----------+--------+
| 1 | 1001 | 90 |
| 1 | 1002 | 100 |
| 2 | 1001 | 85 |
| 2 | 1002 | 90 |
| 2 | 1003 | 95 |
+----------+----------+--------+
```

- **発展演習 2**

主キー制約に違反するため実行できない

```
ERROR 1062 (23000) at line 12: Duplicate entry '1-1001' for key '成績.PRIMARY'
```

主キーのまとめ

主キーの目的
- テーブル内の各行を一意に識別。

主キーの性質
- 一意性：重複しない。
- 不変性：値が変更されない。

主キー制約
- 一意性を保証。
- テーブル定義時に設定。

主キーの構成
- 単一または複数の属性で構成可能。

10-3. 外部キーと参照整合性制約

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

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

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**

参照整合性制約のイメージ

商品から
お選びください

枝豆はないんですか？

テーブルのデータを別のテーブルから参照するときの
制約として、**参照整合性制約**がある

参照整合性制約の定義と目的

- **参照整合性制約**は、リレーショナルデータベースにおいて、**異なるテーブル間の関連付けられたデータ**が**一貫性を保つ**ようにするための制約
- **外部キーの値**が、**別のテーブルの主キーと一致する**ことを保証することができる

外部キー

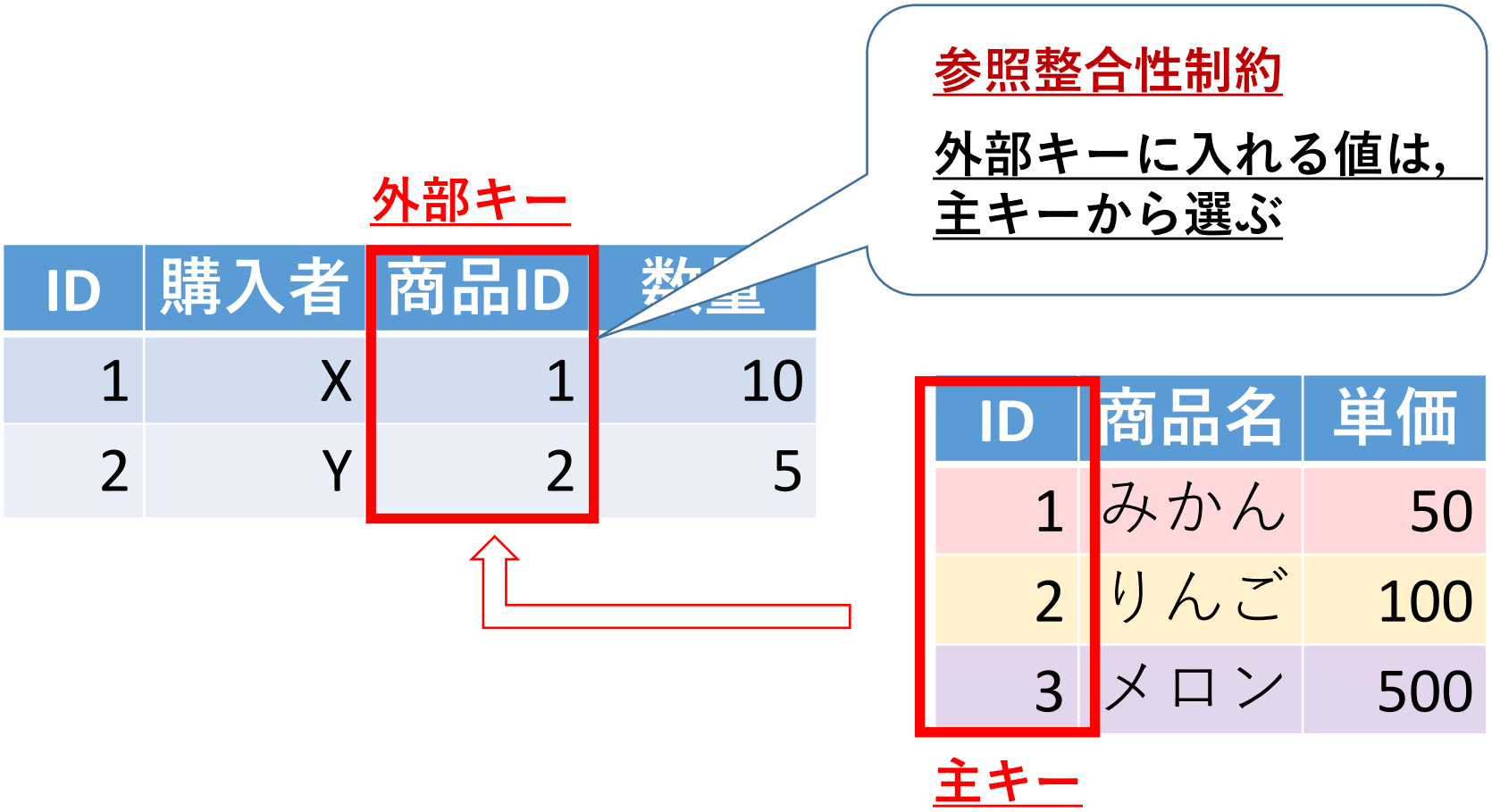
ID	購入者	商品ID	数量
1	X	1	10
2	Y	2	5

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

主キー

参照整合性制約の例

商品テーブルに ID が存在しない場合、購入テーブルに、そのIDを持つ行を挿入することはできない



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

結合の例

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

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

SQLの実行により
新しく生成される
テーブル

演習２．外部キー，参照整合性制約

【トピックス】

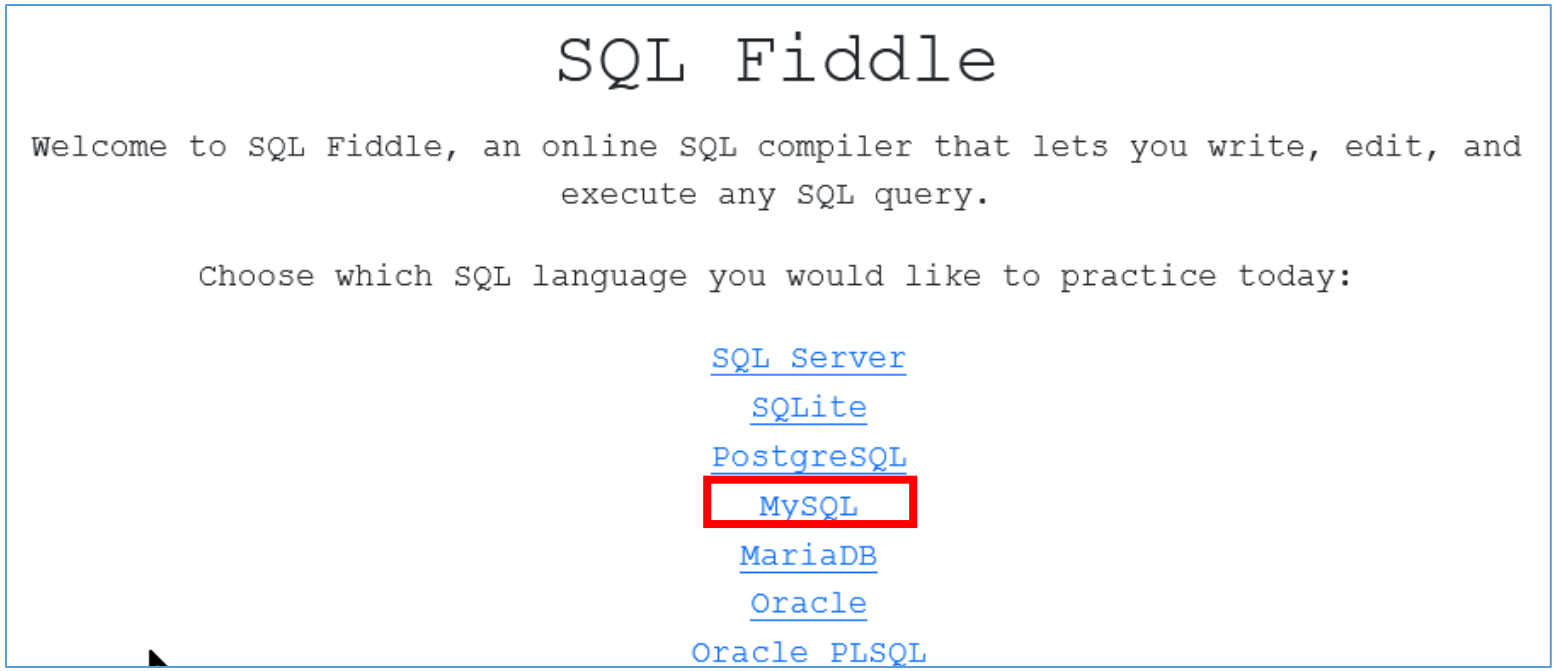
1. **主キー**
2. **外部キー**
3. **参照整合性制約**
4. **PRIMARY KEY**
5. **FOREIGN KEY … REFERENCES**

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 PRIMARY KEY,
     商品名 TEXT,
     単価 INTEGER);
INSERT INTO 商品 VALUES(1, 'みかん', 50);
INSERT INTO 商品 VALUES(2, 'りんご', 100);
INSERT INTO 商品 VALUES(3, 'メロン', 500);
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 下のパネルで、**結果を確認。**

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

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 PRIMARY KEY,
     商品名 TEXT,
     単価 INTEGER);
INSERT INTO 商品 VALUES(1, 'みかん', 50);
INSERT INTO 商品 VALUES(2, 'りんご', 100);
INSERT INTO 商品 VALUES(3, 'メロン', 500);
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));
INSERT INTO 購入 VALUES(1, 'X', 1, 10);
INSERT INTO 購入 VALUES(2, 'Y', 2, 5);
SELECT * FROM 購入;
```

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。**

ID	購入者	商品ID	数量
1	X	1	10
2	Y	2	5

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 PRIMARY KEY,
     商品名 TEXT,
     単価 INTEGER);
INSERT INTO 商品 VALUES(1, 'みかん', 50);
INSERT INTO 商品 VALUES(2, 'りんご', 100);
INSERT INTO 商品 VALUES(3, 'メロン', 500);
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));
INSERT INTO 購入 VALUES(1, 'X', 1, 10);
INSERT INTO 購入 VALUES(2, 'Y', 2, 5);
SELECT 購入.購入者, 商品.商品名, 商品.単価 * 購入.数量
FROM 購入
INNER JOIN 商品 ON 購入.商品ID = 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 下のパネルで、**結果を確認。**

購入者	商品名	商品.単価 * 購入.数量
X	みかん	500
Y	りんご	500

２つのテーブルの利用により，
各購入者の購入商品とその総額が分かる

発展演習３．参照整合性制約の違反
参照整合性制約に違反する場合、実行できないことを確認する

次の SQL を実行しなさい（１行増やしている）
上のパネル

```
CREATE TABLE 商品 (
    ID INTEGER PRIMARY KEY,
    商品名 TEXT,
    単価 INTEGER);
INSERT INTO 商品 VALUES(1, 'みかん', 50);
INSERT INTO 商品 VALUES(2, 'りんご', 100);
INSERT INTO 商品 VALUES(3, 'メロン', 500);

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));
INSERT INTO 購入 VALUES(1, 'X', 1, 10);
INSERT INTO 購入 VALUES(2, 'Y', 2, 5);
INSERT INTO 購入 VALUES(3, 'X', 22, 1);
```

発展演習３のヒント：
「22」は、参照整合性制約に違反

外部キー

ID	購入者	商品ID	数量
1	X	1	10
2	Y	2	5
3	x	22	1

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

主キー

発展演習 3

参照整合性制約に違反するため実行できない

```
ERROR 1452 (23000) at line 17: Cannot add or update a child row: a foreign key
CONSTRAINT `購入_ibfk_1` FOREIGN KEY (`商品ID`) REFERENCES `商品` (`ID`))
```

外部キー、参照整合性制約のまとめ

外部キーの定義

- 他のテーブルの主キーを参照するキー

参照整合性制約の目的

- リレーショナルデータベース内で異なるテーブル間のデータが一貫性を保つようにする
- 外部キーが参照するテーブルの主キーと一致することを保証

参照整合性制約の例

- 商品テーブルに存在しないIDは、購入テーブルに挿入できない。

FOREIGN KEY … REFERENCESの使用

- テーブル定義時に、外部キーが別のテーブルの主キーを参照することを示す。

```
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));
```

10-4. データベース設計

データベースの構築手順

データベース設計

データベース生成

※最初、データベースは**空**

ID	購入者	商品ID	数量

ID	名前	単価

テーブル定義

「こういうテーブルを使いたい」と設定するだけなので、テーブルは**空**

ID	購入者	商品ID	数量
1	X	1	10
2	Y	2	5

ID	名前	単価
1	みかん	50
2	りんご	100
3	りんご	150

テーブル生成

主キー制約の重要性

- **データの一意性の保証**

例: 従業員テーブルで、各従業員に割り当てられた**一意の従業員ID**により、同一人物の**重複登録を防止**

- **データの識別とアクセスの容易化**

例: 従業員IDを用いて特定の従業員を識別。外部キーが機能するためにも利用

参照整合性制約の重要性

- **データの整合性の維持**

例：従業員テーブルの各従業員の部署が、部署テーブルに実在することを保証

- **データの信頼性の向上**

例：存在しない従業員が、給与データに挿入することを防止

- **データの更新時の安全性**

部署のデータを削除する際、その部署に所属する従業員を確認

- **関連データの整理**

FOREIGN KEY と **REFERENCES** を使用して、異なるテーブル間の関連を明確に示すことができる。

データベースの構造が**理解しやすく**なり、**データベースの管理や変更が容易**になる。

１対多の関連

１人の購入者が、複数の購入を行う

外部キー

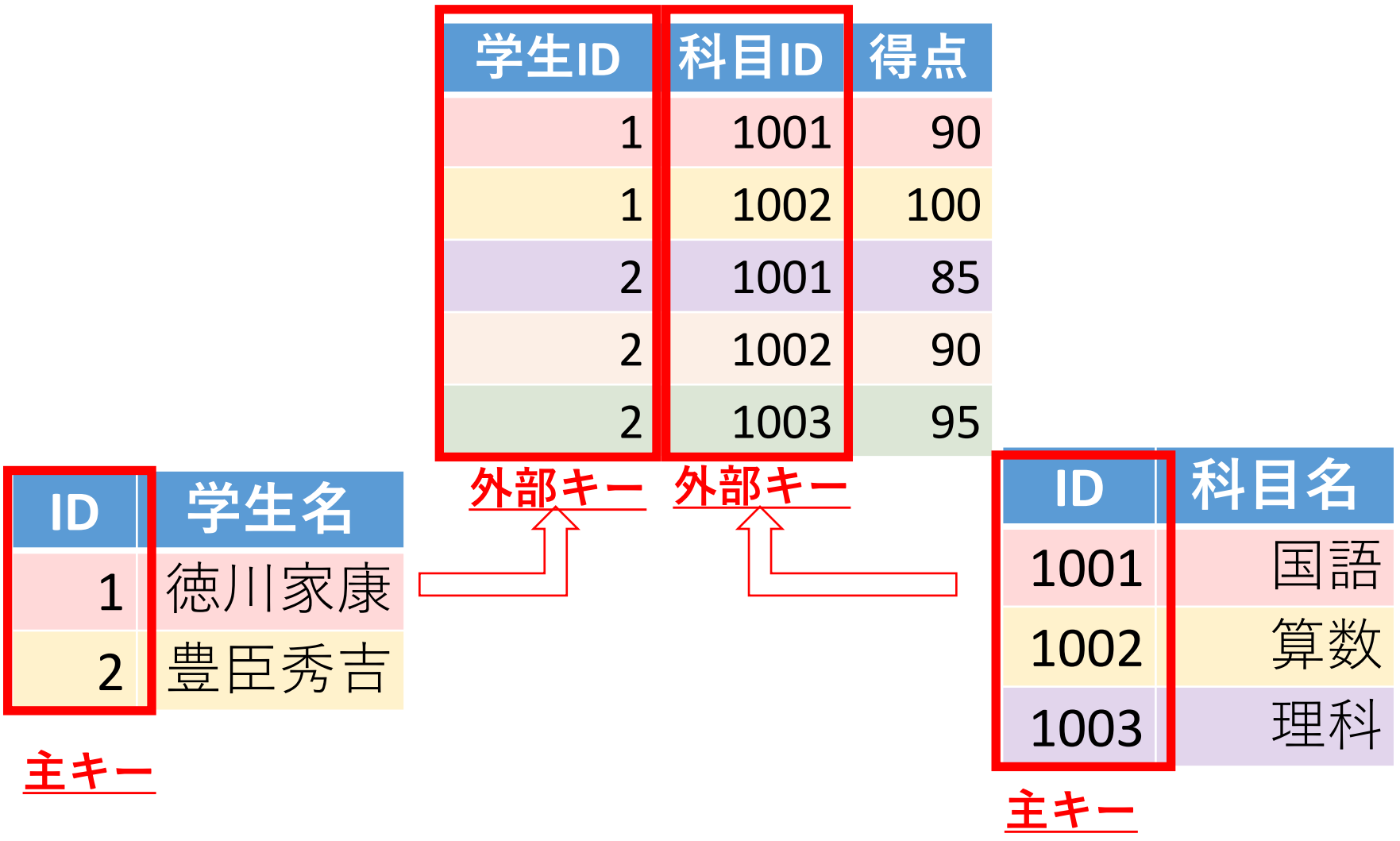
ID	購入者	商品ID	数量
1	X	1	10
2	Y	2	5
3	X	2	2
4	X	3	4
5	Y	3	1

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

主キー

多対多の関連の例

１人の学生が、複数の科目を受講する。１つの科目には複数の学生が参加する



学生ID	科目ID	得点
1	1001	90
1	1002	100
2	1001	85
2	1002	90
2	1003	95

外部キー　外部キー

ID	学生名
1	徳川家康
2	豊臣秀吉

主キー

ID	科目名
1001	国語
1002	算数
1003	理科

主キー

データベース設計のプロセス

１．テーブル名の決定

例: 「従業員」、「顧客」、「商品」など、データの種類や目的に応じた名前。

２．属性の設定

例: 従業員テーブルの属性は「従業員ID」、「名前」、「住所」

３．データ型の選択

例: 「従業員ID」は整数型（INTEGER）、「名前」は文字列型(TEXT)。

４．制約の設定

例: 主キー制約など

５．索引（インデックス）の作成

例: 頻繁に検索される「従業員ID」や「名前」に索引を設定。

６．テーブル間の関係性

例: 「従業員」テーブルの「部署ID」が「部署」テーブルの「部署ID」を参照（外部キーとして）。

データベース設計の実践例

- シナリオ

ある大学の学生、講義、および成績管理システム。

- 目的

学生の個人情報、登録講義、取得成績を効率的に管理。

「成績管理」のデータベース設計の実践例

シナリオ

ある大学の学生、講義、および成績管理システム。

目的

学生の個人情報、登録講義、取得成績を効率的に管理。

- **学生テーブル**:
  - 属性: 学生ID (主キー), 名前, 専攻
  - 主キー: 学生ID
- **講義テーブル**:
  - 属性: 講義ID (主キー), 講義名, 担当教員
  - 主キー: 講義ID
- **成績テーブル**:
  - 属性: 学生ID (外部キー), 講義ID (外部キー), 成績
  - 外部キー: 学生ID (学生テーブル参照), 講義ID (講義テーブル参照)
  - 主キー: 学生IDと講義ID

データベースシステムとアプリケーションの連携

メリット

- **データの一元管理**

データが一箇所に集中され、情報の整合性と一貫性が保たれる

- **効率的なデータアクセス**

アプリケーションは、データベースシステムにアクセスし、迅速なデータ検索と更新が可能

- **拡張性と柔軟性**

データベースの構造を変更しても、アプリケーション側の調整が最小限で済む

- **データのセキュリティ強化**

データベースレベルでのセキュリティ対策により、データの保護が強化される

データベース設計のまとめ

主キー制約の重要性

- データの一意性保証と識別の容易化。

参照整合性制約の重要性

- データの整合性維持、信頼性向上、更新時の安全性確保、関連データの整理。

１対多、多対多の関連

- 例: １人の購入者が複数の購入、１人の学生が複数の科目を受講。

データベース設計のプロセス

- テーブル名、属性、データ型、制約、索引の設定、テーブル間の関係性の決定。

データベースシステムとアプリケーションの連携のメリット

- データの一元管理、効率的なデータアクセス、拡張性と柔軟性、データのセキュリティ強化。

発展演習４．データベース設計の実現

目的：データベース設計を実際に実現するスキルを向上

資料の**「成績管理」のデータベース設計の実践例**　をもとに、SQLでテーブル定義を行いなさい

余裕があれば INSERT INTO でデータを追加してみましょう

発展演習４の解答例

```
CREATE TABLE 学生 (
    学生ID INTEGER PRIMARY KEY,
    名前 TEXT,
    専攻 TEXT
);
CREATE TABLE 講義 (
    講義ID INTEGER PRIMARY KEY,
    講義名 TEXT,
    担当教員 TEXT
);
CREATE TABLE 成績 (
    学生ID INTEGER,
    講義ID INTEGER,
    成績 INTEGER,
    PRIMARY KEY (学生ID, 講義ID),
    FOREIGN KEY (学生ID) REFERENCES 学生(学生ID),
    FOREIGN KEY (講義ID) REFERENCES 講義(講義ID)
);
```

発展演習４の解答例（続き）

```
INSERT INTO 学生 VALUES (1, '山田太郎', '情報工学');
INSERT INTO 学生 VALUES (2, '鈴木花子', '数学');
INSERT INTO 学生 VALUES (3, '佐藤次郎', '物理学');
INSERT INTO 学生 VALUES (4, '田中美咲', '情報工学');
INSERT INTO 学生 VALUES (5, '中村匠', '数学');
INSERT INTO 講義 VALUES (1, 'データベース基礎', '加藤教授');
INSERT INTO 講義 VALUES (2, '線形代数', '木村教授');
INSERT INTO 講義 VALUES (3, 'プログラミング入門', '山本教授');
INSERT INTO 講義 VALUES (4, '確率統計', '木村教授');
INSERT INTO 成績 VALUES (1, 1, 85);
INSERT INTO 成績 VALUES (1, 2, 78);
INSERT INTO 成績 VALUES (1, 3, 92);
INSERT INTO 成績 VALUES (2, 1, 90);
INSERT INTO 成績 VALUES (2, 2, 95);
INSERT INTO 成績 VALUES (3, 1, 82);
INSERT INTO 成績 VALUES (3, 3, 88);
INSERT INTO 成績 VALUES (4, 1, 86);
INSERT INTO 成績 VALUES (4, 2, 83);
INSERT INTO 成績 VALUES (5, 2, 91);
SELECT 学生.名前, 講義.講義名, 成績.成績 FROM 成績 INNER JOIN 学生 ON 成績.学生ID = 学生.学生ID INNER JOIN 講義 ON 成績.講義ID = 講義.講義ID;
```

- 成績テーブルを基準に
- 学生テーブルと講義テーブルを結合し
- 各学生の名前、受講した講義名、その成績を表示

名前	講義名	成績
山田太郎	データベース基礎	85
鈴木花子	データベース基礎	90
佐藤次郎	データベース基礎	82
田中美咲	データベース基礎	86
山田太郎	線形代数	78
鈴木花子	線形代数	95
田中美咲	線形代数	83
中村匠	線形代数	91
山田太郎	プログラミング入門	92
佐藤次郎	プログラミング入門	88

全体まとめ

主キー

- 各テーブル行を一意に識別するキー
- 値は一意であり、重複しない
- 商品テーブルの例：各商品に一意のIDが割り当てられる
- 主キー制約：一意性を保証し、重複を防止。

外部キーと参照整合性制約

- 他のテーブルの主キーを参照するキー
- リレーショナルデータベースで異なるテーブル間のデータ一貫性を保証
- 参照整合性制約の例：商品テーブルのIDが購入テーブルに存在することを保証

データベース設計プロセス

- テーブル名、属性、データ型の選択
- 制約（主キー制約など）の設定
- 索引の作成
- テーブル間の関係性の定義

データベースシステムとアプリケーションの連携

- データの一元管理、効率的なアクセス
- 拡張性、柔軟性、セキュリティの強化

① 理論と実践の両立

実際のデータベース設計において、主キー、外部キー、参照整合性制約などの概念を実際にどのように適用するかを学ぶ。例えば、主キーを用いてデータの一意性を保証し、外部キーを用いて異なるテーブル間のデータの整合性を保つ方法などがある。

② データベース設計の理解の深化

データベース設計の各段階（テーブル名の決定、属性の設定、データ型の選択など）の重要性と、それぞれのステップで考慮すべき点を深く理解する。実際のシナリオ（例えば、学生と講義の関係）に応用する方法を理解する。

③ データベース運用能力の向上

主キーと外部キーの適切な設定を学びます。そのことで、データベース内の情報が一貫性を保ち、信頼性が高まることを理解する。これにより、データベースの日常的な運用や、将来的な変更に対応する際の柔軟性が高まる。