13. データ管理とデータウエアハウス

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

① 反復練習による上達

② データの整理や分析に関するスキルの向上

③ 理解度の確認、難易度の高い演習に挑戦することによる成長

アウトライン

1. イントロダクション
2. 日時の扱い
3. データ管理とデータウエアハウス
4. 演習

SQLFiddle のサイトにアクセス

ウェブブラウザを使用

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

https://sqlfiddle.com/

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。

3. この授業では「**MySQL**」を使用

「**MySQL**」をクリック

SQLFiddle の画面（2024年1月に更新）

- **SQL を編集するパネル**

SQL問い合わせ、テーブル定義、データの追加

実行ボタン

結果ウィンドウ

MySQL Online Compiler

Query

```
  Description VARCHAR(255)
);

INSERT INTO Product(Name, Description) VALUES ('Entity Framework Extensions', 'Use <a
href="https://entityframework-extensions.net/" target="_blank">Entity Framework Extensions</a> to extend
your DbContext with high-performance bulk operations.');
INSERT INTO Product(Name, Description) VALUES ('Dapper Plus', 'Use <a href="https://dapper-plus.net/"
target="_blank">Dapper Plus</a> to extend your IDbConnection with high-performance bulk operations.');
INSERT INTO Product(Name, Description) VALUES ('C# Eval Expression', 'Use <a href="https://eval-
expression.net/" target="_blank">C# Eval Expression</a> to compile and execute C# code at runtime.');

-- QUERY database
SELECT * FROM Product;
SELECT * FROM Product WHERE ProductID = 1;
```

Submit

Legal & Licensing

ProductID	Name	Description
1	Entity Framework Extensions	Use Entity Framework Extensions to extend your DbContext with high-performance bulk operations.
2	Dapper Plus	Use Dapper Plus to extend your IDbConnection with high-performance bulk operations.
3	C# Eval Expression	Use C# Eval Expression to compile and execute C# code at runtime.

ProductID	Name	Description
1	Entity Framework Extensions	Use Entity Framework Extensions to extend your DbContext with high-performance bulk operations.

18 Shares Share Tweet Share Share Print

13-1. 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

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**

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

○ 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、集約、ソート、副問い合わせ**など、高度な操作も可能

テーブル「商品」からデータを取得するSQLの例

- **SELECT** * **FROM** 商品;
- **SELECT** 商品名 **FROM** 商品;
- **SELECT DISTINCT** 商品名 **FROM** 商品;
- **SELECT** 商品名, 単価 **FROM** 商品 **WHERE** 単価 > 80;
- **SELECT** 商品名, 単価 **FROM** 商品 **WHERE** 単価 **BETWEEN** 80 **AND** 85;
- **SELECT AVG**(単価) **FROM** 商品;

  AVG, MAX, MIN, SUM: 平均、最大、最小、合計

  COUNT: 行数
- **SELECT** * **FROM** 商品 **WHERE** 商品名 **LIKE** '%ん';

  「*ん」、「ん*」「*ん*」のように書くことができる

  Access では % でなく *
- **SELECT** * **FROM** 商品 **WHERE** 商品名 **IN** ('みかん', 'りんご');

リレーショナルデータベースシステムの機能

	機能	SQL のキーワード
テーブル定義	テーブル定義	CREATE TABLE
	データ型	CHAR, TEXT, INTEGER, REAL, DATETIME, BIT, NULL
	オートナンバー	MySQL では AUTO_INCREMENT Access では AUTOINCREMENT
	主キー	PRIMARY KEY
	参照整合性制約	FOREIGN KEY, REFERENCES
問い合わせ（クエリ）	射影、選択、結合	SELECT FROM WHERE
	重複行除去	DISTINCT
	比較，範囲指定，パターンマッチ，AND/OR	=, <, >, <>, !=, <=, >=, BETWEEN, LIKE, AND, OR, IS NULL, IS NOT NULL
	集計・集約	GROUP BY, MAX, MIN, COUNT, AVG, SUM
	並べ替え（ソート）	ORDER BY
	副問い合わせ	IN
データ操作	挿入、削除、更新	INSERT INTO, DELETE FROM WHERE, UPDATE SET WHERE
トランザクション	開始、コミット、ロールバック	BEGIN TRANSACTION（MySQL では START TRANSACTION），COMMIT, ROLLBACK

テーブルと属性

テーブル

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

「**ID**」と「**商品名**」と「**単価**」の**属性**

属性のデータ型

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

属性名

テーブル
の本体

整数、主キー

INTEGER PRIMARY KEY

テキスト

TEXT

整数

INTEGER

データ型と SQL

データ型	SQL のキーワードのうち主なもの
未定・未知・非存在	**NULL**
短いテキスト	**char**
長いテキスト	**text**
数値	**integer, real**
日付，時刻，日時	**datetime**
ブール値	**bit, bool**

※ **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

- **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

主キー

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 PRIMAY KEY,
    商品名 TEXT,
    単価 INTEGER);
```

主キー

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 PRIMARY 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

SQLFiddle のサイトにアクセス

① ウェブブラウザを使用、SQLFiddle の MySQL を選ぶ

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

https://sqlfiddle.com/

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。

3. この授業では「**MySQL**」を使用

「**MySQL**」をクリック

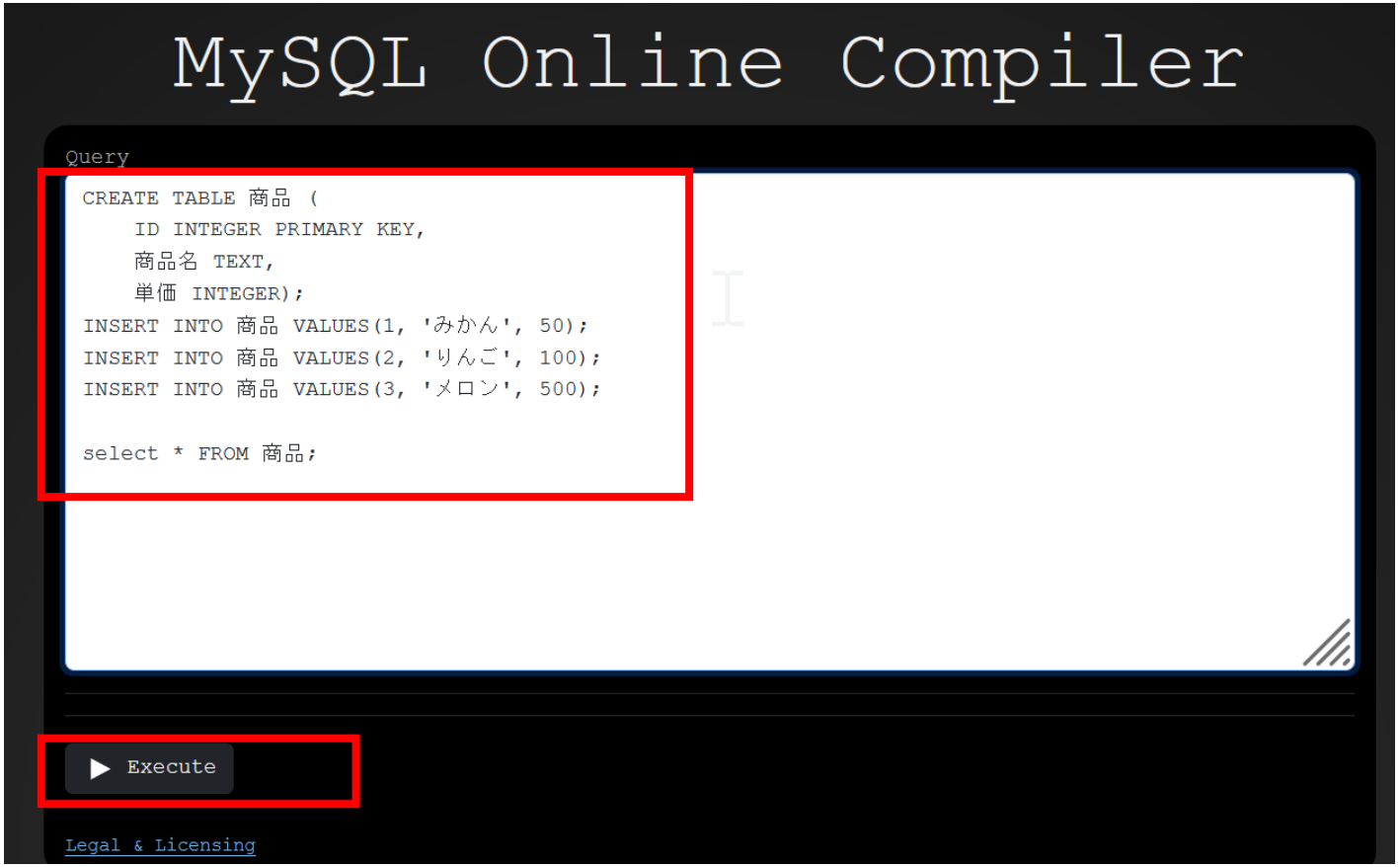
② **パネル**に、**テーブル定義**と**データの追加**と**問い合わせ（クエリ）**を行う SQL を入れる。

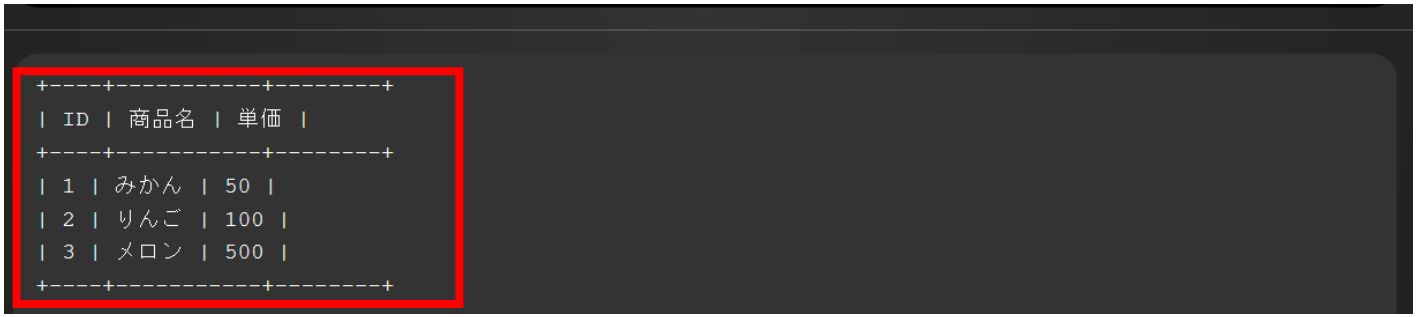
元からある SQL は不要なので消す

```
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;
```

③ 「**Execute**」（または「**Submit**」）をクリック。SQLが実行され、下側のウインドウに結果が表示される。結果を確認





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**：**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

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

SQLFiddle のサイトにアクセス

① ウェブブラウザを使用、SQLFiddle の MySQL を選ぶ

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

https://sqlfiddle.com/

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。

3. この授業では「**MySQL**」を使用

「**MySQL**」をクリック

② **パネル**に、**テーブル定義**と**データの追加**と**問い合わせ（クエリ）**を行う SQL を入れる。

前の演習で用いた SQL は不要である

```
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 商品;
select * FROM 購入;
```

③ 「**Execute**」（または「**Submit**」）をクリック。SQLが実行され、下側のウインドウに結果が表示される。結果を確認

Query

```
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 商品;
select * FROM 購入;
```

Submit

Legal & Licensing

```
+----+----------+--------+
| ID | 商品名 | 単価 |
+----+----------+--------+
| 1 | みかん | 50 |
| 2 | りんご | 100 |
| 3 | メロン | 500 |
+----+----------+--------+
+----+----------+---------+--------+
| ID | 購入者 | 商品ID | 数量 |
+----+----------+---------+--------+
| 1 | X | 1 | 10 |
| 2 | Y | 2 | 5 |
+----+----------+---------+--------+
```

13-2. 日時の扱い

現在日時の表示

現在日時の取得に now() を使用

- **SQL Fiddle の MySQL**

Query

```
select now();
```

now()
2024-01-04 02:36:50

- **Access**

テーブル1 × クエリ1 ×

```
SELECT now();
```

Expr1000
2024/01/04 11:38:44

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、購入日時**
- **属性のデータ型**：**数値、テキスト、数値、数値、日時**
- 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,
  注文日時 DATETIME,
  FOREIGN KEY (商品ID) REFERENCES 商品(ID));
```

演習３． now() による現在日時の取得

【トピックス】

1. **now()**

SQLFiddle のサイトにアクセス

① ウェブブラウザを使用、SQLFiddle の MySQL を選ぶ

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

https://sqlfiddle.com/

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。

3. この授業では「**MySQL**」を使用

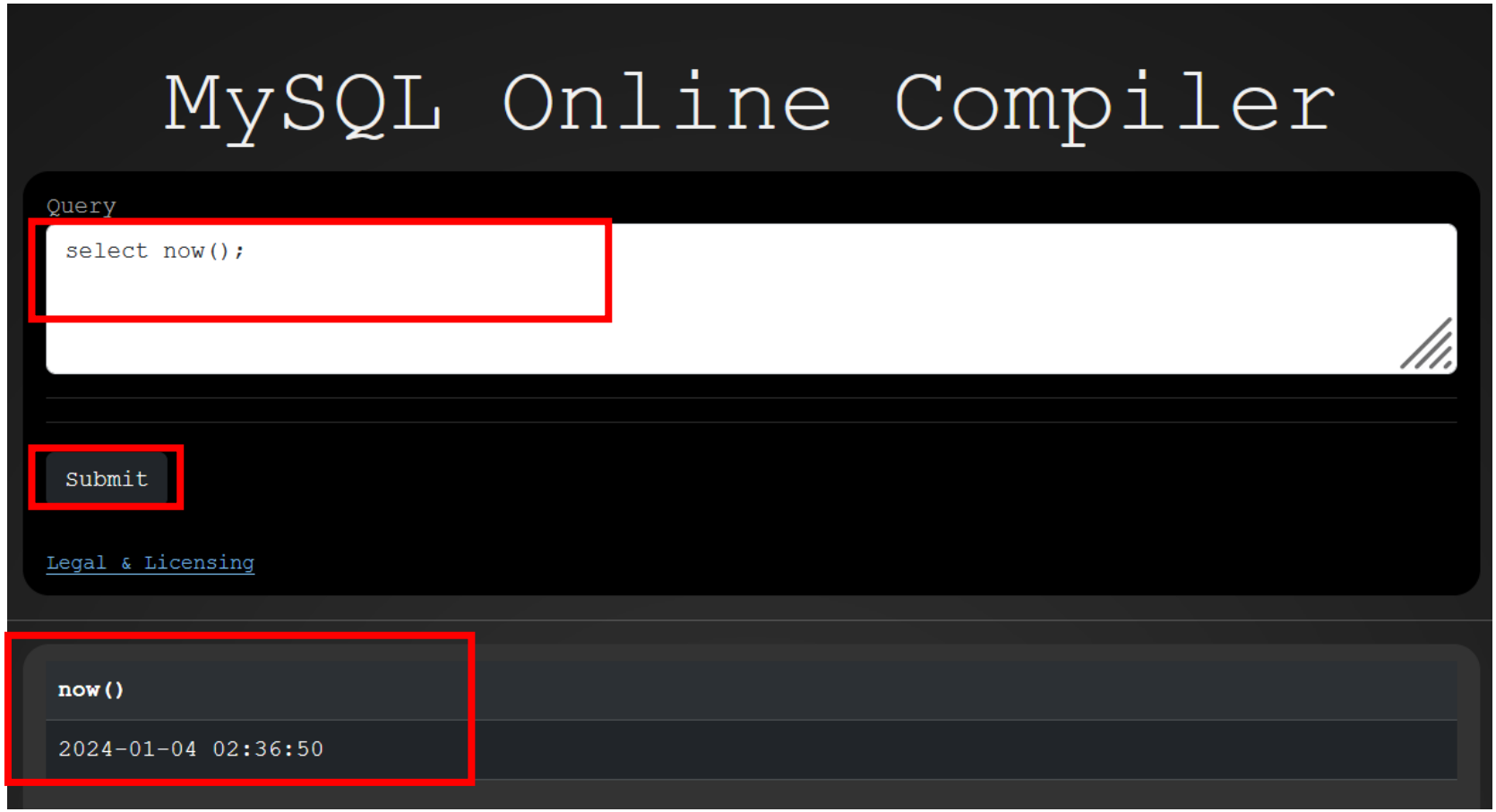
「**MySQL**」をクリック

② **パネル**に、**テーブル定義**と**データの追加**と**問い合わせ（クエリ）**を行う SQL を入れる。

前の演習で用いた SQL は不要である

```
select now();
```

③ 「**Execute**」（または「**Submit**」）をクリック。SQLが実行され、下側のウインドウに結果が表示される。結果を確認



演習４．日時を扱うテーブル

【トピックス】

1. **日時**
2. **DATETIME**

SQLFiddle のサイトにアクセス

① ウェブブラウザを使用、SQLFiddle の MySQL を選ぶ

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

https://sqlfiddle.com/

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。

3. この授業では「**MySQL**」を使用

「**MySQL**」をクリック

② **パネル**に、**テーブル定義**と**データの追加**と**問い合わせ（クエリ）**を行う SQL を入れる。

前の演習で用いた SQL は不要である

```
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,
  購入日時 DATETIME,
  FOREIGN KEY (商品ID) REFERENCES 商品(ID));
INSERT INTO 購入 VALUES(1, 'X', 1, 10, '2023-01-04 09:00:00');
INSERT INTO 購入 VALUES(2, 'Y', 2, 5, '2023-01-04 10:00:00');
select * FROM 商品;
select * FROM 購入;
```

③ 「**Execute**」（または「**Submit**」）をクリック。SQLが実行され、下側のウインドウに結果が表示される。結果を確認

ここまでのまとめ

- **SQL を用いた現在日時の表示** （MySQL や Access などで動く）

```
select now();
```

- **SQL での日時の書き方**

```
'2023-01-04 09:00:00'
```

- **日時を扱うテーブル定義**

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

13-3. データウエアハウス

データの一元管理と保存の利点

- **データの一元管理**

さまざまな種類のデータを一か所で管理することで、データアクセスと利用が容易になる。

- **データの長期保存**

長期間にわたるデータの安全な保存（バックアップなど）が容易になる

- **データ分析**

さまざまな種類のデータを組み合わせた分析が可能になる

データの一元管理と保存

２つの異なるアプローチ

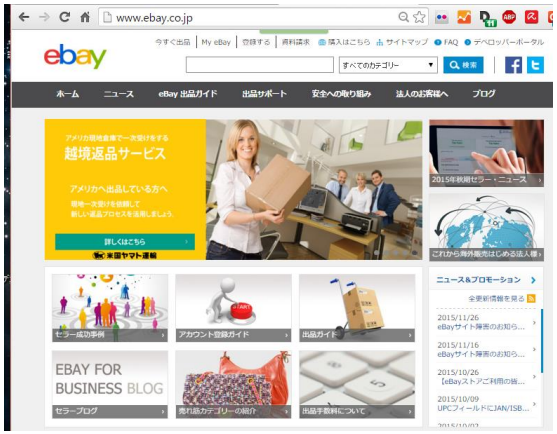
	データウエアハウス	オンライントランザクション
主な目的	データ分析、データからのルール発見	銀行の取引、オンライン予約、販売管理
機能	**長期間のデータを用いたデータ分析**	**トランザクション処理**
使用される技術	SQL、データの挿入、結合、グループ化	SQL、データの挿入と削除と更新、単純な問い合わせ
主な理念	**履歴データ**	**正規化**

データウエアハウスの活用例 ①

◆大量の商品が出品され、購入されるオークションサイト

・不正行動の監視

・出品者、購入者が「どうすればより満足するか」の分析



eBay Web ページ
http://www.ebay.co.jp/

出品、購入の状況を丸ごと記録

出典: Teradata 社 Web ページ
http://jpn.teradata.jp/library/nyumon/ins_1904.html

データウエアハウスの活用例 ②

◆「空席で飛ばすくらいなら、安値でも売りたい」と思っている航空会社

・満席になりそうか、空席はいくつになりそうかを的確に予測

・空席をぴったり埋めるために、ダイナミックに値下げ／値上げ

予約、キャンセルの状況を丸ごと記録

出典: Teradata 社 Web ページ
http://jpn.teradata.jp/library/nyumon/ins_1904.html

データウエアハウスとオンライントランザクションの比較

- **オンライントランザクション：最新情報を使用**

予約テーブル

氏名	予約内容
XX	おせちA
YY	おせちB
ZZ	おせちA

価格テーブル

商品	価格
おせちA	10000
おせちB	5000

- **データウエアハウス：データは一度格納されると消えることなく残り続ける。「履歴データ」**の概念を重視

予約テーブル

氏名	予約内容	予約日	キャンセル日
XX	おせちA	2023-12-01	
YY	おせちB	2023-12-04	
ZZ	おせちB	2023-12-05	2023-12-06
ZZ	おせちA	2023-12-06	

価格テーブル

商品	価格	価格改定日
おせちA	12000	2023-11-10
おせちA	10000	2023-11-20
おせちB	5000	2023-11-10

オンライントランザクションの特性

- **データの値の変更や削除**を行う**オンラントランザクションシステム**では、**正規化は欠かせない**
- **正規化**は、**リレーショナルデータベースのテーブルを適切な形に再構成**することで、**データの冗長性を排除**し、**データの整合性**を向上させるプロセス

正規化前

氏名	予約内容	価格
XX	おせちA	10000
YY	おせちB	5000
ZZ	おせちA	10000

冗長なデータがある

正規化後

氏名	予約内容
XX	おせちA
YY	おせちB
ZZ	おせちA

商品	価格
おせちA	10000
おせちB	5000

冗長なデータがない

正規化により、元のテーブルにあった**冗長性を排除**.

データウエアハウスの特性

- **データウエアハウス**では、 **「履歴データ」**の概念を重視
- **データの各行に、日時情報を付加**し、**時間の経過に応じたデータの変化を保持する**
- **一度保存されたデータ**は、原則として**削除されることはない**
- データの**値の変更**は、原則として**行われない．データの値の変更等に伴う異状が発生しない**

予約テーブル

氏名	予約内容	予約日	キャンセル日
XX	おせちA	2023-12-01	
YY	おせちB	2023-12-04	
ZZ	おせちB	2023-12-05	2023-12-06
ZZ	おせちA	2023-12-06	

価格テーブル

商品	価格	価格改定日
おせちA	12000	2023-11-10
おせちA	10000	2023-11-20
おせちB	5000	2023-11-10

「商品の価格は１つに決まっている」という考え方ではなく、
商品の価格の履歴データを保持

ここまでのまとめ

データの一元管理

- さまざまなのデータを一元管理。管理と利用を容易に。
- 複数のデータを組み合わせた分析が可能に

データウエアハウス

- 「履歴データ」の概念。データの時間に基づく変化を保持。**長期間のデータを用いた分析を可能に**。
- **一度保存されたデータは恒久的に保持**され、削除や変更は原則として行われない。

オンライントランザクション

- トランザクションをリアルタイム処理。**最新のデータをオンラインで共有**。
- データの値の変更等に伴い**異状が発生**する可能性がある。そのため、**正規化**によりデータの冗長性を排除し、整合性を保つ。

13-4. 演習

- ある商店は「**みかん**」、「りんご」、「**メロン**」の３商品を扱っている
- それぞれの商品の単価は変化する。その履歴を次のようなテーブルで扱う

商品テーブル

ID	商品名	単価	改訂日時
1	みかん	50	2023-12-01 09:00:00
2	りんご	100	2023-12-01 09:00:00
3	メロン	500	2023-12-01 09:00:00
4	りんご	150	2024-01-01 09:00:00
5	メロン	400	2024-01-01 09:00:00

- 購入を次のようなテーブルで扱う

購入テーブル

ID	購入者	商品ID	数量	購入日時
1	X	1	10	2023-12-10 10:00:00
2	Y	2	5	2023-12-20 12:00:00
3	Y	4	20	2024-01-05 09:00:00
4	Z	5	3	2024-01-05 11:00:00

演習５．データウエアは渦の構築

【トピックス】

1. **日時**
2. **DATETIME**
3. **履歴データ**

SQLFiddle のサイトにアクセス

① ウェブブラウザを使用、SQLFiddle の MySQL を選ぶ

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

https://sqlfiddle.com/

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。

3. この授業では「**MySQL**」を使用

「**MySQL**」をクリック

② **パネル**に、**テーブル定義**と**データの追加**と**問い合わせ（クエリ）**を行うSQLを入れる。

前の演習で用いたSQLは不要である

```
CREATE TABLE 商品 (
    ID INTEGER PRIMARY KEY,
    商品名 TEXT,
    単価 INTEGER,
    改訂日時 DATETIME);
INSERT INTO 商品 VALUES(1, 'みかん', 50, '2023-12-01 09:00:00');
INSERT INTO 商品 VALUES(2, 'りんご', 100, '2023-12-01 09:00:00');
INSERT INTO 商品 VALUES(3, 'メロン', 500, '2023-12-01 09:00:00');
INSERT INTO 商品 VALUES(4, 'りんご', 150, '2024-01-01 09:00:00');
INSERT INTO 商品 VALUES(5, 'メロン', 400, '2024-01-01 09:00:00');
CREATE TABLE 購入 (
    ID INTEGER PRIMARY KEY,
    購入者 TEXT,
    商品ID INTEGER,
    数量 INTEGER,
    購入日時 DATETIME,
    FOREIGN KEY (商品ID) REFERENCES 商品(ID));
INSERT INTO 購入 VALUES(1, 'X', 1, 10, '2023-12-10 10:00:00');
INSERT INTO 購入 VALUES(2, 'Y', 2, 5, '2023-12-20 12:00:00');
INSERT INTO 購入 VALUES(3, 'Y', 4, 20, '2024-01-05 09:00:00');
INSERT INTO 購入 VALUES(4, 'Z', 5, 3, '2024-01-05 11:00:00');
select * FROM 商品;
select * FROM 購入;
```

③ 「**Execute**」（または「**Submit**」）をクリック。SQLが実行され、下側のウインドウに結果が表示される。結果を確認

MySQL Online Compiler

Query

```
CREATE TABLE 商品 (
    ID INTEGER PRIMARY KEY,
    商品名 TEXT,
    単価 INTEGER,
    改訂日時 DATETIME);
INSERT INTO 商品 VALUES(1, 'みかん', 50, '2023-12-01 09:00:00');
INSERT INTO 商品 VALUES(2, 'りんご', 100, '2023-12-01 09:00:00');
INSERT INTO 商品 VALUES(3, 'メロン', 500, '2023-12-01 09:00:00');
INSERT INTO 商品 VALUES(4, 'りんご', 150, '2024-01-01 09:00:00');
INSERT INTO 商品 VALUES(5, 'メロン', 400, '2024-01-01 09:00:00');
CREATE TABLE 購入 (
    ID INTEGER PRIMARY KEY,
    購入者 TEXT,
    商品ID INTEGER,
    数量 INTEGER,
    購入日時 DATETIME,
    FOREIGN KEY (商品ID) REFERENCES 商品(ID));
INSERT INTO 購入 VALUES(1, 'X', 1, 10, '2023-12-10 10:00:00');
INSERT INTO 購入 VALUES(2, 'Y', 2, 5, '2023-12-20 12:00:00');
INSERT INTO 購入 VALUES(3, 'Y', 4, 20, '2024-01-05 09:00:00');
INSERT INTO 購入 VALUES(4, 'Z', 5, 3, '2024-01-05 11:00:00');
select * FROM 商品;
select * FROM 購入;
```

Submit

Legal & Licensing

```
+----+-----------+--------+---------------------+
| ID | 商品名 | 単価 | 改訂日時 |
+----+-----------+--------+---------------------+
| 1 | みかん | 50 | 2023-12-01 09:00:00 |
| 2 | りんご | 100 | 2023-12-01 09:00:00 |
| 3 | メロン | 500 | 2023-12-01 09:00:00 |
| 4 | りんご | 150 | 2024-01-01 09:00:00 |
| 5 | メロン | 400 | 2024-01-01 09:00:00 |
+----+-----------+--------+---------------------+
+----+-----------+----------+--------+---------------------+
| ID | 購入者 | 商品ID | 数量 | 購入日時 |
+----+-----------+----------+--------+---------------------+
| 1 | X | 1 | 10 | 2023-12-10 10:00:00 |
| 2 | Y | 2 | 5 | 2023-12-20 12:00:00 |
| 3 | Y | 4 | 20 | 2024-01-05 09:00:00 |
| 4 | Z | 5 | 3 | 2024-01-05 11:00:00 |
+----+-----------+----------+--------+---------------------+
```

④ **問い合わせ（クエリ）**を行う SQL を**追加**。「**Execute**」（または「**Submit**」）をクリックして実行し、結果を確認

```
SELECT * FROM 商品
JOIN 購入 ON 商品.ID = 購入.商品ID;
```

```
SELECT * FROM 商品
JOIN 購入 ON 商品.ID = 購入.商品ID;
```

ID	商品名	単価	改訂日時	ID	購入者	商品ID	数量	購入日時
1	みかん	50	2023-12-01 09:00:00	1	X	1	10	2023-12-10 10:00:00
2	りんご	100	2023-12-01 09:00:00	2	Y	2	5	2023-12-20 12:00:00
4	りんご	150	2024-01-01 09:00:00	3	Y	4	20	2024-01-05 09:00:00
5	メロン	400	2024-01-01 09:00:00	4	Z	5	3	2024-01-05 11:00:00

⑤ **問い合わせ（クエリ）**を行う SQL を**追加**。「**Execute**」（または「**Submit**」）をクリックして実行し、結果を確認

```
SELECT 購入.購入日時, 購入.購入者, 購入.数量 * 商品.単価
FROM 商品
JOIN 購入 ON 商品.ID = 購入.商品ID;
```

```
SELECT 購入.購入日時, 購入.購入者, 購入.数量 * 商品.単価
FROM 商品
JOIN 購入 ON 商品.ID = 購入.商品ID;
```

購入日時	購入者	購入.数量 * 商品.単価
2023-12-10 10:00:00	X	500
2023-12-20 12:00:00	Y	500
2024-01-05 09:00:00	Y	3000
2024-01-05 11:00:00	Z	1200

⑥ **問い合わせ（クエリ）**を行う SQL を**追加**。「**Execute**」（または「**Submit**」）をクリックして実行し、結果を確認

```
SELECT 購入.購入者, SUM(購入.数量 * 商品.単価)
FROM 商品
JOIN 購入 ON 商品.ID = 購入.商品ID
GROUP BY 購入.購入者;
```

２つのテーブルを使い、購入者ごとに申し込みの合計金額を求める

```
SELECT 購入.購入者, SUM(購入.数量 * 商品.単価)
FROM 商品
JOIN 購入 ON 商品.ID = 購入.商品ID
GROUP BY 購入.購入者;
```

購入者	SUM(購入.数量 * 商品.単価)
X	500
Y	3500
Z	1200

⑦ **問い合わせ（クエリ）**を行う SQL を**追加**。「**Execute**」（または「**Submit**」）をクリックして実行し、結果を確認

```
SELECT 商品名, MAX(改訂日時)
FROM 商品
GROUP BY 商品名;
```

各商品の最新の
改訂日時を得ている。

```
SELECT 商品名, MAX(改訂日時)
FROM 商品
GROUP BY 商品名;
```

商品名	MAX(改訂日時)
みかん	2023-12-01 09:00:00
りんご	2024-01-01 09:00:00
メロン	2024-01-01 09:00:00

⑧ **問い合わせ（クエリ）**を行う SQL を**追加**。「**Execute**」（または「**Submit**」）をクリックして実行し、結果を確認

```
SELECT 商品名, 単価, 改訂日時
FROM 商品
WHERE (商品名, 改訂日時) IN (
    SELECT 商品名, MAX(改訂日時)
    FROM 商品
    GROUP BY 商品名);
```

```
SELECT 商品名, 単価, 改訂日時
FROM 商品
WHERE (商品名, 改訂日時) IN (
    SELECT 商品名, MAX(改訂日時)
    FROM 商品
    GROUP BY 商品名);
```

商品名	単価	改訂日時
みかん	50	2023-12-01 09:00:00
りんご	150	2024-01-01 09:00:00
メロン	400	2024-01-01 09:00:00

副問い合わせで、最新の改訂日時である行を得る。その結果を用いて最新価格を得る

自習１．Yによる購入

購入テーブルを用いて、購入者が 「Y」 のすべての購入情報を得るSQLを作成しなさい

ID	購入者	商品ID	数量	購入日時
2	Y	2	5	2023-12-20 12:00:00
3	Y	4	20	2024-01-05 09:00:00

ヒント：SELECT を使用

自習２．商品「りんご」を購入した人の取得

商品名が 「**りんご**」である商品を購入したすべての購入者を得るSQLを作成しなさい。 DISTINCT による重複行の除去も行うこと。

```
+----------+
| 購入者 |
+----------+
| Y |
+----------+
```

ヒント：SELECT、DISTINCT、JOIN、WHERE を使用

自習３．購入者別の申し込み数の計算

目的：購入者ごとに、申し込みの回数を得る

購入テーブルを使用して、購入者ごとに、購入の回数を得る SQL を作成しなさい。

```
| 購入者 | COUNT(*) |
+----------+----------+
| X | 1 |
| Y | 2 |
| Z | 1 |
+----------+----------+
```

ヒント：COUNT と GROUP BY を使用

解答例

自習１.

SELECT *

FROM 購入

WHERE 購入者 = '**Y**';

ID	購入者	商品ID	数量	購入日時
2	Y	2	5	2023-12-20 12:00:00
3	Y	4	20	2024-01-05 09:00:00

自習２.

SELECT DISTINCT(購入.購入者)

FROM 購入

JOIN 商品 ON 購入.商品ID = 商品.ID

WHERE 商品.商品名 = '**りんご**';

購入者
Y

解答例

自習３.

SELECT 購入者, COUNT(*)

FROM 購入

GROUP BY 購入者;

購入者	COUNT(*)
X	1
Y	2
Z	1

全体まとめ

- データの一元管理により、多様なデータを組み合わせた分析が可能になる。
- **オンライントランザクション**では、**リアルタイムのデータ処理**により、**オンラインでの情報共有**を行う。
- **データウェアハウス**は**「履歴データ」を重視**し、**データは一度格納されると削除、変更されることなく保存**される。
- 購入テーブル（属性は、ID、購入者、商品ID、数量、購入日時）では、「購入日時」の属性を用いて購入履歴を管理できるようになる。
- 商品テーブル（属性は、ID、名前、単価、改訂日時）では、「改訂日時」の属性を持ちいて価格の履歴を管理できるようになる。
- **SQL**では、「**DATETIME**」を使って**日時**や**履歴**を扱うことができる。
- **日時のデータ**は SQL では、「**'2024-01-05 09:00:00'**」のような形式で表される。
- MySQL や Access では、「**select now();**」により、現在の日時が表示される