de-13. データ管理の基礎：オンラインイントランザクションとデータウェアハウスの特徴と活用

（データベース演習）

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

金子邦彦

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：最新情報のみを使用**

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：「履歴データ」を重視する**

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

履歴データの概念

- **データの各行に日時情報**を付加
- データ変化があるたびに新しい行を挿入
- **一度保存されたデータは削除しない**
- 過去のデータの変化の履歴を保持

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

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**

データ管理のまとめ

データウエアハウス

- 「履歴データ」の利用
- データの時間の時間による分析を可能にする
- **一度保存されたデータは恒久的に保持**される
- 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

- 最新のデータを保持する
- データの値の変更や削除に伴い異状が発生する可能性がある.
- 正規化によりデータの冗長性を排除し，整合性を確保する

Access での注意点

- **SQLビューでは、SQL文を１つずつ実行**

  （複数まとめての一括実行ができない）

- **CREATE TABLE** では、「実行」の後、**画面が変化しない**が実行できている
- **INSERT INTO** では、「実行」の後、**確認表示**が出る。その後、**画面が変化しない**が実行できている

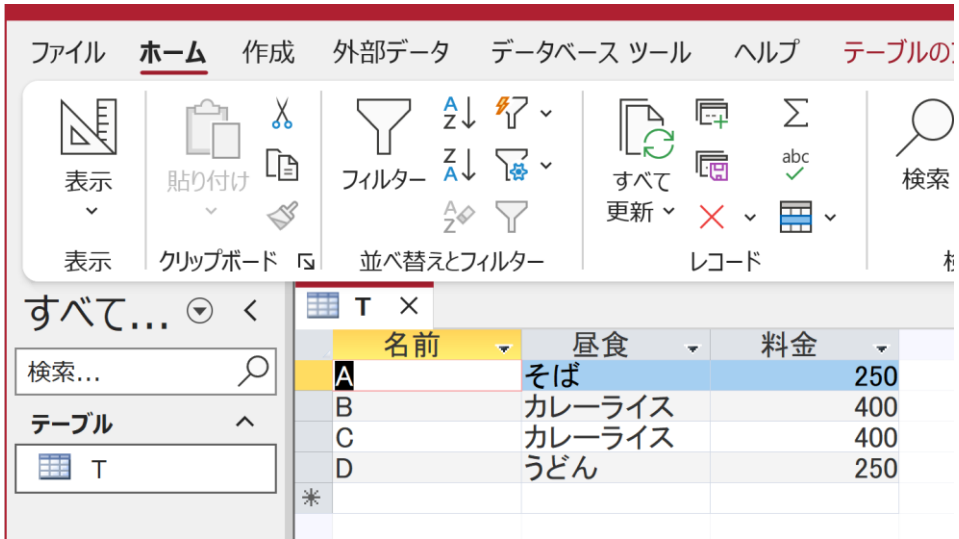
Access でのテーブルデータの確認

- SQL で確認

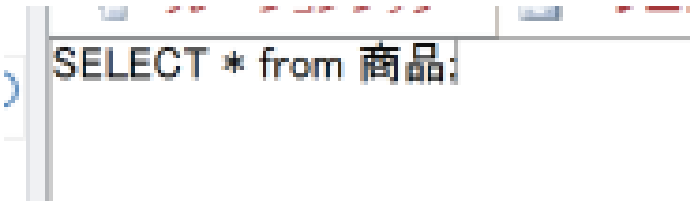
SELECT * FROM T;

	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
✳			

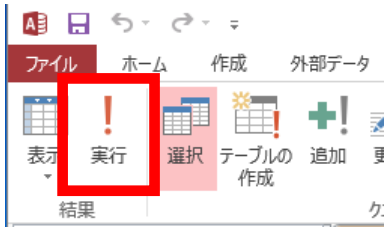
- **テーブルビューで、「テーブル名」をダブルクリック**



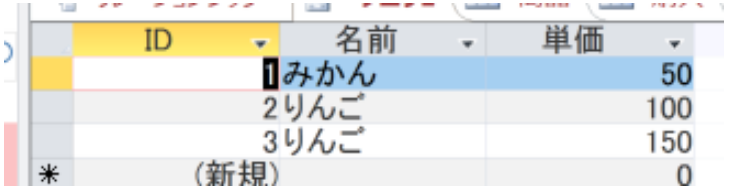
SQL 問い合わせ（クエリ）で使用する２つのビュー



SQL ビュー
SQL 文の**作成**、**編集**

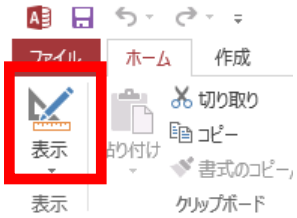


実行



	ID	名前	単価
	1	みかん	50
	2	りんご	100
	3	りんご	150
*		（新規）	0

データシートビュー
問い合わせ（クエリ）の結果



表示＋SQL ビュー

マウス操作でビューを切り替え

Access の SQL ビューを用いた問い合わせ

① Access の **SQLビュー**開く

② **SQL 文**の**編集。select, from, where** を使用

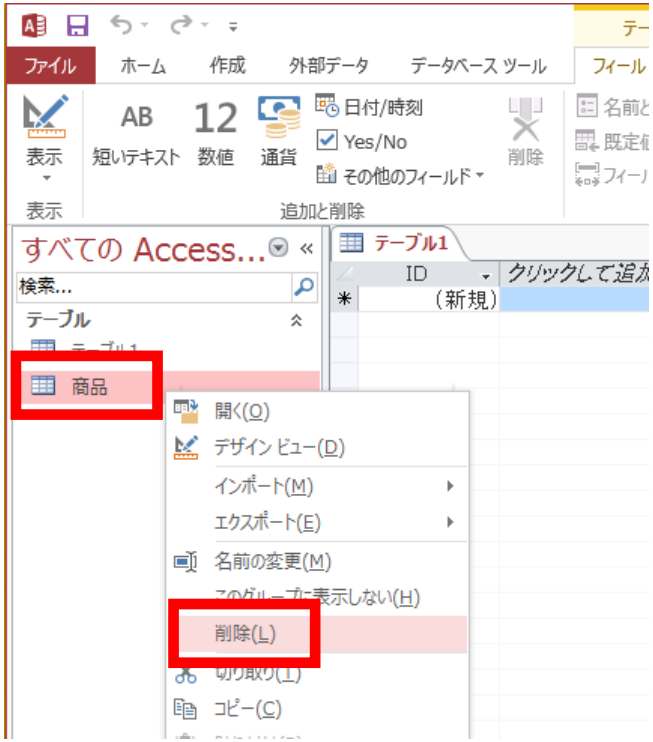
例: select * from テーブル名 where 列1 = 値1;

③ **SQL 文**の**実行**

実行の結果、**データシートビュー**に画面が変わり、そこに**問い合わせの結果**が表示される

④ さらにSQL 文の編集、実行を続ける場合には、**画面を SQL ビューに切り替える**

間違ってしまったときは、テーブルの削除を行ってからやり直した方が早い場合がある



テーブルビューで、削除したいテーブルを**右クリック**して、「**削除**」

テーブルを削除するときは、
間違って必要な**テーブル**を削除しないように、十分に注意する！
（元に戻せない）

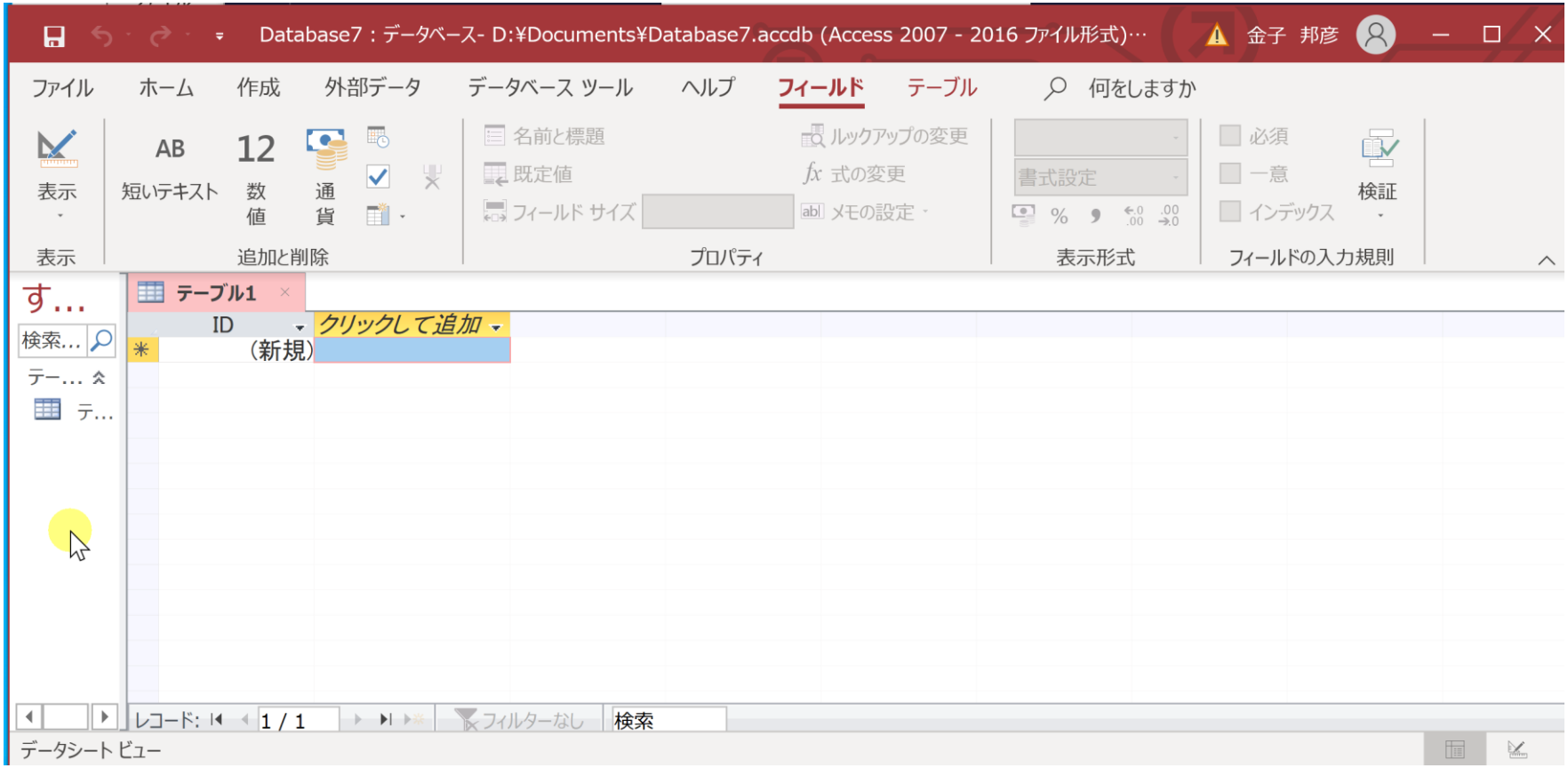
演習の手順

1. パソコンを使用する
   前もって Access をインストールしておくこと

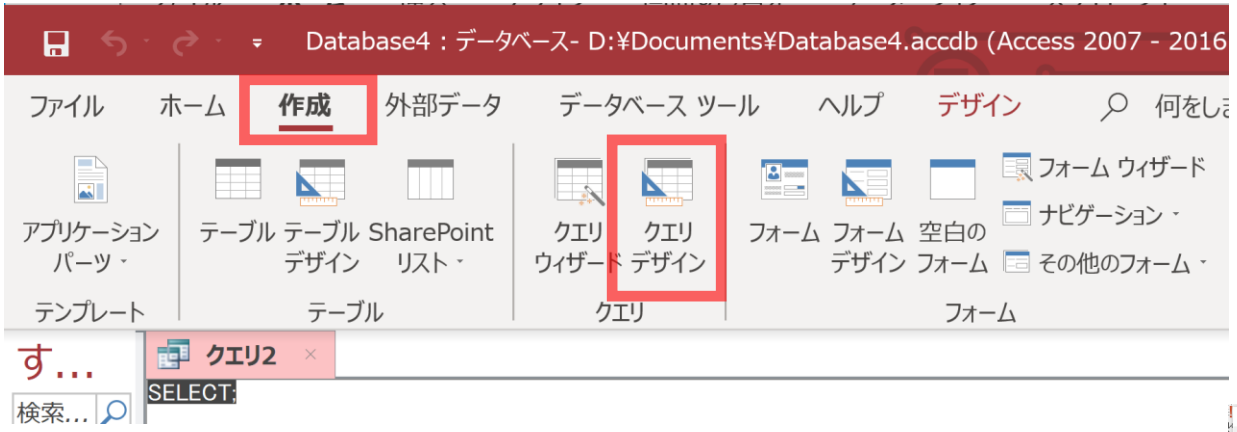
2. Access を起動する

3. Access で、「**空のデータベース**」を選び、「**作成**」をクリック.

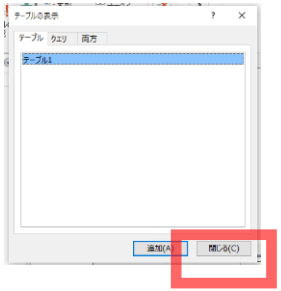
4. **テーブルツール画面**が表示されることを確認



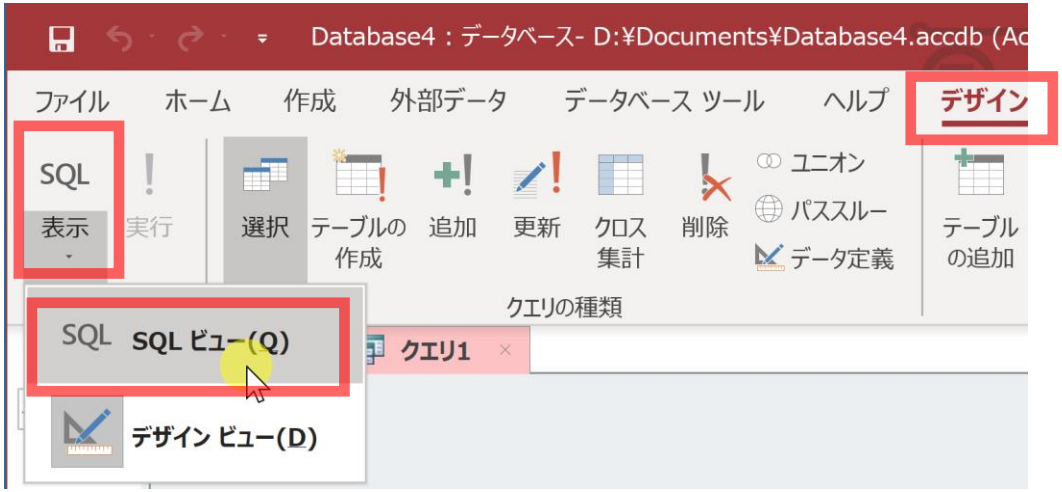
5. 次の手順で、**SQLビュー**を開く.



①「**作成**」タブで、「**クエリデザイン**」をクリック



このような表示が出たときは「**閉じる**」をクリック



②「**デザイン**」タブで、「**表示**」を展開し「**SQLビュー**」を選ぶ

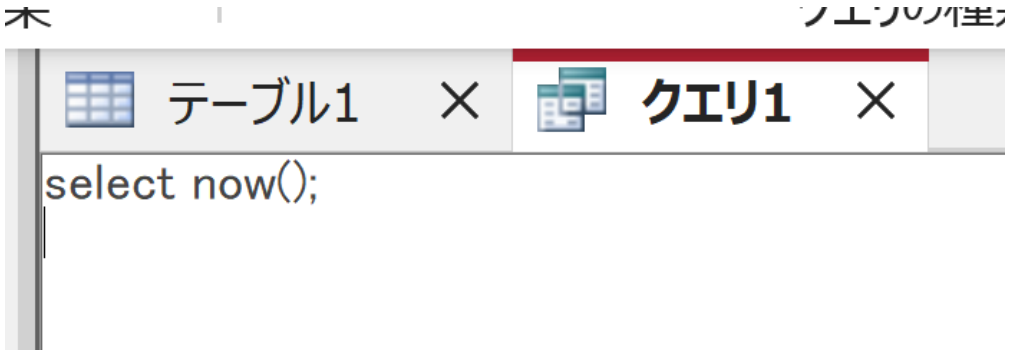
演習１．now() による現在日時の取得

【トピックス】

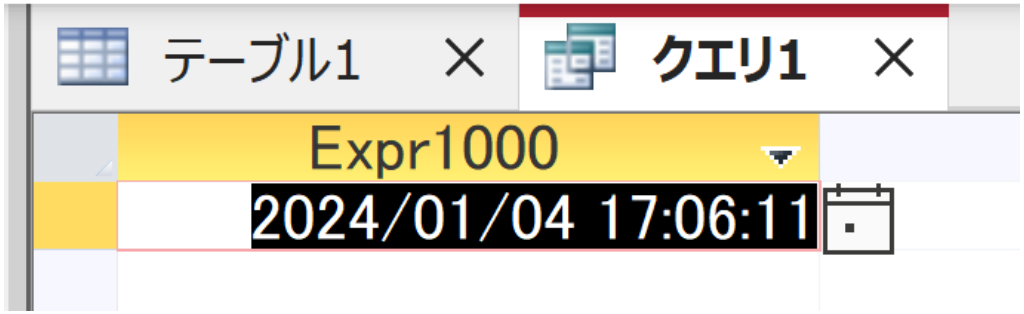
1. **now()**

SQL ビューに、次の SQL を入れ、「**実行**」ボタンで、**SQL 文**を実行．結果を確認

```
select now();
```



実行結果



表示の幅はマウス操作で変えることができる

演習２．日時を扱うテーブル

【トピックス】

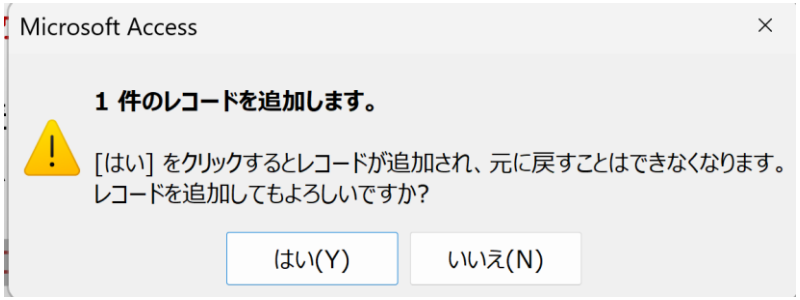
1. **日時**
2. **DATETIME**

① **SQL ビュー**に、次の SQL を１つずつ入れ、「**実行**」ボタンで、**SQL文**を実行．結果を確認

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

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));
```

② **SQL ビュー**に、次の SQL を１つずつ入れ、「**実行**」ボタンで、**SQL文**を実行．結果を確認

```
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 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');
```

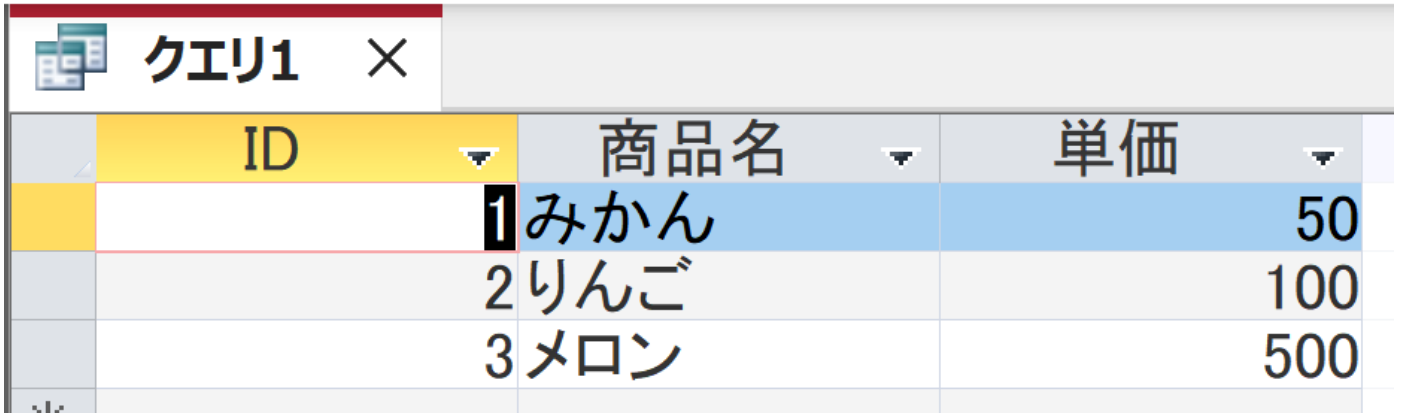


INSERT INTOでは、「実行」の後、確認表示が出る。その後、**画面が変化しない**が実行できている

③ **SQL ビュー**に、次の SQL を１つずつ入れ、「**実行**」ボタンで、**SQL文**を実行．結果を確認

```
select * FROM 商品;
```

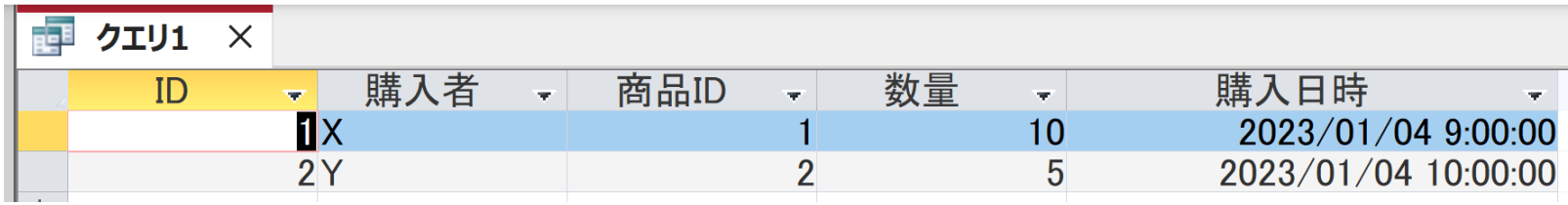
実行結果



④ **SQL ビュー**に、次の SQL を１つずつ入れ、「**実行**」ボタンで、**SQL文**を実行．結果を確認

```
select * FROM 購入;
```

実行結果



ID	購入者	商品ID	数量	購入日時
1	X	1	10	2023/01/04 9:00:00
2	Y	2	5	2023/01/04 10:00:00

演習３．データウエアは渦の構築

【トピックス】

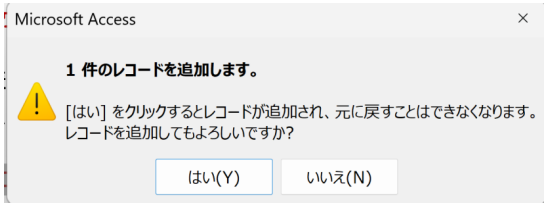
1. **日時**
2. **DATETIME**
3. **履歴データ**

① **SQL ビュー**に、次の SQL を１つずつ入れ、「**実行**」ボタンで、**SQL文**を実行．結果を確認

```
CREATE TABLE 商品 (
    ID INTEGER PRIMARY KEY,
    商品名 TEXT,
    単価 INTEGER,
    改訂日時 DATETIME);

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));
```

② **SQL ビュー**に、次の SQL を１つずつ入れ、「**実行**」ボタンで、**SQL文**を実行．結果を確認

```
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');

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');
```

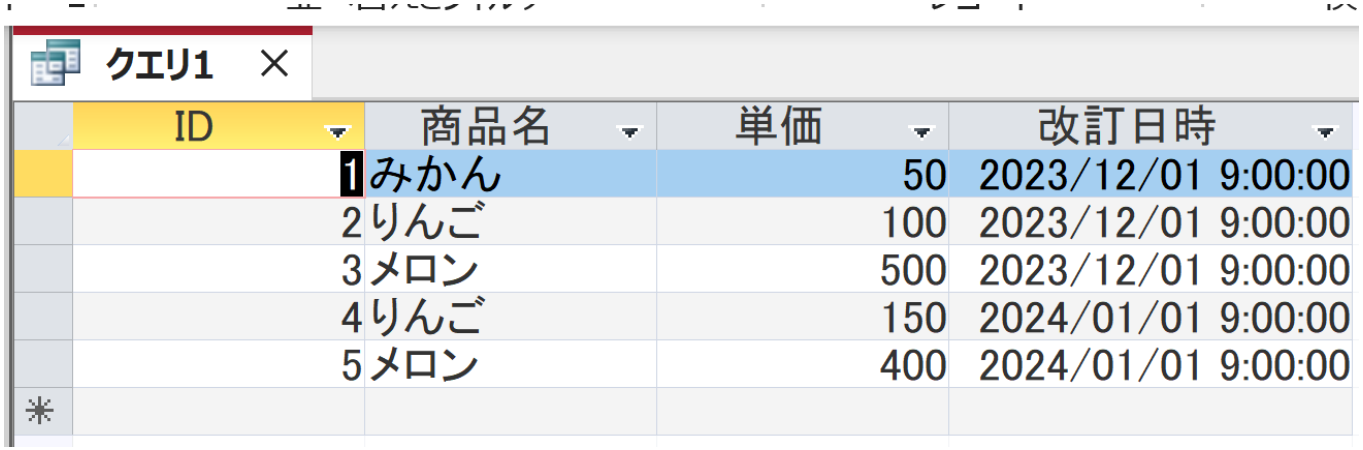


INSERT INTOでは、「実行」の後、確認表示が出る。その後、**画面が変化しない**が実行できている

③ **SQL ビュー**に、次の SQL を入れ、「**実行**」ボタンで、**SQL文**を実行．結果を確認

```
select * FROM 商品;
```

実行結果



表示の幅はマウス操作で変えることができる

④ **SQL ビュー**に、次の SQL を入れ、「**実行**」ボタンで、**SQL文**を実行．結果を確認

```
select * FROM 購入;
```

実行結果

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 9:00:00
4	Z	5	3	2024/01/05 11:00:00

⑤ **SQL ビュー**に、次の SQL を１つずつ入れ、「**実行**」ボタンで、**SQL文**を実行．結果を確認

結合

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

商品.ID	商品名	単価	改訂日時	購入.ID	購入者	商品ID	数量
1	みかん	50	2023/12/01 9:00:00	1	X	1	10
2	りんご	100	2023/12/01 9:00:00	2	Y	2	5
4	りんご	150	2024/01/01 9:00:00	3	Y	4	20
5	メロン	400	2024/01/01 9:00:00	4	Z	5	3

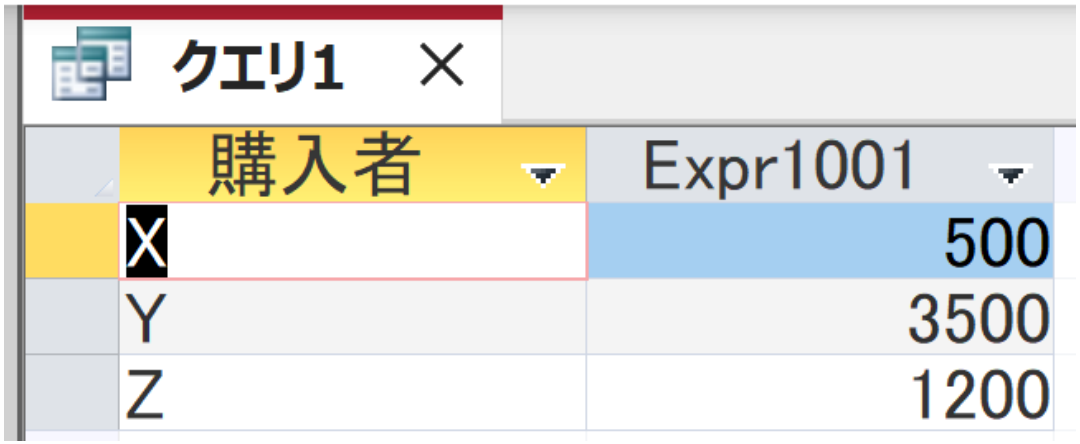
クエリ1

購入日時	購入者	Expr1002
2023/12/10 10:00:00		500
2023/12/20 12:00:00	Y	500
2024/01/05 9:00:00	Y	3000
2024/01/05 11:00:00	Z	1200

⑥ **SQL ビュー**に、次の SQL を入れ、「**実行**」ボタンで、**SQL文**を実行．結果を確認

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

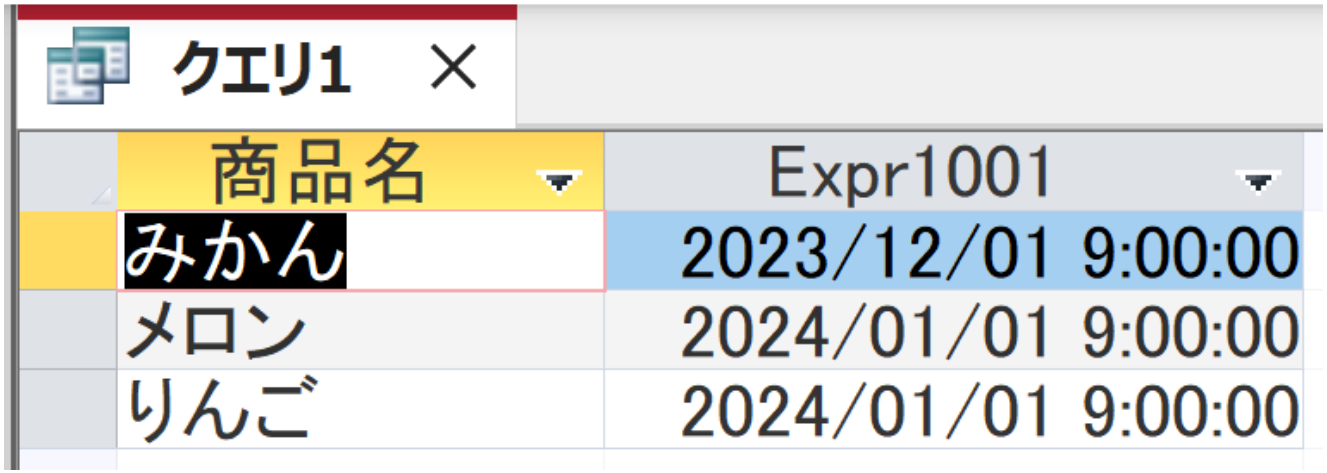
```
SELECT 購入.購入者, SUM(購入.数量 * 商品.単価)
FROM 商品
INNER JOIN 購入 ON 商品.ID = 購入.商品ID
GROUP BY 購入.購入者;
```



⑦ **SQL ビュー**に、次の SQL を入れ、「**実行**」ボタンで、**SQL文**を実行．結果を確認

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;
```

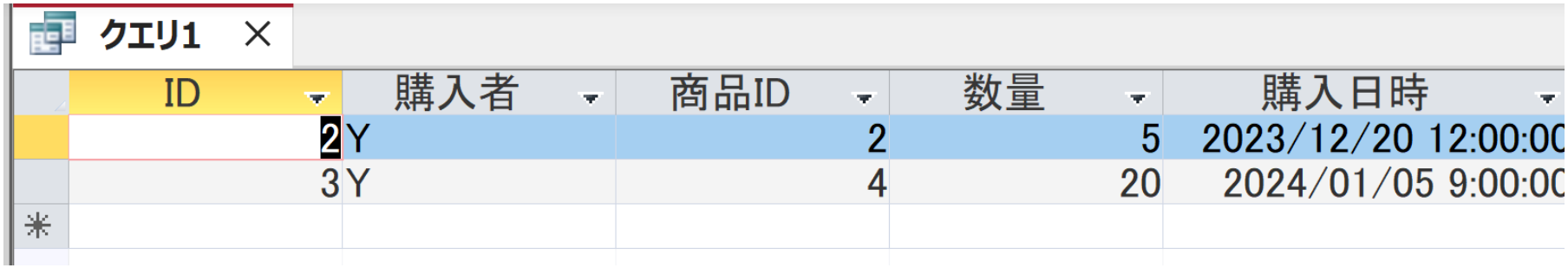


表示の幅はマウス操作で
変えることができる

⑧ **SQL ビュー**に、次の SQL を入れ、「**実行**」ボタンで、**SQL文**を実行．結果を確認

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

```
SELECT *
FROM 購入
WHERE 購入者 = 'Y';
```



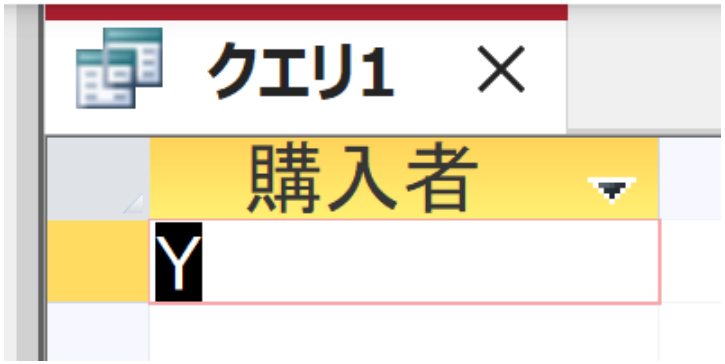
クエリ1

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 9:00:00

⑨ **SQL ビュー**に、次の SQL を入れ、「**実行**」ボタンで、**SQL文**を実行．結果を確認

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

```
SELECT DISTINCT(購入.購入者)
FROM 購入
INNER JOIN 商品 ON 購入.商品ID = 商品.ID
WHERE 商品.商品名 = 'りんご';
```



⑩ **SQL ビュー**に、次の SQL を入れ、「**実行**」ボタンで、**SQL文**を実行．結果を確認

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

```
SELECT 購入者, COUNT(*)
FROM 購入
GROUP BY 購入者;
```