授業内容（講義と演習）

1. スキーマの基本設計
2. 演習：スキーマの設計, 関係（テーブル）の作成と ERD の描画
3. リレーションの正規化
4. 演習：wineShop の正規化

I. 教科書第 4 章「データベース設計」の説明と理解

4.1 データベース設計の概要

データベースの設計は，概念設計，論理設計，物理設計の 3 段階により行われる

概念設計は，データベースによって管理の対象とするものを実世界から抽出する．実体 (Entity，エンティティ)，関連（relationship，リレーションシップ）によってモデル，E-R モデル(ER model)を作成する．

データ収集，調査（書類，帳簿，聞きこみ），データモデリングを行うこと．

論理設計は，概念設計によって作成された概念モデルを，特定のデータモデルに対応した論理モデルに変換する．E-R モデルに基づきデータ管理するリレーショナルモデル（relational model），すなわち，リレー

ション（表，テーブル，table）を作成してく．

専門用語

- 関係データベース（relational database，RDB）は，table(テーブル,表)より構成され，テーブルは関係ともいう．
- E-R モデルの属性（attribute）をテーブルの列（column）としてデータ型を決定し，表と列に対する制約の定義は論理設計で行う．テーブルの行（row，record）はモデルを構成するデータ（instance，インスタンス，実例）記録する．

物理設計は，論理設計において正規化した表の定義を，実装へ移行していくプロセスである．この段階でできたものは物理モデルと呼ぶ．それは実装環境とする基礎ソフトウエアのデータベース（MySQL, Oracle, PostgreSQL, SQL server, SQLite などなど）に依存するので，論理設計でできた表（テーブル，リレーション）を崩したり，修正したりすることが多い．

4.2 概念データモデルのモデリング技法

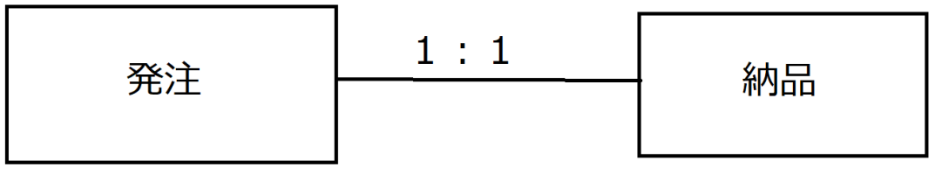
E-R モデルのカーディナリティ（Cardinality）について

カーディナリティとは，数学で基数あるいは濃度という意味の用語．IT の分野では，リレーショナルデータベースにおいてあるテーブルの同一の列（カラム,属性）に含まれる異なる値の数（バリエーション）のことを指すことが多い．

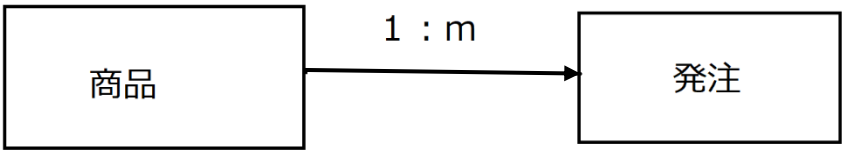
エンティティ（実体）の間のリレーションシップ（関連, instance）の対応関係のことを，カーディナリティという．

E-R におけるカーディナリティには，次の三つがある．

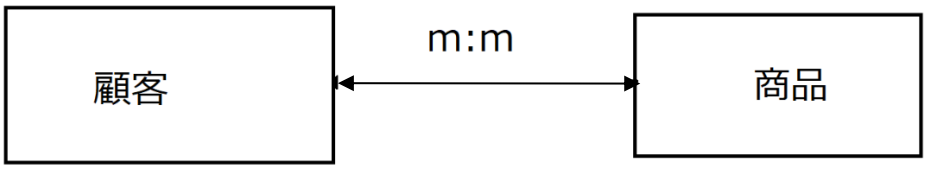
- １対１（1:1）：　「発注」と「納品」



- １対多（1:m か 1:n）：一つの商品が繰り返し発注された場合，「商品」と「発注」



- 多対多（m:m か n:n）：一人の顧客が多数の商品を購入し，一種類の商品は複数の顧客に販売される場合，「顧客」と「商品」



II.　関係（表，テーブル）スキーマの基本設計原則

- 設計原則

① 実体(entity)または関係(relationship)を表すテーブルの中の各属性（列名）が相互に独立性があり（互に無関係），推移関係がないようにデザインすること．唯一性がある主キー（primary key）を設けること．（第１正規形）

② 表と表の間に関係がある場合は外部キー（foreign key）でつながる．また，データ分類用に辞書類テーブルを用意しておいた方がよい．

③ 表のサイズは小さくにするか，大きくにするかがデザイン次第で，**データ不足または無駄の多いデザインをしないこと**．作る前の調査や聞き取り，作成した後のテスト・改良など徹底的にすること．

➢ キー

候補キー（candidate key）

リレーションの全属性集合の部分集合がそのリレーションのタプル（tuple，複数の構成要素からなる組）の一意識別の能力をもつような性質をもつ属性の極小組を候補キーという．

主キー（primary key）PK

プライマリーキー(主キー)とは，作成したテーブルの中の１つまたは複数のカラムの組み合わせに対して設定するもので，テーブルに格納されているデータを識別するための目印のようなものである．

【定義】キーの制約

主キーは次の条件を満たさなければならない。

(1)主キーはタプルの一意識別能力を備えていること

(2)主キーを構成する属性は空（NULL）をとらないこと

但し，MySQL では，一つのリレーション（テーブル）に１つの主キーしか定義できない．次のようにやってみたら，エラーがでた．

```
mysql> create table test(
    -> id int primary key not null auto_increment,
    -> email varchar(100) primary key not null,
    -> password char(10),
    -> status char(8));
ERROR 1068 (42000): Multiple primary key defined
```

外部キー（foreign key）FK

外部キーとは，リレーショナルデータベース（RDB）で，テーブルのある列に，別のテーブルの特定の列に含まれる項目しか入力できないようにする制約．関連したテーブル間を結ぶために設定する列のことで，データの整合性をデータベースに保証させるために利用できる．

【定義】(外部キー制約)

リレーション R(…,Ai,…)の属性 Ai がリレーション S(B1,…)に関する外部キーであるならば，R の任意のタプル t に対して，t[Ai]は空であるか，そうでない場合には S にあるタプル u が存在して，t[Ai]＝u[B1] でなければならない，ここに，t[Ai]＝u[B1]はそれぞれ t と u の Ai 値と B1 値を表す．

すなわち，R の外部キーと参照先とする S の主キーは,名前が一致なくてもよいが，この二つのキーの型と値は一致しなければならない.

MySQL では，主キーと外部キーを表１のように示す.

表１　MySQL におけるリレーション及びそのキーの表示

```
mysql> desc locality;
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| locaID   | int          | NO   | PRI | NULL    | auto_increment |
| locality | varchar(100) | YES  |     | NULL    |                |
| country  | varchar(30)  | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
mysql> desc status;
+---------+-------------+------+-----+---------+----------------+
| Field   | Type        | Null | Key | Default | Extra          |
+---------+-------------+------+-----+---------+----------------+
| stID    | int         | NO   | PRI | NULL    | auto_increment |
| status  | varchar(20) | YES  |     | NULL    |                |
| forwhat | varchar(20) | YES  |     | NULL    |                |
+---------+-------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
mysql> desc wine;
+---------+--------------+------+-----+---------+----------------+
| Field   | Type         | Null | Key | Default | Extra          |
+---------+--------------+------+-----+---------+----------------+
| wineID  | int          | NO   | PRI | NULL    | auto_increment |
| name    | varchar(100) | YES  |     | NULL    |                |
| locaID  | int          | YES  | MUL | NULL    |                |
| kind    | varchar(32)  | YES  |     | NULL    |                |
| color   | varchar(8)   | YES  |     | NULL    |                |
| vintage | int          | YES  |     | NULL    |                |
| price   | int          | YES  |     | NULL    |                |
| picture | varchar(150) | YES  |     | NULL    |                |
| comment | text         | YES  |     | NULL    |                |
| stID    | int          | YES  | MUL | NULL    |                |
```

```
+---------+-------------+------+-----+---------+----------------+
10 rows in set (0.00 sec)
```

III. ワインショップ（wineShop）のデザイン例

テーブル名（属性名 1，属性名 2,…）

１．顧客管理用スキーマ（schema，構造）設計とテーブル作成

顧客管理は EC サイトにおいて極めに重要であり，アクセスやリスポンスのスピードとセキュリティに応じるデザインをケースバイケースで考えてみよう．

一般に，EC サイトに登録時の ID とするメールアドレスを主キーに使われることが多くなっており，これはメルアドの重複排除と顧客一人ずつ異なる ID（email）を配分するとのデータベースをデザインする側の考えである．また，セキュリティ用に ID 登録と同時にパスワードも設定しなければならない．

非正規形（概念設計）
顧客(メールアドレス，パスワード，パスワード更新日時,顧客 ID,名前(firstname，lastname),電話番号(自宅，スマホ),住所（郵便番号，住所（国，都道府県，市，区，町，番地,アパート名，番号)），状態，登録日時)

案①のスキーマ設計・テーブル作成・ERD 描画
データベースの基本デザイン原則として，一つの表は 255 列以内に互いに独立する属性を設けることできる．ID・パスワードと顧客の個人情報を一緒に納めるデザイン例は案①に示す．

スキーマ：
顧客(メールアドレス，パスワード，パスワード更新日時,顧客 ID，firstname，lastname,電話番号,郵便番号，住所,状態,登録日時)
customer(email，password，pwUpdateTime，customerID，firstName，lastName，phone，postcode，address，status,regDateTime)

一般プログラミング方法で使われている変数の型定義は，デー

タベースの属性にも適用されている．整数型，実数型，文字型，テキスト型，日時型などある．スキーマ設計とテーブル作成においては，各属性はどの型で定義するのか考えなければならない．

E-R モデル（実体関係モデル）を ERD で可視化し表現することは，抽象的スキーマ設計を見えるようにし，できたデータベースの検討・評価・改良につながる．ERD とは，Entity Relationship Diagram，情報実体関連図のことで，業務を構成する「情報」がデータベースで管理されている場合，「情報」の中で示される「ひと，もの，かね」等がデータベースの中でどのような形で表現されているかについて示すための図である．

近年，ERD の描画は従来の手書き描画からオンライン ERD 描画ツールの利用へ切替られ，労力軽減と質の向上になった．但し，これらのツールは，入力データとして SQL 命令によるテーブルの定義が使われる．それて，先に MySQL のテーブル作る命令を習ってから，ERD の描画方法を学びしよう．

MySQL の create table でのテーブル作成方法
①Mysql 会話式命令での作成
②sql ファイル xxxx.sql を Mysql の外側であらかじめ作成しておき，Mysql にログインしたあと，source xxxx.sql でテーブルを作成する．

手順

ステップ 1.メモ帳などで sql ファイルを作成する．
ステップ２．ブラウザーにて dbdiagram.io を利用して ERD を描画し，スキーマを検討・改良など繰り返しする．
ステップ 3.sql ファイルを Linux サーバーにアップロードする．MySQL による実装し，テーブルを作る．

テーブル作る sql ファイルの作成と ERD 描画の順で，今から説明していく．
まず，メモ帳で下記のテーブル作成する MySQL 命令を書き，customer.sql として保存する．コマ「,」で区切りして属性名　型名の順で書く．主キーの書き方が間違いのないように．

customer.sql のコード:

```
create table customer(
     email varchar(150) primary key not NULL,
     password varchar(20),
     pwUpdateTime datetime,
     customerID varchar(10),
     firstName varchar(30),
     lastName varchar(16),
     phone varchar(20),
     postcode char(8),
     address varchar(300),
     status char(10),
     regDateTime datetime);
```

次に，ブラウザーにて dbdiagram.io を利用して ERD を描画する．ログインした後，import で MySQL を，Upload sql で事前に準備した sql ファイルを選び，Submit で指定した sql ファイルを送信する．レイアウトを調整して，Export で描画された ERD を PDF か png としてダウンロードする．

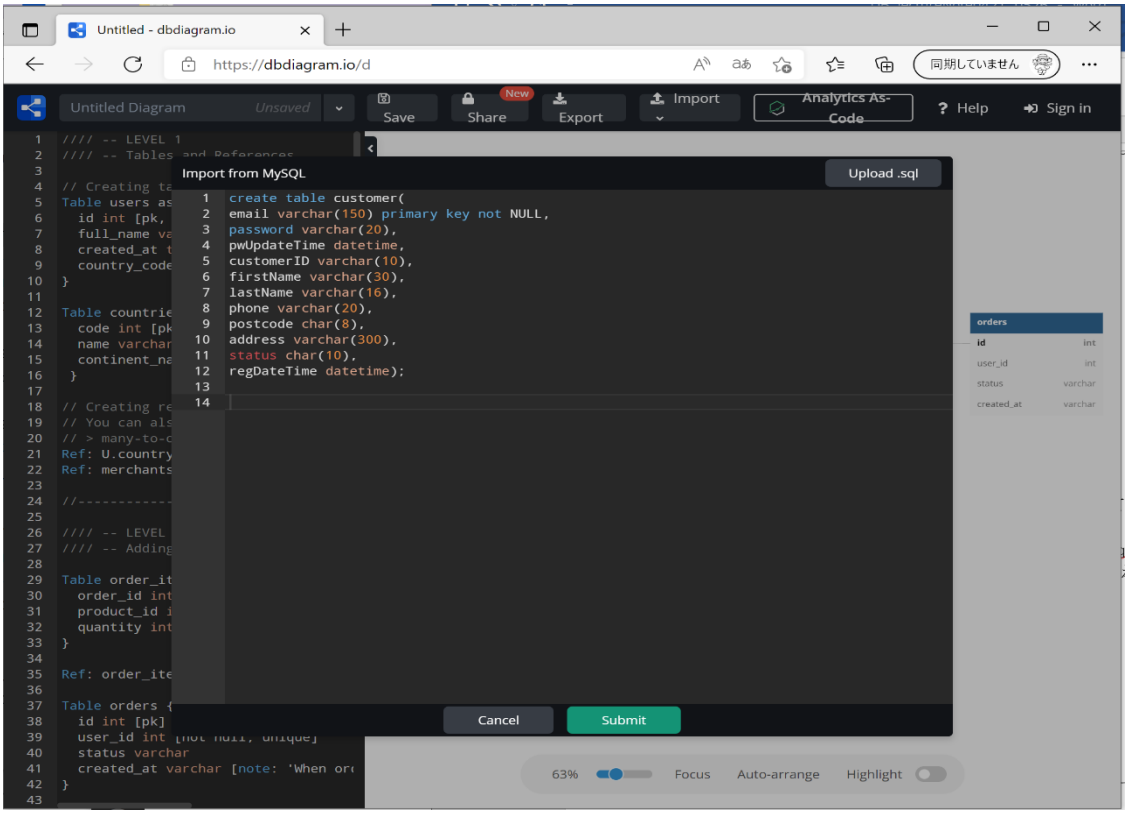


図１　dbdiagram.io での MySQL コードをアップロードする画面

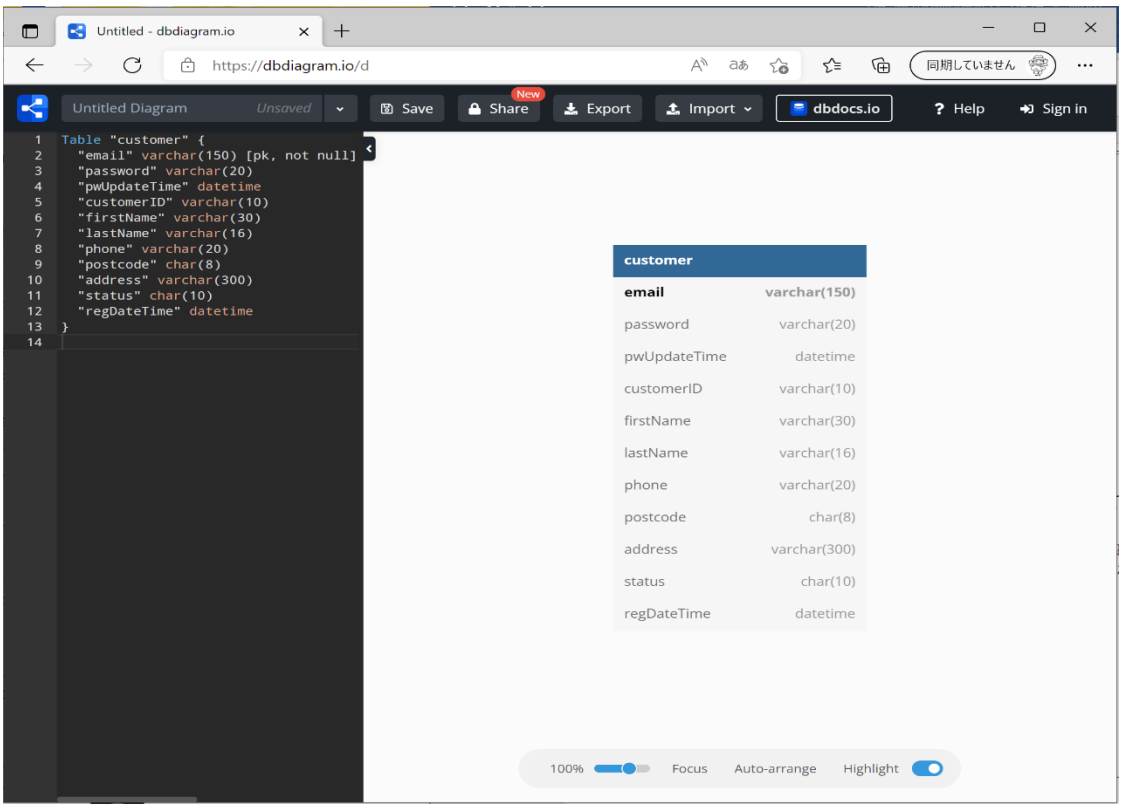


図２　dbdiagram.io での ERD を描画する画面

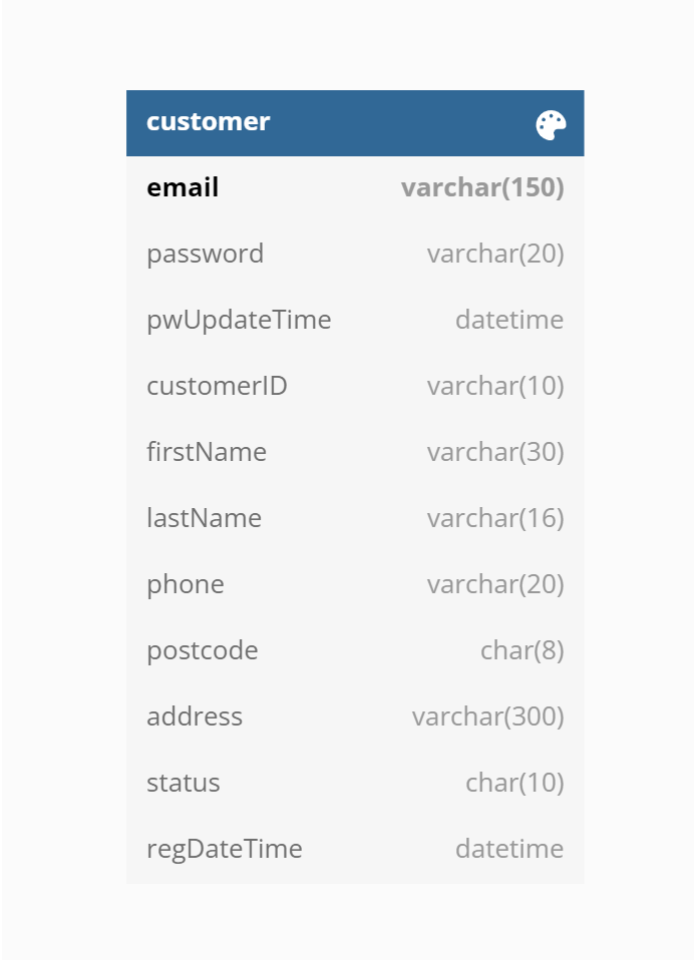


図３　描画された customer ERD

案②のスキーマ設計・テーブル作成・ERD 描画

顧客の登録 ID（email）とパスワードを機密情報として顧客の個人情報に分けて，別に管理することはセキュリティ対策としての考えもある．

この場合は，二つのテーブルを設け，それぞれの主キーに email を使い，これは「一つのメールアドは一つの顧客に対応する」というデータベースの「1 対 1」設計方法である．なお，MySQL での「１：１」の関係を ERD で表現する際，二つのテーブルが描かれるが，テーブルの間につながり線は表示しない．また，二つのテーブルを作成しデータを入力する際に，一人の客に対してそれぞれのテーブルに同じ email データを入力しなければならないことを注意してほしい．

スキーマ：
idpw(メールアドレス, パスワード, パスワード更新日時)
顧客(メールアドレス, 顧客 ID, firstname, lastname,電話番号,
郵便番号, 住所,状態,登録日時)

idpw(email, password, pwUpdateTime)
customer(email, customerID, firstName, lastName, phone,
postcode, address, status, regDateTime)

customer_1to1.sql コード:

```
create table idpw(
   email varchar(150) primary key not NULL,
   password varchar(20),
   updateTime datetime);

create table customer(
   email varchar(150) primary key not NULL,
   customerID varchar(10),
   firstName varchar(30),
   lastName varchar(16),
   phone varchar(20),
   postcode char(8),
   address varchar(300),
   status char(10),
   regDateTime datetime);
```

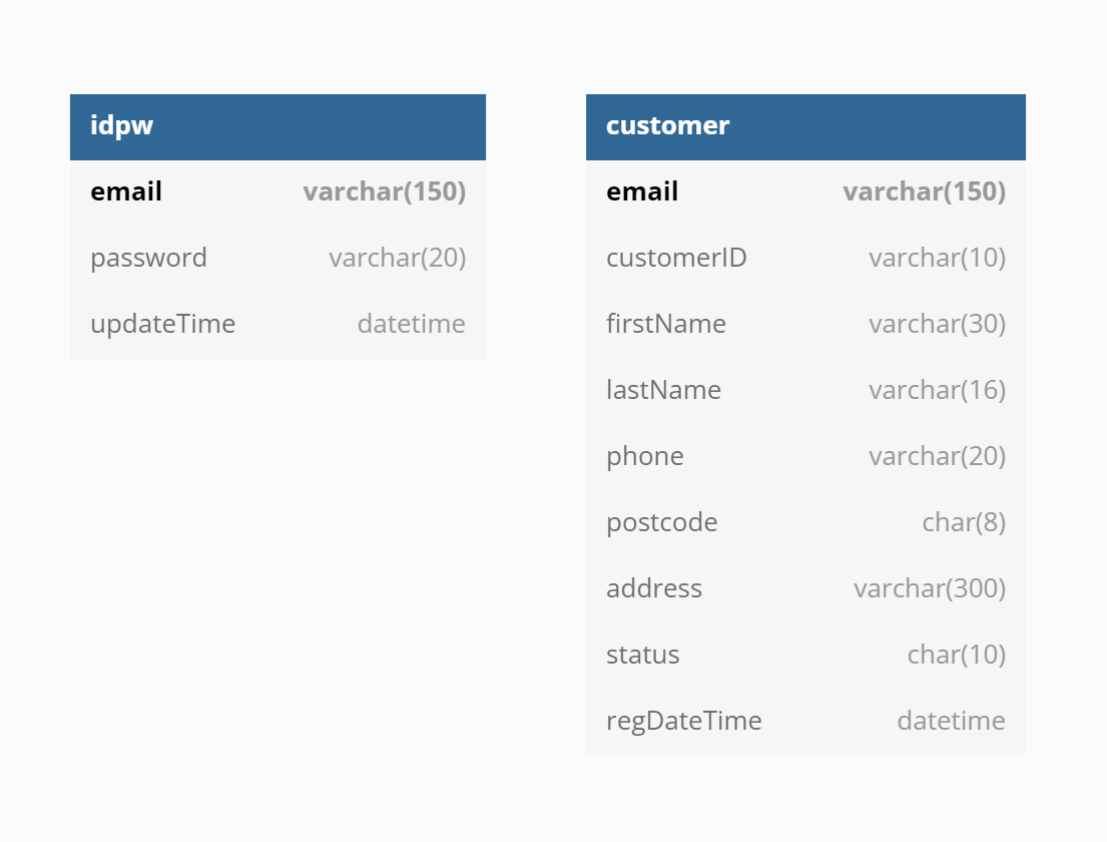


図４　idpw と customer(1:1) ERD

案②の ERD を見ると，セキュリティ対策と言って，テーブルを増えるだけで，案①のテーブルを二つに分けたメリットがないことが明らかにしている．

案③のスキーマ設計・テーブル作成・ERD 描画

主キーの１対多数（1:m）のデザインはデータベース設計上よく採用された方法である．案②の 1:1 のような方法は，1:m の特例として使う考えもある．

顧客の登録 ID（email）とパスワードを機密情報として顧客の個人情報に分けて，二つのテーブルを設ける．案②と違った点として，テーブル idpw の主キーに email を使い，これは重複するメルアドが作らせないために重要である．一方，顧客個人情報管理するテーブル customer には，主キーに顧客 ID を使い，email は外部キーとして idpw の email に参照する．これは注文関係などをデザインするときに，顧客 ID は email を使うより，顧客が独自のコード（会員コード，店用カードのバーコード，マイナンバー，EC サイ

ト登録時発行された顧客コードなど）を主キーとして使うケースは多いからである．MySQL での「１：m」の関係を ERD で表現する際，二つのテーブルとテーブルの間につながり１：＊の線が描かれる．

しかし，このデザイン案には大きな落とし穴がある．テーブル customer にある多数の客は同じ email に参照することが許してしまう可能性がある．セキュリティ上危険であるため，二つのテーブル idpw と customer を作成した後，データを入力する際，idpw にある既存 email データを customer の email の入力データとして２度と使わせないように，1:1 のようなデータを入力する対策を講じなければならない．

スキーマ：
idpw(メールアドレス，パスワード，パスワード更新日時)
顧客(顧客 ID，firstname，lastname，メールアドレス，電話番号，郵便番号，住所，状態，登録日時)

idpw(email，password，pwUpdateTime)
customer(customerID，firstName，lastName，email，phone，postcode，address，status，regDateTime)

customer_1tom.sql コード

```
create table idpw(
     email varchar(150) primary key not NULL,
     password varchar(20),
     updateTime datetime);

create table customer(
   customerID varchar(10) primary key not NULL,
   firstName varchar(30),
   lastName varchar(16),
   email varchar(150),
   phone varchar(20),
   postcode char(8),
   address varchar(300),
```

```
status char(10),
regDateTime datetime,
foreign key (email) references idpw(email));
```

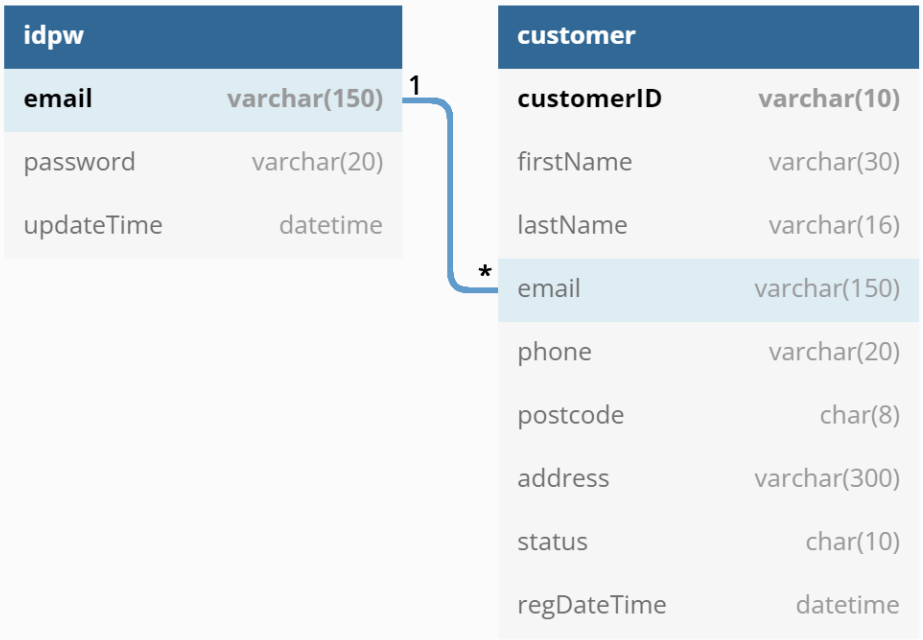


図５　idpw と customer（1:m）　ERD

案④のスキーマ設計・テーブル作成・ERD 描画

案③の考え方を改め，一人の顧客が自分のメルアドしか使えない，一人が複数のメルアドを作成できるため，２段認証にも対応できる．idpw と customer の二つテーブルを設ける．

スキーマ：

顧客(顧客 ID，firstname，lastname,電話番号,郵便番号，住所,状態,登録日時)

idpw(メールアドレス，パスワード，顧客 ID，パスワード更新日時)

customer(customerID，firstName，lastName，phone，postcode，address，status，regDateTime)

idpw(email，password，customerID，pwUpdateTime)

customer_idpw_1tom.sql コード

```
create table customer(
    customerID varchar(10) primary key not NULL,
    firstName varchar(30),
    lastName varchar(16),
    phone varchar(20),
    postcode char(8),
    address varchar(300),
    status char(10),
    regDateTime datetime);

create table idpw(
    email varchar(150) primary key not NULL,
    password varchar(20),
    customerID varchar(10),
    updateTime datetime,
    foreign key (customerID) references
customer(customerID));
```

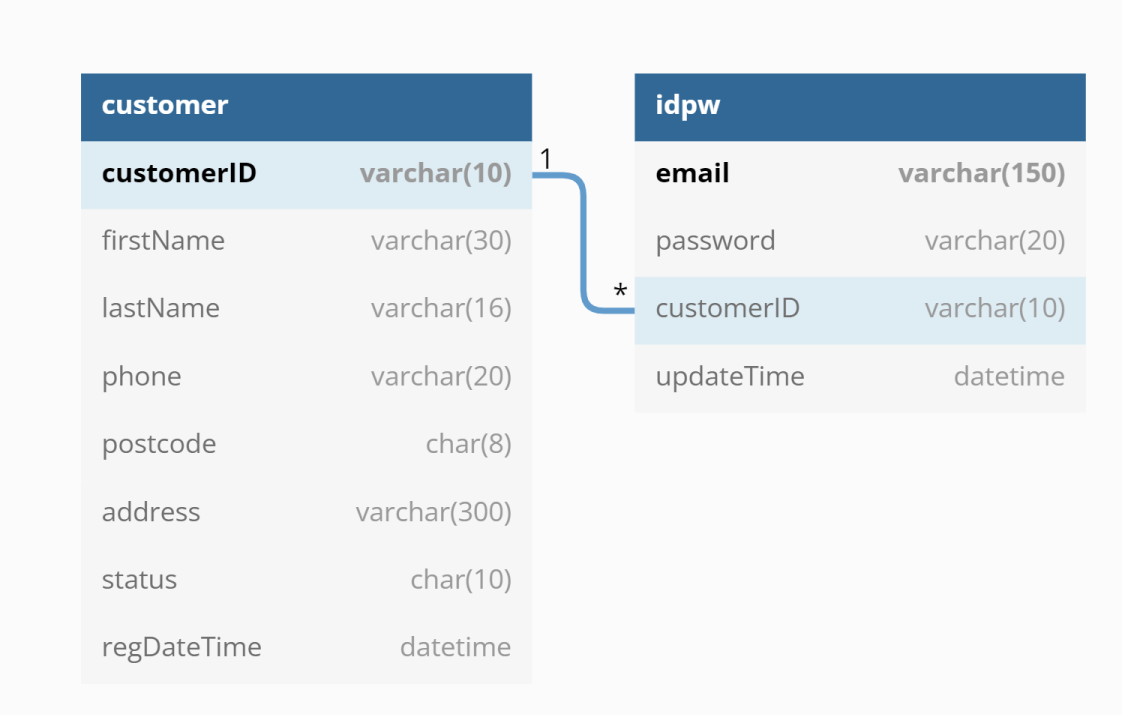


図６　顧客管理 ERD

このデザイン案は，顧客情報 customer を先に定義した後，認証管理 idpw を定義するため，物理設計では，先にテーブルの customer に顧客コードをデータベースに作ってもらったあと，顧客個人情報を入力して登録する．その後，テーブル idpw にメールアドレス，パスワード，顧客 ID を入力することは，EC サイトの認証・登録の順序と逆になっていることに留意しなければならない．よって，データベースと連携する EC サイトを設計するときに，データベースの論理設計と物理設計について詳しく把握することは必要不可欠である．

情報資産を保有するデータベースは機密性（Confidentiality），完全性(Integrity)，可用性(Availability)をバランスよく保ちながらセキュリティ対策を講じるべきである．

この授業では，案④を採用し実装してみる．

III. 教科書第５章　リレーションの正規化の説明と理解

リレーションの正規化とは，データベースで保持するデータの冗長性（じょうちょうせい）を排除し，一貫性と効率性を確保するためのデータ形式へ変換する作業のこと．一般的に第 3 正規形までで十分とされているため，第 3 正規形までを取り上げる．

- 非正規形
  属性値に繰り返しをもつような属性が存在するリレーション（テーブル，表）のこと．
  ① 属性の定義域が配列，集合，リストのような繰り返し項目をもつ場合
  ② 属性が複数項目の組のような複合項目の場合
  ③ 属性が複合項目の繰り返しになっている場合

非正規形をなくす作業は正規化である．

- 第１正規形
  リレーションが単純な定義域上で定義されていること．「1 つのセルには１つの値しか含まれない」とのこと．
- 第２正規形
  ① リレーション R が第１正規形であり，かつ

② リレーションＲのすべての非キー属性がＲの候補キーに対して完全関数従属であること．すなわち「部分関数従属を排除し，完全関数従属にする」こと．

- 第３正規形
  ① リレーションＲが第２正規形であり，かつ
  ② リレーションＲのすべての非キー属性がＲのどの候補キーに対しても推移的関数従属でない．　「第２正規形のテーブルから，推移的関数従属している列が切り出されたもの」である．

正規化のメリットとしては，データを一元管理し，データの整合性を保ちやすくなる．

IV.　商品（ワインとワインセット）管理の正規化及びスキーマ設計とテーブルの作成

1.非正規形

(1)　wine

wine(wineID，名前，産地（国名，産地名），品種，色，ビンテージ，価格，画像ファイル名，コメント，状態）

(2)　wineSet

wineSet(setID,名前，セットの構成（(wineID,本数)1,(wineID，本数)2，…，(wineID，本数）n ），価格，コメント，状態）

2.スキーマの設計と正規化　　主キー　　外部キー

- ✧ 生産地（産地 ID，産地名，国名）
- ✧ ワイン状態（状態 ID,状態,対象）
- ✧ ワイン（ワイン ID,名前，産地 ID，品種，色,ビンテージ,

価格，画像ファイル名, コメント, 状態 ID)

- ✧ ワインセット（セット ID，名前, コメント，状態 ID)
- ✧ ワインセット詳細(セット ID, ワイン ID，ワインの本数)
- ✧

locality(locaID, locality, country)

wstatus(stID,status,forwhat)

wine(wineID, name, locaID, kind, color, vintage, price, picture,comment,stID)

wineSet(setID, name,comment,stID)

setDetail(setID, wineID, quantity)

＊外部キーの参照先は，先に定義しなければならない順序に注意.

上記の商品（wine）管理のデザインは典型的なデータベースデザインとなっており，リレーションは第３正規形まで正規化された.

テーブル locality, status,wine,wineSet は実体（Entity）である. locality と status は辞書の代わりに定義され，wine, wineSet の外部キーによりそちらの主キーと互いに参照され，無駄のない産地名や在庫状態を記録できる.

setDetail は wine と wineSet の関係（relationship）を示すテーブルで，外部キーで二つの表に参照している.

3. ワイン管理用５つのテーブルの定義

```
create table  locality(
  locaID int primary key not null auto_increment,
  locality varchar(100),
  country varchar(30));
```

```
create table wstatus(
  stID int primary key not null auto_increment,
  status varchar(20),
  forwhat varchar(20));

create table wine (
 wineID int auto_increment not null primary key,
 name varchar(100),
 locaID int,
 kind varchar(32),
 color varchar(8),
 vintage int,
 price int,
 picture varchar(150),
 comment text,
 stID int,
 foreign key(locaID) references locality(locaID),
 foreign key(stID) references wstatus(stID));

create table wineSet(
 setID varchar(20) not null primary key,
 name varchar(32),
 comment text,
 stID int,
 foreign key(stID) references wstatus(stID));

create table setDetail(
 setID varchar(20),
 wineID int,
 quantity int,
 foreign key(setID) references wineSet(setID),
 foreign key(wineID) references wine(wineID));
```

4.ERD の描画

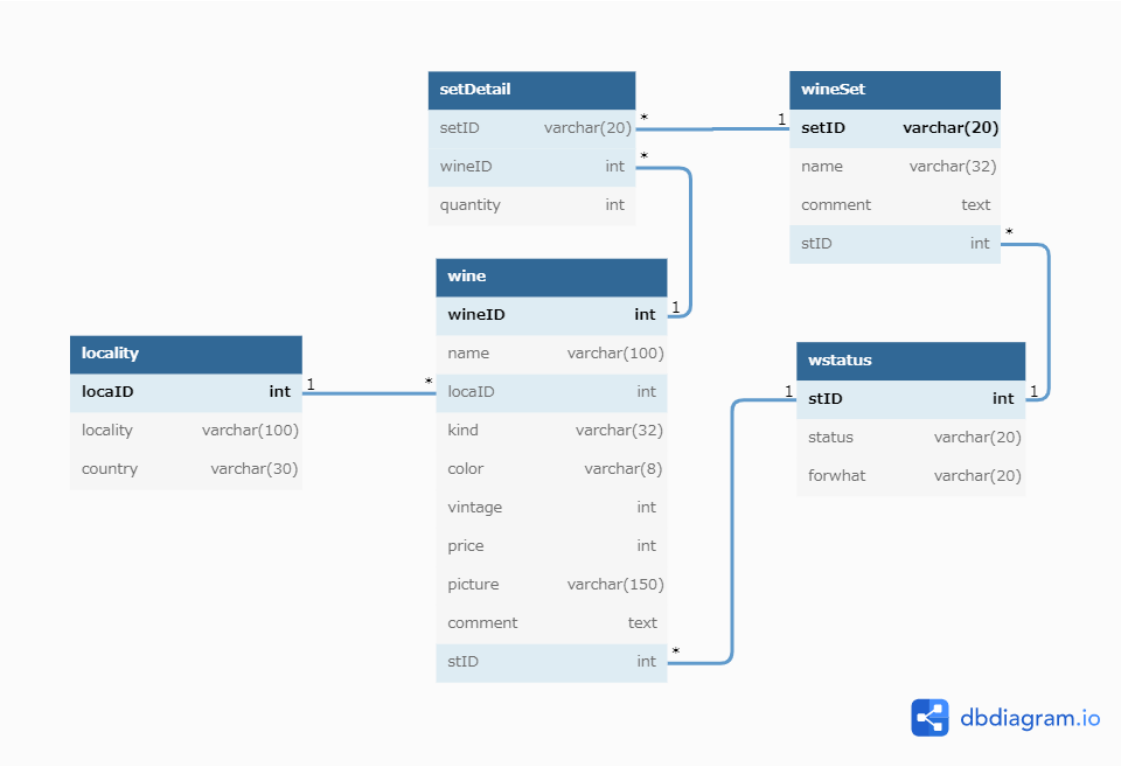


図 7　商品(wine)管理 ERD

演習課題・前期中間実技試験課題

- 課題 1.　授業内容を理解した上で，WineShop スキーマの設計，7 つテーブルの作成と ERD の描画を行ってください．

  使用するツール：　https://dbdiagram.io/

  ソース：　wineShop.sql

提出物：WineShop ERD の PDF か画像をメール添付による提出してください．

提出先：sningping@kumamoto-nct.ac.jp

- 課題 2.　教科書演習課題 4.4

  LibraryReservation(図書館予約)スキーマの設計，テーブルの作成，ERD の描画を行ってください．

  (1)　提案されたデータモデルとスキーマが適切かどうかを検討する．

  (2)　提案されたスキーマに基づいて，実装できるスキーマ(英字)に変換する．主キー，外部キーの記法に注意して下さい．

(3)　(2)でできたスキーマに基づきテーブルを定義して下さい.

(4)　ERD を描画する.

提出物:LibraryReservation ERD の PDF か画像をメール添付による提出してください.

提出先:sningping@kumamoto-nct.ac.jp