"PERSEMBLE"

PERSONAL ENSEMBLE DATABASE

OUTLINE

An individual's wardrobe and beauty products are of importance, as stated by Rachel Zoe, "Style is a way to say who you are without having to speak." As important as these items are in constructing ones style, they tend to end up in a state of array. Shoes end up forgotten at the back of the closet and that perfect shade of lipstick is buried under a bag of Lush bath bombs. Paralleling physical organization, simplification in your personal ensemble, or "Persemble," allows you to fully utilize your wardrobe and beauty products.

Persemble is a HTML, PHP, JavaScript, AJAX, and MySQL database-driven website that offers wardrobe and beauty organization and simplification. Leonardo de Vinci said, "Simplicity is the ultimate form of sophistication." By knowing what you have and in what quantity in both your wardrobe and beauty items, you can decrease waste in avoiding duplicate product purchases and increase the implementation of the items you already have.

DATABASE OUTLINE

Persemble is a database that represents a person's wardrobe closet, as well as their powder room item menagerie. The database concept is straightforward, but the relative complexity of category and subcategory possibilities makes this highly practical candidate for a database.

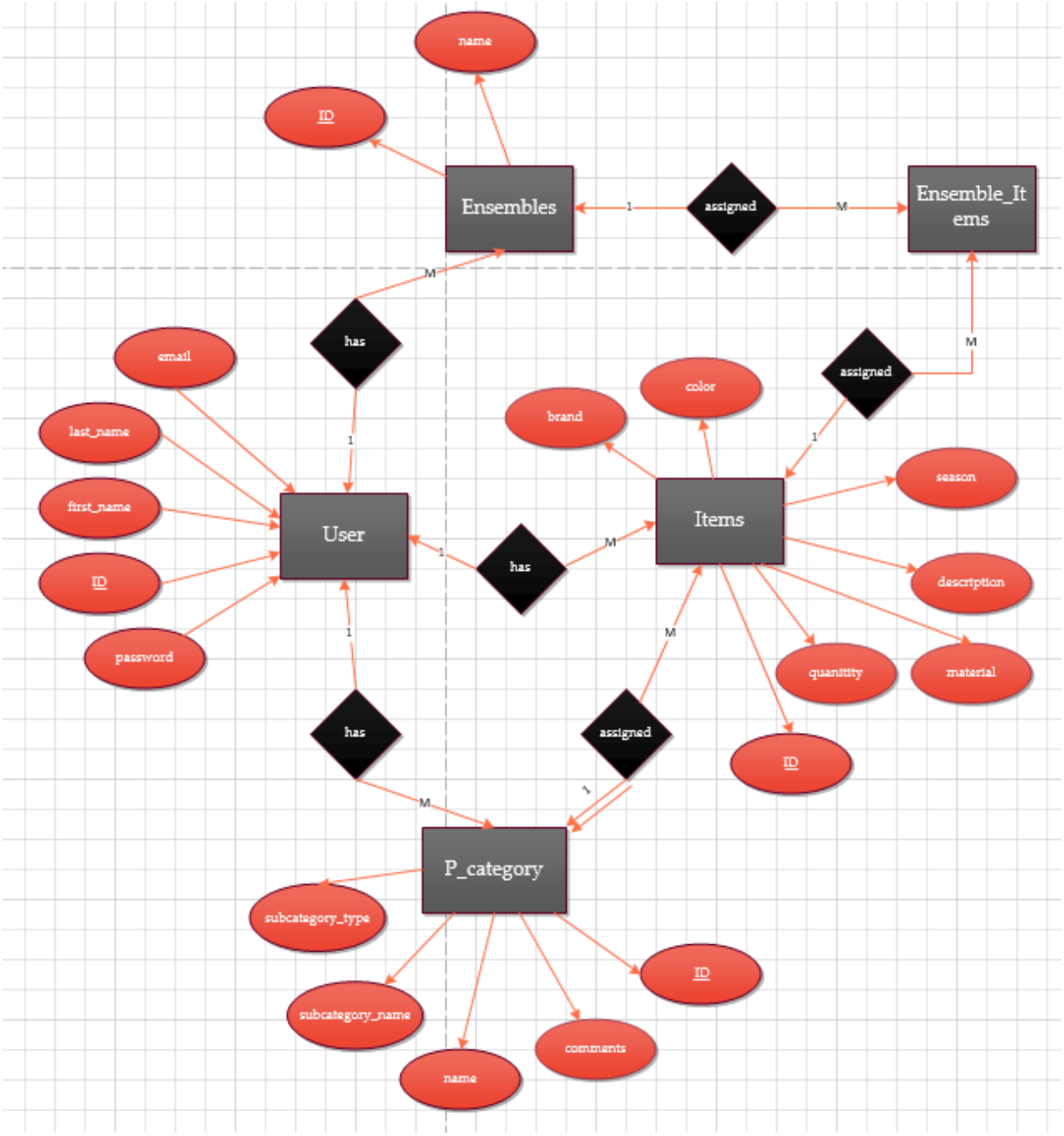
Entities

- User
- Items
- P_category
- Ensembles
- *Ensemble_Items*

Relationships

- A User *has* an Ensembles, Items, and P_category. It is possible for the user to have zero-to-many of Ensembles, Items, and P_category, and the entities, in return, can have one User.
- P_category is assigned to Items. This is a one-to-many relationship, Items must be assigned a P_category and a P_category may be assigned zero-to-many Items.
- Items are assigned to an Ensemble. The relationship between these two entities is a many-to-many, items can be assigned many Ensembles and Ensembles can be assigned many Items. The entity Ensemble_Items represents this many-to-many relationship.

ER DIAGRAM OF DATABASE



DATABASE SCHEMA

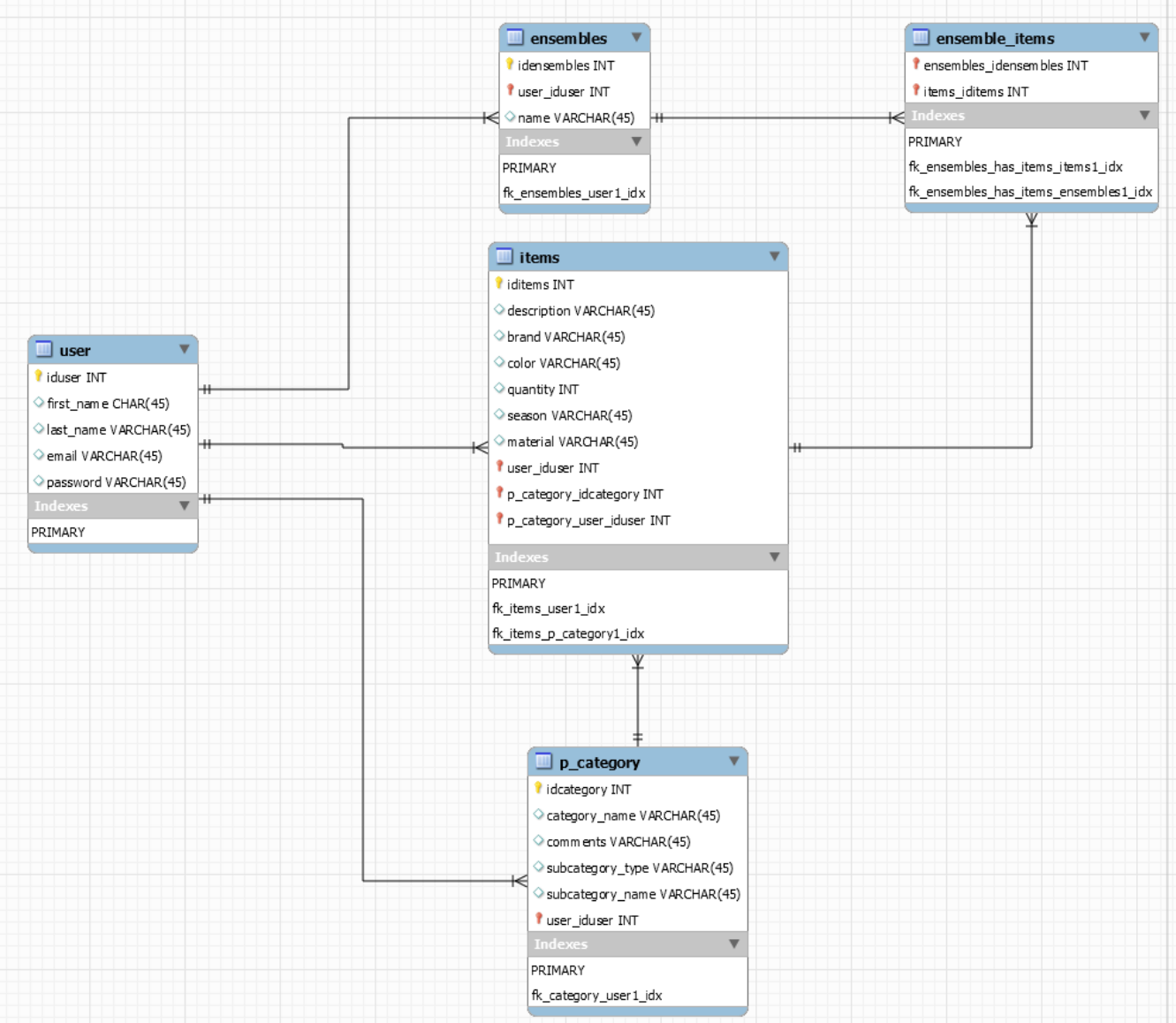


TABLE CREATION QUERIES

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

CREATE SCHEMA IF NOT EXISTS `mcclured-db` DEFAULT CHARACTER SET utf8
COLLATE utf8_general_ci ;
USE `mcclured-db` ;

-- -----------------------------------------------------
-- Table `mcclured-db`.`user`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mcclured-db`.`user` ;

CREATE TABLE IF NOT EXISTS `mcclured-db`.`user` (
  `iduser` INT NOT NULL AUTO_INCREMENT,
  `first_name` CHAR(45) NULL,
  `last_name` VARCHAR(45) NULL,
  `email` VARCHAR(45) NULL,
  `password` VARCHAR(45) NULL,
  PRIMARY KEY (`iduser`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mcclured-db`.`ensembles`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mcclured-db`.`ensembles` ;

CREATE TABLE IF NOT EXISTS `mcclured-db`.`ensembles` (
  `idensembles` INT NOT NULL AUTO_INCREMENT,
  `user_iduser` INT NOT NULL,
  `name` VARCHAR(45) NULL,
  PRIMARY KEY (`idensembles`, `user_iduser`),
  INDEX `fk_ensembles_user1_idx` (`user_iduser` ASC),
  CONSTRAINT `fk_ensembles_user1`
    FOREIGN KEY (`user_iduser`)
    REFERENCES `mcclured-db`.`user` (`iduser`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mcclured-db`.`ensemble_items`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mcclured-db`.`ensemble_items` ;
```

```
CREATE TABLE IF NOT EXISTS `mcclured-db`.`ensemble_items` (
  `ensembles_idensembles` INT NOT NULL,
  `items_iditems` INT NOT NULL,
  PRIMARY KEY (`ensembles_idensembles`, `items_iditems`),
  INDEX `fk_ensembles_has_items_items1_idx` (`items_iditems` ASC),
  INDEX `fk_ensembles_has_items_ensembles1_idx`
(`ensembles_idensembles` ASC),
  CONSTRAINT `fk_ensembles_has_items_ensembles1`
    FOREIGN KEY (`ensembles_idensembles`)
    REFERENCES `mcclured-db`.`ensembles` (`idensembles`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_ensembles_has_items_items1`
    FOREIGN KEY (`items_iditems`)
    REFERENCES `mcclured-db`.`items` (`iditems`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mcclured-db`.`p_category`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mcclured-db`.`p_category` ;

CREATE TABLE IF NOT EXISTS `mcclured-db`.`p_category` (
  `idcategory` INT NOT NULL AUTO_INCREMENT,
  `category_name` VARCHAR(45) NULL,
  `comments` VARCHAR(45) NULL,
  `subcategory_type` VARCHAR(45) NULL,
  `subcategory_name` VARCHAR(45) NULL,
  `user_iduser` INT NOT NULL,
  PRIMARY KEY (`idcategory`, `user_iduser`),
  INDEX `fk_category_user1_idx` (`user_iduser` ASC),
  CONSTRAINT `fk_category_user1`
    FOREIGN KEY (`user_iduser`)
    REFERENCES `mcclured-db`.`user` (`iduser`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mcclured-db`.`items`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mcclured-db`.`items` ;

CREATE TABLE IF NOT EXISTS `mcclured-db`.`items` (
  `iditems` INT NOT NULL AUTO_INCREMENT,
  `category_idcategory` INT NOT NULL,
  `description` VARCHAR(45) NULL,
```

```
  `brand` VARCHAR(45) NULL,
  `color` VARCHAR(45) NULL,
  `quantity` INT NULL,
  `season` VARCHAR(45) NULL,
  `material` VARCHAR(45) NULL,
  `user_iduser` INT NOT NULL,
  PRIMARY KEY (`iditems`, `category_idcategory`, `user_iduser`),
  INDEX `fk_items_category1_idx` (`category_idcategory` ASC),
  INDEX `fk_items_user1_idx` (`user_iduser` ASC),
  CONSTRAINT `fk_items_category1`
    FOREIGN KEY (`category_idcategory`)
    REFERENCES `mcclured-db`.`p_category` (`idcategory`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_items_user1`
    FOREIGN KEY (`user_iduser`)
    REFERENCES `mcclured-db`.`user` (`iduser`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mcclured-db`.`ensemble_items`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mcclured-db`.`ensemble_items` ;

CREATE TABLE IF NOT EXISTS `mcclured-db`.`ensemble_items` (
  `ensembles_idensembles` INT NOT NULL,
  `items_iditems` INT NOT NULL,
  PRIMARY KEY (`ensembles_idensembles`, `items_iditems`),
  INDEX `fk_ensembles_has_items_items1_idx` (`items_iditems` ASC),
  INDEX `fk_ensembles_has_items_ensembles1_idx`
(`ensembles_idensembles` ASC),
  CONSTRAINT `fk_ensembles_has_items_ensembles1`
    FOREIGN KEY (`ensembles_idensembles`)
    REFERENCES `mcclured-db`.`ensembles` (`idensembles`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_ensembles_has_items_items1`
    FOREIGN KEY (`items_iditems`)
    REFERENCES `mcclured-db`.`items` (`iditems`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

GENERAL USE QUERIES

```
-
- get the user information based on email and password
-

SELECT iduser, first_name, last_name, email, password FROM user WHERE
email=[$var] AND password=[$var]

-
- get the count of of users with email and password.  for use in check
to see if user exists.
 -

SELECT COUNT(*) FROM user WHERE email=[$var] AND password=[$var]

-
- delete user
-

DELETE FROM user WHERE iduser=[$var]

-
- insert all user information into database
-

INSERT INTO user (last_name, first_name, email, password) VALUES
([$var], [$var], [$var], [$var])

-
- update user
-

UPDATE user SET last_name=[$var], first_name=[$var], email=[$var],
password=[$var] WHERE iduser=[$var]

-
- get all ensembles based on iduser, no duplicates
-

SELECT idensembles, name FROM ensembles WHERE user_iduser=[$var] GROUP
BY 2 ORDER BY name ASC

-
- get all ensembles based on iduser with duplicates
-

SELECT idensembles, name FROM ensembles WHERE user_iduser=[$var]

-
```

```
- get all ensembles based on iduser and name
-

SELECT idensembles, name FROM ensembles WHERE user_iduser=[$var]AND
name=[$var]

-
- get ensemble name based on iduser and idensembles
-

SELECT name FROM ensembles WHERE user_iduser=[$var] AND
idensembles=[$var]

-
- delete ensemble based on idensemble
-

DELETE FROM ensembles WHERE idensembles=[$var]

-
- delete all ensembles based on iduser
-

DELETE FROM ensembles WHERE user_iduser=[$var]

-
- get all items in an ensemble based on idensembles
-

SELECT items.iditems, items.description, items.brand, items.color,
items.quantity, items.season, items.material,
category.category_name, category.comments, category.subcategory_type,
category.subcategory_name
FROM p_category
INNER JOIN items ON p_category.idcategory = items.category_idcategory
NNER JOIN ensemble_items ON items.iditems =
ensemble_items.items_iditems
WHERE ensemble_items.ensembles_idensembles = [$var]

-
- delete item from an enesemble
-

DELETE FROM ensemble_items WHERE ensembles_idensembles = [$var] AND
items_iditems = [$var]

-
- add ensemble based on iduser and name
-

INSERT INTO ensembles (user_iduser, name) VALUES ([$var], [$var])
```

```
-
- update ensemble
-

UPDATE ensembles SET name=[$var] WHERE user_iduser=[$var]

-
- get all categories with duplicates
-

SELECT idcategory, category_name, comments, subcategory_type,
subcategory_name
FROM p_category
WHERE user_iduser = [$var]

-
- get all category names based on iduser with no duplicates
-

SELECT category.category_name
FROM p_category
INNER JOIN items ON p_category.idcategory = items.category_idcategory
INNER JOIN user ON items.user_iduser = user.iduser
WHERE user.iduser = [$var]
GROUP BY 1 ORDER BY p_category.category_name ASC

-
- get all categories based on category name
-

SELECT category.category_name
FROM p_category
INNER JOIN items ON p_category.idcategory = items.category_idcategory
 INNER JOIN user ON items.user_iduser = user.iduser
WHERE user.iduser = [$var] AND p_category.category_name = [$var]

-
- get category name based on iduser, idcategory
-

SELECT category_name
FROM p_category
WHERE user_iduser = [$var] AND idcategory = [$var]

-
- add category
-

INSERT INTO category (category_name, comments, subcategory_type,
subcategory_name, user_iduser) VALUES ([$var], [$var], [$var], [$var],
[$var])
```

```
-
- delete all categories based on iduser
-

DELETE FROM p_category WHERE user_iduser=[$var]

-
- delete category based on iduser and idcategory
-

DELETE FROM p_category WHERE idcategory=[$var]

-
- get all items based on iduser
-

SELECT iditems, description, brand, color, quantity, season, material,
category_idcategory FROM items where user_iduser = [$var]

-
- get all items based on iduser and not in given ensemble already
-

SELECT items.iditems, items.description, items.brand, items.color,
items.quantity, items.season, items.material,
items.category_idcategory
FROM items
LEFT JOIN ensemble_items ON items.iditems =
ensemble_items.items_iditems
WHERE (ensemble_items.ensembles_idensembles!=[$var]
OR ensemble_items.ensembles_idensembles IS NULL)
AND items.user_iduser =[$var]

-
- add item
-

INSERT INTO  items(category_idcategory, description, brand, color,
quantity, season, material, user_iduser)
VALUES ([$var], [$var], [$var], [$var], [$var], [$var], [$var],
[$var])

-
- delete all items based on iduser
-

DELETE FROM items WHERE user_iduser=[$var]

-
- delete item based on iditem
-
```

```
DELETE FROM items WHERE iditems=[$var]

-
- add item to ensemble
-

INSERT INTO ensemble_items (ensembles_idensembles, items_iditems)
VALUES ([$var], [$var])
```

CONCLUSION

Persemble, the Personal Ensemble Database, changed slightly since the first project proposal. All changes made were to streamline the database in removal of any unutilized entities or relationships. For example, previously there was an entity for subcategory with a relationship to P_category, which was changed to an *attribute* in P_category. Likewise, Persemble was originally an entity with a relationship to User, Ensembles, and Items. As the database progressed, it was discovered that the Persemble entity could be discarded and all relationships held to Ensembles and Items were passed to the User entity as one-to-many.

Another modification created was a relationship from User to P_Category. This was done as to provide only access for the user of Persemble to their categories in the database. Lastly, the addition of the Ensemble_Items entity was made to represent the many-to-many relationship between Ensembles and Items entities.

The database-driven website was created with minimalist design and high functionality in mind. Pictures and color were kept to a minimum and chosen to correlate with the purpose of Persemble. Simplicity in the web design, diagrams, and coding are to reflect the purpose of Persemble itself, to simplify and organize.

Note: Please see accompanying Quick How-To Guide on using Persemble, thank you.