University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: October 22nd, 2025

Group Number: 39

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Sarah Liang	45379450	j8i7t	sarahliang13013@gmail.com
Satsuki Onome	50010305	d1k9c	sonome2005@gmail.com
Dayna Yoon	51355618	s9g8l	dawon020411@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

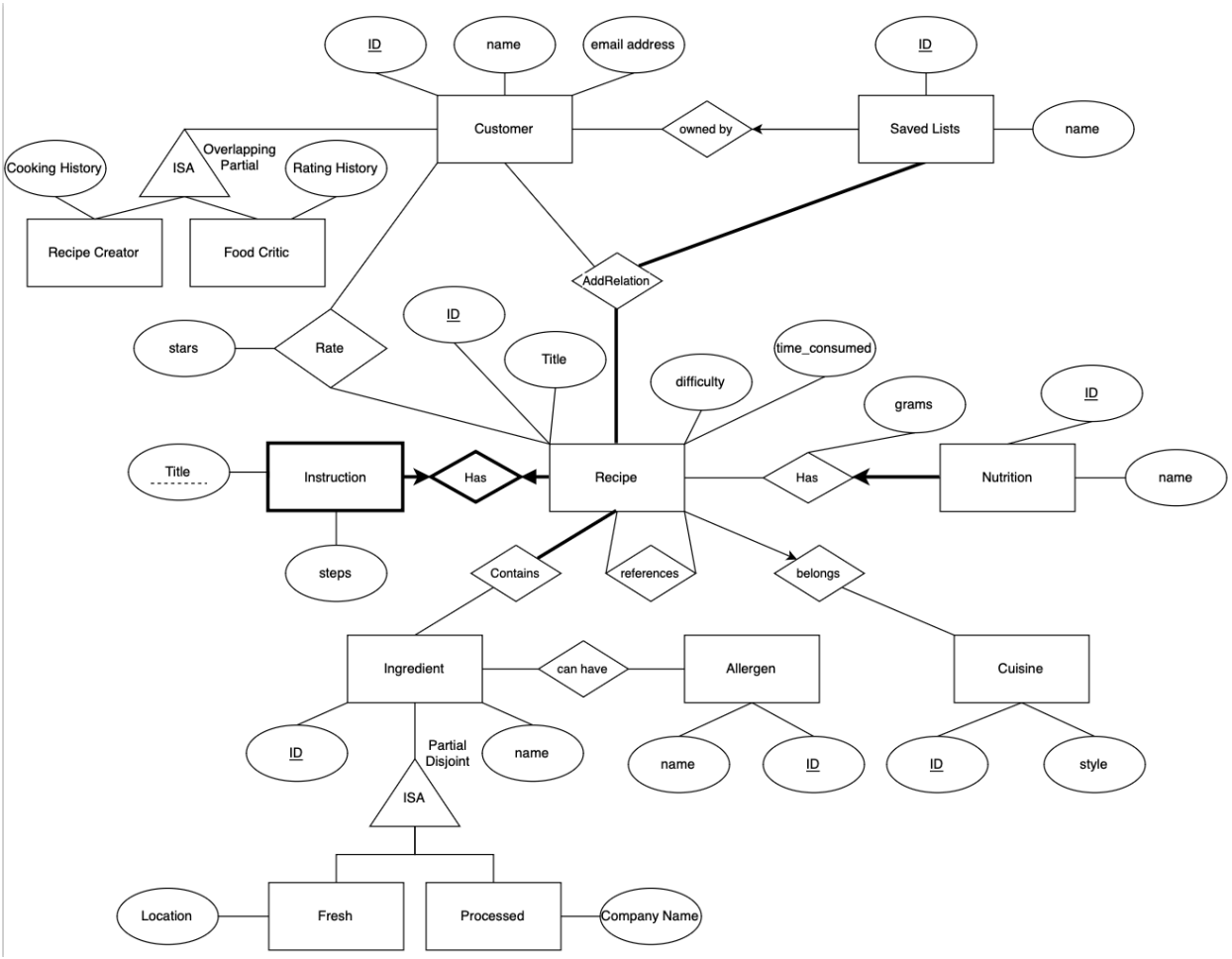
University of British Columbia, Vancouver
Department of Computer Science

Summary of Project:

The domain of this application is culinary and nutrition management, and more specifically, is focused around recipe organization. It encompasses areas such as storing recipes, giving instructions for cooking, and tracking nutritional or dietary information. Moreover, the system also includes community engagement features such as rating or saving recipes, which extend the domain into social and collaborative cooking.

University of British Columbia, Vancouver
Department of Computer Science

ER diagram:



Changes made:

1. The weak entity Instruction has a title that is a partial key, so that it can become a part of the primary key.
2. Fresh or Process ingredients have specific attributes “location” and “company name”. This makes the ISA relationship useful.
3. Replaced “Title” attribute from instructions to step(INTEGER) since there is already a “Title” attribute in Recipe Entity. instead of steps attribute in Instruction, replaced with “notes” attribute to add specific details of the instruction.
4. The relation name ‘Add’ is changed to ‘AddRelation’ due to SQL grammatical error.
5. The attribute name ‘time’ changed to ‘time_consumed’ due to SQL grammatical error.

University of British Columbia, Vancouver
Department of Computer Science

Relational Schema

1. Entities

Customer(ID: INTEGER, name: CHAR(50), email_address: VARCHAR2(500))
PK: ID
CK: email_address (unique)
constraints: email_address NOT NULL

SavedLists(ID: INTEGER, name:CHAR(50), ownerID: INTEGER)
PK: ID
FK: ownerID -> customer(ID)
constraint: ownerID NOT NULL

Recipe(ID: INTEGER, title: CHAR(100), time_consumed: INTEGER, difficulty: CHAR(20), cuisineID: INTEGER)
PK: ID
CK: title (unique)
FK: cuisineID -> Cuisine(ID)

Instruction(step: INTEGER, notes: VARCHAR2(500), recipeID: INTEGER)
PK: (step, recipeID)
FK: recipeID -> Recipe(ID)

Ingredient(ID: INTEGER, name: CHAR(100))
PK: ID
CK: name (unique)

Allergen(ID: INTEGER, name: CHAR(100))
PK: ID

Cuisine(ID: INTEGER, style: CHAR(50))
PK: ID

Nutrition(ID: INTEGER, name: VARCHAR2(500), grams: DECIMAL(6, 2), recipeID: INTEGER)
PK: ID
FK: recipeID -> Recipe(ID)
Constraint: recipeID NOT NULL

Fresh(ID: INTEGER, location: CHAR(100))
PK, FK: ID -> Ingredient(ID)

Processed(ID: INTEGER, companyName: CHAR(100))
PK, FK: ID -> Ingredient(ID)

RecipeCreator(ID: INTEGER, CookingHistory: VARCHAR2(500))
PK, FK: ID -> Customer(ID)
FoodCritic(ID: INTEGER, RatingHistory: VARCHAR2(500))
PK, FK: ID -> Customer(ID)

2. Relationships

AddRelation (CustomerID: INTEGER, RecipeID: INTEGER)
PK: (CustomerID, RecipeID)
FK: CustomerID -> Customer(ID)
RecipeID -> Recipe(ID)

(Owned by relationship already implemented as SavedLists.ownerID)

Rate(CustomerID: INTEGER, RecipeID: INTEGER, stars: INTEGER)
PK: (CustomerID, RecipeID)
FK: CustomerID -> Customer(ID)
RecipeID -> Recipe(ID)
Constraint: stars BETWEEN 1 AND 5

(Instruction relationship already implemented as a weak entity)

(Has(Recipe, Nutrition) relationship already implemented as Nutrition.recipeID -> Recipe(ID))

Contain(RecipeID: INTEGER, IngredientID: INTEGER)
PK: (RecipeID, IngredientID)
FK: RecipeID -> Recipe(ID)
IngredientID -> Ingredient(ID)

ReferencesRelation(recipeID: INTEGER, ReferencedRecipeID: INTEGER)
PK: (RecipeID, ReferencedRecipeID)
FK: RecipeID -> Recipe(ID)
ReferencedRecipeID -> Recipe(ID)

(Belongs relationship already implemented as Recipe.cuisineID -> Cuisine(ID)

CanHave(ingredientID: INTEGER, allergenID: INTEGER)
PK: (ingredientID, allergenID)
FK: ingredientID -> Ingredient(ID)
allergenID -> Allergen(ID)

University of British Columbia, Vancouver
Department of Computer Science

Functional Dependencies

Customer(ID, name, email_address):
1. ID -> name, email_address
2. email_address -> ID, name

SavedLists(ID, name, ownerID)
1. ID -> name, ownerID
2. ownerID, name -> ID

Recipe(ID, title, time_consumed, difficulty, cuisineID)
1. ID -> title, time_consumed, difficulty, cuisineID
2. title -> ID, time_consumed, difficulty, cuisineID

Instruction(recipeID, step)
1. recipeID, step -> notes

Ingredient(ID, name)
1. ID -> name
2. name -> ID

Allergen(ID, name)
1. ID -> name
2. name -> ID (names are unique)

Cuisine(ID, style)
1. ID -> style

Nutrition(ID, name, grams, recipeID)
1. ID -> name, grams, recipeID
2. recipeID, name -> grams

Fresh(ID, location)
1. ID -> location

Processed(ID, companyName)
1. ID -> companyName

RecipeCreator(ID, cookingHistory)
1. ID -> cookingHistory

FoodCritic(ID, ratingHistory)
1. ID -> ratingHistory

AddRelation(customerID, recipeID)

No functional dependency.No further attributes other than PK

Rate(customerID, recipeID, stars)

1. customerID, recipeID -> stars

Contain(RecipeID, IngredientID)

No functional dependency. No further attributes other than PK

ReferencesRelation(recipeID, referencedRecipeID)

No functional dependency. No further attributes other than PK

CanHave(ingredientID, allergenID)

No functional dependency. No further attributes other than PK

Normalization

Customer: BCNF holds

SavedLists: BCNF holds

Recipe: BCNF holds

Instruction: BCNF holds

Ingredient: BCNF holds

Allergen: BCNF holds

Cuisine: BCNF holds

Nutrition: BCNF does not hold, 3NF does not hold.

Nutrition(ID, name, grams, recipeID)

1. ID -> name, grams, recipeID
2. name, recipeID -> grams

closures:

ID^+ = {ID, name, grams, recipeID}

$name, recipeID^+$ = {name, grams, recipeID}

violates 3NF since, $name, recipeID^+$ = {name, grams, recipeID} is not part of the key.

Decompose on name, recipeID -> grams

Nutrition1(name, recipeID, grams), Nutrition2(ID, name, recipeID)

both FD holds 3NF.

A. Nutrition1(name, recipeID, grams), Nutrition2(ID, name, recipeID)

Fresh: BCNF holds

Processed: BCNF holds

RecipeCreator: BCNF holds

FoodCritic: BCNF holds

AddRelation, Rate, Contain, ReferencesRelation, CanHave: BCNF holds

University of British Columbia, Vancouver
Department of Computer Science

Final Normalized Tables

Entity/Relation	All Attributes	Primary Key	Foreign Key	Candidate Key
Customer	ID, name, email_address	ID	N/A	email_address
SavedLists	ID, name, ownerID	ID	ownerID	ownerID, name
Recipe	ID, title, time_consumed, difficulty, cuisineID	ID	cuisineID	title
Instruction	step, recipeID, notes	step, recipeID	recipeID	N/A
Ingredient	ID, name	ID	N/A	name
Allergen	ID, name	ID	N/A	name
Cuisine	ID, style	ID	N/A	N/A
Nutrition (pre normalization)	ID, name, grams, recipeID	ID	recipeID	name, recipeID
Nutrition1	name, grams, recipeID	name, recipeID	recipeID	N/A
Nutrition2	ID, name, recipeID	ID	recipeID	N/A
Fresh	ID, location	ID	ID (ISA)	N/A
Processed	ID, companyName	ID	ID (ISA)	N/A
RecipeCreator	ID, CookingHistory	ID	ID (ISA)	N/A
FoodCritic	ID, RatingHistory	ID	ID (ISA)	N/A
AddRelation	CustomerID, RecipeID	CustomerID, RecipeID	CustomerID, RecipeID	N/A
Rate	CustomerID, RecipeID, stars	CustomerID, RecipeID,	CustomerID, RecipeID	N/A
Contain	RecipeID, IngredientID	RecipeID, IngredientID	RecipeID, IngredientID	N/A
ReferencesRelation	RecipeID, ReferencedRecipeID	RecipeID, ReferencedRecipeID	RecipeID, ReferencedRecipeID	N/A

CanHave	IngredientID, AllergenID	IngredientID, AllergenID	IngredientID, AllergenID	N/A

SQL DDL statements

```
CREATE TABLE Customer (
  ID INTEGER,
  name CHAR(50),
  email_address CHAR(100) NOT NULL,
  UNIQUE (email_address),
  PRIMARY KEY (ID)
);

CREATE TABLE SavedLists (
  ID INTEGER,
  name CHAR(50),
  ownerID INTEGER NOT NULL,
  UNIQUE (ownerID, name),
  PRIMARY KEY (ID),
  FOREIGN KEY (ownerID) REFERENCES Customer(ID)
);

CREATE TABLE Cuisine (
  ID INTEGER,
  style CHAR(50),
  PRIMARY KEY (ID)
);

CREATE TABLE Recipe (
  ID INTEGER,
  title CHAR(100),
  time_consumed INTEGER,
  difficulty CHAR(20),
  cuisineID INTEGER,
  UNIQUE (title),
  PRIMARY KEY (ID),
  FOREIGN KEY (cuisineID) REFERENCES Cuisine(ID)
);

CREATE TABLE Instruction (
  step INTEGER,
  recipeID INTEGER,
  notes VARCHAR2(500),
  PRIMARY KEY (step, recipeID),
  FOREIGN KEY (recipeID) REFERENCES Recipe(ID)
);
```

```
CREATE TABLE Ingredient (
  ID INTEGER,
  name CHAR(100),
  UNIQUE (name),
  PRIMARY KEY (ID)
);

CREATE TABLE Allergen (
  ID INTEGER,
  name CHAR(100),
  UNIQUE (name),
  PRIMARY KEY (ID)
);

CREATE TABLE Nutrition1 (
  name CHAR(100),
  grams DECIMAL(6,2),
  recipeID INTEGER NOT NULL,
  PRIMARY KEY (name, recipeID),
  FOREIGN KEY (recipeID) REFERENCES Recipe(ID)
);

CREATE TABLE Nutrition2 (
  ID INTEGER,
  name CHAR(100),
  recipeID INTEGER,
  PRIMARY KEY (ID),
  FOREIGN KEY (recipeID) REFERENCES Recipe(ID)
);

CREATE TABLE Fresh (
  ID INTEGER,
  location CHAR(100),
  PRIMARY KEY (ID),
  FOREIGN KEY (ID) REFERENCES Ingredient(ID)
  ON DELETE CASCADE
  -- ON UPDATE CASCADE
);

CREATE TABLE Processed (
  ID INTEGER,
  companyName CHAR(100),
  PRIMARY KEY (ID),
```

```
  FOREIGN KEY (ID) REFERENCES Ingredient(ID)
  ON DELETE CASCADE
  -- ON UPDATE CASCADE
);


CREATE TABLE RecipeCreator (
  ID INTEGER,
  cookingHistory VARCHAR2(500),
  PRIMARY KEY (ID),
  FOREIGN KEY (ID) REFERENCES Customer(ID)
  ON DELETE CASCADE
  -- ON UPDATE CASCADE
);

CREATE TABLE FoodCritic (
  ID INTEGER,
  ratingHistory VARCHAR2(500),
  PRIMARY KEY (ID),
  FOREIGN KEY (ID) REFERENCES Customer(ID)
  ON DELETE CASCADE
  -- ON UPDATE CASCADE
);


CREATE TABLE AddRelation (
  CustomerID INTEGER,
  RecipeID INTEGER,
  PRIMARY KEY (CustomerID, RecipeID),
  FOREIGN KEY (CustomerID) REFERENCES Customer(ID),
  FOREIGN KEY (RecipeID) REFERENCES Recipe(ID)
);

CREATE TABLE Rate (
  CustomerID INTEGER,
  RecipeID INTEGER,
  stars INTEGER,
  PRIMARY KEY (CustomerID, RecipeID),
  FOREIGN KEY (CustomerID) REFERENCES Customer(ID),
  FOREIGN KEY (RecipeID) REFERENCES Recipe(ID),
  CHECK (stars BETWEEN 1 AND 5)
);
```

```
CREATE TABLE Contain (
  RecipeID INTEGER,
  IngredientID INTEGER,
  PRIMARY KEY (RecipeID, IngredientID),
  FOREIGN KEY (RecipeID) REFERENCES Recipe(ID),
  FOREIGN KEY (IngredientID) REFERENCES Ingredient(ID)
);

CREATE TABLE ReferencesRelation (
  RecipeID INTEGER,
  ReferencedRecipeID INTEGER,
  PRIMARY KEY (RecipeID, ReferencedRecipeID),
  FOREIGN KEY (RecipeID) REFERENCES Recipe(ID),
  FOREIGN KEY (ReferencedRecipeID) REFERENCES Recipe(ID)
);

CREATE TABLE CanHave (
  IngredientID INTEGER,
  AllergenID INTEGER,
  PRIMARY KEY (IngredientID, AllergenID),
  FOREIGN KEY (IngredientID) REFERENCES Ingredient(ID),
  FOREIGN KEY (AllergenID) REFERENCES Allergen(ID)
);
```

University of British Columbia, Vancouver
Department of Computer Science

Insert Statements (5 tuples each)

```
INSERT INTO Customer VALUES (1, 'Satsuki Onome', 'satsuki@example.com');
INSERT INTO Customer VALUES (2, 'Dayna Yoon', 'dayna@example.com');
INSERT INTO Customer VALUES (3, 'Sarah Liang', 'sarah@example.com');
INSERT INTO Customer VALUES (4, 'cpsc304', 'cpsc304@example.com');
INSERT INTO Customer VALUES (5, 'ubc cs', 'ubccs@example.com');

INSERT INTO RecipeCreator VALUES (1, 'Created 10 recipes this month');
INSERT INTO RecipeCreator VALUES (2, 'Specializes in Korean dishes');
INSERT INTO RecipeCreator VALUES (3, 'Focuses on Italian fusion');
INSERT INTO RecipeCreator VALUES (4, 'Tests community recipes');
INSERT INTO RecipeCreator VALUES (5, 'New creator, learning sushi');

INSERT INTO FoodCritic VALUES (1, 'Rated 50 recipes in total');
INSERT INTO FoodCritic VALUES (2, 'Prefers spicy foods');
INSERT INTO FoodCritic VALUES (3, 'Writes detailed reviews');
INSERT INTO FoodCritic VALUES (4, 'Cares about presentation');
INSERT INTO FoodCritic VALUES (5, 'Values ingredient quality');

INSERT INTO SavedLists VALUES (1, 'Favorites', 1);
INSERT INTO SavedLists VALUES (2, 'To Try', 2);
INSERT INTO SavedLists VALUES (3, 'High Protein', 3);
INSERT INTO SavedLists VALUES (4, 'Spicy Food', 4);
INSERT INTO SavedLists VALUES (5, 'Comfort Meals', 5);

INSERT INTO Cuisine VALUES (1, 'Japanese');
INSERT INTO Cuisine VALUES (2, 'Italian');
INSERT INTO Cuisine VALUES (3, 'Korean');
INSERT INTO Cuisine VALUES (4, 'Chinese');
INSERT INTO Cuisine VALUES (5, 'Western');

INSERT INTO Recipe VALUES (1, 'Pasta Carbonara', 25, 'Medium', 2);
INSERT INTO Recipe VALUES (2, 'Tteokbokki', 20, 'Easy', 3);
INSERT INTO Recipe VALUES (3, 'Sushi', 45, 'Hard', 1);
INSERT INTO Recipe VALUES (4, 'Fried Rice', 15, 'Easy', 4);
INSERT INTO Recipe VALUES (5, 'Burger', 30, 'Medium', 5);

INSERT INTO Instruction VALUES (1, 1, 'Boil pasta until al dente.');
INSERT INTO Instruction VALUES (2, 1, 'Mix egg and cheese in bowl.');
INSERT INTO Instruction VALUES (1, 2, 'Boil rice cakes in spicy sauce.');
INSERT INTO Instruction VALUES (1, 3, 'Prepare sushi rice and roll with fillings.');
INSERT INTO Instruction VALUES (1, 4, 'Stir-fry rice with vegetables.');
```

```
INSERT INTO Ingredient VALUES (1, 'Egg');
INSERT INTO Ingredient VALUES (2, 'Peanut');
INSERT INTO Ingredient VALUES (3, 'Pasta Noodle');
INSERT INTO Ingredient VALUES (4, 'Cheese');
INSERT INTO Ingredient VALUES (5, 'Soy');
INSERT INTO Ingredient VALUES (6, 'Tomato');
INSERT INTO Ingredient VALUES (7, 'Lettuce');
INSERT INTO Ingredient VALUES (8, 'Garlic');
INSERT INTO Ingredient VALUES (9, 'Milk');
INSERT INTO Ingredient VALUES (10, 'Chicken');

INSERT INTO Fresh VALUES (2, 'Vancouver Farm Market');
INSERT INTO Fresh VALUES (5, 'Vancouver Farm Market');
INSERT INTO Fresh VALUES (6, 'BC Tomato Farm');
INSERT INTO Fresh VALUES (7, 'Green Leaf Farm');
INSERT INTO Fresh VALUES (10, 'Free Range Poultry');

INSERT INTO Processed VALUES (1, 'Golden Eggs Farm.');
INSERT INTO Processed VALUES (3, 'Walmart');
INSERT INTO Processed VALUES (4, 'Dairyland Foods');
INSERT INTO Processed VALUES (8, 'Garlic Picks.');
INSERT INTO Processed VALUES (9, 'Happy Cow Dairy');

INSERT INTO Allergen VALUES (1, 'Egg');
INSERT INTO Allergen VALUES (2, 'Dairy');
INSERT INTO Allergen VALUES (3, 'Gluten');
INSERT INTO Allergen VALUES (4, 'Peanut');
INSERT INTO Allergen VALUES (5, 'Soy');

INSERT INTO Nutrition1 VALUES ('Protein', 12.50, 1);
INSERT INTO Nutrition1 VALUES ('Fat', 8.30, 1);
INSERT INTO Nutrition1 VALUES ('Carbs', 65.20, 2);
INSERT INTO Nutrition1 VALUES ('Protein', 10.00, 3);
INSERT INTO Nutrition1 VALUES ('Fat', 5.40, 4);

INSERT INTO Nutrition2 VALUES (1, 'Protein', 1);
INSERT INTO Nutrition2 VALUES (2, 'Fat', 1);
INSERT INTO Nutrition2 VALUES (3, 'Carbs', 2);
INSERT INTO Nutrition2 VALUES (4, 'Protein', 3);
INSERT INTO Nutrition2 VALUES (5, 'Fat', 4);

INSERT INTO AddRelation VALUES (1, 1);
INSERT INTO AddRelation VALUES (1, 2);
```

```
INSERT INTO AddRelation VALUES (2, 3);
INSERT INTO AddRelation VALUES (3, 4);
INSERT INTO AddRelation VALUES (4, 5);

INSERT INTO Rate VALUES (1, 1, 5);
INSERT INTO Rate VALUES (1, 2, 4);
INSERT INTO Rate VALUES (2, 3, 5);
INSERT INTO Rate VALUES (3, 4, 3);
INSERT INTO Rate VALUES (4, 5, 4);

INSERT INTO Contain VALUES (1, 3);
INSERT INTO Contain VALUES (1, 1);
INSERT INTO Contain VALUES (1, 4);
INSERT INTO Contain VALUES (2, 2);
INSERT INTO Contain VALUES (4, 5);

INSERT INTO ReferencesRelation VALUES (1, 2);
INSERT INTO ReferencesRelation VALUES (2, 3);
INSERT INTO ReferencesRelation VALUES (3, 4);
INSERT INTO ReferencesRelation VALUES (4, 5);
INSERT INTO ReferencesRelation VALUES (5, 1);

INSERT INTO CanHave VALUES (1, 1);
INSERT INTO CanHave VALUES (4, 2);
INSERT INTO CanHave VALUES (3, 3);
INSERT INTO CanHave VALUES (5, 5);
INSERT INTO CanHave VALUES (2, 4);
```

University of British Columbia, Vancouver
Department of Computer Science

Some constraints can't be expressed by SQL DDL (#8)

ISA Relationship

- The ISA relationships (Customer → RecipeCreator/FoodCritic and Ingredient → Fresh/Processed) involve participation and overlap/disjoint constraints that cannot be represented in SQL DDL. These will be enforced later using assertions.

Total Participation

- Each Recipe must have at least one Instruction, but this total participation constraint cannot be expressed in SQL DDL. It will be handled later using assertions.

Logical Constraint

- The self-reference restriction in ReferencesRelation (RecipeID ≠ ReferencedRecipeID) cannot be represented directly in SQL DDL and will be handled in the application layer

AI Tool Use Acknowledgement

We hereby state that we did not use any AI tools for any parts of this assignment.