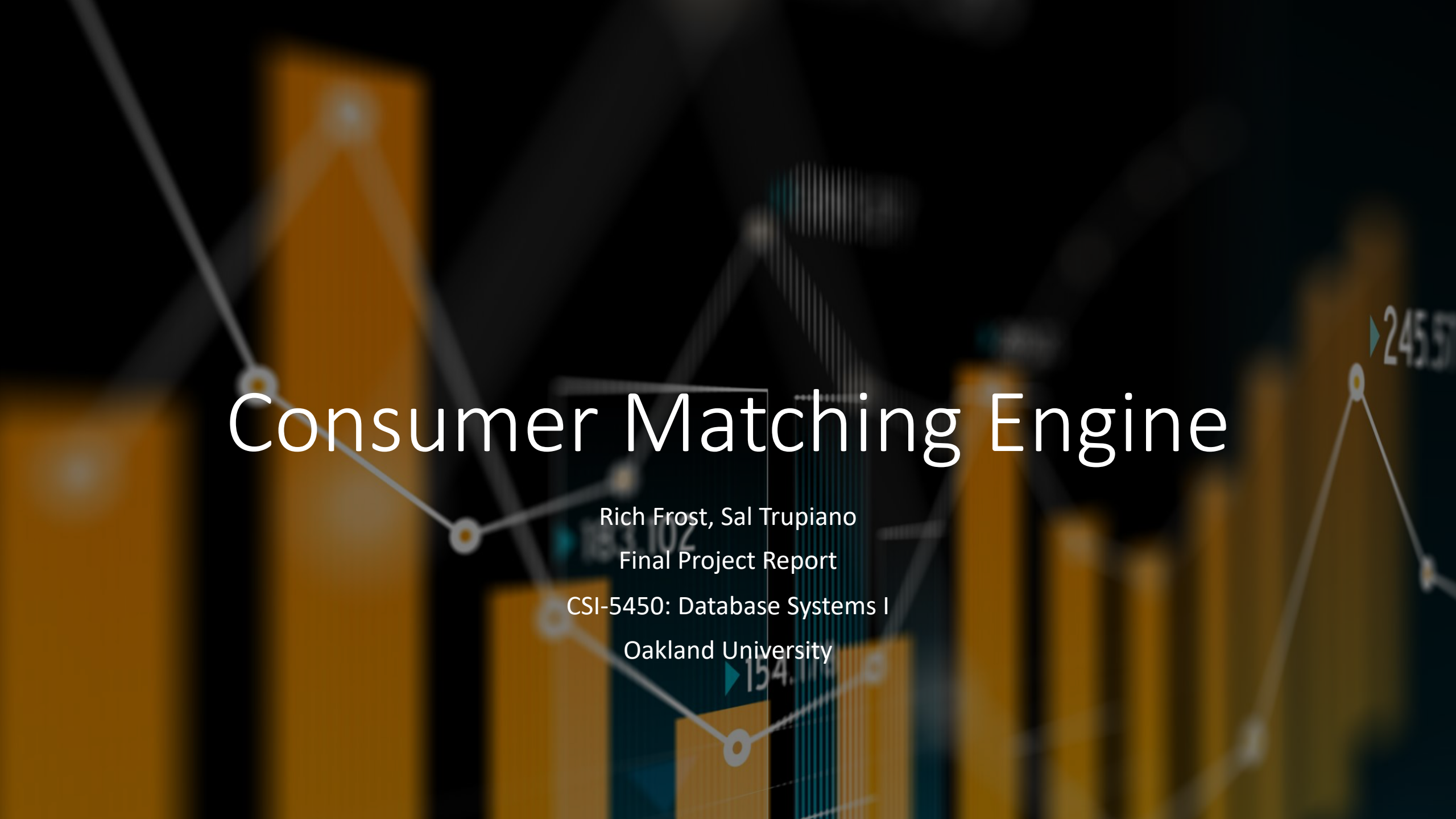


Consumer Matching Engine

Rich Frost, Sal Trupiano

Final Project Report

CSI-5450: Database Systems I

Oakland University

Contents

- Overview
- Business Rules
- ER Diagram
- Query Specifications
- DDL
- Functions & Reports
- Partial Demonstration

Consumer Matching Engine (CME)

We have all heard these or even said these ourselves:

So many choices, what should I do?	I can't find what I want. Do you have something close?	Would you please help pick. What would you do?

We are continually searching for complex items that uniquely suit us.

Cars or Trucks	House or Apartments	Even Relationships

CME helps analyze items of any domain and recommend the ones that best match our unique needs and peferences!

How CME Works!

CME understands that all objects (or people) have features

- The same type of object can have different values for each feature
- People each perfer certain values for feature (color of car, color of eyes, …)
- The importance of each feature varies by person (4 Wheel Drive, School System, …)

CME understands some features are similar to others - Affinity

- Silver Wheels or Aluminum Wheels?
- 4 Bedrooms or 5 Bedrooms

CME searches inventory and finds the best matches for a user

- Based on what they selected for each feature
- Based on how important every feature is to them
- Based on the affinity of different feature values

Business Rules

The application is to remain modular and adaptable to different domains. Examples includes finding best matches on cars, homes, or even dating.

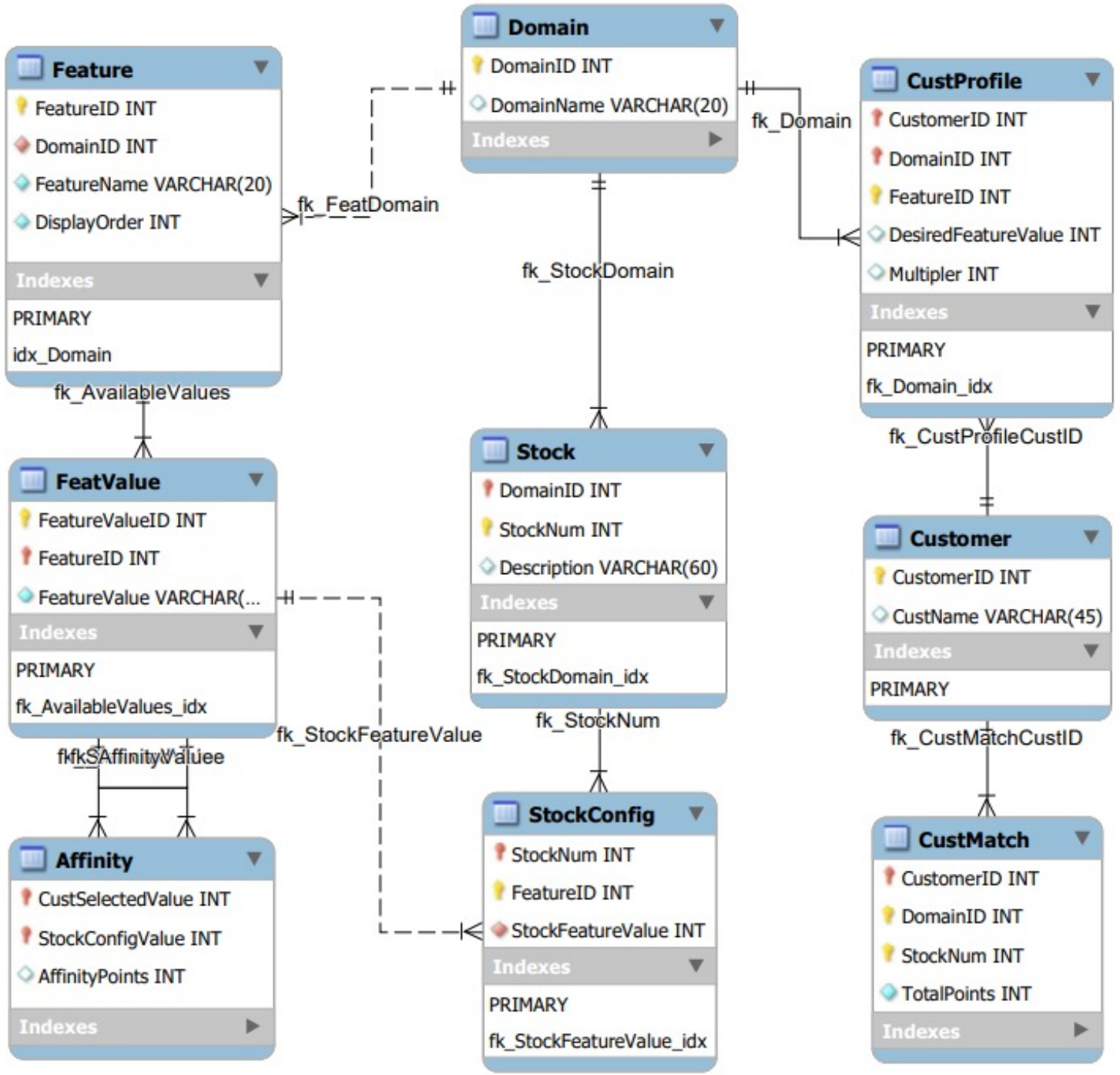
The application should allow features and affinity for matching to be changed.

The feature selections and feature importance that a user makes should be saved and retrieved for the user.

The application will only suggest matches for inventory units that are available for sale.

A user can use the system across multiple domains and selections in one domain will not affect other domains.

ER Diagram


Feature
FeatureID INT
DomainID INT
FeatureName VARCHAR(20)
DisplayOrder INT
Indexes
PRIMARY
idx_Domain
Domain
DomainID INT
DomainName VARCHAR(20)
Indexes
CustProfile
CustomerID INT
DomainID INT
FeatureID INT
DesiredFeatureValue INT
Multipler INT
Indexes
PRIMARY
fk_Domain_idx
fk_FeatDomain
fk_Domain
fk_StockDomain
fk_AvailableValues
fk_CustProfileCustID
FeatValue
FeatureValueID INT
FeatureID INT
FeatureValue VARCHAR(...
Indexes
PRIMARY
fk_AvailableValues_idx
Stock
DomainID INT
StockNum INT
Description VARCHAR(60)
Indexes
PRIMARY
fk_StockDomain_idx
Customer
CustomerID INT
CustName VARCHAR(45)
Indexes
PRIMARY
fk_StockFeatureValue
fk_StockNum
fk_CustMatchCustID
Affinity
CustSelectedValue INT
StockConfigValue INT
AffinityPoints INT
Indexes
StockConfig
StockNum INT
FeatureID INT
StockFeatureValue INT
Indexes
PRIMARY
fk_StockFeatureValue_idx
CustMatch
CustomerID INT
DomainID INT
StockNum INT
TotalPoints INT
Indexes

Query Specifications

- Customer Profile Creation
- Creation of Affinity View
- Match Result Calculation Routing

Customer Profile Creation

- Create Customer record:

```
Use CsiProject;

INSERT INTO Customer
  (CustName)
VALUES
  ('Sal');
```

- Create Customer Profile record:

```
Use CsiProject;

INSERT INTO CustProfile
  (CustomerID,DomainId,FeatureID,DesiredFeatureValue,Multipler)
VALUES
  (1,1,1,1,5);
```

```
Use CSIProject;

Drop view if exists StockAffinity;
Create view StockAffinity as
  Select S.DomainID, StockNum, S.Description as StockDesc,
         F.DisplayOrder, CFG.FeatureID, F.FeatureName, CFG.StockFeatureValue, FV.FeatureValue,
         A.CustSelectedValue, A.StockConfigValue, A.AffinityPoints
    from  (Stock S natural join StockConfig CFG)          -- Configured Stock
          join Feature F on F.featureID = CFG.featureID  -- Feature name and Display order
          join FeatValue FV on FV.FeatureValueID = CFG.StockFeatureValue  -- The value of feature for this stock num
          Join Affinity A on A.StockConfigValue = CFG.StockFeatureValue  -- The affinity points for each value
    Order by S.DomainID, S.Stocknum, FV.FeatureID;
```

Creation of Affinity View

- Creates a Table View, which allows for easier calculation of the Affinity Value for a customer, instead of generating a complex table with multiple joins each time a calculation needs to be performed.

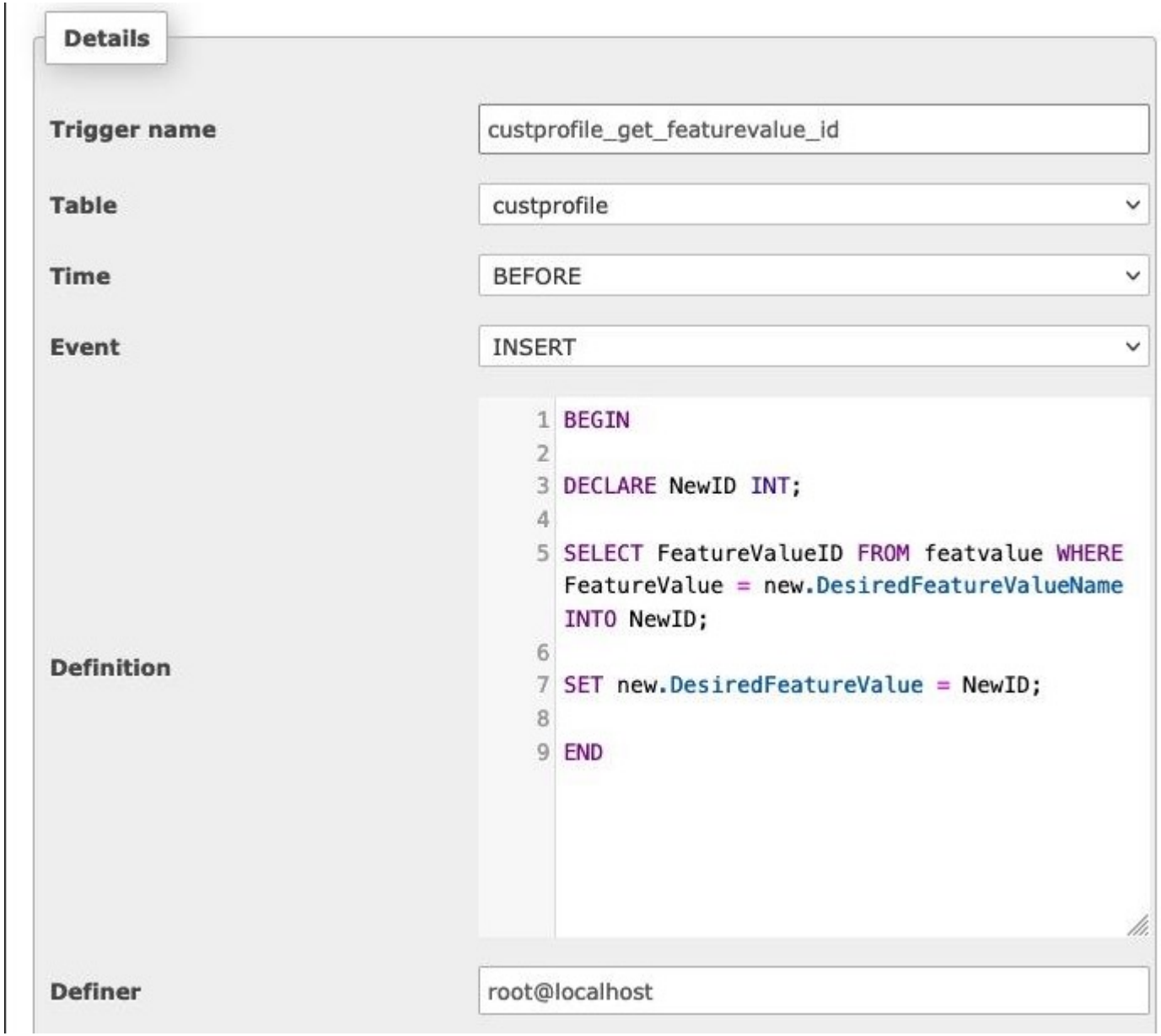
Match Result Calculation Routine

- The logic in these queries reads the customer-desired features and priorities, then calculates the points for each stock item.

```
Use CSIProject;

-- Join StockAffinity with customer selections (in CustProfile table) matching
-- to get the records for each stock unit that have the Affintiy value matching what
-- Customer selected.  This is done by matching the what the user selected C.DesiredFeatureValues)
-- with affinty record for selecting that value.
-- The weighetPoints are calculated my using the importance to the customer.
-- The result table will contain one row for each feature per stock item with the points for that feature.
Drop Table if exists StockPoints;
Create Table StockPoints
    (Select S.DomainID, C.CustomerID, S.StockNum, S.StockDesc,
        S.FeatureID, S.StockFeatureValue, S.FeatureName, S.DisplayOrder, S.FeatureValue,
        S.CustSelectedValue, S.StockConfigValue, S.AffinityPoints,
        DesiredFeatureValue, Multipler, (AffinityPoints * Multipler) as WeightedAffinityPoints
    from StockAffinity S join CustProfile C
        on S.CustSelectedValue = C.DesiredFeatureValue and
        S.DomainID = 1 and CustomerID = 1)  --   *!*!*!*!*!*! Need to get Domain and Customer ID from PHP code  *!*!*!*!*!
    order by stocknum, featureID;
;

--
-- Sum the points for each feature to get the Total Points for each vehicle, order by stock with most points
--
Drop Table if exists StockMatch;
Create Table StockMatch
 Select DomainID, CustomerID, StockNum, StockDesc, sum(WeightedAffinityPoints) as TotalPoints
    from  StockPoints
    Group by StockNum
    order by TotalPoints DESC;

--
-- Delete the stock picks for the current customer and insert all records from StockMatch
--
 Delete
    from CustMatch
    where DomainID=1 and CustomerID=1;  -- *!*!*!*!* Change to get DomainID and CustID from PHP Interface *!*!*!*!*

Insert into CustMatch
  Select CustomerID, DomainID, StockNum, TotalPoints
  from StockMatch;

-- The rankings of the stock units are now in CustMatch and should be display order by TotalPoints
-- Select by CustomerID and DomainID
```

Insert Trigger Finds ID for Text Values

- The User Interface passes Text values to the SQL code.
- Use of a BEFORE-INSERT trigger looks up the numeric key corresponding to the text
- Simplifies User Interface code and assures data consistency



Data Definition Language

Query Specifications for our DDL, with comments.

- Domain
- Feature
- FeatValue

- Defines the different domains
- Defines the features of each domain
- Defines the values available for each feature

```
DROP TABLE IF EXISTS `Domain` ;

CREATE TABLE IF NOT EXISTS `Domain` (
  `DomainID` INT NOT NULL,
  `DomainName` VARCHAR(20) NULL,
  PRIMARY KEY (`DomainID`))
ENGINE = InnoDB;
```

```
DROP TABLE IF EXISTS `Feature` ;

CREATE TABLE IF NOT EXISTS `Feature` (
  `FeatureID` INT NOT NULL COMMENT 'Key value for freature, System assigned',
  `DomainID` INT NOT NULL,
  `FeatureName` VARCHAR(20) NOT NULL COMMENT 'Name of the feature( i.e:  \"Body Style\', Color, Transmission, Engine, Wheel size)',
  `DisplayOrder` INT NOT NULL,
  PRIMARY KEY (`FeatureID`),
  INDEX `idx_Domain` (`DomainID` ASC) ,
  CONSTRAINT `fk_FeatDomain`
    FOREIGN KEY (`DomainID`)
    REFERENCES `Domain` (`DomainID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
DROP TABLE IF EXISTS `FeatValue` ;

CREATE TABLE IF NOT EXISTS `FeatValue` (
  `FeatureValueID` INT NOT NULL AUTO_INCREMENT COMMENT 'Key for table',
  `FeatureID` INT NOT NULL COMMENT 'The ID for the feature being defined.  (FeatureID 6->Color)',
  `FeatureValue` VARCHAR(25) NOT NULL COMMENT 'A text description for the feature (i.e.:  Red, Yellow, \'5sp manual\", \'18 inch black rims\',
  \'Turbocharged V6 2.5L\',  \'Long range battery pack\''',
  PRIMARY KEY (`FeatureValueID`, `FeatureID`),
  INDEX `fk_AvailableValues_idx` (`FeatureID` ASC) ,
  CONSTRAINT `fk_AvailableValues`
    FOREIGN KEY (`FeatureID`)
    REFERENCES `Feature` (`FeatureID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
DROP TABLE IF EXISTS `Affinity` ;

CREATE TABLE IF NOT EXISTS `Affinity` (
  `CustSelectedValue` INT NOT NULL COMMENT 'Value that an affinity is being define for.  (Value ID:3 --> Red) ',
  `StockConfigValue` INT NOT NULL COMMENT 'Value for which an affinity is being defiend for.  (eg AffinityValueID: 33-> Orange)',
  `AffinityPoints` INT NULL COMMENT 'The number of Points awarded for an inventory item haveing the AffinityValue when the cusotmer chooses SelectedValue
  ',
  PRIMARY KEY (`CustSelectedValue`, `StockConfigValue`),
  INDEX `fk_AffinityValue_idx` (`StockConfigValue` ASC) ,
  CONSTRAINT `fk_SlelectedValue`
    FOREIGN KEY (`CustSelectedValue`)
    REFERENCES `FeatValue` (`FeatureValueID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_AffinityValue`
    FOREIGN KEY (`StockConfigValue`)
    REFERENCES `FeatValue` (`FeatureValueID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Affinity

Defines the affinity (or nearness) of different values of a feature

Stock & StockConfig

```
DROP TABLE IF EXISTS `Stock` ;

CREATE TABLE IF NOT EXISTS `Stock` (
  `DomainID` INT NOT NULL,
  `StockNum` INT NOT NULL AUTO_INCREMENT COMMENT 'The stock number for the item',
  `Description` VARCHAR(60) NULL,
  PRIMARY KEY (`StockNum`, `DomainID`),
  INDEX `fk_StockDomain_idx` (`DomainID` ASC) ,
  CONSTRAINT `fk_StockDomain`
    FOREIGN KEY (`DomainID`)
    REFERENCES `Domain` (`DomainID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

- Stock: contains entry for each unit in current inventory
- StockConfig: contains line item detail for each feature of the stock item

```
DROP TABLE IF EXISTS `StockConfig` ;

CREATE TABLE IF NOT EXISTS `StockConfig` (
  `StockNum` INT NOT NULL COMMENT 'Stock numbr for Item',
  `FeatureID` INT NOT NULL COMMENT 'The Feature being specified for the item.  (This really not needed as it can be looked up from featureValue --
  >Feature.Name',
  `StockFeatureValue` INT NOT NULL COMMENT 'The Value of the feature that this item has \n(e.g. stockNum=1423, Feature=Color,Value=Red\n',
  PRIMARY KEY (`StockNum`, `FeatureID`),
  INDEX `fk_StockFeatureValue_idx` (`StockFeatureValue` ASC) ,
  CONSTRAINT `fk_StockNum`
    FOREIGN KEY (`StockNum`)
    REFERENCES `Stock` (`StockNum`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_StockFeatureValue`
    FOREIGN KEY (`StockFeatureValue`)
    REFERENCES `FeatValue` (`FeatureValueID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```sql
DROP TABLE IF EXISTS `CustProfile` ;

CREATE TABLE IF NOT EXISTS `CustProfile` (
  `CustomerID` INT NOT NULL,
  `DomainID` INT NOT NULL,
  `FeatureID` INT NOT NULL COMMENT 'ID of  the feature',
  `DesiredFeatureValue` INT NULL COMMENT 'This references the key in FeatureValue table ',
  `Multipler` INT NULL COMMENT 'Multiplier value for the feature, that is how important the feature is to the
  customer.   Not important would have low value, important features would have highe value.   Will need to do
  analysis to determine the right range. (maybe 1-5?, 1-10?, 1-20?, ....',
  PRIMARY KEY (`CustomerID`, `FeatureID`, `DomainID`),
  INDEX `fk_Domain_idx` (`DomainID` ASC) ,
  CONSTRAINT `fk_Domain`
    FOREIGN KEY (`DomainID`)
    REFERENCES `Domain` (`DomainID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_CustProfileCustID`
    FOREIGN KEY (`CustomerID`)
    REFERENCES `Customer` (`CustomerID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```sql
DROP TABLE IF EXISTS `Customer` ;

CREATE TABLE IF NOT EXISTS `Customer` (
  `CustomerID` INT NOT NULL AUTO_INCREMENT COMMENT 'Key for customer table\n',
  `CustName` VARCHAR(45) NULL COMMENT 'Name of customer',
  PRIMARY KEY (`CustomerID`))
ENGINE = InnoDB;
```

Customer CustProfile

Stores the selections customer has made for each Domain

CustMatch

```
DROP TABLE IF EXISTS `CustMatch` ;

CREATE TABLE IF NOT EXISTS `CustMatch` (
  `CustomerID` INT NOT NULL AUTO_INCREMENT,
  `DomainID` INT NOT NULL,
  `StockNum` INT NOT NULL,
  `TotalPoints` INT NOT NULL,
  PRIMARY KEY (`CustomerID`, `StockNum`, `DomainID`),
  CONSTRAINT `fk_CustMatchCustID`
    FOREIGN KEY (`CustomerID`)
    REFERENCES `Customer` (`CustomerID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Project Functions & Reports

- **Customer Matching Profile:** After User Selects their Domain, they are then presented with a dynamic form that queries the DB for Feature Categories and populates drop-down boxes with the values for each feature category.
- **Customer Best Match Calculation:** Queries Affinity, Stock, Feature, and Customer tables to combine, calculate, and display data to the user their matches. Involves many complex calculations using a View.
- **Consumer Match Report:** Displays the best matches for the User based on the User's feature selection, how they prioritized the features, and the available inventory.

Demonstration

- Fully-Constructed Database with Sample Data
  - Vehicles
  - Homes
  - Friendships
- User Interface written in HTML & PHP