CSC/CPE 366-01: Final Design Report

Team MASDA

Mason Stevenson – masonjstevenson@gmail.com

Andrew Voorhees – andrewvoorhees@gmail.com

Steven Thon – steventhon94@gmail.com

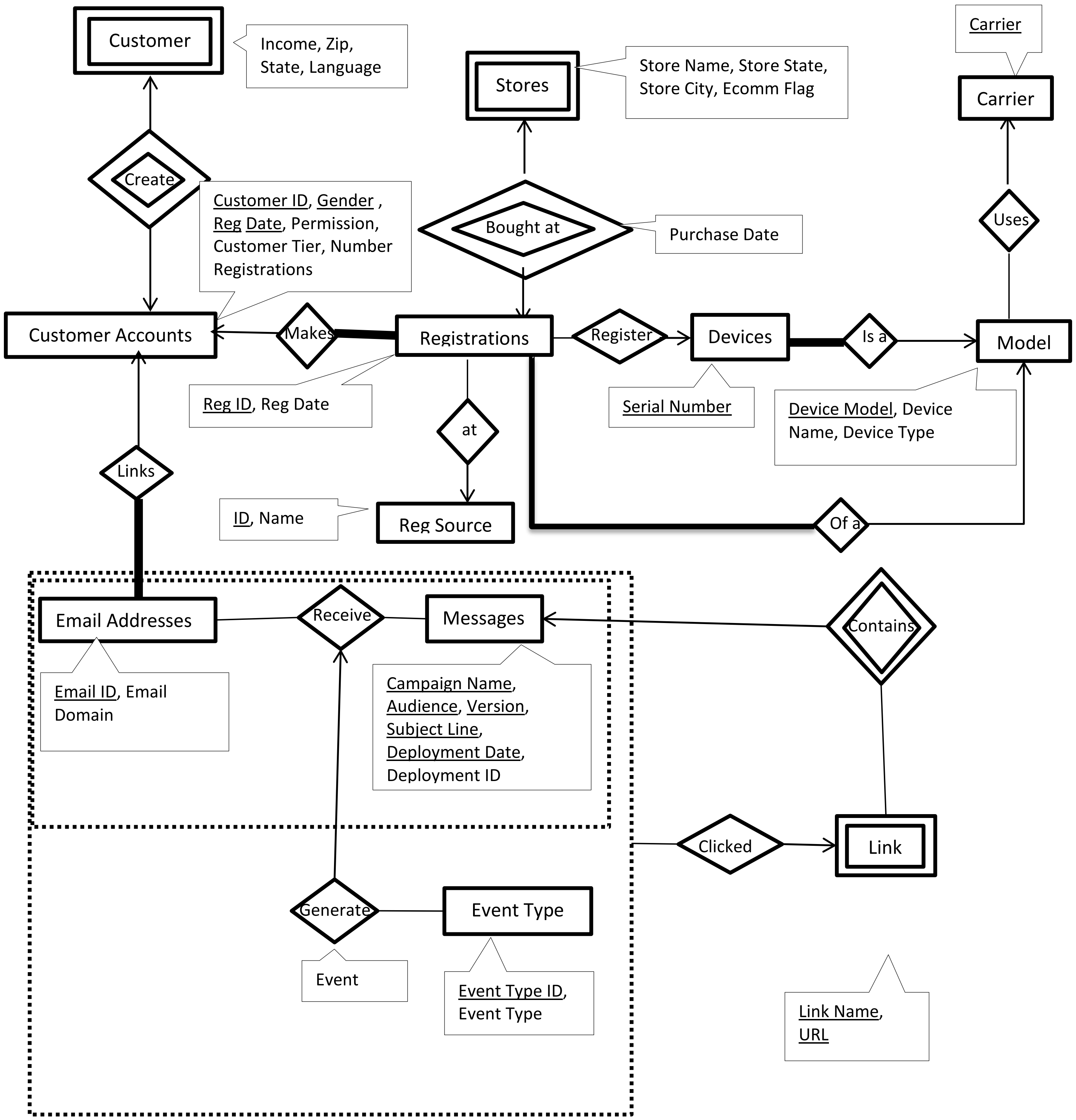
Derek Chan – dc2000usa@gmail.com

Albert Ye – albertxye@gmail.com

Contents

1 – Entity Relationship Model

1.1 – E-R Diagram



Customer
Income, Zip, State, Language
Stores
Store Name, Store State, Store City, Ecomm Flag
Carrier
Carrier
Create
Customer ID, Gender , Reg Date, Permission, Customer Tier, Number Registrations
Bought at
Purchase Date
Uses
Customer Accounts
Makes
Registrations
Register
Devices
Is a
Model
Reg ID, Reg Date
at
Serial Number
Device Model, Device Name, Device Type
Links
ID, Name
Reg Source
Of a
Email Addresses
Receive
Messages
Contains
Email ID, Email Domain
Campaign Name, Audience, Version, Subject Line, Deployment Date, Deployment ID
Clicked
Link
Generate
Event Type
Event
Event Type ID, Event Type
Link Name, URL

1.2 – Entity Sets

- Customer Accounts(***Customer Id*** [String], Permission [String], ***Gender*** [string], ***Reg Date***[DATE], Customer Tier [String], Number Registrations [Int])
- Registrations(***Reg ID*** [String],Reg Source Id [String], Reg Source Name [String], Reg Date [DATE])
- Reg Source(**ID** [Int], Name [String])
- Devices(***Serial Number*** [String])
- Model(***Device Model*** [String], Device Name [String], Device Type [String])
- Email Addresses(***Email Id*** [Int], Email Domain [String])
- Event Type(***Event Type ID*** [String], Event Type Name)
- Carrier(***Carrier*** [String])
- Messages(***Campaign Name*** [String], ***Audience*** [String] ***Version*** [Int], Subject Line [String], **Deployment Date** [Date], Deployment Id [String])

1.3 – Weak Entity Sets

(Discriminators in bold)

- Customer(Income [Double], Zip [Int], State [String], Language [String])
- Stores(Store Name [String], Store State [String], Store City [String], Ecomm Flag [Int])
- Link(**Link Name** [String], ***URL*** [String])

1.4 – Relationship Sets

(Entities, Type, Weak/Strong, Attributes)

- Creates: (Customer, Customer Account), One-to-One, Weak, none
- Makes: (Customer Account, Registration), Many-to-Many, Weak, none
- Bought at: (Registration, Stores), One-to-One, Strong, Purchase Date
- Register: (Registration, Devices)Many-to-One, Weak, none
- At: (Registration, Reg Source), Many-toOne, Strong, none
- Is a: (Devices, Model), Many-to-One, Strong, none
- Of a: (Registration, Model), Many-to-One, Strong, none
- Uses: (Carrier, Model)Many-to-One, Strong, none
- Links to: (Email Addresses, Customer Account), One-to-Many, Strong, none
- Receive: (Email Addresses, Messages), One-to-Many, Strong, none
- Generate: (Event Type, Receive)One-to-Many, Strong, event_date [DATETIME]
- Contains: (Messages, Link), Many-to-One, Weak, none
- Clicked: (Generate, Link)Many-to-One, Strong, none

2 – Relational Model

CustomerAccounts
attributes: id, customer_id, permission, reg_date, tier, number_registrations
primary key: id
unique: customer_id

Customers
attributes: account, income, zip, state, gender, language
primary key: account
foreign keys: “account” referencing table “CustomerAccounts”

Carrier
attributes: id, name
primary keys: id
unique: name

Models
attributes: id, model_id, name, model_type, carrier
primary key: id
foreign keys: “carrier” referencing table “Carrier”
unique: model_id

Devices
attributes: id, serial_num, model
primary keys: id
foreign key: “model” referencing table “Models”
Constraints: Constrained by table “Models”, so “model” cannot be null
unique: serial_num

Registrations
attributes: id, reg_id, customer, device, model, source_id, reg_date
primary keys: id
foreign keys: customer referencing table CustomerAccounts, model referencing table Models, device referencing table Devices, source_id referencing Sources
Constraints: Constrained by table “Models”, so “model” cannot be null. Constrained by table “CustomerAccounts”, so customer cannot be null
unique: reg_id

Sources
attributes: source_id, source_name
primary key: source_id

Stores
attributes: id, name, state, city, ecomm, purchase_date
primary keys: id
foreign keys: “id” referencing table “Registrations”

EmailAddresses
attributes: id, email_id, customer, domain
primary keys: id
foreign keys: “customer” referencing table “CustomerAccounts”
Constraints: Constrained by table “CustomerAccounts”, so customer cannot be null unique: email_id

Messages
attributes: id, campaign_name, audience, email, version, subject_line, dep_date, deployment, campaign
primary keys: id, dep_date
foreign keys: “email” referencing table “EmailAddresses”
unique: (campaign_name, audience, version, subject_line, dep_date)

Recieve
attributes: id, message, email
primary keys: id
foreign keys: "message" referencing table "Messages", "email" referencing table "EmailAddresses"
unique: (message, email)

EventTypes
attributes: id, type_id, name
primary keys: id
unique: type_id

Generate
attributes: id, event_date, recieve, event_type
primary keys: id
unique: (event_date, receive, event_type)
foreign keys: “event_type” referencing table “EventTypes”, "recieve" referencing "Receive”

Links
attributes: id, name, url, message primary keys: id foreign keys: "message" referencing "Messages"
unique: (name, url, message)

Clicked attributes: id, generated, link primary keys: id foreign keys: "generated" referencing table Generate, "link" referencing table "Links" unique: (generated, link)

3 – ETL Process

For our ETL, we used python scripts that comb through the CSV files and make SQL insert scripts for each of the tables in our database. One difficulty we faced when loading the data into the database was the large number of duplicate rows we found. We were able to create filters in the python scripts to eliminate the duplicates in the .sql insert files we generated thus speeding up their runtime.

The names of the python scripts we used are as follows:

Builder-st.py – creates [insert-Registrations.sql, insert-Stores.sql, insert-EmailAddresses.sql]. Insert sql statements for tables Registrations, Stores, and EmailAddresses, respectively.

Builder.py – creates [insert-Generate.sql, insert-Linked.sql, insert-Clicked.sql]. Reads from the email events csv file to create all of its tables.

Builder2.py – creates [insert-Models.sql, insert-Devices.sql]. Built by reading from the device models csv and the device registrations csv respectively.

Eventtypesscript.py – creates [insert-EventTypes.sql]

Messagescript.py – Takes information from email CP_Email csv and creates the messages table. Also, takes in the file CP_Email.csv for the data outputs script Insert-Message.sql with appropriate insert statements.

Recievescript.py – Takes information from email data csv and creates the Recieve table. Also takes in the file CP_Email_Final for the dataoutputs script insert-Recieve.sql with appropriate insert statments

scriptFactory.py – creates [insert-CustomerAccounts.sql]. When two lines in the file are considered to be duplicate entries, filters them by only creating an insert statement for the most recent customer account. (num_registrations is not inserted, see updater.py)

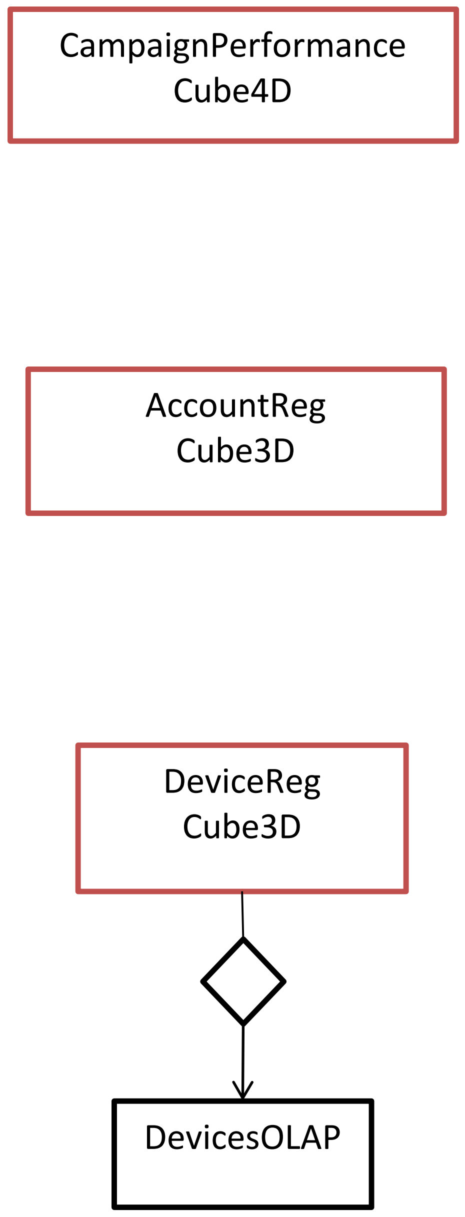
scriptFactory2.py – creates [insert-Customers.sql]. When two lines in the file are considered to be duplicate entries, filters them by only creating an insert statement for the most recent customer.

scriptFactory3.py – creates [insert-Carrier.sql]. Only makes one insert per unique carrier name.

updater.py – creates [update-CustomerAccounts.sql]Updates the CustomerAccounts table to have registration counts.

4 – Data Warehouse

4.1 E-R Diagram



4.2 Data Warehouse Entity Sets

- CampaignPerformanceCube4D(***Campaign Name*** [String], ***Audience*** [String] ***Version*** [Int], **Subject Line** [String], **Deployment Date** [String], Unique Emails Delivered [Int], Unique Emails Opened [Int], Unique Clickers [Int], Unsubscribed [Int])
- AccountRegCube3D(**State** [Int]**, Month** [String], **Year** [Int], **Permission** [String**],** Num Customers [Int])
- DeviceRegCube3D(**Carrier** [String], **Month** [String], **Year** [Int], **Device** [Int], Num Customers [Int])
- DevicesOLAP(**Model Id** [String], Name [String], Model Type [String])

4.3 Data Warehouse Relationship Sets

*All relationships are many to one

- DeviceRegCube3D-> Devices

5 – Data Warehouse Relational Model

CampaignPerformanceCube4D (campaign_name, audience, version, subject_line, dep_date, Unique_Emails_Delivered, Unique_Emails_Opened, Unique_Clickers, Unsubscribed)

- Primary: campaign_name, audience, version, subject_line, dep_date
- Foreign: none

AccountRegCube3D (State, Month, Permission, Num_Customers)

- Primary: State, Month, Permission
- Foreign: none

DeviceRegCube3D (Carrier_Name, Month, Year, Device_Name, Num_Customers)

- Primary: Carrier_Name, Month, Year, Device_Name
- Foreign: Device_Name references DevicesOLAP

DevicesOLAP (id, model_id, name, model_type)

- Primary: id
- Foreign: none

6 – Data Dictionary

6.1 – CampaignPerformanceCube4D

Attribute	**Type**	**Comes From**
campaign_name	Dimension	Messages
audience	Dimension	Messages
version	Dimension	Messages
subject_line	Dimension	Messages
dep_date	Dimension	Messages
Unique_Emails_Delivered	Measure	Recieve
Unique_Emails_Opened	Measure	Generate, Receive
Unique_Clickers	Measure	Generate, Receive
Unsubscribed	Measure	Generate, Receive

6.2 – AccountRegCube3D

Attribute	**Type**	**Comes From**
State	Dimension	Customer
Month	Dimension	CustomerAccounts
Permission	Dimension	CustomerAccounts
Num_Customers	Measure	CustomerAccounts

6.3 – DeviceRegCube3D

Attribute	**Type**	**Comes From**
Carrier_Name	Dimension	Carrier
Month	Dimension	Registrations
Year	Dimension	Registrations
Device_Name	Dimension	Models
Num_Customers	Measure	Registrations