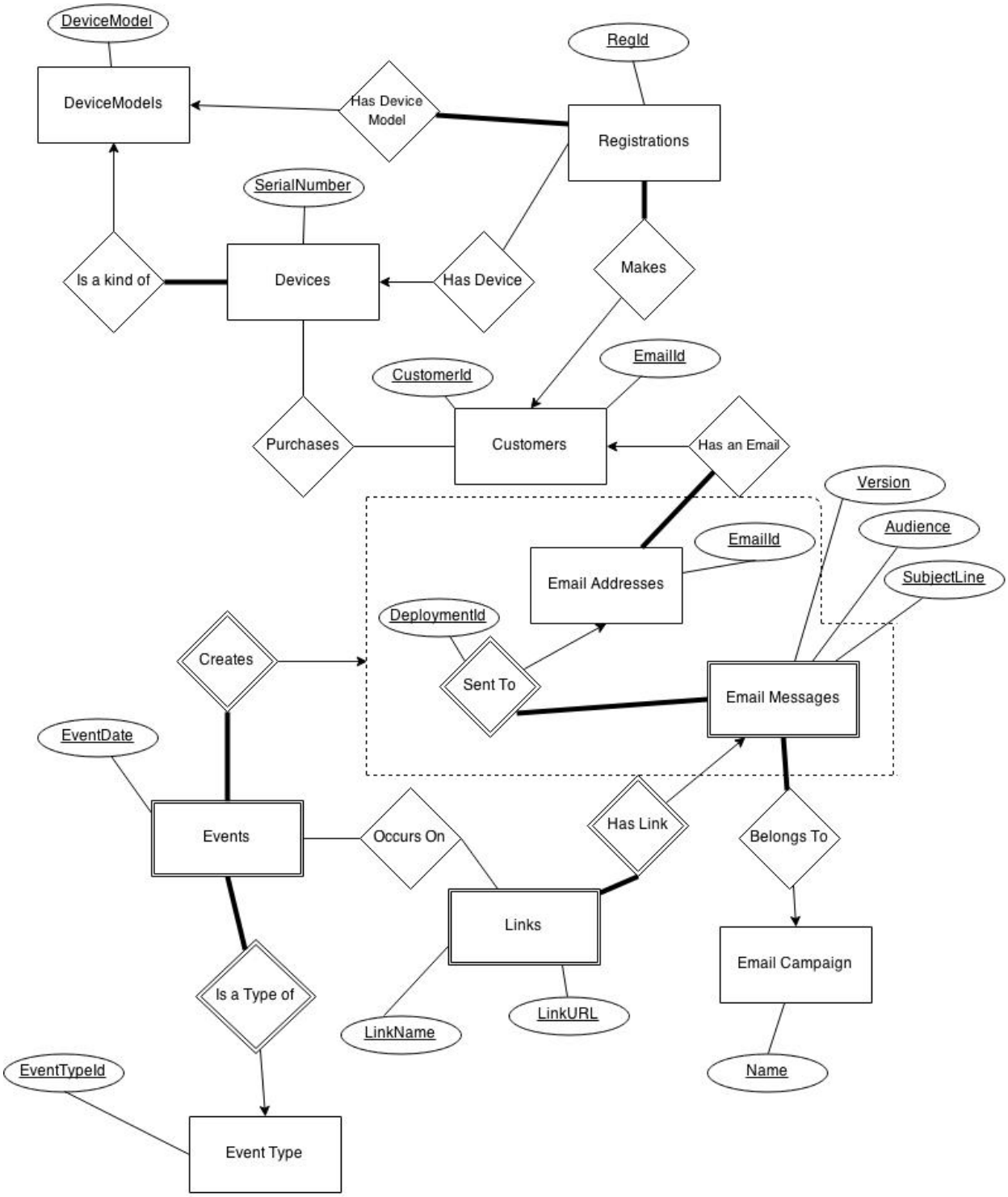
BrandX Database Design and Analytics

Green Team

CPE366 - Database Modeling, Design and Implementation
Section 01

Aaron Pramana • aaron@sociotopia.com
Eric Tran • etran893@gmail.com
Josephine Suen • jsuen@calpoly.edu
Scott Lam • slam08@calpoly.edu
Tim Mendez • tmendez@calpoly.edu

(a) E-R Model for Operational Data Store



ENTITY SETS

EmailCampaigns

- String CampaignName

EmailMessages

Weak entity set

Version, Audience, SubjectLine are pseudo ID

- String Version
- String Audience
- String SubjectLine

Events

Weak entity set

EventDate is pseudo ID

- Timestamp EventDate

Links

Weak entity set

LinkName, LinkURl are pseudo ID

- String LinkName
- String LinkURL

EventTypes

- Int EventTypeID
- String EventTypeName

Devices

- String SerialNumber

DeviceModels

- String DeviceModel
- String DeviceType
- String DeviceName
- String Carrier

Customers

- Int CustomerID
- Int ZIP
- String State
- String Gender
- String IncomeLevel
- Boolean Permission
- String Language
- String CustomerTier
- Int NumberOfRegistrations

EmailAddresses

- String EmailID
- String EmailDomain

Registrations

Date RegistrationDate
Int RegistrationID
Int RegistrationSourceID
String RegistrationSourceName
String DeviceModel

RELATIONSHIP SETS

BelongsTo (EmailMessage, EmailCampaign)

Many-to-one relationship
Identifying relationship for EmailMessage
Required for EmailMessage

SentTo (EmailMessage, EmailAddress)

Many-to-one relationship
Identifying relationship for EmailMessage
Required for EmailMessage

- Int DeploymentId
- Timestamp DeploymentDate

HasLink (EmailMessage, Link)

Many-to-one relationship

Identifying relationship for Link

Required for Link

IsTypeOf (Event, EventType)

- *Many-to-one relationship*
- *Identifying relationship for Event*
- *Required for Event*

HasEmail (Customer, EmailAddress)

One-to-many relationship

Required for EmailAddress

Purchases (Customer, Devices)

Many-to-many relationship

- Date Purchase Date
- String Purchase Store Name
- String Purchase Store State
- String Purchase Store City
- Boolean Ecomm Flag

Makes (Customer, Registration)

One-to-many relationship

Required for Registration

HasDeviceModel (Registration, DeviceModel)

Many-to-one relationship

Required for Registration

HasDevice (Registration, Device)

Many-to-one relationship

IsKindOf (Device, DeviceModel)

Many-to-one relationship

Required for Device

AGGREGATIONS

Creates (SentTo, Event)

Many-to-one relationship

Identifying relationship for Event

Required for Event

(b) Relational Model for Operational Data Store

Customers

- primary key int CustomerID
- int RegSourceID
- varchar RegSourceName
- char ZIP
- char State
- char Gender
- varchar IncomeLevel
- boolean Permission
- char Language
- char CustomerTier

EmailAddresses

- primary key int EmailID
- varchar EmailDomain NOT NULL
- int CustomerID NOT NULL
- foreign key CustomerID references Customers(CustomerID)

EmailMessages

- primary key int EmailMessageID
- varchar Audience NOT NULL
- varchar Version NOT NULL
- varchar SubjectLine NOT NULL
- varchar CampaignName NOT NULL
- unique (Audience, Version, SubjectLine, CampaignName)

EmailMessagesSent

- primary key (EmailMessageID, EmailID)
- int EmailMessageID
- int EmailID
- int DeploymentID NOT NULL
- date DeploymentDate NOT NULL
- foreign key EmailMessageID references EmailMessages(EmailMessageID)
- foreign key EmailID references EmailAddresses(EmailID)

EventTypes

- primary key int EventTypeID
- varchar EventTypeName

Events

- primary key (EmailID, EmailMessageID, EventTypeID)
- datetime EventDate NOT NULL
- int EventTypeID NOT NULL
- int EmailID
- int EmailMessageID
- foreign key EventTypeID references EventTypes(EventTypeID)
- foreign key (EmailMessageID, EmailID) references EmailMessagesSent(EmailMessageID, EmailID)

Links

- primary key (LinkName, LinkURL, EmailMessageID)
- varchar LinkName
- varchar LinkURL
- int EmailMessageID
- foreign key EmailMessageID references EmailMessages(EmailMessageID)

EventLinkLookUp

- primary key (EmailID, EmailMessageID, EventTypeID, LinkName, LinkURL)
- varchar LinkName
- varchar LinkURL
- int EmailMessageID
- int EmailID

- int EventTypeID
- foreign key (EmailMessageID, EmailID) references EmailMessages(EmailMessageID, EmailID)
- foreign key (LinkName, LinkURL) references Links(LinkName, LinkURL)
- foreign key EventTypeID references EventTypes(EventTypeID)

DeviceModels

- primary key varchar DeviceModel
- varchar DeviceName
- varchar DeviceType
- varchar Carrier

Devices

- primary key int DeviceID
- varchar SerialNumber NOT NULL
- varchar DeviceModel NOT NULL
- foreign key DeviceModel references DeviceModels(DeviceModel)

Purchases

- primary key int PurchaseID
- datetime PurchaseDate
- varchar PurchaseStoreName
- varchar PurchaseStoreCity
- char PurchaseStoreState
- boolean ECommFlag
- int CustomerID
- int DeviceID
- varchar DeviceModel NOT NULL
- foreign key CustomerID references Customers(CustomerID)
- foreign key DeviceID references Devices(DeviceID)
- foreign key DeviceModel references DeviceModels(DeviceModel)

Registrations

- primary key int RegistrationID
- date RegistrationDate
- int RegistrationSourceID
- varchar RegistrationSourceName

- varchar DeviceModel NOT NULL
- int CustomerID
- int DeviceID
- foreign key CustomerID references Customers(CustomerID)
- foreign key DeviceID references Devices(DeviceID)
- foreign key DeviceModel references DeviceModels(DeviceModel)

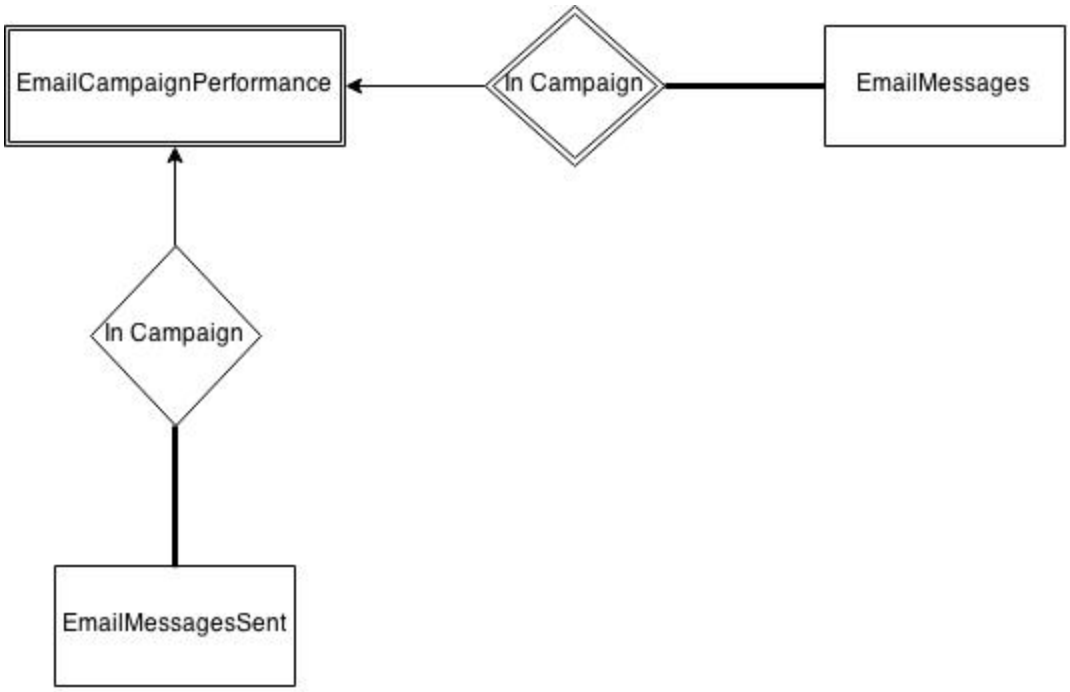
(c) ETL Process

parser.py parses the CSV files and outputs SQL scripts consisting of INSERT statements for each table in the database

1. Create or overwrite files DB-build-TableName.sql
   For each table:
   a. Store primary keys and row counts in global variables
   b. Do not write INSERT if primary key already exists
   c. Every 2000 rows, start a new batch INSERT statement
2. **Parse CP_Account.csv**
   For each row:
   a. Write INSERT for Customers
   b. Write INSERT for Emails
3. **Parse CP_Device_Model.csv**
   For each row:
   a. Write INSERT for DeviceModels
4. **Parse CP_Device.csv**
   Keep a global DeviceID and PurchaseID and increment after each INSERT
   For each row:
   a. If the DeviceModel doesn't exist in CP_Device_Model.csv, write INSERT for DeviceModels
   b. If there is a SerialNumber, write INSERT for Devices
   c. If there isn't a SerialNumber, Device is NULL in Purchase and Registration
   d. Write INSERT for Purchases using incrementing PurchaseID
   e. Write INSERT for Registrations using incrementing DeviceID
5. **Parse CP_Email_Final.csv**
   Keep a global EmailMessageID dictionary mapping Messages (Campaign, Verison, Audience, and SubjectLine) to an ID number
   For each row:
   a. If EmailMessageID already exists, get the ID number to use as a foreign key
   b. If EmailMessageID doesn't exist, update dictionary using the length of the dictionary as the ID number and write INSERT for EmailMessages
   c. Write INSERT for EmailMessagesSent

d. Write INSERT for EventTypes
e. Write INSERT for Events
f. If there is a Link, write INSERT for Links
g. If there is a Link, write INSERT for EventLinkLookUp

(d) E-R Model for Data Warehouse



(e) Relational Model for Data Warehouse

EmailCampaignPerformance

- varchar CampaignName
- varchar Audience
- varchar Version
- date DeploymentDate NOT NULL
- bigint EmailsSent NOT NULL DEFAULT 0
- decimal EmailsBounced NOT NULL DEFAULT 0
- bigint EmailsComplained NOT NULL DEFAULT 0

- bigint EmailsClicked NOT NULL DEFAULT 0
- bigint EmailsOpened NOT NULL DEFAULT 0
- bigint EmailsUnsubscribed NOT NULL DEFAULT 0

EmailMessages

- primary key int EmailMessageID
- varchar Audience NOT NULL
- varchar Version NOT NULL
- varchar SubjectLine NOT NULL
- int DeploymentID NOT NULL
- varchar CampaignName NOT NULL

EmailMessagesSent

- primary key int EmailMessageID
- primary key int EmailID
- int DeploymentID NOT NULL
- date DeploymentDate NOT NULL
- foreign key EmailMessageID references EmailMessages(EmailMessageID)
- foreign key EmailID references EmailAddresses(EmailID)

(f) Data Dictionary

Customers

- CustomerID – from **Customer ID** in the Account Registration Dataset
- RegSourceID – from **Reg Source ID** in the Account Registration Dataset
- RegSourceName – from **Reg Source Name** in the Account Registration Dataset
- ZIP – from **ZIP** in the Account Registration Dataset
- State – from **State** in the Account Registration Dataset
- Gender – from **Gender** in the Account Registration Dataset
- IncomeLevel – from **Income Level** in the Account Registration Dataset
- Permission – from **Permission** in the Account Registration Dataset
- Language – from **Language** in the Account Registration Dataset
- CustomerTier – from **Customer Tier** in the Account Registration Dataset

EmailAddresses

- EmailID – from **Email ID** in the Account Registration Dataset
- EmailDomain – from **Email Domain** in the Account Registration Dataset
- CustomerID – from **Customer ID** in the Account Registration Dataset

EmailMessages

- EmailMessageID – uniquely identifies EmailMessage records
- Audience – from **Audience** in the Email Event Dataset
- Version – from **Version** in the Email Event Dataset
- SubjectLine – from **Subject Line** in the Email Event Dataset
- CampaignName – from **Campaign Name** in the Email Event Dataset

EmailMessagesSent

- EmailMessageID – references EmailMessages
- EmailID – from **Email ID** in the Email Event Dataset
- DeploymentID – from **Deployment ID** in the Email Event Dataset
- DeploymentDate – from **Deployment Date** in the Email Event Dataset

EventTypes

- EventTypeID – from **Event Type ID** in the Email Event Dataset

- EventTypeName – from **Event Type Name** in the Email Event Dataset

Events

- EventDate – from **Event Date** in the Email Event Dataset
- EventTypeID – from **Event Type ID** in the Email Event Dataset
- EmailID – from **Email ID** in the Email Event Dataset
- EmailMessageID – references EmailMessages

Links

- LinkName – from **Link Name** in the Email Event Dataset
- LinkURL – from **Link URL** in the Email Event Dataset
- EmailMessageID – references EmailMessages

EventLinkLookUp

- LinkName – from **Link Name** in the Email Event Dataset
- LinkURL – from **Link URL** in the Email Event Dataset
- EmailMessageID – references EmailMessages
- EmailID – from **Email ID** in the Email Event Dataset
- EventTypeID – from **Event Type ID** in the Email Event Dataset

DeviceModels

- DeviceModel – from **Device Model** in the Device Dataset
- DeviceName – from **Device Name** in the Device Dataset
- DeviceType – from **Device Type** in the Device Dataset
- Carrier – from **Carrier** in the Device Dataset

Devices

- DeviceID – uniquely identifies Device records
- SerialNumber – from **Serial Number** in the Registration Dataset
- DeviceModel – from **Device Model** in the Registration Dataset

Purchases

- PurchaseID – uniquely identifies Purchase records
- PurchaseDate – from **Purchase Date** in the Registration Dataset
- PurchaseStoreName – from **Purchase Store Name** in the Registration Dataset

- PurchaseStoreCity – from **Purchase Store City** in the Registration Dataset
- PurchaseStoreState – from **Purchase Store State** in the Registration Dataset
- boolean ECommFlag – from **Ecomm Flag** in the Registration Dataset
- CustomerID – from **Customer ID** in the Registration Dataset
- DeviceID – references Devices
- DeviceModel – from **Device Model** in the Registration Dataset

Registrations

- RegistrationID – from **Registration ID** in the Registration Dataset
- RegistrationDate – from **Registration Date** in the Registration Dataset
- RegistrationSourceID – from **Registration Source ID** in the Registration Dataset
- RegistrationSourceName – from **Registration Source Name** in the Registration Dataset
- DeviceModel – from **Device Model** in the Registration Dataset
- CustomerID – from **Customer ID** in the Registration Dataset
- DeviceID – references Devices