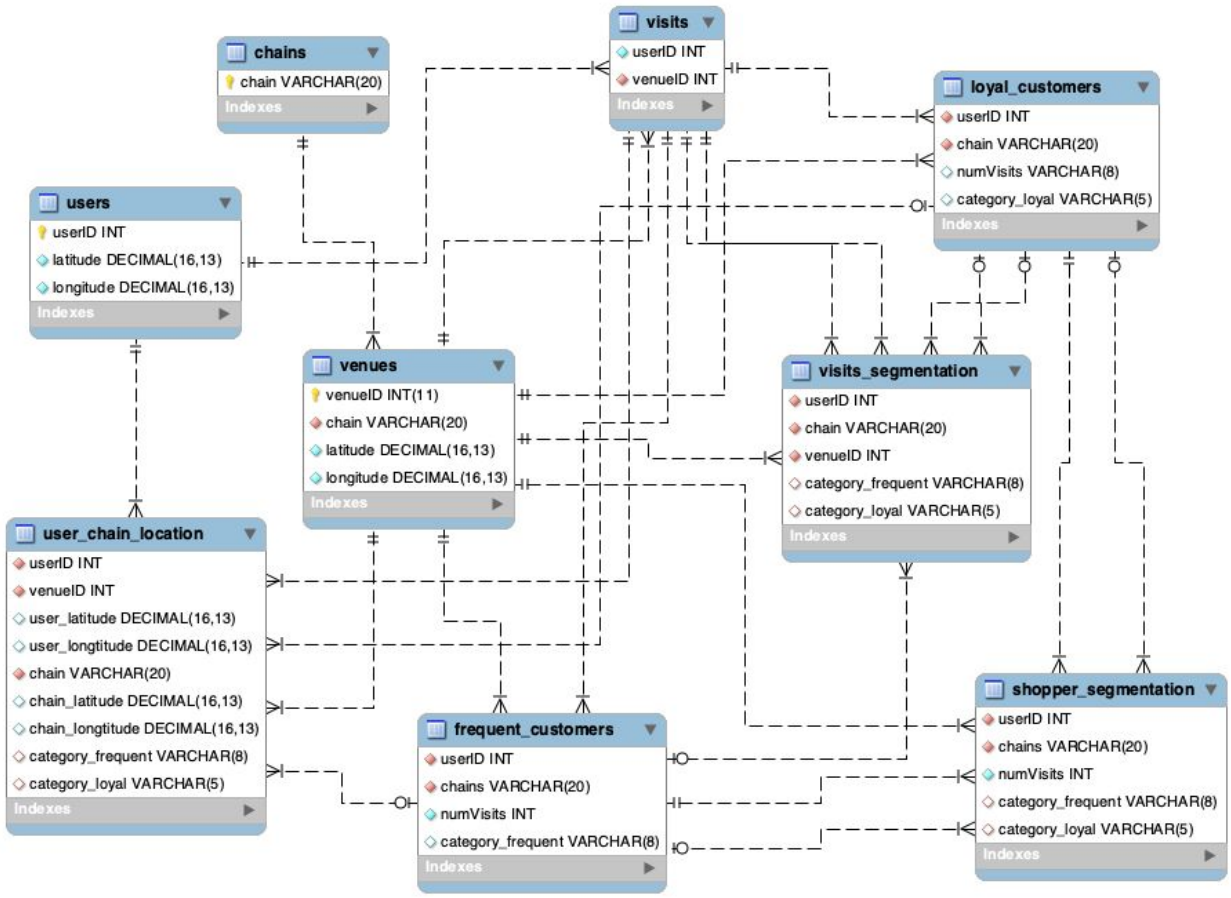
Jiakang Chen, Yun Xiao, Chengyu Jiang, and Jade Sinskul
Prof. Mohr
CIS467 - Data Warehousing
2 December 2020

Excel Inc Mobile Data Case: Data Warehouse Documentation

1. EER Diagram for Excel's Location Analytics



2. Data Dictionary

Users: a customer who only shops at a specific chain

a. user ID (PK) – numeric identifier; INT
b. latitude – latitude of home in decimal degrees, estimated; decimal(16,13)
c. longitude – longitude of home in decimal degrees, estimated; decimal(16,13)

Chains: a customer who only shops at a specific chain

a. chain (PK) – name of chain corresponding to venueName; VARCHAR(20)

Venues: a customer who only shops at a specific chain

a. venueID - numeric identifier; INT
b. chain (FK) – name of chain corresponding to venueName; VARCHAR(20)
c. latitude – latitude of chain in decimal degrees, estimated; decimal(16,13)
d. longitude – longitude of chain in decimal degrees, estimated;

Visits: a customer who only shops at a specific chain

a. user ID (FK) – numeric identifier; INT
b. venueID - numeric identifier; INT

Loyal Customers: a customer who only shops at a specific chain

a. user ID (FK) – numeric identifier; INT
b. chain (FK) – name of chain corresponding to venueName; VARCHAR(20)
c. numVisits - number of visits, count each visits by a certain user who went to a certain chain; INT
d. category_loyal: whether a customer is loyal or not; VARCHAR(5)

Frequent Customers: a customer who had shopped at a particular chain at least two times per year

a. user ID (FK) – numeric identifier; INT
b. chain (FK) – name of chain corresponding to venueName; VARCHAR(20)
c. numVisits - number of visits, count each visits by a certain user who went to a certain chain; INT
d. category_frequent: whether a customer is frequent or not; VARCHAR(8)

Shopper Segmentation: Whether a shopper is a frequent shopper or not, and whether a shopper is loyal to a specific chain or not

a. user ID (FK) – numeric identifier; INT
b. chain (FK) – name of chain corresponding to venueName; VARCHAR(20)
c. numVisits - number of visits, count each visits by a certain user who went to a certain chain; INT
d. category_frequent: whether a customer is frequent or not; VARCHAR(8)
e. category_loyal: whether a customer is loyal or not; VARCHAR(5)

Visits Segmentation: Whether a visit is by a shopper who is frequent or not, and whether the shopper is loyal or not

a. user ID (FK) – numeric identifier; INT
b. chain (FK) – name of chain corresponding to venueName; VARCHAR(20)
c. venueID - numeric identifier; INT
d. category_frequent: whether a customer is frequent or not; VARCHAR(8)
e. category_loyal: whether a customer is loyal or not; VARCHAR(5)

User Chain Location: Each shopper's home location and the chain's location the shopper visited

a. user ID (FK) – numeric identifier; INT
b. user_latitude – latitude of home in decimal degrees, estimated; decimal(16,13)
c. user_longitude – longitude of home in decimal degrees, estimated; decimal(16,13)
d. venueID (FK) – numeric identifier; INT
e. chain (FK) – name of chain corresponding to venueName; VARCHAR(20)
f. chain_latitude – latitude of chain in decimal degrees, estimated; decimal(16,13)
g. chain_longitude – longitude of chain in decimal degrees, estimated; decimal(16,13)

h. category_frequent: whether a customer is frequent or not; VARCHAR(8)
i. category_loyal: whether a customer is loyal or not; VARCHAR(5)

3. Created a Federated Database

```
CREATE SERVER mobilevisits
FOREIGN DATA WRAPPER mysql
OPTIONS (USER 'gba424_student', HOST 'gba424.simon.rochester.edu', PASSWORD
'Student!2020', DATABASE 'mobilevisits');

DROP DATABASE IF EXISTS mobilevisits_federated;
CREATE DATABASE mobilevisits_federated;

USE mobilevisits_federated;

DROP TABLE IF EXISTS chains_fed;
CREATE TABLE chains_fed(
  chain varchar(20) NOT NULL,
  PRIMARY KEY ( chain )
)
ENGINE = FEDERATED
CONNECTION = 'mobilevisits/chains'
;

DROP TABLE IF EXISTS users_fed;
CREATE TABLE users_fed(
  userID int(11) PRIMARY KEY NOT NULL,
  latitude decimal(16,13),
  longitude decimal(16,13)
)
ENGINE = FEDERATED
CONNECTION = 'mobilevisits/users'
;

DROP TABLE IF EXISTS venues_fed;
CREATE TABLE venues_fed(
  venueID int(11) PRIMARY KEY NOT NULL,
  chain varchar(20),
  latitude decimal(16,13),
  longitude decimal(16,13),
  CONSTRAINT venues_fed_fk_chains_fed
    FOREIGN KEY (chain)
       REFERENCES chains_fed (chain)
)
ENGINE = FEDERATED
```

```
CONNECTION = 'mobilevisits/venues'
;

DROP TABLE IF EXISTS visits_fed;
CREATE TABLE visits_fed(
  userID int(11),
  venueID int(11),
  INDEX (userID),
  FOREIGN KEY (userID) REFERENCES users_fed (userID),
  INDEX (venueID),
  FOREIGN KEY (venueID) REFERENCES venues_fed (venueID)
)
ENGINE = FEDERATED
CONNECTION = 'mobilevisits/visits'
;

SELECT * FROM chains_fed;
SELECT * FROM users_fed;
SELECT * FROM venues_fed;
SELECT * FROM visits_fed;


CREATE DATABASE mobilevisits_local;
USE mobilevisits_local;

CREATE TABLE chains_local
AS
SELECT * FROM mobilevisits_federated.chains_fed;

CREATE TABLE users_local
AS
SELECT * FROM mobilevisits_federated.users_fed;

CREATE TABLE venues_local
AS
SELECT * FROM mobilevisits_federated.venues_fed;

CREATE TABLE visits_local
AS
SELECT * FROM mobilevisits_federated.visits_fed;


-- Data Warehousing:

-- Frequent customers
CREATE OR REPLACE VIEW frequent_customers AS
```

```
SELECT v.userID, COUNT(v.venueID) AS numVisits, ve.chain, 'Frequent' AS
Category_Frequent
FROM visits_local v LEFT JOIN venues_local ve ON v.venueID = ve.venueID
GROUP BY v.userID,ve.chain
HAVING numVisits >2;

-- Loyal customers
CREATE OR REPLACE VIEW loyal_customers AS
SELECT t1.userID, t1.numVisits, t1.chain ,'Loyal' AS Category_Loyal FROM(
SELECT u.userID, COUNT(v.venueID) AS numVisits, ve.chain
FROM users_local u LEFT JOIN visits_local v ON u.userID = v.userID
    LEFT JOIN venues_local ve ON v.venueID = ve.venueID
GROUP BY v.userID, ve.chain
) t1
GROUP BY t1.userID
HAVING COUNT(t1.userID)=1 ;

-- shoppers each chain segmentation by frequent and loyal
CREATE OR REPLACE VIEW shopper_each_chain AS
SELECT v.userID , ve.chain, COUNT(v.venueID) AS numVisits
FROM visits_local v LEFT JOIN venues_local ve ON v.venueID = ve.venueID
GROUP BY v.userID, ve.chain;

CREATE OR REPLACE VIEW shopper_segmentation AS
SELECT s.userID, s.chain, s.numVisits,f.Category_Frequent, l.Category_Loyal
FROM shopper_each_chain s LEFT JOIN frequent_customers f ON s.userID = f.userID AND
s.chain = f.chain
    LEFT JOIN loyal_customers l ON l.userID = s.userID AND l.chain = s.chain
;

-- Visits each chain segmentation by frequent and loyal
CREATE OR REPLACE VIEW visit_each_chain AS
SELECT v.userID, v.venueID, c.chain
FROM visits_local v LEFT JOIN venues_local ve ON v.venueID = ve.venueID
GROUP BY v.userID, v.venueID;

CREATE OR REPLACE VIEW visits_segmentation AS
SELECT v.userID, v.venueID, v.chain, f.Category_Frequent
FROM visit_each_chain v LEFT JOIN frequent_customers f ON v.userID=f.userID AND
v.chain=f.chain
    LEFT JOIN loyal_customers l ON l.userID=v.userID AND l.chain=v.chain
;

-- Visits by chain location and user location
CREATE OR REPLACE VIEW user_chain_location AS
SELECT u.userID, u.latitude AS user_latitude, u.longitude AS user_longitude,
```

```
    v.venueID, ve.chain, ve.latitude AS chain_latitude, ve.longitude AS chain_longitude,
    f.Category_Frequent, l.Category_Loyal
FROM users_local u JOIN visits_local v ON u.userID = v.userID
  JOIN venues_local ve ON v.venueID = ve.venueID
  LEFT JOIN frequent_customers f ON u.userID = f.userID AND ve.chain = f.chain
  LEFT JOIN loyal_customers l ON l.userID = u.userID AND l.chain = ve.chain;
```