University of British Columbia, Department of Computer Science

CPSC 304

Cover Page for Project Part 2 – Logical Design

Date: October 14, 2018

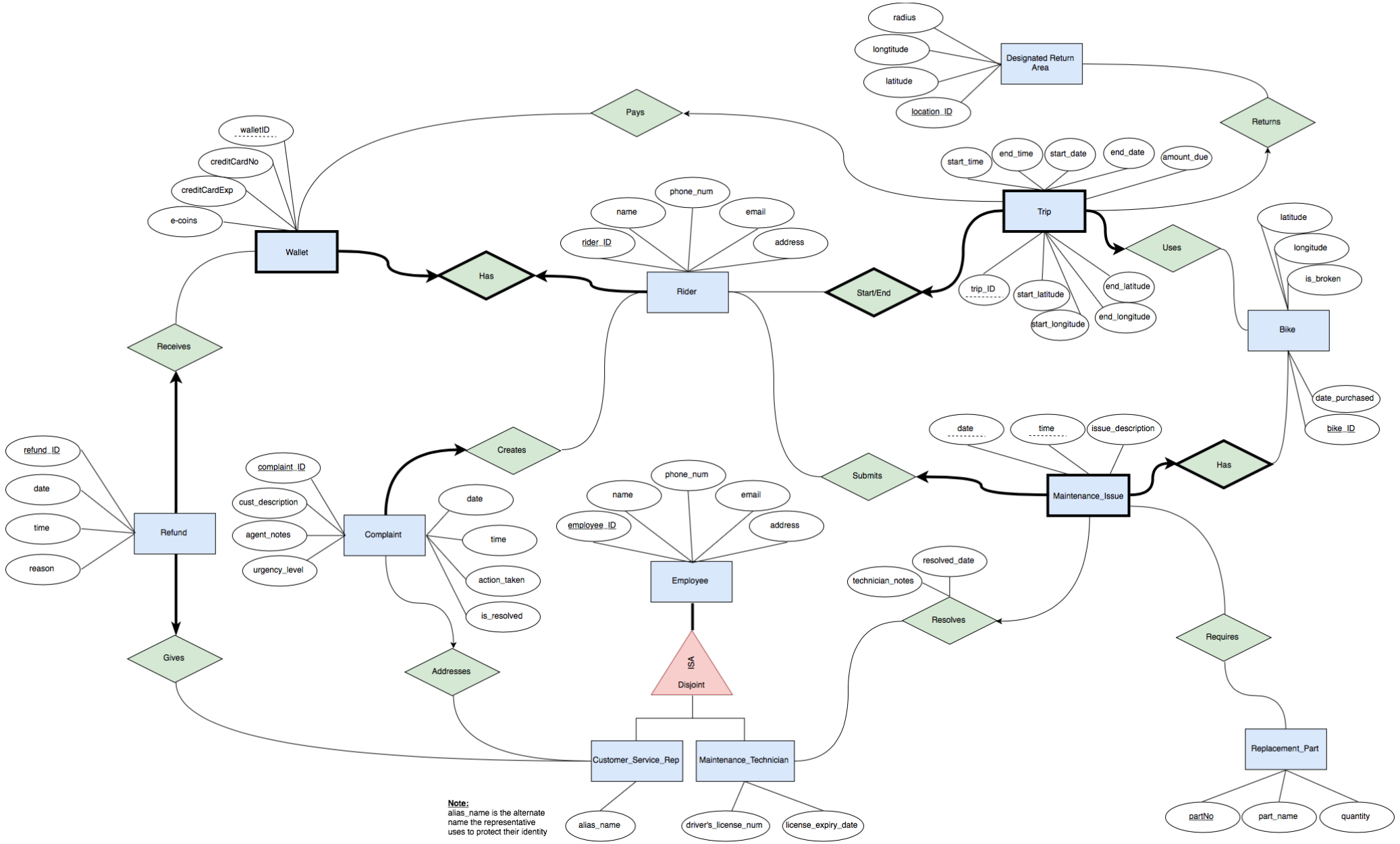
Group Members:

Name	Student Number	CS Userid	Tutorial Section	Email Address
Raghav Thakur	60250157	f4l0b	T1B	raghav.thakur.rt@gmail.com
Jacques Marais	17372079	r4n6	T1G	j.marais@alumni.ubc.ca
Daniel Ng	57421133	a1c9	T1E	dandanmng@gmail.com
Ze Yu Li	49330160	o0w0b	T1A	kevinli19980207@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 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

CPSC 304 – Group 23 – Bike Sharing System



Note:
alias_name is the alternate name the representative uses to protect their identity

CPSC 304 - Group 23
Relational Schemas, Functional Dependencies, and SQL DDL

NOTE: Through identifying the functional dependencies of our tables, it was confirmed that all tables are in BCNF and no decomposition is needed.

Replacement_Part(partNo: Int,
part_name: Char(20),
quantity: Int)
Primary key: partNo
Foreign key: none

Replacement_Part:

- Functional dependencies:
  - partNo → part_name, quantity
- Candidate keys:
  - partNo
- Foreign keys:
  - none

```
CREATE TABLE Replacement_Part(
    partNo          INTEGER,
    part_name       CHAR(20),
    quantity        INTEGER,
    PRIMARY KEY (partNo))
```

Requires(**partNo**: Int,
date: Char(10),
time: Char(10),
bike_ID: Int)
Primary key: (partNo, date, time, bike_ID)
Foreign key: partNo references Replacement_Part
(date, time, bike_ID) references Maintaince_Issue

Requires:

- Functional dependencies:
  - bike_ID, date, time, partNo → bike_ID, date, time, partNo
- Candidate keys:
  - (bike_ID, date, time, partNo)

- Foreign keys:
    - (bike_ID, date, time) references Maintenance_Issue
    - partNo references Replacement_Part

```
CREATE TABLE Requires(
        partNo          INTEGER,
        date            CHAR(10),
        time            CHAR(10),
        bike_ID         INTEGER,
        PRIMARY KEY (partNo, date, time, bike_ID),
        FOREIGN KEY (partNo) REFERENCES Replacement_Part,
                ON DELETE NO ACTION
                ON UPDATE CASCADE
        FOREIGN KEY (date, time, bike_ID) references Maintenance_Issue,
                ON DELETE CASCADE
                ON UPDATE CASCADE)
```

Maintenance_Issue(date: Char(10),
time: Char(10),
issue_description: Char(50),
technican_notes: Char(50),
resolved_date: Char(10),
bike_ID: Int,
rider_ID: Int,
technician_ID, Int)
Primary Key: (date, time, bike_ID)
Foreign Key: bike_ID references Bike
rider_ID references Rider where not null
technician_ID references Maintenance_Technician where not null

Maintenance_Issue

- Functional dependencies:
    - bike_ID, date, time → issue_description, rider_ID, resolved_date, technician_notes, technician_ID
- Candidate keys:
    - (bike_ID, date, time)
- Foreign keys:
    - bike_ID references Bike
    - ride_ID references Rider
    - technician_ID references Maintenance_Technician

```
CREATE TABLE Maintenance_Issue(
        date                CHAR(10),
        time                CHAR(10),
        issue_description   Char(50),
        technican_notes     CHAR(50),
        resolved_date       CHAR(10),
        bike_ID             INTEGER,
        rider_ID            INTEGER NOT NULL,
        technician_ID       INTEGER NOT NULL,
        PRIMARY KEY (date, time, bike_ID),
        FOREIGN KEY (bike_ID) REFERENCES Bike,
                ON DELETE CASCADE
                ON UPDATE CASCADE
        FOREIGN KEY (rider_ID) REFERENCES Rider,
                ON DELETE NO ACTION
                ON UPDATE CASCADE
        FOREIGN KEY (techinician_ID) REFERENCES Maintenance_Technician,
                ON DELETE NO ACTION
                ON UPDATE CASCADE)
```

Customer_Service_Rep(employee_ID: Int,
name: Char(20),
phone_num: Int,
email: Char(20),
address: Char(20),
alias_name: Char(20))

Primary key: employee_ID
Foreign key: none

Customer_Service_Rep

- Functional dependencies:
  - employee_ID → name, phone_num, email, address, alias_name
- Candidate keys:
  - employee_ID
- Foreign keys:
  - none

```
CREATE TABLE Customer_Service_Rep(
        employee_ID         INTEGER,
        name                CHAR(20),
        phone_num           INTEGER,
        email               CHAR(20),
```

address CHAR(20),
alias_name CHAR(20),
PRIMARY KEY (employee_ID))

Maintenance_Technician(employee_ID: Int,
name: Char(20),
phone_num: Int,
email: Char(20),
address: Char(20),
driver's_license_num: Int
license_expiry_date: Char(10))
Primary key: employee_ID
Foreign key: none

Maintenance_Technician

- Functional dependencies:
  - employee_ID → name, phone_num, email, address, driver's_license_num, license_expiry_date
- Candidate keys:
  - employee_ID
- Foreign keys:
  - none

CREATE TABLE Maintenance_Technician(
employee_ID INTEGER,
name CHAR(20),
phone_num INTEGER,
email CHAR(20),
address CHAR(20),
driver's_license_num INTEGER,
license_expiry_date CHAR(10),
PRIMARY KEY (employee_ID))

Complaint(complaint_ID: Int,
rider_ID: Int,
Customer_Service_Rep_ID: Int,
cust_description: Char(50),
agent_notes: Char(50),
urgency_level: Char(10),

date: Char(10),
time: Char(10),
action_taken: Char(10),
is_resolved: Char(10))
Primary Key: complaint_ID
Foreign Key: rider_ID references rider where not null
Customer_Service_Rep_ID references Customer_Service_Rep

Complaint:

- Functional dependencies:
  - complaint_ID → rider_ID, Customer_Service_Rep_ID, cust_description, agent_notes, urgency_level, date, time, action_taken, is_resolved
- Candidate keys:
  - complaint_ID
- Foreign keys:
  - rider_ID references rider
  - Customer_Service_Rep_ID references Customer_Service_Rep_ID

```
CREATE TABLE Complaint(
        complaint_ID        INTEGER,
        rider_ID            INTEGER NOT NULL,
        employee_ID         INTEGER,
        cust_description    CHAR(50),
        agent_notes         CHAR(50),
        urgency_level       CHAR(10),
        date                CHAR(10),
        time                CHAR(10),
        action_taken        CHAR(10),
        is_resolved         CHAR(10),
        PRIMARY KEY (complaint_ID),
        FOREIGN KEY (rider_ID) REFERENCES rider,
                ON DELETE NO ACTION
                ON CASCADE UPDATE
        FOREIGN KEY (Customer_Service_Rep_ID) REFERENCES Customer_Service_Rep,
                ON DELETE NO ACTION
                ON CASCADE UPDATE)
```

Designated_Return_Area(location_ID: Int,
latitude: Int,
longitude: Int,
radius: Int)
Primary Key: location_ID

Foreign Key: None

Designated_Return_Area:

- Functional dependencies:
  - location_ID → latitude, longitude, radius
- Candidate keys:
  - location_ID
- Foreign keys:
  - none

```
CREATE TABLE Designated_Return_Area (
        location_ID     INTEGER,
        latitude        INTEGER,
        longitude       INTEGER,
        radius          INTEGER,
        PRIMARY KEY (location_ID))
```

***Note: We have combined Rider table and Wallet table because a Rider has exactly one Wallet and Wallet is associated with exactly one rider. A Wallet cannot exist without a Rider because it is a weak entity.**

Rider(rider_ID: Int,
wallet_ID: Int,
name: Char(20),
phone_num: Int,
email: Char(20),
address: Char(20), creditCardNo: Int, creditCardExp: Int, e-coins: Int)
Primary key: rider_ID

Rider:

- Functional dependencies:
  - rider_ID → wallet_ID, name, phone_num, email, address, creditCardNo, creditCardExp, e-coins
- Candidate keys:
  - rider_ID
- Foreign keys:
  - none

```
CREATE TABLE rider(
        rider_ID        INTEGER,
```

```
    wallet_ID       INTEGER NOT NULL UNIQUE,
    name            CHAR(20),
    phone_num       INTEGER,
    email           CHAR(20),
    address         CHAR(20),
    creditCardNo    INTEGER,
    creditCardExp INTEGER,
    e-coins         INTEGER,
    PRIMARY KEY (rider_ID))
```

Bike(bike_ID: Int, date_purchased: Int, longitude: Int, latitude: Int, is_broken: Int)
Primary key: bike_ID
Foreign key: none

Bike:

- Functional dependencies:
    - bike_ID → date_purchased, longitude, latitude, is_broken
- Candidate keys:
    - bike_ID
- Foreign keys:
    - none

```
CREATE TABLE bike (
    bike_ID             INTEGER,
    date_purchased      INTEGER,
    longitude           INTEGER,
    latitude            INTEGER,
    is_broken           INTEGER,
    PRIMARY KEY (bike_ID))
```

Trip (trip_ID: Int, **rider_ID**: Int, **bike_ID**: Int, **location_ID**: Int, start_time: Char(10), end_time: Char(10), start_date: Char(10), end_date: Char(10), amount_due: Char(10), start_latitude: Char(20), end_latitude: Char(20), start_longitude: Char(20), end_longitude: Char(20))
Primary Key: (trip_ID, rider_ID)
Foreign Key: bike_ID references bike where not null and unique
rider_ID references rider
location_ID references Designated Return Area

Trip:

- Functional dependencies:
  - trip_ID, rider_ID → bike_ID, location_ID, start_time, end_time, start_date, end_date, amount_due, start_latititude, end_latitude, start_longitude, end_longitude
- Candidate keys:
  - (trip_ID, rider_ID)
- Foreign keys:
  - bike_ID references Bike
  - rider_ID references Rider
  - location_ID references Designated_Return_Area

```
CREATE TABLE Trip(
        trip_ID             INTEGER,
        rider_ID            INTEGER,
        bike_ID             INTEGER NOT NULL UNIQUE,
        location_ID         INTEGER,
        start_time          CHAR(10),
        end_time            CHAR(10),
        start_date          CHAR(10),
        end_date            CHAR(10),
        amount_due          CHAR(10),
        start_latitude      CHAR(20),
        end_latitude        CHAR(20),
        start_longitude     CHAR(20),
        end_longitude       CHAR(20),
        PRIMARY KEY (trip_ID),
        FOREIGN KEY (rider_ID) REFERENCES rider,
                ON DELETE NO ACTION
                ON UPDATE CASCADE
        FOREIGN KEY (bike_ID) REFERENCES bike,
                ON DELETE NO ACTION
                ON UPDATE CASCADE
        FOREIGN KEY (location_ID) REFERENCES location,
                ON DELETE SET NULL
                ON UPDATE CASCADE)
```

Refund(refund_ID: Int, **rider_ID**: Int, **Customer_Service_Rep_ID**: Int, date: Char(10), time: Char(10), reason: Char(50))
Primary Key: refund_ID
Foreign Key: rider_ID references rider where not null
Customer_Service_Rep_ID references Customer_Service_Rep where not null

Refund

- Functional dependencies:
  - refund_ID → date, time, reason, rider_ID, Customer_Service_Rep_ID
- Candidate keys:
  - refund_ID
- Foreign keys:
  - rider_ID references rider
  - Customer_Service_Rep_ID references Customer Service Representative

```
CREATE TABLE refund(
        refund_ID       INTEGER,
        rider_ID        INTEGER NOT NULL,
        employee_ID     INTEGER NOT NULL,
        date            CHAR(20),
        time            CHAR(20),
        reason          CHAR(50),
        PRIMARY KEY (refund_ID),
        FOREIGN KEY (rider_ID) REFERENCES rider,
                ON DELETE NO ACTION
                ON UPDATE CASCADE
        FOREIGN KEY (Customer_Service_Rep_ID) REFERENCES Customer_Service_Rep
                ON DELETE NO ACTION
                ON UPDATE CASCADE)
```