CPSC 304 Project Cover Page

Milestone #: 2
Date: October 15, 2024
Group Number: 93

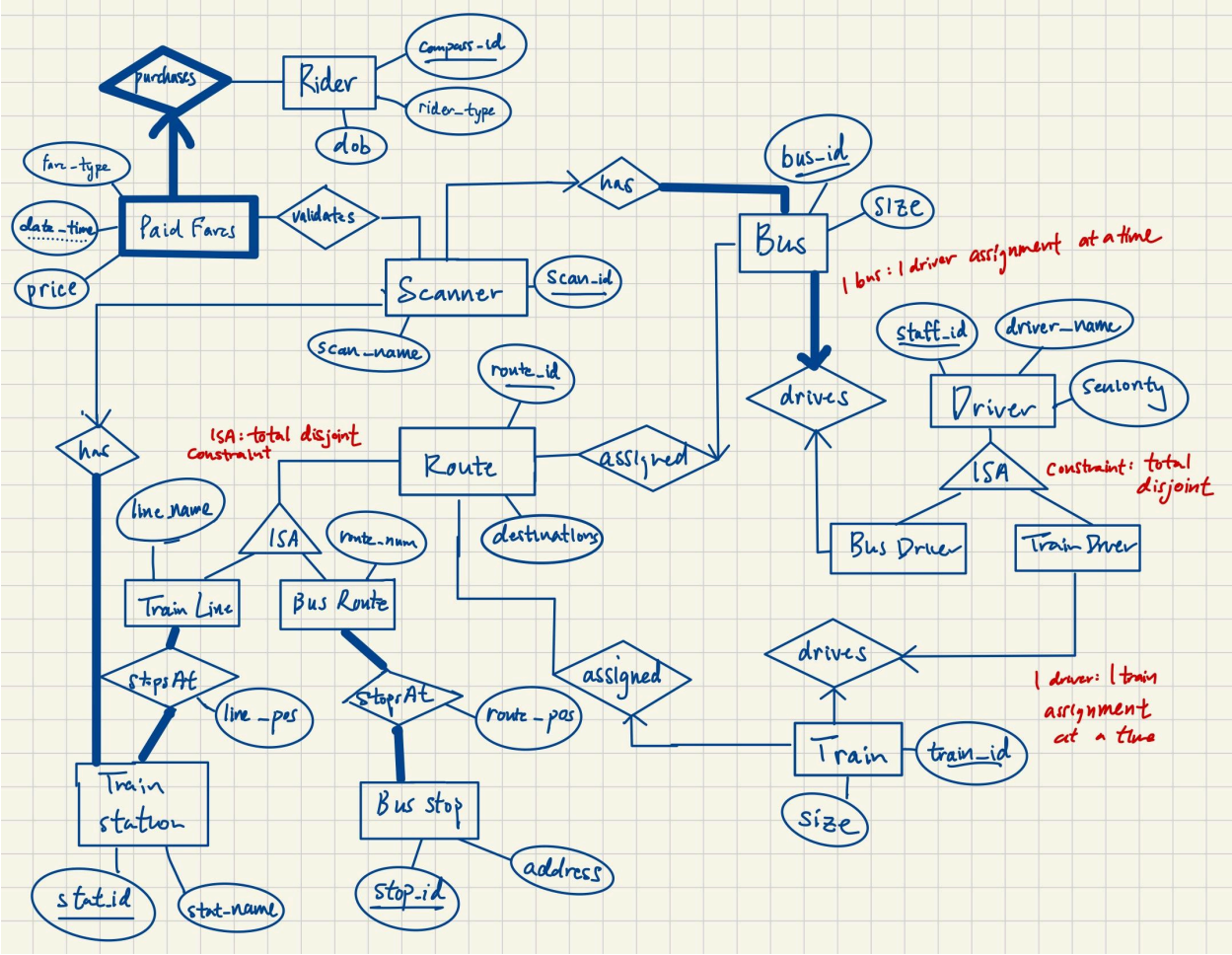
Name	Student Number	CS Alias (Userid)	Preferred Email Address
Sabrina Lou	17539495	t1e8c	sabrinajwlou@gmail.com
Rohan Shukla	80120785	c7a8v	shuklarohan@live.com
Freddi Li	51907897	f5q5r	winnifredsli38@gmail.com

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.

Project Description
Our project is a public transportation system, used within the transportation management sector. It focuses on the operations, logistics, and staff management within a public transit system, based on the public transport available in the Metro Vancouver area. This includes managing routes, vehicles, stations and stops, drivers, riders, and fare payment.

ER Diagram



Changes
- unbolded StopsAt relationship for Train station and Bus stop
- added non-primary key attributes to Train, Rider, and Scanner
- clarified that the one-to-one Bus <-> Bus Driver and Train <-> Train Driver relationship represents active driver assignments (ie. even though drivers may drive more than one bus/train, we will only store "active assignments" where one driver has one bus/train)
- removed incorrect ISA primary key
- clarified ISA overlap constraints (both total disjoint)
- changed weak entity relationship so Paid Fares belongs to Rider

4. Create tables

TrainLineStopsAt(**stat_id: int, rid: int,** line_name: VARCHAR, line_pos: int)
- PK: (stat_id, rid)
- CKs: (stat_id, line_name), (line_name, line_pos), (route_id, line_pos), (route_id, line_name, line_pos)
- line_name, line_pos are not null
- Constraints: can't model participation constraints using Null restrictions

BusRouteStopsAt(**stop_id: int, rid: int,** address: VARCHAR, route_name: VARCHAR, route_pos: int)
- PK: (stop_id, rid)
- CKs: (stop_id, rid), (stop_id, route_name), (address, route_name), (address, rid)
- address, route_name, route_pos are not null
- Constraints: can't model participation constraints using Null restrictions

TrainStation(stat_id: int, stat_name: VARCHAR)
- PK: stat_id

Route**(**rid: int, destination: VARCHAR)
- PK: rid

Rider(compass_id:int,rider_type: VARCHAR,dob: VARCHAR)
- PK: compass_id
- FK:N/A
- CK:N/A
- Constraints: N/A

Scanner(scan_id: int, scan_type: VARCHAR)
- PK: scan_id
- FK:N/A
- CK:N/A
- Constraints: scan_type is not Null

Paid fares(**compass_id:int**, date_time:VARCHAR,fare_type:VARCHAR,price:DECIMAL)
- PK: compass_id, date_time
- FK:compass_id
- CK:N/A
- Constraints: on delete cascade

ValidateFare(**scan_id:int**,**compass_id:int**,**date_time:VARCHAR**)
- PK: compass_id, date_time, scan_id
- FK:compass_id, date_time, scan_id
- CK:N/A
- Constraints: on delete cascade

BusHasScanner(**scan_id:int,bus_id:int**)
- PK: scan_id
- FK:scan_id, bus_id
- CK:N/A
- Constraints: can't model participation constraints using Null restrictions

TrainStationHasScanner(**scan_id:int, station_id:int**)
- PK: scan_id
- FK:scan_id, station_id
- CK:N/A
- Constraints: can't model participation constraints using Null restrictions

Driver(staff_id: int, name: VARCHAR, seniority: VARCHAR)

PK: staff_id
FK: N/A
CK: N/A

BusDriver(**staff_id: int**)
PK: staff_id
FK: staff_id
CK: N/A

TrainDriver(**staff_id: int**)
PK: staff_id
FK: staff_id
CK: N/A

Bus(bus_id: int, bus_size: int)
PK: bus_id
FK: N/A
CK: bus_id

Train(train_id: int, train_size: int)
PK: train_id
FK: N/A
CK: train_id

DrivesBus(**bus_id: int**, **staff_id: int**)
PK: bus_id
FK: N/A
CK: bus_id, staff_id

DrivesTrain(**train_id: int, staff_id: int**)
PK: train_id
FK: N/A
CK: train_id, staff_id

BusAssigned(**bus_id: int, route_id: int**)
PK: bus_id
FK: N/A
CK: bus_id

TrainAssigned(**train_id: int, route_id: int**)
PK: train_id
FK: N/A
CK: train_id

5. Functional Dependencies

TrainLineStopsAt(**stat_id: int**, **rid: int,** line_name: VARCHAR, line_pos: int)
stat_id, route_id-> line_pos, line_name
stat_id, line_name -> line_pos, rid
route_id, line_name, line_pos -> stat_id
line_name, line_pos -> stat_id
route_id, line_pos -> stat_id
route_id-> line_name

BusRouteStopsAt(**stop_id: int, rid: int,** address: VARCHAR, route_name: VARCHAR, route_pos: int)

stop_id, route_id -> route_pos, route_num
stop_id, route_num -> route_pos, rid
address, route_num -> route_pos
route_id, route_num, route_pos -> stop_id
sid -> route_num
stop_id -> address
address -> stop_id

TrainStation(stat_id: int, stat_name: VARCHAR)
stat_id -> stat_name

Route**(**rid: int, destination: VARCHAR)
route_id -> destination

Rider(compass_id:int,rider_type: VARCHAR,dob: VARCHAR)
Compass_id -> rider_type
Compass_id -> dob
dob -> rider_type
Scanner(Scan_id: int, scan_type: VARCHAR)
Scan_id -> scan_type
Paid fares(**compass_id:int**, date_time:VARCHAR,fare_type:VARCHAR,price:int)
fare_type->price
Compass_id, date_time -> fare_type
Compass_id, date_time -> price
ValidateFare(**scan_id**,**compass_id**,**date_time**)
N/A
BusHasScanner(**scan_id,bus_id**)
scan_id->bus_id
TrainStationHasScanner(**scan_id, station_id**)
scan_id->station_id
Driver(staff_id, name, seniority)
staff_id->name, seniority
BusDriver(**staff_id**)
no non-trivial functional dependencies
TrainDriver(**staff_id**)
no non-trivial functional dependencies
Bus(bus_id, bus_size)
bus_id->bus_size
Train(train_id, train_size)
train_id->train_size
DrivesBus(**bus_id**, **staff_id**)
bus_id->staff_id
staff_id->bus_id
DrivesTrain(**train_id, staff_id**)
train_id->staff_id
staff_id->train_id
BusAssigned(**bus_id, route_id**)
bus_id->route_id
TrainAssigned(**train_id, route_id**)
train_id->route_id

6. Normalization(into 3NF)

Finding Minimal Cover (bolded are minimal FDs in standardized form)

TrainLineStopsAt
Stat_id, route_id -> line_pos
Stat_id, route_id -> line_name
Stat_id, line_name -> line_pos
Stat_id, line_name -> rid
route_id, line_name, line_pos -> stat_id
route_id, line_pos -> stat_id
Line_name, line_pos -> stat_id
route_id -> line_name
Key: {stat_id, rid}+ = {stat_id, route_id, line_pos, line_name}

TrainStation
stat_id -> stat_name

Route
route_id -> destination

BusRouteStopsAt
Stop_id, route_id -> route_pos
Stop_id, route_id -> route_num
Stop_id, route_num -> route_pos
Stop_id, route_num -> rid
route_id, route_num, route_pos -> stop_id
Address, route_num -> route_pos
Address, route_id -> route_pos
route_id -> route_num
Stop_id -> address
Address -> stop_id
Key: {stop_id, rid}+ = {stop_id, route_id, route_pos, route_num, address}

Rider
Compass_id -> rider_type
Compass_id -> dob
dob -> rider_type

Paid fares
fare_type->price
Compass_id, date_time -> fare_type
Compass_id, date_time -> price

All tables (original and after decomposition)

TrainStation(stat_id: int, stat_name: VARCHAR)

Route**(**rid: int, destination: VARCHAR)

TrainLineStopsAt(**stat_id: int**, **rid: int,** line_name: VARCHAR, line_pos: int) ->
TrainLineStopsAt1(**stat_id: int**, **rid: int**, line_pos: int)
TrainLineStopsAt2(**rid: int**, line_name: VARCHAR) //represents train line -> TrainLine()

// not null restriction cannot model total participation

BusRouteStopsAt(**stop_id: int**, address: VARCHAR, route_name: VARCHAR, route_pos: int, **rid: int**) ->
BusRouteStopsAt1(**stop_id: int**, **rid: int**, route_pos: int)
BusRouteStopsAt2(**rid: int**, route_num: int) //represents bus route -> BusRoute()
BusRouteStopsAt3(**stop_id**, address) //represents bus stop entity -> BusStop()

// not null restriction cannot model total participation

Rider(compass_id:int, rider_type: VARCHAR, dob: VARCHAR)

Rider1(compass_id:int, dob: VARCHAR) //decomposed table

Rider2(rider_type:VARCHAR, dob: VARCHAR) //decomposed table

Scanner(Scan_id: int, scan_type: VARCHAR)

Paid fares(**compass_id:int**, date_time: VARCHAR,fare_type: VARCHAR, price: DECIMAL)

PaidFares1(**compass_id:int**, date_time: VARCHAR,**fare_type: VARCHAR**) //decomposed table

PaidFares2(fare_type: VARCHAR, price:int) //decomposed table

ValidateFare(**scan_id**,**compass_id**,**date_time**)

BusHasScanner(**scan_id,bus_id**)

TrainStationHasScanner(**scan_id, stat_id**)

Driver(staff_id: int, name: VARCHAR, seniority: VARCHAR)

BusDriver(**staff_id: int**)

TrainDriver(**staff_id: int**)

Bus(bus_id: int, bus_size: int)

Train(train_id: int, train_size: int)

DrivesBus(**bus_id: int**, **staff_id: int**)

DrivesTrain(**train_id: int, staff_id: int**)

BusAssigned(**bus_id: int, route_id: int**)

TrainAssigned(**train_id: int, route_id: int**)

7/8. SQL statements and INSERT

```
CREATE TABLE RIDER1(
        compass_id INTEGER,
        dob VARCHAR,
        PRIMARY KEY(compass_id)
        )

INSERT INTO RIDER1 (compass_id,dob) VALUES
(1, '01/1990'),
(2,'01/2022'),
(3, '02/1950'),
(4, '02/1950'),
(5, '06/1989')
(6, '06/2006');

CREATE TABLE RIDER2(
        rider_type VARCHAR,
        dob VARCHAR,
        PRIMARY KEY (dob)
        )

INSERT INTO RIDER2 (rider_type,dob) VALUES
('adult','01/1990'),
('infant','01/2022'),
('senior', '02/1950'),
('adult', '06/1989'),
('teen', '06/2006');


CREATE TABLE PaidFares1(
        compass_id INTEGER
        date_time VARCHAR
        fare_type VARCHAR
```

```
        PRIMARY KEY(compass_id,date_time)
        FOREIGN KEY(compass_id) REFERENCES Rider1 ON DELETE CASCADE
        FOREIGN KEY(fare_type) REFERENCES PaidFares2 ON DELETE CASCADE
)

INSERT INTO PaidFares1(compass_id,date_time) VALUES
(1,'01/01/2024 12:00'),
(1,'01/01/2024 13:00'),
(2,'02/01/2024 11:00'),
(3,'01/05/2024 08:00'),
(4,'01/06/2024 12:00'),
(5,'02/03/2024 12:50'),
(6,'01/01/2024 12:01');

CREATE TABLE PaidFares2(
        fare_type VARCHAR,
        price DECIMAL(5,2),
        PRIMARY KEY(fare_type)
)

INSERT INTO PaidFares2(fare_type, price) VALUES
('zone 1', 2.00),
('zone 2', 1.50),
('zone 3', 1.25),
('zone 4', 1.00)
('multi zone', 3.00);

CREATE TABLE ValidateFare(
        scan_id INTEGER,
        compass_id INTEGER,
        date_time VARCHAR,
        PRIMARY KEY(scan_id,compass_id,date_time)
        FOREIGN KEY(scan_id) REFERENCES Scanner ON DELETE CASCADE
        FOREIGN KEY(compass_id,date_time) REFERENCES
PaidFares1(compass_id,date_time) ON DELETE CASCADE
        )

INSERT INTO ValidateFare(scan_id,compass_id,date_time) VALUES
(1, 1,'01/01/2024 12:00'),
(1, 1,'01/01/2024 13:00'),
(2, 2,'02/01/2024 11:00'),
(4, 3,'01/05/2024 08:00'),
(2, 4,'01/06/2024 12:00'),
(2, 5,'02/03/2024 12:50'),
```

```
(1, 6,'01/01/2024 12:01');


CREATE TABLE Scanner(
        scan_id INTEGER,
        scan_type VARCHAR NOT NULL,
        PRIMARY KEY (rid)
        )

INSERT INTO Scanner (scan_id, scan_type) VALUES
(1, 'bus'),
(2, 'bus'),
(3, 'bus'),
(4, 'bus'),
(5, 'bus'),
(6, 'bus'),
(7, 'station'),
(8, 'station'),
(9, 'station'),
(10, 'station'),
(11, 'station'),
(12, 'station');



CREATE TABLE TrainStationHasScanner(
        scan_id INTEGER
        stat_id INTEGER
        PRIMARY KEY(scan_id)
        FOREIGN KEY(scan_id) REFERENCES Scanner ON DELETE CASCADE
        FOREIGN KEY(stat_id) REFERENCES TrainStation ON DELETE CASCADE
        )
INSERT INTO TrainStationHasScanner(scan_id, stat_id) VALUES
(7,1),
(8,1),
(9,2),
(10,3),
(11,4);


CREATE TABLE BusHasScanner(
        scan_id INTEGER
        bus_id INTEGER
        PRIMARY KEY(scan_id)
        FOREIGN KEY(scan_id) REFERENCES Scanner ON DELETE CASCADE
        FOREIGN KEY(bus_id) REFERENCES TrainStation ON DELETE CASCADE
        )
INSERT INTO BusHasScanner(scan_id, bus_id) VALUES
(1,1),
```

```
(2,1),
(3,2),
(4,3),
(5,4);



CREATE TABLE Route(
        route_id INTEGER,
        destination VARCHAR,
        PRIMARY KEY (rid)
)

INSERT INTO Route (route_id, destination) VALUES
(100, 'Downtown'),
(200, 'Uptown'),
(101, ‘Airport’),
(201, ‘Waterfront’),
(301, 'Boundary');


CREATE TABLE TrainStation(
        stat_id INTEGER,
        stat_name VARCHAR UNIQUE,
        PRIMARY KEY (stat_id)
)

INSERT INTO TrainStation (stat_id, stat_name) VALUES
(1, 'Central Station'),
(2, 'Union Station'),
(3, 'East Station'),
(4, 'West Station'),
(5, 'North Station');

CREATE TABLE BusStop(
        stop_id INTEGER,
        address VARCHAR NOT NULL UNIQUE,
        PRIMARY KEY (stop_id)
        )

INSERT INTO BusStop (stop_id, address) VALUES
(1, '123 Main St'),
(2, '456 Oak Ave'),
(3, '789 Pine St'),
(4, '101 Maple Rd'),
(5, '202 Elm St');

CREATE TABLE TrainLineStopsAt(
        stat_id INTEGER,
        route_id INTEGER,
```

```
        line_pos INTEGER NOT NULL,
        PRIMARY KEY (stat_id, rid),
        FOREIGN KEY (rid) REFERENCES Route ON DELETE CASCADE,
        FOREIGN KEY (stat_id) REFERENCES TrainStation ON DELETE CASCADE
        )

INSERT INTO TrainLineStopsAt (stat_id, route_id, line_pos) VALUES
(1, 101, 1),
(2, 101, 2),
(3, 201, 1),
(4, 201, 2),
(5, 301, 1);


CREATE TABLE TrainLine(
        route_id INTEGER,
        line_name VARCHAR NOT NULL UNIQUE,
        PRIMARY KEY (rid),
        FOREIGN KEY (rid) REFERENCES Route ON DELETE CASCADE
        )

INSERT INTO TrainLine (route_id, line_name) VALUES
(101, 'Green Line'),
(201, 'Red Line'),
(301, 'Blue Line'),
(401, 'Yellow Line'),
(501, 'Purple Line');

CREATE TABLE BusRouteStopsAt(
        stop_id INTEGER,
        route_id INTEGER,
        route_pos INTEGER NOT NULL,
        PRIMARY KEY (stop_id, rid),
        FOREIGN KEY (rid) REFERENCES Route ON DELETE CASCADE,
        FOREIGN KEY (stop_id) REFERENCES BusStop ON DELETE CASCADE
        )

INSERT INTO BusRouteStopsAt (stop_id, route_id, route_pos) VALUES
(1, 400, '1'),
(2, 400, '2'),
(1, 300, '1''),
(2, 300, '2'),
(1, 100, '1');

CREATE TABLE BusRoute(
        route_idINTEGER,
        route_num INTEGER NOT NULL UNIQUE,
        PRIMARY KEY (rid),
        FOREIGN KEY (rid) REFERENCES Route ON DELETE CASCADE
        )
```

```
INSERT INTO BusRoute (route_id, route_num) VALUES
(10, 99),
(20, 84),
(30, 25),
(40, 33),
(50, 14);

CREATE TABLE DRIVER (
        staff_id INT PRIMARY KEY,
        name VARCHAR NOT NULL,
        seniority VARCHAR NOT NULL
);

INSERT INTO Driver (staff_id, name, seniority) VALUES
(1, ‘Sabrina Lou’, ‘Junior’),
(2, ‘Freddi Li’, ‘Junior’),
(3, ‘Rohan Shukla’, ‘Junior’),
(4, ‘Rachel Pottinger’, ‘Senior’),
(5, ‘Steve Wolfman’, ‘Senior’);
(6, ‘Alan Turing’, ‘Junior’),
(7, ‘Edgar Codd’, ‘Junior’),
(8, ‘John von Neumann’, ‘Junior’),
(9, ‘Taylor Swift’, ‘Senior’),
(10, ‘Porter Robinson’, ‘Senior’);

CREATE TABLE BusDriver (
        staff_id INT PRIMARY KEY,
        FOREIGN KEY (staff_id) REFERENCES Driver(staff_id)
);

INSERT INTO BusDriver (staff_id, name, seniority) VALUES
(1),
(2),
(3),
(4),
(5);

CREATE TABLE TrainDriver (
        staff_id INT PRIMARY KEY,
        FOREIGN KEY (staff_id) REFERENCES Driver(staff_id)
);

INSERT INTO TrainDriver (staff_id, name, seniority) VALUES
(6),
(7),
(8),
(9),
(10);

CREATE TABLE Bus (
        bus_id INT PRIMARY KEY
```

```
        bus_size INT NOT NULL
);

INSERT INTO Bus (bus_id, bus_size) VALUES
(1, 47),
(2, 70),
(3, 50),
(4, 64),
(5, 24);

CREATE TABLE Train (
        train_id INT PRIMARY KEY,
        train_size INT NOT NULL
);

INSERT INTO Train (train_id, train_size) VALUES
(1, 200),
(2, 250),
(3, 300),
(4, 350),
(5, 400);

CREATE TABLE DrivesBus (
        bus_id INT,
        staff_id INT,
        PRIMARY KEY (bus_id),
        FOREIGN KEY (bus_id) REFERENCES Bus(bus_id)
        FOREIGN KEY (staff_id) REFERENCES BusDriver(staff_id)
);

INSERT INTO DrivesBus (bus_id, staff_id) VALUES)
(1, 1),
(2, 2),
(1, 3),
(2, 4),
(1, 5);

CREATE TABLE DrivesTrain (
        train_id INT,
        staff_id INT,
        PRIMARY KEY (train_id),
        FOREIGN KEY (train_id) REFERENCES Train(train_id),
        FOREIGN KEY (staff_id) REFERENCES TrainDriver(staff_id)
);

INSERT INTO DrivesTrain (train_id, staff_id) VALUES)
(1, 1),
(2, 2),
(1, 3),
(2, 4),
(1, 5);
```

```
CREATE TABLE BusAssigned (
        bus_id INT,
        route_id INT,
        PRIMARY KEY (bus_id),
        FOREIGN KEY (bus_id), REFERENCES Bus(bus_id)
        FOREIGN KEY (route_id), REFERENCES Route(route_id)
);

INSERT INTO BusAssigned (bus_id, route_id) VALUES)
(1, 1),
(2, 2),
(1, 3),
(2, 4),
(1, 5);

CREATE TABLE TrainAssigned (
        train_id INT,
        route_id INT,
        PRIMARY KEY (train_id),
        FOREIGN KEY (train_id), REFERENCES Train(train_id)
        FOREIGN KEY (route_id), REFERENCES Route(route_id)
);

INSERT INTO TrainAssigned (train_id, route_id) VALUES)
(1, 1),
(2, 2),
(1, 3),
(2, 4),
(1, 5);
```