University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: 03/01/2025

Group Number: 43

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Cian Geldenhuys	18406314	b1s1e	cgeldenh@student.ubc.ca
David Huang	44803658	m5i5x	dhuang13@student.ubc.ca
Rani Naser	35459338	e0u3r	ranuz23@student.ubc.ca

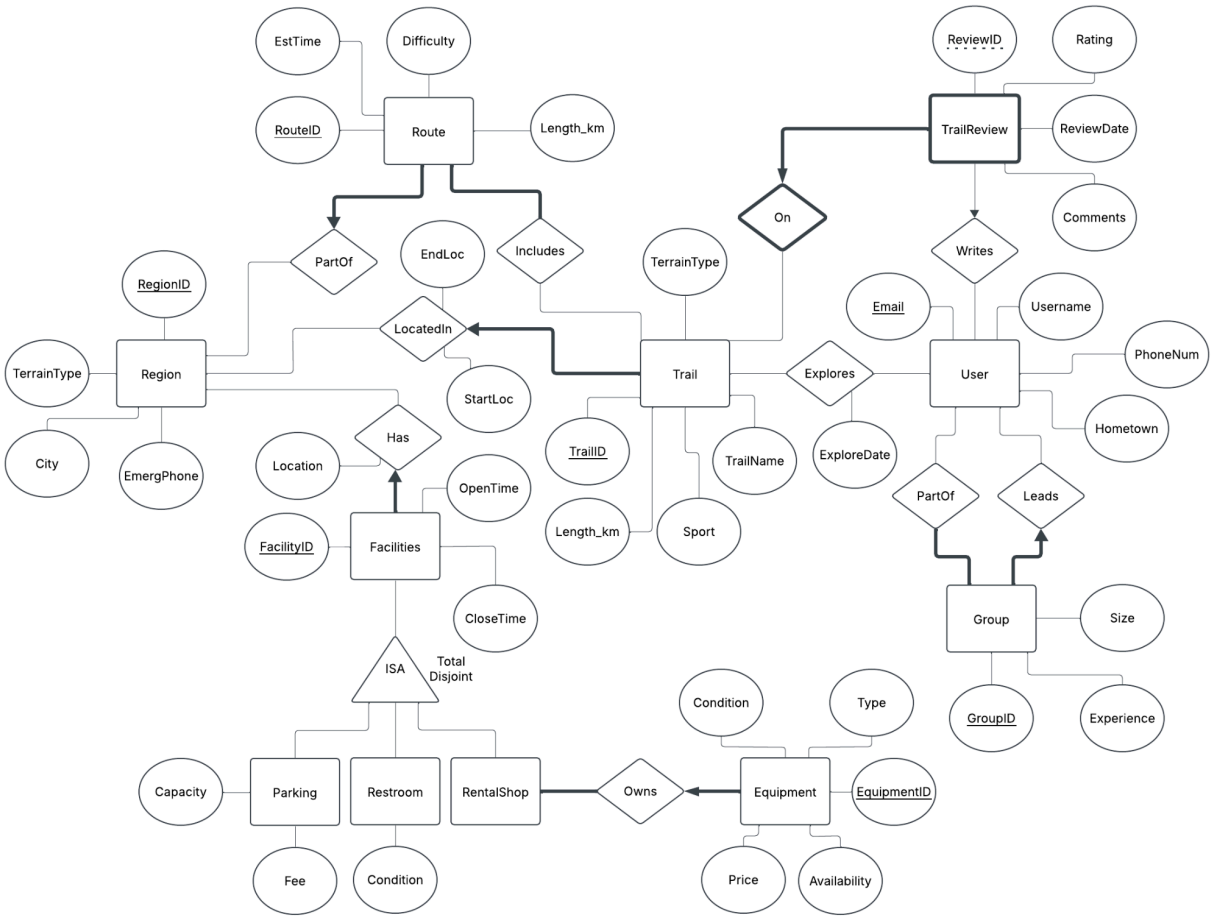
By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

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

2. Project Summary

The domain of our project is fitness, outdoor recreation, navigation, and trail management. Our application aims to provide users with a way to catalog and organize various trails, such as hiking, biking, and ski trails. Users will be able to efficiently plan out all kinds of routes, as well as available facilities to make their hiking activities feel as safe as possible, in addition to writing reviews about their own trail experiences. The goal of the application is to encourage users to visit, review, and share their experiences of trails all across British Columbia.

3. ER Diagram Changes



Our Milestone 1 feedback indicated that a weak entity can only have one parent. We have changed TrailReview only to be a weak entity of Trail, but we left User as a one-to-many relationship to TrailReview. We did not enforce full participation in the case of a User deleting their account. We have also added/removed/moved some attributes to better suit SQL types

and create more FDs that do not include primary keys. Attributes that were added include Sport, ExploreDate, OpenTime, CloseTime, etc.

4. ER Diagram to Relational Model

Underlined = Primary Key
Bold = Foreign Key

Region(RegionID: INTEGER, TerrainType: VARCHAR(50), City: VARCHAR(50), EmergPhone: VARCHAR(13))

- Candidate Key: RegionID

PartOf_Route(RouteID: INTEGER, **RegionID**: INTEGER, Difficulty: VARCHAR(10), EstTime: INTEGER, Length_km: DECIMAL(6, 3))

- Candidate Key: RouteID
- RegionID has the constraint NOT NULL

LocatedIn_Trail(TrailID: INTEGER, **RegionID**: INTEGER, TrailName: VARCHAR(30), Length_km: DECIMAL(6, 3), Sport: VARCHAR(20), TerrainType: VARCHAR(20), StartLoc: VARCHAR(29), EndLoc: VARCHAR(29))

- Candidate Key: TrailID
- RegionID has the constraint NOT NULL

Includes(**RouteID**: INTEGER, **TrailID**: INTEGER)

- Candidate Key: (RouteID, TrailID)
- Cannot enforce full participation constraint on Route without assertions

User(Email: VARCHAR(254), Username: VARCHAR(25), PhoneNum: VARCHAR(13), Hometown: VARCHAR(30))

- Candidate Key: Email, Username, or PhoneNum
- Username has the constraint NOT NULL and UNIQUE
- PhoneNum has the constraint NOT NULL and UNIQUE

Explores(**TrailID**: INTEGER, **Email**: VARCHAR(254), ExploreDate: DATE)

- Candidate Key: (TrailID, Email)

Writes_TrailReview_On(ReviewID: INTEGER, **TrailID**: INTEGER, Rating: INTEGER, ReviewDate: DATE, Comments: VARCHAR(300), **Email**: VARCHAR(254))

- Candidate Key: (ReviewID, TrailID)

Owns_Equipment(EquipmentID: INTEGER, Availability: INTEGER, Type: VARCHAR(30) Condition: VARCHAR(20), Price: REAL, **FacilityID**: INTEGER)

- Candidate Key: EquipmentID
- FacilityID has the constraint NOT NULL
- Cannot enforce full participation constraint on RentalShop without assertions

PartOf(**Email**: VARCHAR(50), **GroupID**: INTEGER)
- Candidate Key: Email, GroupID
- Cannot enforce full participation constraint on Group without assertions

Leads_Group(GroupID: INTEGER, **Email**: VARCHAR(254), Size: INTEGER, Experience: VARCHAR(20))
- Email has the constraint NOT NULL
- Candidate Key: GroupID

Has_Parking(FacilityID: INTEGER, **RegionID**: INTEGER, Location: VARCHAR(29), OpenTime: TIME, CloseTime: TIME, Capacity: INTEGER, Fee: REAL)
- Candidate Key: FacilityID
- RegionID has the constraint NOT NULL

Has_Restroom(FacilityID: INTEGER, **RegionID**: INTEGER, Location: VARCHAR(29), OpenTime: TIME, CloseTime: TIME, Condition: VARCHAR(20))
- Candidate Key: FacilityID
- RegionID has the constraint NOT NULL

Has_RentalShop(FacilityID: INTEGER, **RegionID**: INTEGER, Location: VARCHAR(29), OpenTime: TIME, CloseTime: TIME)
- Candidate Key: FacilityID
- RegionID has the constraint NOT NULL

5. Functional Dependencies

Region:
- RegionID → TerrainType, City, EmergPhone
- City → EmergPhone

PartOf_Route:
- RouteID → RegionID**,** Difficulty, EstTime, Length_km
- Length_km, Difficulty → EstTime

LocatedIn_Trail:
- TrailID → TrailName, Length_km, TerrainType, StartLoc, EndLoc, Sport
- TerrainType → Sport

Includes:

- No non-trivial FD's

User:

- Email → Username, PhoneNum, Hometown
- Username → Email, PhoneNum, Hometown
- PhoneNum → Username, Email, Hometown

Explores:

- No non-trivial FD's

Write_TrailReview_On:

- ReviewID, TrailID → Rating, ReviewDate, Comments, Email

Owns_Equipment:

- EquipmentID → Availability, Type, Condition, Price, FacilityID
- Type, Condition → Cost

PartOf:

- No non-trivial FD's

Leads_Group:

- GroupID → Email, Size, Experience

Has_Parking:

- FacilityID → Location, OpenTime, CloseTime, Capacity, Fee

Has_Restroom:

- FacilityID → Location, OpenTime, CloseTime, Condition

Has_RentalShop:

- FacilityID → Location, OpenTime, CloseTime, Payment

6. Normalization

We will normalize our tables by applying the BCNF algorithm. This ensures that if we break a relation R and put it back together, we should get exactly R. In addition, BCNF will allow us to reduce redundancies in our tables. Since the only relations that violate BCNF are Region, PartOf_Route, LocatedIn_Trail, and Owns_Equipment, we will apply the BCNF algorithm to only those four.

We will start by decomposing Region(RegionID, TerrainType, City, EmergPhone). City → EmergPhone violates BCNF:

> Region_1(City, EmergPhone)
> Region_2(RegionID, TerrainType, **City**)
>
> Region_1 does not violate BCNF because the only dependency is City → EmergPhone, and Region_2 does not violate BCNF because RegionID is a superkey. In Region_2, City becomes a foreign key referencing Region_1.

Next, we will decompose PartOf_Route(RouteID, **RegionID**, Difficulty, EstTime, Length_km). Length_km, Difficulty → EstTime violates BCNF:

> PartOf_Route_1(Length_km, Difficulty, EstTime)
> PartOf_Route_2(RouteID, **RegionID**, **Length_km**, **Difficulty**)
>
> PartOf_Route_1 does not violate BCNF because the only dependency is Length_km, Difficulty → EstTime, and PartOf_Route_2 does not violate BCNF because RouteID is a superkey. In PartOf_Route_2, Length_km and Difficulty become foreign keys referencing PartOf_Route_1.

Next, we will decompose LocatedIn_Trail(TrailID: INTEGER, **RegionID**, TrailName, Length_km, Sport, TerrainType, StartLoc, EndLoc). Terrain → Sport violates BCNF:

> LocatedIn_Trail_1(TerrainType, Sport)
> LocatedIn_Trail_2(TrailID, **RegionID**, TrailName, Length_km, **TerrainType**, StartLoc, EndLoc)
>
> LocatedIn_Trail_1 does not violate BCNF because the only dependency is TerrainType → Sport, and LocatedIn_Trail_2 does not violate BCNF because TrailID is a superkey. In Located_In_Trail_2, TerrainType becomes a foreign key referencing LocatedIn_Trail_1.

The last table to decompose is Owns_Equipment(EquipmentID, Availability, Type, Condition, Price, **FacilityID**). Type, Condition → Price violates BCNF:

> Owns_Equipment_1(Type, Condition, Price)
> Owns_Equipment_2(EquipmentID, Availability, **Type**, **Condition**, **FacilityID**)
>
> Owns_Equipment_1 does not violate BCNF because the only dependency is Type, Condition → Price, and Owns_Equipment_2 does not violate BCNF because EquipmentID is a superkey. In Owns_Equipment_2, Type and Condition become foreign keys referencing Owns_Equipment_1.

Our final relational schema are:

- Region_1(City, EmergPhone)
- Region_2(RegionID, TerrainType, **City**)
- PartOf_Route_1(Length_km, Difficulty, EstTime)
- PartOf_Route_2(RouteID, **RegionID**, **Length_km**, **Difficulty**)
- LocatedIn_Trail_1(TerrainType, Sport)
- LocatedIn_Trail_2(TrailID, **RegionID**, TrailName, Length_km, **TerrainType**, StartLoc, EndLoc)
- Includes(**RouteID**, **TrailID**)
- User(Email, Username, PhoneNum, Hometown)
- Explores(**TrailID**, **Email**, ExploreDate)
- Writes_TrailReview_On(ReviewID, **TrailID**, Rating, ReviewDate, Comments, **Email**)
- Owns_Equipment_1(Type, Condition, Price)
- Owns_Equipment_2(EquipmentID, Availability, **Type**, **Condition**, **FacilityID**)
- PartOf(**Email**, **GroupID**)
- Leads_Group(GroupID, **Email**, Size, Experience)
- Has_Parking(FacilityID, **RegionID**, Location, OpenTime, CloseTime, Capacity, Fee)
- Has_Restroom(FacilityID, **RegionID**, Location, OpenTime, CloseTime, Condition)
- Has_RentalShop(FacilityID, **RegionID**, Location, OpenTime, CloseTime)

7. SQL DDL Statements

The full .sql file can be found in our assignment submission. We are aware that Oracle does not support ON UPDATE CASCADE, and it should be ON UPDATE CASCADE, but to accommodate Oracle in the case where ON UPDATE CASCADE is most suitable, we will restrict any updates to maintain constraints by omitting an ON UPDATE statement (i.e., the default action is to restrict updates unless specified). ON UPDATE and ON DELETE explanations for all foreign keys are provided below:

- Region_2(City)
    - No ON DELETE statement because we do not want to be able to delete a City with a Region in it.
    - ON UPDATE CASCADE because if the City is changed, we want to propagate this information to the Region.
- PartOf_Route_2(RegionID)
    - No ON DELETE statement because we do not want to be able to delete a Region with a Route in it.
    - ON UPDATE CASCADE because if the Region is change,d we want to propagate this information to the Route.
- PartOf_Route_2(Length_km, Difficulty)
    - No ON DELETE statement because we do not want to be able to delete a tuple describing EstTime for a Route.
    - ON UPDATE CASCADE because then we can update our EstTime if better information comes in, and it will be propagated.

- LocatedIn_Trail_2(RegionID)
    - No ON DELETE statement because we do not want to be able to delete a Region with a Trail in it.
    - ON UPDATE CASCADE because if the Region is changed, we want to propagate this information to the Trail.
- LocatedIn_Trail_2(TerrainType)
    - No ON DELETE statement because we do not want to be able to delete a tuple describing the relationship between TerrainType and Sport.
    - No ON UPDATE statement because a TerrainType and associated Sport should not need to change, so restrict this.
- Includes(RouteID)
    - ON DELETE CASCADE because if we delete a Route, we need to delete any relationships with that Route.
    - ON UPDATE CASCADE because if we update a Route, we need to update any relationships with that Route.
- Includes(TrailID)
    - ON DELETE CASCADE because if we delete a Trail, we need to delete any relationships with that Trail.
    - ON UPDATE CASCADE because if we update a Trail, we need to update any relationships with that Trail.
- Explores(TrailID)
    - ON DELETE CASCADE because if we delete a Trail, we need to delete any relationships with that Trail.
    - ON UPDATE CASCADE because if we update a Trail, we need to update any relationships with that Trail.
- Explores(Email)
    - ON DELETE CASCADE because if we delete a User, we need to delete any relationships with that User.
    - ON UPDATE CASCADE because if we update a User, we need to update any relationships with that User.
- Writes_TrailReview_On(TrailID)
    - ON DELETE CASCADE because if we delete a Trail, we need to delete any TrailReview of that Trail.
    - ON UPDATE CASCADE because if we update a Trail, we need to update any TrailReview of that Trail.
- Writes_TrailReview_On(Email)
    - ON DELETE SET NULL because if a User is deleted, we need to keep the review but remove the association with any TrailReview left by the User.
    - ON UPDATE CASCADE because if we update a User, we need to update any TrailReview made by that User.
- Owns_Equipment_2(Type, Condition)
    - ON DELETE CASCADE because if we want to delete a Type, we need to delete any Equipment of that Type.

    - ON UPDATE CASCADE because we want to propagate any change in a Price descriptor to the Equipment.
- Owns_Equipment_2(FacilityID)
    - ON DELETE CASCADE because if we delete a RentalShop, we need to delete any Equipment from that RentalShop.
    - ON UPDATE CASCADE because if we update a RentalShop, we need to update any Equipment from that RentalShop.
- PartOf(Email)
    - ON DELETE CASCADE because if we delete a User, we need to delete any relationships with that User.
    - ON UPDATE CASCADE because if we update a User, we need to update any relationships with that User.
- PartOf(Group)
    - ON DELETE CASCADE because if we delete a Group, we need to delete any relationships with that Group.
    - ON UPDATE CASCADE because if we update a Group, we need to update any relationships with that Group.
- Leads_Group(Email)
    - No ON DELETE because we do not want to be able to delete a User that is the leader of a Group (they must transfer leadership first).
    - ON UPDATE CASCADE because if we update a User, the changes should propagate to the Group.
- Has_Parking(RegionID)
    - ON DELETE CASCADE because if we delete a Region, we need to delete any Parking in that Region.
    - ON UPDATE CASCADE because if we update a Region, we need to update any Parking in that Region.
- Has_Restroom(RegionID)
    - ON DELETE CASCADE because if we delete a Region, we need to delete any Restroom in that Region.
    - ON UPDATE CASCADE because if we update a Region, we need to update any Restroom in that Region.
- Has_RentalShop(RegionID)
    - ON DELETE CASCADE because if we delete a Region, we need to delete any RentalShop, in that Region.
    - ON UPDATE CASCADE because if we update a Region, we need to update any RentalShop, in that Region.

8. INSERT Statements

All INSERT statements can be found in our .sql file in our assignment submission.

9. AI Statement

We did not use AI for any part of this assignment.