University Of Tennessee
Knoxville

Art Of Hiking Application

Description of Tables

Authors:
Gabriel Hanas
Chris Tester
Robert Moncrief
Trevor Jones
Anthony Stewart

Customer:
Bradley Vander Zanden

March 11, 2016

Contents

1 Table: Region

1.1 Attributes

regionId:

Can it be Null: No
Type: Int
Description: An integer that will distinguish the region. Auto incremented.

regionName:

Can it be Null: No
Type: Varchar (255)
Description: The name of the region.

1.2 Keys

Primary Key - `regionId`
The region table will use a standard incrementing regionID to list all of the warnings input by the users.

1.3 Constraints:

Region names must be unique.

2 Table: Trail

2.1 Attributes

traiId:

Can it be Null: No
Type: Int
Description: A unique integer that identifies a trail. Auto incremented.

trailName:

Can it be Null: No
Type: Varchar (255)
Description: A character string that corresponds to the trail name.

trailLength:

Can it be Null: No
Type: Double (3,1)
Description: A doubleing point number representing the length of a trail to one decimal point.

trailDifficulty:

Can it be Null: Yes
Type: Double (3,2)
Description: A doubleing point number that represents the standard difficulty of a trail.

elevationGain:

Can it be Null: No
Type: Double (6,2)
Description: An integer that represents the overall elevation gain of a trail.

elevationGainMap:

Can it be Null: Yes
Type: Varchar (255)
Description: Unknown

region:

Can it be Null: No
Type: Int
Description: An integer that defines which region a trail lies in.

horseAccessible:

Can it be Null: No
Type: Boolean
Description: A true or false statement to indicate if horses are allowed on a trail.

trailDescription:

Can it be Null: Yes
Type: Text
Description: A character string describing the trail and its highlights.

isLoop:

Can it be Null: No
Type: Boolean
Description: Tells if a trail is point to point or a loop.

2.2 Keys

Primary Key - `trailId`
Foreign Key - `region` (Links to Region table)
The `trailId` is the primary key because more than one trail can have the same name. The artificial key, `trailId` uniquely defines a trail.

2.3 Constraints

When inserting, the region must already exist. No two trails can have the same name in the same region.

3 Table: Intersection

3.1 Attributes

intersectionId:

Can it be Null: No
Type: Int
Description: A unique integer that identifies when two or more trails intersect. Auto increment.

longitudePt:

Can it be Null: No
Type: Double
Description: A doubleing point number representing the longitudinal position of the intersection.

latitudePt:

Can it be Null: No
Type: Double
Description: A doubleing point number representing the latitudinal position of the intersection.

3.2 Keys

Primary Key - `intersectionId`
The `intersectionId` is the primary key since it will uniquely define the intersection point. The other fields will give partial information to the location of the intersection but do not specify which intersection it is.

3.3 Constraints

No special constraints.

4 Table: InterestPointType

4.1 Attributes

typeId:

Can it be Null: No
Type: Int
Description: An integer that uniquely identifies a point of interest. Auto increment.

interestPointDescription:

Can it be Null: Yes
Type: Varchar
Description: A short description of the interest point type.

4.2 Keys

Primary Key - `typeId`
The `typeId` is the primary key, because if a point of interest exists, it will always have an id number. There is no guarantee there will be a description available for that interest point.

4.3 Constraints

As long as the entry exists there are not any constraints on the data since it is only a description.

5 Table: InterestPoint

5.1 Attributes

interestPointId:

Can it be Null: No
Type: Int
Description: A unique integer that identifies an interest point. Auto increment.

interestPointName:

Can it be Null: Yes
Type: Varchar
Description: The name of an interest point, if applicable.

typeId:

Can it be Null: No
Type: Int
Description: An integer that defines what type of interest point the data is.

latitudePoint:

Can it be Null: No
Type: Double
Description: The latitudinal location of a trail point.

longitudePoint:

Can it be Null: No
Type: Double
Description: The longitudinal location of a trail point.

5.2 Keys

Primary Key - `interestPointId`
Foreign Key - `typeId` (Links to Interest Point)
The `interestPointId` is the primary key since it is the only field that uniquely identifies the point of interest. For `interestPointName`, as the application expands there is a possibility of two points of interest with the same name to exist.

5.3 Constraints

When an interest point is created, it must have a type.

6 Table: ParkingLot

6.1 Attributes

parkingLotId:

Can it be Null: No
Type: Int
Description: An integer specific to the parking lot. Auto increment.

maxLotCapacity:

Can it be Null: Yes
Type: Int
Description: An integer number that represents the maximum amount of parking spots the given lot has.

latitudePoint:

Can it be Null: No
Type: Double
Description: The latitudinal location of the parking lot.

longitudePoint:

Can it be Null: No
Type: Double
Description: The longitudinal location of the parking lot.

6.2 Keys

Primary Key - `parkingLotId`
The `parkingLotId` is the primary key because no other feilds can describe it.

6.3 Constraints

No special constraints.

7 Table: TrailPoint

7.1 Attributes

trailPointId:

Can it be Null: No
Type: Int
Description: The Id for a point along a specific trail. Auto increment.

latitudePoint:

Can it be Null: No
Type: Double
Description: The latitudinal location of a trail point.

longitudePoint:

Can it be Null: No
Type: Double
Description: The longitudinal location of a trail point.

elevationPoint:

Can in be Null: No
Type: Double
Description: The elevation location of a trail point.

trailId:

Can it be Null: No
Type: Int
Description: An integer that corresponds to a trail. This is okay because we will duplicate trailPoints that lie on multipe trails.

previousPoint:

Can it be Null: Yes
Type: Int
Description: An integer that corresponds to the next trail point. If null, it is the 'common' start of a trail.

7.2 Keys

Primary Key - `trailPointId`
Foreign Key - `trailId` (Links to Trail)
Foreign Key - `previousPoint` (Links to TrailPoint)
The trail point table will hold a list of points that make up a given trail. They will be marked by longitudinal and latitudinal points from the map. Each point will be given a unique id. If trails share a point, the point will be duplicated in these cases.

7.3 Constraints:

No unique constraints.

8 Table: TrailHead

8.1 Attributes

trailHeadId:

Can it be Null: No
Type: Int
Description: An integer that will distinguish all the beginnings of trails. Auto increment.

parkingLotId:

Can it be Null: Yes
Type: Int
Description: A unique integer that will match the starting point to a parking lot if available.

trailPoint:

Can it be Null: No
Type: Int
Description: The id for a starting trail point.

8.2 Keys

Primary Key - `trailHeadId`
Foreign Key - `parkingLotId` (Links to ParkingLot)
Foreign Key - `trailPt` (Links to Trail)
The trail head table will contain the starting points of trails and any parking lot that is connected to them. The `trailHeadId` will be unique to allow the user to select either end if the trail is has more than on starting position.

8.3 Constraints:

The only constraint is the head point must be connected to a trail that already exists in the database.

9 Table: Warnings

9.1 Attributes

warningId:

Can it be Null: No
Type: Int
Description: An integer that will distinguish the Warning.

trailId:

Can it be Null: Yes
Type: Int
Description: A unique integer that will match the warning to a specific trail.

regionId:

Can it be Null: Yes
Type: Int
Description: A unique integer that will match the warning to a specific region.

dateEntered:

Can it be Null: No
Type: DateTime
Description: The date that the warning was entered.

warningText:

Can it be Null: Yes
Type: Text
Description: The text of the warning.

9.2 Keys

Primary Key - `warningId`
Foreign Key - `trailId`
Foreign Key - `regionId`

The warning table will use a standard incrementing warningID to list all of the warnings input by the users.

9.3 Constraints:

Either the regionId or the trailId must be valid.

10 Table: User

10.1 Attributes

userId:

Can it be Null: No
Type: Int
Description: A unique integer that identifies a user to the database. Auto increment.

username:

Can it be Null: No
Type: Varchar
Description: A picked username that will be used to log in for writing reviews.

password:

Can it be Null: No
Type: Varchar
Description: A user designated password to sign into their account.

email:

Can it be Null: No
Type: Varchar
Description: A email to allow the user account to be validated.

10.2 Keys

Primary Key - `userId`
The `userId` is the primary key to retain all artificial keys as primary keys throughout the database for consistency. `username` is a candidate key since it is unique.

10.3 Constraints

Once the user is created, there will be constraints on the `password` and `username`. The `password` will have to be at least eight characters and contain letters and numbers. The `password` will also be salted and hashed for security. The `username` has to be unique and be between 6 and 20 characters.

11 Table: Review

11.1 Attributes

reviewId:

Can it be Null: No
Type: Int
Description: An integer that uniquely identifies the review. Auto increment.

userId:

Can it be Null: No
Type: Int
Description: An integer that links a review to its author.

trailId:

Can it be Null: No
Type: Int
Description: An integer that links a review to a specific trail.

userRating:

Can it be Null: Yes
Type: Int
Description: An integer, one through five, that relates to how much a user liked a trail.

userReview:

Can it be Null: Yes
Type: Text
Description: A written review that describes what a user liked or disliked about a trail.

11.2 Keys

Primary Key - `reviewId`
Foreign Key - `userId` (Links to User table)
Foreign Key - `trailId` (Links to Trail table)
The `reviewId` is the primary key since the `userRating` and `userReview` are not unique fields.

11.3 Constraints

A user must make a rating and/or a review. Once a review or rating is created it must be linked to a user and trail.

12 Table: TrailIntersectionMapping

12.1 Attributes

traillId:

Can it be Null: No
Type: Int
Description: A unique integer that corresponds with a trail.

intersectionId:

Can it be Null: No
Type: Int
Description: A unique integer that corresponds with an intersection.

12.2 Keys

Foreign Key - `trailId` (Links to Trail table)
Foreign Key - `intersectionId` (Links to Intersection table)
The desired intersection is found by using the `trailId` and the `intersectionId` to create a `trailIntersectionId`.

12.3 Constraints

When the table is entered there must be a `trailId` and an `intersectionId` in order to insert.

13 Table: InterestPointTrailMapping

13.1 Attributes

trailId:

Can it be Null: No
Type: Int
Description: An integer that represents a trail.

interestPointId:

Can it be Null: No
Type: Int
Description: An integer representing an interest point.

13.2 Keys

Foreign Key - `trailId` (Links to Trail table)
Foreign Key - `interestPointId` (Links to InterestPoint table)
The junction table will use the two foreign keys together to find the specific trail that a specific point of interest lies on, this junction table is needed when one point of interest is on multiple trails.

13.3 Constraints

When the table is entered there must be a `trailId` and an `interestPointId` in order to find the needed information.

14 Table: ParkingLotTrailMapping

14.1 Attributes

parkingLotId:

Can it be Null: No
Type: Int
Description: An integer number that represents a parking lot.

trailId:

Can it be Null: No
Type: Int
Description: An integer number that represents a trail.

14.2 Keys

Foreign Key - `parkingLotId` (Links to ParkingLot table)
Foreign Key - `trailId` (Links to Trail)
The junction table will use the two foreign keys together to find the specific parking lot that a specific trail is connected to. This junction table is useful when one parking lot connects to multiple trails or a trail has multiple parking lots linked to it.

14.3 Constraints:

When the table is entered there must be a `parkingLotId` and an `trailId` in order to find the needed information.

15 ER Diagram