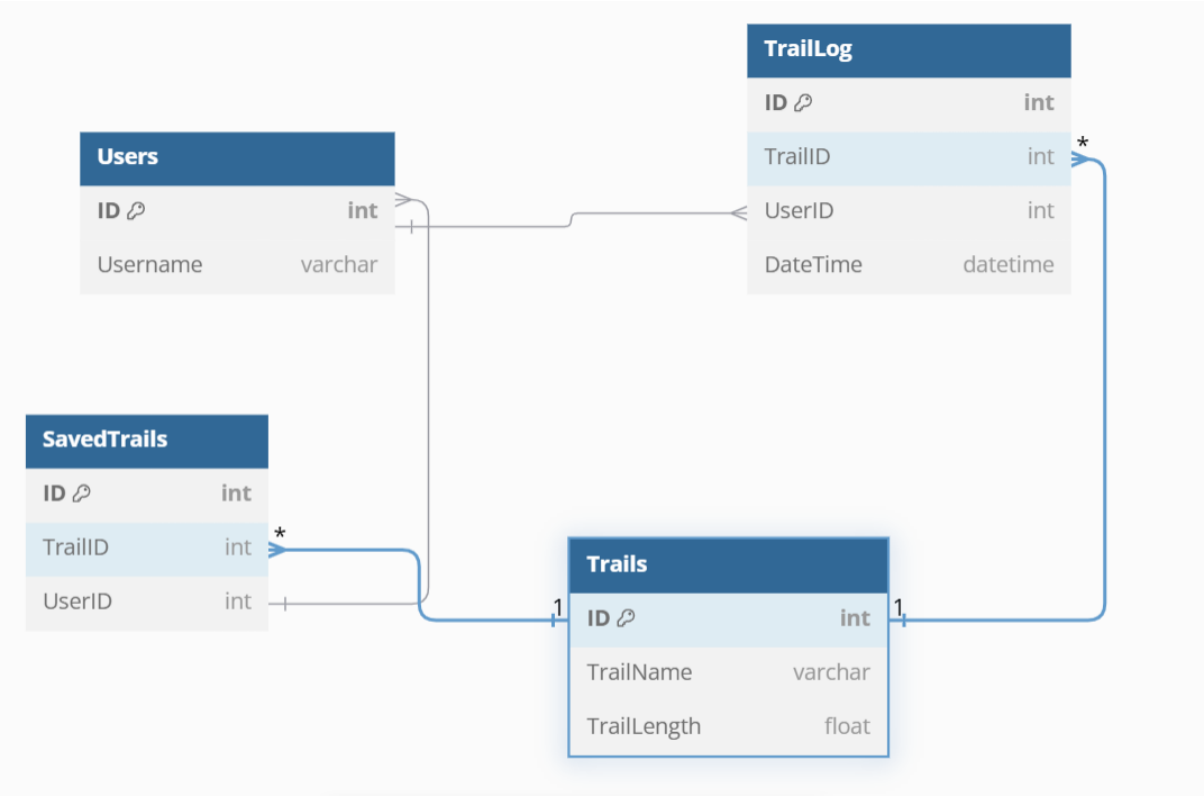
COMP2001 – Information Management and Retrieval: CW1

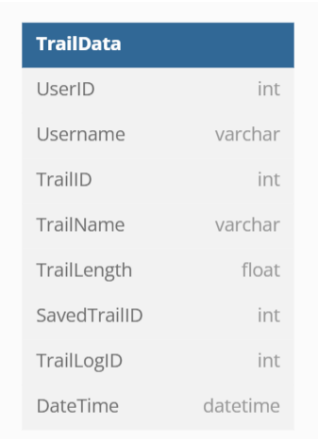
Exercise 1



The objective of 3NF is to eliminate redundancy further by ensuring that non-key attributes are only dependent on the primary key. This reduces the likelihood of update anomalies and improves the database's integrity and efficiency.

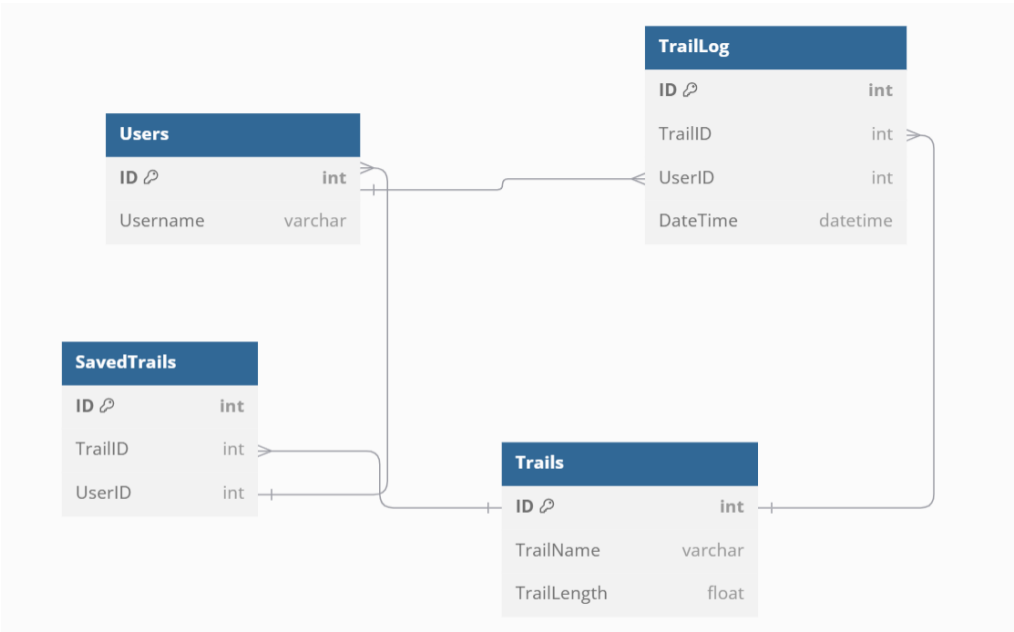
Exercise 2

UNF design:



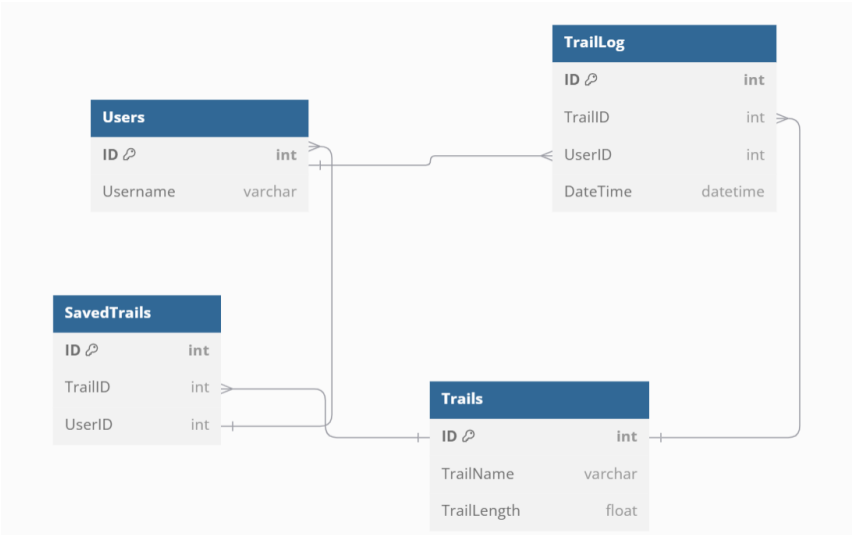
A single table containing all information without normalization constraints.

1NF Design:



1NF has no changes; atomic values are present with no repeating groups.

2NF Design:



2NF has no changes from 3NF; fully functional dependencies are maintained.

Exercise 3

Field Definition grid for Attributes of Entity 'Users'

Entity name: Description
Synonyms: Relationship Link Phrases

Attribute name	Description	Synonym(s)	Data type	Size (*=max)	Possible data values	Optional ?	Validation rules	Key?
ID	Unique identifier for each user	UserID	int	8 digits	Positive integers	No	Must be unique	Primary Key
UserName	Username of the user	UserHandle	varchar	255	Alphanumeric characters	No	Must not be empty	

Field Definition grid for Attributes of Entity 'Trails'

Entity name: Description
Synonyms: Relationship Link Phrases

Attribute name	Description	Synonym(s)	Data type	Size (*=max)	Possible data values	Optional ?	Validation rules	Key?
ID	Unique identifier for each trail	TrailID	Int	8 digits	Positive integers	No	Must be unique	Primary key
TrailName	Name of the trail	TrailTitle	Varchar	255	Text	No	Must not be empty	
TrailLength	Length of the trail (in miles or kilometers)	TrailDistance	float	3 decimal points	Positive floating numbers	Yes	Not Null	

Field Definition grid for Attributes of Entity 'Saved Trails'

Entity name: Description
Synonyms: Relationship Link Phrases

Attribute name	Description	Synonym(s)	Data type	Size (*=max)	Possible data values	Optional ?	Validation rules	Key?
ID	Unique identifier for saved trail records	SavedTrailID	Int	8 digits	Positive integers	No	None	
UserID	Identifier for the user who saved the trail	SavedBy	Int	8 digits	Must match Users.ID	No	Foreign Key to Users.ID	Foreign key
TrailID	Identifier for the saved trail	SavedTrail	int	8 digits	Must match Trails.ID	No	Foreign Key to Trails.ID	Foreign key

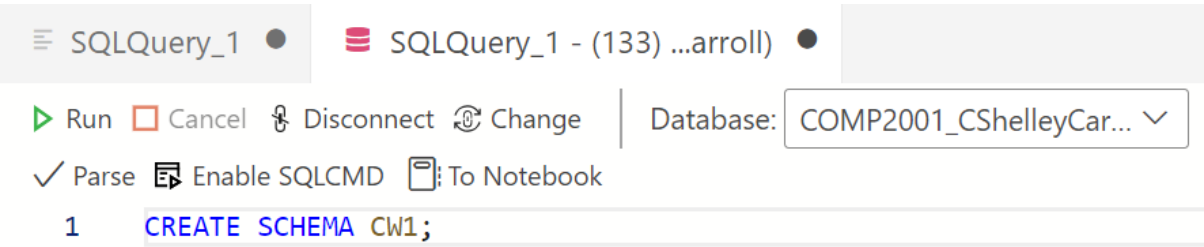
Field Definition grid for Attributes of Entity 'Trail Log'

Entity name: Description
Synonyms: Relationship Link Phrases

Attribute name	Description	Synonym(s)	Data type	Size (*=max)	Possible data values	Optional ?	Validation rules	Key?
ID	Unique identifier for each trail log entry	LogID	int	8 digits	Positive integers	No	Must be unique	Primary key
TrailID	Identifier of the trail being logged	LogTrail	Int	8 digits	Must match Trails.ID	No	Foreign Key to Trails.ID	Foreign key
UserID	Identifier of the user who logged the trail	LoggedBy	Int	8 digits	Must match Users.ID	No	Foreign Key to Users.ID	Foreign key
DateTime	Date and time of the trail log	LogDateTime	Datetime		Valid date-time format	No	Must be in standard datetime format	

Exercise 4

Here I began the creation of the database, firstly creating the schema:



SQLQuery_1 5 ● SQLQuery_1 - (133) ...arroll) 5 ●

Run Cancel Disconnect Change Database: COMP2001_CShelleyCa...

```sql
CREATE TABLE CW1.Users (
    ID int IDENTITY(1,1) primary key,
    UserName varchar(max) NOT NULL
);


CREATE TABLE CW1.Trails (
    ID int IDENTITY(1,1) primary key,
    TrailName varchar(max) not null,
    TrailLength float
);


CREATE TABLE CW1.SavedTrails (
    ID int IDENTITY(1,1) PRIMARY KEY,
    TrailID int not null,
    UserID int null,
    FOREIGN KEY (TrailID) REFERENCES CW1.Trails(ID)
    FOREIGN KEY (UserID) REFERENCES CW1.Trails(ID)
);
CREATE TABLE CW1.TrailLog (
    ID int IDENTITY(1,1) primary key,
    TrailID int not null,
    UserID int null,
    DateTime DATETIME DEFAULT GETDATE(),
    FOREIGN KEY (TrailID) REFERENCES CW1.Trails(ID)
);
```

Search

Search by name of type (t:, v:, f:, or sp:)

Name	Schema	Type
SavedTrails	CW1	Table
TrailLog	CW1	Table
Trails	CW1	Table
Users	CW1	Table

Inserting data using SELECT Statements:

SQLQuery_1 9+ ● | SQLQuery_1 - (133) ...arroll) 9+ ● | DIST-6-505.uopnet.plymouth.ac.uk:COMP20

Run | Cancel | Disconnect | Change | Database: COMP2001_CShelleyCa... | Estimated Plan | Enable Ac

```sql
INSERT INTO CW1.Users (UserName) VALUES ('Alex');
INSERT INTO CW1.Users (UserName) VALUES ('Billy');
INSERT INTO CW1.Users (UserName) VALUES ('Cody');
INSERT INTO CW1.Users (UserName) VALUES ('Daniel');

INSERT INTO CW1.Trails (TrailName, TrailLength) VALUES ('Riverbend Path', 5.2);
INSERT INTO CW1.Trails (TrailName, TrailLength) VALUES ('Maplewood Trail', 8.7);
INSERT INTO CW1.Trails (TrailName, TrailLength) VALUES ('Pine Hollow', 3.5);
INSERT INTO CW1.Trails (TrailName, TrailLength) VALUES ('Willow Springs', 6.1);
INSERT INTO CW1.Trails (TrailName, TrailLength) VALUES ('Aspen Ridge', 10.0);

INSERT INTO CW1.SavedTrails (TrailID, UserID) VALUES (1, 1);
INSERT INTO CW1.SavedTrails (TrailID, UserID) VALUES (2, 1);
INSERT INTO CW1.SavedTrails (TrailID, UserID) VALUES (3, 2);
INSERT INTO CW1.SavedTrails (TrailID, UserID) VALUES (4, 3);
INSERT INTO CW1.SavedTrails (TrailID, UserID) VALUES (5, 4);

INSERT INTO CW1.TrailLog (TrailID, UserID, DateTime) VALUES (1, 1, '2023-11-01 09:30:00');
INSERT INTO CW1.TrailLog (TrailID, UserID, DateTime) VALUES (2, 1, '2023-11-01 14:00:00');
INSERT INTO CW1.TrailLog (TrailID, UserID, DateTime) VALUES (3, 2, '2023-11-02 10:15:00');
INSERT INTO CW1.TrailLog (TrailID, UserID, DateTime) VALUES (4, 3, '2023-11-02 11:45:00');
INSERT INTO CW1.TrailLog (TrailID, UserID, DateTime) VALUES (5, 4, '2023-11-03 13:00:00');
```

SQLQuery_1 - (133) ...arroll) 7 ● | CW1.

Run | Cancel | Disconnect | Change | Data

```sql
SELECT * FROM CW1.Users;

SELECT * FROM CW1.Trails;

SELECT * FROM CW1.SavedTrails;

SELECT * FROM CW1.TrailLog;
```

	ID	UserName
1	1	Alex
2	2	Billy
3	3	Cody
4	4	Daniel

	ID	TrailName	TrailLength
1	1	Riverbend Path	5.2
2	2	Maplewood Trail	8.7
3	3	Pine Hollow	3.5
4	4	Willow Springs	6.1
5	5	Aspen Ridge	10

	ID	TrailID	UserID
1	1	1	1
2	2	2	1
3	3	3	2
4	4	4	3
5	5	5	4

	ID	TrailID	UserID	DateTime
1	1	1	1	2023-11-01 09:30:00.000
2	2	2	1	2023-11-01 14:00:00.000
3	3	3	2	2023-11-02 10:15:00.000
4	4	4	3	2023-11-02 11:45:00.000
5	5	5	4	2023-11-03 13:00:00.000

Exercise 5

Here I needed to put all the data in one table by creating a view statement.

Run Cancel Disconnect Change Database: COMP2001_CShelleyCa...

```sql
CREATE VIEW CW1.UserTrailSummary AS
SELECT
    U.ID AS UserID,
    U.UserName,
    T.ID AS TrailID,
    T.TrailName,
    T.TrailLength,
    ST.ID AS SavedTrailID,
    TL.ID AS TrailLogID,
    TL.DateTime AS TrailLogDateTime
FROM
    CW1.Users U
LEFT JOIN
    CW1.SavedTrails ST ON U.ID = ST.UserID
LEFT JOIN
    CW1.Trails T ON ST.TrailID = T.ID
LEFT JOIN
    CW1.TrailLog TL ON U.ID = TL.UserID AND T.ID = TL.TrailID;
```

```
SELECT * FROM CW1.UserTrailSummary;
```

	UserID	UserName	TrailID	TrailName	TrailLength	SavedTrailID	TrailLogID	TrailLogDateTime
1	1	Alex	1	Riverbend Path	5.2	1	1	2023-11-01 09:30:00.000
2	1	Alex	2	Maplewood Trail	8.7	2	2	2023-11-01 14:00:00.000
3	2	Billy	3	Pine Hollow	3.5	3	3	2023-11-02 10:15:00.000
4	3	Cody	4	Willow Springs	6.1	4	4	2023-11-02 11:45:00.000
5	4	Daniel	5	Aspen Ridge	10	5	5	2023-11-03 13:00:00.000

Exercise 6

Here I needed to implement CRUD

Create – Here I will be adding new data, in this case, a new user into the user table:

```
CREATE PROCEDURE CW1.InsertUser
    @UserName VARCHAR(MAX)
AS
BEGIN
    INSERT INTO CW1.Users (UserName)
    VALUES (@UserName);
END;
```

Now I run this test to add the new user:

```
EXEC CW1.InsertUser @UserName = 'Eddie';
```

Reading: Here I retrieve a user by their ID:

```sql
CREATE PROCEDURE CW1.GetUserByID
    @UserID int
AS
BEGIN
    SELECT * FROM CW1.Users
    WHERE ID = @UserID;
END;
```

When I run this test I retrieve the user with their ID number, and I get the user's details:

```sql
EXEC CW1.GetUserByID @UserID = 1;
```

	ID	UserName
1	1	Alex

Update: Here I am creating a stored procedure and updating the username column in the users table.

```sql
CREATE PROCEDURE CW1.UpdateUser
    @UserID INT,
    @NewUserName VARCHAR(MAX)
AS
BEGIN
    UPDATE CW1.Users
    SET UserName = @NewUserName
    WHERE ID = @UserID;
END;
```

After checking the existing name of the user, I run the command below so the username gets updated.

```sql
EXEC CW1.UpdateUser @UserID = 1, @NewUserName = 'Abel';
```

Delete: Here I am deleting a specific saved trail record from a user.

```
CREATE PROCEDURE CW1.DeleteUser
    @SavedTrailID int
AS
BEGIN
    DELETE FROM CW1.SavedTrails
    WHERE ID = @SavedTrailID;
END;
```

Here I run this command so the Saved Trail record for that user gets deleted.

```
EXEC CW1.DeleteUser @SavedTrailID = 4;
```

Exercise 7

Triggers automatically log a new trail when it is added to the CW1.Trails table. The log table will store the details, such as who created the trail and the timestamp were added by. The log table was created previously so next is to create the trigger:

```
CREATE TRIGGER CW1.InsertTrigger
ON CW1.Trails
AFTER INSERT
AS
BEGIN
    INSERT INTO CW1.TrailLog (TrailID, DateTime)
    SELECT
        ID,
        GETDATE()
    FROM inserted;
END;
```

Testing the trigger by running an insert query to affect the trails table

Run Cancel Disconnect Change Database: COMP2001_

```
INSERT INTO CW1.Trails (TrailName, TrailLength)
VALUES ('Crystal Lake Trail', 7.5);
```

Here is the old data:

	ID	TrailName	TrailLength
1	1	Riverbend Path	5.2
2	2	Maplewood Trail	8.7
3	3	Pine Hollow	3.5
4	4	Willow Springs	6.1
5	5	Aspen Ridge	10

Here are the changes to the Trails Table:

	ID	TrailName	TrailLength
1	1	Riverbend Path	5.2
2	2	Maplewood Trail	8.7
3	3	Pine Hollow	3.5
4	4	Willow Springs	6.1
5	5	Aspen Ridge	10
6	7	Crystal Lake T…	7.5

Here is some old/new data in the Trail Log table which might have been wrong, I have refreshed the table, see bellow:

	ID	TrailID	UserID	DateTime
1	1	1	1	2023-11-01 09:30:00.000
2	2	2	1	2023-11-01 14:00:00.000
3	3	3	2	2023-11-02 10:15:00.000
4	4	4	3	2023-11-02 11:45:00.000
5	5	5	4	2023-11-03 13:00:00.000
6	6	7	1	2024-11-04 04:08:19.443

Fresh trail log:

Run Cancel Disconnect Change Database: COMP2001_CShelleyCa...

```
SELECT * FROM CW1.TrailLog
```

Results Messages

	ID	TrailID	UserID	DateTime
1	1	9	NULL	2024-11-04 14:02:52.177

As you can see, it misses a UserID due to our Complications and limitation with the Trigger, we will need to implement an extra step which will cause 1 data field being populated manually which is not ideal but best solution for now.

```
Run  Cancel  Disconnect  Change   Database: COMP2001_CShelleyCa...
INSERT INTO CW1.TrailLog (UserID)
VALUES (3)
```

Manually inserted user id data at a slightly later time

```
Run  Cancel  Disconnect  Change   Database: COMP2001_CShelleyCa...
CREATE PROCEDURE [CW1].[UpdateTrailLogUser]
    @TrailID INT,
    @NewUserID INT
AS
BEGIN
    UPDATE CW1.TrailLog
    SET UserID = @NewUserID
    WHERE TrailID = @TrailID;
END;
```

From earlier learnt skills, I have adapted a procedure that will look for the latest record in the table TrailLog and update the user with what we want.

```
Run  Cancel  Disconnect  Change   Database: COMP2001_CShelleyCa...
EXEC CW1.UpdateTrailLogUser @TrailID = 9, @NewUserID = 2;
```

Here, I have taken reference of the trail ID that hasn't had a user, and Supplied the datafield with updated data for UserID.

```
Run  Cancel  Disconnect  Change   Database: COMP2001_CShelleyCa...
SELECT * FROM CW1.TrailLog
```

Results Messages

	ID	TrailID	UserID	DateTime
1	1	9	2	2024-11-04 14:02:52.177