Design Document

Systems Design and Databases

TRA Race Entry System

Contents

[2 INTRODUCTION 1](#_Toc497402457)

[3 DATA MODEL 1](#_Toc497402458)

[3.1 UML Class Diagram – Conceptual Model 1](#_Toc497402459)

[3.2 Data Dictionary 2](#_Toc497402460)

[3.3 UML Class Diagram – Implementation Model 3](#_Toc497402461)

[4 Sequence Diagram for Race Registration 4](#_Toc497402462)

[5 Data Definition Language 4](#_Toc497402463)

[5.1 Schema 4](#_Toc497402464)

[5.2 Tables 4](#_Toc497402465)

[6 SQL Queries from the code harness 5](#_Toc497402466)

[6.1 Query 1 5](#_Toc497402467)

[6.2 Query 2 5](#_Toc497402468)

[6.3 Query 3 5](#_Toc497402469)

[6.4 Query 4 5](#_Toc497402470)

# INTRODUCTION

This report consist of :

1. **Database modeling** of the race registration system , which comprise of UML Class diagram for both Conceptual and implemented model together with algorithm for calculating final points for racers.
2. **Sequential diagram** for race registration which covered with diagram and explanation.
3. **Microsoft SQL database** data definition language for building database tables and data manipulation language for insert and select values for the created tables.
4. **Four SQL Queries** upon the specified questions.

# **DATA MODEL**

## UML Class Diagram – Conceptual Model

**Report :**

The following model shows the conceptual model of the registration system , base on the ideal interview there are 5 classes to consider at the current stage :

1. Racer Entry
2. Independent Racer
3. Member
4. Race Marshal
5. Club

So the following diagram indicates the stereotype , attribute , method and unresolved many to many relationship of these 5 classes.

CONCEPTUAL CLASS DIAGRAM HERE – CLASSES, ATTRIBUTES, OPERATIONS, ASSOCIATIONS AND MULTIPLICITY

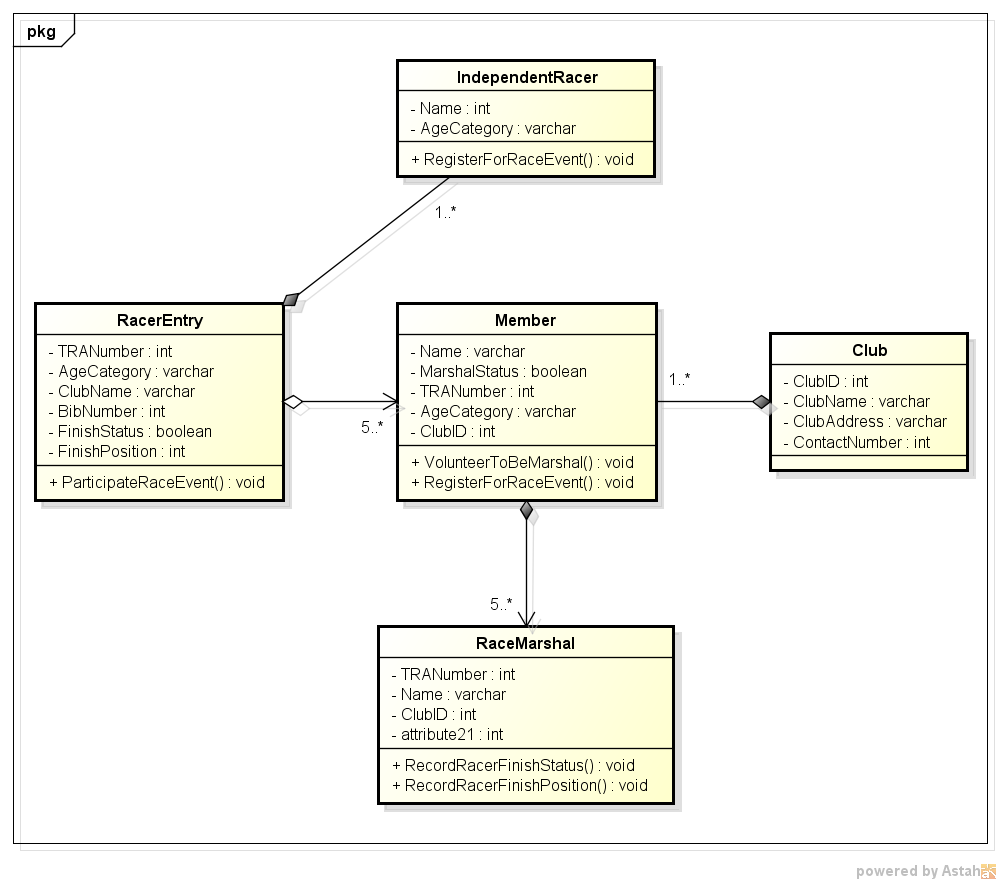


Figure 1: UML Conceptual Class Diagram

## Data Dictionary

The following data dictionary definition is for the operation which calculates the race entrant’s points.

**Report :**

**Data Dictionary**

|  |  |  |
| --- | --- | --- |
| **Name** | **Data Type** | **Description** |
| RaceEventID | Integer | The serial number for each race event |
| BibNumber | Integer | The serial number each racer receive on every race event |
| FinishStatus | Boolean | Weather the racer had finished the match |
| FinishPosition | Integer | The ranking result for each racer |
| Points | Integer | The points coverted from racer's finish position |

For example : On the race event number 2(**RaceEventID**=2) , racer number 4(**BibNumber**=4) finished the race(**FinishStatus**=True) and he ranked as number 4 when he finish the race(**FinishPosition**=4) then the points he will be receive is 94(**Points**=94) base on the conversion method below stats.

**Points Conversion Pseudo Code**

ConvertRankingIntoPoints()

Input RaceEventID

Input BibNumber

Input FinishStatus

Input FinishPosition

IF FinishStatus == true && 1<= FinishPosition <=50 THEN

Points = 100-2\*(FinishPosition-1)

ELSE IF FinishStatus == true && FinishPosition >50 THEN

Points = 1

ELSE

Points = 0

Display “Racer number ” , BibNumber , “ final score is ” , Points , “ on the race event number ” , RaceEventID

**The Pesudo code Explanation**

When call up function ConvertRankingIntoPoints ,

User should input RaceEventID , BibNumber , FinishStatus , FinishPosition as parameter for the program to do calculation.

If the racer had finished the race and ranked among number 1 - 50 , then the program will calculate his points based on formula : Points = 100-2\*(FinishPosition-1).

If the racer had finished the race but ranked lower than 50 , then the program will calculate his points as : Points = 1.

If the racer didn’t finished the race , then the program will calculate his points as : Points = 0.

Finally after calculation the program will display a message to indicate the racer’s score on that race event , for example : “Racer number 4 final score is 94 on the race event number 2”

## UML Class Diagram – Implementation Model

**Report :**

The following diagram shows the 6 database tables which should be implement for the actual registration system :

1. RacerEntry table to record race participants details when they sign up for a race event ( I didn’t create an Age Category table because i set AgeCategory column in this table as varchar data type so user can just key in their corresponding age category for example : ‘Teenager’ or ‘Adult’ or ‘Elder’ when they sign up for the race event ).
2. RaceEvent table to record the history of each race event had been hold.
3. Venue table to record the location where a race event can be hold.
4. Club table to record the club each race participant belongs to ( if Club ID = IND means it’s an independent racer who don’t belongs to any specific club ).
5. Member table to record the TRA member details ( if TRA number = 0000 means it’s not a member ).
6. RaceMarshal table to record the history of which member volunteered to be race marshal in which race event.

This 6 tables implemented with stereotype , attributes , methods , primary key - foreign key link and resolved many to many relationships.

IMPLEMENTATION MODEL WITH PRIMARY KEYS, FOREIGN KEYS, RESOLVED MANY-TO-MANYS , STEREOTYPES AND DIRECTED ASSOCIATIONS

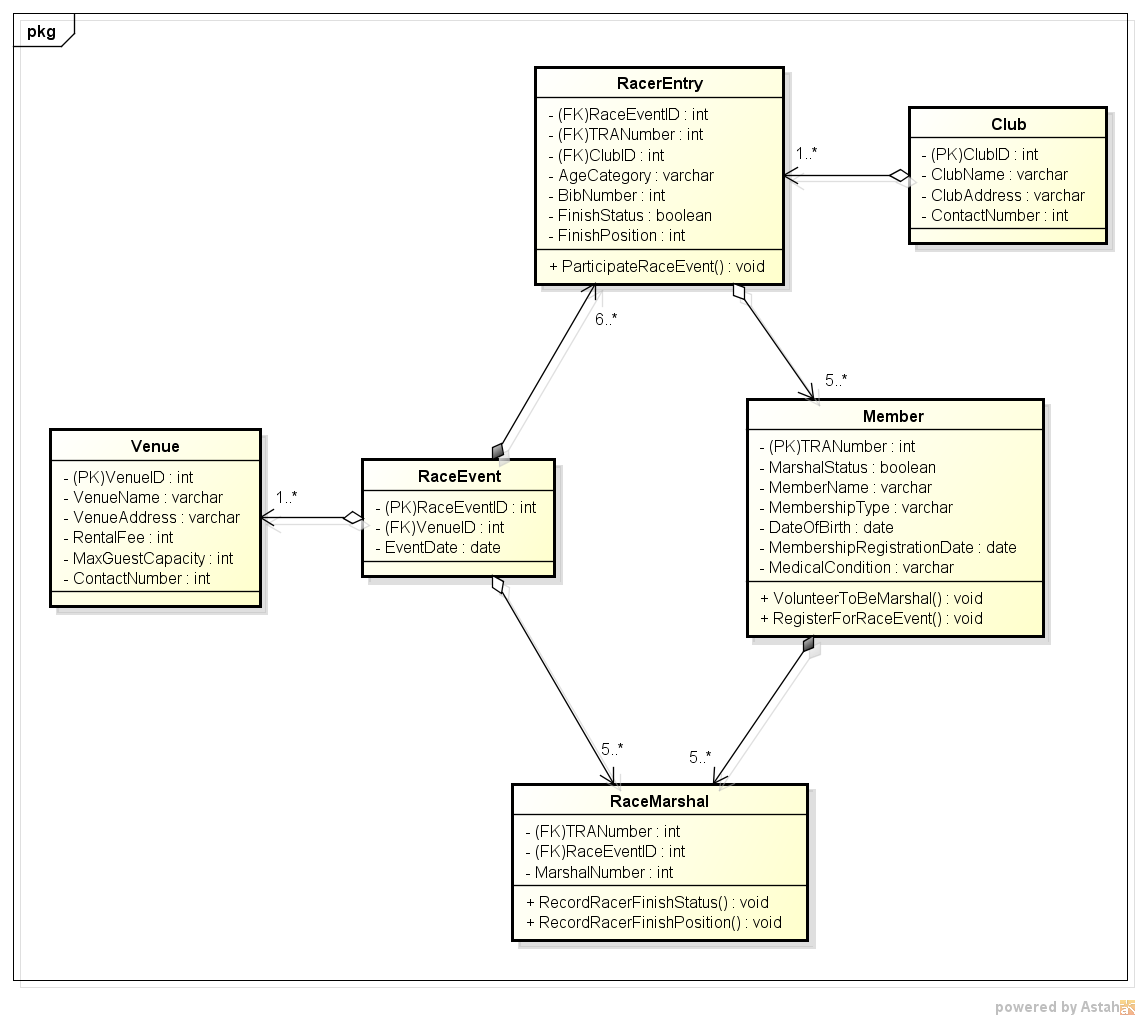


Figure 2: UML Implementation model

# Sequence Diagram for Race Registration

SEQUENCE DIAGRAM FOR THE ACTIVITY OF REGISTERING FOR A RACE.

**Report :**

The following sequence diagram shows the interaction of a new racer using the registration system to sign up for a race event in each steps.

There are 5 entities in this diagram :

1. New racer
2. Login window
3. Main window
4. Registration window
5. Database

(The diagram is in next page)

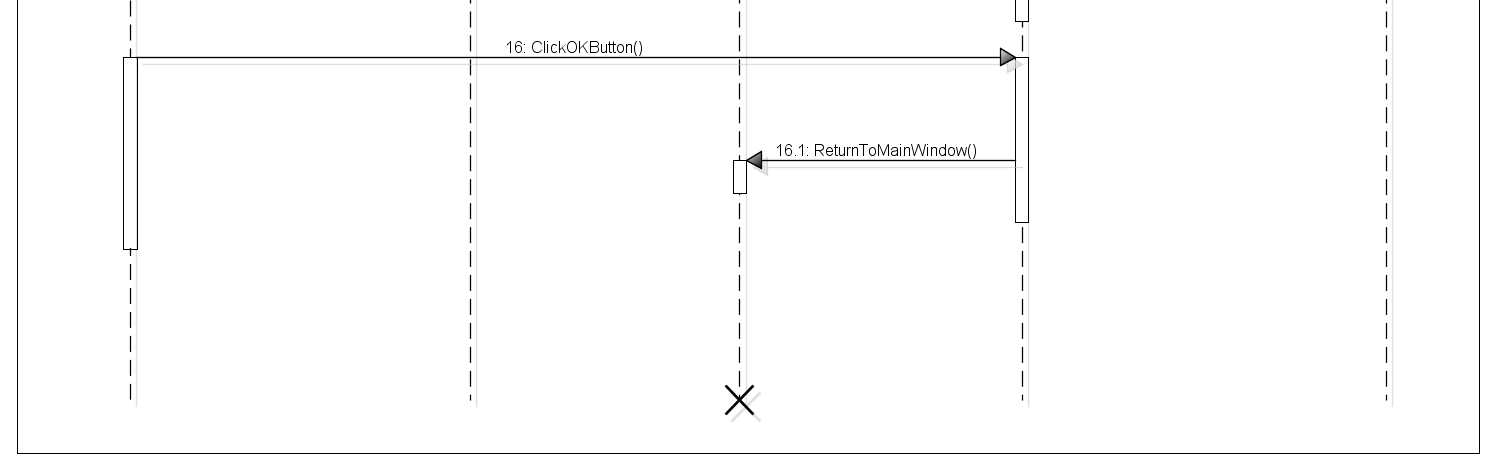
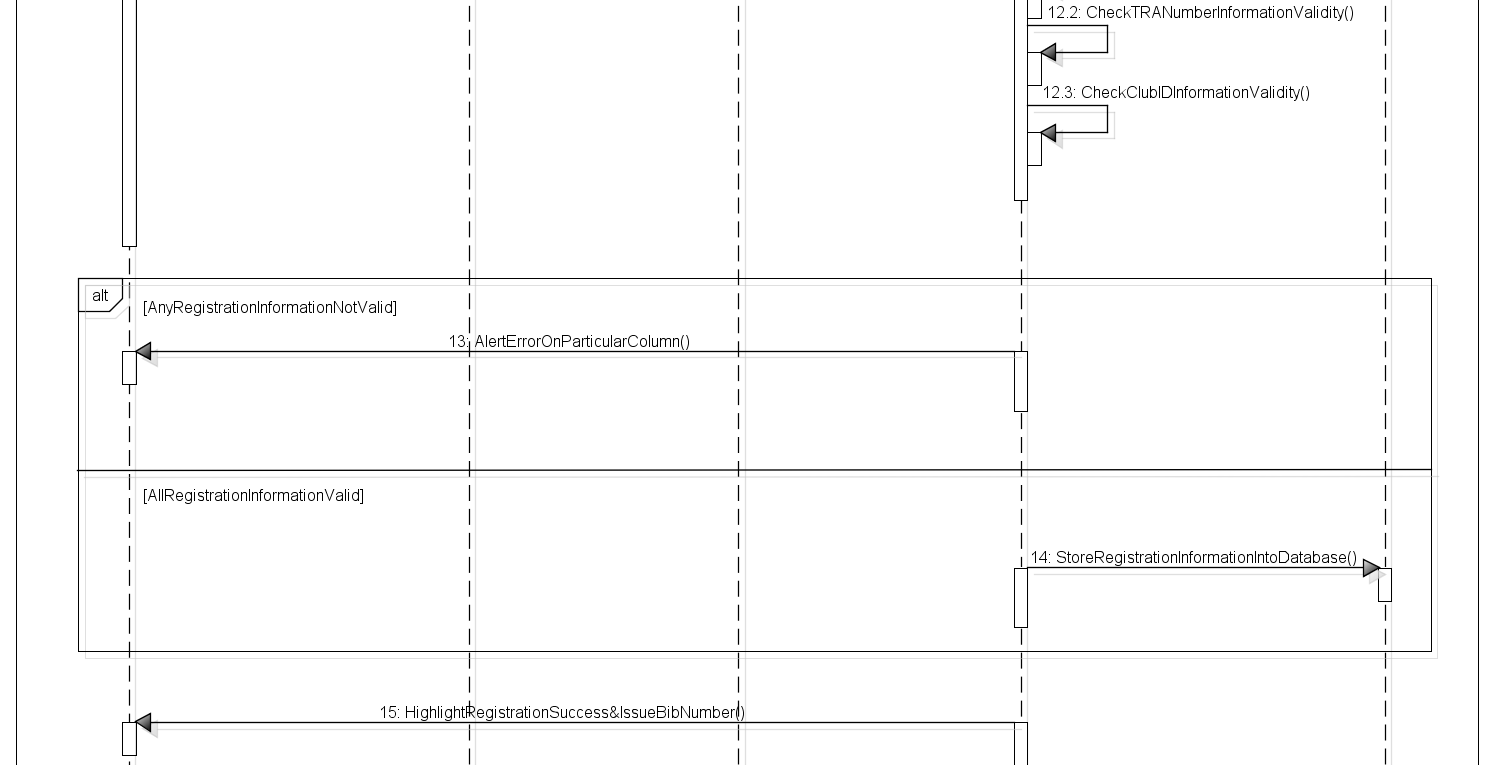
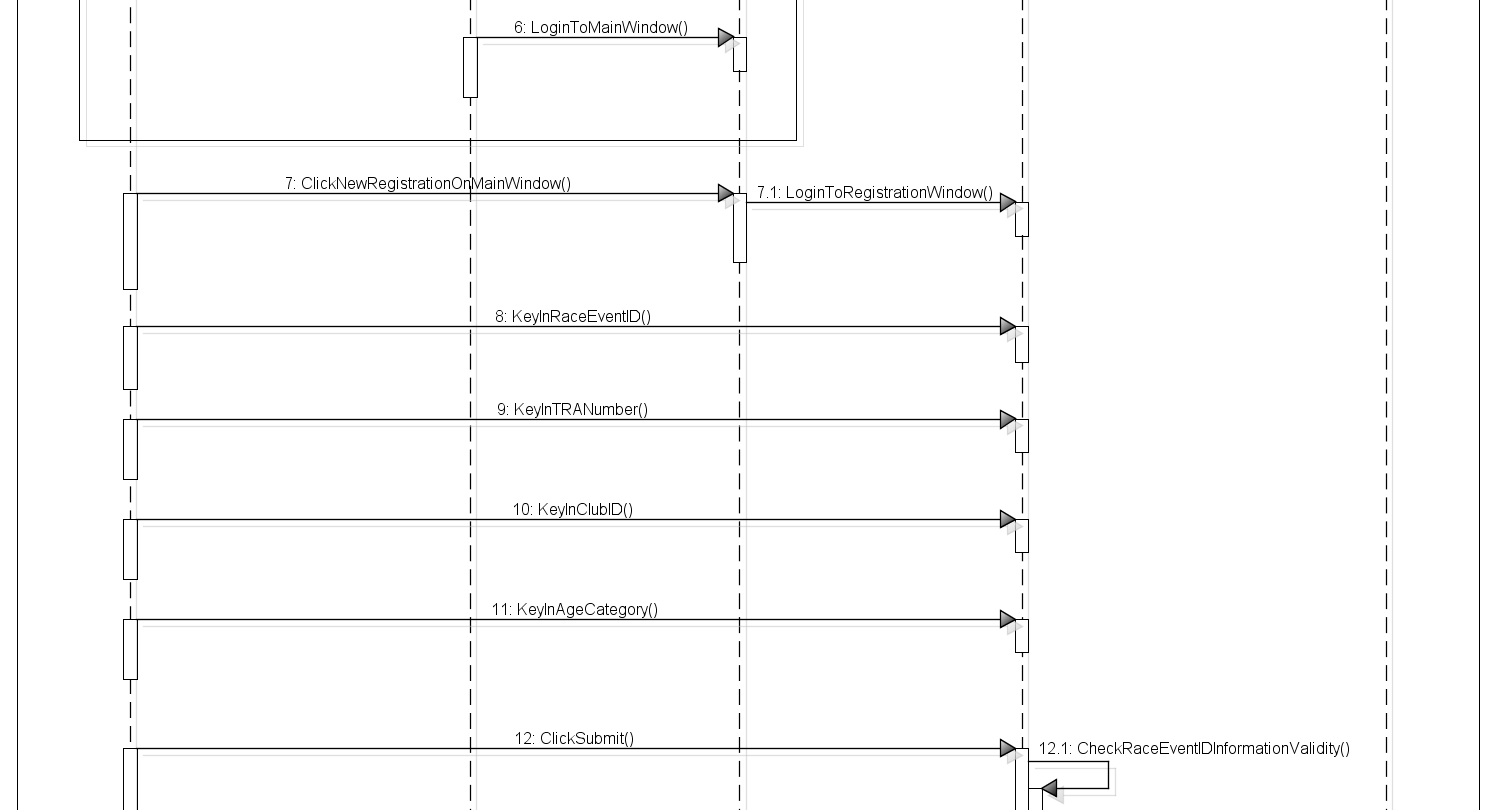
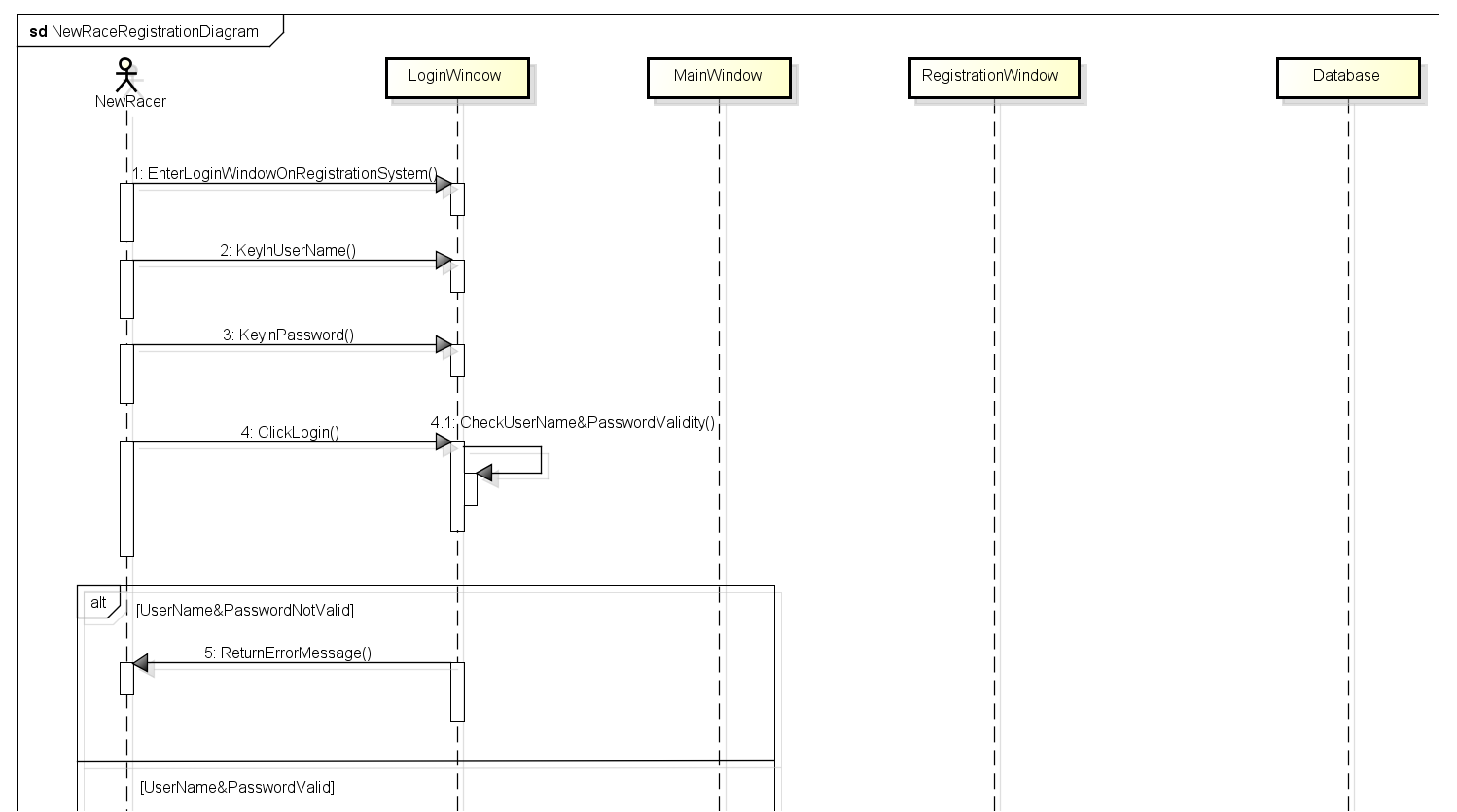


Figure 3: Sequence Diagram for Race Entry

Explain the sequential steps for each entities on the diagram above

**New Racer :**

1.enter the login window -> 2.key in user name -> 3.key in password -> 4.click login -> (pass to Login Window)

7.click new registration button on main window -> (pass to Main Window)

8.key in race event ID -> 9.key in TRA number -> 10.key in club ID -> 11.key in age category -> 12.click submit button -> (pass to Registration Window)

16.click OK button on main window -> (pass to Registration Window)

**Login Window :** 4.1 check user name and password validity

-> 5.alt [user name and password not valid] return error message

-> 6.alt [user name and password valid] login to main window -> (pass to New Racer)

**Main Window :**

7.1 login to registration window -> (pass to New Racer)

**Registration Window :**

12.1 check entered race event ID information validity -> 12.2 check entered TRA number information validity

-> 12.3 check entered club ID information validity

-> 13.alt [any registration information not valid] alert error on particular column

-> 14.alt [all registration information not valid] store registration information into database -> 15.highlight registration success and issue bib number -> (pass to New Racer)

16.1 return to main window -> END

# Data Definition Language

## Schema

CREATE SCHEMA RaceSchema

go

## Tables

QUERIES FOR CREATING THE RACE ENTRY DATABASE :

**Report :**

Data Definition Language For Create Tables

Create database dbRaceRegistrationSystem

go

Use dbRaceRegistrationSystem

Create table Club (

[ClubID] varchar(10) primary key not null ,

[ClubName] varchar(20) not null ,

[ClubAddress] varchar(50) ,

[ContactNumber] int ,

);

Select \* from Club

Use dbRaceRegistrationSystem

Create table Member (

[TRANumber] int primary key not null identity(1,1) ,

[MarshalStatus] varchar(5) not null ,

[MemberName] varchar(20) not null ,

[MembershipType] varchar(10) ,

[DateOfBirth] date ,

[MembershipRegistrationDate] date ,

[MedicalCondition] varchar(50) ,

);

Select \* from Member

Use dbRaceRegistrationSystem

Create table Venue (

[VenueID] int primary key not null identity(1,1) ,

[VenueName] varchar(20) not null ,

[VenueAddress] varchar(50) not null ,

[RentalFee] int ,

[MaxGuestCapacity] int ,

[ContactNumber] int ,

);

Select \* from Venue

Use dbRaceRegistrationSystem

Create table RaceEvent (

[RaceEventID] int primary key not null identity(1,1) ,

[VenueID] int foreign key references Venue(VenueID) not null ,

[EventDate] date not null ,

);

Select \* from RaceEvent

Use dbRaceRegistrationSystem

Create table RacerEntry (

[RaceEventID] int foreign key references RaceEvent(RaceEventID) not null ,

[TRANumber] int foreign key references Member(TRANumber) ,

[ClubID] varchar(10) foreign key references Club(ClubID) not null ,

[AgeCategory] varchar(10) not null ,

[BibNumber] int not null ,

[FinishStatus] varchar(5) not null ,

[FinishPosition] int ,

);

Select \* from RacerEntry

Use dbRaceRegistrationSystem

Create table RaceMarshal (

[TRANumber] int foreign key references Member(TRANumber) not null ,

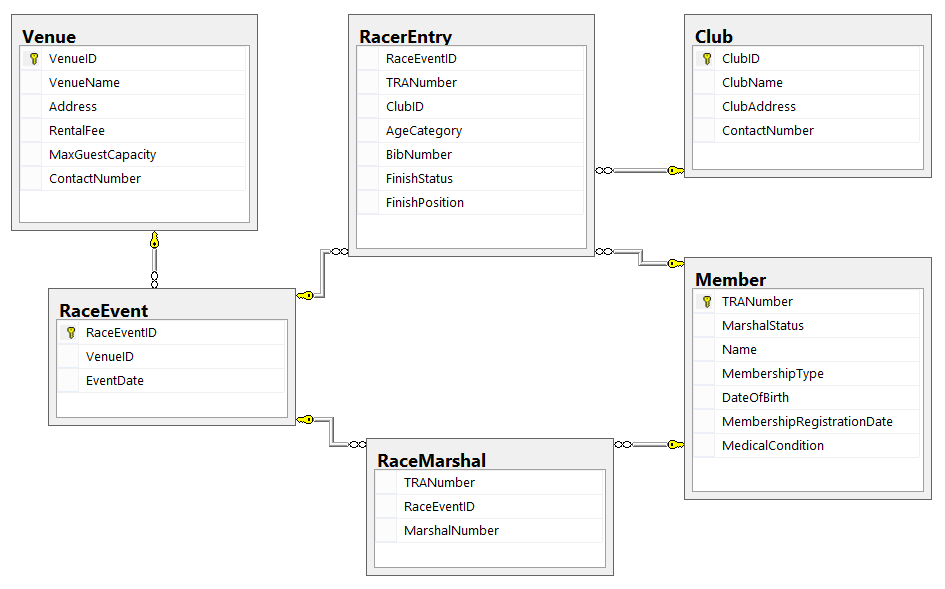
[RaceEventID] int foreign key references RaceEvent(RaceEventID) not null ,

[MarshalNumber] int not null ,

);

Select \* from RaceMarshal

The dbRaceRegistrationSystem Database Diagram



Data Manipulation Language For Insert Values Into Each Table Created

INSERT INTO Club (ClubID, ClubName, ClubAddress, ContactNumber)

VALUES

('1', 'AClub', 'ARoad', '99999999') ,

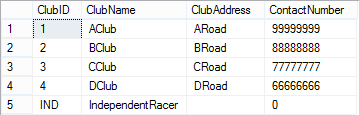
('2', 'BClub', 'BRoad', '88888888') ,

('3', 'CClub', 'CRoad', '77777777') ,

('4', 'DClub', 'DRoad', '66666666') ,

('IND', 'IndependentRacer', '', '')

Select \* from Club



SET IDENTITY\_INSERT Member ON

INSERT INTO Member (TRANumber, MarshalStatus, MemberName, MembershipType, DateOfBirth, MembershipRegistrationDate ,MedicalCondition)

VALUES

('1111', 'yes', 'AName', '1stClass' ,'1990-1-1', '2017-1-1', 'Asthma') ,

('2222', 'yes', 'BName', '2ndClass' ,'1990-2-1', '2017-2-1', 'Asthma') ,

('3333', 'yes', 'CName', '3rdClass' ,'1990-3-1', '2017-3-1', '') ,

('4444', 'yes', 'DName', 'standard' ,'1990-4-1', '2017-4-1', '') ,

('5555', 'yes', 'EName', 'standard' ,'1990-5-1', '2017-5-1', '') ,

('6666', 'no', 'FName', 'standard' ,'1990-6-1', '2017-6-1', '') ,

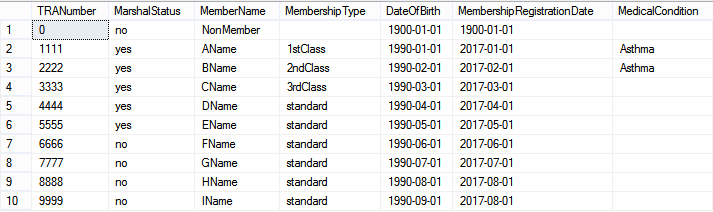
('7777', 'no', 'GName', 'standard' ,'1990-7-1', '2017-7-1', '') ,

('8888', 'no', 'HName', 'standard' ,'1990-8-1', '2017-8-1', '') ,

('9999', 'no', 'IName', 'standard' ,'1990-9-1', '2017-8-1', '') ,

('0000', 'no', 'NonMember', '' ,'', '', '')

Select \* from Member



SET IDENTITY\_INSERT Member OFF

SET IDENTITY\_INSERT Venue ON

INSERT INTO Venue (VenueID, VenueName, VenueAddress, RentalFee, MaxGuestCapacity, ContactNumber)

VALUES

('1', 'AVenue', 'Address1', '1000','300', '99999999') ,

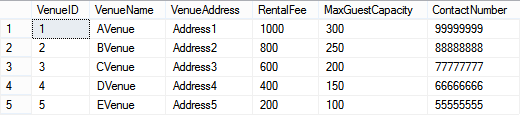
('2', 'BVenue', 'Address2', '800' ,'250', '88888888') ,

('3', 'CVenue', 'Address3', '600' ,'200', '77777777') ,

('4', 'DVenue', 'Address4', '400' ,'150', '66666666') ,

('5', 'EVenue', 'Address5', '200' ,'100', '55555555')

Select \* from Venue



SET IDENTITY\_INSERT Venue OFF

SET IDENTITY\_INSERT RaceEvent ON

INSERT INTO RaceEvent (RaceEventID, VenueID, EventDate)

VALUES

('1', '5', '2017-12-1'),

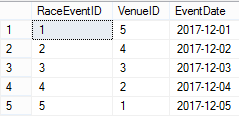
('2', '4', '2017-12-2'),

('3', '3', '2017-12-3'),

('4', '2', '2017-12-4'),

('5', '1', '2017-12-5')

Select \* from RaceEvent



INSERT INTO RacerEntry (RaceEventID, TRANumber, ClubID, AgeCategory, BibNumber, FinishStatus, FinishPosition)

VALUES

('1', '6666', '1', 'Adult', '1', 'yes', '1'),

('1', '7777', '2', 'Adult', '2', 'no', ''),

('1', '8888', '3', 'Adult', '3', 'yes', '2'),

('1', '9999', '4', 'Teenager', '4', 'no', ''),

('1', '0000', 'IND', 'Adult', '5', 'yes', '3'),

('1', '0000', 'IND', 'Teenager', '6', 'no', ''),

('1', '0000', 'IND', 'Adult', '7', 'yes', '4') ,

('1', '1111', '1', 'Adult', '', '', ''),

('1', '2222', '2', 'Adult', '', '', ''),

('1', '3333', '3', 'Adult', '', '', ''),

('1', '4444', '4', 'Adult', '', '', ''),

('1', '5555', '4', 'Adult', '', '', ''),

('2', '6666', '1', 'Adult', '1', 'no', ''),

('2', '7777', '2', 'Adult', '2', 'yes', '1'),

('2', '8888', '3', 'Adult', '3', 'no', ''),

('2', '9999', '4', 'Teenager', '4', 'yes', '2'),

('2', '0000', 'IND', 'Adult', '5', 'no', ''),

('2', '0000', 'IND', 'Teenager', '6', 'yes', '3'),

('2', '0000', 'IND', 'Adult', '7', 'no', ''),

('2', '1111', '1', 'Adult', '', '', ''),

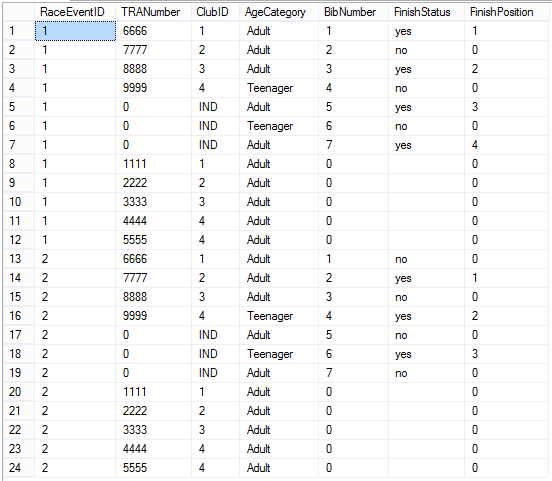
('2', '2222', '2', 'Adult', '', '', ''),

('2', '3333', '3', 'Adult', '', '', ''),

('2', '4444', '4', 'Adult', '', '', ''),

('2', '5555', '4', 'Adult', '', '', '')

Select \* from RacerEntry



INSERT INTO RaceMarshal (TRANumber, RaceEventID, MarshalNumber)

VALUES

('1111', '1', '1') ,

('2222', '1', '2') ,

('3333', '1', '3') ,

('4444', '1', '4') ,

('5555', '1', '5') ,

('1111', '2', '1') ,

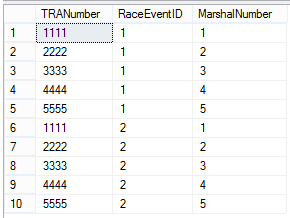
('2222', '2', '2') ,

('3333', '2', '3') ,

('4444', '2', '4') ,

('5555', '2', '5')

Select \* from RaceMarshal



## Query 1

Build a list of races that the specified member has not already entered.

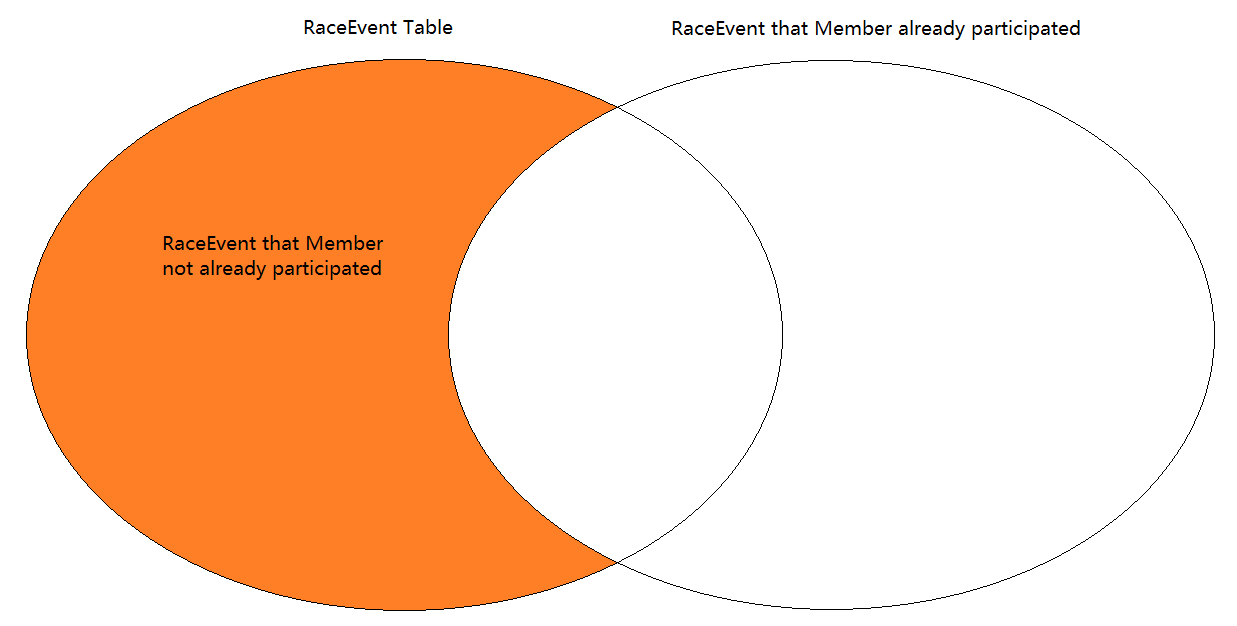
**Diagram :**

Build a list of races that the specified member has not already entered Means

The full list of RaceEvent that Exclude those Member had participated , so pseudo code

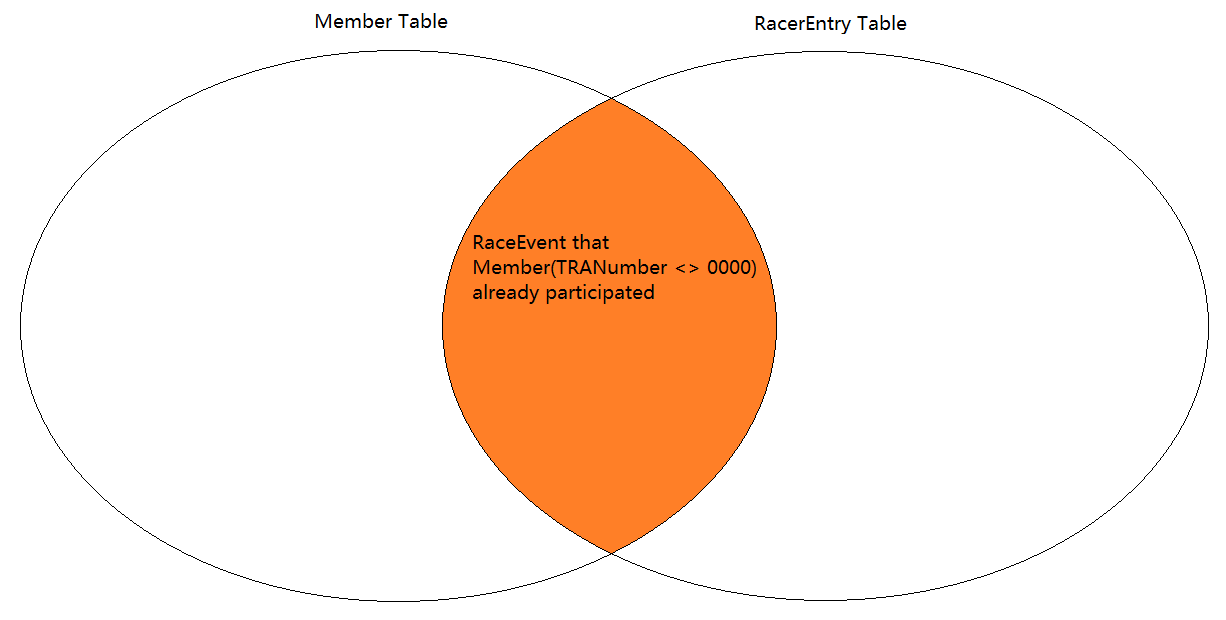
->

RaceEvent Left Outer Join Events which Member already participated



->

RaceEvent Left Outer Join (Member Inner Join RacerEntry)



->

**Report :**

Select Distinct RaceEvent.RaceEventID ,VenueID,EventDate From RaceEvent

Left Join

(RacerEntry Inner Join Member

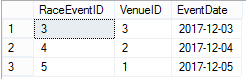
On RacerEntry.TRANumber <> 0000

)

On RaceEvent.RaceEventID=RacerEntry.RaceEventID

Where RacerEntry.RaceEventID IS NULL

Order By RaceEventID, VenueID ,EventDate Asc



## Query 2

Insert a new race entry into the database.

**Report :**

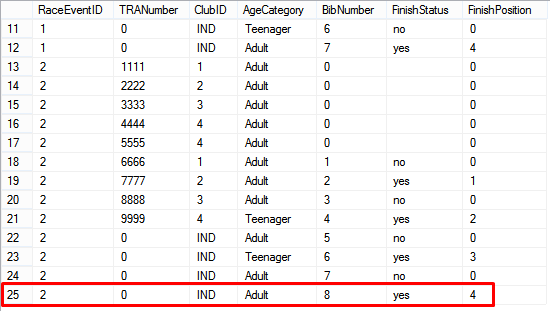
INSERT INTO RacerEntry (RaceEventID, TRANumber, ClubID, AgeCategory, BibNumber, FinishStatus, FinishPosition)

VALUES

('2', '0000', 'IND', 'Adult', '8', 'yes', '4')

Select \* From RacerEntry

Order By RaceEventID, BibNumber Asc



## Query 3

The purpose of this query is to count the total number of points for every member

**Report :**

SELECT TRANumber ,

COUNT(

CASE

WHEN FinishStatus='yes' THEN 1

ELSE null

END

) AS CompleteRaceNumber,

SUM (

CASE

WHEN FinishStatus='yes' and FinishPosition >= 1 and FinishPosition <=50 THEN 100-2\*(FinishPosition-1)

WHEN FinishStatus='yes' and FinishPosition > 50 THEN '1'

ELSE '0'

END

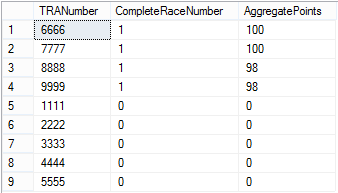
) AS AggregatePoints

FROM RacerEntry

Where TRANumber <> 0000

GROUP BY TRANumber

ORDER BY TRANumber Asc



## Query 4

The purpose of this query generate a list of entries for all races (raceid, venue, date, bib num, member name, age cat at time of entry, club at time of entry, position, points)

**Report :**

Select RacerEntry.RaceEventID, VenueName, EventDate, VenueAddress, BibNumber,

RacerEntry.ClubID, MemberName, AgeCategory, ClubName, RacerEntry.TRANumber,

FinishStatus, FinishPosition,

SUM (

CASE

WHEN FinishStatus='yes' and FinishPosition >= 1 and FinishPosition <=50 THEN 100-2\*(FinishPosition-1)

WHEN FinishStatus='yes' and FinishPosition > 50 THEN '1'

ELSE '0'

END

) AS Points

From RacerEntry, Member, Club, Venue, RaceEvent

Where RacerEntry.TRANumber=Member.TRANumber AND

RacerEntry.ClubID=Club.ClubID AND

RaceEvent.VenueID=Venue.VenueID AND

RaceEvent.RaceEventID=RacerEntry.RaceEventID

Group by RacerEntry.RaceEventID, VenueName, EventDate, VenueAddress, BibNumber,

RacerEntry.ClubID, MemberName, AgeCategory, ClubName, RacerEntry.TRANumber,

FinishStatus, FinishPosition

Order by RaceEventID, EventDate, BibNumber, ClubID, MemberName, TRANumber, FinishPosition Asc

