**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ АСТРАХАНСКОЙ ОБЛАСТИ**

**Государственное бюджетное профессиональное образовательное учреждение  
Астраханской области «Астраханский колледж вычислительной техники»**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **К защите допустить:** | | | | | | | |
| **Старший методист УМС** | | | | | | | |
|  |  | | | **Л.О.Земцова** | | | |
| **«** |  | **»** |  | | **20** |  | **г.** |

**курсовая работа**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| по МДК | 04.01 | Технология разработки и защиты баз данных | | | | | | |
|  | | | | | | | | |
|  | | | 6 семестр | | | | | |
| Тема курсовой работы | | | Разработка базы данных | | | | | |
|  | | | ***«***Комиссионный автосалон» | | | | | |
|  | | |  | | | | | |
| АКВТ.09.02.07.КР00.00ПЗ | | | | | | | | |
| Листов: | | | | 41 |  | | | |
|  | |  | | | | | | |
|  | |  | | | | | | |
| Выполнил | |  | | | | Студент гр. | | ПБ – 31 |
|  | |  | | | |  | | |
|  | |  | | | |  |  | |
| Проверил | |  | | | | Преподаватель | | |
|  | |  | | | | Мулеев Р.Р. | | |

**2022**

**Государственное бюджетное профессиональное образовательное учреждение**

**Астраханской области «Астраханский колледж вычислительной техники»**

|  |
| --- |
| УТВЕРЖДЕНО |
| Цикловой комиссией |
| специальности 09.02.07 |
| ПЦК \_\_\_\_\_\_\_\_\_ Ю.С. Андрианова |
| «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. |
| Специальность | *09.02.07 Информационные системы и программирование* | | |
| МДК 04.01 | *Технология разработки и защиты баз данных* | | |
| Группа | *ПБ– 31* | | |
| **ЗАДАНИЕ** | | | |
| **на курсовое проектирование** | | | |
| ***[ДАННЫЕ УДАЛЕНЫ]*** | | | |
| (фамилия, имя, отчество студента) | | | |
| Тема курсового проекта | | ***Разработка БД*** | |
| ***«Спортивный клуб»*** | | | |

Содержание пояснительной записки курсового проекта

Содержание

Введение

1. Проектирование базы данных
2. Анализ и описание предметной области информационной системы
3. Проектирование концептуальной модели
4. Построение логической модели, схемы базы данных
5. Выбор СУБД
6. Реализация базы данных
7. Физическое проектирование
8. Написание исходного кода БД
9. Написание запросов, функций и хранимых процедур

Заключение

Список использованных источников

Приложения

|  |  |  |  |
| --- | --- | --- | --- |
| Руководитель |  |  | Р.Р. Мулеев |
| Студент |  |  |  |

**Аннотация**

Данный документ на разработку базы данных имеет 41 страницы, 4 раздела, 3 рисунка, 3 источника использованной литературы. Таблицы и приложения отсутствуют.

**Содержание**

[Введение 5](#_Toc105528421)

[1. Проектирование БД 6](#_Toc105528422)

[1.1 Анализ и описание предметной области информационной системы 6](#_Toc105528423)

[1.2 Проектирование концептуальной модели 7](#_Toc105528424)

[1.3 Построение логической модели, схемы базы данных 8](#_Toc105528425)

[1.4 Выбор СУБД 15](#_Toc105528426)

[2 Реализация базы данных 16](#_Toc105528427)

[2.1 Физическое проектирование БД 16](#_Toc105528428)

[2.2 Написание исходного кода БД 16](#_Toc105528429)

[2.3 Написание запросов, функций и хранимых процедур 22](#_Toc105528430)

[Заключение 40](#_Toc105528431)

[Список используемой литературы 41](#_Toc105528432)

# Введение

Современные условия для работы по обслуживанию достигло того уровня, когда весь документооборот осуществляется на компьютерах, и вся информация храниться в специальных базах, потому растёт значимость информационных систем, которые позволяют обеспечить информационную поддержку процессов принятия решений. Базы данных являются одним из основных элементов большинства информационных систем.

База данных (далее БД) - это организованная структура, в которой в специальном формате хранится информация.

Система управления базой данных (далее СУБД) - это программа, с помощью которой в компьютер вводится информация, просматривается, сортируется и т.д.

БД «Спортивный клуб» предназначена для структуризации информации об шахматистах, их тренерах, турнирах и партиях.

# Проектирование БД

## Анализ и описание предметной области информационной системы

Спортивный клуб – общественная или частная организация, объединяющая спортсменов и любителей спорта.

Целью спортивного клуба является:

* вовлечение граждан в систематические занятия физической культурой и спортом;
* создание условий гражданам для занятий физической культурой и спортом по месту жительства;
* пропаганда спортивного образа жизни, формирование потребности у граждан в систематических занятиях физической культурой и спортом.

Предметная область представляет собой базу данных, направлена на информационную поддержку спортивного клуба. База данных даёт возможность проведение аналитики и создание отчётности по деятельности всего клуба, а также каждого игрока и тренера, автоматизирует некоторые процессы. Ввиду всех возможностей базы данных возникает необходимость в автоматизации информационных процессов.

Входные документы:

* заявка на вступление в клуб;
* расписание турниров;
* расписание партий турнира.

Выходные документы:

* отчет о деятельности клуба.

База данных создана для участников, которые могут просматривать информацию, и руководителей клуба, которые могут просматривать, вносить, изменять, удалять информацию в базе данных в полном объёме.

БД должна осуществлять хранение данных о:

* ведение списков спортсменов и тренеров;
* учёт проводимых соревнований (с ведением их архива);
* учёт травм, полученных спортсменами. Необходимо предусмотреть:
* возможность перехода спортсмена от одного тренера к другому;
* составление рейтингов спортсменов;
* составление рейтингов тренеров;
* выдачу информации по соревнованиям;
* выдачу информации по конкретному спортсмену;
* подбор возможных кандидатур на участие в соревнованиях (соответствующего уровня мастерства, возраста и без травм).

## Проектирование концептуальной модели

Сущности:

1. Тренер - содержит информацию о тренере.

Атрибуты: ФИО.

1. Спортсмен - содержит информацию о спортсмене.

Атрибуты: ФИО, рейтинг.

1. Соревнования - содержит информацию о турнирах.

Атрибуты: дата подписания, комиссионные, дата продажи.

1. Травмы - содержит информацию о травмах полученные спортсменами.

Атрибуты: ФИО, номер телефона, адрес.

Связи:

1. Тренер тренирует спортсменов. Тренер может тренировать много игроков, а у игроков может быть много тренеров (связь много к многим).
2. Спортсмен участвует в турнирах. Спортсмен может принимать участие в нескольких турнирах, а у в турнире принимают участие много спортсменов (связь много к многим).
3. Спортсмен получат травмы. Спортсмен может получить много травм, а травму могут получить много спортсменов (связь много к многим).

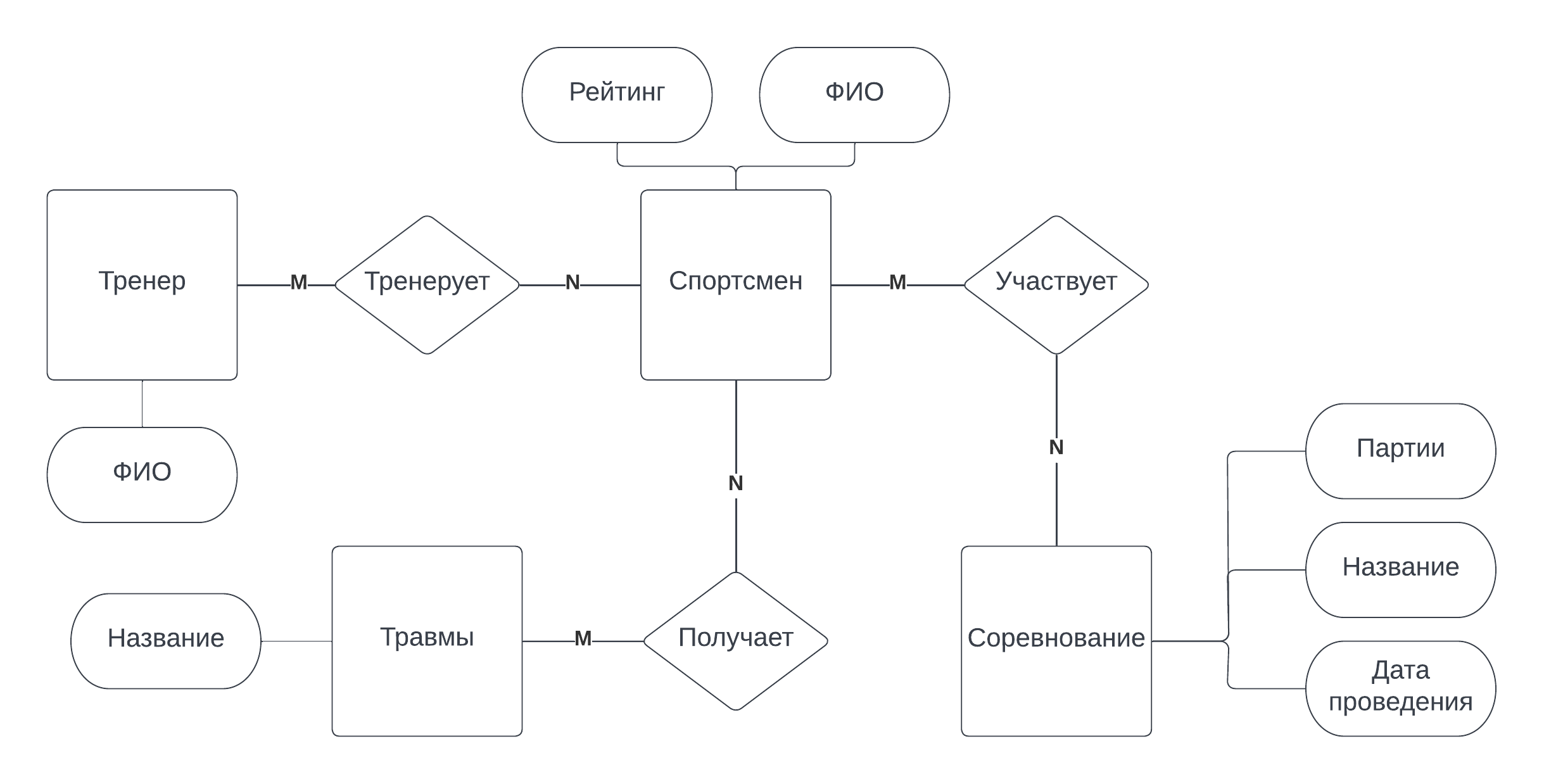


Рисунок 1 – Концептуальная модель

## Построение логической модели, схемы базы данных

1. Призовое место (TopPleces):
   1. Идентификатор призового места (TopPlecesID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Название (TopPlecesName):
* нулевое значение: not null;
* тип поля: nvarchar.

1. Турнир (Event):
   1. Идентификатор турнира (EventID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Название турнира (NameEvent):
* нулевое значение: not null;
* тип поля: nvarchar.
  1. Призовой фонд (PrizeFund):
* нулевое значение: not null;
* тип поля: int;
* маска ввода: больше 0.
  1. Дата начала турнира (DataStart):
* нулевое значение: not null;
* тип поля: date.
  1. Дата окончания турнира (DataFinish):
* нулевое значение: not null;
* тип поля: date.
  1. Идентификатор статуса (StatusID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Status (поле StatusID).

1. Игрок партии (ConsignmentPlayer):
   1. Идентификатор игрока партии (ConsignmentPlayerID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Идентификатор партии (ConsignmentID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Consignment (поле ConsignmentID).
  1. Идентификатор игрока (PlayerID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Player (поле FIDEID).
  1. Игрок играл белыми (IsWhile):
* нулевое значение: not null;
* тип поля: bit.
  1. Результат партии (Result):
* нулевое значение: not null;
* тип поля: float;
* маска ввода: значения 0, 0.5, 1.
  1. Очки рейтинга за партию (Score):
* нулевое значение: not null;
* тип поля: int.

1. Призовые места на турнире (TopPlecesEvent):
   1. Идентификатор призового места турнира (TopPlecesEventID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Идентификатор призового места (TopPlecesID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу TopPleces (поле TopPlecesID).
  1. Идентификатор турнира (EventID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Event (поле EventID).
  1. Идентификатор игрока (PlayerID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Player (поле FIDEID).

1. Тренер (Trainer):
   1. Идентификатор тренера (TrainerID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Фамилия, имя, отчество (FirstName, MiddleName, LastName):
* нулевое значение: FirstName и MiddleName not null, LastName null;
* тип поля: nvarchar;
* нормализированная форма атрибута ФИО.
  1. День рождения (Birthday):
* нулевое значение: not null;
* тип поля: date.

1. Спортсмен (Player):
   1. Идентификатор клиента (FIDEID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, официальный идентификатор шахматиста, установленный ФИДЕ.
  1. Фамилия, имя, отчество (FirstName, MiddleName, LastName):
* нулевое значение: FirstName и MiddleName not null, LastName null;
* тип поля: nvarchar;
* нормализированная форма атрибута ФИО.
  1. День рождения (Birthday):
* нулевое значение: not null;
* тип поля: date.
  1. Рейтинг ЭЛО (ELORating)
* нулевое значение: not null;
* тип поля: int;
* маска ввода: больше 0.

1. Статус турнира (Status):
   1. Идентификатор статуса договора (StatusID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Наименование (Name):
* нулевое значение: not null;
* тип поля: nvarchar.

1. Травмы спортсмена (InjuryPlayer):
   1. Идентификатор травмы спортсмена (InjuryPlayerID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Идентификатор травмы (InjuryID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Injury (поле InjuryID).
  1. Идентификатор спортсмена (PlayerID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Player (поле FIDEID).
  1. Дата получения травмы (DateInjury)
* нулевое значение: not null;
* тип поля: date.

1. Призовые места тренера игрока (TopPlecesEventTrainer):
   1. Идентификатор призовое место тренера игрока (TopPlecesEventTrainerID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Идентификатор тренера (TrainerId):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Trainer (поле TrainerId).
  1. Идентификатор призового места игрока (TopPlecesEventID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу TopPlecesEvent (поле TopPlecesEventID).

1. Тренера игрока (TrainerPlayer):
   1. Идентификатор тренера (TrainerPlayerID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Идентификатор тренера (TrainerId):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Trainer (поле TrainerId).
  1. Идентификатор игрока (PlayerID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Player (поле FIDEID).

1. Игроки турнира (EventPlayer):
   1. Идентификатор игроков турнира (EventPlayerID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Идентификатор турнира (EventID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Event(поле EventID).
  1. Идентификатор призового места игрока (PlayerID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Player (поле FIDEID).

1. Партия (Consignment):
   1. Идентификатор партии (ConsignmentID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Идентификатор турнира (EventID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Event (поле EventID).
  1. Начало партии (DateStart):
* нулевое значение: not null;
* тип поля: date.
  1. Окончание партии (DateFinish):
* нулевое значение: not null;
* тип поля: date.

1. Травма (Injury):
   1. Идентификатор травмы (InjuryID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Идентификатор турнира (EventID):
* нулевое значение: not null;
* тип поля: nvarchar.

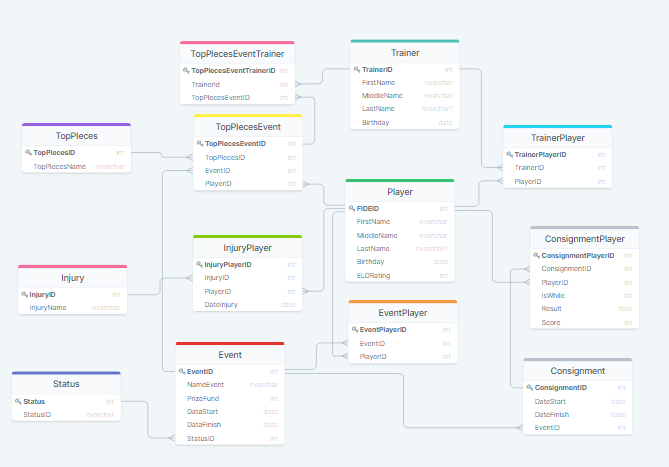


Рисунок 2 – Логическая модель

## Выбор СУБД

СУБД предназначена для централизованного управления базой данных в интересах всех работающих в этой системе. Используемые в настоящее время СУБД, обладающих средствами обеспечения целостности данных и надёжной безопасности, что даёт возможность разработчикам гарантировать большую безопасность данных при меньших затратах сил на низкоуровневое программирование.

В качестве средства реализации БД была выбрана система Microsoft SQL Server Management Studio 2019 (далее MSSQL).

MSSQL– это мощная и надежная система управления данными, обеспечивающая множество функций, защиту данных и высокую производительность для внедренных приложений-клиентов и локальных хранилищ данных. MSSQL предназначен для упрощенного развертывания и быстрого создания прототипов; его можно получить бесплатно и свободно распространять вместе с приложениями. Также можно сказать, что он удовлетворяет требованиям, предъявляемым к системам распределенной обработки информации. Эта СУБД поддерживает: тиражирование данных, параллельную обработку, создание и обработку больших баз данных на недорогих аппаратных платформах.

MSSQL упрощает развёртывание, управление и оптимизацию данных предприятия и аналитических приложений. Как платформа управления данными предприятия, он предоставляет единую консоль управления, которая позволяет администраторам данных, находящимся в любом месте вашей организации, отслеживать, управлять и настраивать все базы данных и связанные службы по всему предприятию. Он предоставляет расширяемую инфраструктуру управления, которая может быть легко запрограммирована при помощи SQL Management Objects, позволяя пользователям переделывать и расширять их среду управления и независимым поставщикам программных продуктов создавать дополнительные инструменты и функциональность для дальнейшего расширения возможностей, поставляемых по умолчанию.

Средством, связывающим клиента с сервером, является язык SQL - язык структурированных запросов. В данном курсовом проекте были использованы основные операторы этого языка. Этот язык является очень распространенным на данный момент и имеет ряд особенностей.

# Реализация базы данных

## Физическое проектирование БД

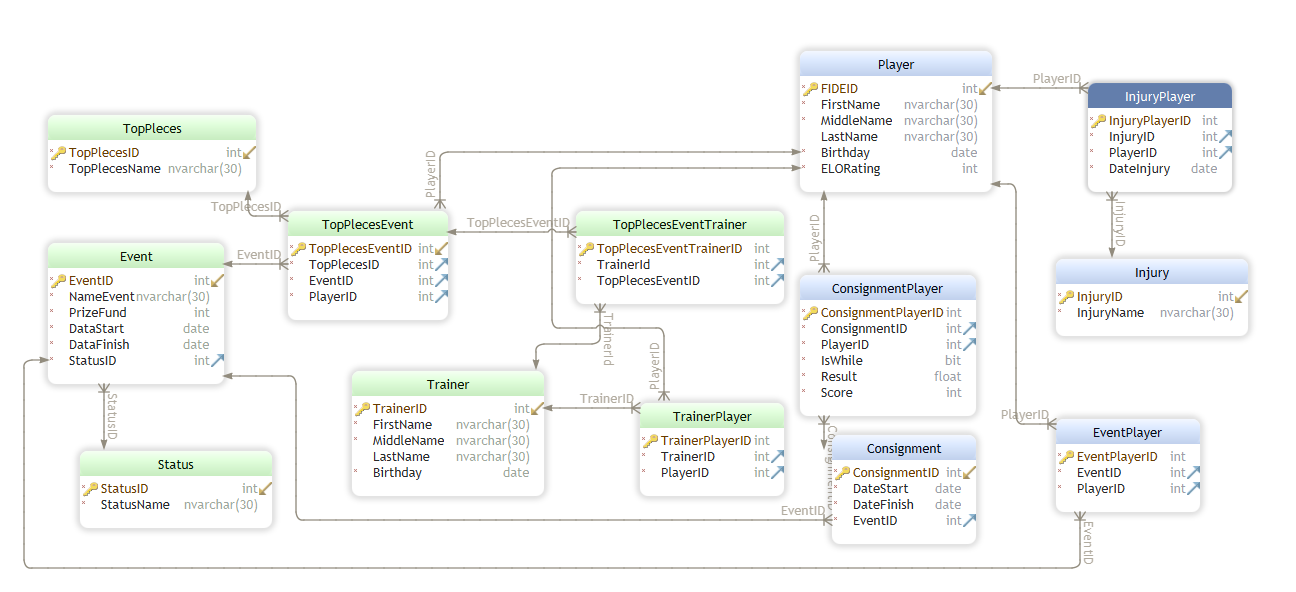


Рисунок 3 – Диаграмма базы данных

## Написание исходного кода БД

* 1. Injury

Данная таблица создана с целью хранения информации о травме.

Листинг:

CREATE TABLE Injury(

InjuryID INT identity(1,1) NOT NULL,

InjuryName NVARCHAR(30) NOT NULL

);

ALTER TABLE

Injury ADD CONSTRAINT injury\_injuryid\_primary PRIMARY KEY(InjuryID);

* 1. TopPleces

Данная таблица создана с целью хранения информации о призовых местах.

Листинг:

CREATE TABLE TopPleces(

TopPlecesID INT identity(1,1) NOT NULL,

TopPlecesName NVARCHAR(30) NOT NULL

);

ALTER TABLE

TopPleces ADD CONSTRAINT toppleces\_topplecesid\_primary PRIMARY KEY(TopPlecesID);

* 1. Status

Данная справочная таблица создана с целью хранения информации о статусе турнира.

Листинг:

CREATE TABLE Status(

StatusID INT identity(1,1) NOT NULL,

StatusName NVARCHAR(30) NOT NULL

);

ALTER TABLE

Status ADD CONSTRAINT status\_status\_primary PRIMARY KEY(StatusID);

* 1. Trainer

Данная таблица создана с целью хранения информации о тренере

Листинг:

CREATE TABLE Trainer(

TrainerID INT identity(1,1) NOT NULL,

FirstName NVARCHAR(30) NOT NULL,

MiddleName NVARCHAR(30) NOT NULL,

LastName NVARCHAR(30) NULL,

Birthday DATE NOT NULL

);

ALTER TABLE

Trainer ADD CONSTRAINT trainer\_trainerid\_primary PRIMARY KEY(TrainerID);

* 1. TrainerPlayer

Данная таблица создана с целью хранения информации о тренере игрока и связи таблиц Trainer и Player.

Листинг:

CREATE TABLE TrainerPlayer(

TrainerPlayerID INT identity(1,1) NOT NULL,

TrainerID INT NOT NULL,

PlayerID INT NOT NULL

);

ALTER TABLE

TrainerPlayer ADD CONSTRAINT trainerplayer\_trainerplayerid\_primary PRIMARY KEY(TrainerPlayerID);

ALTER TABLE

TrainerPlayer ADD CONSTRAINT trainerplayer\_playerid\_foreign FOREIGN KEY(PlayerID) REFERENCES Player(FIDEID);

ALTER TABLE

TrainerPlayer ADD CONSTRAINT trainerplayer\_trainerid\_foreign FOREIGN KEY(TrainerID) REFERENCES Trainer(TrainerID);

* 1. InjuryPlayer

Данная таблица создана с целью хранения информации о травмах спортсменов и связи таблиц Injury и Player.

Листинг:

CREATE TABLE InjuryPlayer(

InjuryPlayerID INT identity(1,1) NOT NULL,

InjuryID INT NOT NULL,

PlayerID INT NOT NULL,

DateInjury DATE NOT NULL

);

ALTER TABLE

InjuryPlayer ADD CONSTRAINT injuryplayer\_injuryplayerid\_primary PRIMARY KEY(InjuryPlayerID);

ALTER TABLE

InjuryPlayer ADD CONSTRAINT injuryplayer\_playerid\_foreign FOREIGN KEY(PlayerID) REFERENCES Player(FIDEID);

ALTER TABLE

InjuryPlayer ADD CONSTRAINT injuryplayer\_injuryid\_foreign FOREIGN KEY(InjuryID) REFERENCES Injury(InjuryID);

* 1. Player

Данная таблица создана с целью хранения информации о спортсмене.

Листинг:

CREATE TABLE Player(

FIDEID INT NOT NULL,

FirstName NVARCHAR(30) NOT NULL,

MiddleName NVARCHAR(30) NOT NULL,

LastName NVARCHAR(30) NULL,

Birthday DATE NOT NULL,

ELORating INT NOT NULL default 1000

);

ALTER TABLE

Player ADD CONSTRAINT player\_fideid\_primary PRIMARY KEY(FIDEID);

alter table

Player ADD CONSTRAINT Player\_check\_ELORating CHECK (ELORating >= 0)

alter table

Player ADD CONSTRAINT Player\_check\_FIDEID CHECK (DATALENGTH(CAST(FIDEID AS varchar(7))) = 7)

* 1. TopPlecesEvent

Данная таблица создана с целью хранения информации о призовых местах турнира и связи таблиц TopPleces и Event.

Листинг:

CREATE TABLE TopPlecesEvent(

TopPlecesEventID INT identity(1,1) NOT NULL,

TopPlecesID INT NOT NULL,

EventID INT NOT NULL,

PlayerID INT NOT NULL

);

ALTER TABLE

TopPlecesEvent ADD CONSTRAINT topplecesevent\_toppleceseventid\_primary PRIMARY KEY(TopPlecesEventID);

ALTER TABLE

TopPlecesEvent ADD CONSTRAINT topplecesevent\_topplecesid\_foreign FOREIGN KEY(TopPlecesID) REFERENCES TopPleces(TopPlecesID);

ALTER TABLE

TopPlecesEvent ADD CONSTRAINT topplecesevent\_playerid\_foreign FOREIGN KEY(PlayerID) REFERENCES Player(FIDEID);

ALTER TABLE

TopPlecesEvent ADD CONSTRAINT topplecesevent\_eventid\_foreign FOREIGN KEY(EventID) REFERENCES Event(EventID);

* 1. EventPlayer

Данная таблица создана с целью хранения информации о игроки турнира и связи таблиц Event и Player.

Листинг:

CREATE TABLE EventPlayer(

EventPlayerID INT identity(1,1) NOT NULL,

EventID INT NOT NULL,

PlayerID INT NOT NULL

);

ALTER TABLE

EventPlayer ADD CONSTRAINT eventplayer\_eventplayerid\_primary PRIMARY KEY(EventPlayerID);

ALTER TABLE

EventPlayer ADD CONSTRAINT eventplayer\_playerid\_foreign FOREIGN KEY(PlayerID) REFERENCES Player(FIDEID);

ALTER TABLE

EventPlayer ADD CONSTRAINT eventplayer\_eventid\_foreign FOREIGN KEY(EventID) REFERENCES Event(EventID);

* 1. Event

Данная таблица создана с целью хранения информации о турнирах.

Листинг:

CREATE TABLE Event(

EventID INT identity(1,1) NOT NULL,

NameEvent NVARCHAR(30) NOT NULL,

PrizeFund INT NOT NULL,

DataStart DATE NOT NULL,

DataFinish DATE NOT NULL,

StatusID INT NOT NULL

);

ALTER TABLE

Event ADD CONSTRAINT event\_eventid\_primary PRIMARY KEY(EventID);

ALTER TABLE

Event ADD CONSTRAINT event\_statusid\_foreign FOREIGN KEY(StatusID) REFERENCES Status(StatusID);

alter table

Event ADD CONSTRAINT Event\_check\_PrizeFund CHECK (PrizeFund > 0)

* 1. Consignment

Данная таблица создана с целью хранения информации о партиях.

Листинг:

CREATE TABLE Consignment(

ConsignmentID INT identity(1,1) NOT NULL,

DateStart DATE NOT NULL,

DateFinish DATE NOT NULL,

EventID INT NOT NULL

);

ALTER TABLE

Consignment ADD CONSTRAINT consignment\_consignmentid\_primary PRIMARY KEY(ConsignmentID);

ALTER TABLE

Consignment ADD CONSTRAINT consignment\_eventid\_foreign FOREIGN KEY(EventID) REFERENCES Event(EventID);

* 1. ConsignmentPlayer

Данная таблица создана с целью хранения информации о игроках партии и связи таблиц Consignment и Player.

Листинг:

CREATE TABLE ConsignmentPlayer(

ConsignmentPlayerID INT identity(1,1) NOT NULL,

ConsignmentID INT NOT NULL,

PlayerID INT NOT NULL,

IsWhile BIT NOT NULL,

Result FLOAT NOT NULL,

Score INT NOT NULL

);

ALTER TABLE

ConsignmentPlayer ADD CONSTRAINT consignmentplayer\_consignmentplayerid\_primary PRIMARY KEY(ConsignmentPlayerID);

ALTER TABLE

ConsignmentPlayer ADD CONSTRAINT consignmentplayer\_playerid\_foreign FOREIGN KEY(PlayerID) REFERENCES Player(FIDEID);

ALTER TABLE

ConsignmentPlayer ADD CONSTRAINT consignmentplayer\_consignmentid\_foreign FOREIGN KEY(ConsignmentID) REFERENCES Consignment(ConsignmentID);

alter table

ConsignmentPlayer ADD CONSTRAINT ConsignmentPlayer\_check\_Result CHECK (Result in (0, 0.5, 1))

* 1. TopPlecesEventTrainer

Данная таблица создана с целью хранения информации о призовые места тренера игрока и связи таблиц Trainer и TopPlecesEvent.

Листинг:

CREATE TABLE TopPlecesEventTrainer(

TopPlecesEventTrainerID INT identity(1,1) NOT NULL,

TrainerId INT NOT NULL,

TopPlecesEventID INT NOT NULL

);

ALTER TABLE

TopPlecesEventTrainer ADD CONSTRAINT toppleceseventtrainer\_toppleceseventtrainerid\_primary PRIMARY KEY(TopPlecesEventTrainerID);

ALTER TABLE

TopPlecesEventTrainer ADD CONSTRAINT toppleceseventtrainer\_trainerid\_foreign FOREIGN KEY(TrainerId) REFERENCES Trainer(TrainerID);

ALTER TABLE

TopPlecesEventTrainer ADD CONSTRAINT toppleceseventtrainer\_toppleceseventid\_foreign FOREIGN KEY(TopPlecesEventID) REFERENCES TopPlecesEvent(TopPlecesEventID);

## Написание запросов, функций и хранимых процедур

1. InjuryAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Injury и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице Injury по названию и типу.

Листинг процедуры:

create PROCEDURE InjuryAction (

@Action tinyint,

@InjuryID INT,

@InjuryName NVARCHAR(30)

)

As

Begin

if(@Action = 1)

delete Injury where InjuryID = @InjuryID

else

if(@Action = 2)

insert Injury (InjuryName)

values (@InjuryName)

else

if(@Action = 3)

UPDATE [dbo].[Injury]

SET InjuryName = @InjuryName

where InjuryID = @InjuryID

end

Пример использования:

* Удаление: execute InjuryAction 1,2,’’

При удалении важно указать первым (@Action) 1 и вторым (@InjuryID) существующий InjuryID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute InjuryAction 2,1,’qwe’

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute InjuryAction 3,1,’wqw’

При обновлении важно указать первым (@Action) 3 и вторым (@InjuryID) существующий InjuryID, а остальное заполнить нужными значениями.

1. TopPlecesAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу TopPleces и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице TopPleces по названию и типу.

Листинг процедуры:

create PROCEDURE TopPlecesAction (

@Action tinyint,

@TopPlecesID INT,

@TopPlecesName NVARCHAR(30)

)

As

Begin

if(@Action = 1)

delete TopPleces where TopPlecesID = @TopPlecesID

else

if(@Action = 2)

insert TopPleces (TopPlecesName)

values (@TopPlecesName)

else

if(@Action = 3)

UPDATE [dbo].[TopPleces]

SET TopPlecesName = @TopPlecesName

where TopPlecesID = @TopPlecesID

end

Пример использования:

* Удаление: execute TopPlecesAction 1,2,’’

При удалении важно указать первым (@Action) 1 и вторым (@TopPlecesID) существующий TopPlecesID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute TopPlecesAction 2,1, ’qw’

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute TopPlecesAction 3,1, ’qw’

При обновлении важно указать первым (@Action) 3 и вторым (@TopPlecesID) существующий TopPlecesID, а остальное заполнить нужными значениями.

1. StatusAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Status и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице Status по названию и типу.

Листинг процедуры:

Create PROCEDURE StatusAction (

@Action tinyint,

@StatusID INT,

@Name NVARCHAR(30)

)

As

Begin

if(@Action = 1)

Delete Status where StatusID = @StatusID

else

if(@Action = 2)

insert Status (NameStatus)

values (@Name)

else

if(@Action = 3)

UPDATE [Dbo].[Status]

SET Name = @Name

where StatusID = @StatusID

end

Пример использования:

* Удаление: execute StatusAction 1,2,’’

При удалении важно указать первым (@Action) 1 и вторым (@StatusID) существующий StatusID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute StatusAction 2,1,’’

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute StatusAction 3,1,’’

При обновлении важно указать первым (@Action) 3 и вторым (@StatusID) существующий StatusID, а остальное заполнить нужными значениями.

1. TrainerAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Trainer и включает следующие переменные: @Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице Trainer по названию и типу.

Листинг процедуры:

create PROCEDURE TrainerAction (

@Action tinyint,

@TrainerID INT,

@FirstName NVARCHAR(30),

@MiddleName NVARCHAR(30),

@LastName NVARCHAR(30),

@Birthday DATE

)

As

Begin

if(@Action = 1)

delete Trainer where TrainerID = @TrainerID

else

if(@Action = 2)

insert Trainer (FirstName,MiddleName,LastName,Birthday)

values (@FirstName,@MiddleName,@LastName,@Birthday)

else

if(@Action = 3)

UPDATE [dbo].[Trainer]

SET FirstName = @FirstName

,MiddleName = @MiddleName

,LastName = @LastName

,Birthday = @Birthday

where TrainerID = @TrainerID

end

Пример использования:

* Удаление: execute TrainerAction 1,2,0,null, null, null, null

При удалении важно указать первым (@Action) 1 и вторым (@TrainerID) существующий TrainerID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute TrainerAction 2,1, 1, ’qw’, ’fr’, ’saD’, 12.12.2012

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute TrainerAction 3,1, , 1, ’qw’, ’fr’, ’saD’, 12.12.2012

При обновлении важно указать первым (@Action) 3 и вторым (@TrainerID) существующий TrainerID, а остальное заполнить нужными значениями.

1. TrainerPlayerAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу TrainerPlayer и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице TrainerPlayer по названию и типу.

Листинг процедуры:

create PROCEDURE TrainerPlayerAction (

@Action tinyint,

@TrainerPlayerID INT,

@TrainerID INT,

@PlayerID INT

)

As

Begin

if(@Action = 1)

delete TrainerPlayer where TrainerPlayerID = @TrainerPlayerID

else

if(@Action = 2)

insert TrainerPlayer (TrainerID,PlayerID)

values (@TrainerID,@PlayerID)

else

if(@Action = 3)

UPDATE [dbo].[TrainerPlayer]

SET TrainerID = @TrainerID

,PlayerID = @PlayerID

where TrainerPlayerID = @TrainerPlayerID

end

Пример использования:

* Удаление: execute TrainerPlayerAction 1,2,0,0

При удалении важно указать первым (@Action) 1 и вторым (@TrainerPlayerID) существующий TrainerPlayerID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute TrainerPlayerAction 2,1,1,2

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute TrainerPlayerAction 3, 1,1,2

При обновлении важно указать первым (@Action) 3 и вторым (@TrainerPlayerID) существующий TrainerPlayerID, а остальное заполнить нужными значениями.

1. InjuryPlayerAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу InjuryPlayer и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице InjuryPlayer по названию и типу.

Листинг процедуры:

create PROCEDURE InjuryPlayerAction (

@Action tinyint,

@InjuryPlayerID INT,

@InjuryID INT,

@PlayerID INT,

@DateInjury DATE

)

As

Begin

if(@Action = 1)

delete InjuryPlayer where InjuryPlayerID = @InjuryPlayerID

else

if(@Action = 2)

insert InjuryPlayer (InjuryID,PlayerID,DateInjury)

values (@InjuryID,@PlayerID,@DateInjury)

else

if(@Action = 3)

UPDATE [dbo].[InjuryPlayer]

SET InjuryID = @InjuryID

,PlayerID = @PlayerID

,DateInjury = @DateInjury

where InjuryPlayerID = @InjuryPlayerID

end

Пример использования:

* Удаление: execute InjuryPlayerAction 1,2,0,0,null

При удалении важно указать первым (@Action) 1 и вторым (@InjuryPlayerID) существующий InjuryPlayerID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute InjuryPlayerAction 2,1,1,’12.12.2003’

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute InjuryPlayerAction 3,1,1,’12.12.2003’

При обновлении важно указать первым (@Action) 3 и вторым (@InjuryPlayerID) существующий InjuryPlayerID, а остальное заполнить нужными значениями.

1. PlayerAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Player и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице Player по названию и типу.

Листинг процедуры:

create PROCEDURE PlayerAction (

@Action tinyint,

@FIDEID INT,

@FirstName NVARCHAR(30),

@MiddleName NVARCHAR(30),

@LastName NVARCHAR(30),

@Birthday DATE,

@ELORating INT

)

As

Begin

if(@Action = 1)

delete Player where FIDEID = @FIDEID

else

if(@Action = 2)

insert Player (FIDEID,FirstName,MiddleName,LastName,Birthday,ELORating)

values (@FIDEID,@FirstName,@MiddleName,@LastName,@Birthday,@ELORating)

else

if(@Action = 3)

UPDATE [dbo].[Player]

SET FIDEID = @FIDEID

,FirstName = @FirstName

,MiddleName = @MiddleName

,LastName = @LastName

,Birthday = @Birthday

,ELORating = @ELORating

where FIDEID = @FIDEID

end

Пример использования:

* Удаление: execute PlayerAction 1,2121211,null,null,null,null,null

При удалении важно указать первым (@Action) 1 и вторым (@FIDEID) существующий FIDEID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute PlayerAction 2, 2121211,’sdf’,’sd’,’sdf’,’12.12.2012’,100

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute PlayerAction 3, 2121211,’sdf’,’sd’,’sdf’,’12.12.2012’,100

При обновлении важно указать первым (@Action) 3 и вторым (@FIDEID) существующий FIDEID, а остальное заполнить нужными значениями.

1. TopPlecesEventAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу TopPlecesEvent и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице TopPlecesEvent по названию и типу.

Листинг процедуры:

create PROCEDURE TopPlecesEventAction (

@Action tinyint,

@TopPlecesEventID INT,

@TopPlecesID INT,

@EventID INT,

@PlayerID INT

)

As

Begin

if(@Action = 1)

delete TopPlecesEvent where TopPlecesEventID = @TopPlecesEventID

else

if(@Action = 2)

insert TopPlecesEvent (TopPlecesID,EventID,PlayerID)

values (@TopPlecesID,@EventID,@PlayerID)

else

if(@Action = 3)

UPDATE [dbo].[TopPlecesEvent]

SET TopPlecesID = @TopPlecesID

,EventID = @EventID

,PlayerID = @PlayerID

where TopPlecesEventID = @TopPlecesEventID

end

Пример использования:

* Удаление: execute TopPlecesEventAction 1,2,1,1,1

При удалении важно указать первым (@Action) 1 и вторым (@TopPlecesEventID) существующий TopPlecesEventID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute TopPlecesEventAction 2, 1,1,1,1

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute TopPlecesEventAction 3,1,1,1,1

При обновлении важно указать первым (@Action) 3 и вторым (@TopPlecesEventID) существующий TopPlecesEventID, а остальное заполнить нужными значениями.

1. EventPlayerAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу EventPlayer и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице EventPlayer по названию и типу.

Листинг процедуры:

create PROCEDURE EventPlayerAction (

@Action tinyint,

@EventPlayerID INT,

@EventID INT,

@PlayerID INT

)

As

Begin

if(@Action = 1)

delete EventPlayer where EventPlayerID = @EventPlayerID

else

if(@Action = 2)

insert EventPlayer (EventID,PlayerID)

values (@EventID,@PlayerID)

else

if(@Action = 3)

UPDATE [dbo].[EventPlayer]

SET EventID = @EventID

,PlayerID = @PlayerID

where EventPlayerID = @EventPlayerID

end

Пример использования:

* Удаление: execute EventPlayerAction 1,1,1,1

При удалении важно указать первым (@Action) 1 и вторым (@EventPlayerID) существующий EventPlayerID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute EventPlayerAction 2, 1,1,1

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute EventPlayerAction 3, 1,1,1

При обновлении важно указать первым (@Action) 3 и вторым (@EventPlayerID) существующий EventPlayerID, а остальное заполнить нужными значениями.

1. EventAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Event и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице Event по названию и типу.

Листинг процедуры:

create PROCEDURE EventAction (

@Action tinyint,

@EventID INT,

@NameEvent NVARCHAR(30),

@PrizeFund INT,

@DataStart DATE,

@DataFinish DATE,

@StatusID INT

)

As

Begin

if(@Action = 1)

delete Event where EventID = @EventID

else

if(@Action = 2)

insert Event (NameEvent,PrizeFund,DataStart,DataFinish,StatusID)

values (@NameEvent,@PrizeFund,@DataStart,@DataFinish,@StatusID)

else

if(@Action = 3)

UPDATE [dbo].[Event]

SET NameEvent = @NameEvent

,PrizeFund = @PrizeFund

,DataStart = @DataStart

,DataFinish = @DataFinish

,StatusID = @StatusID

where EventID = @EventID

end

Пример использования:

* Удаление: execute EventAction 1,1,null,0,null,null,0

При удалении важно указать первым (@Action) 1 и вторым (@EventID) существующий EventID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute EventAction 2, 1, ‘esseg’,10,’12.12.2003’, ’22.12.2003’,1

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute EventAction 3, 1, ‘esseg’,10,’12.12.2003’, ’22.12.2003’,1

При обновлении важно указать первым (@Action) 3 и вторым (@EventID) существующий EventID, а остальное заполнить нужными значениями.

1. ConsignmentAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Consignment и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице Consignment по названию и типу.

Листинг процедуры:

create PROCEDURE ConsignmentAction (

@Action tinyint,

@ConsignmentID INT,

@DateStart DATE,

@DateFinish DATE,

@EventID INT

)

As

Begin

if(@Action = 1)

delete Consignment where ConsignmentID = @ConsignmentID

else

if(@Action = 2)

insert Consignment (DateStart,DateFinish,EventID)

values (@DateStart,@DateFinish,@EventID)

else

if(@Action = 3)

UPDATE [dbo].[Consignment]

SET DateStart = @DateStart

,DateFinish = @DateFinish

,EventID = @EventID

where ConsignmentID = @ConsignmentID

end

Пример использования:

* Удаление: execute ConsignmentAction 1,1,null,null,0

При удалении важно указать первым (@Action) 1 и вторым (@ConsignmentID) существующий ConsignmentID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute ConsignmentAction 2, 1,’12.12.2003’, ’22.12.2003’,1

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute ConsignmentAction 3, 1,’12.12.2003’, ’22.12.2003’,1

При обновлении важно указать первым (@Action) 3 и вторым (@ConsignmentID) существующий ConsignmentID, а остальное заполнить нужными значениями.

1. ConsignmentPlayerAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу ConsignmentPlayer и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице ConsignmentPlayer по названию и типу.

Листинг процедуры:

create PROCEDURE ConsignmentPlayerAction (

@Action tinyint,

@ConsignmentPlayerID INT,

@ConsignmentID INT,

@PlayerID INT,

@IsWhile BIT,

@Result FLOAT,

@Score INT

)

As

Begin

if(@Action = 1)

delete ConsignmentPlayer where ConsignmentPlayerID = @ConsignmentPlayerID

else

if(@Action = 2)

insert ConsignmentPlayer (ConsignmentID,PlayerID,IsWhile,Result,Score)

values (@ConsignmentID,@PlayerID,@IsWhile,@Result,@Score)

else

if(@Action = 3)

UPDATE [dbo].[ConsignmentPlayer]

SET ConsignmentID = @ConsignmentID

,PlayerID = @PlayerID

,IsWhile = @IsWhile

,Result = @Result

,Score = @Score

where ConsignmentPlayerID = @ConsignmentPlayerID

end

Пример использования:

* Удаление: execute ConsignmentPlayerAction 1,1,1,1,0,-12

При удалении важно указать первым (@Action) 1 и вторым (@ConsignmentPlayerID) существующий ConsignmentPlayerID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute ConsignmentPlayerAction 2, 1, 1,1,0,-12

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute ConsignmentPlayerAction 3, 1, 1,1,0,-12

При обновлении важно указать первым (@Action) 3 и вторым (@ConsignmentPlayerID) существующий ConsignmentPlayerID, а остальное заполнить нужными значениями.

1. TopPlecesEventTrainerAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу TopPlecesEventTrainer и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице TopPlecesEventTrainer по названию и типу.

Листинг процедуры:

create PROCEDURE TopPlecesEventTrainerAction (

@Action tinyint,

@TopPlecesEventTrainerID INT,

@TrainerId INT,

@TopPlecesEventID INT

)

As

Begin

if(@Action = 1)

delete TopPlecesEventTrainer where TopPlecesEventTrainerID = @TopPlecesEventTrainerID

else

if(@Action = 2)

insert TopPlecesEventTrainer (TrainerId,TopPlecesEventID)

values (@TrainerId,@TopPlecesEventID)

else

if(@Action = 3)

UPDATE [dbo].[TopPlecesEventTrainer]

SET TrainerId = @TrainerId

,TopPlecesEventID = @TopPlecesEventID

where TopPlecesEventTrainerID = @TopPlecesEventTrainerID

end

Пример использования:

* Удаление: execute TopPlecesEventTrainerAction 1,1,null,null

При удалении важно указать первым (@Action) 1 и вторым (@TopPlecesEventTrainerID) существующий TopPlecesEventTrainerID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute TopPlecesEventTrainerAction 2, 1,1,1

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute TopPlecesEventTrainerAction 3, 1,1,1

При обновлении важно указать первым (@Action) 3 и вторым (@TopPlecesEventTrainerID) существующий TopPlecesEventTrainerID, а остальное заполнить нужными значениями.

1. TopTrainer

Описание: процедура создана для вывода рейтинга трененров.

Листинг:

create PROCEDURE TopTrainer

As

Begin

select t.\*,CountEvent from Trainer t

left join (select

TrainerID,

Count(TopPlecesEventID) as CountEvent

from TopPlecesEventTrainer group by TrainerID) tce on t.TrainerID = tce.TrainerId

end

Выходные данные: все поля таблицы Trainer, CountEvent.

1. EventInfo

Описание: процедура создана для вывода информации по турнирам.

Листинг:

create PROCEDURE EventInfo

As

Begin

select e.\*,CountPlayers,PlayerID from Event e

inner join (select

EventID,

Count(PlayerID) as CountPlayers

from EventPlayer group by EventID) tce on e.EventID = tce.EventID

inner join EventPlayer ep on e.EventID = ep.EventID

inner join Player p on ep.PlayerID = p.FIDEID

end

Выходные данные: все поля таблицы Event, CountEventPlayer, PlayerID.

1. Condidate

Описание: процедура создана для вывода информации по всем договорам. Переменные: @YearStart возраст начало диапазона,@YearFinish возраст конец диапозона,@ELORating рейтинг ЭЛО

Листинг:

create PROCEDURE Condidate

(

@YearStart int,

@YearFinish int,

@ELORating int

)

As

Begin

select FIDEID, CountInjury from Player e

left join (select

PlayerID,

Count(PlayerID) as CountInjury

from InjuryPlayer group by PlayerID) tce on e.FIDEID = tce.PlayerID

where Year(GETDATE())-Year(Birthday) >= @YearStart and Year(GETDATE())-Year(Birthday) <= @YearFinish and ELORating >= @ELORating

end

Выходные данные: FIDEID, CountInjury.

1. checkInsertConsignment

Триггер создан как ограничение таблицы Consignment на вставку и обновление полей с условием: дата начала партии не может больше даты окончания партии.

Листинг:

create trigger checkInsertConsignment on Consignment

for insert, update

as

declare @DateStart date, @DateFinish date

Select @DateStart=DateStart, @DateFinish=DateFinish

from inserted

if (@DateFinish >= @DateStart)

Begin

print 'Необходимо правильно указать дату!'

ROLLBACK TRANSACTION

End

1. checkInsertEvent

Триггер создан как ограничение таблицы Event на вставку и обновление полей с условием: дата начала турнира не может больше даты окончания турнира.

Листинг:

create trigger checkInsertEvent on Event

for insert, update

as

declare @DateStart date, @DateFinish date

Select @DateStart=DateStart, @DateFinish=DateFinish

from inserted

if (@DateFinish >= @DateStart)

Begin

print 'Необходимо правильно указать дату!'

ROLLBACK TRANSACTION

End

# Заключение

В данной курсовой работе был проведен детальный анализ предметной области спортивный клуб. Раскрыто само понятие БД и способы его построения в соответствии с основными этапами проектирования, осуществляемыми в рамках современных информационных технологий.

В конечном итоге, на основе MSSQL была разработана база данных для спортивного клуба.

Разрабатываемая БД поможет отслеживать информацию о каждом игроке и тренере.

Использование БД в реальном клубе позволит автоматизировать деятельность сотрудников. Создаст благоприятные условия труда, позволит всегда отражать актуальную информацию.

Поставленная цель курсовой работы достигнута, задачи решены в полном объеме.

# Список используемой литературы

1. <https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%BE%D1%80%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BB%D1%83%D0%B1>
2. <https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D0%B4%D1%83%D0%BD%D0%B0%D1%80%D0%BE%D0%B4%D0%BD%D0%B0%D1%8F_%D1%88%D0%B0%D1%85%D0%BC%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D1%84%D0%B5%D0%B4%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D1%8F>
3. <https://www.fide.com/>