Вариант №28

Спортсмены и команды  
(athletes and team)

Хижний Евгений, Василин Юрий| 21 группа

***Будем продолжать работу с целью улучшить оценку***

# Описание задания

# 

# Постановка задачи

1. Выбрать вариант. Каждое задание выполняется двумя студентами.
2. Согласовать выбранный вариант у старосты так, чтобы не было пересечения с другими подгруппами из 2 человек. Описание (копия) задания – это первый раздел отчёта. Обратите внимание на то, что нотации на диаграммах могут быть различными (см. лекции).
3. Выяснить семантику данных в выбранном варианте. Описать бизнес, который моделирует ваша схема/схемы данных в виде диаграммы Use-Case. Это второй раздел отчёта.
4. Составить словарик использованных в базе терминов (имен атрибутов, отношений/сущностей/таблиц) и терминов бизнеса. Этот словарь входит в отчет по работе.
5. Указать ограничения возможностей схемы данных из задания. Описание ограничений также входит в отчет по работе.
6. Используя любой доступный нам инструмент (ERWin, Astach и др.) переделать схему базы данных. Убрать абстрактные имена столбцов, заменив их на реальные имена. Уточнить или пополните схему. В том числе уточнить ограничения целостности и необходимость использования триггеров.
7. Преобразованная схема базы (более точно, концептуальная и физическая схемы) входит в отчет в виде диаграмм и скриптов.
8. Написать три - пять существенно различных сложных запросов с соединениями таблиц и подзапросами. Это последний раздел отчёта.

# Описание бизнеса. Семантика.

# Моделируемой областью являются студенты-спортсмены и команды в средней школе.

# База данных для отдела легкой атлетики хранит следующие данные:

# Профайл для команд, с гендерным разделением, тренерами, дивизионом;

# Профайл для студентов;

# Список спортивных событий;

# Список студенческих грамот;

Отдельные спортсмены могут отбираться в команду. Спортсмен может состоять в одной, нескольких командах либо не состоять вовсе. Команда находится в дивизионе.

Кроме того, необходимо следить за баллами, начисляемыми каждому студенту-спортсмену за участие в том или ином виде спорта для вручения студенческих грамот.

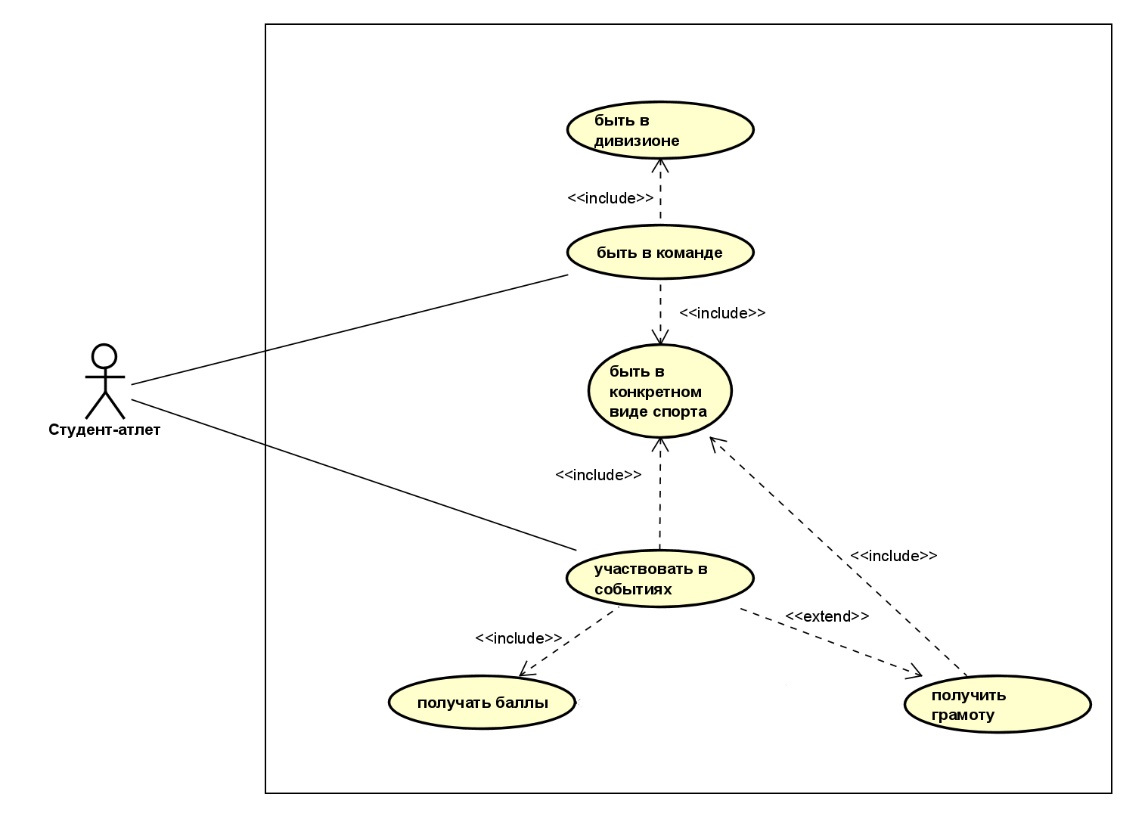
База данных должна отслеживать студентов-спортсменов в течение 5 лет с любыми данными. Спортсмен может участвовать в нескольких видах спорта в течение обучения.

Мы решили (исходя из представленной таблицы), что студенческую грамоту студент-спортсмен получает после 5 лет обучения при достижении определенных балов, которые даются за участие вспортивных соревнованиях.

Итак, возможны два варианта развития событий:

1. Командный спорт
   1. Студент вступает в команду
   2. Команда участвует в определённом виде спорта
3. Студент участвует в событиях
4. За участие он получает баллы и грамоты
5. Грамота присуждается в определённом виде спорта

# Use-case диаграмма



# Словарь использованных в базе терминов

***Виды спорта (Sports)*** - таблица описывает вид спорта, к которому могут принадлежать:

1. команды(teams)
2. спортивные события(events)
3. студенческие грамоты (student letters)

и имеет следующие атрибуты:

1. код вида спорта - **sport\_code** (тип целочисленный) - первичный ключ
2. название вида спорта - **sport\_description** (тип строковый)

***Студенты-атлеты (Student\_athletes)***­­­­­­­­­­­­­ - таблица описывает определенного студента-атлета и имеет следующие атрибуты:

1. идентификатор студента - **student\_id** (тип целочисленный) - первичный ключ
2. очки на сегодняшний день - **points\_awarded\_to\_date** (тип целочисленный) – все очки, заработанные студентом за участия в спортивных событиях
3. имя - **first\_name** (тип строковый)
4. второе личное имя – **middle\_name** (тип строковый)
5. фамилия - **last\_name** (тип строковый)
6. дата рождения - **date\_of\_birth** (тип дата)
7. пол - **gender** (тип символьный) – ‘M’-мужской, ‘W’-женский
8. адрес проживания - **student\_address** (тип строковый)
9. примечание - **other\_details** (тип строковый)

***Дивизионы (Divisions)*** - таблица описывает дивизион, к которому принадлежит команда и имеет следующие атрибуты:

1. код дивизиона - **division\_code** (тип целочисленный) - первичный ключ
2. название дивизиона - **division\_name** (тип строковый)

***Команды (Teams)*** - таблица описывает команду, к которой может принадлежать студент-спортсмен и имеет следующие атрибуты:

1. идентификатор команды - **team\_id** (тип целочисленный) - первичный ключ
2. пол - **gender** (тип символьный) – ‘M’-мужской, ‘W’-женский
3. код дивизиона - **division\_code** (тип целочисленный) - внешний ключ из “дивизионы (division)”
4. код вида спорта - **sport\_code** (тип целочисленный) - внешний ключ из “виды спорта (sports)”
5. дата основания команды - **team\_date** (тип дата)
6. название команды - **team\_name** (тип строковый)
7. описание команды - **team\_description** (тип целочисленный)
8. имя тренера - **coach\_name** (тип строковый)
9. примечание - **other\_details** (тип строковый)

***Студенты в команде (Students\_in\_teams)*** – таблица описывает связь конкретных студентов с конкретной командой и имеет следующие атрибуты:

1. идентификатор студента - **student\_id** (тип целочисленный) - внешний ключ из “студенты-атлеты (Students\_athletes)”
2. идентификатор команды - **team\_id** (тип целочисленный) - внешний ключ из “команды (teams)”- может быть пустым

По условию студент может не быть участником команды, в таком случае мы будем называть его индивидуальным спортсменом и тогда поле *идентификатор команды -* ***team\_id*** может являться пустым, но в данную таблицу такие спортсмены должны заноситься

***Спортивные соревнования (Events)*** – таблица описывает спортивные соревнования и имеет следующие атрибуты:

1. идентификатор спортивного соревнования - **event\_id** (тип целочисленный) - первичный ключ
2. код вида спорта - **sport\_code** (тип целочисленный) - внешний ключ из “виды спорта (Sports)”
3. название события - **event\_name** (тип строковый)
4. локация события - **event\_location** (тип строковый)
5. дата начало спортивного события - **start\_date** (тип дата)
6. дата конца спортивного события - **end\_date** (тип дата)
7. примечание - **other\_details** (тип строковый)

***Студенты в спортивных соревнованиях (Students\_in\_Events***) - таблица описывает связь конкретных студентов из конкретной команды (если студент состоит в ней) с конкретными спортивными соревнования и имеет следующие атрибуты:

1. идентификатор студента - **student\_id** (тип целочисленный) - внешний ключ из “Студенты в команде (Students\_in\_teams)”
2. идентификатор команды - **team\_id** (тип целочисленный) - внешний ключ из “ Студенты в команде (Students\_in\_teams)”- может быть нулевым
3. идентификатор спортивного события **- event\_id** (тип целочисленный) - внешний ключ из “Спортивные соревнования (Events)”
4. результат студента - **student\_result** (тип целочисленный) – место, которое занял студент в спортивном событии
5. очки студента - **student\_points\_awarded** (тип целочисленный) – формируются в зависимости от места, которое занял студент в соревновании (1 место = 10 балов, 2 место = 7 балов, 3 место = 4 бала, 4 место и т.д = 1 бал)

***Студенческие грамоты (Student\_Letters)*** – описывает конкретные грамоты по конкретному виду спорта, которую получил конкретный ученик. *Грамота выдается, если спортсмен набрал за 5 лет 200 балов* из расчета, что в году 10 спортивных соревнований по виду спорта и он занимал как минимум 3-е место каждый раз.

Имеет следующие свойства:

1. идентификатор студента - **student\_id** (тип целочисленный) - внешний ключ из “Студент-атлет (Students\_athletes)”
2. код вида спорта - **sport\_code** (тип целочисленный) - внешний ключ из “вид спорта (Sports)”
3. дата награждения - **date\_awarded** (тип дата)

# Ограничения возможностей схемы данных из задания

1. В таблице ***Студенты-атлеты (Student\_athletes)***­­­­­­­­­­­­­ хранится сумма балов в очки на сегодняшний день - **points\_awarded\_to\_date** по всем видам спорта, т.е. неизвестно, за какие виды спорта он получил балы
2. Под видами спорта подразумевается разделения на конкретные дисциплины некоего вида спорта, т.е., например, бег на 100 метров и бег на 600 метров в нашей базе являются разными видами спорта

# Типы связей

# **Students\_in\_Teams** и **Student\_Athletes** соединены идентифицирующей связью один ко многим по атрибуту **student\_id**

# **Students\_in\_Teams** и **Teams** соединены неидентифицирующей связью (т.к. студент может выступать без команды) один ко многим по атрибуту **team\_id**

# Т.о**. Student\_Athletes** и **Teams** соединены связью многие ко многим

# **Students\_in\_Events** и **Students\_in\_Teams** соединены идентифицирующей связью один ко многим по атрибутам **student\_id** и **team\_id**

# **Students\_in\_Events** и **Events** соединены идентифицирующей связью один к одному по атрибуту **event\_id**

# Т.о. **Students\_in\_Events** и **Events** соединены связью многие ко многим

# **Events** и **Sports** соединены идентифицирующей связью один ко многим по атрибуту **sport\_code**

# **Teams** и **Divisions** соединены идентифицирующей связью один ко многим по атрибуту **division\_code**

# **Teams** и **Sports** соединены идентифицирующей связью один ко многим по атрибуту **sport\_code**

# **Student\_Letters** и **Student\_Athletes** соединены идентифицирующей связью один ко многим по атрибуту **student\_id**

# **Student\_Letters** и **Sports** соединены идентифицирующей связью один ко многим по атрибуту **sport\_code**

# Изменения

Нами были внесены следующие изменения:

1. Абстрактный атрибут **other\_details** из ***Студенты-атлеты (Student\_athletes)***­­­­­­­­­­­­­ заменен на логический атрибут профессионал ***- professional*** – является ли студент профессиональным спортсменом(**true**) или нет(**false**)
2. Абстрактный атрибут **other\_details** из ***Команды (Teams)*** заменен на целочисленный атрибут количество атлетов ***- number\_of\_athletes,*** заполнятся при помощи триггера. По умолчанию равен 0
3. Абстрактный атрибут **other\_details** из ***Спортивные соревнования (Events)*** заменен на логический атрибут ***межшкольное*** - ***interschool*** – является ли соревнование между школами(**true**) или внутри одной школы(**false**)
4. Идентификатор команды - **team\_id** в **Students\_in\_Teams** и **Students\_in\_Events** могут быть пустыми
5. По умолчанию атрибут очки на сегодняшний день – **points\_awarded\_to\_date** из таблицы ***Студенты-атлеты (Student\_Athletes)***­­­­­­­­­­­­­ равен 0
6. Создана новая таблица **Тренера (*Coaches*)** со следующими атрибутами:
   1. идентификатор тренера - **coach\_id** (тип целочисленный) - первичный ключ
   2. имя - **first\_name** (тип строковый)
   3. второе личное имя – **middle\_name** (тип строковый)
   4. фамилия - **last\_name** (тип строковый)
   5. должность тренера - **coach\_position** (тип строковый) – должность, занимаемая тренером (например, главный тренер, тренер по физической подготовке и т.д.)
   6. дата рождения - **date\_of\_birth** (тип дата)
   7. пол - **gender** (тип символьный) – ‘M’-мужской, ‘W’-женский
   8. адрес проживания - **student\_address** (тип строковый)

Тренер может тренировать только одну команду, у команды может быть больше одного тренера

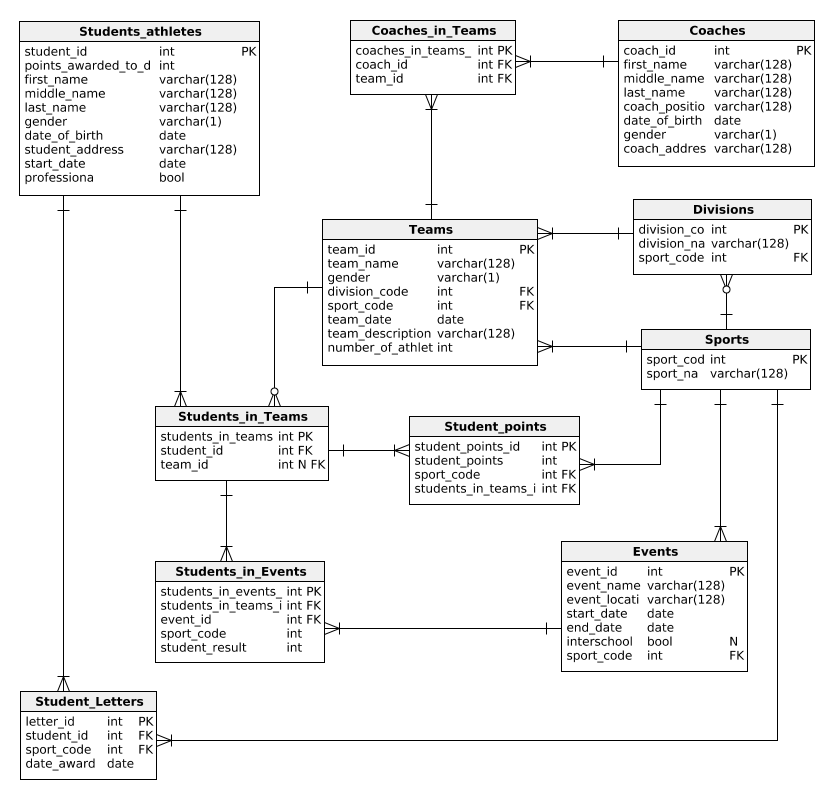
1. Создана новая таблица **Тренера *в команде (Coaches\_in\_teams)*** – таблица описывает связь конкретных тренеров с конкретной командой и имеет следующие атрибуты:
   1. идентификатор тренера в команде - **coaches\_in\_teams\_id** (тип целочисленный) – первичный ключ, идентификатор связи конкретного тренера с конкретной командой
   2. идентификатор тренера - **coach\_id** (тип целочисленный) - внешний ключ из “тренера (*coaches*)”
   3. идентификатор команды - **team\_id** (тип целочисленный) - внешний ключ из “команды (teams)”
2. В таблице ***Команды (Teams)*** удален следующий атрибут:
   1. имя тренера - **coach\_name** (тип строковый)
3. Созданы 3 новые связи:
   1. **Coaches\_in\_Teams** и **Teams** соединены идентифицирующей связью один ко многим по атрибуту team\_id
   2. **Coaches\_in\_Teams** и ***Coaches*** соединены идентифицирующей связью один ко многим по атрибуту coach\_id
   3. Т.о**. *Coaches*** и **Teams** соединены связью многие ко многим
4. В таблицу ***Студенты-атлеты (Student\_athletes)*** добавлен следующий атрибут:
   1. Дата поступления в школу - **start\_date** (тип дата)
5. В таблицу ***Студенты в команде (Students\_in\_teams)*** добавлен следующий атрибут:
   1. идентификатор студента в команде - **students\_in\_teams\_id** (тип целочисленный) – первичный ключ, идентификатор связи конкретного студента с конкретной командой
6. В таблицу ***Студенты в спортивных соревнованиях (Students\_in\_events***) добавлен следующие атрибуты:
   1. идентификатор студента в спортивном событии - **students\_in\_events\_id** (тип целочисленный) – первичный ключ, идентификатор связи конкретного студента из конкретной команды с конкретным спортивным событием
7. В таблице ***Студенты в спортивных соревнованиях (Students\_in\_events***) удалены следующие атрибуты:
   1. идентификатор студента - **student\_id** (тип целочисленный) - внешний ключ из “Студенты в команде (Students\_in\_teams)”
   2. идентификатор команды - **team\_id** (тип целочисленный) - внешний ключ из “ Студенты в команде (Students\_in\_teams)”
   3. очки студента - **student\_points\_awarded** (тип целочисленный) – формируются в зависимости от места, которое занял студент в соревновании (1 место = 10 балов, 2 место = 7 балов, 3 место = 4 бала, 4 место и т.д. = 1 бал)
8. Создана новая таблица **Очки студента** **(*Student\_points*),** в которой суммируются все заработанные очки конкретного студента в команде, которые он получил, участвуя в спортивном событии, в зависимости от места, которое он занял по следующему правилу:
   1. 1 место = 10 балов
   2. 2 место = 7 балов
   3. 3 место = 4 бала
   4. 4 место и т.д = 1 бал

Запись в таблицу идет при помощи триггера. Имеет следующие атрибуты:

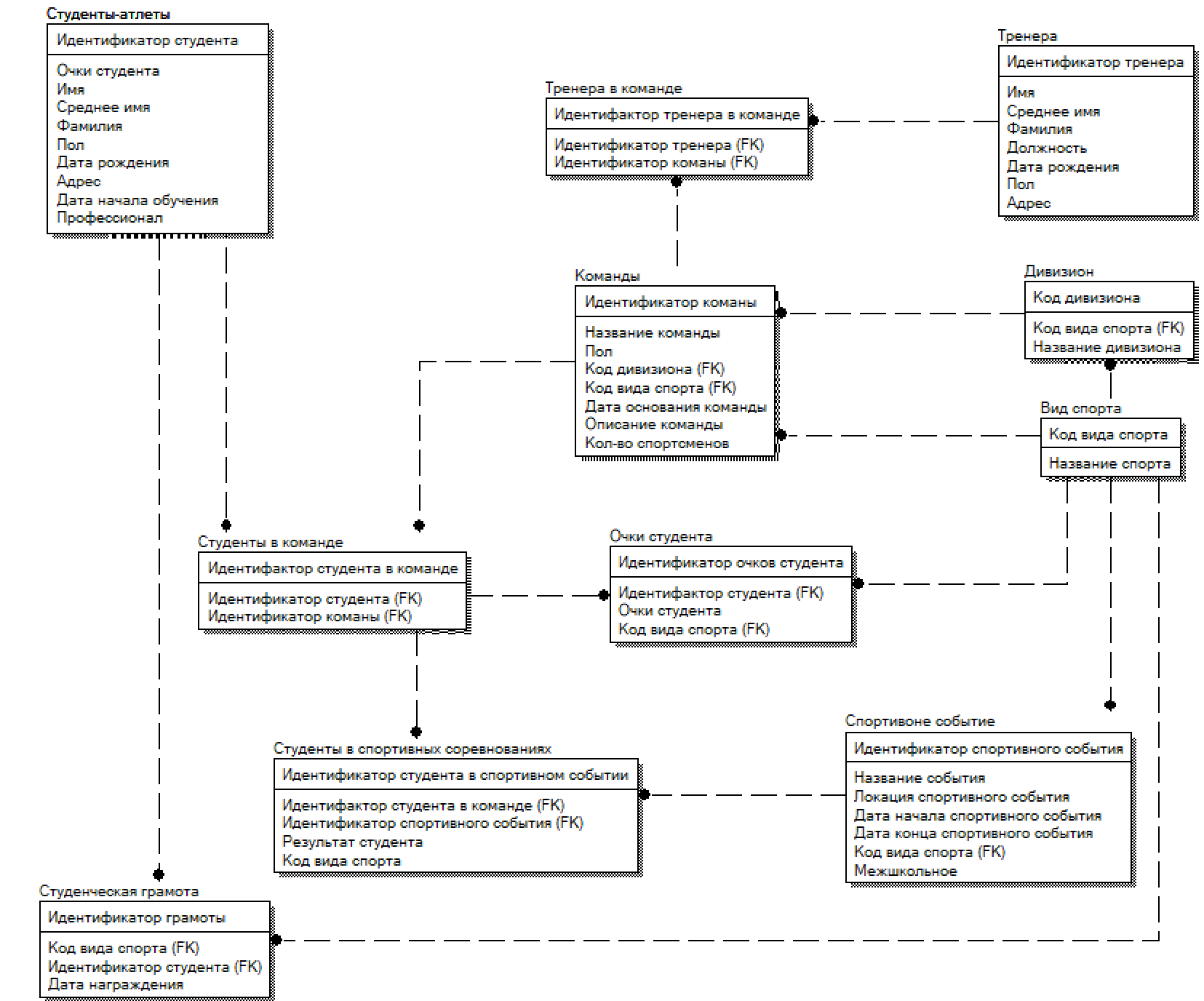
* 1. Идентификатор очков студента - **student\_points\_id** (тип целочисленный) – первичный ключ, идентификатор суммы всех очков конкретного студента в команде с конкретным видом спорта
  2. очки студента - **student\_points** (тип целочисленный)
  3. код вида спорта - **sport\_code** (тип целочисленный) - внешний ключ из “вид спорта (Sports)”
  4. идентификатор студента в команде - **students\_in\_teams\_id** (тип целочисленный) – первичный ключ, идентификатор связи конкретного студента с конкретной командой

1. В таблице ***Студенческие грамоты (Student\_Letters)*** добавлен следующий атрибут:
   1. идентификатор грамоты – **letter\_id** – первичный ключ

Физическая схема преобразованной базы данных



Концептуальная схема преобразованной базы данных



Преобразованная база в виде SQL скрипта

-- tables

-- Table: Coaches

CREATE TABLE Coaches (

    coach\_id int NOT NULL,

    first\_name varchar(128) NOT NULL,

    middle\_name varchar(128) NOT NULL,

    last\_name varchar(128) NOT NULL,

    coach\_position varchar(128) NOT NULL,

    date\_of\_birth date NOT NULL,

    gender varchar(1) NOT NULL,

    coach\_address varchar(128) NOT NULL,

    CONSTRAINT Coaches\_pk PRIMARY KEY (coach\_id)

);

-- Table: Coaches\_in\_Teams

CREATE TABLE Coaches\_in\_Teams (

    coaches\_in\_teams\_id int NOT NULL,

    coach\_id int NOT NULL,

    team\_id int NOT NULL,

    CONSTRAINT Coaches\_in\_Teams\_pk PRIMARY KEY (coaches\_in\_teams\_id)

);

-- Table: Divisions

CREATE TABLE Divisions (

    division\_code int NOT NULL,

    division\_name varchar(128) NOT NULL,

    sport\_code int NOT NULL,

    CONSTRAINT Divisions\_pk PRIMARY KEY (division\_code)

);

-- Table: Events

CREATE TABLE Events (

    event\_id int NOT NULL,

    event\_name varchar(128) NOT NULL,

    event\_location varchar(128) NOT NULL,

    start\_date date NOT NULL,

    end\_date date NOT NULL,

    sport\_code int NOT NULL,

interschool boolean NOT NULL,

    CONSTRAINT Events\_pk PRIMARY KEY (event\_id)

);

-- Table: Sports

CREATE TABLE Sports (

    sport\_code int NOT NULL,

    sport\_name varchar(128) NOT NULL,

    CONSTRAINT Sports\_pk PRIMARY KEY (sport\_code)

);

-- Table: Student\_Letters

CREATE TABLE Student\_Letters (

    letter\_id int NOT NULL,

    student\_id int NOT NULL,

    sport\_code int NOT NULL,

    date\_awarded date NOT NULL,

    CONSTRAINT Student\_Letters\_pk PRIMARY KEY (letter\_id)

);

-- Table: Students\_athletes

CREATE TABLE Students\_athletes (

    student\_id int NOT NULL,

    points\_awarded\_to\_date int NOT NULL DEFAULT 0,

    first\_name varchar(128) NOT NULL,

    middle\_name varchar(128) NOT NULL,

    last\_name varchar(128) NOT NULL,

    gender varchar(1) NOT NULL,

    date\_of\_birth date NOT NULL,

    student\_address varchar(128) NOT NULL,

    start\_date date NOT NULL,

professional boolean NOT NULL,

    CONSTRAINT Students\_athletes\_pk PRIMARY KEY (student\_id)

);

-- Table: Student\_points

CREATE TABLE Student\_points (

    student\_points\_id int NOT NULL,

    student\_points int NOT NULL DEFAULT 0,

    sport\_code int NOT NULL,

    students\_in\_teams\_id int NOT NULL,

    CONSTRAINT Student\_points\_pk PRIMARY KEY (student\_points\_id)

);

-- Table: Students\_in\_Events

CREATE TABLE Students\_in\_Events (

    students\_in\_events\_id int NOT NULL,

    students\_in\_teams\_id int NOT NULL,

    event\_id int NOT NULL,

    sport\_code int NOT NULL,

    student\_result int NOT NULL,

    CONSTRAINT Students\_in\_Events\_pk PRIMARY KEY (students\_in\_events\_id)

);

-- Table: Students\_in\_Teams

CREATE TABLE Students\_in\_Teams (

    students\_in\_teams\_id int NOT NULL,

    student\_id int NOT NULL,

    team\_id int NULL,

    CONSTRAINT Students\_in\_Teams\_pk PRIMARY KEY (students\_in\_teams\_id)

);

-- Table: Teams

CREATE TABLE Teams (

    team\_id int NOT NULL,

    team\_name varchar(128) NOT NULL,

    gender varchar(1) NULL,

    division\_code int NOT NULL,

    sport\_code int NOT NULL,

    team\_date date NULL,

    team\_description varchar(128) NULL,

    number\_of\_athletes int NOT NULL DEFAULT 0,

    CONSTRAINT Teams\_pk PRIMARY KEY (team\_id)

);

-- foreign keys

-- Reference: Coaches\_in\_Teams\_Coaches (table: Coaches\_in\_Teams)

ALTER TABLE Coaches\_in\_Teams ADD CONSTRAINT Coaches\_in\_Teams\_Coaches

FOREIGN KEY Coaches\_in\_Teams\_Coaches (coach\_id)

    REFERENCES Coaches (coach\_id);

-- Reference: Coaches\_in\_Teams\_Teams (table: Coaches\_in\_Teams)

ALTER TABLE Coaches\_in\_Teams ADD CONSTRAINT Coaches\_in\_Teams\_Teams

FOREIGN KEY Coaches\_in\_Teams\_Teams (team\_id)

    REFERENCES Teams (team\_id);

-- Reference: Divisions\_Sports (table: Divisions)

ALTER TABLE Divisions ADD CONSTRAINT Divisions\_Sports

FOREIGN KEY Divisions\_Sports (sport\_code) REFERENCES Sports (sport\_code);

-- Reference: Events\_Sports (table: Events)

ALTER TABLE Events ADD CONSTRAINT Events\_Sports FOREIGN KEY Events\_Sports (sport\_code)

    REFERENCES Sports (sport\_code);

-- Reference: Student\_Letters\_Sports (table: Student\_Letters)

ALTER TABLE Student\_Letters ADD CONSTRAINT Student\_Letters\_Sports

FOREIGN KEY Student\_Letters\_Sports (sport\_code) REFERENCES Sports (sport\_code);

-- Reference: Student\_Letters\_Students\_athletes (table: Student\_Letters)

ALTER TABLE Student\_Letters ADD CONSTRAINT Student\_Letters\_Students\_athletes

FOREIGN KEY Student\_Letters\_Students\_athletes (student\_id)

    REFERENCES Students\_athletes (student\_id);

-- Reference: Student\_points\_Sports (table: Student\_points)

ALTER TABLE Student\_points ADD CONSTRAINT Student\_points\_Sports

FOREIGN KEY Student\_points\_Sports (sport\_code) REFERENCES Sports (sport\_code);

-- Reference: Student\_points\_Students\_in\_Teams (table: Student\_points)

ALTER TABLE Student\_points ADD CONSTRAINT Student\_points\_Students\_in\_Teams

FOREIGN KEY Student\_points\_Students\_in\_Teams (students\_in\_teams\_id)

    REFERENCES Students\_in\_Teams (students\_in\_teams\_id);

-- Reference: Students\_in\_Events\_Events (table: Students\_in\_Events)

ALTER TABLE Students\_in\_Events ADD CONSTRAINT Students\_in\_Events\_Events

FOREIGN KEY Students\_in\_Events\_Events (event\_id) REFERENCES Events (event\_id);

-- Reference: Students\_in\_Events\_Students\_in\_Teams (table: Students\_in\_Events)

ALTER TABLE Students\_in\_Events ADD CONSTRAINT Students\_in\_Events\_Students\_in\_Teams

FOREIGN KEY Students\_in\_Events\_Students\_in\_Teams (students\_in\_teams\_id)

    REFERENCES Students\_in\_Teams (students\_in\_teams\_id);

-- Reference: Students\_in\_Teams\_Students\_athletes (table: Students\_in\_Teams)

ALTER TABLE Students\_in\_Teams ADD CONSTRAINT Students\_in\_Teams\_Students\_athletes

FOREIGN KEY Students\_in\_Teams\_Students\_athletes (student\_id)

    REFERENCES Students\_athletes (student\_id);

-- Reference: Students\_in\_Teams\_Teams (table: Students\_in\_Teams)

ALTER TABLE Students\_in\_Teams ADD CONSTRAINT Students\_in\_Teams\_Teams

FOREIGN KEY Students\_in\_Teams\_Teams (team\_id) REFERENCES Teams (team\_id);

-- Reference: Teams\_Divisions (table: Teams)

ALTER TABLE Teams ADD CONSTRAINT Teams\_Divisions

FOREIGN KEY Teams\_Divisions (division\_code)REFERENCES Divisions (division\_code);

-- Reference: Teams\_Sports (table: Teams)

ALTER TABLE Teams ADD CONSTRAINT Teams\_Sports

FOREIGN KEY Teams\_Sports (sport\_code) REFERENCES Sports (sport\_code);

# Триггеры

1. Запись данных таблицу **Очки студента** **(*Student\_points*)** идет при помощи ***триггера* “*init\_Student\_points”***, который срабатывает после внесения данных в таблицу ***Студенты в спортивных соревнованиях (Students\_in\_events*).** Также этот триггер добавляет эти балы к очкам на сегодняшний день - **points\_awarded\_to\_date** в ***Студенты в команде* (Students\_in\_Teams)**независимо от вида спорта.

***Нужен для формирования данных в таблице Очки студента (Student\_points).*** С помощью данной таблицы делать записи при помощи события ‘**init\_Student\_Letters’** проще, так как все балы студента в конкретном виде спорта уже просуммированы при помощи триггера, достаточно просто выбрать запись, соответствующую критериям

# Код триггера

DELIMITER //

CREATE TRIGGER init\_Student\_points

AFTER INSERT ON `Students\_in\_Events`

FOR EACH ROW

BEGIN

    -- переменная для хранения вводимого индекса в Student\_points

    DECLARE id INTEGER;

    --проверяем, существует ли комбинация

    --конкретного студента в команде с конкретным видом спорта

    --если существует, то в переменную id записывается

    --идентификатор суммы всех очков

    --конкретного студента в команде с конкретным видом спорта

    SET id := (SELECT student\_points\_id FROM Student\_points

        WHERE sport\_code=NEW.sport\_code AND students\_in\_teams\_id=NEW.students\_in\_teams\_id);

    --если нет, то переменная принимает максимальное значения

    --индекса из `Student\_points`+1,

    --т.е. будет создаваться новая запись в в Student\_points

    if id IS NULL THEN

       SET id := (SELECT count(\*) FROM Student\_points) + 1;

    END IF;

    --если занял 1-ое место

    IF NEW.student\_result=1 THEN

    --если не существует комбинации такого вида спорта и студента-спортсмена,

    --то вносим новые данные

        INSERT INTO `student\_points`

        SET student\_points\_id = id,

        student\_points=10,

        students\_in\_teams\_id=NEW.students\_in\_teams\_id,

        sport\_code=NEW.sport\_code

        --если такой id найден, т.е. существует

        --комбинация такого вида спорта и студента-спортсмена,

        --то к student\_point прибавляем 10

            ON DUPLICATE KEY UPDATE

            student\_points=student\_points+10;

        --прибавляем эти балы

        --к очкам на сегодняшний день - points\_awarded\_to\_date

        --в Студенты в команде независимо от вида спорта

        UPDATE Students\_athletes, Students\_in\_Teams

            SET Students\_athletes.points\_awarded\_to\_date =

Students\_athletes.points\_awarded\_to\_date + 10

            WHERE Students\_athletes.student\_id=Students\_in\_Teams.student\_id  and

            Students\_in\_Teams.students\_in\_teams\_id = new.students\_in\_teams\_id;

    END IF;

    --остальные условия имеют аналогичный принцип действия

    --если занял 2-ое место

    IF NEW.student\_result=2 THEN

       INSERT INTO `student\_points`

        SET student\_points\_id = id,

         student\_points=7,

         students\_in\_teams\_id=NEW.students\_in\_teams\_id,

         sport\_code=NEW.sport\_code

            ON DUPLICATE KEY UPDATE

            student\_points=student\_points+7;

    END IF;

    --если занял 3-е место

    IF NEW.student\_result=3 THEN

        INSERT INTO `student\_points`

        SET student\_points\_id = id,

        student\_points=4,

        students\_in\_teams\_id=NEW.students\_in\_teams\_id,

        sport\_code=NEW.sport\_code

            ON DUPLICATE KEY UPDATE

            student\_points=student\_points+4;

    END IF;

    --если занял 4-ое место и выше

    IF NEW.student\_result>3 THEN

       INSERT INTO `Student\_points`

        SET student\_points\_id = id,

        student\_points=1,

        students\_in\_teams\_id=NEW.students\_in\_teams\_id,

        sport\_code=NEW.sport\_code

            ON DUPLICATE KEY UPDATE

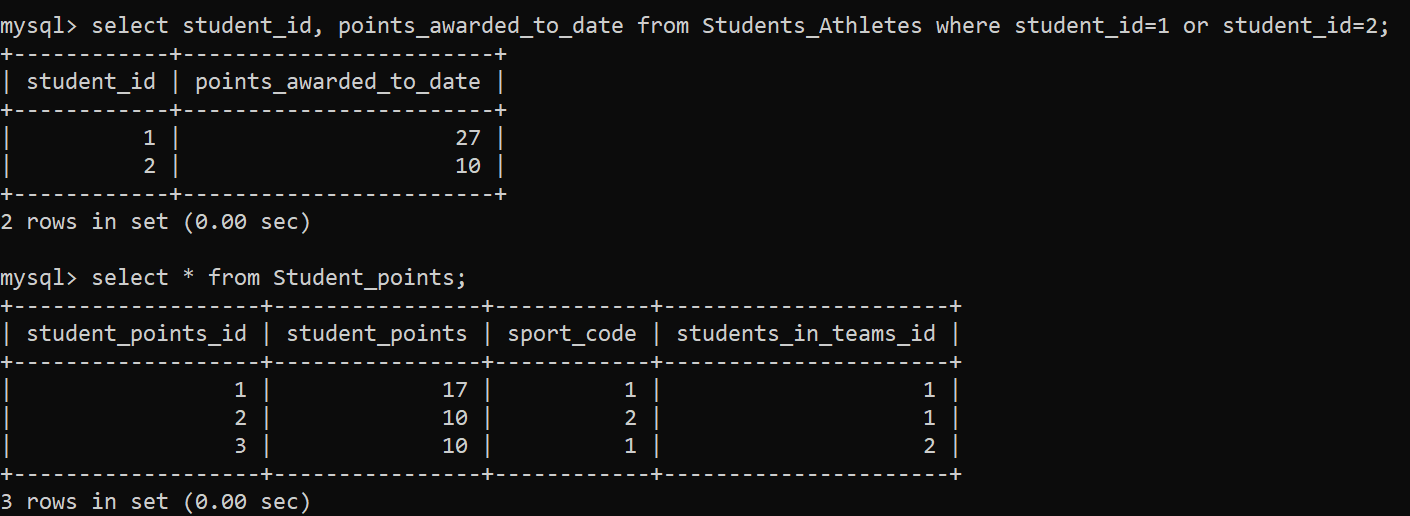
            student\_points=student\_points+1;

    END IF;

END //

# Демонстрация работы триггера

# 



1. Проверка даты и времени начала и конца спортивного события при помощи триггера ‘***event\_time\_error’***. Дата начала не должна быть позже даты конца.

***Нужен для проверки корректности ввода***

# Код триггера

DELIMITER //

CREATE TRIGGER event\_time\_error

BEFORE INSERT

ON Events FOR EACH ROW

BEGIN

IF (NEW.start\_date  > NEW.end\_date)

THEN

    SIGNAL sqlstate '45001'

    set message\_text = “Неверные даты начала и конца”;

END IF;

END//

DELIMITER;

# Демонстрация работы триггера

# 

1. Кол-во атлетов в команде записывает в атрибут ***количество атлетов (number\_of\_athletes)*** при помощи триггера ‘***init\_number\_of\_athletes’***, который после добавления записи в ***Студенты в спортивных соревнованиях (Students\_in\_events***) прибавляет к записи в Команды (Teams), которой соответствует введенный team\_id, 1.

***Нужен для подсчета кол-ва спортсменов в команде***

# Код триггера

DELIMITER //

CREATE TRIGGER init\_number\_of\_athletes

BEFORE INSERT ON Students\_in\_Teams FOR EACH ROW

BEGIN

UPDATE Teams

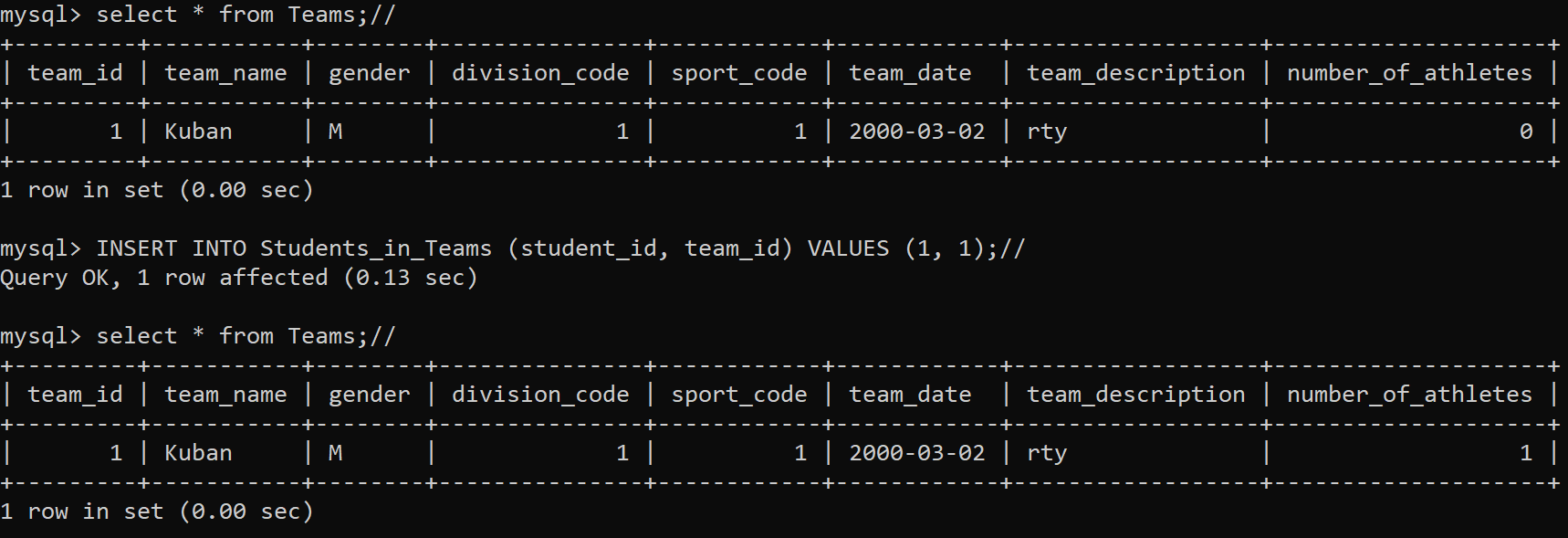
    SET number\_of\_athletes =  number\_of\_athletes + 1

    WHERE Teams.team\_id = new.team\_id;

END//

DELIMITER;

# Демонстрация работы триггера



# Событие

С помощью события происходит ***отслеживание того, когда надо вручать грамоты***, происходит при помощи события “**init\_Student\_Letters”** - оно представляет собой задачу, которая выполняется согласно плану. В нашем случае оно будет срабатывать в конце учебного года (25-ого мая) и будет заполнят таблицу ***Студенческие грамоты (Student\_Letters)*** если выполняются следующие условия:

* 1. Студент-спортсмен закончил школу, т.е. на момент окончания учебного года он поступил 5 лет назад (года учебные)
  2. Он набрал 200+ балов в конкретном виде спорта

Т.о. он может получить несколько грамот т.к.:

* 1. Он может участвовать в нескольких видах спорта
  2. Он может участвовать в спортивном событии как в составе команды (и их может быть больше одной), так и как индивидуальный спортсмен

***Используется для автоматизации процесса выдачи грамот***

# Код события

--Event: init\_Student\_Letters

DELIMITER //

CREATE EVENT init\_Student\_Letters

ON SCHEDULE EVERY 1 YEAR

STARTS '2019-05-25'

DO

    BEGIN

    INSERT INTO Student\_Letters (student\_id, sport\_code, date\_awarded)

        SELECT Students\_athletes.student\_id,

            Student\_points.sport\_code,

            DATE\_FORMAT(NOW(),'%Y-%m-%d')

        FROM Student\_points, Students\_in\_Teams, Students\_athletes

        WHERE

        --прошло ровно 5 лет

        YEAR(NOW())-YEAR(Students\_athletes.start\_date) = 5 AND

        MONTH(Students\_athletes.start\_date) - MONTH(NOW) = 5 AND

        DAY(Students\_athletes.start\_date) - DAY(NOW()) = 24 AND

        --очков больше 200

        Student\_points.student\_point >= 200 AND

        --берем конкретного студента из конкретной команды

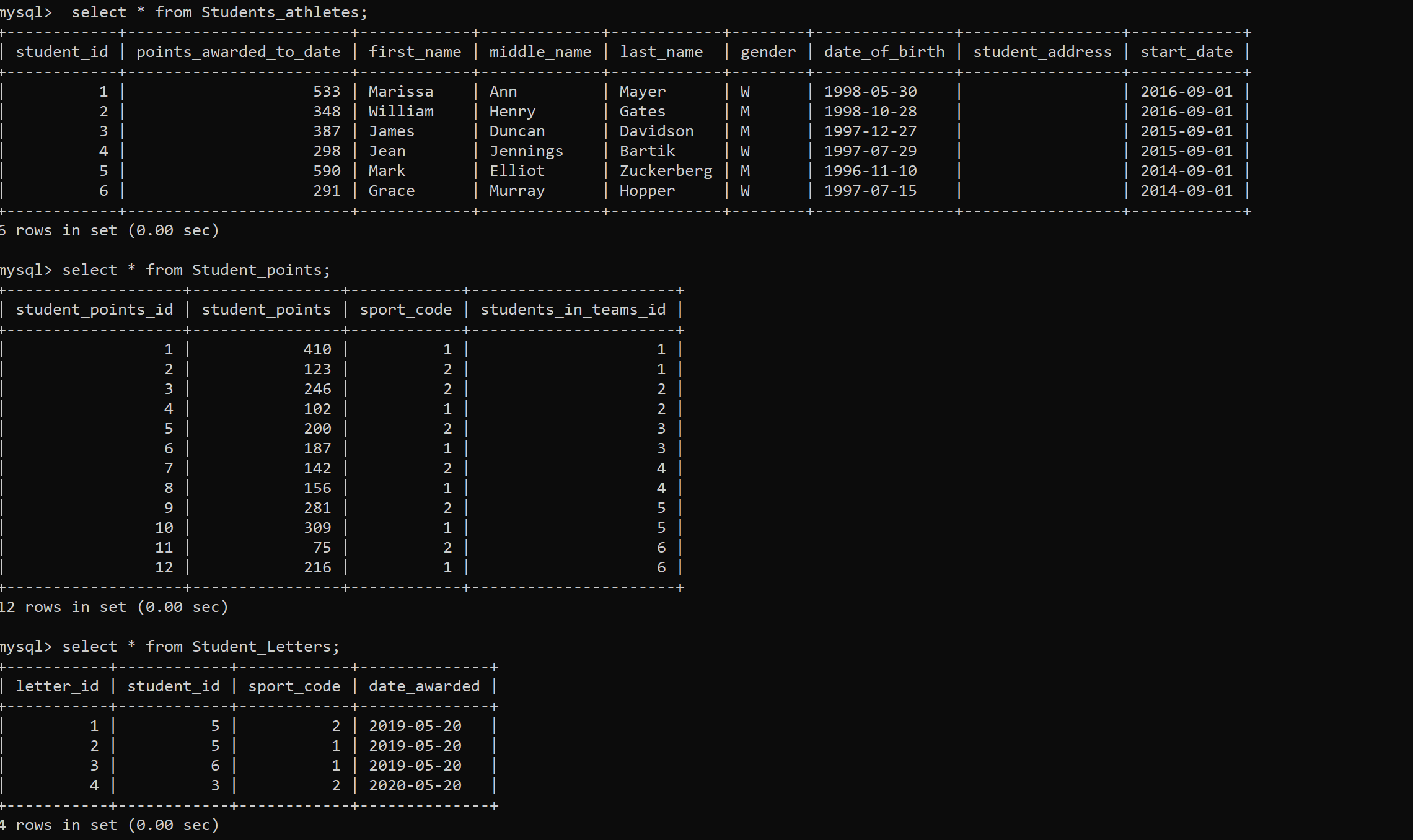
        Student\_points.students\_in\_teams\_id=Students\_in\_Teams.students\_in\_teams\_id AND

        Students\_in\_Teams.student\_id=Students\_athletes.student\_id;

    END

//

# Демонстрация работы события



В таблице **Student\_Letters** внесены студенты, которые поступили 5 и больше лет назад и по конкретному виду спорта набрали больше 200 балов (студенты с id 1 и 2 поступили 4года назад, так что их не считаем)

# Примеры запросов:

1. Вывести из **Student\_Letters** student\_id, кол-во его вхождений в **Student\_Letters ,** % дубликатов

SELECT student\_id,

    COUNT(\*) as 'Всего вхождений',

    1 - COUNT(DISTINCT student\_id)/COUNT(\*) as '% Дубликатов'

    FROM Student\_Letters

    WHERE NOT student\_id IN

    (

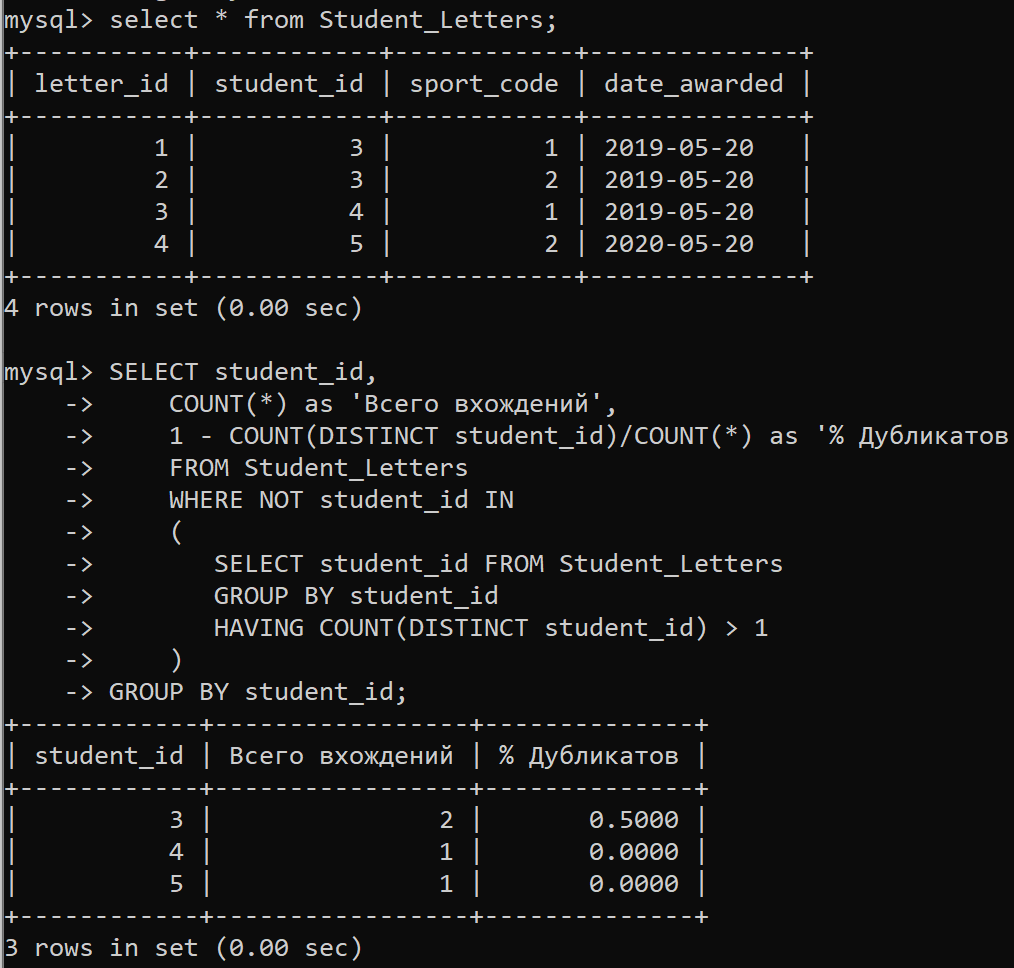
       SELECT student\_id FROM Student\_Letters

       GROUP BY student\_id

       HAVING COUNT(DISTINCT student\_id) > 1

    )

GROUP BY student\_id;



1. Вывести фамилию, имя, второе имя, название команды и название вида спорта всех студентов, которые находятся в 1-ой лиге бега на 100 метров

SELECT S.last\_name, S.first\_name, S.middle\_name, T.team\_name, SP.sport\_name

FROM Students\_athletes S

JOIN Students\_in\_Teams ST

    ON S.student\_id=ST.student\_id

JOIN Teams T

    ON  ST.team\_id=T.team\_id

JOIN Sports SP

    ON T.sport\_code=SP.sport\_code

WHERE  T.division\_code=(

    SELECT D.division\_code

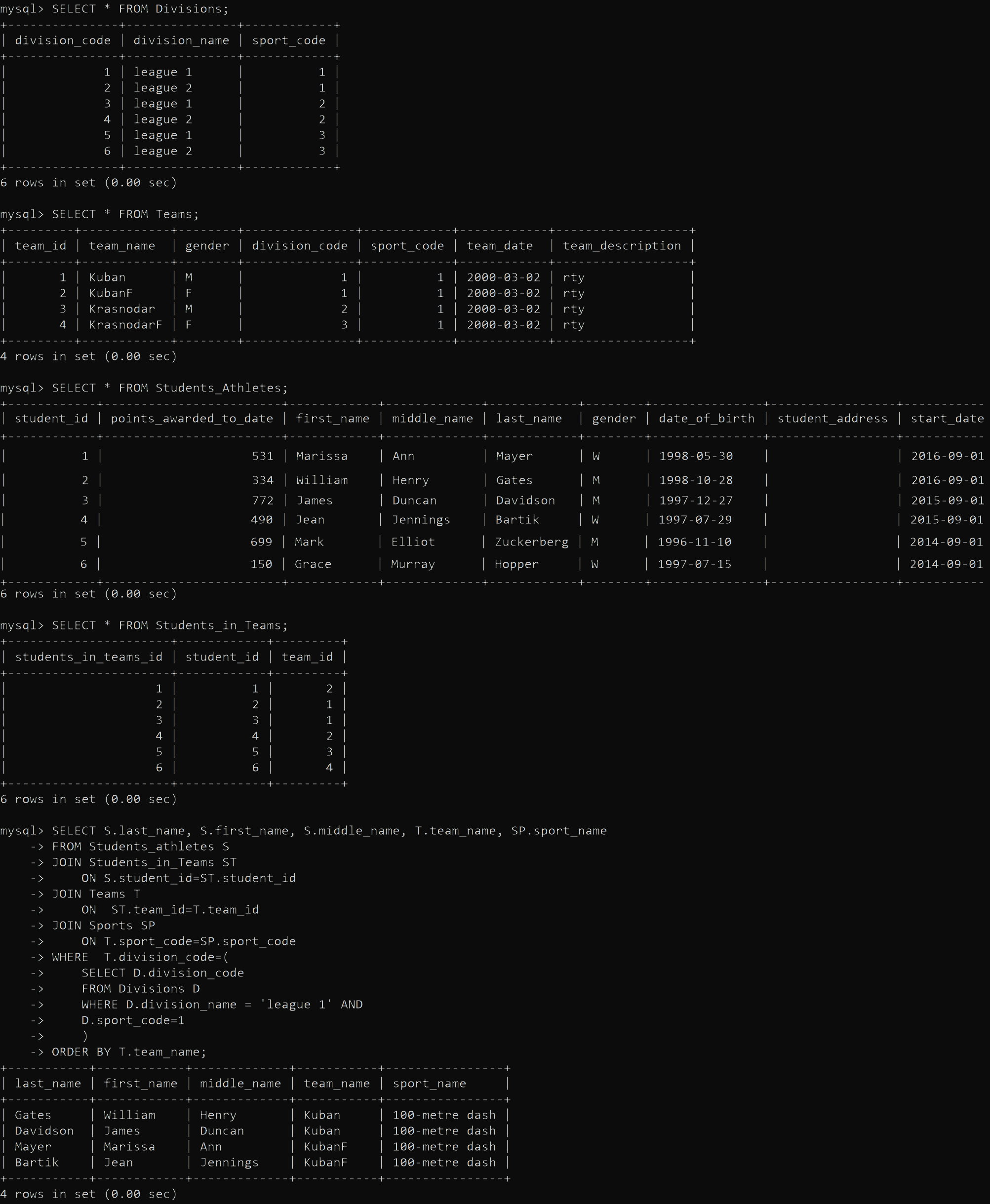
    FROM Divisions D

    WHERE D.division\_name = 'league 1' AND

    D.sport\_code=1

    )

ORDER BY T.team\_name;



1. Вывести идентификатор главного тренера, его фамилию, имя и общую сумму набранных их подопечными балов в порядке убывания по балам.(главный тренер - 'head\_coach')

SELECT CO.coach\_id, CO.lASt\_name, sum(.student\_points)

FROM

    teams AS t

    Inner JOIN students\_in\_teams AS ST

        ON ST.team\_id = t.team\_id

    Inner JOIN student\_points AS SP

        ON SP.students\_in\_teams\_id = ST.students\_in\_teams\_id

    Inner JOIN Coaches\_in\_Teams AS CT

        ON T.team\_id=CT.team\_id

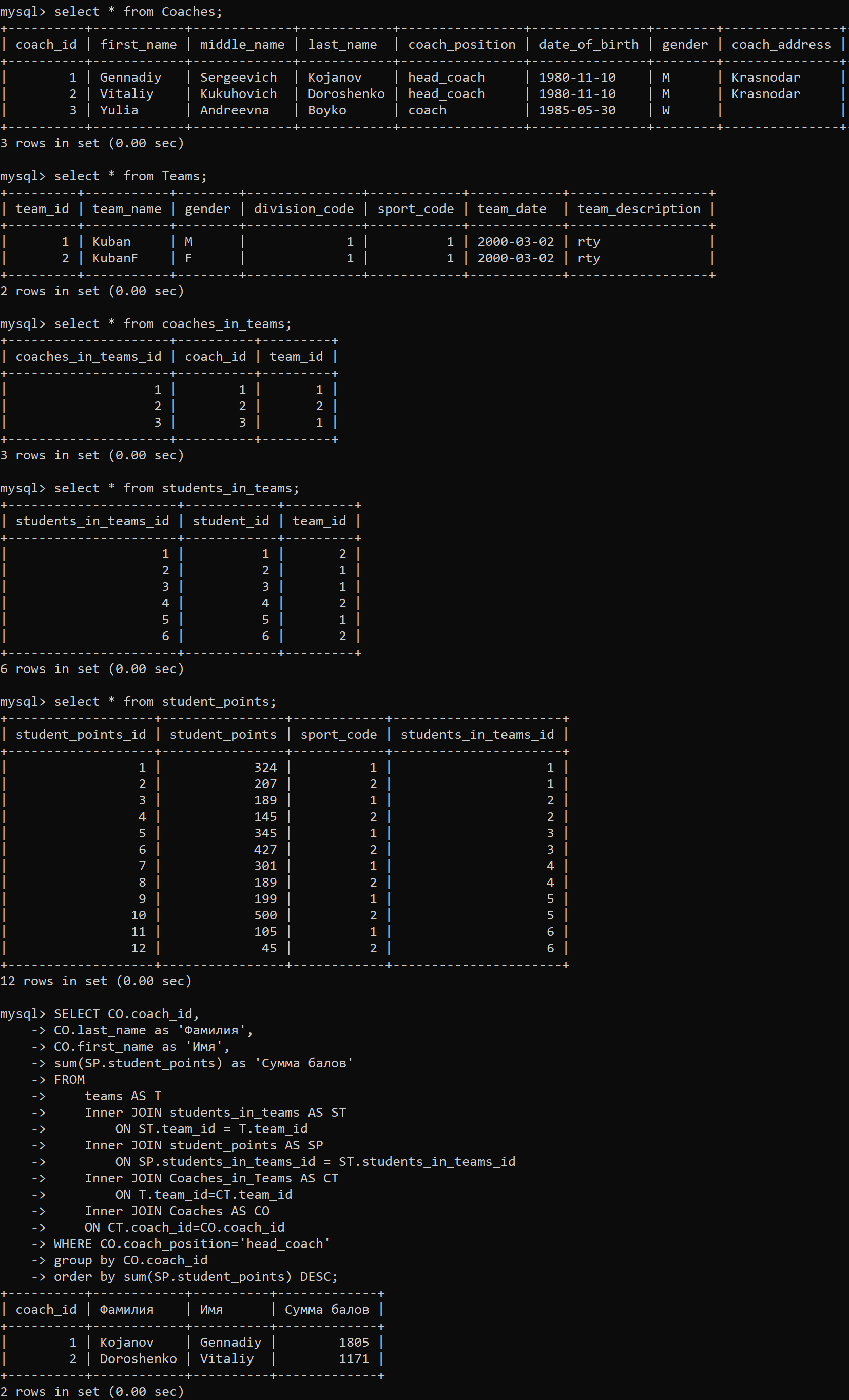
    Inner JOIN Coaches AS CO

     ON CT.coach\_id=CO.coach\_id

WHERE CO.coach\_position='head\_coach'

GROUP BY CO.coach\_id;

ORDER BY sum(SP.student\_points) DESC;



Тренер с идентификатором 3 не вывелся, так как он не главный тренер