Baza danych - Lekkoatletyka

Autorzy: Przemysław Rychter, Filip Przygoński

Politechnika Wrocławska
Wydział Elektroniki
Prowadzący: dr inż. Piotr Lechowicz

1 Wstęp

1.1 Opis systemu

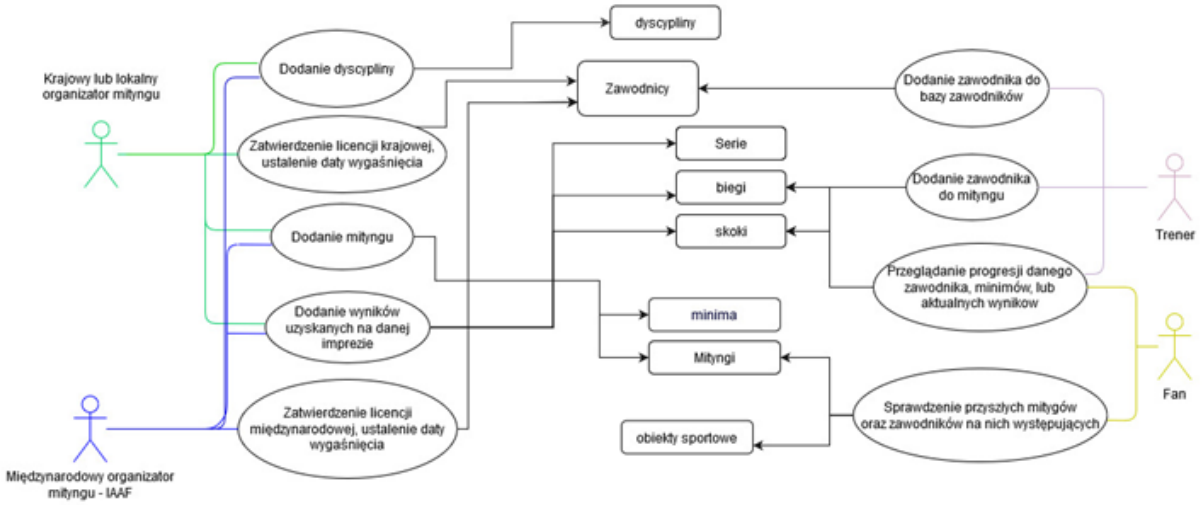
Celem projektu było zrealizowanie bazy danych wyników sportowych z dyscyplin lekkoatletycznych. System przechowuje informacje o zawodnikach oraz ich klubach wraz z ich atrybutami niezbędnymi z punktu widzenia użytkownika, wyniki uzyskane w konkretnych seriach podczas danych zawodów, informację o zgłoszonych zawodnikach na przyszłe zawody, rangę zawodów, minima uprawniające do startu. Rzeczywistość przenieśliśmy na dane, tworząc tabele odpowiadające obiektom oraz zdarzeniom, tak aby zgromadzić wszystkie niezbędne informacje z punktu widzenia różnych użytkowników bazy. Staraliśmy się również, zredukować nadmiarowość danych wykorzystując poznane mechanizmy relacyjnych baz danych. Nazwy tabel odzwierciedlają dane, które są w nich przechowywane, dlatego nie ma potrzeby omawiania każdej z nich, jednak kilka tabel może być początkowo nie zrozumiałych. Tabela "limits" służy określeniu minimów jakie muszą spełniać zawodnicy aby zapis (wpis rekordu do signups) mógł zostać wykonany. Tabela series przechowuje zdarzenia rozpoczynające się w konkretnym punkcie w czasie tzn. konkretny bieg części zawodników z całej stawki zgłoszonych na dane zawody w danej dyscyplinie. W naszej bazie stowrzyliśmy osobne tabele do przechowywania wyników skoków oraz biegów ponieważ rezultaty uzyskane przez zawodników z obu tych rodzinach konkurencji muszą być przechowywane w innym formacie.

1.2 Opis wymagań funkcjonalnych

- Umieszczenie nowego zawodnika w bazie.
- Wyszukiwania wyników, z różnymi kryteriami

- Możliwość dodania wydarzenia (mityngu lekkoatletycznego) wraz ze specyfikacją minimów określających minimalne wyniki w danej dyscyplinie uprawniające do startu.
- Zgłoszenie zawodnika do startu w danym mityngu, danej dyscyplinie.
- Uzupełnienie wyników zawodników po uprzednim umieszczeniu (zgłoszeniu), określenie serii w której startował.
- Modyfikacja wyników - dyskwalifikacja.

1.3 Diagram przypadków użycia

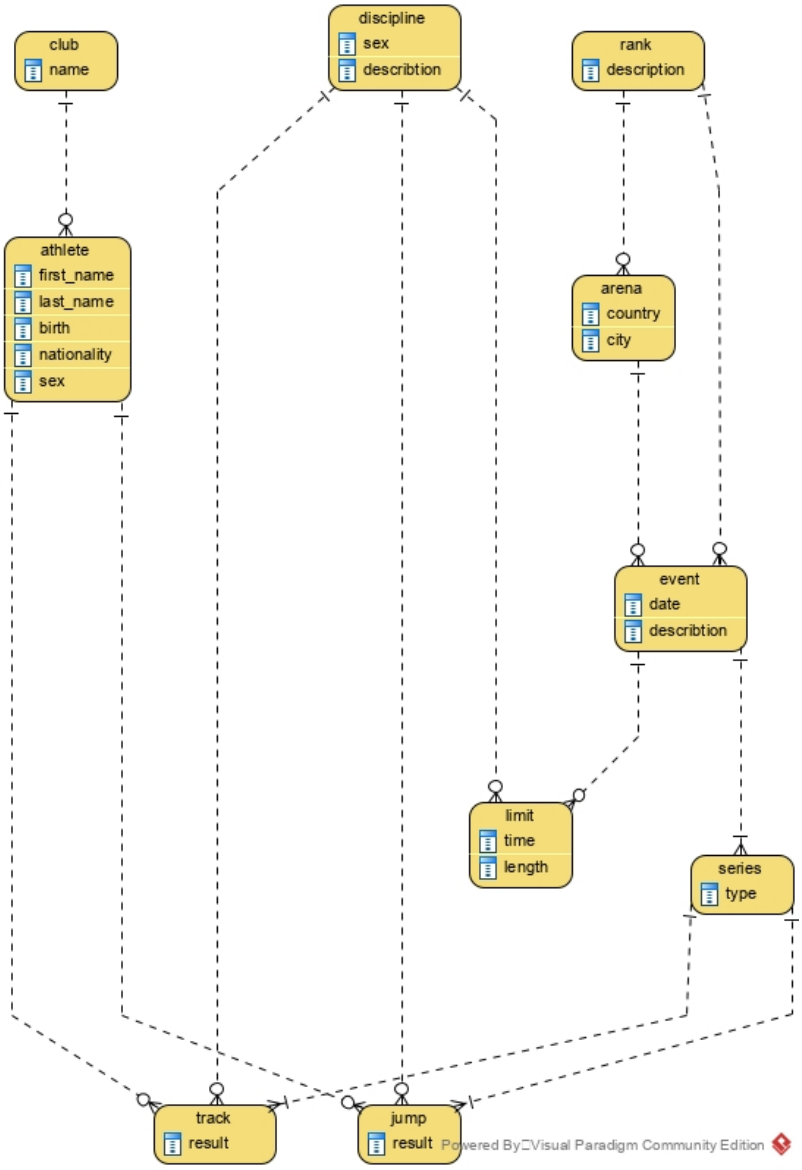


Rysunek 1: Diagram przypadków użycia

Diagram przedstawia akcje możliwe do wykonania przez różnych użytkowników bazy. Akcje te byłyby możliwe do wykonania po zdefiniowaniu specyficznych dostępów do bazy w zależności od typu użytkownika.

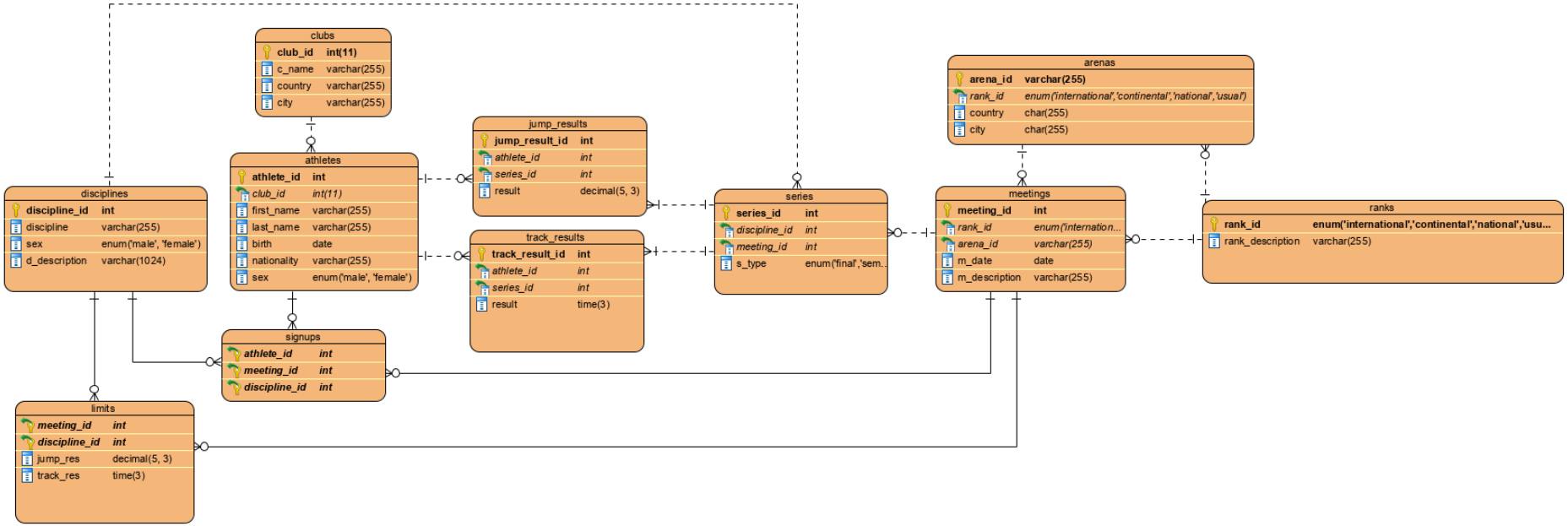
2 Model bazy danych

2.1 Model logiczny



Rysunek 2: Model logiczny

2.2 Model fizyczny



Rysunek 3: Model fizyczny

2.3 Opis ważniejszych transakcji

- Zgłoszenie zawodnika na zawody wymaga dodania rekordu do tabeli signup, powinno być ono wykonywane z użycie procedur "signup_track" lub "signup_jump", które sprawdzają czy zawodnik spełnia minimum uprawniające do startu w danej dyscyplinie na danych zawodach.
- Tworzenie serii startowych oprócz poporstu dodania rekordów, może byż wykonywane poprzez funkcję "create_series" która tworzy serie bazując na ilości zawodników w pojedyńczej serii, oraz pojemności dyscypliny, funkcja ta dodatkowo usuwa z tabeli signup rekordy nadmiarowe czyli zawodników którzy mimo spełnienia minimum, nie będą mogli wystartować w zawodach ponieważ ich obecność przekracza ilość dostępnych miejsc dla startujących w danej dyscyplinie.

3 Implementacja bazy danych

3.1 Środowisko programistyczne

Jako system zarządzający bazą danych wybraliśmy MySQL, a jako narzędzie wspomagające administrację bazy wybraliśmy MySQL Workbench. Użyliśmy również Visual Paradigm do wygenerowania kodu SQL tworzącego bazę danych na podstawie wcześniej stworzonego w tym programie diagramu.

3.2 Przykładowe skrypty tworzące bazę danych

W poniższych przykładach w miejscach gdzie użyto jako dyscypliny biegu, analogicznie będzie wyglądał kod dla skoków, różnić będzie się jedynie typem danych (bieg - Time, skok - Decimal), oraz że bieg jest tym lepszy im mniejszy czas, podczas gdy skok jest tym lepszy im większa odległość.

Listing 1: Stworzenie tabeli atletów

```
CREATE TABLE athletes (
    athlete_id  int NOT NULL AUTO_INCREMENT,
    club_id     int,
    first_name  varchar(255) NOT NULL,
    last_name   varchar(255) NOT NULL,
    birth       date NOT NULL,
    nationality varchar(255) NOT NULL,
    sex         enum('male','female') NOT NULL,
    PRIMARY KEY (athlete_id),
    FOREIGN KEY (club_id) REFERENCES clubs(club_id)
);
```

Listing 2: Stworzenie tabeli mityngów

```
CREATE TABLE meetings (
    meeting_id    int NOT NULL AUTO_INCREMENT,
    rank_id       enum('international','continental',
                  'national','usual') NOT NULL,
    arena_id      varchar(255)  NOT NULL,
    m_date        date NOT NULL,
    m_description varchar(255),
    PRIMARY KEY   (meeting_id),
    FOREIGN KEY   (rank_id) REFERENCES ranks(rank_id),
    FOREIGN KEY   (arena_id) REFERENCES arenas(arena_id)
);
```

Listing 3: Stworzenie tabeli wyników biegów

```
CREATE TABLE track_results (
    track_result_id int NOT NULL AUTO_INCREMENT,
    athlete_id      int NOT NULL,
    series_id       int NOT NULL,
    result          time(3),
    PRIMARY KEY     (track_result_id),
    FOREIGN KEY     (athlete_id) REFERENCES
                    athletes(athlete_id),
    FOREIGN KEY     (series_id) REFERENCES series(series_id)
);
```

Listing 4: Stworzenie tabeli zapisów na mityngi

```
CREATE TABLE signups (
    athlete_id      int NOT NULL,
    meeting_id      int NOT NULL,
    discipline_id   int NOT NULL,
    PRIMARY KEY     (athlete_id, meeting_id, discipline_id),
    FOREIGN KEY     (athlete_id) REFERENCES
                    athletes(athlete_id),
    FOREIGN KEY     (meeting_id) REFERENCES
                    meetings(meeting_id),
    FOREIGN KEY     (discipline_id) REFERENCES
                    disciplines(discipline_id) );
```

3.3 Widoki i funkcje bazy danych

Listing 5: Widok wyświetlający wszystkie zapisane wyniki biegów, dołączając do nich informacje o atlecie, meetingu oraz dyscyplinie

```
CREATE OR REPLACE VIEW ‘athlete_track_results‘ AS
    SELECT
        a.athlete_id AS "athlete_id",
        a.first_name AS "first_name",
        a.last_name AS "last_name",
        a.birth AS "birth_date",
        a.nationality AS "nationality",
        c.c_name AS "club",
        d.discipline AS "discipline",
        IF(d.sex = ’male’,’M’,’F’) AS "sex",
        t.result AS "result",
        m.m_date AS "meeting_date",
        ar.arena_id AS "stadium",
        ar.city AS "stadium_city",
        ar.country AS "stadium_country",
        r.rank_id AS "meeting_rank",
        m.m_description AS "meeting",
        s.s_type AS "series"
    FROM
        track_results t
        JOIN athletes a ON a.athlete_id = t.athlete_id
        LEFT JOIN clubs c ON c.club_id = a.club_id
        JOIN series s ON s.series_id = t.series_id
        JOIN disciplines d ON d.discipline_id=s.discipline_id
        JOIN meetings m ON m.meeting_id = s.meeting_id
        JOIN arenas ar ON ar.arena_id = m.arena_id
        JOIN ranks r ON m.rank_id = r.rank_id;
```

Listing 6: Widok wyświetlający najlepsze wyniki we wszystkich dyscyplinach biegowych

```
CREATE OR REPLACE VIEW ‘track_records‘ AS
    SELECT
        discipline,
        sex,
        MIN(result) AS "Record",
        first_name,
        last_name,
        nationality,
        club,
        meeting_date,
        stadium,
        stadium_city,
        stadium_country,
        meeting_rank,
        meeting,
        series
    FROM
        athlete_track_results
    GROUP BY
        discipline, sex;
```

Listing 7: Procedura tworząca limit dla danego meetingu i dyscypliny na podstawie wyników z ostatniego roku z meetingów o tej samej randze, oraz umieszczenie obliczonego limitu w tabeli ‘limits’

```
CREATE PROCEDURE lekkoatletyka.create_limit_track
(IN meeting_id_p INT, discipline_id_p INT)
BEGIN
    DECLARE meeting_date date;
    DECLARE event_rank VARCHAR(13);
    DECLARE average_result TIME(3);

    SELECT m_date
    INTO meeting_date
    FROM meetings m
    WHERE m.meeting_id=meeting_id_p;

    SELECT rank_id
    INTO event_rank
    FROM meetings m
    WHERE m.meeting_id=meeting_id_p;

    SELECT
        AVG(t.result)
    INTO
        average_result
    FROM
        track_results t
        JOIN athletes a ON a.athlete_id = t.athlete_id
        LEFT JOIN clubs c ON c.club_id = a.club_id
        JOIN series s ON s.series_id = t.series_id
        JOIN meetings m ON m.meeting_id = s.meeting_id
        WHERE m.m_date>DATE_ADD(meeting_date,INTERVAL -1 YEAR)
            AND m.rank_id = event_rank
            AND s.discipline_id = discipline_id_p;

    INSERT INTO limits (meeting_id, discipline_id, track_res)
    VALUES (meeting_id_p, discipline_id_p, average_result)
    ON DUPLICATE KEY UPDATE track_res=average_result;
END;
```

Listing 8: Procedura zapisująca danego atletę, na daną dyscyplinę na danych zawodach - sprawdza czy atleta spełnia limit zawodów

```
CREATE PROCEDURE lekkoatletyka.signup_track
(IN athlete_id_p INT, meeting_id_p INT, discipline_id_p INT)
BEGIN
    DECLARE meeting_limit TIME(3);
    DECLARE meeting_date date;
    DECLARE athlete_best_result TIME(3);
    DECLARE event_rank VARCHAR(13);

    SELECT track_res INTO meeting_limit
    FROM limits WHERE meeting_id=meeting_id_p
    AND discipline_id=discipline_id_p;

    SELECT m_date INTO meeting_date
    FROM meetings WHERE meeting_id=meeting_id_p;

    SELECT rank_id INTO event_rank
    FROM meetings WHERE meeting_id=meeting_id_p;

    SELECT MIN(t.result)
    INTO athlete_best_result
    FROM track_results t
        JOIN athletes a ON a.athlete_id = t.athlete_id
        LEFT JOIN clubs c ON c.club_id = a.club_id
        JOIN series s ON s.series_id = t.series_id
        JOIN meetings m ON m.meeting_id = s.meeting_id
    WHERE m.m_date > DATE_ADD(meeting_date, INTERVAL -1 YEAR)
        AND m.rank_id = event_rank
        AND s.discipline_id = discipline_id_p
        AND a.athlete_id = athlete_id_p;
    IF meeting_limit IS NOT NULL THEN
        IF athlete_best_result<meeting_limit THEN
            INSERT INTO signups
            VALUES (athlete_id_p, meeting_id_p,
            discipline_id_p);
            END IF;
    END IF;
    IF meeting_limit IS NULL THEN
        INSERT INTO signups VALUES
        (athlete_id_p, meeting_id_p, discipline_id_p);
    END IF;
END;
```

Listing 9: Funkcja tworząca serie, w tabeli są dodawane rekordy serii bazując na ilości zawodnikow z tabeli signup, pojemności dyscypliny na danych zawodach, pojemości pojedynczej serii, jeżeli w tabeli signup znajduje się więcej zawodników niż ilość przewidziany miejs, zawodnicy z najgorszymi wynikami zostaną usunięci.

```
CREATE FUNCTION `create_series`(
    typ_param ENUM('normal','knockout'),
    meeting_id_param int,
    discipline_id_param int,
    amount_param int,
    serie_size_param int,
    promotion int
) RETURNS varchar(255) CHARSET utf8mb4
    DETERMINISTIC
BEGIN
    DECLARE signed_up int DEFAULT 0;
    DECLARE athletes_to_remove int DEFAULT 0;
    DECLARE serie_amount int DEFAULT 0;
    DECLARE ret VARCHAR(255) DEFAULT '';
    DECLARE serie_id_p int;

    SELECT COUNT(*)
    INTO signed_up FROM signups
    WHERE discipline_id = discipline_id_param
    AND meeting_id = meeting_id_param;

    IF signed_up > amount_param   THEN
        SET athletes_to_remove = (signed_up - amount_param);

    DELETE FROM signups AS ss WHERE ss.athlete_id IN
    (SELECT onlyRES.result FROM
    (SELECT a.athlete_id ,
    IF(discipline_id=discipline_id_param,MIN(t.result),NULL)
    AS result
    FROM athletes a
    LEFT JOIN track_results t ON a.athlete_id = t.athlete_id
    LEFT JOIN series s        ON t.series_id = s.series_id
    GROUP BY a.athlete_id
    ORDER BY MIN(t.result) IS NULL ASC, MIN(t.result)
    LIMIT athletes_to_remove
    ) AS onlyRES
    )
    AND ss.discipline_id = disciplin_id_param
    AND ss.meeting_i = meeting_id_param;
    END IF;
```

```
    SELECT COUNT(*)INTO signed_up
    FROM signups
    WHERE discipline_id = discipline_id_param
    AND meeting_id = meeting_id_param;

    IF typ_param = 'normal' THEN
    SET ret = 'Created series id\'s (all final type): ';
    SET serie_amount = Ceil(signed_up/serie_size_param);
    label: WHILE serie_amount > 0 DO
    SET serie_amount = serie_amount - 1;
    INSERT INTO series (discipline_id, meeting_id,s_type)
    SELECT discipline_id_param, meeting_id_param, 'final'
    FROM DUAL;
    SELECT MAX(series_id) INTO serie_id_p  from series;
    SET ret = CONCAT(ret, CONVERT(serie_id_p,char),' ');
    END WHILE label;
    END IF;

    RETURN ret;
END;
```

Listing 10: Procedura dyskwalifikująca danego atletę z danego meetingu - ustawienie wyników na NULL

```
CREATE PROCEDURE lekkoatletyka.athlete_dsq
(IN athlete_id_p INT, meeting_id_p INT)
BEGIN
    UPDATE track_results
        SET result = NULL
        WHERE athlete_id=athlete_id_p AND series_id IN
            (SELECT series_id
            FROM series
            WHERE meeting_id = meeting_id_p);
    UPDATE jump_results
        SET result = NULL
        WHERE athlete_id=athlete_id_p AND series_id IN
            (SELECT series_id
            FROM series
            WHERE meeting_id = meeting_id_p);
END;
```

4 Przypadki użycia

Listing 11: Wyszukanie użyte w funkcji create_series. Wyszukanie przyporządkowuje KAŻDEMU atlecie z bazy jego najlepszy wynik w wybranej dyscyplinie biegowej.

```
SELECT a.athlete_id,
IF(discipline_id = discipline_id_param,MIN(t.result), NULL)
AS result
FROM athletes a
LEFT JOIN track_results t ON a.athlete_id = t.athlete_id
LEFT JOIN series s        ON t.series_id = s.series_id
GROUP BY a.athlete_id
ORDER BY MIN(t.result) IS NULL ASC, MIN(t.result)
```

Listing 12: Wyświetlenie wszystkich wyników atlety w karierze

```
SELECT first_name,last_name,discipline,result,meeting_date
FROM athlete_track_results
WHERE athlete_id = WYBRANY_ATLETA;
```

Przykładowy rezultat:

first_name	last_name	discipline	result	meeting_date
Usain	Bolt	100m	00:00:09.690	2008-08-16
Usain	Bolt	200m	00:00:19.300	2008-08-20
Usain	Bolt	100m	00:00:09.580	2009-08-16

Listing 13: Wyświetlenie wszystkich najlepszych wyników

```
SELECT first_name,last_name,discipline,record,meeting_date
FROM track_records;
```

Przykładowy rezultat:

first_name	last_name	discipline	result	meeting_date
Usain	Bolt	100m	00:00:09.690	2008-08-16
Usain	Bolt	200m	00:00:19.190	2008-08-20
Wayde	van Niekerk	400m	00:00:43.030	2016-08-14

Listing 14: Przykładowe tworzenie limitu, zapisywanie atletów i tworzenie serii

```
CALL create_limit_track(MITYNG, DYSCYPLINA);
CALL signup_track(ATLETA_1, MITYNG, DYSCYPLINA);
[...]
CALL signup_track(ATLETA_40, MITYNG, DYSCYPLINA);
CALL create_series('normal', MITYNG, DYSCYPLINA, 24, 8, 0);
```

Powyższy kod tworzy odpowiedni limit dla mityngu, zapisuje 40 różnych atletów, następnie z tych atletów wybierze 24 najlepszych, stworzy 3 $\left(\frac{24}{8}\right)$ serie, oraz zwróci ich ID dla organizatora.

5 Podsumowanie

Udało nam się utworzyć z naszych pomysłów bazę danych. Posiada ona sporą większość zakładanych przez nas podczas etapu planowania funkcjonalności. Dzięki temu że zanim zrobiliśmy praktyczną część projektu dobrze zaplanowaliśmy jak baza powinna wyglądać, udało nam się w głównej mierze uniknąć problemów. Zdaliśmy sobie sprawę że posiadanie kopii zapasowej całej bazy, albo skryptów tworzących tabele, dane tabel, widoki etc. jest bardzo ważne. Bez nich, gdyby wystąpił poważny błąd, moglibyśmy stracić mnóstwo czasu na odtwarzanie bazy oraz danych.