

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
(повна назва інституту/факультету)

КАФЕДРА інформатики та програмної інженерії
(повна назва кафедри)

КУРСОВА РОБОТА
з дисципліни «Бази даних»
(назва дисципліни)

на тему: »База даних для збереження даних про футбольні
чемпіонати»

Студента (ки) 2 курсу ІП-11 групи
спеціальності 121 «Інженерія програмного
забезпечення»

Панченко С.В

(прізвище та ініціали)

Керівник

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала

Кількість балів: _____ Оцінка ECTS _____

Члени комісії

(підпис)

(вчене звання, науковий ступінь, прізвище та ініціали)

(підпис)

(вчене звання, науковий ступінь, прізвище та ініціали)

(підпис)

(вчене звання, науковий ступінь, прізвище та ініціали)

Київ – 2022 рік

**Національний технічний університет України «Київський політехнічний
інститут імені Ігоря Сікорського»**

Факультет Інформатики та обчислювальної техніки
(повна назва)

Кафедра Інформатики та програмної інженерії
(повна назва)

Дисципліна Бази даних

Курс 2 Група ІП-11 Семестр 1

**З А В Д А Н Н Я
НА КУРСОВУ РОБОТУ СТУДЕНТУ**

Панченко Сергій Віталійович
(прізвище, ім'я, по батькові)

1. Тема роботи База даних для збереження даних про футбольні
чемпіонати

керівник роботи

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

2. Строк подання студентом роботи

3. Вихідні дані до роботи

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно
розробити)

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання курсового проекту	Строк виконання етапів проекту	Примітка
1.	Вступ	24.12.2022	
2.	Опис предметного середовища	24.12.2022	
3.	Постановка задачі	24.12.2022	
4.	Побудова ER-моделі	24.12.2022	
5.	Побудова реляційної схеми	24.12.2022	
6.	Створення бази даних з допомогою обраної СУБД	25.12.2022	
7.	Імпортування даних	25.12.2022	
8.	Створення користувачів та реалізація їх функціоналу	25.12.2022	
9.	Створення SQL запитів	25.12.2022	
10.	Висновок	25.12.2022	
11.	Перелік посилань	26.12.2022	

Студент

(підпис)

(прізвище та ініціали)

Керівник роботи

(підпис)

(прізвище та ініціали)

АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 81 сторінок, 10 рисунки, 12 таблиць, 1 посилання.

Об'єкт дослідження: база даних для збереження даних про футбольні чемпіонати.

Мета роботи: закріплення навичок з проектування, реалізації реляційних баз даних та їх використання для практичних задач.

Проведено аналіз предметного середовища, визначено сутності, атрибути та зв'язки між об'єктами. Побудована ER-модель предметного середовища, реляційну схему бази даних, виділено первинні та зовнішні ключі, визначено обмеження для підтримки цілісності даних. Розроблено скрипти для побудови спроектованої бази даних, імпортовано дані в неї, виконано різноманітні запити до неї.

Виконана програмна реалізація бази даних для збереження інформації про футбольні чемпіонати (варіант 23).

Зміст

Вступ.....	7
1 Опис предметної області.....	8
2 Постановка завдання.....	9
3 ER-Діаграма.....	10
3.1 Бізнес правила.....	10
3.2 Вибір сутностей.....	10
3.3 Набори атрибутів сутностей.....	11
Таблиця 3.1 — Сутності та їхні атрибути.....	11
Продовження таблиці 3.1.....	12
4 Реляційна модель бази даних.....	14
4.1 Побудова необхідних відношень та визначення первинних та зовнішніх ключів.....	14
4.2 Визначення обмежень цілісності для спроектованих відношень.....	14
5 Реалізація бази даних.....	16
5.1 Створення бази даних у форматі системи управління базою даних PostgreSQL.....	16
5.2 Імпортування даних в таблицю.....	20
6 Створення користувачів бази даних.....	23
6.1 Створення представника комітету ліги.....	23
6.2 Створення судді.....	23
7 SQL Запити.....	24
7.1 Тригери.....	24
7.1.1 check_action_match_trigger.....	24
7.1.2 Тригер yellow_card_trigger.....	26
7.1.3 Тригер red_card_trigger.....	27
7.1.4 Тригер red_card_out_trigger.....	29
7.1.5 Тригер border_player_team_player.....	30
7.2 Функції та процедури.....	31
7.2.1 Функція count_goals.....	31
7.2.2 Функція win_lose.....	32
7.2.3 Функція who_win.....	34
7.3 SQL запити.....	35
7.3.1 Список найкращих гравців.....	35
7.3.2 Список країн за кількістю матчів.....	36
7.3.3 Судді, які видали найбільшу кількість жовтих карток.....	37
7.3.4 Країни за найбільшою кількістю голів.....	38
7.3.5 Середній вік гравців по командах.....	39
7.3.6 Список гравців та їх тренерів.....	40
7.3.7 Середній час до першого гола.....	41
7.3.8 Список стадіонів, міст, країн.....	41
7.3.9 Країна з найбільшою кількістю національних команд.....	42
7.4 Представлення.....	43
7.4.1 team_trainer.....	43
7.4.2 player_country.....	44
7.5 Індекси.....	46
Висновок.....	47
Перелік посилань.....	48

ВСТУП

Бази даних є важливою частиною будь-якого бізнесу, адже за допомогою них можна зберігати важливі дані про клієнтів, прослідковувати залежності між явищами. На відміну від минулого, де все зберігалось у журналах, архівах, бази даних дають можливість швидко отримувати інформацію, що нас цікавить.

Обрана мною тема є актуальною, тому що спорт є важливою частиною нашого життя, а футбольні чемпіонати збирають сотні тисяч глядачів на стадіонах усіх куточків світу. До того ж подібні заходи потребують великих сил на їхню організацію, з чим дуже допомагають бази даних.

Дана робота присвячена вивченню розробки програмного забезпечення з SQL, і стосується проектування бази даних для збереження інформації про футбольні чемпіонати, що допоможе організаторам цих заходів. Задача полягає в аналізі предметної області, побудови ER-діаграми та датологічної схеми реляційної бази даних, написанні SQL скриптів та запитів до неї. Виконання роботи проведено з допомогою PostgreSQL, оскільки вона є потужним заобом побудови баз даних, а також її найзручніше використовувати під ОС Linux Ubuntu 22.10 .

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

Основним об'єктом дослідження предметної області є футбольні чемпіонати та процеси пов'язані з ними. Змагання з даного виду спорту включають у себе проведення матчів, які відбувається між командами з різних країн світу.

Зазвичай чемпіонати організовують на стадіонах у великих містах. Для спонсорів чемпіонатів важливо мати уявлення про те, яка аудиторія може прийти на захід, щоб ефективно прорекламувати певний продукт на оптимальну кількість людей.

Кожен матч проводиться у супроводі рефері та двох його асистентів, які бажано набираються з різних куточків світу для об'єктивності та неупередженості суджень під час гри.

Футбол є командною грою, а тому кожна з груп складається з гравців, які мають певні ролі: воротар, нападаючий, капітан тощо. Усі команди мають свого тренера, який готує гравців до гри.

Матч має бути обмежений у часі, тому важливо реєструвати, коли і з яких причин сталася та чи інша подія. Такими можуть бути: пенальті, забиття голу, вихід\вхід гравця на поле з різних причин(жовта картка, червона картка за повторне порушення тощо).

2 ПОСТАНОВКА ЗАВДАННЯ

Метою даної роботи є розробка бази даних для збереження результатів чемпіонатів. Тобто організація даних має бути такою в базі, щоб було максимально зручно вибудовувати план проведення футбольних заходів.

Виділимо основні задачі:

Комітет ліги має мати можливість:

- Змінювати місце та час проведення матчів
- Вирішувати, які команди будуть грати на матчах
- Призначати рефері та його(її) асистентів
- Змінювати склад команди

Суддя має мати змогу:

- Зараховувати голи
- Реєструвати пенальті
- Змушувати гравців йти/заходити на поле

Усі учасники можуть:

- Переглядати склад команд
- Дізнаватися час проведення чемпіонатів
- Дізнаватися інформацію про голи, пенальті, входи/виходи гравців з поля

З ER-ДІАГРАМА

Після аналізу було виділено такі сутності та зв'язки між ними:

3.1 Бізнес правила

- 1) Час забиття голу, пенальті, виходу/входу гравця не може бути більшим за час проведення матчу + додаткові 30 хвилин, які надає суддя за потреби. Однак нічого не може відбуватися під час перерви, яка триває 15 хвилин після першого тайму.
- 2) Гравець не може отримати більше двох жовтих карток
- 3) Під час видання другої жовтої карточки видається червона карточка
- 4) Після видачі червої карточки гравець негайно покидає поле
- 5) Команда не може мати більше 11 гравців
- 6) Команда не може мати більше 7 гравців в запасі
- 7) Гравці можуть бути лише в одній команді
- 8) Тренери можуть бути лише в одній команді

3.2 Вибір сутностей

- Подія
- Тип події
- Місто
- Країна
- Матч
- Гравець
- Команда
- Стадіон
- Роль гравця
- Тренер
- Суддя

3.3 Набори атрибутів сутностей

Таблиця 3.1 — Сутності та їхні атрибути

Сутність	Атрибути
country	id short_name name
city	id name country_id
stadium	id name city_id
team	id name country_id
trainer	id name team_id
player_role	id name
player	id team_id name player_role_id age is_substitute
referee	id name country_id
match	id stage first_time_start first_time_end second_time_start second_time_end is_extra_time extra_time_start extra_time_end stadium_id first_team_id second_team_id referee_id first_assistant_id second_assistant_id
action_type	id name

Продовження таблиці 3.1

action

id
action_time
reason
action_type_id
player_id
match_id

Сутність **country** пов'язана *один до багатьох* із сутністю **city**, адже декілька міст можуть розташовуватися в одній країні.

Сутність **country** пов'язана *один до багатьох* із сутністю **team**, адже декілька команд можуть бути з однієї країни.

Сутність **country** пов'язана *один до багатьох* із сутністю **referee**, адже декілька суддів можуть бути з однієї країни.

Сутність **city** пов'язана *один до багатьох* із сутністю **stadium**, адже декілька стадіонів можуть розташовуватися в одному місті.

Сутність **stadium** пов'язана *один до багатьох* із сутністю **match**, адже декілька матчів можуть проводитися на одному стадіоні.

Сутність **team** пов'язана *один до багатьох* із сутністю **trainer**, адже одна команда може мати декількох тренерів.

Сутність **team** пов'язана *один до багатьох* із сутністю **player**, адже декілька гравців можуть грати в одній команді(правила забороняють грати гравцям одночасно в двох командах)

Сутність **team** пов'язана *один до багатьох* із сутністю **match**, адже одна команда може грати в декількох матчах.

Сутність **player_role** пов'язана *один до багатьох* із сутністю **player**, оскільки декілька гравців можуть мати одну роль.

Сутність **referee** пов'язана *один до багатьох* із сутністю **match**, бо на один матч припадає по одному судді та два асистенти на декілька матчів.

Сутність **action_type** пов'язана *один до багатьох* із сутністю **action**, бо на один тип події може припадати декілька подій.

4 РЕЛЯЦІЙНА МОДЕЛЬ БАЗИ ДАНИХ

4.1 Побудова необхідних відношень та визначення первинних та зовнішніх ключів

На рисунку можна побачити, що база даних знаходить у 3 нормальній формі, оскільки: поля таблиці декомпозовані, атрибути функціонально повно залежать від первинного ключа, кожен неключовий атрибут не є транзитивно залежним від первинного ключа.

4.2 Визначення обмежень цілісності для спроектованих відношень

- 1) Якщо не існує зовнішніх ключів, що посиляються на значення головного ключа батьківського рядка, то тільки у цьому випадку його можна видалити. Для цього треба додати до таблиці обмеження з відповідними параметрами: `ALTER TABLE child ADD CONSTRAINT fk_parent_child FOREIGN KEY (parent_id) REFERENCES parent (id) ON UPDATE CASCADE ON DELETE RESTRICT`
- 2) Обов'язкові атрибути таблиць мають обмеження `NOT NULL` для запобігання помилок при роботі з даними



Рисунок 4.1 — Реляційна схема бази даних

5 РЕАЛІЗАЦІЯ БАЗИ ДАНИХ

5.1 Створення бази даних у форматі системи управління базою даних PostgreSQL

```
CREATE TABLE IF NOT EXISTS country (  
id INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,  
short_name varchar(3),  
name VARCHAR(40)  
);
```

```
CREATE TABLE IF NOT EXISTS city (  
id INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,  
name VARCHAR(40),  
country_id SERIAL  
);
```

```
ALTER TABLE city ADD CONSTRAINT fk_country_city FOREIGN  
KEY(country_id)  
REFERENCES country (id) ON UPDATE CASCADE ON DELETE RESTRICT;
```

```
CREATE TABLE IF NOT EXISTS stadium (  
id INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,  
name VARCHAR(40),  
city_id SERIAL  
);
```

```
ALTER TABLE stadium ADD CONSTRAINT fk_city_stadium FOREIGN  
KEY(city_id)  
REFERENCES city (id) ON UPDATE CASCADE ON DELETE RESTRICT;
```



```
CREATE TABLE IF NOT EXISTS team (  
id INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,  
name VARCHAR(40),  
country_id SERIAL  
);
```

```
ALTER TABLE team ADD CONSTRAINT fk_country_team FOREIGN  
KEY(country_id)  
REFERENCES country (id) ON UPDATE CASCADE ON DELETE RESTRICT;
```

```
CREATE TABLE IF NOT EXISTS trainer (  
id INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,  
name VARCHAR(40),  
team_id SERIAL  
);
```

```
ALTER TABLE trainer ADD CONSTRAINT fk_team_trainer FOREIGN  
KEY(team_id)  
REFERENCES team (id) ON UPDATE CASCADE ON DELETE RESTRICT;
```

```
CREATE TABLE IF NOT EXISTS player_role (  
id INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,  
name VARCHAR(40)  
);
```

```
CREATE TABLE IF NOT EXISTS player (  
id INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,  
team_id SERIAL,  
name VARCHAR(40),  
player_role_id SERIAL,
```

```
age INT,  
is_substitute bool  
);
```

```
ALTER TABLE player ADD CONSTRAINT fk_player_role_player FOREIGN  
KEY(player_role_id)  
REFERENCES player_role (id) ON UPDATE CASCADE ON DELETE  
RESTRICT;
```

```
CREATE TABLE IF NOT EXISTS referee (  
id INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,  
name VARCHAR(40),  
country_id SERIAL  
);
```

```
ALTER TABLE referee ADD CONSTRAINT fk_country_referee FOREIGN  
KEY(country_id)  
REFERENCES country (id) ON UPDATE CASCADE ON DELETE RESTRICT;
```

```
CREATE TABLE IF NOT EXISTS match (  
id INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,  
stage varchar(1) check (stage IN ('Q', 'S', 'F')),  
first_time_start timestamp,  
first_time_end timestamp,  
second_time_start timestamp,  
second_time_end timestamp,  
is_extra_time bool,  
extra_time_start timestamp,  
extra_time_end timestamp,  
stadium_id SERIAL,
```

```
first_team_id SERIAL,  
second_team_id SERIAL,  
referee_id SERIAL,  
first_assistant_id SERIAL,  
second_assistant_id SERIAL  
);
```

```
ALTER TABLE match ADD CONSTRAINT fk_stadium_match FOREIGN  
KEY(stadium_id)  
REFERENCES stadium (id) ON UPDATE CASCADE ON DELETE RESTRICT;
```

```
ALTER TABLE match ADD CONSTRAINT fk_match_first_team  
FOREIGN KEY(first_team_id) REFERENCES team (id)  
ON UPDATE CASCADE ON DELETE RESTRICT;
```

```
ALTER TABLE match ADD CONSTRAINT fk_match_second_team  
FOREIGN KEY(second_team_id) REFERENCES team (id)  
ON UPDATE CASCADE ON DELETE RESTRICT;
```

```
ALTER TABLE match ADD CONSTRAINT fk_referee_match FOREIGN  
KEY(referee_id)  
REFERENCES referee (id) ON UPDATE CASCADE ON DELETE RESTRICT;
```

```
ALTER TABLE match ADD CONSTRAINT fk_first_assistant_match  
FOREIGN KEY(first_assistant_id) REFERENCES referee (id)  
ON UPDATE CASCADE ON DELETE RESTRICT;
```

```
ALTER TABLE match ADD CONSTRAINT fk_second_assistant_match  
FOREIGN KEY(second_assistant_id) REFERENCES referee (id)  
ON UPDATE CASCADE ON DELETE RESTRICT;
```

```
CREATE TABLE IF NOT EXISTS action_type (  
id INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,  
name VARCHAR(40)  
);
```

```
CREATE TABLE IF NOT EXISTS action (  
id INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,  
action_time timestamp,  
reason VARCHAR(40),  
action_type_id SERIAL,  
player_id SERIAL,  
match_id SERIAL  
);
```

```
ALTER TABLE action ADD CONSTRAINT fk_action_type_action  
FOREIGN KEY(action_type_id) REFERENCES action_type (id)  
ON UPDATE CASCADE ON DELETE RESTRICT;
```

```
ALTER TABLE action ADD CONSTRAINT fk_player_action  
FOREIGN KEY(player_id) REFERENCES player (id)  
ON UPDATE CASCADE ON DELETE RESTRICT;
```

```
ALTER TABLE action ADD CONSTRAINT fk_match_action  
FOREIGN KEY(match_id) REFERENCES match (id)  
ON UPDATE CASCADE ON DELETE RESTRICT;
```

5.2 Імпортування даних в таблицю

Для імпортування даних у СУБД PostgreSQL існує два методи:

- 1) Імпортувати дані для кожної таблиці з .csv-файлів

```

course_work=# \copy country FROM '/tmp/country.csv' DELIMITER ',' CSV;
COPY 139
course_work=# \copy city FROM '/tmp/city.csv' DELIMITER ',' CSV;
COPY 1926
course_work=# \copy stadium FROM '/tmp/stadium.csv' DELIMITER ',' CSV;
COPY 2023
course_work=# \copy team FROM '/tmp/team.csv' DELIMITER ',' CSV;
COPY 1560
course_work=# \copy trainer FROM '/tmp/trainer.csv' DELIMITER ',' CSV;
COPY 1560

```

Рисунок 5.2.1 — Імпортування даних в таблиці country, city, stadium, team, trainer за допомогою CSV-файлів

```

postgres=# \c course_work;
You are now connected to database "course_work" as user "postgres".
course_work=# \copy action FROM '/tmp/action.csv' DELIMITER ',' CSV;
COPY 379725
course_work=# 

```

Рисунок 5.2.2 - Імпортування даних в таблицю action за допомогою CSV-файлів

```

course_work=# \copy player FROM '/tmp/player.csv' DELIMITER ',' CSV;
COPY 19404
course_work=# \copy referee FROM '/tmp/referee.csv' DELIMITER ',' CSV;
COPY 2999
course_work=# \copy match FROM '/tmp/match.csv' DELIMITER ',' CSV;
COPY 99999

```

Рисунок 5.2.3 - Імпортування даних в таблиці player, referee, match action за допомогою CSV-файлів

2) Імпортувати дані з скрипта .sql

```

postgres@localhost:~$ psql -h localhost -d course_work -f '/tmp/player_role.sql'
Password for user postgres:
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1

```

Рисунок 5.2.4 - Імпортування даних в таблицю player_role за допомогою SQL-скриптів

```
postgres@localhost:~$ psql -h localhost -d course_work -f '/tmp/action_type.sql'
Password for user postgres:
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
postgres@localhost:~$ psql
psql (14.5 (Ubuntu 14.5-0ubuntu0.22.04.1))
Type "help" for help.
```

Рисунок 5.2.5 - Імпортування даних в таблицю action_type за допомогою SQL-скриптів

6 СТВОРЕННЯ КОРИСТУВАЧІВ БАЗИ ДАНИХ

6.1 Створення представника комітету ліги

```
course_work=# CREATE USER league_member; --
```

Рисунок 6.1 — Створення користувача представника комітету ліги

```
course_work=# GRANT ALL PRIVILEGES ON
               country,
               city,
               stadium,
               match,
               referee,
               team,
               player,
               trainer
TO
               league_member;

GRANT SELECT ON
               action,
               action_type,
               player_role
TO
               league_member;
GRANT
GRANT
course_work=#
```

Рисунок 6.2 — Надання прав представнику ліги

6.2 Створення судді

```
course_work=# CREATE USER judge;
GRANT ALL PRIVILEGES ON
               action,
               action_type,
               player_role
TO
               judge;
CREATE ROLE
GRANT
course_work=#
```

Рисунок 6.3 — Створення судді та надання йому прав

7 SQL ЗАПИТИ

7.1 Тригери

7.1.1 check_action_match_trigger

Створення тригера для таблиці action, щоб перевіряти аргументи на првильність при виконнанні INSERT та UPDATE

```
CREATE FUNCTION check_action_match()  
RETURNS TRIGGER  
AS $$  
DECLARE  
    fts timestamp;  
    fte timestamp;  
    sts timestamp;  
    ste timestamp;  
    iet BOOL;  
    ets timestamp;  
    ete timestamp;  
    t_one INT;  
    t_two INT;  
BEGIN  
    SELECT  
        first_time_start,  
        first_time_end,  
        second_time_start,  
        second_time_end,  
        is_extra_time,  
        extra_time_start,  
        extra_time_end,  
        first_team_id,  
        second_team_id  
    INTO  
        fts,  
        fte,  
        sts,  
        ste,  
        iet,  
        ets,  
        ete,  
        t_one,
```



```

        t_two
FROM
    match
WHERE
    match.id = NEW.match_id;
IF NOT ( NEW.action_time BETWEEN fts AND fte
        OR NEW.action_time BETWEEN sts AND ste
        OR ( iet AND NEW.action_time BETWEEN ets
AND ete )
    )
THEN
    RAISE EXCEPTION 'Action time is not in right borders';
END IF;

    IF NOT ( NEW.player_id IN (SELECT id FROM player WHERE
player.team_id = t_one)
        OR NEW.player_id IN (SELECT id FROM player WHERE
player.team_id = t_two)
    )
THEN
    RAISE EXCEPTION 'Player does not belong to either team';
END IF;
RETURN NEW;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER check_action_match_trigger
BEFORE INSERT OR UPDATE
ON action
FOR EACH ROW
EXECUTE PROCEDURE check_action_match();

```

id	fts	fte	sts	ste	iet	ets	ete	t_one	t_two
1	2018-10-16 04:52:41	2018-10-16 05:37:41	2018-10-16 05:52:41	2018-10-16 06:37:41	t	2018-10-16 06:52:41	2018-10-16 07:22:41	836	148

Рисунок 7.1 — Рядок з атрибутами, що описують матч

```
SELECT id, name, team_id FROM player WHERE team_id = 836;
```

id	name	team_id
10365	Sisinia Elansky	836
10366	Yosef Wolffsen	836
10367	Iosune Lasheras	836
10368	Casiano Travesedo	836
10369	Assunta Asturias	836
10370	Conceso Cousido	836
10371	Eimantas Izusquieta	836
10372	Montserrat Cabalhanas	836
10373	Houssam Uggias	836
10374	Alejos Luessen	836
10375	Grisha Esquer	836

Рисунок 7.2 — список гравців команди з id 836, що грає у матчі з рисунка 7.1

```
course_work=# INSERT INTO action VALUES (default,'2018-10-16 01:01:01', 'sdsdsdsd', 1, 10365, 1);
ERROR:  Action time is not in right borders
CONTEXT:  PL/pgSQL function check_action_match() line 42 at RAISE
course_work=#
```

Рисунок 7.3 — Повідомлення про помилку, що час події не попадає у часові межі проведення матчу

```
course_work=# INSERT INTO action VALUES (default,'2018-10-16 04:52:42', 'sdsdsdsd', 1, 10333, 1);
ERROR:  Player does not belong to either team
CONTEXT:  PL/pgSQL function check_action_match() line 49 at RAISE
course_work=#
```

Рисунок 7.4 — Повідомлення про помилку, що гравець, який виконав подію, не належить до жодної з двох команд матчу

7.1.2 Тригер yellow_card_trigger

Створення тригера для таблиці action, щоб гравець мав не більше двох жовтих карток за матч

```
CREATE FUNCTION yellow_card()
RETURNS TRIGGER
AS $$
DECLARE
    counter INT;
BEGIN
    IF NEW.action_type_id = 1
    THEN
        SELECT
            COUNT(*)
```

```

        INTO
            counter
        FROM
            action
        WHERE
            action_type_id = 1
            AND player_id = NEW.player_id
            AND match_id = NEW.match_id;

        IF counter + 1 > 2
        THEN
            RAISE EXCEPTION 'Player can not be
given with more than two yellow cards';
        END IF;
    END IF;
    RETURN NEW;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER yellow_card_trigger
    BEFORE INSERT
    ON action
    FOR EACH ROW
        EXECUTE PROCEDURE yellow_card();

```

```

course_work=# INSERT INTO action VALUES (1000005,'2018-10-16 04:52:42', 'sdsdsdsd', 1, 10367, 1);
INSERT 0 1
course_work=# INSERT INTO action VALUES (1000006,'2018-10-16 04:53:42', 'sdsdsdsd', 1, 10367, 1);
INSERT 0 1
course_work=# INSERT INTO action VALUES (1000007,'2018-10-16 04:53:42', 'sdsdsdsd', 1, 10367, 1);
ERROR:  Player can not be given with more than two yellow cards
CONTEXT:  PL/pgSQL function yellow_card() line 20 at RAISE
course_work=# 

```

Рисунок 7.5 — Повідомлення про помилку, що гравець не може мати більше ніж дві жовті картки

7.1.3 Тригер red_card_trigger

Створення тригера для таблиці action, щоб надати гравцю червону картку після отримання двох жовтих

```

CREATE FUNCTION red_card()
    RETURNS TRIGGER
    AS $$

```

```

DECLARE
    counter INT;
BEGIN
    IF NEW.action_type_id = 1
    THEN
        SELECT
            COUNT(*)
        INTO
            counter
        FROM
            action
        WHERE
            action_type_id = 1
            AND player_id = NEW.player_id
            AND match_id = NEW.match_id;

        IF counter = 2
        THEN
            INSERT INTO
                action
            VALUES (
                NEW.id + 1,
                NEW.action_time,
                NEW.reason,
                2,
                NEW.player_id,
                NEW.match_id
            );

            RAISE NOTICE 'Inserted red card';
        END IF;
    END IF;
    RETURN NEW;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER red_card_trigger
    AFTER INSERT
    ON action
    FOR EACH ROW
        EXECUTE PROCEDURE red_card();

```

```

course_work=# INSERT INTO action VALUES (1000009,'2018-10-16 04:52:42', 'sdsdsdsd', 1, 10368, 1);
INSERT 0 1
course_work=# INSERT INTO action VALUES (1000010,'2018-10-16 04:53:42', 'sdsdsdsd', 1, 10368, 1);
NOTICE: Inserted red card
INSERT 0 1

```

Рисунок 7.6 — Вставка червоної картки після двох жовтих

```

course_work=# SELECT action.id, action_type.name, player.name FROM action
INNER JOIN player ON player.id = action.player_id
INNER JOIN action_type ON action_type.id = action.action_type_id
WHERE action.player_id = 10368 AND action.match_id = 1;

```

id	name	name
1000010	Yellow Card	Casiano Travesedo
1000011	Red Card	Casiano Travesedo
1000009	Yellow Card	Casiano Travesedo

(3 rows)

Рисунок 7.7 — Результат роботи red_card_trigger

7.1.4 Тригер red_card_out_trigger

Створення тригера для таблиці action, щоб гравець полишав поле після отримання червоної картки

```

CREATE FUNCTION red_card_out()
RETURNS TRIGGER
AS $$
DECLARE
    counter INT;
BEGIN
    IF NEW.action_type_id = 2
    THEN
        INSERT INTO
            action
        VALUES (
            NEW.id + 1,
            NEW.action_time,
            NEW.reason,
            4,
            NEW.player_id,
            NEW.match_id
        );

        RAISE NOTICE 'Player Out';
    
```

```

        END IF;
        RETURN NEW;
    END;
$$
LANGUAGE plpgsql;

```

```

CREATE TRIGGER red_card_out_trigger
    AFTER INSERT
    ON action
    FOR EACH ROW
    EXECUTE PROCEDURE red_card_out();

```

```

course_work=# INSERT INTO action VALUES (1000017,'2018-10-16 04:53:42', 'Rude Language', 2, 10369, 1);
NOTICE: Player Out
INSERT 0 1
course_work=# SELECT action.id, action_type.name, player.name FROM action
INNER JOIN player ON player.id = action.player_id
INNER JOIN action_type ON action_type.id = action.action_type_id
WHERE action.player_id = 10369 AND action.match_id = 1;

```

id	name	name
1000017	Red Card	Assunta Asturias
1000018	Player Out	Assunta Asturias

```

(2 rows)

```

Рисунок 7.8 — Результат роботи red_card_out_trigger

7.1.5 Тригер border_player_team_player

Створення тригера для таблиці player, щоб щоб команда, до якої належить гравець, не мала більше 7 гравців у запасі і 11 активних гравців

```

CREATE FUNCTION border_player_team()
    RETURNS TRIGGER
    AS $$
    DECLARE
        counter INT;
    BEGIN
        SELECT COUNT(*) INTO counter FROM player
        WHERE team_id = NEW.team_id AND is_substitute =
NEW.is_substitute;

        IF NEW.is_substitute AND counter >=7
            THEN RAISE EXCEPTION 'Number of substitute
players can not be higher than 7';
        ELSIF counter >=11
            THEN RAISE EXCEPTION 'Number of regular
players can not be higher than 11';

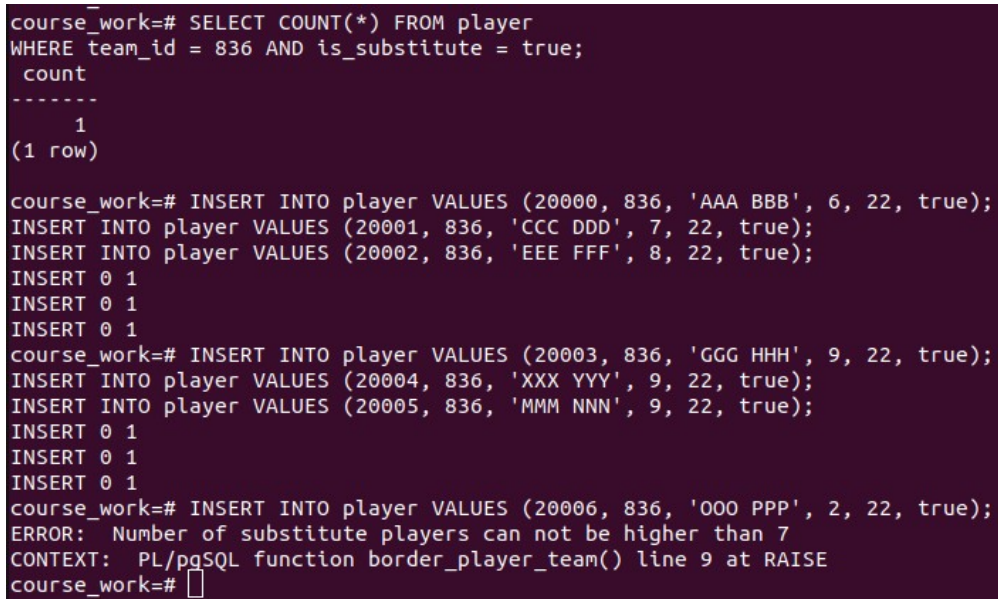
```

```

        END IF;
        RETURN NEW;
    END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER border_player_team_trigger
    BEFORE INSERT OR UPDATE
    ON player
    FOR EACH ROW
        EXECUTE PROCEDURE border_player_team();

```



```

course_work=# SELECT COUNT(*) FROM player
WHERE team_id = 836 AND is_substitute = true;
 count
-----
      1
(1 row)

course_work=# INSERT INTO player VALUES (20000, 836, 'AAA BBB', 6, 22, true);
INSERT INTO player VALUES (20001, 836, 'CCC DDD', 7, 22, true);
INSERT INTO player VALUES (20002, 836, 'EEE FFF', 8, 22, true);
INSERT 0 1
INSERT 0 1
INSERT 0 1
course_work=# INSERT INTO player VALUES (20003, 836, 'GGG HHH', 9, 22, true);
INSERT INTO player VALUES (20004, 836, 'XXX YYY', 9, 22, true);
INSERT INTO player VALUES (20005, 836, 'MMM NNN', 9, 22, true);
INSERT 0 1
INSERT 0 1
INSERT 0 1
course_work=# INSERT INTO player VALUES (20006, 836, 'OOO PPP', 2, 22, true);
ERROR:  Number of substitute players can not be higher than 7
CONTEXT:  PL/pgSQL function border_player_team() line 9 at RAISE
course_work=# 

```

Рисунок 7.9 — Повідомлення про перевищення допустимої кількості гравців у запасі

7.2 Функції та процедури

7.2.1 Функція count_goals

Створення функції count_goals для підрахунку кількості голів, які забила команда під час матчу

```

CREATE OR REPLACE FUNCTION count_goals(mid INT, tid INT)
    RETURNS INT
    AS $$
    DECLARE
        t_one INT;
        t_two INT;

```

```

BEGIN
    RETURN (
        SELECT COUNT(*) FROM action
        INNER JOIN player ON player.id = player_id
        WHERE action.action_type_id = 6
        AND action.match_id = mid
        AND player.team_id = tid
    );
END;
$$
LANGUAGE plpgsql;

```

```

course_work=# SELECT first_team_id, second_team_id FROM match WHERE id = 1;
 first_team_id | second_team_id 
-----+-----
          836 |          148 
(1 row)

course_work=# SELECT count_goals(1, 836);
 count_goals 
-----
          0 
(1 row)

course_work=# SELECT count_goals(1, 148);
 count_goals 
-----
          1 
(1 row)

course_work=# 

```

Рисунок 7.10 — Використання функції count_goals для підрахунку голів

7.2.2 Функція win_lose

Створення функції win_lose для визначення, чи виграла дана команда певний матч

```

CREATE OR REPLACE FUNCTION win_lose(mid INT, tid INT)
RETURNS INT
AS $$
DECLARE
    t_one INT;
    t_two INT;
    g_one INT;
    g_two INT;
    g_cur INT;
    g_other INT;

```



```

match.id;

        res INT;
BEGIN
    SELECT first_team_id, second_team_id
    INTO t_one, t_two FROM match WHERE mid =

SELECT count_goals(mid, t_one) INTO g_one;
SELECT count_goals(mid, t_two) INTO g_two;

    IF tid = t_one
    THEN
        g_cur := g_one;
        g_other := g_two;
    ELSE
        g_cur := g_two;
        g_other := g_one;
    END IF;

    IF g_cur > g_other
    THEN
        res := 1;
    ELSIF g_cur = g_other
    THEN
        res := 0;
    ELSE
        res := -1;
    END IF;

    RETURN res;
END;
$$
LANGUAGE plpgsql;

```

```

course_work=# SELECT id, first_team_id, second_team_id FROM match WHERE id = 1;
 id | first_team_id | second_team_id 
----+-----+-----
  1 |          836 |          148 
(1 row)

course_work=# SELECT win_lose(1, 836);
 win_lose 
-----
        -1 
(1 row)

course_work=# SELECT win_lose(1, 148);
 win_lose 
-----
         1 
(1 row)

course_work=# 

```

Рисунок 7.11 — Використання функції win_lose

7.2.3 Функція who_win

Створення функції who_win для визначення команди, яка виграла під час матчу

```

CREATE OR REPLACE PROCEDURE who_win(mid INT)
AS $$
DECLARE
    t_one INT;
    t_two INT;
    n_one VARCHAR(40);
    n_two VARCHAR(40);
    r_one INT;
    r_two INT;
BEGIN
    SELECT first_team_id, second_team_id
    INTO t_one, t_two FROM match WHERE mid = match.id;

    SELECT name INTO n_one FROM team WHERE id = t_one;
    SELECT name INTO n_two FROM team WHERE id = t_two;

    SELECT win_lose(mid, t_one) INTO r_one;
    SELECT win_lose(mid, t_two) INTO r_two;

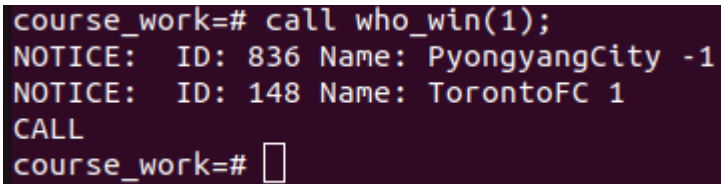
    RAISE NOTICE 'ID: % Name: % %', t_one, n_one, r_one;

```

```

        RAISE NOTICE 'ID: % Name: % %', t_two, n_two, r_two;
END;
$$
LANGUAGE plpgsql;

```



```

course_work=# call who_win(1);
NOTICE:  ID: 836 Name: PyongyangCity -1
NOTICE:  ID: 148 Name: TorontoFC 1
CALL
course_work=# 

```

Рисунок 7.12 — Повідомлення про інформацію щодо результатів матчу

7.3 SQL запити

7.3.1 Список найкращих гравців

Запит для побудови списку гравців відсортованих за кількістю голів

```

SELECT
    player.name,
    total_goals
FROM
    (SELECT
        player_id,
        COUNT(goal) AS total_goals
    FROM
        (SELECT
            player_id,
            action.action_type_id AS goal
        FROM
            action
        WHERE
            action.action_type_id = 6
        ) AS a
    GROUP BY
        player_id
    ) AS b
INNER JOIN
    player
ON
    player.id = player_id

```

```
ORDER BY
    total_goals DESC;
```

name	total_goals
Purísima Flesch	18
Nosakhare Rafols	16
Wilson Hanslmaier	16
Hssain Muhlsteffen	15
Rosari Barreras	15
Ussumane Malumbres	15

Рисунок 7.13 — Список найкращих гравців

7.3.2 Список країн за кількістю матчів

Створення списку країн, відсортованих за кількістю прийнятих матчів.

```
SELECT
    country.name, COUNT(country.name) AS total_accepted
FROM
    match
INNER JOIN
    stadium
ON
    stadium.id = match.stadium_id
INNER JOIN
    city
ON
    city.id = stadium.city_id
INNER JOIN
    country
ON
    country.id = city.country_id
GROUP BY
    country.name
ORDER BY
    total_accepted DESC;
```

name	total_accepted
Poland	11958
UnitedStatesofAmerica	5875
England	5828
Germany	4887
Spain	3498
France	2862
Italy	2646
Japan	2433

Рисунок 7.14 — Список країн, відсортованих за кількістю прийнятих матчів

7.3.3 Судді, які видали найбільшу кількість жовтих карток

Створення запиту для створення списку суддів, що віддали найбільшу кількість жовтих карток

```

SELECT
    referee.name,
    total_yellow
FROM
(
    SELECT
        rid,
        COUNT(rid) AS total_yellow
    FROM
        (
            SELECT
                referee.id AS rid
            FROM
                action
            INNER JOIN
                match
            ON
                match.id = action.match_id
            INNER JOIN
                referee
            ON
                referee.id = match.referee_id
            WHERE
                action.action_type_id = 1
        ) AS a
    GROUP BY
        rid

```

```

) AS b
INNER JOIN
    referee
ON
    b.rid = referee.id
ORDER BY
    total_yellow DESC;

```

name	total_yellow
Nurys Ustynyuk	31
Hyusein Giesecke	29
Branislav Solposto	29
Yarisa Anikushin	29
Cristiana Haversath	29
Urcesino Arenes	29
Isam Ochsenmeier	28
Rosy Mikhelson	28
Esdras Kusgens	28
Nasera Winstanley	28

Рисунок 7.15 — Список суддів, які видали найбільшу кількість жовтих карток

7.3.4 Країни за найбільшою кількістю голів

Створення списку країн, відсортованих за сумарною кількістю голів, які забили команди, що належать до даних держав.

```

SELECT
    country_name,
    COUNT(country_name) AS total_goals
FROM
    (
        SELECT
            country.name AS country_name
        FROM
            action
        INNER JOIN
            player
        ON
            player.id = action.player_id
        INNER JOIN
            team
        ON
            team.id = player.team_id
    )

```

```

INNER JOIN
    country
ON
    team.country_id = country.id
WHERE
    action.action_type_id = 6
) AS a
GROUP BY
    country_name
ORDER BY
    total_goals DESC;

```

country_name	total_goals
Poland	10319
UnitedStatesofAmerica	6439
England	6194
Germany	5115
Spain	3816
France	2983
Italy	2564
Japan	2097
Portugal	2044
Romania	2008

Рисунок 7.16 — Список країн, відсортовані за кількістю забитих голів

7.3.5 Середній вік гравців по командах

Створення запиту сортування команд за середнім віком гравців у них у зростаючому порядку.

```

SELECT
    team.id,
    team.name,
    ROUND(avg_age, 3) as average_age
FROM
    (
        SELECT
            team_id,
            AVG(age) as avg_age
        FROM
            player
        GROUP BY

```

```

        team_id
) AS a
INNER JOIN
    team
ON
    team.id = team_id
ORDER BY
    average_age ASC;

```

id	name	average_age
244	Siad	23.333
492	VfLLübeck	23.385
1260	SantAndreu	23.417
159	O'Higgins	23.462
1232	NKMaribor	23.727
759	Atlante	23.846
890	Chemik	23.889
170	Nublense	23.889

Рисунок 7.17 — Список команд, відсортованих за середнім віком гравців

7.3.6 Список гравців та їх тренерів

Створення списку команд та їхніх тренерів.

```

SELECT
    team.name,
    trainer.name
FROM
    trainer
INNER JOIN
    team
ON
    team.id = trainer.team_id;

```

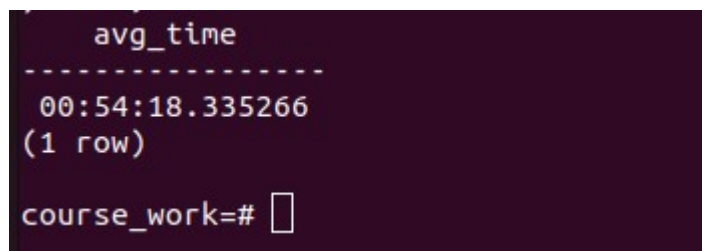
name	name
Besëlidhja	Elisabetta Tosca
FlamurtariVlorë	Joslyn Espiga
KFLaçi	Klaas Nunheiser
Teuta	Elyse Andreason
Skënderbeu	Tsveta Anselmo
JSK	Rosali Dobretsky
MCOoran	Elhassan Bandeiras
MCEE	Mamasa Esquiva

Рисунок 7.18 — Список команд та їхніх тренерів

7.3.7 Середній час до першого гола

Створення запиту для визначення гола, який був забитий через найкоротшу кількість часу після початку гри.

```
SELECT
    AVG(m_diff_time) AS avg_time
FROM
    (
        SELECT
            mid,
            MIN(diff_time) AS m_diff_time
        FROM
            (
                SELECT
                    match.id as mid,
                    action.action_time - match.first_time_start AS
diff_time
                FROM
                    action
                INNER JOIN
                    match
                ON
                    match.id = action.match_id
                WHERE
                    action.action_type_id = 6
            ) AS a
        GROUP BY
            mid
    ) AS b;
```



```
avg_time
-----
00:54:18.335266
(1 row)

course_work=#
```

Рисунок 7.19 — Середній час до першого гола

7.3.8 Список стадіонів, міст, країн

Створення списку стадіонів, міст, країн, у яких вони знаходяться

```

SELECT
    country.name,
    city.name,
    stadium.name
FROM
    stadium
INNER JOIN
    city
ON
    stadium.city_id = city.id
INNER JOIN
    country
ON
    country.id = city.country_id;

```

name	name	name
Albania	Lezhë	StadiumiBesëlidhja
Albania	Vlorë	StadiumiFlamurtari
Albania	Laçi	StadiumiLaçi
Albania	Durrës	StadiumiNikoDovana
Albania	Tirana	StadiumiSelmanStërmasi
Albania	Korçë	StadiumiSkënderbeu
Algeria	TiziOuzu	Stade1erNovembre1954
Algeria	Oran	StadeAhmedZabana
Algeria	ElEulma	StadeAmarHareche
Algeria	Béjaïa	Stadedel'UnitéMaghrébine
Algeria	Chlef	StadeMohamedBoumezrag

Рисунок 7.20 - Список стадіонів, міст, країн

7.3.9 Країна з найбільшою кількістю національних команд

```

SELECT
    country.name,
    t_count
FROM
    (
        SELECT
            cid,
            COUNT(tid) AS t_count
        FROM
            (
                SELECT

```

```

        country.id AS cid,
        team.id AS tid
    FROM
        country
    INNER JOIN
        team
    ON
        team.country_id = country.id
) AS a
GROUP BY
    cid
ORDER BY
    t_count DESC
LIMIT 1
) AS b
INNER JOIN
    country
ON
    country.id = b.cid;

```

```

ON
    country.id = b.cid;
  name | t_count
-----+-----
  Poland |      177
(1 row)

```

Рисунок 7.21 — Країна з найбільшою кількістю національних команд

7.4 Представлення

7.4.1 team_trainer

Створення представлення для відображення списку команди та її тренера

```

CREATE VIEW team_trainer AS
    SELECT
        team.id AS team_id,
        team.name AS team_name,
        trainer.id AS trainer_id,
        trainer.name AS trainer_name
    FROM
        team

```

```

INNER JOIN
    trainer
ON
    trainer.team_id = team.id;

```

```

course_work=# CREATE VIEW team_trainer AS
SELECT
    team.id AS team_id,
    team.name AS team_name,
    trainer.id AS trainer_id,
    trainer.name AS trainer_name
FROM
    team
INNER JOIN
    trainer
ON
    trainer.team_id = team.id;
CREATE VIEW
course_work=# SELECT * FROM team_trainer;
course_work=# SELECT * FROM team_trainer LIMIT 10;
 team_id | team_name | trainer_id | trainer_name
-----+-----+-----+-----
      1 | Besëlidhja |      1 | Elisabetta Tosca
      2 | FlamurtariVlorë |      2 | Joslyn Espiga
      3 | KFLaçi |      3 | Klaas Nunheiser
      4 | Teuta |      4 | Elyse Andreason
      5 | Skënderbeu |      5 | Tsveta Anselmo
      6 | JSK |      6 | Rosali Dobretsky
      7 | MCOoran |      7 | Elhassan Bandejas
      8 | MCEE |      8 | Mamasa Esquiva
      9 | ASOChlef |      9 | Rakesh Tillens
     10 | RCKouba |     10 | Asteria Gates
(10 rows)

```

Рисунок 7.22 — Представлення team_trainer

7.4.2 player_country

Створення представлення для відображення списку гравців та країн, за яких вони грають

```

CREATE VIEW player_country AS
SELECT
    player.id AS player_id,
    player.name AS player_name,

```

```

country.id AS country_id,
country.name AS country_name
FROM
    player
INNER JOIN
    team
ON
    player.team_id = team.id
INNER JOIN
    country
ON
    country.id = team.country_id;

```

```

course_work=# CREATE VIEW player_country AS
SELECT
    player.id AS player_id,
    player.name AS player_name,
    country.id AS country_id,
    country.name AS country_name
FROM
    player
INNER JOIN
    team
ON
    player.team_id = team.id
INNER JOIN
    country
ON
    country.id = team.country_id;
CREATE VIEW
course_work=# SELECT * FROM player_country LIMIT 10;
 player_id |   player_name   | country_id | country_name
-----+-----+-----+-----
      0 | Rostislav Puschen |           1 | Albania
      1 | Tasnim Engels    |           1 | Albania
      2 | Celestina Tressl |           1 | Albania
      3 | Joshua Karkosch  |           1 | Albania
      4 | Susan Bachish    |           1 | Albania
      5 | Rasheeda Thiemens |           1 | Albania
      6 | Roselin Mirallas |           1 | Albania
      7 | Vashti Globisch  |           1 | Albania
      8 | Misbah Moebus    |           1 | Albania
      9 | Rosiane Hendricks |           1 | Albania
(10 rows)

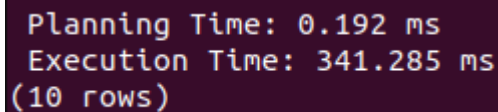
```

Рисунок 7.23 — Представления player_country

7.5 Індекси

Виконаємо простий SQL-запит, де покажемо подію та її назву. Спочатку без використання індекса

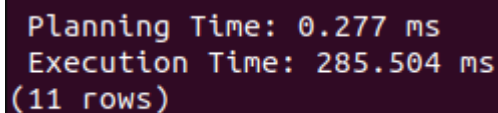
```
EXPLAIN ANALYZE
SELECT
    action.id,
    action.action_type_id,
    action_type.name
FROM
    action
INNER JOIN
    action_type
ON
    action.action_type_id = action_type.id
ORDER BY
    action.id DESC;
```



```
Planning Time: 0.192 ms
Execution Time: 341.285 ms
(10 rows)
```

Рисунок 7.24 — Запит без використання індексів

```
CREATE INDEX action_index ON action (id, action_type_id);
```



```
Planning Time: 0.277 ms
Execution Time: 285.504 ms
(11 rows)
```

Рисунок 7.25 — Запит з використанням індексів

Бачимо, що ефективність збільшилася у 1.19 разів. Ця різниця стане ще помітнішою на великих базах даних. Отже, індекси є важливими у роботі з базами даних.

ВИСНОВОК

Бази даних — це важливий інструмент для продуктивної роботи будь-якого сервісу. Вони дозволяють зручно та швидко зберігати інформацію та обробляти її, щоб виявити певні тенденції у поведінці споживачів продукту чи ринку.

У цій роботі було спроектовано базу даних для збереження результатів про чемпіонати футбольної ліги. У ході написання праці було вивчено предметне середовище, бізнес-правила, створено ER-діаграму, реляційну модель бази даних у третій нормальній формі. Також були написані SQL-скрипти, користувачі бази даних, тригери, функції та процедури.

ПЕРЕЛІК ПОСИЛАНЬ

1. <https://www.postgresql.org/docs/>
2. <https://www.postgresqltutorial.com/>