Санкт-Петербургский государственный университет

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ – ПРОЦЕССОВ УПРАВЛЕНИЯ

Мартынов Павел

Курсовая работа

Киберспортивная дисциплина Dota 2

Направление прикладная математика и информатика

Преподаватель : Филиппов Р.О.

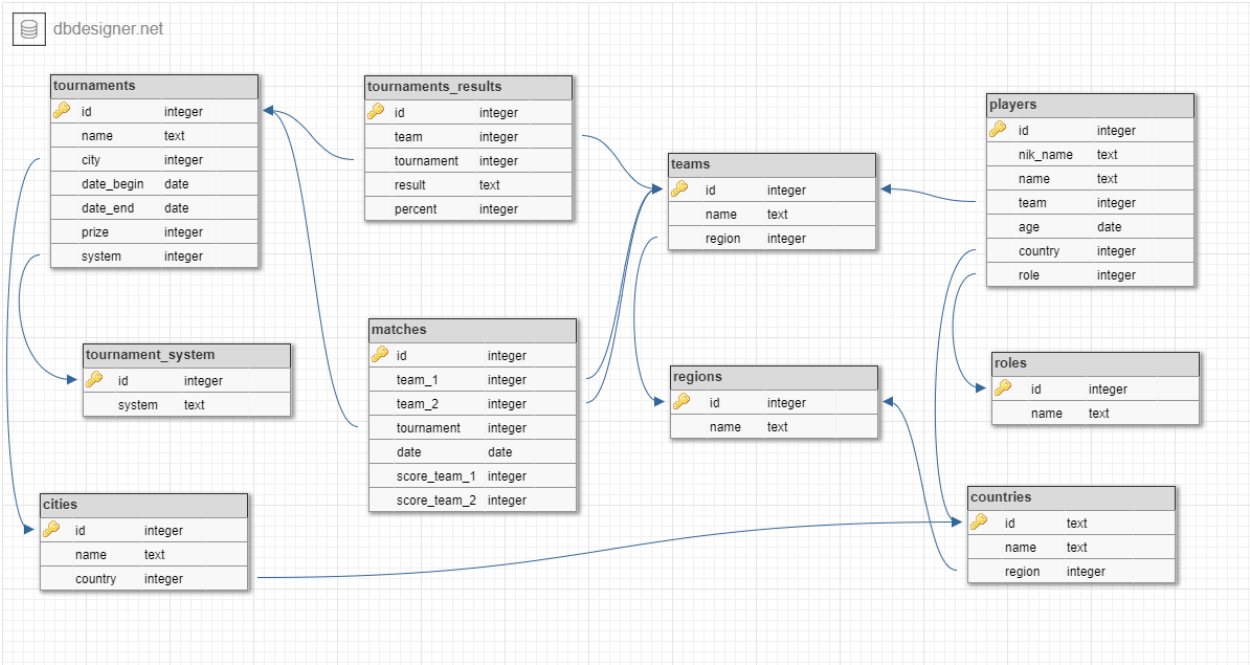
Санкт-Петербург

2017

Оглавление

Глава 1. Схема

Схема базы данных:



Players (игроки)

- Id
- Nik_name (ник в игре)
- Name (реальное имя)
- Team (команда, за которую играет)
- Age (дата рождения)
- Country (страна)
- Role (роль в игре)

Roles (роли)

- Id
- Name (название роли)

Teams (команды)

- Id
- Name (название команды)
- Region (регион, который команда представляет)

Regions (Регионы)

- Id
- Name (название региона)

Countries (страны)

- Id

- Name (название страны)
- Region (регион)

Cities (города)

- Id
- Name
- Country

Tournaments (турниры)

- Id
- Name (название турнира)
- City (город проведения)
- Date_begin (дата начала)
- Date_end (дата окончания)
- Prize (призовой фонд)
- System (система)

Tournaments_system (системы проведения турниров)

- Id
- System (название системы)

Tournaments _results (результат турнира)

- Id
- Team (команда)
- Tournament (турнир)
- Result (место)
- Percent (процент от призового фонда)

Matches (матчи)

- Id
- Team_1 (первая команда)
- Team_2 (вторая команда)
- Tournament (турнир)
- Date (дата матча)
- Score_team_1 (очков у первой команды)
- Score_team_2 (очков второй команды)

Глава 2. Описание базы данных

Players
Id. PRYMARY KEY

Nik_name, Name, Age. Тип string

Team. Один игрок одновременно играет только в одной команде. Внешний ключ на таблицу teams. Тип integer

Country. Внешний ключ на таблицу countries. Тип integer

Role. Внешний ключ на таблицу roles. Тип integer

Teams (команды)
Id. PRYMARY KEY

Name. Тип string

Region. Внешний ключ на таблицу regions. Тип integer

Regions (Регионы)
Id. PRYMARY KEY

Name. Тип string

Countries (страны)
Id. PRYMARY KEY

Name. Тип string

Region. Внешний ключ на таблицу regions. Тип integer

Cities (города)
Id. PRYMARY KEY

Name. Тип string

Country. Внешний ключ на таблицу countries

Tournaments (турниры)
Id. PRYMARY KEY

Name. Тип string

City. Город проведения турнира. Внешний ключ на таблицу cities

Date_begin, Date_end. Тип string

Prize. Призовой фонд на всех турнирах – деньги. Тип integer

System. Внешний ключ на таблицу tournaments_system

Tournaments_system (системы проведения турниров)
Id. PRYMARY KEY

System. В пределах этой БД было решено систему проведения разделить только на single (одно поражение – вылет с турнира) и double elimination (после первого поражения – падение в сетку лузеров). Тип string

Tournaments _results (результат турнира)
Id. PRYMARY KEY

Team. Внешний ключ на таблицу teams

Tournament. Внешний ключ на таблицу tournaments

Result. Место указываю строкой так как на большинстве турниров однозначно определяются только первые 3-4 места, в зависимости от системы проведения. Например, 4 команды делят 13-16 места, не сражаются между собой и забирают одинаковый процент от призового фонда

Percent. Тип integer

Matches (матчи)
Id. PRYMARY KEY

Team_1, Team_2. Внешний ключ на таблицу teams

Tournament. Внешний ключ на таблицу tournaments

Date. Дата указывается string’ом

Score_team_1, Score_team_2. Тип integer

Глава 3. Легкие запросы

1) Выбираем название команд из региона с id = 1

```
SELECT name AS team FROM teams WHERE region = 1;
```

Оптимизация:

Индексы: teams(region)

2) Выбираем названия турниров с призовым фондом больше или равным 1000000 и датой начала не позднее месяца назад

```
SELECT name AS tournament FROM tournaments WHERE prize >= 1000000 AND date_begin > current_date - interval '1 month';
```

Оптимизация:

Индексы: tournaments (prize), tournaments(date_begin)

3) Выбираем id команд с турнира с индексом 10 и сортируем их по убыванию выигранного процента от призового фонда

```
SELECT team AS team_id FROM tournaments_results WHERE tournament = 10 ORDER BY percent DESC;
```

Оптимизация:

Индексы: tournaments_results (tournament)

4) Выбираем команды и количество игрков в ней (те что занесены в БД)

```
SELECT team, count(*) AS number_of_players FROM players GROUP BY team;
```

Оптимизация:

Индексы: players (team)

Глава 4. Средние запросы

1) Выбираем ники и имена игроков которые играют на роли semi-support с сортировкой их по возрасту

```
SELECT nik_name, players.name AS real_name FROM players INNER JOIN roles ON players.role = roles.id WHERE roles.name = 'semi-support' ORDER BY age;
```

Оптимизация:

Индексы: players (role), roles(id), roles(name)

2) Выбираем игроков, их ники, страну рождения и их команду, у которые страна рождения географически принадлежит тому же региону, что и команда за которую они играют, сортируем по возрасту

```
SELECT nik_name AS nik, temp.name, temp.country_name, teams.name AS team FROM teams INNER JOIN (SELECT players.nik_name, players.name, players.age, countries.name AS country_name, team, region FROM players INNER JOIN countries ON players.country = countries.id) AS temp ON temp.team = teams.id WHERE teams.region = temp.region ORDER BY age;
```

Оптимизация:

Индексы: players (nik_name), players (team), players(country), countries(id), teams (region)

3) Выбираем команду и количество турниров на которых она побывала (из внесенных в БД), выводим по убыванию

```
SELECT name, count(tournament) AS registered_tournaments FROM teams INNER JOIN tournaments_results ON teams.id = tournaments_results.team GROUP BY name ORDER BY registered_tournaments DESC;
```

Оптимизация:

Индексы: = tournaments_results (team), teams (id)

Во всех случаях индексация не привела к повышению производительности из-за слишком маленького объема данных

Глава 5. Сложные запросы

1) Выбираем сумму призовых фондов по регионам проведения турниров, не учиывая The Internationalы (правда их в БД всего 1), сортируем по убыванию

```
SELECT r.name AS region_name, sum(t.prize) AS prize_sum
FROM tournaments AS t
INNER JOIN cities
ON (t.city = cities.id AND t.name NOT LIKE '%International%')
INNER JOIN countries AS c
ON (cities.country = c.id)
INNER JOIN regions AS r
ON (c.region = r.id)
GROUP BY r.name
ORDER BY prize_sum DESC;
```

2) Выбираем все команды, выводим название, количество игроков в ней, которым больше 26, сортируем по этому значению (по убыванию) и выводим разницу в возрасте между самым старым и самым молодым игроками

```
SELECT t.name AS team_name, count(p.name) AS players,
(AGE((SELECT MAX(age) FROM players WHERE team = t.id),
   (SELECT MIN(age) FROM players WHERE team = t.id))
        ) AS dif
FROM players AS p
RIGHT JOIN teams AS t
ON (p.team = t.id AND
p.age < date(current_date - interval '26 year'))
```

```
GROUP BY team_name, dif
ORDER BY players DESC;
```

3) Выбираем матчи сыгранные не "всухую" и выводим в привычном для обчного пользователя виде

```
SELECT t_1.name AS team1, concat(to_char(score_team_1, '9'), ':', to_char(score_team_2, '9')) AS score, t_2.name AS team2
FROM matches
INNER JOIN teams AS t_1
ON (team_1 = t_1.id AND score_team_1 > 0)
INNER JOIN teams AS t_2
ON (team_2 = t_2.id AND score_team_2 > 0)
```

Ссылка на репозиторий: https://github.com/varlogen/spbu_db