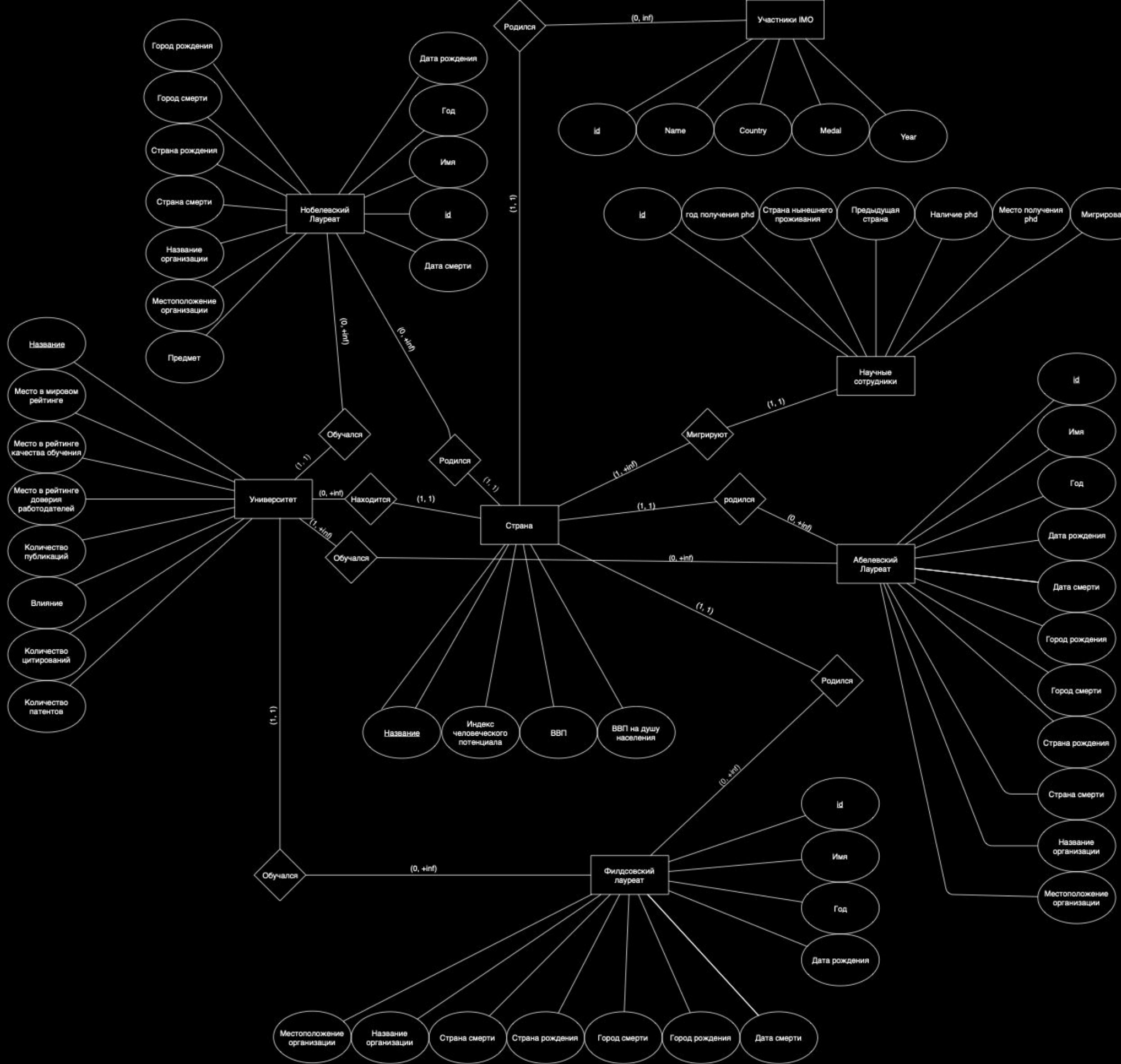
Семестровый проект по БД

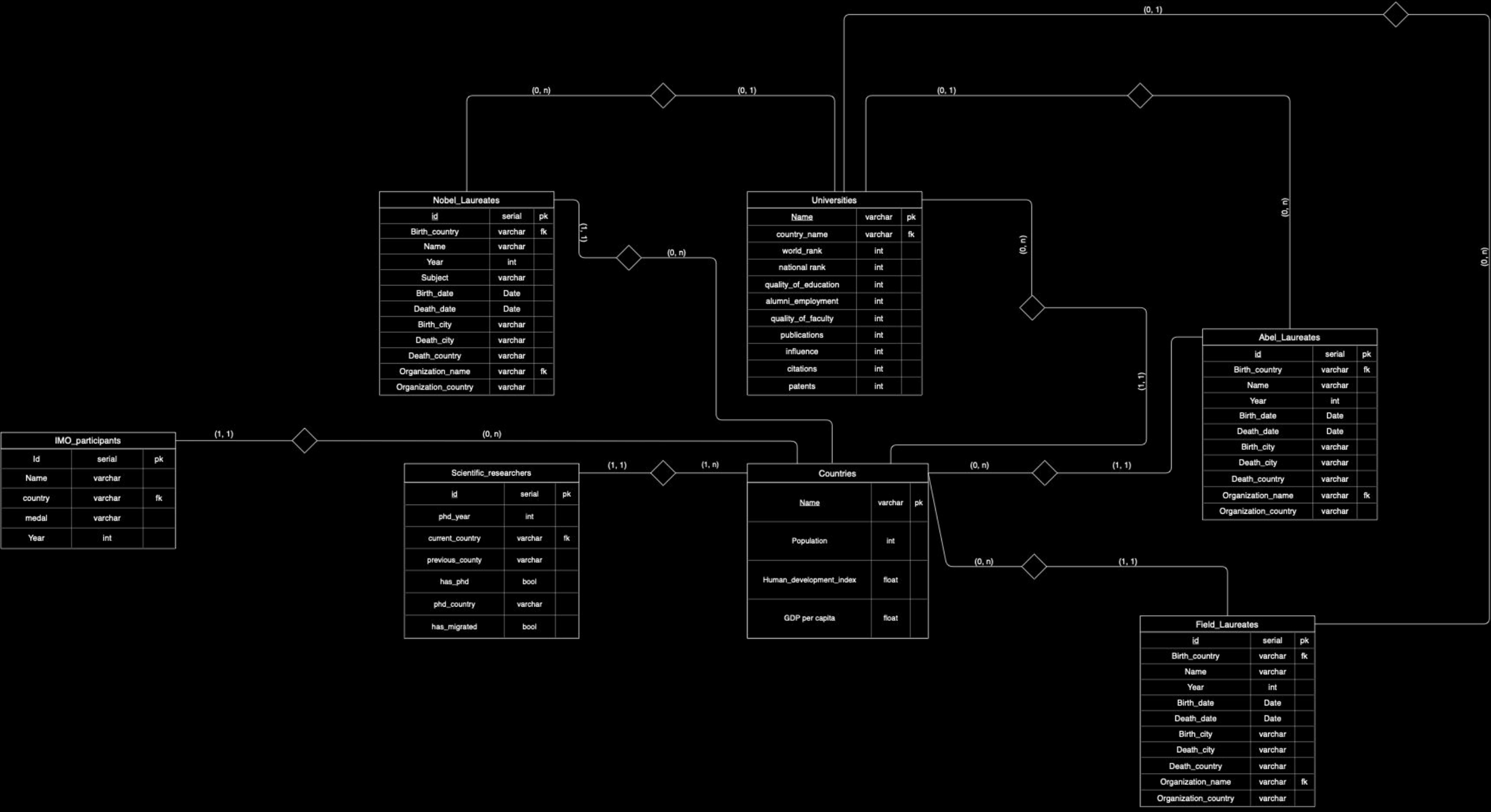
Долгов Максим, 824

Описание предметной области

В построенной базе данных содержится информация, которая характеризует образование и науку стран мира. Например, присутствуют лауреаты различных премий, рейтинг университетов и результаты олимпиад.

ER - model


Родился
(0, inf)
Участники IMO
id
Name
Country
Medal
Year
(1, 1)
Город рождения
Город смерти
Страна рождения
Страна смерти
Название организации
Местоположение организации
Предмет
Нобелевский Лауреат
Дата рождения
Год
Имя
id
Дата смерти
(0, +inf)
(0, +inf)
id
год получения phd
Страна нынешнего проживания
Предыдущая страна
Наличие phd
Место получения phd
Мигрировал ли
Научные сотрудники
(1, 1)
Мигрируют
(1, +inf)
Название
Место в мировом рейтинге
Место в рейтинге качества обучения
Место в рейтинге доверия работодателей
Количество публикаций
Влияние
Количество цитирований
Количество патентов
Обучался
(1, 1)
Родился
(1, 1)
Университет
(0, +inf)
Находится
(1, 1)
Страна
(1, +inf)
Обучался
(0, +inf)
родился
(0, +inf)
Абелевский Лауреат
id
Имя
Год
Дата рождения
Дата смерти
Город рождения
Город смерти
Страна рождения
Страна смерти
Название организации
Местоположение организации
(1, 1)
(1, 1)
Родился
Название
Индекс человеческого потенциала
ВВП
ВВП на душу населения
(1, 1)
(0, +inf)
Обучался
(0, +inf)
Филдсовский лауреат
id
Имя
Год
Дата рождения
Местоположение организации
Название организации
Страна смерти
Страна рождения
Город смерти
Город рождения
Дата смерти


(0, 1)
(0, n)
(0, 1)
(0, 1)
(0, n)
(0, n)
(1, 1)
(0, n)
Nobel_Laureates
id serial pk
Birth_country varchar fk
Name varchar
Year int
Subject varchar
Birth_date Date
Death_date Date
Birth_city varchar
Death_city varchar
Death_country varchar
Organization_name varchar fk
Organization_country varchar
Universities
Name varchar pk
country_name varchar fk
world_rank int
national rank int
quality_of_education int
alumni_employment int
quality_of_faculty int
publications int
influence int
citations int
patents int
Abel_Laureates
id serial pk
Birth_country varchar fk
Name varchar
Year int
Birth_date Date
Death_date Date
Birth_city varchar
Death_city varchar
Death_country varchar
Organization_name varchar fk
Organization_country varchar
IMO_participants
Id serial pk
Name varchar
country varchar fk
medal varchar
Year int
(1, 1)
(0, n)
(1, 1)
(1, n)
(0, n)
(1, 1)
Scientific_researchers
id serial pk
phd_year int
current_country varchar fk
previous_county varchar
has_phd bool
phd_country varchar
has_migrated bool
Countries
Name varchar pk
Population int
Human_development_index float
GDP per capita float
(0, n)
(1, 1)
(0, n)
(1, 1)
Field_Laureates
id serial pk
Birth_country varchar fk
Name varchar
Year int
Birth_date Date
Death_date Date
Birth_city varchar
Death_city varchar
Death_country varchar
Organization_name varchar fk
Organization_country varchar

TR - model

```sql
select organization_country, count(*) as h
from (select organization_country, organization_name, count(full_name) as kek
      from nobel_laureates
      where organization_name is not null
      group by organization_name, organization_country
      order by kek desc
     ) as t
group by organization_country
order by h desc
```

	organization_country	h
1	United_States_of_America	114
2	United_Kingdom	39
3	Germany	30
4	France	24
5	Federal_Republic_of_Germany	21
6	Japan	14
7	Switzerland	10
8	Sweden	8
9	Netherlands	6
10	Italy	6
11	Denmark	6
12	Canada	5
13	Belgium	4
14	Australia	4
15	Union_of_Soviet_Socialist_Republics	4
16	Austria	4
17	Russia	3

SELECT - запрос, показывающий количество университетов в стране, в которой есть нобелевский лауреат

```sql
select earliest_country as country,
       round(cast(number_of_migrated_from as decimal) /
            (select count(*) from scientific_researchers where earliest_country is not null), 4) as number_of_migrated_from,
       round(cast(number_of_migrated_to as decimal) /
            (select count(*) from scientific_researchers where country_2016 is not null), 4) as number_of_migrated_to
from (select earliest_country, count(earliest_country) as number_of_migrated_from
      from scientific_researchers
      group by earliest_country
      order by number_of_migrated_from desc) t
        inner join
    (select country_2016, count(country_2016) as number_of_migrated_to
      from scientific_researchers
      group by country_2016
      order by number_of_migrated_to desc) s on earliest_country = country_2016
```

	country	number_of_migrated_from	number_of_migrated_to
1	US	0.1591	0.1823
2	BR	0.0671	0.0669
3	IN	0.0657	0.0499
4	CN	0.0642	0.0501
5	GB	0.0616	0.0655
6	ES	0.0452	0.0449
7	IT	0.0433	0.0411
8	RU	0.0323	0.0283
9	PT	0.0301	0.0303
10	AU	0.0288	0.037
11	CA	0.0221	0.0219
12	FR	0.0207	0.0171

SELECT - запрос, показывающий количество уехавших и приехавших в страну, деленное на количество записей в таблице, в которых соответственно кто-то уехал или приехал.

```sql
select *
from (select full_name, organization_name, world_rank
      from ((select full_name, organization_name from nobel_laureates)
            union
            (select laureate_name as full_name, orgaization_name from field_laureates)
            union
            (select laureate_name as full_name, educated_at as organization_name from abel_laureates)) t
               inner join universities on
           t.organization_name = university_name) y
order by world_rank desc
limit 1
```

	full_name	organization_name	world_rank
1	Efim_Zelmanov	Novosibirsk_State_University	985

SELECT - запрос, показывающий лауреата Абелевской/Нобелевской/Филдсовской награды, который обучался в самом неприступном вузе.

```sql
(select hdi_rank, laureate_name, country
 from ((select * from countries) x
         inner join
         (select * from field_laureates) y on x.country_name = y.country)
 order by hdi_rank desc
 limit 1)
union

(select hdi_rank, laureate_name, country_of_citizenship
 from ((select * from countries) a
         inner join
         (select * from abel_laureates) b on a.country_name = b.country_of_citizenship)
 order by hdi_rank desc
 limit 1)

union

(select hdi_rank, full_name, birth_country
 from ((select * from countries) x
         inner join
         (select * from nobel_laureates) y on x.country_name = y.birth_country)
 order by hdi_rank desc
 limit 1)
```

	hdi_rank	laureate_name	country
1	177	Ellen_Johnson_Sirleaf	Liberia
2	130	S._R._Srinivasa_Varadhan	India
3	75	Artur_Avila	Brazil

SELECT - запрос, показывающий соответственно Филдсовского, Абелевского, Нобелевского лауреата, который родился в наиболее неразвитой стране

```sql
CREATE OR REPLACE FUNCTION Points(gold_medals bigint, silver_medals bigint, bronze_medals bigint)
    RETURNS bigint AS
$$
BEGIN
    return 3 * gold_medals + 2 * silver_medals + bronze_medals;
END;
$$ LANGUAGE plpgsql;

select x.imo_year,
       x.country,
       Points(x.number_of_gold_medals, x.number_of_silver_medals, x.number_of_bronze_medals) as Points
from (select a.imo_year, a.country, number_of_bronze_medals, number_of_silver_medals, number_of_gold_medals
      from (select imo_year, country, count(award) as number_of_gold_medals
            from imo_participants
            where award = 'Gold_medal'
            group by imo_year, country) a
               inner join
           (select imo_year, country, count(award) as number_of_silver_medals
            from imo_participants
            where award = 'Silver_medal'
            group by imo_year, country) b
           on (a.imo_year = b.imo_year and a.country = b.country)
               inner join
           (select imo_year, country, count(award) as number_of_bronze_medals
            from imo_participants
            where award = 'Bronze_medal'
            group by imo_year, country) d on (d.imo_year = a.imo_year and d.country = a.country)) x
order by Points Desc
```

	imo_year	country	points
1	2017	VNM	15
2	1993	RUS	15
3	2004	RUS	15
4	2011	SGP	15
5	2014	JPN	15
6	2016	RUS	15
7	2005	ROU	15
8	1997	RUS	14
9	2000	VNM	14
10	1994	RUS	14
11	1989	USS	14
12	1996	HUN	14
13	2014	NLD	14
14	1994	BGR	14
15	1991	ROU	14
16	2000	TWN	14
17	2012	IRN	14
18	1998	TWN	14
19	2009	PRK	14
20	2014	SGP	14
21	2011	TUR	14
22	2002	BGR	14
23	2005	TWN	14

SELECT - запрос, с помощью которого была введена собственная функция отображающая успешность на IMO, построена таблица команд с баллами в каждый год

```sql
CREATE OR REPLACE FUNCTION Rating(alumni_employment integer, quality_of_faculty integer, publications integer,
                                  influence integer, citations integer, patents integer)
    RETURNS integer AS
$$
BEGIN
    return 3 * alumni_employment + 7 * quality_of_faculty + 1 * publications + 1 * influence + 1 * citations +
           2 * patents;
END;
$$ LANGUAGE plpgsql;

select university_name
from (select *
      from universities
      order by Rating(alumni_employment, quality_of_faculty, publications, influence, citations, patents) ASC) x
```

	university_name
1	Harvard_University
2	Harvard_University
3	Stanford_University
4	Massachusetts_Institute_of_Technology
5	University_of_Cambridge
6	University_of_California,_Berkeley
7	Columbia_University
8	University_of_Oxford
9	University_of_Chicago
10	University_of_California,_Los_Angeles
11	Princeton_University

SELECT - запрос с помощью которого, используя места в мире в определенных категориях для университетов, можно сделать свою функцию для определения рейтинга

```sql
create or replace
    temp view Universities_in_the_first_thousand_from_russia(rank, name) as
    select world_rank as rank, university_name as name from universities
    where country_name = 'Russia';
select *
from Universities_in_the_first_thousand_from_russia
```

	rank	name
1	59	Lomonosov_Moscow_State_University
2	250	Moscow_Institute_of_Physics_and_Technology
3	406	Saint_Petersburg_State_University
4	755	National_Research_Nuclear_University_MEPhI
5	985	Novosibirsk_State_University

VIEW - представление, показывающее российские университеты в первой тысяче лучших.

```sql
create or replace
    temp view nobel_prize_in_physics(name, year) as
select full_name as name, year
from nobel_laureates
where category = 'Physics'
order by year desc;
select *
from nobel_prize_in_physics
```

	name	year
1	F._Duncan_M._Haldane	2016
2	David_J._Thouless	2016
3	J._Michael_Kosterlitz	2016
4	Takaaki_Kajita	2015
5	Arthur_B._McDonald	2015
6	Shuji_Nakamura	2014
7	Isamu_Akasaki	2014
8	Hiroshi_Amano	2014
9	Peter_W._Higgs	2013
10	François_Englert	2013
11	Serge_Haroche	2012

VIEW - представление, показывающее Нобелевских лауреатов по физике в порядке, обратном хронологическому

```sql
create or replace
    temp view top_ten_countries as
select *
from countries
where hdi_rank <= 10
order by hdi_rank;
select *
from top_ten_countries
```

	country_name	population	gdp_per_capita	hdi_rank
1	Norway	4610820	37800	1
2	Australia	20264082	29000	2
3	Switzerland	7523934	32700	3
4	Denmark	5450661	31100	4
5	Netherlands	16491461	28600	5
6	Ireland	4062235	29600	6
7	Germany	82422299	27600	6
8	United_States	298444215	37800	8
9	New_Zealand	4076140	21600	9
10	Canada	33098932	29800	9

VIEW - представление, показывающее первую 10 стран по индексу человеческого развития.

```sql
create or replace
    temp view russian_laureates(name, prize) as
    select laureate_name as name, 'Abel prize' as prize
    from abel_laureates
    where country_of_citizenship = 'Russia'

    union

    select full_name as name, 'Nobel prize' as prize
    from nobel_laureates
    where birth_country = 'Russia'

    union

    select laureate_name as name, 'Field prize' as prize
    from field_laureates
    where country = 'Russia';
select *
from russian_laureates
```

	name	prize
1	Boris_Leonidovich_Pasternak	Nobel prize
2	Pavel_Alekseyevich_Cherenkov	Nobel prize
3	Grigory_Margulis	Field prize
4	Nikolay_Nikolaevich_Semenov	Nobel prize
5	Paul_Karrer	Nobel prize
6	Ivan_Alekseyevich_Bunin	Nobel prize
7	Mikhail_Aleksandrovich_Sholokhov	Nobel prize
8	Vitaly_L._Ginzburg	Nobel prize
9	Yakov_Sinai	Abel prize
10	Vladimir_Voevodsky	Field prize
11	Ilya_Prigogine	Nobel prize
12	Stanislav_Smirnov	Field prize
13	Ivan_Petrovich_Pavlov	Nobel prize
14	Aleksandr_Isayevich_Solzhenitsyn	Nobel prize
15	Grigori_Perelman	Field prize
16	Andre_Geim	Nobel prize
17	Konstantin_Novoselov	Nobel prize

VIEW - представление, показывающее лауреатов премий из России.

```sql
create or replace
    temp view number_of_laureates as
select coalesce(a.country, b.country, c.country) as country,
       (case when c1 is NULL then 0 else c1 end) + (case when c2 is NULL then 0 else c2 end) +
       (case when c3 is NULL then 0 else c3 end) as count
from ((select country, count(*) as c1
       from field_laureates
       group by country) a
        full join
     (select country_of_citizenship as country, count(*) as c2 from abel_laureates group by country_of_citizenship) b
     on a.country = b.country
        full join (select birth_country as country, count(*) as c3 from nobel_laureates group by country) c
                  on c.country = b.country)
order by count desc;
```

	country	count
1	United_States_of_America	276
2	United_Kingdom	96
3	France	65
4	Germany	61
5	Canada	33
6	Sweden	31
7	Russia	25
8	Japan	24

VIEW - представление, отображающее количество лауреатов премий по странам

```sql
create or replace
    temp view share_of_migrated as
select cast(_true as float) / _all as share_of_migrated
from ((select count(*) as _all
      from scientific_researchers) a
       cross join (select count(*) as _true from scientific_researchers where has_migrated = 'True') b)
```

	share_of_migrated
1	0.15208964264082375

VIEW - представление, показывающее долю уехавших из страны относительно участников опроса

	birth_country	number_of_laureates	rank
1	United_States_of_America	257	1
2	United_Kingdom	84	2
3	Germany	61	3
4	France	51	4
5	Sweden	29	5
6	Japan	24	6
7	Canada	18	8
8	Netherlands	18	8
9	Italy	17	10
10	Russia	17	10

```sql
select *, count(*) over (order by number_of_laureates desc) as rank
from (select distinct birth_country, count(*) as number_of_laureates
      from nobel_laureates
      where birth_country is not null
      group by birth_country
      order by number_of_laureates desc) x
```

SELECT - запрос, показывающий номер страны в рейтинге нобелевских лауреатов, использующий аналитическую функцию.

```sql
select *,
       sum(number_of_laureates) over (order by number_of_laureates desc) /
       (select count(*) from nobel_laureates) as share
from (select distinct birth_country, count(*) as number_of_laureates
      from nobel_laureates
      where birth_country is not null
      group by birth_country
      order by number_of_laureates desc) x
```

	birth_country	number_of_laureates	share
1	United_States_of_America	257	0.28460686600221483942
2	United_Kingdom	84	0.3776301218161683278
3	Germany	61	0.44518272425249169435
4	France	51	0.50166112956810631229
5	Sweden	29	0.53377630121816168328
6	Japan	24	0.56035437430786267996
7	Canada	18	0.60022148394241417497
8	Netherlands	18	0.60022148394241417497

SELECT - запрос, показывающий суммарную долю нобелевских лауреатов стран в порядке убывания количества лауреатов, использующий аналитическую функцию.

```sql
select *,
       sum(number_of_migrants) over (order by number_of_migrants desc) /
       (select count(*) from scientific_researchers) as share
from (select distinct earliest_country, count(*) as number_of_migrants
      from scientific_researchers
      where earliest_country is not null
      group by earliest_country
      order by number_of_migrants desc) x
```

	earliest_country	number_of_migrants	share
1	US	2123	0.12858734100545124 17
2	BR	903	0.18328285887341005451
3	IN	875	0.23628104179285281647
4	CN	847	0.28758328285887341005
5	GB	804	0.33628104179285281647
6	ES	646	0.37540884312537855845

SELECT - запрос, показывающий суммарную долю уехавших из стран, использующий аналитическую функцию.

	earliest_country	number_of_migrants	share
1	AO	1	0.0015142337976983646 3
2	ZM	1	0.0015142337976983646 3
3	VI	1	0.0015142337976983646 3
4	UZ	1	0.0015142337976983646 3
5	TJ	1	0.0015142337976983646 3
6	SV	1	0.0015142337976983646 3
7	SO	1	0.0015142337976983646 3
8	RE	1	0.0015142337976983646 3
9	PY	1	0.0015142337976983646 3

```
select *, sum(number_of_migrants) over (partition by number_of_migrants ) /
       (select count(*) from scientific_researchers) as share
from (select distinct earliest_country, count(*) as number_of_migrants
      from scientific_researchers where earliest_country is not null
      group by earliest_country
      order by number_of_migrants desc) x
```

SELECT - запрос, использующий аналитическую функцию, показывающий долю страны, с данным количеством нобелевских лауреатов.

```sql
CREATE OR REPLACE FUNCTION Points(gold_medals bigint, silver_medals bigint, bronze_medals bigint)
    RETURNS bigint AS
$$
BEGIN
    return 3 * gold_medals + 2 * silver_medals + bronze_medals;
END;
$$ LANGUAGE plpgsql;

select x.imo_year,
       x.country,
       Points(x.number_of_gold_medals, x.number_of_silver_medals, x.number_of_bronze_medals) as Points
from (select a.imo_year, a.country, number_of_bronze_medals, number_of_silver_medals, number_of_gold_medals
      from (select imo_year, country, count(award) as number_of_gold_medals
            from imo_participants
            where award = 'Gold_medal'
            group by imo_year, country) a
               inner join
           (select imo_year, country, count(award) as number_of_silver_medals
            from imo_participants
            where award = 'Silver_medal'
            group by imo_year, country) b
           on (a.imo_year = b.imo_year and a.country = b.country)
               inner join
           (select imo_year, country, count(award) as number_of_bronze_medals
            from imo_participants
            where award = 'Bronze_medal'
            group by imo_year, country) d on (d.imo_year = a.imo_year and d.country = a.country)) x
order by Points Desc
```

	imo_year	country	points
1	2017	VNM	15
2	1993	RUS	15
3	2004	RUS	15
4	2011	SGP	15
5	2014	JPN	15
6	2016	RUS	15
7	2005	ROU	15
8	1997	RUS	14
9	2000	VNM	14
10	1994	RUS	14
11	1989	USS	14
12	1996	HUN	14
13	2014	NLD	14
14	1994	BGR	14
15	1991	ROU	14
16	2000	TWN	14
17	2012	IRN	14
18	1998	TWN	14
19	2009	PRK	14
20	2014	SGP	14
21	2011	TUR	14
22	2002	BGR	14
23	2005	TWN	14

Хранимая процедура, показывающая количество очков страны на IMO.

```sql
create or replace function is_not_zero(number integer)
    returns integer as
$$
begin

    return case number
                when 0 then 0
                else 1
           end;

end;
$$ language plpgsql;


select (is_not_zero(0))
union
select (is_not_zero(1))
```

	is_not_zero
1	0
2	1

Хранимая процедура, возвращающая единицу, если число ненулевое и 0 в ином случае

```
create or replace function new_country_update() returns trigger as
$$
begin
    if (tg_op = 'UPDATE') then
        update nobel_laureates set birth_country = new.country_name where birth_country = old.country_name;
        update field_laureates set country = new.country_name where country = old.country_name;
        update abel_laureates set country_of_citizenship = new.country_name where country_of_citizenship = old.country_name;
    end if;
end ;
$$ language plpgsql;

CREATE TRIGGER country_update
    before update
    on countries
    for each row
    when (old.country_name is distinct from new.country_name)
execute procedure new_country_update();
```

Триггер, обновляющий информацию о странах в таблице с лауреатами

```
QUERY PLAN
                  Sort Method: top-N heapsort  Memory: 25kB
                  ->  Nested Loop  (cost=10000000000.16..10000003778.42 rows=880 width=38) (actual time=0.013..6.288 rows=495 lo
                        ->  Seq Scan on nobel_laureates  (cost=10000000000.00..10000000029.03 rows=903 width=34) (actual time=0.
                        ->  Bitmap Heap Scan on countries countries_2  (cost=0.16..4.17 rows=1 width=13) (actual time=0.004..0.0
                              Recheck Cond: ((country_name)::text = (nobel_laureates.birth_country)::text)
                              Heap Blocks: exact=495
                              ->  Bitmap Index Scan on countries_pkey  (cost=0.00..0.16 rows=1 width=0) (actual time=0.002..0.00
                                    Index Cond: ((country_name)::text = (nobel_laureates.birth_country)::text)
Planning Time: 3.118 ms
Execution Time: 8.561 ms
```

```
QUERY PLAN
            Sort Key: countries_2.hdi_rank DESC
            Sort Method: top-N heapsort  Memory: 25kB
          ->  Hash Join  (cost=5.87..37.33 rows=880 width=38) (actual time=0.072..0.662 rows=495 loops=1)
                Hash Cond: ((nobel_laureates.birth_country)::text = (countries_2.country_name)::text)
                ->  Seq Scan on nobel_laureates  (cost=0.00..29.03 rows=903 width=34) (actual time=0.012..0.235 rows=903 loops=1)
                ->  Hash  (cost=3.72..3.72 rows=172 width=13) (actual time=0.048..0.048 rows=172 loops=1)
                      Buckets: 1024  Batches: 1  Memory Usage: 16kB
                      ->  Seq Scan on countries countries_2  (cost=0.00..3.72 rows=172 width=13) (actual time=0.007..0.026 rows=172 loops=1)
Planning Time: 0.850 ms
Execution Time: 1.410 ms
```

Оптимизированный запрос, после добавления индексов(и каких-то рандомных on-ов и off-ов различных scan-ов)

Источники и актуальность данных

- Рейтинг университетов и информация про них - https://www.kaggle.com/mylesoneill/world-university-rankings#cwurData.csv - 2012

- Страны, их население, ВВП на душу населения - https://www.kaggle.com/fernandol/countries-of-the-world - 2018

- Индекс человеческого потенциала - https://www.kaggle.com/undp/human-development - 2015

- Нобелевские лауреаты - https://www.kaggle.com/nobelfoundation/nobel-laureates - 2016

- Абелевские лауераты - https://ru.wikipedia.org/wiki/Абелевская_премия - 2019

Вывод

Была построена база данных, по которой можно судить о некоторых характеристиках образования в различных странах, использовать информацию о лауреатах различных премий и тому подобное.
Я научился взаимодействовать с данными, обрабатывать их (например, с помощью питона), получать различную информацию средствами SQL.
Данная база данных может быть полезна для обработки информации о различных лауреатах премий, для обработки информации о вузах.