Санкт-Петербургский государственный университет
Факультет прикладной математики – процессов управления

Людкевич Николай Сергеевич

Курсовая работа

База данных клиентов и услуг туристического оператора

Проектирование баз данных

2020
Санкт-Петербург

ОПИСАНИЕ СХЕМЫ БАЗЫ ДАННЫХ

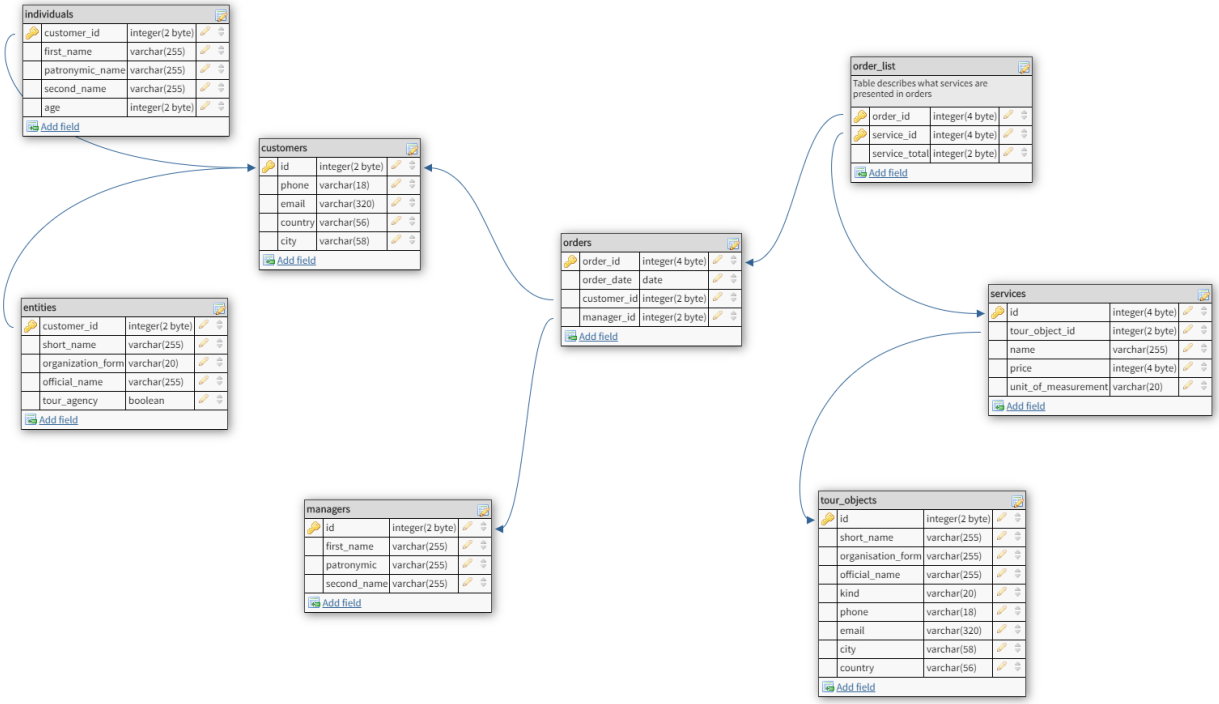

Рисунок 1. Схема базы данных.

В этой работе представлена упрощенная версия базы данных клиентов и услуг для туристического оператора. Она состоит из 8 таблиц: *customers*, *entities*, *individuals*, *managers*, *tour_objects* (отношение 1:1), *services*, *orders* (отношение 1:m) и *order_list* (отношение m:m).

Таблица *customers* хранит в себе данные всех клиентов: идентификатор *id* (первичный ключ), номера телефонов *phones*, адреса электронной почты *email*, города *city* и страны *countries*. Так как клиентами могут быть как физические, так и юридические лица, было принято решение присваивать первым положительные индексы, а вторым — неположительные, для того, и хранить данные, специфические для этих двух типов клиентов в отдельных таблицах.

Таблица *entities* хранит в себе специфические данные юридических лиц: короткое название *short_name*, под которым оно может быть широко известно, организационная форма *organization_form*, полное название *official_name*, а так же является ли юридическое лицо туристическим агентством *tour_agency* (это может быть важно при подсчете комиссии и составлении договоров).

Таблица *individuals* хранит в себе специфические данные юридических лиц: имя *first_name*, отчество *patronymic_name*, фамилия *second_name*, и возраст *age* (ограничение на возраст 18+).

Поля *customer_id* таблиц *entities* и *individuals* являются как первичными, так и внешними ключами и ссылаются на первичный ключ таблицы *customers*.

Таблица *managers* хранит в себе данные менеджеров: имя *first_name*, отчество *patronymic_name*, фамилия *second_name*. Первичный ключ: идентификатор *id*.

Таблица *tour_objects* хранит в себе данные туристических объектов, и также являются юридическими лицами. Соответственно, она содержит идентификатор *id* (первичный ключ), короткое название *short_name*, под которым оно может быть широко известно, организационная форма *organization_form*, полное название *official_name*, номера телефонов *phones*, адреса электронной почты *email*, города *city* и страны *countries*, а также поле типа объекта *kind*.

Таблица *services* хранит информацию услугах, предоставляемых туристическими объектами: идентификатор *id* (первичный ключ), идентификатор туристического объекта *tour_object_id* (внешний ключ, ссылающийся на первичный ключ таблицы *tour_objects*), наименование услуги *name*, цена услуги *price* за единицу *unit_of_measurements*.

Таблица *orders* хранит информацию об оформленных заказах: номер заказа *id* (первичный ключ), дату заказа *order_date*, идентификатор клиента *customer_id* (внешний ключ, ссылающийся на первичный ключ таблицы *customers*) и идентификатор менеджера *manager_id* (внешний ключ, ссылающийся на первичный ключ таблицы *managers*).

Таблица *order_list* хранит список услуг туристических объектов для каждого заказа. Она содержит номер заказа *order_id* (внешний ключ, ссылающийся на первичный ключ таблицы *orders*), идентификатор услуги *service_id* (внешний ключ, ссылающийся на первичный ключ таблицы *services*) и количество услуг в рамках заказа *service_total*. В данной таблице первичным ключом является комбинация столбцов *order_id* и *service_id*.

Схема базы данных представлена на рисунке 1. Она была составлена с помощью онлайн-сервиса app.dbdesigner.net. Дамп базы данных можно найти в моем репозитории на GitHub: https://github.com/jambinoid/tour_agency_database.

Замечание. База данных предоставляет собой демонстрацию полученных знаний в рамках курса «Проектирование баз данных». В реальной базе туристического оператора должно присутствовать куда больше таблиц и полей. Так, у физических лиц и юридических лиц есть свои уникальные идентификаторы: паспортные данные и ИНН. Они необходимы для составления договора, но не внесены в данную базу данных для того, чтобы не засорять примеры плохо усваиваемыми цифровыми идентификаторами. Кроме того, объекты так же могут быть физическими лицами (например, гиды), поэтому имеет смысл реализовывать хранение их данных аналогичным способом, что и данные клиентов, и т.д.

ЛЕГКИЕ ЗАПРОСЫ

Ниже приведены примеры легких запросов для представленной базы данных.

1. Отсортировать все услуги по возрастанию цены:

```
SELECT *
FROM services
ORDER BY price;
```

Вывод:

```
 id | tour_object_id |                name                | price | unit_of_measurements
----+----------------+------------------------------------+-------+----------------------
  4 |              3 | Экскурсия для ребенка              |     0 | Штука
  1 |              1 | Экскурсия для взрослого            |   200 | Штука
  6 |              5 | Входной билет                      |   200 | Штука
  7 |              5 | Экскурсия                          |   300 | Штука
  5 |              4 | Обед                               |   350 | Штука
  3 |              3 | Экскурсия для взрослого            |   400 | Штука
 10 |              7 | Ужин                               |   450 | Штука
 15 |             10 | Завтрак                            |   500 | Штука
  8 |              6 | Входной билет в усадьбу            |   500 | Штука
  9 |              6 | Входной билет в дом                |   500 | Штука
  2 |              2 | Транспортное обслуживание по городу |  1300 | Час
 14 |             10 | Уборка                             |  1600 | Штука
 13 |             10 | Стандартный номер на двоих         |  2000 | Ночь
 12 |              9 | Услуги гида                        |  4000 | День
 11 |              8 | Перевозка СПб-Тула-СПб             |  5000 | Штука
(15 строк)
```

Анализ:

```
                                                 QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Sort  (cost=15.86..16.19 rows=130 width=584) (actual time=0.044..0.044 rows=15 loops=1)
   Sort Key: price
   Sort Method: quicksort  Memory: 26kB
   ->  Seq Scan on services  (cost=0.00..11.30 rows=130 width=584) (actual time=0.028..0.030 rows=15 loops=1)
 Planning Time: 0.245 ms
 Execution Time: 0.070 ms
(6 строк)
```

Создание индекса по полю, по которому происходит сортировка:

```
CREATE INDEX idx_service_price ON services(price);
```

Анализ оптимизации:

```
                                                 QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------
 Index Scan using idx_service_price on services  (cost=0.14..12.36 rows=15 width=584) (actual time=0.034..0.040
rows=15 loops=1)
 Planning Time: 0.500 ms
 Execution Time: 0.060 ms
(3 строки)
```

2. Выбрать все туристические агентства из заказчиков-юридических лиц:

```
SELECT *
FROM entities
WHERE tour_agency IS TRUE;
```

Вывод:

```
 customer_id |     short_name     | organisation_form |     official_name      | tour_agency
-------------+--------------------+-------------------+------------------------+-------------
           0 | Дискавери          | ООО               | Дискавери              | t
          -3 | Турагенство Радуга | ИП                | Соловьев Петр Петрович | t
(2 строки)
```

Анализ:

```
                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------
 Seq Scan on entities  (cost=0.00..1.04 rows=2 width=1551) (actual time=0.024..0.026 rows=2 loops=1)
   Filter: (tour_agency IS TRUE)
   Rows Removed by Filter: 2
 Planning Time: 0.107 ms
 Execution Time: 0.044 ms
(5 строк)
```

Создание индекса по полю, по которому происходит выборка:

```
CREATE INDEX idx_tour_agency ON entities(tour_agency) WHERE (tour_agency IS TRUE);
```

Анализ:

```
                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------
 Index Scan using idx_tour_agency on entities  (cost=0.13..12.16 rows=2 width=1551) (actual time=0.042..0.044 rows=2
loops=1)
 Planning Time: 0.526 ms
 Execution Time: 0.064 ms
(3 строки)
```

3. Вывести количество заказов по отсортированным датам:

```
SELECT order_date, COUNT(customer_id)
FROM orders
GROUP BY order_date
ORDER BY order_date;
```

Вывод:

```
 order_date | count
------------+-------
 2019-06-10 |     2
 2019-06-23 |     1
 2019-07-11 |     1
 2019-07-14 |     1
 2019-07-21 |     1
 2019-07-29 |     1
 2019-08-01 |     2
 2019-08-10 |     1
 2019-08-28 |     1
(9 строк)
```

Анализ:

```
                                                                 QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=50.24..50.74 rows=200 width=12) (actual time=0.061..0.062 rows=9 loops=1)
   Sort Key: order_date
   Sort Method: quicksort  Memory: 25kB
   ->  HashAggregate  (cost=40.60..42.60 rows=200 width=12) (actual time=0.046..0.049 rows=9 loops=1)
         Group Key: order_date
         ->  Seq Scan on orders  (cost=0.00..30.40 rows=2040 width=6) (actual time=0.029..0.031 rows=11 loops=1)
 Planning Time: 0.147 ms
 Execution Time: 0.124 ms
(8 строк)
```

Создание индекса по полю, по которому происходит сортировка:

```
CREATE INDEX idx_order_date ON orders(order_date);
```

Анализ:

```
                                                                 QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------
 GroupAggregate  (cost=0.14..12.46 rows=11 width=12) (actual time=0.053..0.070 rows=9 loops=1)
   Group Key: order_date
   ->  Index Scan using idx_order_date on orders  (cost=0.14..12.30 rows=11 width=6) (actual time=0.043..0.049
rows=11 loops=1)
 Planning Time: 0.650 ms
 Execution Time: 0.114 ms
(5 строк)
```

4. Вывести id клиентов, которые оформляли заказ более одного раза:

```
SELECT *
FROM
    (SELECT customer_id, COUNT(order_date) AS orders_num
    FROM orders
    GROUP BY customer_id) AS count_orders_by_customer
WHERE orders_num > 1;
```

Вывод:

```
customer_id | orders_num
------------+------------
           1 |          2
(1 строка)
```

Анализ:

```
                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------
 HashAggregate  (cost=1.19..1.33 rows=4 width=10) (actual time=0.055..0.060 rows=1 loops=1)
   Group Key: orders.customer_id
   Filter: (count(orders.order_date) > 1)
   Rows Removed by Filter: 9
   ->  Seq Scan on orders  (cost=0.00..1.11 rows=11 width=6) (actual time=0.030..0.034 rows=11 loops=1)
 Planning Time: 0.163 ms
 Execution Time: 0.105 ms
(7 строк)
```

Создание индекса по полю, по которому происходит агрегация:

```
CREATE INDEX idx_orders_num ON orders(order_date);
```

Анализ:

```
                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 GroupAggregate  (cost=10000000001.30..10000000001.55 rows=4 width=10) (actual time=0.044..0.047 rows=1 loops=1)
   Group Key: orders.customer_id
   Filter: (count(orders.order_date) > 1)
   Rows Removed by Filter: 9
   ->  Sort  (cost=10000000001.30..10000000001.33 rows=11 width=6) (actual time=0.034..0.034 rows=11 loops=1)
         Sort Key: orders.customer_id
         Sort Method: quicksort  Memory: 25kB
         ->  Seq Scan on orders  (cost=10000000000.00..10000000001.11 rows=11 width=6) (actual time=0.019..0.022
rows=11 loops=1)
 Planning Time: 0.600 ms
 Execution Time: 0.086 ms
(10 строк)
```

СРЕДНИЕ ЗАПРОСЫ

Ниже приведены примеры средних запросов для представленной базы данных.

1. Вывести информацию об услугах, предоставляемых музеями: краткое название объекта, город, название услуги и ее цену:

```
SELECT tour_objects.short_name, tour_objects.city, services.name, services.price
FROM tour_objects INNER JOIN services
ON tour_objects.id = services.tour_object_id
WHERE tour_objects.kind = 'Музей'
ORDER BY tour_objects.id;
```

Вывод:

```
               short_name                | city |          name           | price
-----------------------------------------+------+-------------------------+-------
 Музей "Тульский Кремль"                 | Тула | Экскурсия для взрослого |   200
 Тульский музей изобразительных искусств | Тула | Экскурсия для взрослого |   400
 Тульский музей изобразительных искусств | Тула | Экскурсия для ребенка   |     0
 Музей оружия                            | Тула | Входной билет           |   200
 Музей оружия                            | Тула | Экскурсия               |   300
 Ясная Поляна                            | Тула | Входной билет в усадьбу |   500
 Ясная Поляна                            | Тула | Входной билет в дом     |   500
(7 строк)
```

Анализ:

```
                                                        QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 Sort  (cost=11.59..11.59 rows=1 width=1172) (actual time=0.104..0.105 rows=7 loops=1)
   Sort Key: tour_objects.id
   Sort Method: quicksort  Memory: 26kB
   ->  Hash Join  (cost=10.39..11.58 rows=1 width=1172) (actual time=0.074..0.088 rows=7 loops=1)
         Hash Cond: (services.tour_object_id = tour_objects.id)
         ->  Seq Scan on services  (cost=0.00..1.15 rows=15 width=522) (actual time=0.026..0.030 rows=15 loops=1)
         ->  Hash  (cost=10.38..10.38 rows=1 width=652) (actual time=0.035..0.035 rows=4 loops=1)
               Buckets: 1024  Batches: 1  Memory Usage: 9kB
               ->  Seq Scan on tour_objects  (cost=0.00..10.38 rows=1 width=652) (actual time=0.019..0.027 rows=4
loops=1)
                     Filter: ((kind)::text = 'Музей'::text)
                     Rows Removed by Filter: 6
 Planning Time: 0.339 ms
 Execution Time: 0.162 ms
(13 строк)
```

Создание индекса по столбцам, по которым происходит сортировка и выборка:

```
CREATE INDEX idx_museums ON tour_objects(id, kind) WHERE (kind = 'Музей');
```

Анализ:

```
                                                        QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=10000000000.13..10000000009.48 rows=2 width=1172) (actual time=0.053..0.095 rows=7 loops=1)
   Join Filter: (tour_objects.id = services.tour_object_id)
   Rows Removed by Join Filter: 53
   ->  Index Scan using idx_museums on tour_objects  (cost=0.13..8.14 rows=1 width=652) (actual time=0.034..0.036
rows=4 loops=1)
   ->  Seq Scan on services  (cost=10000000000.00..10000000001.15 rows=15 width=522) (actual time=0.004..0.005
rows=15 loops=4)
 Planning Time: 0.658 ms
 Execution Time: 0.122 ms
(7 строк)
```

2. Вывести информацию о всех юр. лицах в базе: краткое название, организационную форму, официальное название, город, страну

```
SELECT short_name, organisation_form, official_name, city, country
FROM
    (SELECT entities.short_name, entities.organisation_form, entities.official_name,
            customers.city, customers.country
    FROM entities INNER JOIN customers
    ON customers.id = entities.customer_id) AS entity_customers
UNION
SELECT short_name, organisation_form, official_name, city, country
FROM tour_objects
ORDER BY short_name;
```

Вывод:

short_name	organisation_form	official_name	city	country
Белорусский Тракторный Завод	ОАО	Белорусский Тракторный Завод	Минск	Беларусь
Гостевой дом "Европейский"	ООО	Гостевой дом "Европейский"	Тула	Россия
Дискавери	ООО	Дискавери	Санкт-Петербург	Россия
Кафе "Гурман"	ИП	Борисова Тамара Павловна	Тула	Россия
Лицей №1973	ГБОУ	Лицей №1973 Центрального района Санкт-Петербурга	Санкт-Петербург	Россия
Музей "Тульский Кремль"	ГУК ТО	Объединение "Историко-краеведческий и художественный музей"	Тула	Россия
Музей оружия	ФГБУК	Тульский государственный музей оружия	Тула	Россия
Пассажирские перевозки	ООО	Пассажирские перевозки	Санкт-Петербург	Россия
Пассажирские перевозки	ООО	Пассажирские перевозки	Тула	Россия
Ресторан "Дворянская усадьба"	ОАО	Ресторан "Дворянская усадьба"	Тула	Россия
Тульский музей изобразительных искусств	ГУК ТО	Объединение "Историко-краеведческий и художественный музей"	Тула	Россия
Турагенство Радуга	ИП	Соловьев Петр Петрович	Тула	Россия
Экскурсии и гиды	ИП	Иванов Алексей Петрович	Тула	Россия
Ясная Поляна	ГМиПЗ	Музей-усадьба Л.Н.Толстого "Ясная поляна"	Тула	Россия

(14 строк)

Анализ:

```
                                                   QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 Sort  (cost=14.13..14.16 rows=14 width=1812) (actual time=0.221..0.222 rows=14 loops=1)
   Sort Key: entities.short_name
   Sort Method: quicksort  Memory: 28kB
   ->  HashAggregate  (cost=13.72..13.86 rows=14 width=1812) (actual time=0.116..0.123 rows=14 loops=1)
         Group Key: entities.short_name, entities.organisation_form, entities.official_name, customers.city,
customers.country
         ->  Append  (cost=1.09..13.55 rows=14 width=1812) (actual time=0.047..0.085 rows=14 loops=1)
               ->  Hash Join  (cost=1.09..12.24 rows=4 width=1812) (actual time=0.046..0.051 rows=4 loops=1)
                     Hash Cond: (customers.id = entities.customer_id)
                     ->  Seq Scan on customers  (cost=0.00..10.90 rows=90 width=266) (actual time=0.017..0.019
rows=10 loops=1)
                     ->  Hash  (cost=1.04..1.04 rows=4 width=1550) (actual time=0.019..0.019 rows=4 loops=1)
                           Buckets: 1024  Batches: 1  Memory Usage: 9kB
                           ->  Seq Scan on entities  (cost=0.00..1.04 rows=4 width=1550) (actual time=0.011..0.013
rows=4 loops=1)
               ->  Seq Scan on tour_objects  (cost=0.00..1.10 rows=10 width=1812) (actual time=0.024..0.030 rows=10
loops=1)
 Planning Time: 0.527 ms
 Execution Time: 0.282 ms
(15 строк)
```

Создание индекса по столбцу, по которому происходит сортировка:

```
CREATE INDEX idx_name ON tour_objects(short_name);
```

Анализ:

```
                                                                                              QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=10000000038.85..10000000038.89 rows=14 width=1812) (actual time=0.142..0.144 rows=14 loops=1)
   Sort Key: entities.short_name
   Sort Method: quicksort  Memory: 28kB
   ->  HashAggregate  (cost=10000000038.45..10000000038.59 rows=14 width=1812) (actual time=0.077..0.082 rows=14 loops=1)
         Group Key: entities.short_name, entities.organisation_form, entities.official_name, customers.city, customers.country
         ->  Append  (cost=0.27..10000000038.27 rows=14 width=1812) (actual time=0.020..0.056 rows=14 loops=1)
               ->  Nested Loop  (cost=0.27..36.96 rows=4 width=1812) (actual time=0.019..0.029 rows=4 loops=1)
                     ->  Index Scan using entities_pkey on entities  (cost=0.13..12.19 rows=4 width=1550) (actual time=0.009..0.011 rows=4 loops=1)
                     ->  Index Scan using customers_pkey on customers  (cost=0.14..6.16 rows=1 width=266) (actual time=0.002..0.002 rows=1 loops=4)
                           Index Cond: (id = entities.customer_id)
               ->  Seq Scan on tour_objects  (cost=10000000000.00..10000000001.10 rows=10 width=1812) (actual time=0.020..0.024 rows=10 loops=1)
 Planning Time: 0.896 ms
 Execution Time: 0.211 ms
(13 строк)
```

3. Вывести траты на каждый заказ: id заказа и его затраты на него:

```
SELECT order_list.order_id, SUM(services.price * order_list.service_total) AS price
FROM order_list
INNER JOIN services
ON order_list.service_id = services.id
GROUP BY order_list.order_id
ORDER BY order_list.order_id;
```

Вывод:

```
order_id | price
----------+-------
        1 |   200
        2 |   200
        3 | 11800
        4 |  1500
        5 |   400
        6 | 16450
        7 |   400
        8 | 27050
        9 |  6000
       10 |  4600
       11 | 10200
(11 строк)
```

Анализ:

```
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 Sort  (cost=64.17..64.67 rows=200 width=12) (actual time=0.101..0.102 rows=11 loops=1)
   Sort Key: order_list.order_id
   Sort Method: quicksort  Memory: 25kB
   ->  HashAggregate  (cost=54.53..56.53 rows=200 width=12) (actual time=0.087..0.091 rows=11 loops=1)
         Group Key: order_list.order_id
         ->  Hash Join  (cost=1.34..38.48 rows=2140 width=10) (actual time=0.056..0.072 rows=33 loops=1)
               Hash Cond: (order_list.service_id = services.id)
               ->  Seq Scan on order_list  (cost=0.00..31.40 rows=2140 width=10) (actual time=0.026..0.029 rows=33
loops=1)
               ->  Hash  (cost=1.15..1.15 rows=15 width=8) (actual time=0.023..0.023 rows=15 loops=1)
                     Buckets: 1024  Batches: 1  Memory Usage: 9kB
                     ->  Seq Scan on services  (cost=0.00..1.15 rows=15 width=8) (actual time=0.012..0.016 rows=15
loops=1)
 Planning Time: 0.331 ms
 Execution Time: 0.178 ms
(13 строк)
```

Создание индекса по столбцу, по которому происходит сортировка и агрегация:

```
CREATE INDEX idx_order_id ON order_list(order_id);
```

Анализ:

```
                                                     QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 GroupAggregate  (cost=26.11..26.77 rows=33 width=12) (actual time=0.104..0.117 rows=11 loops=1)
   Group Key: order_list.order_id
   ->  Sort  (cost=26.11..26.20 rows=33 width=10) (actual time=0.099..0.102 rows=33 loops=1)
         Sort Key: order_list.order_id
         Sort Method: quicksort  Memory: 26kB
         ->  Hash Join  (cost=12.69..25.28 rows=33 width=10) (actual time=0.065..0.086 rows=33 loops=1)
               Hash Cond: (order_list.service_id = services.id)
               ->  Index Scan using idx_order_id on order_list  (cost=0.14..12.63 rows=33 width=10) (actual time=0.036..0.045 rows=33 loops=1)
               ->  Hash  (cost=12.36..12.36 rows=15 width=8) (actual time=0.019..0.019 rows=15 loops=1)
                     Buckets: 1024  Batches: 1  Memory Usage: 9kB
                     ->  Index Scan using services_pkey on services  (cost=0.14..12.36 rows=15 width=8) (actual time=0.006..0.012 rows=15 loops=1)
 Planning Time: 0.613 ms
 Execution Time: 0.163 ms
(13 строк)
```

СЛОЖНЫЕ ЗАПРОСЫ

Ниже приведены примеры сложных запросов для представленной базы данных.

1. Найти самого продуктивного менеджера (критерий: количество предоставленных услуг на одного человека):

```
SELECT managers.second_name, managers.first_name, managers.patronymic_name,
       count_services_by_managers.service_total AS service_total_max
FROM
    (SELECT orders.manager_id, SUM(count_services_by_order.service_num) AS service_total
    FROM orders RIGHT JOIN
            (SELECT order_id, COUNT(service_id) AS service_num
            FROM order_list
            GROUP BY order_id) AS count_services_by_order
    ON orders.order_id = count_services_by_order.order_id
    GROUP BY orders.manager_id) AS count_services_by_managers
INNER JOIN managers
ON managers.id = count_services_by_managers.manager_id
    AND count_services_by_managers.service_total = (SELECT MAX(count_services_by_order.service_num) AS
service_max
        FROM orders RIGHT JOIN
                (SELECT order_id, COUNT(service_id) AS service_num
            FROM order_list
            GROUP BY order_id) AS count_services_by_order
            ON orders.order_id = count_services_by_order.order_id);
```

Вывод:

```
second_name | first_name | patronymic_name | service_total_max
------------+------------+-----------------+-------------------
 Карпатян    | Мария      | Сергеевна       |                 6
 Александров | Александр  | Александрович   |                 6
(2 строки)
```

2. Вывести информацию о всех иногородних клиентах (не из Тулы): полное имя или полное название, номер телефона, эл. почту, город, страну:

```
SELECT customers_names.name, customers.phone, customers.email, customers.city, customers.country
FROM
    (SELECT customer_id, CONCAT(second_name, ' ', first_name, ' ', patronymic_name) AS name
    FROM individuals
            UNION
    SELECT customer_id, CONCAT(organisation_form, ' ', official_name) AS name
    FROM entities) as customers_names
INNER JOIN customers
ON customers_names.customer_id = customers.id
    AND customers.city != 'Тула';
```

Вывод:

```
                    name                     |     phone     |            email             |      city       |  country
---------------------------------------------+---------------+------------------------------+-----------------+-----------
 ООО Дискавери                               | 79615555555   | discovery.tours@randommail.ru | Санкт-Петербург | Россия
 Иванова Марина Александовна                 | 79990033222   |                              | Москва          | Россия
 Буше Сергей Сергеевич                       | 79100002233   | serega_wot@randommail.ru     | Москва          | Россия
 Иванов Иван Иванович                        | 79103156567   | ivanov_ii@randommail.ru      | Казань          | Россия
 ГБОУ Лицей №1973 Центрального района Санкт-Петербурга | 79990444444   | litseum1973@randommail.ru    | Санкт-Петербург | Россия
 ОАО Белорусский Тракторный Завод            | 3759023213362 | tractor.zavod@randommail.com | Минск           | Беларусь
 Жалелов Адиль Скриптонитович                | 79330032131   | jalelov_adil@randommail.com  | Астана          | Казахстан
(7 строк)
```

3. Вывести информацию о заказах в июле 2019 года: дату, полное имя или полное название заказчика, полное имя менеджера, затраты на заказ:

```
SELECT orders.order_id, orders.order_date, customers_names.customer_name, managers_names.manager_name,
prices.price
FROM orders
INNER JOIN
    (SELECT customer_id, CONCAT(second_name, ' ', first_name, ' ', patronymic_name) AS customer_name
    FROM individuals
            UNION
    SELECT customer_id, CONCAT(organisation_form, ' ', official_name) AS customer_name
    FROM entities) as customers_names
ON orders.customer_id = customers_names.customer_id
INNER JOIN
    (SELECT id, CONCAT(second_name, ' ', first_name, ' ', patronymic_name) AS manager_name
    FROM managers) as managers_names
ON orders.manager_id = managers_names.id
INNER JOIN
    (SELECT order_list.order_id, SUM(services.price * order_list.service_total) AS price
    FROM order_list
    INNER JOIN services
    ON order_list.service_id = services.id
    GROUP BY order_list.order_id
    ORDER BY order_list.order_id) as prices
ON orders.order_id = prices.order_id
    AND orders.order_date BETWEEN '2019-07-01' AND '2019-07-31';
```

Вывод:

```
 order_id | order_date |          customer_name            |        manager_name         | price
----------+------------+-----------------------------------+-----------------------------+-------
        4 | 2019-07-11 | Иванов Иван Иванович              | Сомина Александра Олеговна |  1500
        5 | 2019-07-14 | Людкевич Сергей Вячеславович      | Лашхидзе Наталья Петровна  |   400
        6 | 2019-07-21 | ОАО Белорусский Тракторный Завод | Иванов Максим Иванович     | 16450
        7 | 2019-07-29 | Иванова Марина Александовна       | Артемов Артем Артемович    |   400
(4 строки)
```