S3. Контест: таблица результатов (SQL, 30 баллов)

ограничение по времени на тест: 15 секунд
ограничение по памяти на тест: 1024 мегабайта
ввод: стандартный ввод
вывод: стандартный вывод

Это необычная задача — вам надо написать SQL-запрос. В качестве решения вы должны отослать один запрос к базе данных, который возвращает требуемые данные. Запрос может содержать произвольное количество подзапросов, других конструкций, быть сколь угодно навороченным, но это должен быть один запрос (в нём не должна встречаться точка с запятой для разделения разных запросов).

При проверке вашего решения используется PostgreSQL 15.1. В качестве входных данных вам предоставляется дамп состояния базы данных. Обратите внимание, что время работы вашего решения на тесте включает восстановление состояния базы данных из дампа, но это время значительно меньше ограничения по времени. Вы можете использовать сервис http://sqlfiddle.com/ как инструмент для запуска запросов.

В этой задаче вам предстоит написать запрос к базе данных простейшей системы проведения соревнований по программированию. Вы прямо сейчас участвуете в подобном соревновании. Время почувствовать себя в роли разработчика системы для проведения таких соревнований!

Напишите запрос к базе данных, который построит таблицу результатов для соревнования с максимальным `id`.

Вывод должен включать всех пользователей, кто сделал хотя бы одну попытку в этом соревновании. Вывод должен включать 5 колонок:

- `rank` — место пользователя в контесте (пользователи с одинаковыми результатами делят место);
- `user_id` — `id` пользователя;
- `user_name` — `name` пользователя;
- `problem_count` — количество решённых в контесте задач (если одна задача решена многократно, то всё-равно учитывается как одна задача);
- `latest_successful_submitted_at` — время, когда была решена последняя из решённых задач у этого пользователя (если одна задача решена многократно, то задача считается решённой в момент первого решения), иными словами, последний момент времени, когда у пользователя увеличилось количество решённых задач.

Строки следует сортировать по невозрастанию `problem_count`, затем по неубыванию `latest_successful_submitted_at`, затем по возрастанию `user_id`.

Пользователи, которые решили одинаковое количество задач (имеют равные `problem_count`) и имеют равные значения `latest_successful_submitted_at`, должны поделить одно место. Обратите внимание, что если несколько пользователей делят места, то в нумерации мест образуется разрыв. Например, если первое место делят два пользователя, то следующий пользователь должен получить место 3 (то есть последовательность мест имеет вид: $1, 1, 3$).

Внимательно ознакомьтесь с примерами вывода. Ваш запрос должен иметь **в**

Route 256 [Middle]

Участник

→ О группе

Контест - Go (Middles)

Соревнование идет

00:58:20

→ Пересчёт ограничений по времени

Это соревнование использует политику пересчёта ограничений по времени по языкам программирования. Система автоматически увеличивает ограничения по времени для некоторых языков в соответствии с множителями. Независимо от множителя языка, ограничение по времени не может превысить 30 секунд. Прочтите детали по ссылке.

→ Языки

Следующие языки могут быть использованы как дополнительные для решения задач соревнования

Контест - Go (Middles):

- PostgreSQL 15.1

→ Материалы соревнования

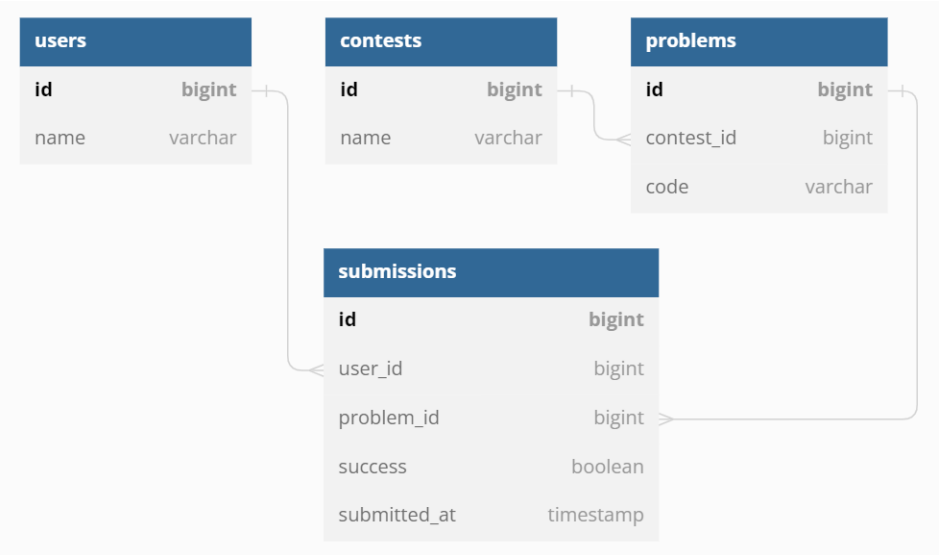
- problem-b-tests.zip
- problem-c-tests.zip
- problem-d-tests.zip

точности такой же вывод на примерах.

- `users` — пользователи системы (описываются двумя полями: `id` и `name`),
- `contests` — контесты в системе (описываются двумя полями: `id` и `name`),
- `problems` — задачи в системе, каждая задача принадлежит одному контесту (описываются тремя полями: `id`, `contest_id` и `code`, где `code` — это кодовое короткое название задачи),
- `submissions` — отосланные попытки решения задач, каждая попытка принадлежит одной задаче и одному пользователю (описываются 5 полями: `id`, `user_id`, `problem_id`, `success` и `submitted_at`, где `success` — это булевское значение была ли попытка успешной и `submitted_at` — дата-время, когда попытка была совершена).

Таким образом, `contests` и `problems` находятся в отношении «один ко многим», `submissions` и `users` находятся в отношении «многие к одному», `submissions` и `problems` находятся в отношении «многие к одному».

Изучите входные данные примера, чтобы подробно ознакомиться со схемой базы данных. Диаграмма ниже иллюстрирует схему базы данных.



Входные данные

Входными данными в этой задаче является дамп базы данных. Вам он может быть полезен для ознакомления с состоянием базы данных для конкретного теста. В качестве решения вы должны отправить один SQL-запрос.

Выходные данные

Ваш SQL-запрос должен вывести результаты соревнования с максимальным `id` в требуемом формате.

Внимательно ознакомьтесь с примерами вывода. Ваш запрос должен иметь **в точности** такой же вывод на примерах.

Примеры

входные данные

Скопировать

```
create table users (
  id bigint primary key,
  name varchar not null
);

create table contests (
  id bigint primary key,
  name varchar not null
);

create table problems (
  id bigint primary key,
  contest_id bigint,
```

- problem-e-tests.zip
- problem-f-tests.zip
- problem-g-tests.zip
- problem-h-tests.zip
- problem-i-tests.zip
- problem-s1-tests.zip
- problem-s2-tests.zip
- problem-s3-tests.zip

```
  code varchar not null,
  constraint fk_problems_contest_id foreign key (contest_id) references
contests (id)
);

create unique index on problems (contest_id, code);

create table submissions (
  id bigint primary key,
  user_id bigint,
  problem_id bigint,
  success boolean not null,
  submitted_at timestamp not null,
  constraint fk_submissions_user_id foreign key (user_id) references
users (id),
  constraint fk_submissions_problem_id foreign key (problem_id)
references problems (id)
);

insert into users
values (1, 'Marie Curie'),
       (2, 'Stephen Hawking'),
       (3, 'Ada Lovelace'),
       (4, 'Albert Einstein'),
       (5, 'Archimedes');

insert into contests
values (1, 'Sandbox-Juniors'),
       (2, 'Sandbox-Seniors'),
       (3, 'Contest-Juniors'),
       (4, 'Contest-Seniors');

insert into problems
values (1, 1, 'A'),
       (2, 2, 'A'),
       (3, 3, 'A'),
       (4, 3, 'B'),
       (5, 4, 'A'),
       (6, 4, 'B');

insert into submissions
values (1, 2, 2, false, '2023-02-05 11:01:00'),
       (2, 2, 2, true, '2023-02-05 11:02:00'),
       (3, 2, 6, true, '2023-02-05 11:03:01'),
       (4, 2, 1, true, '2023-02-05 11:04:00'),
       (5, 2, 1, true, '2023-02-05 11:05:00'),
       (6, 3, 6, true, '2023-02-05 11:06:00'),
       (17, 1, 6, true, '2023-02-05 11:03:00'),
       (8, 1, 2, true, '2023-02-05 11:08:00'),
       (9, 1, 1, false, '2023-02-05 11:09:00'),
       (10, 3, 1, false, '2023-02-05 11:10:00'),
       (11, 5, 5, false, '2023-02-05 11:11:00'),
       (13, 2, 6, true, '2023-02-05 11:03:00'),
       (14, 3, 6, false, '2023-02-05 11:05:59'),
       (15, 1, 6, true, '2023-02-05 11:04:00');
```

выходные данные Скопировать

```
 rank | user_id |    user_name    | problem_count |
latest_successful_submitted_at
------+---------+-----------------+---------------+---------------------
----------
    1 |       1 | Marie Curie     |             1 | 2023-02-05 11:03:00
    1 |       2 | Stephen Hawking |             1 | 2023-02-05 11:03:00
    3 |       3 | Ada Lovelace    |             1 | 2023-02-05 11:06:00
    4 |       5 | Archimedes      |             0 |
(4 rows)
```

входные данные Скопировать

```
create table users (
  id bigint primary key,
  name varchar not null
);

create table contests (
  id bigint primary key,
  name varchar not null
);

create table problems (
```

```
  id bigint primary key,
  contest_id bigint,
  code varchar not null,
  constraint fk_problems_contest_id foreign key (contest_id) references
contests (id)
);

create unique index on problems (contest_id, code);

create table submissions (
  id bigint primary key,
  user_id bigint,
  problem_id bigint,
  success boolean not null,
  submitted_at timestamp not null,
  constraint fk_submissions_user_id foreign key (user_id) references
users (id),
  constraint fk_submissions_problem_id foreign key (problem_id)
references problems (id)
);

insert into users
values (1, 'Olivia'),
       (2, 'Henry'),
       (3, 'Lucas'),
       (4, 'John'),
       (5, 'Charlotte'),
       (6, 'Henry');

insert into contests
values (3, 'Main'),
       (1, 'Practice');

insert into problems
values (1, 3, 'A'),
       (2, 3, 'B'),
       (3, 1, 'A');

insert into submissions
values (10, 3, 2, false, '2023-02-05 11:05:12'),
       (20, 3, 2, true, '2023-02-05 11:07:49'),
       (30, 3, 2, true, '2023-02-05 11:07:49'),
       (40, 3, 1, false, '2023-02-05 11:01:32'),
       (50, 3, 1, false, '2023-02-05 11:11:46'),
       (60, 3, 1, false, '2023-02-05 11:27:05'),
       (70, 6, 2, false, '2023-02-05 11:04:00'),
       (80, 6, 2, true, '2023-02-05 11:05:00'),
       (90, 6, 2, false, '2023-02-05 11:06:00'),
       (100, 6, 2, true, '2023-02-05 11:07:00'),
       (110, 6, 1, false, '2023-02-05 11:08:00'),
       (120, 6, 1, true, '2023-02-05 11:09:00'),
       (130, 2, 2, false, '2023-02-05 11:00:01'),
       (150, 5, 1, false, '2023-02-05 11:07:48'),
       (160, 5, 1, true, '2023-02-05 11:07:49'),
       (170, 5, 1, true, '2023-02-05 11:07:50'),
       (180, 1, 3, false, '2023-02-04 15:00:01'),
       (190, 1, 3, true, '2023-02-04 15:00:01'),
       (200, 5, 3, true, '2023-02-04 15:00:01'),
       (210, 5, 3, false, '2023-02-04 15:00:01'),
       (220, 2, 3, false, '2023-02-04 15:00:01'),
       (230, 6, 3, false, '2023-02-04 15:00:01'),
       (240, 6, 3, false, '2023-02-04 15:00:01');
```

выходные данные Скопировать

```
 rank | user_id | user_name | problem_count |
latest_successful_submitted_at
------+---------+-----------+---------------+---------------------------
----
    1 |       6 | Henry     |             2 | 2023-02-05 11:09:00
    2 |       3 | Lucas     |             1 | 2023-02-05 11:07:49
    2 |       5 | Charlotte |             1 | 2023-02-05 11:07:49
    4 |       2 | Henry     |             0 |
(4 rows)
```

Время на сервере: 05.02.2023 16:06:27 (h1).
Десктопная версия, переключиться на мобильную.
Privacy Policy

На платформе