Исследование базы данных StackOverflow

В нашем распоряжении база данных StackOverflow — сервиса вопросов и ответов о программировании, которая хранит данные о постах за 2008 год. StackOverflow похож на социальную сеть — пользователи сервиса задают вопросы, отвечают на посты, оставляют комментарии и ставят оценки другим ответам. В таблицах также есть информация и о более поздних оценках, которые эти посты получили.

Цель проекта: проанализировать данные сервиса StackOverflow с помощью различных SQL-запросов к базе данных. Продемонстрировать изученные операторы, функции и методы работы с базой данных.

Описание данных:

Познакомимся с данными, которые хранят таблицы:

1. Stackoverflow.badges

Хранит информацию о значках, которые присуждаются за разные достижения. Например, пользователь, правильно ответивший на большое количество вопросов про PostgreSQL, может получить значок postgresql.

Поле	Описание
`id`	Идентификатор значка, первичный ключ таблицы
`name`	Название значка
`user_id`	Идентификатор пользователя, которому присвоили значок, внешний ключ, отсылающий к таблице `users`
`creation_date`	Дата присвоения значка

2. Stackoverflow.post_types

Содержит информацию о типе постов. Их может быть два:

- `Question` — пост с вопросом;
- `Answer` — пост с ответом.

Поле	Описание
`id`	Идентификатор поста, первичный ключ таблицы
`type`	Тип поста

3. Stackoverflow.posts

Содержит информацию о постах:

Поле	Описание
`id`	Идентификатор поста, первичный ключ таблицы
`title`	Заголовок поста
`creation_date`	Дата создания поста
`favorites_count`	Число, которое показывает, сколько раз пост добавили в «Закладки»
`last_activity_date`	Дата последнего действия в посте, например комментария

Поле	Описание
last_edit_date	Дата последнего изменения поста
user_id	Идентификатор пользователя, который создал пост, внешний ключ к таблице users
parent_id	Если пост написали в ответ на другую публикацию, в это поле попадёт идентификатор поста с вопросом
post_type_id	Идентификатор типа поста, внешний ключ к таблице post_types
score	Количество очков, которое набрал пост
views_count	Количество просмотров

4. Stackoverflow.users

Содержит информацию о пользователях.

Поле	Описание
id	Идентификатор пользователя, первичный ключ таблицы
creation_date	Дата регистрации пользователя
display_name	Имя пользователя
last_access_date	Дата последнего входа
location	Местоположение
reputation	Очки репутации, которые получают за хорошие вопросы и полезные ответы
views	Число просмотров профиля пользователя

5. Stackoverflow.vote_types

Содержит информацию о типах голосов. Голос — это метка, которую пользователи ставят посту. Типов бывает несколько:

- UpMod — такую отметку получают посты с вопросами или ответами, которые пользователи посчитали уместными и полезными.
- DownMod — такую отметку получают посты, которые показались пользователям наименее полезными.
- Close — такую метку ставят опытные пользователи сервиса, если заданный вопрос нужно доработать или он вообще не подходит для платформы.
- Offensive — такую метку могут поставить, если пользователь ответил на вопрос в грубой и оскорбительной манере, например, указав на неопытность автора поста.
- Spam — такую метку ставят в случае, если пост пользователя выглядит откровенной рекламой.

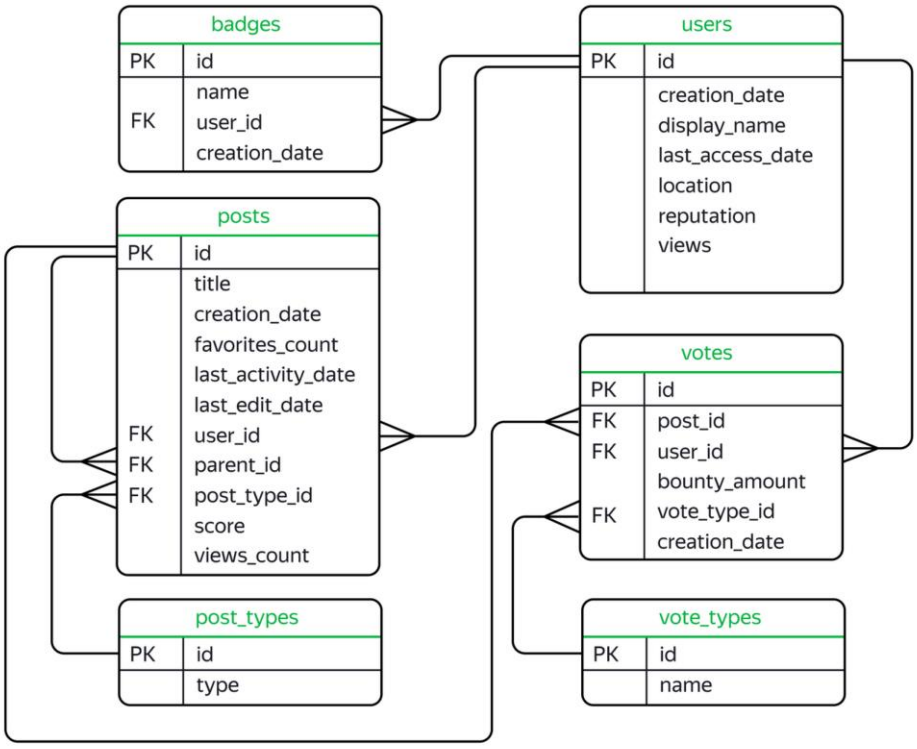
Поле	Описание
id	Идентификатор типа голоса, первичный ключ
name	Название метки

6. Stackoverflow.votes

Содержит информацию о голосах за посты.

Поле	Описание
`id`	Идентификатор голоса, первичный ключ
`post_id`	Идентификатор поста, внешний ключ к таблице posts
`user_id`	Идентификатор пользователя, который поставил посту голос, внешний ключ к таблице users
`bounty_amount`	Сумма вознаграждения, которое назначают, чтобы привлечь внимание к посту
`vote_type_id`	Идентификатор типа голоса, внешний ключ к таблице vote_types
`creation_date`	Дата назначения голоса

ER-диаграмма базы данных



Исследуем данные сервиса StackOverflow с помощью запросов к базе данных:

1. Найдём количество вопросов, которые набрали больше 300 очков или как минимум 100 раз были добавлены в «Закладки»:

```
SELECT COUNT(id)
FROM stackoverflow.posts
WHERE post_type_id = 1 AND
score > 300 OR
favorites_count >= 100
```

Результат

count
1355

2. Посчитаем, сколько в среднем в день задавали вопросов с 1 по 18 ноября 2008 включительно. Округлим результат до целого числа:

```
SELECT ROUND(AVG(amount))
FROM (
      SELECT creation_date::date AS day,
      COUNT(id) AS amount
      FROM stackoverflow.posts
      WHERE post_type_id = 1 AND
      creation_date::date BETWEEN '2008-11-01' AND '2008-11-18'
      GROUP BY day) amount_per_day
```

Результат

count
383

3. Посчитаем, сколько пользователей получили значки сразу в день регистрации:

```
SELECT COUNT(DISTINCT u.id)
FROM stackoverflow.users u
JOIN stackoverflow.badges b ON u.id = b.user_id
WHERE b.creation_date::date = u.creation_date::date
```

Результат

count
7047

4. Посчитаем, сколько уникальных постов пользователя с именем Joel Coehoorn получили хотя бы один голос:

```
SELECT COUNT(DISTINCT p.id)
FROM stackoverflow.posts p
JOIN stackoverflow.votes v ON p.id = v.post_id
WHERE p.user_id IN (
                  SELECT id
                  FROM stackoverflow.users
                  WHERE display_name LIKE '%Joel Coehoorn')
```

Результат

count
12

5. Выгрузим все поля таблицы vote_types. Добавим к таблице поле rank, в которое войдут номера записей в обратном порядке. Отсортируем данные по полю id:

```
SELECT *,
       RANK() OVER(ORDER BY id DESC) AS rank
FROM stackoverflow.vote_types
ORDER BY id
```

Результат

id	name	rank
1	AcceptedByOriginator	15
2	UpMod	14
3	DownMod	13
4	Offensive	12
5	Favorite	11
6	Close	10
7	Reopen	9
8	BountyStart	8
9	BountyClose	7
10	Deletion	6
11	Undeletion	5
12	Spam	4
13	InformModerator	3
14	ModeratorReview	2
15	ApproveEditSuggestion	1

6. Отберём 10 пользователей, которые поставили больше всего голосов типа Close. Отобразим таблицу из двух полей: идентификатором пользователя и количеством голосов. Отсортируем данные сначала по убыванию количества голосов, потом по убыванию значения идентификатора пользователя. Ограничим таблицу 10 строками:

```
SELECT user_id,
       COUNT (vt.id) as vote_amount
FROM stackoverflow.votes v
JOIN stackoverflow.vote_types vt ON v.vote_type_id = vt.id
WHERE name = 'Close'
GROUP BY v.user_id
ORDER BY 2 DESC, 1 DESC
LIMIT 10
```

Результат

user_id	vote_amount
20646	36
14728	36
27163	29
41158	24
24820	23
9345	23
3241	23
44330	20
38426	19

7. Отберём 10 пользователей по количеству значков, полученных в период с 15 ноября по 15 декабря 2008 года включительно. Отобразим несколько полей:

- идентификатор пользователя;
- число значков;
- место в рейтинге — чем больше значков, тем выше рейтинг.

Пользователям, которые набрали одинаковое количество значков, присвоим одно и то же место в рейтинге. Отсортируем записи по количеству значков по убыванию, а затем по возрастанию значения идентификатора пользователя. Ограничим таблицу 10 строками:

```
SELECT user_id,
       COUNT(id),
       DENSE_RANK() OVER(ORDER BY COUNT(id) DESC)
FROM stackoverflow.badges
WHERE creation_date::date BETWEEN '2008-11-15' AND '2008-12-15'
GROUP BY user_id
ORDER BY COUNT(id) DESC, user_id
LIMIT 10
```

Результат

user_id	count	dense_rank
22656	149	1
34509	45	2
1288	40	3
5190	31	4
13913	30	5
893	28	6
10661	28	6
33213	25	7
12950	23	8
25222	20	9

8. Посчитаем, сколько в среднем очков получает пост каждого пользователя. Сформируем таблицу из следующих полей:

- заголовок поста;
- идентификатор пользователя;
- число очков поста;
- среднее число очков пользователя за пост, округлённое до целого числа.

Не будем учитывать посты без заголовка, а также посты, которые набрали ноль очков:

```
SELECT title,
       user_id,
       score,
       ROUND(AVG(score) OVER(PARTITION BY user_id)) AS avg_score
FROM stackoverflow.posts
WHERE title IS NOT NULL AND
      score <> 0
```

Результат

title	user_id	score	avg_score
Diagnosing Deadlocks in SQL Server 2005	1	82	573
How do I calculate someone's age in C#?	1	1743	573
Why doesn't IE7 copy <pre><code> blocks to the clipboard correctly?	1	37	573
Calculate relative time in C#	1	1348	573
Wrapping StopWatch timing with a delegate or lambda?	1	92	573

title	user_id	score	avg_score
Practical non-image based CAPTCHA approaches?	1	318	573
Parameterize an SQL IN clause	1	953	573
Escaping Bracket [in a CONTAINS() clause?	1	10	573
Binary Data in MySQL	2	169	76
Filling a DataSet or DataTable from a LINQ query result set	2	114	76
Why doesn't SQL Full Text Indexing return results for words containing #?	2	19	76
Cross platform Encryption / Decryption applications for secure file transport	2	3	76
Best way to implement request throttling in ASP.NET MVC?	3	196	196
How can I lookup data about a book from its barcode number?	4	69	39
Code to make a DHTMLEd control replace straight quotes with curly quotes	4	9	39
Can I listen on a port (using HttpListener or other .NET code) on Vista without requiring administrator priveleges?	5	36	23
PAD (Portable Application Description) files for shareware / freeware	5	2	23
What's your top feature request for Silverlight?	5	12	23
Is there an embeddable Webkit component for Windows / C# development?	5	72	23
Is there a reliable way to prevent cheating in a web based contest where anonymous users can vote?	5	12	23
How to set up a CSS switcher	5	23	23
How do I convert a .NET console application to a Winforms or WPF application	5	34	23
What's the best way to display a video with rounded corners in Silverlight?	5	4	23
Is it possible to slipstream the Visual Studio 2008 SP1 install?	5	9	23
Creating a mini-site in ASP.NET that works on Blackberry, Windows Mobile, and iPhone	5	14	23

9. Отобразим заголовки постов, которые были написаны пользователями, получившими более 1000 значков. Не будем учитывать посты без заголовков:

```
SELECT title
FROM stackoverflow.posts
WHERE user_id IN (SELECT user_id
                  FROM stackoverflow.badges
                  GROUP BY user_id
                  HAVING COUNT(id) > 1000)
AND title IS NOT NULL
```

Результат

title
What's the strangest corner case you've seen in C# or .NET?
What's the hardest or most misunderstood aspect of LINQ?
What are the correct version numbers for C#?
Project management to go with GitHub

10. Напишем запрос, который выгрузит данные о пользователях из США. Разделим пользователей на три группы в зависимости от количества просмотров их профилей:

- группа 1 — пользователи с числом просмотров больше, либо равным 350;
- группа 2 — пользователи с числом просмотров меньше 350, но больше либо равно 100;
- группа 3 — пользователи с числом просмотров меньше 100.

В итоговой таблице отобразим идентификатор пользователя, количество просмотров профиля и группу. Не будем учитывать пользователей с нулевым количеством просмотров:

```
SELECT id,
       views,
       CASE
       WHEN views >= 350 THEN 1
       WHEN views >= 100 AND views < 350 THEN 2
       WHEN views < 100 THEN 3
       END
FROM stackoverflow.users
WHERE location LIKE '%United States%' AND
views <> 0
```

Результат

id	views	case
3	24396	1
13	35414	1
23	757	1
25	3837	1
36	505	1
43	394	1
45	1971	1
50	1616	1
64	866	1
67	8848	1
72	1475	1
73	169	2
81	311	2
85	1112	1
91	11734	1
93	662	1
102	945	1
105	374	1
112	1972	1
115	3901	1
152	155	2
159	214	2
166	265	2

11. Дополним предыдущий запрос: отобразим также лидеров каждой группы — пользователей, которые набрали максимальное число просмотров в своей группе. Выведем поля с идентификатором пользователя, группой и количеством просмотров. Отсортируем таблицу по убыванию просмотров, а затем по возрастанию значения идентификатора:

```
WITH a AS (SELECT id,
       views,
       CASE
       WHEN views >= 350 THEN 1
       WHEN views >= 100 AND views < 350 THEN 2
       WHEN views < 100 THEN 3
       END AS category
FROM stackoverflow.users
WHERE location LIKE '%United States%' AND
```

```
views <> 0),

b AS (SELECT *,
      RANK() OVER(PARTITION BY category ORDER BY views DESC) AS rn
      FROM a)

SELECT b.id,
       b.category,
       b.views
FROM b
WHERE rn = 1
ORDER BY 3 DESC, 1
```

Результат

id	category	views
16587	1	62813
9094	2	349
9585	2	349
15079	2	349
33437	2	349
3469	3	99
4829	3	99
19006	3	99
22732	3	99
403434	3	99

12. Посчитаем ежедневный прирост новых пользователей в ноябре 2008 года. Сформируем таблицу с полями:

- номер дня;
- число пользователей, зарегистрированных в этот день;
- сумму пользователей с накоплением.

```
WITH a AS (SELECT EXTRACT(DAY FROM creation_date) AS day,
           COUNT(id) AS count_id
           FROM stackoverflow.users
           WHERE creation_date::date BETWEEN '2008-11-01' AND '2008-11-30'
           GROUP BY day)
SELECT *,
SUM(count_id) OVER (ORDER BY day) AS users_cnt
FROM a
```

Результат

day	count_id	users_cnt
1	34	34
2	48	82
3	75	157
4	192	349
5	122	471
6	132	603
7	104	707
8	42	749
9	45	794
10	93	887
11	113	1000

day	count_id	users_cnt
12	113	1113
13	96	1209
14	89	1298
15	42	1340
16	32	1372
17	84	1456
18	89	1545
19	107	1652
20	95	1747
21	81	1828
22	40	1868
23	50	1918
24	84	2002
25	104	2106
26	98	2204
27	71	2275
28	56	2331
29	44	2375
30	33	2408

13. Для каждого пользователя, который написал хотя бы один пост, найдём интервал между регистрацией и временем создания первого поста. Отобразим:

- идентификатор пользователя;
- разницу во времени между регистрацией и первым постом.

Ограничим вывод 5 строками:

```
WITH a AS (SELECT user_id,
           p.creation_date AS first_post,
           ROW_NUMBER() OVER(PARTITION BY user_id ORDER BY p.creation_date) as rn,
           u.creation_date AS reg
           FROM stackoverflow.posts p
           JOIN stackoverflow.users u ON u.id = p.user_id
           ORDER BY user_id)

SELECT user_id,
       first_post - reg AS daydiff
FROM a
WHERE rn = 1
LIMIT 5
```

Результат

user_id	daydiff
1	9:18:29
2	14:37:03
3	3 days, 16:17:09
4	15 days, 5:44:22
5	1 day, 14:57:51

14. Выведем общую сумму просмотров постов за каждый месяц 2008 года. Если данных за какой-либо месяц в базе нет, такой месяц можно пропустить. Результат отсортируем по убыванию общего количества просмотров:

```
SELECT (DATE_TRUNC('month', creation_date))::date AS month,
          SUM(views_count) AS sum_by_month
          FROM stackoverflow.posts
          WHERE EXTRACT(YEAR FROM creation_date::date) = 2008 AND
          views_count IS NOT NULL
          GROUP BY month
          ORDER BY sum_by_month DESC
```

Результат

month	views_by_month
2008-09-01	452928568
2008-10-01	365400138
2008-11-01	221759651
2008-12-01	197792841
2008-08-01	131367083
2008-07-01	669895

Данные в каждом месяца значительно отличаются. Возможно, повышенная активность в сентябре и октябре связана с началом учебного года. Малая активность в июле может свидетельствовать о неполноте данных.

15. Выведем имена самых активных пользователей, которые в первый месяц после регистрации (включая день регистрации) дали больше 100 ответов. Для каждого имени пользователя выведем количество уникальных значений user_id. Отсортируем результат по полю с именами в лексикографическом порядке и ограничим таблицу 12 строками:

```
SELECT u.display_name,
       COUNT(DISTINCT user_id)
FROM stackoverflow.posts p
JOIN stackoverflow.post_types pt ON p.post_type_id = pt.id
JOIN stackoverflow.users u ON u.id = p.user_id
WHERE DATE_TRUNC('day', p.creation_date) >= DATE_TRUNC('day', u.creation_date) AND
DATE_TRUNC('day', p.creation_date) <= DATE_TRUNC('day', u.creation_date) + INTERVAL '1 month' AND
pt.type = 'Answer'
GROUP BY u.display_name
HAVING COUNT(*) > 100
ORDER BY display_name
LIMIT 12
```

Результат

display_name	count
Adam Bellaire	1
Adam Davis	1
Alan	8
Amy B	1
Ben Hoffstein	1
Brian	15
CesarB	1
Chris	29
Craig	10
Dale Ragan	1
Dan	21
Greg	12

Многим популярным именам таким, как Alan, Dan или Chris соответствует несколько значений user_id. Данные лучше не анализировать по имени, иначе результаты могут быть некорректными.

16. Выведем количество постов за 2008 год по месяцам. Отберём посты от пользователей, которые зарегистрировались в сентябре 2008 года и сделали хотя бы один пост в декабре того же года. Отсортируем таблицу по убыванию значения месяца:

```
SELECT DATE_TRUNC('month', creation_date)::date AS month,
COUNT(id) AS post_amount
FROM stackoverflow.posts
WHERE creation_date::date BETWEEN '2008-01-01' AND '2008-12-31' AND
        user_id IN (SELECT u.id
                    FROM stackoverflow.users u
                    JOIN stackoverflow.posts p ON p.user_id = u.id
                    WHERE u.creation_date::date BETWEEN '2008-09-01' AND '2008-09-30' AND
                          p.creation_date::date BETWEEN '2008-12-01' AND '2008-12-31')
GROUP BY month
ORDER BY month DESC
```

Результат

month	post_amount
2008-12-01	17641
2008-11-01	18294
2008-10-01	27171
2008-09-01	24870
2008-08-01	32

В итоговой таблице встречаются аномальные значения: пользователи, зарегистрированные в сентябре, были активны и в августе. Возможно, присутствуют ошибки в данных.

17. Используя данные о постах, выведем несколько полей:

- идентификатор пользователя, который написал пост;
- дату создания поста;
- количество просмотров у текущего поста;
- сумму просмотров постов автора с накоплением.

Отсортируем данные по возрастанию идентификаторов пользователей, а данные об одном и том же пользователе — по возрастанию даты создания поста:

```
SELECT user_id,
       creation_date,
       views_count,
       SUM(views_count) OVER (PARTITION BY user_id ORDER BY creation_date)
FROM stackoverflow.posts
ORDER BY 1, 2
```

Результат

user_id	creation_date	views_count	sum
1	2008-07-31 23:41:00	480476	480476
1	2008-07-31 23:55:38	136033	616509
1	2008-08-12 04:59:35	72431	688940
1	2008-08-14 03:17:18	0	688940
1	2008-08-21 14:18:42	27533	716473
1	2008-09-25 21:58:05	6507	722980
1	2008-10-24 08:39:47	14969	737949
1	2008-11-26 04:15:42	5505	743454

user_id	creation_date	views_count	sum
1	2008-12-03 16:16:43	296436	1039890
2	2008-08-01 04:59:34	79087	79087
2	2008-08-01 05:09:56	65443	144530
2	2008-08-04 05:51:57	7628	152158
2	2008-10-13 23:46:09	2593	154751
2	2008-12-15 03:59:56	0	154751
3	2008-08-04 06:39:40	0	0
3	2008-08-27 13:36:08	0	0
3	2008-08-29 04:50:50	41138	41138

18. Выясним, сколько в среднем дней в период с 1 по 7 декабря 2008 года включительно пользователи взаимодействовали с платформой. Для каждого пользователя отберём дни, в которые они опубликовали хотя бы один пост. Округлим результат до целого числа:

```
WITH a AS (SELECT user_id,
           COUNT(DISTINCT creation_date::date) AS active_days
           FROM stackoverflow.posts
           WHERE creation_date::date BETWEEN '2008-12-01' and '2008-12-07'
           GROUP BY user_id)
SELECT ROUND(AVG(active_days))
FROM a
```

Результат

round
2

19. Выясним, на сколько процентов менялось количество постов ежемесячно с 1 сентября по 31 декабря 2008 года? Отобразим таблицу со следующими полями:

- номер месяца;
- количество постов за месяц;
- процент, который показывает, насколько изменилось количество постов в текущем месяце по сравнению с предыдущим.

Если постов стало меньше, укажем значение процента отрицательным, если больше — положительным. Округлим значение процента до двух знаков после запятой:

```
WITH posts_cnt AS (SELECT EXTRACT(MONTH FROM creation_date) AS month,
                   COUNT(id) AS post_amount
                   FROM stackoverflow.posts
                   WHERE creation_date::date BETWEEN '2008-09-01' AND '2008-12-31'
                   GROUP BY month)
SELECT *,
       ROUND(((post_amount::numeric / LAG(post_amount) OVER (ORDER BY month)) - 1) * 100, 2) AS post_growth
FROM posts_cnt
```

Результат

month	post_amount	post_growth
9	70371	
10	63102	-10.33
11	46975	-25.56
12	44592	-5.07

20. Выгрузим данные активности пользователя, который опубликовал больше всего постов за всё время. Выведем данные за октябрь 2008 года в таком виде:

- номер недели;
- дата и время последнего поста, опубликованного на этой неделе.

```
WITH a AS (SELECT user_id,
           creation_date
FROM stackoverflow.posts
WHERE user_id IN (SELECT user_id
                  FROM stackoverflow.posts
                  GROUP BY user_id
                  ORDER BY COUNT(user_id) DESC
                  LIMIT 1))

SELECT EXTRACT(WEEK FROM creation_date::date) AS week,
       MAX(creation_date) AS last_post_dt
FROM a
WHERE EXTRACT(MONTH FROM creation_date) = 10
GROUP BY week
```

Результат

week	last_post_dt
40	2008-10-05 09:00:58
41	2008-10-12 21:22:23
42	2008-10-19 06:49:30
44	2008-10-31 22:16:01
43	2008-10-26 21:44:36