

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
Национальный исследовательский ядерный университет «МИФИ»



**Институт интеллектуальных кибернетических
систем.**

Кафедра кибернетики (№22)

Направление подготовки 09.03.04 Программная инженерия

**Отчёт о работе по курсу
«Базы данных (теоретические основы баз данных)»**

Вариант: онлайн журнал(habr)

Выполнил: Радецкий Алексей
Группа: Б22-554
Преподаватель: Петровская А.В.

Москва 2024

Оглавление

1. Формулировка задания	3
2. Концептуальная модель базы данных	3
2.1. Конкретизация предметной области.....	3
2.2. Описание предметной области.....	3
2.3. Описание атрибутов.....	4
3. Логическое проектирование	5
4. Физическое проектирование.....	6
4.2. Заполнение базы данных	8
4.3. Результаты заполнения.....	10
5. Выполнение запросов	13

Система рассчитана на работу с зарегистрированными пользователями, как с авторами, так и с читателями статей. Пользователь может являться автором. Пользователи могут читать статьи, что отражается на просмотрах, комментировать статьи, добавлять их

в закладки (поставить лайк), оценивать статьи, что должно отражаться на их рейтинге, оценивать авторов. Авторы могут создавать статьи и относить их к разделам сайта. Комментарии можно писать не только на статьи, но и на другие комментарии (дерево комментариев).

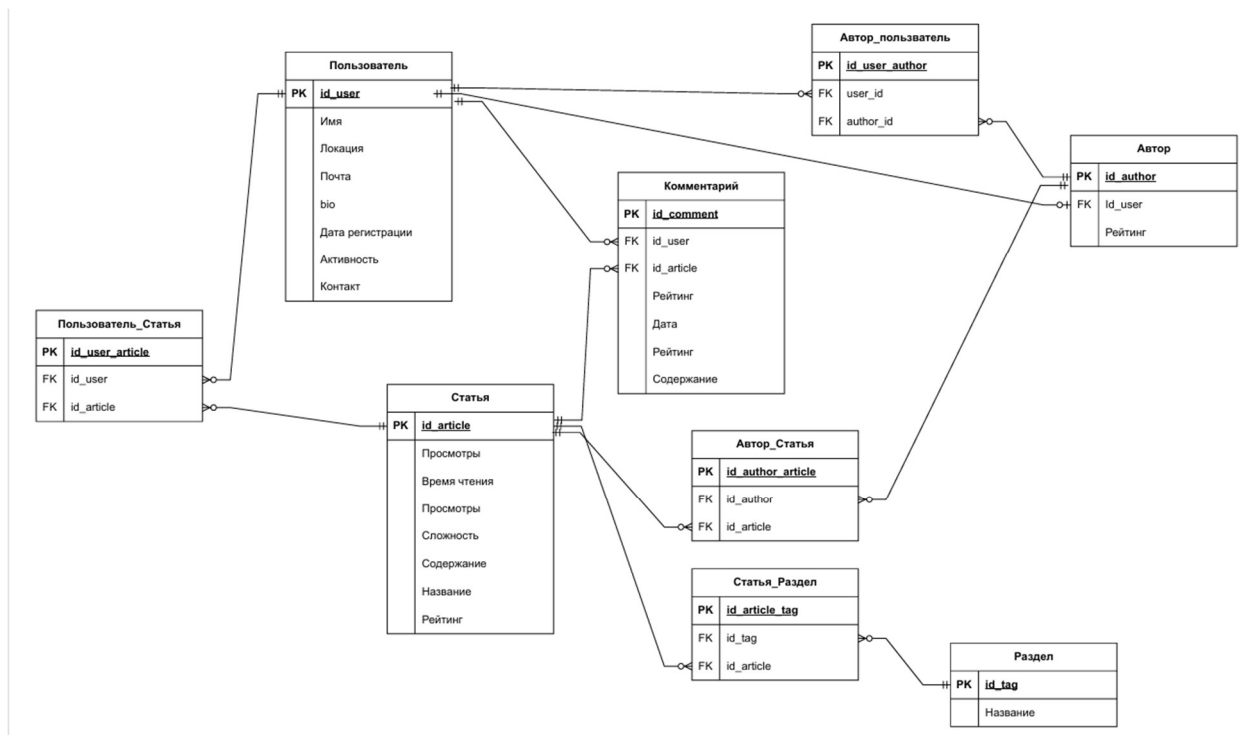
2.3. Описание атрибутов

В процессе анализа были выделены следующие атрибуты, название и описание которых приведены в таблице ниже:

Имя атрибута	Расшифровка
id	Уникальный идентификатор. Есть у каждого объекта.
Дата регистрации	Дата регистрации пользователя
Активность	Дата и время последней активности пользователя
Контакт	Контактные данные оставленные пользователем для других пользователей
bio	Информация о пользователе, которую он оставил для других пользователей
Локация	Страна пользователя
Почта	Почта пользователя
Имя	Никнейм пользователя
Рейтинг	Оценка в диапазоне [-100;100] сформированная, на основе мнения других пользователей
Дата	Время создания статьи или комментария
Тег	Раздел сайта (тема) к которому относится статья
Время чтения	Время необходимое для прочтения статьи
Просмотры	Количество просмотров

3. Логическое проектирование

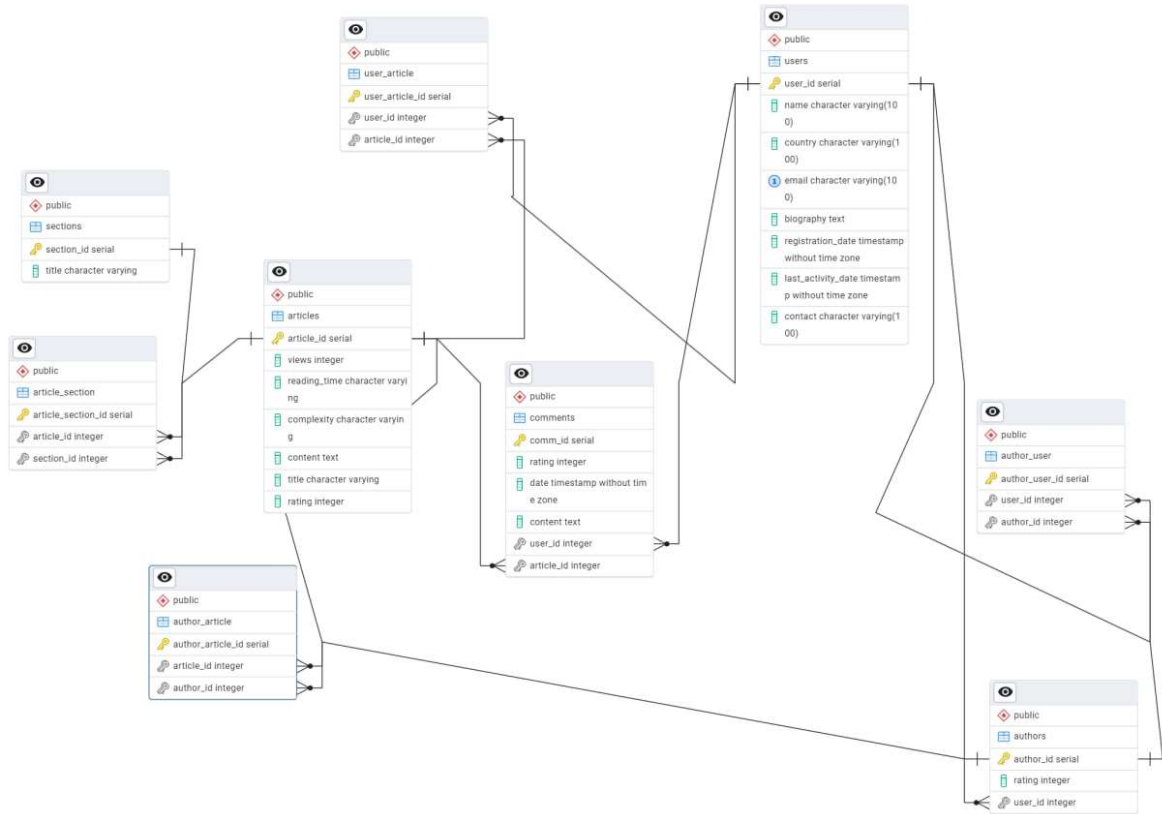
Следующим шагом на основе КМПО была разработана логическая модель базы данных, представленная ниже:



Есть 4 промежуточных таблицы: пользователь_статья, автор_статья, статья_раздел, автор_пользователь. С их помощью реализованы отношения многие ко многим. Все таблицы находятся в нормальной форме.

4. Физическое проектирование

В качестве СУБД для реализации разработанной базы данных была выбрана PostgreSQL. В связи с проведённым анализом предметной области была проработана следующая физическая схема БД. Она представлена на следующем рисунке



4.1. Создание таблиц

Ниже приведен код на python (использовалась orm sqlalchemy) для создания таблиц, описанных выше.

User:

```
5 Base = declarative_base()
6
7
8 class User(Base): 3 usages
9     __tablename__ = 'users'
10
11     user_id = Column(Integer, primary_key=True, autoincrement=True)
12     name = Column(String, nullable=False)
13     country = Column(String, nullable=False)
14     email = Column(String, nullable=False, unique=True)
15     biography = Column(Text, nullable=True)
16     registration_date = Column(DateTime, server_default=func.now())
17     last_activity_date = Column(DateTime, onupdate=func.now())
18     contact = Column(String, nullable=True)
```

Comment

```
21 class Comment(Base): 3 usages
22     __tablename__ = 'comments'
23
24     comm_id = Column(Integer, primary_key=True, autoincrement=True)
25     rating = Column(Integer, nullable=False)
26     parent_id = Column(Integer, nullable=True)
27     date = Column(DateTime, server_default=func.now())
28     content = Column(Text, nullable=False)
29     user_id = Column(Integer, ForeignKey('users.user_id'))
30     article_id = Column(Integer, ForeignKey('articles.article_id'))
31
```

Article

```
33 class Article(Base): 2 usages
34     __tablename__ = 'articles'
35
36     article_id = Column(Integer, primary_key=True, autoincrement=True)
37     views = Column(Integer, default=0)
38     date = Column(DateTime, server_default=func.now())
39     reading_time = Column(String, nullable=False)
40     complexity = Column(String, nullable=False)
41     content = Column(Text, nullable=False)
42     title = Column(String, nullable=False)
43     rating = Column(Integer, nullable=True)
```

User_article

```
46 class UserArticle(Base): 2 usages
47     __tablename__ = 'user_article'
48
49     user_article_id = Column(Integer, primary_key=True, autoincrement=True)
50     user_id = Column(Integer, ForeignKey('users.user_id'))
51     article_id = Column(Integer, ForeignKey('articles.article_id'))
52
```

Author_user

```
54 class AuthorUser(Base):
55     __tablename__ = 'author_user'
56
57     author_user_id = Column(Integer, primary_key=True, autoincrement=True)
58     user_id = Column(Integer, ForeignKey('users.user_id'))
59     author_id = Column(Integer, ForeignKey('authors.author_id'))
60
```

Author

```
62 class Author(Base): 3 usages
63     __tablename__ = 'authors'
64
65     author_id = Column(Integer, primary_key=True, autoincrement=True)
66     rating = Column(Integer, nullable=True)
67     user_id = Column(Integer, ForeignKey('users.user_id'))
68
```

Author_article

```
70 class AuthorArticle(Base): 3 usages
71     __tablename__ = 'author_article'
72
73     author_article_id = Column(Integer, primary_key=True, autoincrement=True)
74     article_id = Column(Integer, ForeignKey('articles.article_id'))
75     author_id = Column(Integer, ForeignKey('authors.author_id'))
```

Section

```
78 class Section(Base): 3 usages
79     __tablename__ = 'sections'
80
81     section_id = Column(Integer, primary_key=True, autoincrement=True)
82     title = Column(String, nullable=False)
```

Article_section

```
85 class ArticleSection(Base): 2 usages
86     __tablename__ = 'article_section'
87
88     article_section_id = Column(Integer, primary_key=True, autoincrement=True)
89     article_id = Column(Integer, ForeignKey('articles.article_id'))
90     section_id = Column(Integer, ForeignKey('sections.section_id'))
```

Создание таблиц

```
93 if __name__ == "__main__":
94     engine = create_engine("postgresql://postgres:923709@localhost/mephi_db_habr_v1", echo=True)
95
96     engine.echo = False
97     Base.metadata.drop_all(engine)
98     engine.echo = True
99     Base.metadata.create_all(engine)
```

4.2. Заполнение базы данных

Заполнение базы данных проводилось при помощи Python, sqlalchemy, faker. Были подготовлены csv файлы со сгенерированными комментариями и тегами статей. Остальные данные генерировались с помощью библиотеки faker.

Порядок вызова функций для заполнения таблиц:

```
10
11 if __name__ == "__main__":
12     engine = create_engine("postgresql://postgres:923709@localhost/mephi_db_habr_v1", echo=True)
13     Session = sessionmaker(bind=engine)
14     session = Session()
15     fake = Faker()
16
17     unit_users(session, fake)
18     unit_authors(session, fake)
19     article_count = unit_articles(session, fake)
20     unit_sections(session, fake, article_count)
21     unit_comments(session, fake, article_count)
22
23     session.close()
```


Таблица users

```
def unit_users(session: Session, fake: Faker) -> list[User]: 2 usages
    users = []
    for i in range(c.USER_NUM):
        name = fake.unique.user_name()
        country = fake.country()
        email = fake.unique.email(),
        biographic = fake.text(max_nb_chars=200)
        registration_date = fake.date()
        last_activity_date = fake.date_between(start_date=date.fromisoformat(registration_date))
        contact = fake.basic_phone_number()
        users.append(User(name=name, country=country, email=email, biography=biographic, registration_date=reg:
            last_activity_date=last_activity_date, contact=contact))
    session.add_all(users)
    session.commit()
    return users
```

Таблица authors

```
9 def unit_authors(session: Session, fake: Faker) -> list[Author]: 2 usages
10     a = [i for i in range(1, c.USER_NUM + 1)]
11     user_id_sample = random.sample(a, c.AUTHOR_NUM)
12     authors = []
13     for i in range(c.AUTHOR_NUM):
14         rating = random.randint(-1 * c.RATING_MAX_ABS, c.RATING_MAX_ABS)
15         user_id = user_id_sample[i]
16         authors.append(Author(rating=rating, user_id=user_id))
17     session.add_all(authors)
18     session.commit()
19     return authors
```

Таблицы article, author_article, user_article

```
10 def unit_articles(session: Session, fake: Faker) -> int: 2 usages
11     articles = []
12     author_article = []
13     article_id = 1
14     for i in range(c.AUTHOR_NUM):
15         for j in range(random.randint(a: 0, b: c.ARTICLE_COEFF)):
16             views = random.randint(a: 0, b: 100000)
17             reading_time = random.choice(['30 min', '12 min', '1 hour', 'less then 5 min', '10 min', '2 hour'])
18             complexity = random.choice(['hard', 'medium', 'easy'])
19             content = 'Let\'s imagine that it\'s the text of an article' + fake.unique.text(1000)
20             title = 'It\'s title' + fake.unique.text(25)
21             date_article = fake.date()
22             rating = random.randint(-1 * c.RATING_MAX_ABS, c.RATING_MAX_ABS)
23             articles.append(Article(views=views, date=date_article, reading_time=reading_time, complexity=complexity,
24                                     content=content, title=title, rating=rating))
25             author_article.append(AuthorArticle(article_id=article_id, author_id=(i + 1)))
26             while random.randint(a: 1, b: 10) <= 3:
27                 author_article.append(AuthorArticle(article_id=article_id, author_id=random.randint(a: 1, c.AUTHOR_NUM)))
28             article_id += 1
29
30     article_id -= 1
31     user_article = []
32     article_arr = [i for i in range(1, article_id + 1)]
33     user_arr = [i for i in range(1, c.USER_NUM + 1)]
34     article_sample = random.sample(article_arr, article_id - 1000)
35     for i in range(len(article_sample)):
36         user_count = random.randint(a: 1, b: 100)
37         user_sample = random.sample(user_arr, user_count)
38         for j in range(len(user_sample)):
39             user_article.append(UserArticle(user_id=user_sample[j], article_id=article_sample[i]))
40
41     session.add_all(articles)
42     session.add_all(author_article)
43     session.add_all(user_article)
44     session.commit()
45     return article_id
```

Таблица sections

```
11 def unit_sections(session: Session, fake: Faker, article_count: int) -> list[Section]: 2 usages
12     sections = []
13     section_article = []
14     sections_from_file = []
15     article_arr = [i for i in range(1, article_count + 1)]
16     with open('sections.csv', newline='\n', encoding='utf-8') as csvfile:
17         reader = csv.reader(csvfile)
18         for row in reader:
19             sections_from_file.append(row[0])
20     section_count = len(sections_from_file)
21     for i in range(section_count):
22         sections.append(Section(title=sections_from_file[i]))
23         article_sample = random.sample(article_arr, random.randint(a: 20, b: 1000))
24         for j in range(len(article_sample)):
25             section_article.append(ArticleSection(article_id=article_sample[j], section_id=i + 1))
26     session.add_all(sections)
27     session.commit()
28     session.add_all(section_article)
29     session.commit()
30     return sections
```

Таблица comments

```
8 def unit_comments(session: Session, fake: Faker, article_count: int) -> list[Comment]: 2 usages
9     comms = []
10     comms_from_file = []
11     with open('comments.csv', newline='\n', encoding='utf-8') as csvfile:
12         reader = csv.reader(csvfile)
13         for row in reader:
14             comms_from_file.append(row[0])
15     comms_count = len(comms_from_file)
16     for i in range(article_count):
17         comms_to_article = random.sample(comms_from_file, random.randint(a: 1, comms_count))
18         for j in range(len(comms_to_article)):
19             rating = random.randint(-1 * c.RATING_MAX_ABS, c.RATING_MAX_ABS)
20             date_comm = fake.date()
21             user_id = random.randint(a: 1, c.USER_NUM)
22             article_id = i + 1
23             comms.append(Comment(rating=rating, date=date_comm, content=comms_to_article[j], user_id=user_id,
24                                 article_id=article_id))
25     session.add_all(comms)
26     session.commit()
27     return comms
```

4.3. Результаты заполнения

Далее представлены результаты работы программы на примере таблиц, соответствующих функциям, приведенным выше.

Authors

	author_id [PK] integer	rating integer	user_id integer
1	1	60	4237
2	2	-79	4996
3	3	-58	1552
4	4	64	6030
5	5	-62	7349
6	6	45	7634
7	7	29	6534
8	8	-97	9615
9	9	18	2738
10	10	-4	2899
11	11	17	3498

Users

	user_id [PK] integer	name character varying	country character varying	email character varying	biograph text	registration_date timestamp with time zone	last_activity_date timestamp with time zone	contact character varying
1	1	villegaschase	El Salvador	jordan56@example.org	Exac...	1993-07-11 ...	2010-06-28 ...	383-965-6405
2	2	mollymccall	Tokelau	rhoward@example.com	Crim...	1990-04-02 ...	2021-02-05 ...	(898)604-8474
3	3	lee98	Austria	wmata@example.com	Mor...	2011-04-24 ...	2019-04-09 ...	5494613747
4	4	ashley41	Suriname	vbaker@example.net	Both...	1982-03-02 ...	1995-10-15 ...	(209)407-9970
5	5	nevans	Turkmenistan	peteronmadison@exam...	Offi...	1976-12-30 ...	1981-11-14 ...	4425884193
6	6	greenedebor...	Cuba	adamsnicole@example.c...	Atte...	1986-04-22 ...	1988-07-12 ...	(401)934-5804
7	7	kthornton	Nauru	nsims@example.com	Littl...	1973-01-11 ...	1987-04-02 ...	(520)858-9830
8	8	xstout	Armenia	spencermatthew@exam...	Loo...	2024-11-21 ...	2024-11-25 ...	(344)338-8871
9	9	nicholasclay	Norfolk Island	nlester@example.org	May ...	2011-02-18 ...	2017-10-25 ...	668-641-5777
10	10	xwallace	Uganda	hunter76@example.net	Inter...	1972-09-11 ...	1996-09-15 ...	379-829-8457
11	11	knoxdeborah	Chad	renee86@example.org	Parti...	2010-01-03 ...	2012-05-29 ...	(250)800-3591

Articles

	article_id [PK] integer	views integer	date timestamp with time zone	reading_time character	complexity character	content text	title character varying	rating integer
1	1	55499	1988-12-31 ...	10 min	hard	Let's imagine that it's the text of an articleEffo...	It's titleItem worker blood raise.	69
2	2	61268	1978-07-16 ...	2 hour	easy	Let's imagine that it's the text of an articleKey ...	It's titleNor car commercial as.	-4
3	3	42516	2014-03-11 ...	1 hour	easy	Let's imagine that it's the text of an articleEven...	It's titleMan bar all election.	-40
4	4	72708	2009-10-01 ...	10 min	easy	Let's imagine that it's the text of an articleTrav...	It's titleColor movie when page.	-36
5	5	78255	1976-07-18 ...	1 hour	medi...	Let's imagine that it's the text of an articleSinc...	It's titleBelieve pass tax body.	-84
6	6	29898	1970-01-02 ...	12 min	hard	Let's imagine that it's the text of an articleLet ...	It's titleOften heavy lead with.	14
7	7	70066	2004-03-09 ...	30 min	easy	Let's imagine that it's the text of an articleSch...	It's titlePm economy do rich.	56
8	8	93023	1974-08-02 ...	10 min	easy	Let's imagine that it's the text of an articleEast...	It's titleItself give part answer.	-44
9	9	75254	1992-03-13 ...	12 min	hard	Let's imagine that it's the text of an articleHan...	It's titleMe right can man.	19
10	10	74970	2011-01-01 ...	30 min	easy	Let's imagine that it's the text of an articleVote...	It's titleYard push water brother.	-32
11	11	97033	1971-04-13 ...	30 min	hard	Let's imagine that it's the text of an articleSec...	It's titleImpact hope specific.	-43
12	12	5309	1977-11-28 ...	less th...	hard	Let's imagine that it's the text of an articleSite ...	It's titleCareer thus now.	87

Comments

	comm_id [PK] integer	rating integer	parent_id integer	date timestamp without time zone	content text	user_id integer	article_id integer
1	1	-50	[null]	2006-03-04 00:00:00	Спасибо за статью! Очень интер...	5767	1
2	2	-69	[null]	1980-02-25 00:00:00	А как насчет безопасности? Этот...	9185	1
3	3	81	[null]	1970-05-03 00:00:00	Спасибо за информацию, очень ...	6584	1
4	4	-33	[null]	1978-07-23 00:00:00	Спасибо за подробный обзор! Оч...	763	1
5	5	-91	[null]	1992-04-15 00:00:00	Спасибо за статью, очень захват...	9018	1
6	6	47	[null]	1986-03-05 00:00:00	А как насчет поддержки старых ...	1453	1
7	7	93	[null]	1994-04-21 00:00:00	Если бы я знал, что здесь публик...	9038	1
8	8	-68	[null]	1982-06-21 00:00:00	Не могу поверить, что этот матер...	9578	1
9	9	43	[null]	2008-08-11 00:00:00	Спасибо за подробный анализ! О...	9562	1
10	10	-100	[null]	1978-06-27 00:00:00	Я бы посоветовал использовать ...	797	1
11	11	9	[null]	2002-07-09 00:00:00	Статья очень познавательна, мн...	7509	1
12	12	-6	[null]	1976-10-02 00:00:00	Статья очень понравилась, мног...	79	1
13	13	55	[null]	1995-12-07 00:00:00	Я бы посоветовал использовать ...	5716	1
14	14	22	[null]	1999-06-16 00:00:00	Очень познавательная статья!	5177	1

Sections

	section_id [PK] integer	title character varying
1	1	python
2	2	комбинаторика
3	3	алгоритмы сортировки
4	4	машинное обучение
5	5	нейронные сети
6	6	искусственный интеллект
7	7	большие данные
8	8	облачные вычисления
9	9	кибербезопасность
10	10	блокчейн

Article_section

	article_section_id [PK] integer	article_id integer	section_id integer
1	1	7893	1
2	2	22287	1
3	3	6240	1
4	4	7976	1
5	5	5042	1
6	6	12618	1

Author_article

	author_article_id [PK] integer	article_id integer	author_id integer
1	1	1	1
2	2	2	2
3	3	3	2
4	4	4	2
5	5	5	2
6	6	5	1260
7	7	6	2
8	8	7	2
9	9	8	3

User_article

	user_article_id [PK] integer	user_id integer	article_id integer
1	1	9459	20756
2	2	3208	20756
3	3	9334	20756
4	4	9607	20756
5	5	4653	20756
6	6	4692	20756
7	7	2580	20756
8	8	3421	20756
9	9	8575	20756
10	10	8050	20756

5. Выполнение запросов

В этом разделе приведены различные запросы к реализованной базе данных — их краткие описания, непосредственно запрос на языке SQL и результат выполнения.

1.1 Для каждого автора найти количество просмотров его статей

```
2 SELECT sum(a.views) AS all_views, aa.author_id, u.name FROM ((articles a
INNER JOIN author_article aa
ON a.article_id = aa.article_id) INNER JOIN authors au ON aa.author_id =
au.author_id)
INNER JOIN users u ON au.user_id = u.user_id
GROUP BY aa.author_id, u.name
ORDER BY all_views DESC
LIMIT 100
```

	all_views bigint	author_id integer	name character varying
1	1905300	83	gmendez
2	1882296	1336	dawnjackson
3	1874935	565	melindajohnson
4	1869332	1433	leonard56
5	1847895	753	ronnieharper
6	1846750	1914	lynchmary

1.2. Найти автора с самым большим количеством статей

```
SELECT count(aa.article_id), aa.author_id, u.name FROM (author_article aa
INNER JOIN authors au ON aa.author_id = au.author_id)
INNER JOIN users u ON au.user_id = u.user_id
GROUP BY aa.author_id, u.name
ORDER BY count(aa.article_id) DESC
LIMIT 100
```


	count bigint	author_id integer	name character varying
1	37	1651	lmurray
2	37	179	brownjohn
3	36	330	khammond
4	35	1565	brian98
5	34	523	marydean
6	34	565	melindajohnson
7	34	1629	fcole
8	34	439	davidwhite
9	34	1075	brianmelton

1.3 Вывести пользователей, написавших больше n комментариев за последние сутки

```
SELECT c.user_id, u.name, count(c.comm_id) AS comm_count FROM comments c
INNER JOIN users u ON c.user_id = u.user_id
GROUP BY c.user_id, u.name
HAVING count(c.comm_id) > 170
ORDER BY count(c.comm_id) DESC
```

	user_id integer	name character varying	comm_count bigint
1	8086	johnstonadam	174
2	4838	sabrinachavez	174

1.4.В каком регионе больше всего авторов

```
SELECT u.country, count(u.user_id) AS au_count FROM users u INNER JOIN
authors a ON u.user_id = a.user_id
GROUP BY u.country
ORDER BY au_count DESC
```

	country character varying	au_count bigint
1	Congo	15
2	Korea	15
3	Heard Island and McDonald Islands	15
4	Morocco	14
5	Comoros	14
6	Puerto Rico	14
7	Nigeria	13
8	Cocos (Keeling) Islands	13

2.1. Вывести всех пользователей за какой-либо период, у которых x% комментариев, написанных ими, имеют рейтинг меньше y.

```
SELECT
    c.user_id,
    u.name,
    (SUM(CASE WHEN c.rating > 0 THEN 1 ELSE 0 END) * 100.0 / COUNT(*)) AS
positive_comment_percentage
FROM comments c INNER JOIN users u ON c.user_id = u.user_id
WHERE c.date > '01.01.2020'
GROUP BY c.user_id, u.name
ORDER BY c.user_id DESC
LIMIT 100
```

	user_id integer	name character varying	positive_comment_percentage numeric
1	10000	thomasclark	54.5454545454545455
2	9999	reidalicia	63.6363636363636364
3	9998	jacob74	26.6666666666666667
4	9997	breanna50	66.6666666666666667
5	9996	ckennedy	53.8461538461538462
6	9995	bishopdenise	22.2222222222222222
7	9994	veronicadrake	50.0000000000000000
8	9993	larsoniennifer	33.3333333333333333

2.2 Для каждого тега вывести топ авторов по максимальному рейтингу статьи за какой-то период.

```
SELECT author_id, section_id, max_rating, rankk FROM
(SELECT
  aa.author_id AS author_id,
  sa.section_id AS section_id,
  max(a.rating) AS max_rating,
  ROW_NUMBER() OVER (PARTITION BY sa.section_id ORDER BY MAX(a.rating)
DESC) AS rankk
FROM (articles a INNER JOIN author_article aa ON a.article_id =aa.article_id)
INNER JOIN article_section sa ON sa.article_id = a.article_id
GROUP BY aa.author_id, sa.section_id
ORDER BY sa.section_id, max(a.rating) DESC)
WHERE rankk <= 5
```

	author_id integer	section_id integer	max_rating integer	rankk bigint
1	1938	1	100	1
2	1758	1	100	2
3	254	1	100	3
4	1500	1	100	4
5	1916	1	100	5
6	1294	2	100	1
7	9	2	100	2
8	142	2	100	3
9	832	2	99	4

2.3. Вывести авторов, чаще всего отвечающих на комментарии под своими статьями

```
SELECT
  aa.author_id,
  COUNT(*) AS comment_count
FROM
  (comments c INNER JOIN author_article aa ON c.article_id = aa.article_id)
INNER JOIN authors a ON a.author_id = aa.author_id
WHERE a.user_id = c.user_id
GROUP BY aa.author_id
ORDER BY comment_count DESC
LIMIT 100
```

	author_id integer	comment_count bigint
1	54	2
2	769	2
3	1198	2
4	1621	2
5	1434	2
6	1154	2
7	66	1
8	111	1
9	120	1

2.4 Для каждого тега сравнить средний рейтинг статей этого тега, со средним рейтингом по всем статьям

```
SELECT
    sa.section_id,
    AVG(a.rating) AS avg_section_rating,
    (SELECT AVG(rating) AS a FROM articles) AS avg_rating
FROM
    article_section sa INNER JOIN articles a ON sa.article_id = a.article_id
GROUP BY sa.section_id
LIMIT 1000
```

	section_id integer	avg_section_rating numeric	avg_rating numeric
1	58	-0.94243070362473347548	0.30189648360331884631
2	8	2.7254901960784314	0.30189648360331884631
3	184	-2.7542372881355932	0.30189648360331884631
4	116	1.3196078431372549	0.30189648360331884631
5	87	-0.61049723756906077348	0.30189648360331884631
6	71	6.7326732673267327	0.30189648360331884631
7	68	-1.2222222222222222	0.30189648360331884631
8	51	-1.1652892561983471	0.30189648360331884631
9	146	1.3155339805825243	0.30189648360331884631

3.1. Получить медианный рейтинг самой популярной статьи по авторам

```
with max_rating_t AS (
    SELECT
        aa.author_id AS author_id,
        max(a.rating) AS max_rating,
        ROW_NUMBER() over(order by max(a.rating) asc) as rank
    FROM articles a INNER JOIN author_article aa ON a.article_id =
aa.article_id
    GROUP BY aa.author_id),
median AS (
    SELECT
        ROUND(SUM(mr.rank) / COUNT(*)) AS med
    FROM max_rating_t mr)

SELECT
    mt.max_rating
FROM max_rating_t mt CROSS JOIN median
WHERE mt.rank = median.med
```


	max_rating integer	
1		92

3.2 Для тега, осмотреть динамику прироста общего числа просмотров по годам/дням/месяцам




```
WITH all_dates AS (
    SELECT generate_series('1970-01-01'::date, '2024-12-31'::date, '1
day'::interval) AS date
),
exist_dates AS (
    SELECT
        SUM(a.views) AS views,
        a.date AS date
    FROM
        articles a INNER JOIN article_section sa ON a.article_id = sa.article_id
    WHERE sa.section_id = 1
    GROUP BY a.date
),
day_views AS (
    SELECT
        ad.date AS date,
        COALESCE(ed.views, 0) AS views
    FROM
        all_dates ad LEFT JOIN exist_dates ed ON ad.date = ed.date
)
SELECT
    year,
    total_views,
    ROUND(((total_views - LAG(total_views, 1) OVER (ORDER BY year)) /
(total_views + 1) * 100), 2)
FROM(
    SELECT
        DATE_PART('year', date) AS year,
        SUM(views) AS total_views
    FROM day_views
    GROUP BY DATE_PART('year', date)
)
```

	year double precision	total_views numeric	round numeric
1	1970	546604	[null]
2	1971	1168596	53.23
3	1972	818600	-42.76
4	1973	911750	10.22
5	1974	658153	-38.53
6	1975	643612	-2.26
7	1976	835631	22.98
8	1977	981165	14.83
9	1978	414355	-136.79
10	1979	1001605	58.63

3.3. Найти самые популярные статьи для каждого тега.




Где популярная статья - это статья которая входит в тройку лучших уникальных рейтингов

```
WITH top_views AS (  
  SELECT  
    a.article_id AS article_id,  
    sa.section_id AS section_id,  
    dense_rank() over(partition by sa.section_id order by a.views desc) as  
    rank  
  FROM  
    articles a INNER JOIN article_section sa ON a.article_id = sa.article_id  
)  
SELECT top_views.article_id, top_views.section_id, top_views.rank FROM  
top_views  
WHERE top_views.rank <= 3  
LIMIT 1000
```

	article_id  integer	section_id  integer	rank  bigint
1	7193	1	1
2	25106	1	2
3	22275	1	3
4	15182	2	1
5	12789	2	2
6	10242	2	3
7	23749	3	1
8	6067	3	2
9	5798	3	3

3.4 Построить иерархию комментариев. Комментарий, ответы к нему, ответы к ответам и тд.

```
WITH RECURSIVE comments_graph AS (  
  SELECT  
    c.comm_id,  
    0 AS depth,  
    CAST(comm_id AS VARCHAR(255)) AS path  
  FROM comments c  
  WHERE parent_id IS NULL  
  UNION  
  SELECT  
    c.comm_id,  
    cg.depth + 1,  
    CAST(cg.path || '.' || c.comm_id AS VARCHAR(255)) AS path  
  FROM comments c, comments_graph cg  
  WHERE cg.comm_id = c.parent_id  
)  
SELECT * FROM comments_graph  
ORDER BY depth DESC  
LIMIT 1000
```

	comm_id integer 	depth integer 	path character varying (255) 
1	1272898	3	1.1272893.1272897.1272898
2	1272897	2	1.1272893.1272897
3	1272893	1	1.1272893
4	1272896	1	1.1272896
5	1272894	1	1.1272894
6	1272895	1	1.1272895
7	7	0	7
8	6	0	6
9	3	0	3
10	11	0	11