# Реляционная модель

## PostgreSQL

![sql](data:text/html; charset=utf-8;base64,)

sql

## Коллекции и сущности

n - максимально необходимый размер строки для текста. Оптимальное n = 150 символов.

### Client

Назначение: хранение данных о клиентах.

Типы данных полей: - id int(4) - name text(n) - email varchar(40) - workplace text(n) - sex varchar(6) - password varchar(20) - phone char(10) - age int(4) - birthdate date(8) - salary int(4) - self\_employment\_status varchar(13): “self-employed”, “entrepreneur”, “idle” - owned\_property text[10] (10\*n) - marital\_status varchar(10): “married”, “single” - spouse\_workplace text(n) - spouse\_salary int(4) - amount\_of\_children int(4) - rating decimal(4)

### Credit

Назначение: хранение данных о кредите.

Типы данных полей: - id int(4) - loan\_name varchar(100) - opening\_date datetime(8) - expiration\_time int(4) - amount decimal(17) - interest\_rate decimal(4) - monthly\_payment decimal(17) - next\_payment\_date date(8) - debt decimal(17) - payments\_overdue int(4) - co-borrowers: text[5] (5\*n) - deposit: decimal(17)

### CreditHistory

Назначение: хранение данных о кредитной истории конкретного клиента.

Типы данных полей: - id int(4) - client\_id int(4) - loan\_id int(4) - status varchar(20): “opened”, “closed”, “expired” - closing\_date datetime(8)

### CreditRequest

Назначение: хранение данных о заявке на кредит.

Типы данных полей: - id int(4) - client\_id int(4) - loan\_id int(4) - request\_time datetime(8) - status varchar(20): “processing”, “approved”, “rejected”

### Admin

Назначение: хранение данных о сотруднике банка.

Типы данных полей: - id int(4) - name text(n) - email varchar(40) - sex varchar(6) - password varchar(20) - age int(4) - phone char(10) - post text(n) - birthdate date(8)

### InteractionHistory

Назначение: хранение данных о взаимодействии администратора с заявками по кредитам

Типы данных полей: - id int(4) - credit\_request\_id int(4) - admin\_id int(4) - processing\_date datetime(8) - decision boolean(1)

## Оценка объема информации

Средний размер информации, хранимой в модели: - Client: 13 + 4 \* 6 + 6 + 8 + 10 \* 2 + 20 + 40 + n \* 13 = 2081 байт - Credit: 4 \* 4 + 8 \* 2 + 5*n + 17*  4 + 100 = 950 байт - CreditRequest: 4 \* 3 + 8 + 20 = 40 байт - CreditHistory: 4 \* 3 + 20 = 32 байт - InteractionHistory: 1 + 4 \* 3 + 8 = 21 байт - Admin: 4 \* 2 + 6 + 8 + 10 + 20 + 40 + n \* 2 = 392 байт

В среднем пользователь будет: - брать 5 кредитов. - отправлять количество заявок, равному 8. - хранить в истории как минимум столько кредитов, сколько он взял: 5 - каждая заявка будет обработана и хранится в истории взаимодействия с клиентом: 5. - в среднем заявки обработает 4 сотрудника банка.

При количестве клиентов, равных clients:

## Избыточность данных

В таблице Client избыточными является поля self\_employment\_status, age, marital\_status - 27 байт В таблице Admin избыточным является поле age - 4 байта В таблице Credit избыточным является поле monthly\_payment, next\_payment\_date - 25 байт

Тогда формула для чистого объема данных примет вид:

Тогда избыточность данных (отношение между фактическим объемом модели и “чистым” объемом данных):

## Направление роста модели

При добавлении сущности Client и Admin дополнительных сущностей не создается.

При создании сущности CreditRequest создается сущность Credit (в случае одобрения кредита, 210 байт), сущность CreditHistory (в случае одобрения кредита, 32 байта), и сущность InteractionHistory (в любом случае, 21 байт).

## Примеры данных

### Таблица Client

| PK id | name | email | workplace | sex | password | phone | age | birthdate | salary | self\_employment\_status | owned\_property | marital\_status | spouse\_workplace | spouse\_salary | amount\_of\_children | rating |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | Иванов И.И. | ivanov@example.com | Программист | male | secret123 | 1234567890 | 30 | 1993-05-15 | 60000 | TRUE | [“Квартира”] | married | Офис менеджер | 50000 | 2 | 7.75 |
| 2 | Петрова А.С. | petrova@example.com | Дизайнер | female | pass456 | 0987654321 | 28 | 1995-08-20 | 55000 | FALSE | [“Дом”] | married | Не работает | NULL | 1 | 4.50 |
| 3 | Сидоров П.П. | sidorov@example.com | Учитель | male | qwerty789 | 1122334455 | 45 | 1978-11-30 | 70000 | TRUE | [“Квартира”] | single | Школа | 40000 | 3 | 9.90 |

### Таблица Admin

| PK id | name | email | sex | password | age | phone | post | birthdate |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | Кузнецов А.Н. | kuznetsov@example.com | male | pass123 | 32 | 3216549870 | Администратор | 1991-04-12 |
| 2 | Васильева И.М. | vasilieva@example.com | female | mypassword | 29 | 9876543210 | Владелец | 1994-07-25 |
| 3 | Федоров С.В. | fedorov@example.com | male | secret456 | 40 | 4567891230 | Старший администратор | 1983-09-30 |
| 4 | Сергеева Т.П. | sergeeva@example.com | female | qwerty789 | 36 | 6543210987 | Стажер | 1987-11-05 |

### Таблица Credit

| PK id | loan\_name | opening\_date | expiration\_time | amount | interest\_rate | monthly\_payment | next\_payment\_date | debt | payments\_overdue | co-borrowers | deposit |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | Ипотека | 2024-01-15 09:00:00 | 36 | 5000000.00 | 15.00 | 14999.99 | 2024-11-15 | 300000.00 | 1122334455 | [Иванов И.И., Иванов И.И.] | 100000.00 |
| 2 | Автокредит | 2024-03-20 11:30:00 | 24 | 300000.00 | 16.50 | 13400.50 | 2024-10-20 | 150000.00 | 1122334455 | [Петрова А.С., Иванов И.И.] | 50000.00 |
| 3 | Молодежный кредит | 2024-05-10 14:45:00 | 12 | 1000000.00 | 17.25 | 8900.75 | 2024-09-10 | 40000.00 | 1122334455 | [Сидоров П.П.] | 200000.00 |
| 4 | Кредит наличными | 2024-02-05 08:15:00 | 48 | 2000000.00 | 14.75 | 5200.25 | 2025-02-05 | 100000.00 | 1122334455 | [Смирнова Е.В., Иванов И.И.] | 75000.00 |

### Таблица CreditRequest

| PK id | FK client\_id | FK loan\_id | request\_time | status |
| --- | --- | --- | --- | --- |
| 1 | 101 | 201 | 2024-10-01 10:30:00 | processing |
| 2 | 102 | 202 | 2024-10-02 14:15:00 | approved |
| 3 | 103 | 203 | 2024-10-03 09:45:00 | rejected |
| 4 | 104 | 204 | 2024-10-04 11:20:00 | approved |

### Таблица CreditHistory

| PK id | FK loan\_id | FK client\_id | status |
| --- | --- | --- | --- |
| 1 | 201 | 101 | opened |
| 2 | 202 | 102 | closed |
| 3 | 203 | 103 | expired |
| 4 | 204 | 104 | closed |

### Таблица InteractionHistory

| PK id | FK credit\_request\_id | FK admin\_id | processing\_date | decision |
| --- | --- | --- | --- | --- |
| 1 | 111 | 101 | 2024-10-05 09:00:00 | TRUE |
| 2 | 222 | 102 | 2024-10-06 14:30:00 | FALSE |
| 3 | 333 | 103 | 2024-10-07 11:15:00 | TRUE |
| 4 | 444 | 104 | 2024-10-08 16:45:00 | FALSE |

## Примеры запросов

### Авторизация пользователя

SELECT COUNT(\*)  
FROM Client  
WHERE id = $id  
 AND password = $password;

### Регистрация пользователя

Проверка на существование пользователя с такой же почтой

SELECT COUNT(\*)  
FROM Client  
WHERE email = $email;

Если количество равно 0, то можно продолжить регистрацию

INSERT INTO Client (name, email, password)  
VALUES ($full\_name, $email, $password);

### Авторизация администратора

SELECT COUNT(\*)  
FROM Admin  
WHERE id = $id  
 AND password = $password;

### Регистрация администратора

Проверка на существование сотрудника с такой же почтой

SELECT COUNT(\*)  
FROM Admin  
WHERE email = $email;

Если количество равно 0, то можно продолжить регистрацию

INSERT INTO Admin (name, email, password)  
VALUES ($full\_name, $email, $password);

### Поиск заявки по кредиту

Поиск доступен для любого типа параметров, приведем пример поиска по статусу:

SELECT \*  
FROM CreditRequest  
WHERE status = $status;

### Создание заявки по кредиту

INSERT INTO CreditRequest (FK\_client\_id, FK\_loan\_id, request\_time, status)  
VALUES (1, 1, '2024-10-10 12:00:00', 'processing');

### Создание нового кредита

INSERT INTO Credit (  
 loan\_name,  
 opening\_date,  
 expiration\_time,  
 amount,  
 interest\_rate,  
 monthly\_payment,  
 next\_payment\_date,  
 debt,  
 payments\_overdue,  
 co-borrowers,  
 deposit  
) VALUES (  
 'Ипотека',  
 '2024-10-10 12:00:00',  
 24,  
 400000.00,  
 16.00,  
 17500.00,  
 '2024-11-10',  
 200000.00,  
 1122334455,  
 ['Иванов И.И.'],  
 80000.00  
);

### Вынесение решения по кредиту

#### Одобрение кредита

UPDATE CreditRequest  
SET status = 'rejected'  
WHERE id = 1;

#### Отклонение кредита

UPDATE CreditRequest  
SET status = 'rejected'  
WHERE id = 1;

### Изменение данных в личном кабинете пользователя

#### Пример для смены номера телефона

UPDATE Client  
SET phone = '9876543210'   
WHERE id = 1;

#### Пример для вычисления и записи возраста в age

UPDATE Client  
SET age = EXTRACT(YEAR FROM AGE(birthdate));

### Изменение данных в личном кабинете администратора

#### Пример для смены должности

UPDATE Admin  
SET post = 'Старший администратор'  
WHERE id = 1;

#### Пример для вычисления и записи возраста в age

UPDATE Admin  
SET age = EXTRACT(YEAR FROM AGE(birthdate));

### Изменение статуса кредита в кредитной истории

UPDATE CreditHistory  
SET status = 'closed'  
WHERE id = 1;

### Просмотр информации о кредитной истории

#### Просмотр статистики по кредитам пользователя

SELECT  
 ch.id AS credit\_history\_id,  
 c.name AS client\_name,  
 c.email AS client\_email,  
 cr.loan\_name AS loan\_name,  
 cr.amount AS loan\_amount,  
 cr.interest\_rate AS loan\_interest\_rate,  
 cr.monthly\_payment AS monthly\_payment,  
 ch.status AS credit\_status,  
 cr.expiration\_time AS loan\_duration  
FROM  
 CreditHistory ch  
JOIN  
 Client c ON ch.FK\_client\_id = c.id  
JOIN  
 Credit cr ON ch.FK\_loan\_id = cr.id  
WHERE  
 cr.expiration\_time < 12;

### Массовый импорт/экспорт из бд

#### Импорт на примере таблицы Client

COPY Client(id, name, email, workplace, sex, password, phone, age, birthdate, salary, self\_employment\_status, owned\_property, marital\_status, spouse\_workplace, spouse\_salary, amount\_of\_children, rating)  
FROM '/path/to/your/client\_data.csv'  
DELIMITER ','  
CSV HEADER;

#### Экспорт на примере таблицы Client

COPY Client(id, name, email, workplace, sex, password, phone, age, birthdate, salary, self\_employment\_status, owned\_property, marital\_status, spouse\_workplace, spouse\_salary, amount\_of\_children, rating)  
TO '/path/to/exported/client\_data.csv'  
DELIMITER ','  
CSV HEADER;

# Нереляционная модель

## MongoDB

![nosql](data:text/html; charset=utf-8;base64,)

nosql

## Коллекции и сущности

n - максимально необходимый размер строки для текста. Оптимальное n = 150 символов.

### Client

{  
  
 "\_id": ObjectId, // 12  
 "name": String, // n  
 "email": String, // n  
 "workplace": String, // n  
 "sex": String, // n  
 "password": String, // n  
 "phone": String, // n  
 "age": Number, //8  
 "birthdate": Date, // 8  
 "salary": Number, // n  
 "self\_employment\_status": String, // n "self-employed", "entrepreneur", "idle"  
 "owned\_property": [String], // 10n  
 "marital\_status": String, // n "married", "single"  
 "spouse\_workplace": String, // n  
 "spouse\_salary": Number, // 8  
 "amount\_of\_children": Number,//8  
 "rating": Number, // 8  
 "credit\_history": [  
 {  
 "\_id": ObjectId, //12  
 "loan\_id": ObjectId, // 12  
 "status": String, // n "opened", "expired", "closed"  
 "closing\_date": Date //8  
 }  
 ]  
}

### Admin

{  
 "\_id": ObjectId, // 12  
 "name": String, // n  
 "email": String, // n  
 "sex": String, // n  
 "password": String, // n  
 "age": Number, // 8  
 "phone": String, // n  
 "post": String, // n  
 "birthdate": Date, // 8  
 "interaction\_history": [  
 {  
 "\_id": ObjectId, // 12  
 "credit\_request\_id": ObjectId, // 12  
 "processing\_date": Date, // 8  
 "decision": Boolean // 1  
 }  
 ]  
}

### Credit

{  
 "\_id": ObjectId, // 12  
 "loan\_name": String, // n  
 "opening\_date": Date, // 8  
 "expiration\_time": Number, // 8  
 "amount": Number, // 8  
 "interest\_rate": Number, // 8  
 "monthly\_payment": Number, // 8  
 "next\_payment\_date": Date, // 8  
 "debt": Number, // 8  
 "payments\_overdue": Number, // 8  
 "co-borrowers": [String], // 5n  
 "deposit": Number // 8  
}

### CreditRequest

{  
 "\_id": ObjectId, // 12  
 "client\_id": ObjectId, // 12  
 "loan\_id": ObjectId, // 12  
 "request\_time": Date, // 8  
 "status": String // n "processing", "approved", "rejected"  
}

## Оценка объема информации

Средний размер информации, хранимой в модели: - Client: 12 \* 3 + 8 \* 6 + n*20 = 3084 байт - Credit: 8*  9 + 12 + n \* 6 = 984 байт - CreditRequest: 8 + 12*3 + n = 194 байт - Admin: 1 + 8*  3 + 12 \* 3 + n\*6 = 961 байт

В среднем пользователь будет: - брать 5 кредитов. - отправлять количество заявок, равному 8. - каждая заявка будет обработана и хранится в истории взаимодействия с клиентом: 5.

При количестве клиентов, равных clients:

## Избыточность данных

В таблице Client избыточными является поля self\_employment\_status, age, marital\_status - 168 байт В таблице Admin избыточным является поле age - 8 байт В таблице Credit избыточным является поле monthly\_payment, next\_payment\_date - 16 байт

Тогда формула для чистого объема данных примет вид:

Тогда избыточность данных (отношение между фактическим объемом модели и “чистым” объемом данных):

## Направление роста модели

При добавлении сущности Client и Admin дополнительных сущностей не создается.

При создании сущности CreditRequest создается сущность Credit (в случае одобрения кредита) - сущность не дублирует информации; сущность CreditHistory (в случае одобрения кредита) - так же не дублируется, так как при создании пользователя она уже создана; сущность InteractionHistory (в любом случае) - не дублируется, учитывается в сущность Admin.

## Примеры данных

### Коллекция Client

{  
 "\_id": { "$oid": "60c72b2f9b1e8b001c8e4f5a" },  
 "name": "Иванов И.И.",  
 "email": "ivanov@example.com",  
 "workplace": "Программист",  
 "sex": "male",  
 "password": "secret123",  
 "phone": "1234567890",  
 "age": 30,  
 "birthdate": { "$date": "1993-05-15T00:00:00Z" },  
 "salary": 60000,  
 "self\_employment\_status": true,  
 "owned\_property": ["Квартира"],  
 "marital\_status": "single",  
 "spouse\_workplace": "Офис менеджер",  
 "spouse\_salary": 50000,  
 "amount\_of\_children": 2,  
 "rating": 7.75,  
 "credit\_history": [  
 {  
 "\_id": { "$oid": "60c72b2f9b1e8b001c8e4f5b" },  
 "loan\_id": { "$oid": "60c72b2f9b1e8b001c8e4f5c" },  
 "status": "opened"  
 }  
 ]  
}

### Коллекция Admin

{  
 "\_id": { "$oid": "60c72b2f9b1e8b001c8e4f5d" },  
 "name": "Кузнецов А.Н.",  
 "email": "kuznetsov@example.com",  
 "sex": "male",  
 "password": "pass123",  
 "age": 32,  
 "phone": "3216549870",  
 "post": "Администратор",  
 "birthdate": { "$date": "1991-04-12T00:00:00Z" },  
 "interaction\_history": [  
 {  
 "\_id": { "$oid": "60c72b2f9b1e8b001c8e4f5e" },  
 "credit\_request\_id": { "$oid": "60c72b2f9b1e8b001c8e4f5f" },  
 "processing\_date": { "$date": "2024-10-05T09:00:00Z" },  
 "decision": true  
 }  
 ]  
}

### Коллекция Credit

{  
 "\_id": { "$oid": "60c72b2f9b1e8b001c8e4f60" },  
 "loan\_name": "Ипотека",  
 "opening\_date": { "$date": "2024-01-15T09:00:00Z" },  
 "expiration\_time": 36,  
 "amount": 5000000,  
 "interest\_rate": 15.00,  
 "monthly\_payment": 14999.99,  
 "next\_payment\_date": { "$date": "2024-11-15T00:00:00Z" },  
 "debt": 300000,  
 "payments\_overdue": 1,  
 "co-borrowers": ["Иванов И.И."],  
 "deposit": 100000  
}

### Коллекция CreditRequest

{  
 "\_id": { "$oid":"60c72b2f9b1e8b001c8e4f61" },  
 "client\_id": { "$oid":"60c72b2f9b1e8b001c8e4f5a" },  
 "loan\_id": { "$oid":"60c72b2f9b1e8b001c8e4f60" },  
 "request\_time": { "$date":"2024-10-01T10:30:00Z" },  
 "status": "rejected"  
}

## Примеры запросов

### Авторизация пользователя

user = db.Client.find\_one({  
 "email": "user@example.com",  
 "password": "user\_password"  
})

### Регистрация пользователя

db.Client.insert\_one({  
 "name": "Иван Иванов",  
 "email": "ivanov@example.com",  
 "workplace": "Программист",  
 "sex": "male",  
 "password": "secret123",  
 "phone": "1234567890",  
 "age": 30,  
 "birthdate": "1993-05-15T00:00:00Z",  
 "salary": 60000,  
 "self\_employment\_status": True,  
 "owned\_property": ["Квартира"],  
 "marital\_status": "single",  
 "spouse\_workplace": "Офис менеджер",  
 "spouse\_salary": 50000,  
 "amount\_of\_children": 2,  
 "rating": 7.75,  
 "credit\_history": []  
})

### Авторизация администратора

admin = db.Admin.find\_one({  
 "email": "admin@example.com",  
 "password": "admin\_password"  
})

### Регистрация администратора

db.Admin.insert\_one({  
 "name": "Кузнецов А.Н.",  
 "email": "kuznetsov@example.com",  
 "sex": "male ",  
 "password": "password123",  
 "age": 32,  
 "phone": "3216549870",  
 "post": "Администратор",  
 "birthdate": "1991-04-12T00:00:00Z",  
 "interaction\_history": []  
})

### Поиск заявки по кредиту

credit\_request = db.CreditRequest.find\_one({  
 "client\_id": ObjectId("client\_id"),  
 "loan\_id": ObjectId("loan\_id")  
})

### Создание заявки по кредиту

db.CreditRequest.insert\_one({  
 "client\_id": ObjectId("client\_id"),  
 "loan\_id": ObjectId("loan\_id"),  
 "request\_time": "2024-10-10T12:00:00Z",  
 "status": "processing"  
})

### Создание нового кредита

db.Credit.insert\_one({  
 "loan\_name": "Ипотека",  
 "opening\_date": "2024-01-15T09:00:00Z",  
 "expiration\_time": 36,  
 "amount": 500000.00,  
 "interest\_rate": 5.00,  
 "monthly\_payment": 14999.99,  
 "next\_payment\_date": "2024-11-15T00:00:00Z",  
 "debt": 300000.00,  
 "payments\_overdue": 1,  
 "co-borrowers": None,  
 "deposit": None  
})

### Вынесение решения по кредиту

#### Одобрение кредита

db.CreditRequest.update\_one(  
 { "\_id": ObjectId("request\_id") },  
 { "$set": { "status": "approved" } }  
)

#### Отклонение кредита

db.CreditRequest.update\_one(  
 { "\_id": ObjectId("request\_id") },  
 { "$set": { "status": "rejected" } }  
)

### Изменение данных в личном кабинете пользователя

#### Пример изменения номера телефона и зарплаты в личном кабинете пользователя

db.Client.update\_one(  
 { "\_id": ObjectId("user\_id") },  
 { "$set": { "phone": "9876543210", "salary": 65000 } }  
)

#### Пример для вычисления и записи возраста в age

client\_collection = db["Client"]  
  
current\_date = datetime.now()  
  
for client in client\_collection.find({"birthdate": {"$exists": True}}):  
 birthdate = client["birthdate"]  
 age = current\_date.year - birthdate.year - ((current\_date.month, current\_date.day) < (birthdate.month, birthdate.day))  
   
 client\_collection.update\_one(  
 {"\_id": client["\_id"]},  
 {"$set": {"age": age}}  
 )

### Изменение данных в личном кабинете администратора

#### Пример изменения поста и номера телефона в личном кабинете администратора

db.Admin.update\_one(  
 { "\_id": ObjectId("admin\_id") },  
 { "$set": { "phone": "9876543210", "post": "Старший администратор" } }  
)

#### Пример для вычисления и записи возраста в age

admin\_collection = db["Admin"]  
  
for admin in admin\_collection.find({"birthdate": {"$exists": True}}):  
 birthdate = admin["birthdate"]  
 age = current\_date.year - birthdate.year - ((current\_date.month, current\_date.day) < (birthdate.month, birthdate.day))  
   
 admin\_collection.update\_one(  
 {"\_id": admin["\_id"]},  
 {"$set": {"age": age}}  
 )

### Изменение статуса кредита в кредитной истории

db.Client.update\_one(  
 { "\_id": ObjectId("client\_id"), "credit\_history.loan\_id": ObjectId("loan\_id") },  
 { "$set": { "credit\_history.$.status": "closed" } }  
)

### Просмотр статистики по кредитам пользователя

#### Пример статистики по кредитам пользователя, например, кредиты с долгом больше 100,000

pipeline = [  
 { "$match": { "\_id": ObjectId("client\_id") } },  
 { "$unwind": "$credit\_history" },  
 {  
 "$lookup": {  
 "from": "Credit",  
 "localField": "credit\_history.loan\_id",  
 "foreignField": "\_id",  
 "as": "credit\_info"  
 }  
 },  
 { "$unwind": "$credit\_info" },  
 { "$match": { "credit\_info.debt": { "$gt": 100000 } } },  
 { "$project": { "name": 1, "credit\_info.loan\_name": 1, "credit\_info.debt": 1 } }  
]  
  
stats = list(db.Client.aggregate(pipeline))

### Массовый импорт/экспорт из бд

#### Импорт на примере таблицы Client

mongoimport --db your\_db --collection Client --file path\_to\_file.json --jsonArray

#### Экспорт на примере таблицы CLient

mongoexport --db your\_db --collection Client --out path\_to\_output.json --jsonArray

# Сравнение моделей

## Удельный объем информации

Нереляционная модель имеет больший удельный объем информации по сравнению с реляционной. Это связано с тем, что в нереляционной модели данные часто дублируются для повышения скорости доступа, в то время как реляционная модель использует нормализацию для уменьшения избыточности.

### Сравнения для моделей

* Для модели Client: 5120 байт(MongoDB) vs 719 байт(PostgreSQL)
* Для модели Credit: 3000 байт(MongoDB) vs 210 байт(PostgreSQL)
* Для модели CreditRequest: 1280 байт(MongoDB) vs 40 байт(PostgreSQL)
* Для модели CreditHistory: 640 байт(MongoDB) vs 32 байта(PostgreSQL)
* Для модели InteractionHistory: 640 байт(MongoDB) vs 21 байт(PostgreSQL)
* Для модели Admin: 44068 байт(MongoDB) vs 392 байт(PostgreSQL)

## Запросы по отдельным use-cases

Количество запросов для выполнения операций в нереляционной модели будет меньше, чем в реляционной, особенно при работе с большими объемами данных.

### Сравнения количества запросов для отдельных use-cases

* Просмотр кредитной истории: 1 запрос(MongoDB) vs 3 запроса(PostgreSQL)
* Создание заявки на кредит: 1 запрос(MongoDB) vs 2 запроса(PostgreSQL)
* Авторизация/Регистрация: 1 запрос(MongoDB) vs 1 запрос(PostgreSQL)
* Поиск клиента: 1 запрос(MongoDB) vs 2 запроса(PostgreSQL)
* Обновление данных клиента: 1 запрос(MongoDB) vs 1 запрос(PostgreSQL)
* Получение статистики по кредитам: 1 запрос(MongoDB) vs 2 запроса(PostgreSQL)
* Изменение статуса кредита: 1 запрос(MongoDB) vs 1 запрос(PostgreSQL)
* Массовый импорт данных: 1 запрос(MongoDB) vs N запросов(PostgreSQL)
* Получение истории взаимодействий с заявками: 1 запрос(MongoDB) vs 3 запроса(PostgreSQL)
* Получение информации о кредитах клиента: 1 запрос(MongoDB) vs 3 запроса(PostgreSQL)

## Количество задействованных коллекций

* Просмотр кредитной истории: 1(MongoDB) vs 3(PostgreSQL)
* Создание заявки на кредит: 1(MongoDB) vs 2(PostgreSQL)
* Авторизация/Регистрация: 1(MongoDB) vs 1(PostgreSQL)
* Поиск клиента: 1(MongoDB) vs 2(PostgreSQL)
* Обновление данных клиента: 1(MongoDB) vs 1(PostgreSQL)
* Получение статистики по кредитам: 1(MongoDB) vs 2(PostgreSQL)
* Изменение статуса кредита: 1(MongoDB) vs 1(PostgreSQL)
* Массовый импорт данных: 1(MongoDB) vs N(PostgreSQL)
* Получение истории взаимодействий с заявками: 1(MongoDB) vs 3(PostgreSQL)
* Получение информации о кредитах клиента: 1(MongoDB) vs 3(PostgreSQL)

# Вывод

На основе проведенного анализа можно сделать следующие выводы: - **Удельный объем информации**: Реляционная модель демонстрирует меньший объем данных по сравнению с нереляционной моделью. - **Количество запросов**: Нереляционная модель показывает лучшие результаты по количеству запросов для большинства операций. - **Число используемых коллекций**: Нереляционная модель требует меньше коллекций для выполнения аналогичных операций. - **Упрощение масштабирования**: Нереляционные модели часто проще масштабировать горизонтально, что делает их более подходящими для распределенных систем.

Так как процесс оформления кредита - процесс не самый быстрый (необходимо анализировать профиль, оценивать рейтинг клиента и его кредитную историю), то можно пожертвовать быстродействием системы в пользу меньшего объема памяти (что особенно актуально для больших банков). Кроме того, при масштабировании системы понадобится придумывать более гибкие, компактные коллекции нереляционной модели. Таким образом, предпочтительнее выглядит использование реляционной модели.