Раздел 3. «Информационно-коммуникационные технологии»

ҒТАМР 28.17.25
ӘОЖ 004.65
DOI: 10.53002/076

В.С. Луценко, Ю.С. Клопов, А.У. Алжанов

Карагандинский индустриальный университет, г. Темиртау
(E-mail: a.alzhanov@tttu.edu.kz)

SPRING FRAMEWORK және POSTGRESQL негізіндегі микроқызметтерде деректердің дәрістілігін қамтамасыз етудің тиімді әдістері

Жұмыс Java/Spring Framework және PostgreSQL платформасына негізделген микросервис архитектурасында транзакцияларды өңдеуді оңтайландыру әдістерін қарастырады. Транзакция қақтығыстары, тығырықтанулар және үстеме шығындар сияқты бір мезгілде деректерге қол жеткізумен байланысты негізгі мәселелерді сипаттайды. Оқшаулау деңгейлерін, оптимистік және пессимистік құлыптарды қолдануды қоса алғанда, транзакцияларды басқарудың әртүрлі тәсілдеріне талдау жүргізіледі. Алынған деректер негізінде деректердің сәйкестігін қамтамасыз ету және бәсекелестік қателерді болдырмаудың тиімді әдістері ұсынылады. Зерттеу нәтижелері сенімді және тиімді деректерді өңдеуді қажет ететін жоғары жүктемелі микросервис қосымшаларын әзірлеу үшін пайдалы болуы мүмкін.

Түйін сөздер: микросервистер, транзакциялар, деректер консистенциясы, оқшаулау деңгейлері, оптимистік құлыптау, пессимистік құлыптау, параллельді бақылау, жоғары жүктеме жүйелері, импотенттілік, тұйықталулар, өнімділік, масштабтау.

Кіріспе

Микросервис архитектурасына негізделген заманауи жоғары жүктемелі қолданбалар икемділік, масштабтау және өнімділікті сақтай отырып, деректердің жоғары үйлесімділігін талап етеді. Микросервистерді пайдалану күрделі жүйелерді бөлек әзірлеуге, орналастыруға және масштабтауға болатын бөлек, тәуелсіз компоненттерге бөлуге мүмкіндік береді. Дегенмен, мұндай бөлу, әсіресе деректердің сәйкестігі мен транзакцияны басқару контекстінде жаңа қиындықтарды тудырады [3, б. 54].

Микросервис архитектурасында әрбір қолданба өзінің дерекқорларын пайдалана алады, асинхронды хабарламалар немесе REST API арқылы басқа компоненттермен әрекеттесе алады. Бұл деректер тұтастығына қауіп төндіреді, өйткені қызметтер арасындағы өзара әрекеттесулер жоғары қатарлас орталарда орын алады және жиі асинхронды болады. Мысалы, ретсіз дерекқор жазбалары және ағындар арасындағы жарыстар дұрыс емес деректерге әкелуі мүмкін, бұл әсіресе қаржылық, медициналық және аналитикалық жүйелер үшін өте маңызды.

Микросервис қосымшаларын әзірлеу үшін кеңінен қолданылатын Java фреймворктерінің бірі дерекқормен жұмысты жеңілдету және деректер күйін басқару үшін барлық құралдарды қамтамасыз ететін Spring Framework болып табылады [2, б. [225]. Дегенмен, олармен қалай жұмыс істеу керектігін түсінбеу дұрыс емес нәтижелерге және жүйе тиімділігінің төмендеуіне әкелуі мүмкін.

Бұл зерттеу Spring Framework және PostgreSQL негізіндегі микросервис архитектурасында деректердің сәйкестігін қамтамасыз ету әдістерін талдауға, сондай-ақ жоғары жүктемелі жүйелерде пайдаланудың ең тиімді тәсілдерін анықтауға арналған. Және ол қаржылық, аналитикалық және кәсіпорын шешімдері сияқты заманауи жоғары жүктемелі жүйелер үшін маңызды болып табылатын микросервис қосымшаларының тұрақтылығы мен өнімділігін арттыруға бағытталған.

Мәселені анықтау үшін қарапайым өтініш жазылды, оның жалғыз мақсаты техникалық конференция тыңдаушыларының спикерлерге берген «лайктарын» жазу болып табылады. Лайк, өз кезегінде баяндама тақырыбы бойынша жиналады. Қызмет коды 1-суретте көрсетілген.

Раздел 3. «Информационно-коммуникационные технологии»

```
public void addLikesToSpeaker(Likes likes) {
    speakersRepository.findByTalkName(likes.getTalkName())
            .ifPresentOrElse(speaker -> {
                speaker.setLikes(speaker.getLikes() + likes.getLikes());
                speakersRepository.save(speaker);
                log.info("{} likes successfully added", likes.getLikes());
            }, () -> {
                log.warn("Speaker with talk {} not found",
likes.getTalkName());
            });
}
```

Сурет 1. Қарастырылып отырған қызметтің әдіс коды.

Қызмет жұмыс істейтін деректер қоры кестесінің түрі 1-кестеде көрсетілген.

Кесте 1. Тесттерді іске қосу алдында кестенің дерекқордағы көрінісі.

id	first_name	last_name	talk_name	likes	updated
1	Myrkymbai	Kyrtymbai	Spring best practice	0	2025-02-14 13:06:13:021

Қолданбаны сынау үшін біз ашық бастапқы бағдарламалық құралды қолданамыз – Gatling және қолданбамызға арналған «Көктемдегі ең жақсы тәжірибе» есебі үшін хабар брокерінің тақырыбына ұнатулары бар 2 мың хабарлама жіберетін жазбаша тест. Қолданбаның хабарламаларын оқу 5 ағында болады. Тесттерді аяқтағаннан кейін барлық «ұнатулар» қосылмағанын, атап айтқанда шамамен 57% жоғалғанын көруге болады. 2 кестені қараңыз.

Кесте 2. Бірінші сынақтан кейін кестені мәліметтер базасында көрсету.

id	first_name	last_name	talk_name	likes	updated
1	Myrkymbai	Kyrtymbai	Spring best practice	851	2025-02-14 13:25:10:280

Бұл мәселе жазбаша өтініште транзакцияны басқарудың болмауына байланысты туындайды. Жазбаша қызмет басқа 4 ағында (хабарлар 5 ағында тұтынылады) дәл осындай әрекеттің орындалуы мүмкін екеніне назар аудармай, деректер базасынан ұнатулар өрісінің мәнін оқиды және оған сұрауда алынған ұнатулар санын қосады және қайта жазу орын алады. Яғни, бір мән бірден 5 ағында бір уақытта оқылады, бірақ өңдеуге ең көп уақыт кеткен соңғы мән ғана жазылады.

Бұл мәселені қолданбамызда транзакцияны дұрыс басқаруды орнату арқылы шешуге болады. Транзакция – бұл деректермен жұмыс істеудің логикалық бірлігін білдіретін дәйекті дерекқор операцияларының тобы. Транзакция деректер тұтастығын сақтай отырып және басқа параллельдік транзакциялардан тәуелсіз толық және сәтті орындалуы мүмкін немесе мүлде орындалмауы мүмкін, бұл жағдайда оның ешқандай әсері болмауы керек.

Spring Framework транзакциялармен жұмыс істеу үшін барлық қажетті құралдарды ұсынады, сонымен қатар дерекқормен жұмыс істеу үшін барлық қажетті абстракцияларды қамтамасыз ететін Spring Data JPA жобасын ұсынады. PostgreSQL сонымен қатар оқшаулау құралдарын пайдалана отырып, ACID (Atomicity, Consistency, Isolation, Reliability) транзакцияларын жүзеге асыруды

толығымен қолдайды [6, б. [157]. Дегенмен, сіз тек қана рамкаға сене алмайсыз, ол барлық жұмысты өзі орындамайды және оған транзакциямен қалай жұмыс істеу керектігін түсіндіру керек.

Spring Framework-тегі пайдалы құралдардың бірі @Transactional аннотациясы болып табылады, дегенмен, егер бұл аннотацияны жазбаша қолданбада дерекқор әдісіне орнатсақ, бұл жағдайды нашарлатады және біз үлкен шығындарға тап боламыз. Мәліметтер базасында транзакцияны оқшаулаудың 4 деңгейі бар, олардың әрқайсысы параллель транзакциялар арасындағы қолайлы өзара әрекеттесу дәрежесін және өнімділік пен бір мезгілде қателерден қорғау деңгейі арасындағы теңгерімдерді анықтайды:

1. Read Uncommitted – транзакция басқа транзакциямен әлі жасалмаған деректерді оқи алады (лас оқу).

2. Оқу орындалды – транзакция тек басқа транзакциялармен жасалған өзгерістерді көреді. Лас оқуларға жол берілмейді, бірақ қайталанбайтын оқулар мәселе болуы мүмкін.

3. Қайталанатын оқу – транзакция басында оқылған деректер аяқталғанға дейін өзгеріссіз қалуына кепілдік береді. Бұл лас оқуларға да, қайталанбайтын оқуларға да жол бермейді.

4. Сериализацияланатын – транзакциялардың толық сериялануын қамтамасыз ететін оқшаулаудың ең жоғары деңгейі. Барлық параллельді транзакциялар ретімен орындалғандай орындалады. Бұл деңгей деректердің максималды сәйкестігіне кепілдік береді [5, б. 47].

Сәйкесінше, пайдаланылған оқшаулау деңгейі неғұрлым жоғары болса, дерекқор соғұрлым көп жұмыс істеуі керек, осылайша өнімділікті төмендетеді, бірақ деректер сәйкестігін жақсартады. PostgreSQL Read Uncommitted функциясына қолдау көрсетпейді, ал ең төменгі және әдепкі мән Read Committed болып табылады және сипаттамадан көріп отырғаныңыздай, бұл деңгей біздің қолданбаны бір мезгілде оқу мен жазудан қорғамайды.

Қолданба конфигурациясындағы оқшаулау деңгейін Қайталанатын оқуға орнату арқылы біз іс жүзінде ешқандай шығын қалмағанын көреміз, кесте 3.

Кесте 3. Қайталанатын оқудағы екінші сынақтан кейін дерекқордағы кесте көрінісі

id	first_name	last_name	talk_name	likes	updated
1	Myrkymbai	Kyrtymbai	Spring best practice	1996	2025-02-14 14:01:04:340

Дегенмен, 4 «ұнату» деректерінің жоғалуы әлі де орын алады және қолданба қателері журналын қарасаңыз, келесі хабарды көре аласыз: «Бір мезгілде жаңартуға байланысты кіруді сериялау мүмкін болмады.» Бұл дерекқор бізге деректерді қайта жазуға мүмкіндік бермеді, бірақ деректердің өзі жоғалды. Бұл қатені Spring Retry жобасын пайдалану және қайталау әдісін қолданбамыздағы @Retry аннотациясымен белгілеу арқылы түзетуге болады, осылайша проблемалық жазу әрекетін қайталап көріңіз. Тағы бір сынақты өткізгеннен кейін дерекқорда барлық 2000 лайк орнатылғанын көруге болады.

Оқшаулау деңгейіне ауысқанда дәл осындай нәтижелер алынды - Серияланатын, өйткені ДҚ өзі барлық параллель транзакцияларды кезекке орналастырады.

Дегенмен, дерекқордағы транзакцияны оқшаулау режимін өзгертуді қажет етпейтін тағы бір шешім бар, атап айтқанда, құлыптарды пайдалану. Көп пайдаланушылық жүйелерде деректерге қол жеткізуді басқару үшін қолданылатын құлыптардың 2 түрі бар:

• Оптимистік құлыптау – транзакциялар арасындағы қайшылықтар аз болады деп болжайды. Бұл тәсіл қолданба деңгейінде жұмыс істейді және деректер нұсқасына негізделген.

• Пессимистік құлыптау транзакциялар арасында көптеген қақтығыстар болатынын болжайды және олардың алдын ала алдын алу қажет. Бұл тәсіл дерекқор деңгейінде жұмыс істейді және «ТАҢДАУ ... ҮШІН ЖАҢАРТУ» синтаксисін пайдаланады [1. бірге. 305]

Сипатталған мәселені шешу үшін оптимистік құлыптау тәсілін қолдану қолайлы емес, өйткені әдепкі бойынша бәсекелестік жоғары, бірақ эксперименттің тазалығы үшін бұл тәсілді де тексеру

қажет. Қолданбада құлыптарды қолданатын болсақ, онда оқшаулау деңгейін ең төменгі деңгейге орнатуға болады, PostgreSql жағдайында бұл оқу орындалады. Оптимистік құлыптауды жүзеге асыру үшін @Retry әдісі қалдырылды, бірақ жаңартылған өріс @Version аннотациясымен белгіленді және дерекқордан JPA репозиторийіне жазбаны шығарып алу әдісі @Lock(LockModeType.OPTIMISTIC) белгісімен белгіленді. Енді дерекқорға жаңа мән жазуға әрекет жасағанда, қолданба жаңартылған өрістің жазба оқылған кездегі мәннен қазіргі уақытта ерекшеленетінін тексереді, ал егер айырмашылықтар болса, жазбаның @Retry блогында қайталануын тудыратын ерекше жағдайды шығарады. Тестті өткізгеннен кейін біз тағы да дерекқорда 2000 лайк түрінде сәтті нәтиже алдық.

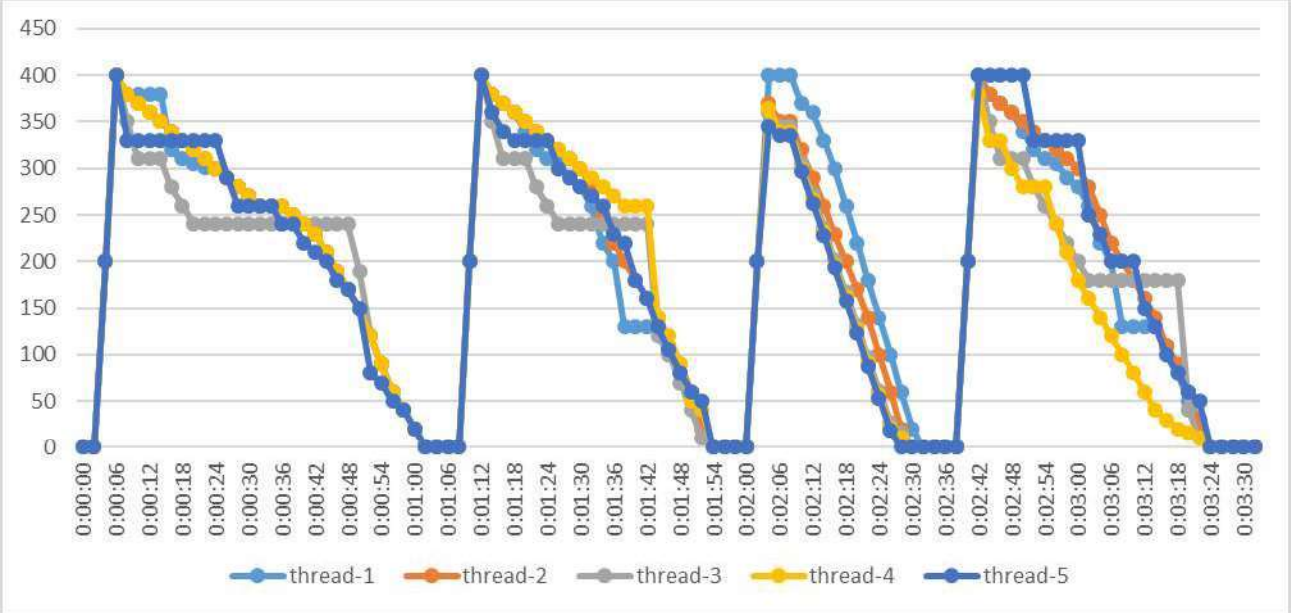
Пессимистік құлыптауды қолдану үшін @Retry блогын жоюға болады және @Version енді қажет емес. Ал дерекқордан деректерді алу әдісі @Lock(LockModeType.PESSIMISTIC_WRITE) арқылы түсіндіріледі. Соңғы сынақты орындағаннан кейін біз дерекқордың қайтадан 4-ші кестенің дәйекті күйінде екенін көреміз.

Кесте 4. Қайталанатын оқудағы екінші сынақтан кейін дерекқордағы кесте көрінісі

id	first_name	last_name	talk_name	likes	updated
1	Myrkymbai	Kyrtymbai	Spring best practice	200	2025-02-14 15:37:40:054

Сипатталған 4 әдістің әрқайсысы мәселені шешті, ол тәжірибе жүзінде расталды, бірақ әрбір тәсілдің өнімділігін бағалау қажет. 2-суретте төрт дәйекті сынақ үшін кезектегі өңделмеген хабарламалар санының уақытқа қатысты графигі көрсетілген.

Сурет 2. 5 ағындағы кезектен хабарларды талдау жылдамдығы.



Солдан оңға қарай:
1. Оқшаулау режимі – Қайталанатын оқу + Қайталау;
2. Оқшаулау режимі – Серияланатын + Қайталау;
3. Оқшаулау режимі – оқу орындалды + пессимистік құлыптау;
4. Оқшаулау режимі – оқуға берілген + оптимистік құлыптау.

Зерттеулер көрсеткендей, деректерге параллель қол жеткізудің үлкен көлемі болуы мүмкін жоғары жүктемелі ортада пессимистік құлыптау өнімділікті жақсартады (деректерді өңдеу

жылдамдығы), сонымен қатар жазылуы қажет код көлемін азайтады. Бұл ерекшеліктерді тастағаннан кейін дерекқордағы жазбаны қайталау логикасына қосымша шақырулардың болмауына байланысты. Дегенмен, параллельділік соншалықты жоғары емес орталарда оптимистік құлыптау немесе оқшаулау деңгейін жай ғана өзгерту өнімділікке көп әсер етпестен мәселені шеше алады.

Қорытынды

Қорытындылай келе, микросервис архитектурасына негізделген жоғары жүктемелі жүйелерде деректердің сәйкестігін қамтамасыз ету – аса маңызды әрі күрделі міндеттердің бірі. Зерттеу барысында Spring Framework пен PostgreSQL құралдарын пайдалана отырып, транзакцияны басқару мен құлыптау тәсілдерінің әртүрлі нұсқалары қарастырылды. Тәжірибелік сынақтар көрсеткендей, оқшаулау деңгейлерін өзгерту, қайталау механизмдерін қолдану, сондай-ақ оптимистік және пессимистік құлыптау әдістері деректердің жоғалу мәселесін шешуге мүмкіндік береді.

Әдістердің әрқайсысы нақты жағдайға байланысты өзіндік артықшылықтар мен шектеулерге ие:

1. Пессимистік құлыптау жоғары жүктемелі ортада өнімділікті сақтай отырып, деректердің жоғалуын болдырмайды және қосымша код көлемін азайтады;
2. Оптимистік құлыптау параллельділік төмен ортада тиімді болып табылады және өнімділікке айтарлықтай әсер етпейді;
3. Оқшаулау деңгейлерін арттыру транзакция қауіпсіздігін күшейтсе де, кей жағдайларда өнімділікті төмендетуі мүмкін;

Сонымен, жоғары жүктемелі микросервистік қосымшаларды әзірлеу кезінде деректердің тұтастығын сақтау үшін таңдалатын әдіс жүйенің жүктеме сипатына, параллельділік деңгейіне және өнімділік талаптарына байланысты болуы тиіс. Бұл жұмыстың нәтижелері қаржылық, медициналық және аналитикалық жүйелер сияқты деректердің дәлдігі аса маңызды салаларда тұрақты әрі сенімді шешімдерді құруға негіз бола алады.

Әдебиеттер тізімі

1. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. // 2016 Addison-Wesley Professional, C. 305- 309;
2. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind
3. Reliable Scalable, and Maintainable Systems. // 2017 O'Reilly Media, C. 225-232;
4. Newman S. Building Microservices: Designing Fine-Grained Systems. 2nd
5. Edition. // O'Reilly Media, 2021 C. 53-59;
6. PostgreSQL Global Development Group. PostgreSQL 15.10 Documentation. [Электронный ресурс]. URL: https://postgresql.org/ (дата обращения: 13.12.2024)., Chapter 13: Concurrency Control;
7. Fowler M. Refactoring: Improving the Design of Existing Code. 2nd Edition. // 2018 Addison-Wesley Professional, C. 47-55;

В.С. Луценко, Ю.С. Клопов, А.У. Алжанов

Эффективные методы обеспечения лекционности данных в МИКРОУСЛУГАХ на основе SPRING FRAMEWORK и POSTGRESQL

В работе рассматриваются методы оптимизации обработки транзакций в архитектуре микросервисов, основанной на платформе Java/Spring Framework и PostgreSQL. Описывает основные проблемы, связанные с одновременным доступом к данным, такие как конфликты транзакций, тупиковые ситуации и накладные расходы. Проводится анализ различных подходов к управлению транзакциями, включая уровни изоляции, использование оптимистических и пессимистических блокировок. На основе полученных данных предлагаются эффективные методы обеспечения согласованности данных и предотвращения конкурентных ошибок. Результаты исследования могут быть полезны для

разработки высоконагруженных микросервисных приложений, требующих надежной и эффективной обработки данных.

Ключевые слова: микросервисы, транзакции, согласованность данных, уровни изоляции, оптимистическая блокировка, пессимистическая блокировка, параллельное отслеживание, системы с высокой нагрузкой, импотенция, замыкания, производительность, масштабируемость.

V.S. Lutsenko, Y.S. Klopov, A.U. Alzhanov

Effective methods for ensuring data integrity in micro-services based on the SPRING FRAMEWORK and POSTGRESQL

The work considers methods for optimizing transaction processing in the microservice architecture based on the Java/Spring Framework and PostgreSQL platform. Describes the main problems associated with simultaneous data access, such as transaction conflicts, deadlocks, and overhead. Analysis of various ways of managing transactions is carried out, including the levels of isolation, the use of optimistic and pessimistic locks. Based on the data obtained, effective methods are proposed to ensure data compliance and avoid competitive errors. The results of the study can be useful for developing high-load microservice applications that require reliable and efficient data processing.

Keywords: microservices, transactions, data consistency, isolation levels, optimistic locking, pessimistic locking, parallel monitoring, high load systems, impotence, dead ends, performance, scaling.

References

1. Gamma E., Helm R., Johnson R., Vlissides J. Дизайн Үлгілері: Қайта пайдалануға болатын Нысанға Бағытталған Бағдарламалық Құрал Элементтері. // 2016 Addison-Wesley Professional, Бірге. 305- 309;

2. Kleppmann M. Деректерді Көп Қажет Ететін Қосымшаларды Жобалау: Артындағы Үлкен Идеялар

3. Сенімді Масштабталатын Және Техникалық қызмет көрсететін Жүйелер. // 2017 O ' Reilly Media, Бірге. 225-232;

4. Newman S. Building Микросервистері: Ұсақ Түйіршікті Жүйелерді Жобалау. 2-ші

5. Басылым. // O ' Рейли Медиа, 2021 Жылдан Бастап. 53-59;

6. PostgreSQL Жаһандық Даму Тобы. PostgreSQL 15.10 Құжаттама. [Электронный ресурс]. МЕКЕН-ЖАЙЫ: https://postgresql.org/ (қол жеткізілген күні: 13.12.2024)., 13 Тарау: Параллельді Бақылау;

7. Фаулер М. Рефакторинг: Қолданыстағы Кодтың Дизайнын Жетілдіру. 2-Ші Басылым. // 2018 Addison-Wesley Professional, Бірге. 47-55;