KOMPOZITSIYA VA AGREGATSIYA: MA'LUMOTLAR BAZALARIDA MURAKKAB MUNOSABATLARNI MODELLASHTIRISH

Mirsaid Yusupov Abdulaziz o'g'li
Farg'ona davlat universiteti Amaliy matematika va informatika kafedrasi o'qituvchisi
E-mail: mirsaidbeky@gmail.com
Ziyodaxon Qosimjonova Zohidjon qizi
Farg'ona davlat universiteti Amaliy matematika yo'nalishi 3-bosqich 23.07-guruh talabasi
E-mail: qosimjonovaziyodaxon@gmail.com

ANNOTATSIYA: Ushbu maqola relyatsion ma'lumotlar bazalari va ob'ektga yo'naltirilgan loyihalash kontekstida kompozitsiya va agregatsiya konseptsiyalarini o'rganishga bag'ishlangan. Kompozitsiya ob'ektlar o'rtasidagi murakkab bog'liqliklarni modellashtirish uchun fundamental mexanizmni ifodalaydi, bunda bitta ob'ekt boshqa ob'ektni fizik jihatdan o'z ichiga oladi va uning hayotiy siklini boshqaradi. Agregatsiya ushbu kontekstda bog'liq ob'ektlarni oldindan yuklash orqali ma'lumotlarga kirishni optimallashtirish jarayoni sifatida belgilanadi. Ishda kompozitsion munosabatlarning klassik misollari, jumladan, talabalar va kurslar, avtomobillar va dvigatellar o'rtasidagi bog'liqliklar ko'rib chiqilib, kuchli bog'liqlik va kaskadli operatsiyalar tamoyillari namoyish etiladi. Ma'lumotlarni tanlash strategiyalarini, jumladan, dangasa va faol yuklashni tahlil qilishga va ularning axborot tizimlari samaradorligiga ta'siriga alohida e'tibor qaratiladi. Tadqiqot kompozitsion semantikani hisobga olgan holda ma'lumotlar bazasi sxemalarini loyihalashning metodologik jihatlarini qamrab oladi va predmet sohasining o'ziga xosligiga qarab modellashtirish uchun optimal yondashuvlarni tanlash bo'yicha tavsiyalar taqdim etadi.

Kalit so'zlar: kompozitsiya, agregatsiya, relyatsion ma'lumotlar bazalari, ob'ektga yo'naltirilgan loyihalash, kaskadli operatsiyalar, ma'lumotlarni tanlash strategiyalari, ob'ektning hayotiy sikli

АННОТАЦИЯ: Данная статья посвящена исследованию концепций композиции и акселерации в контексте реляционных баз данных и объектно-ориентированного проектирования. Композиция представляет собой фундаментальный механизм моделирования сложных взаимосвязей между сущностями, при котором один объект физически содержит другой и управляет его жизненным циклом. Акселерация в данном контексте определяется как процесс оптимизации доступа к данным через предварительную загрузку связанных объектов. В работе рассматриваются классические примеры композиционных отношений, включая связи между студентами и курсами,

автомобилями и двигателями, демонстрирующие принципы сильной зависимости и каскадных операций. Особое внимание уделяется анализу стратегий выборки данных, включая ленивую и активную загрузку, а также их влиянию на производительность информационных систем. Исследование охватывает методологические аспекты проектирования схем баз данных, учитывающих композиционную семантику, и предлагает рекомендации по выбору оптимальных подходов к моделированию в зависимости от специфики предметной области.

Ключевые слова: композиция, акселерация, реляционные базы данных, объектно-ориентированное проектирование, каскадные операции, стратегии выборки данных, жизненный цикл объектов

ANNOTATION: This article examines the concepts of composition and acceleration in the context of relational databases and object-oriented design. Composition represents a fundamental mechanism for modeling complex relationships between entities, where one object physically contains another and manages its lifecycle. Acceleration in this context is defined as the process of optimizing data access through preloading of related objects. The paper discusses classical examples of compositional relationships, including connections between students and courses, cars and engines, demonstrating principles of strong dependency and cascading operations. Special attention is given to analyzing data fetching strategies, including lazy and eager loading, and their impact on information system performance. The research covers methodological aspects of designing database schemas that account for compositional semantics and offers recommendations for selecting optimal modeling approaches depending on domain specifics.

Keywords: composition, acceleration, relational databases, object-oriented design, cascading operations, data fetching strategies, object lifecycle

KIRISH

Zamonaviy axborot tizimlarini loyihalash va ishlab chiqish jarayonida real dunyo ob'ektlari o'rtasidagi murakkab munosabatlarni to'g'ri modellashtirish muhim ahamiyat kasb etadi. Ma'lumotlar bazalari nazariyasi va amaliyotida turli xil bog'liqlik turlari mavjud bo'lib, ularning har biri o'ziga xos semantik ma'no va texnik xususiyatlarga ega. Kompozitsiya va agregatsiya kabi konseptsiyalar ob'ektga yo'naltirilgan dasturlash paradigmalaridan kelib chiqqan bo'lsada, ular relyatsion ma'lumotlar bazalarida ham keng qo'llaniladi va muhim amaliy ahamiyatga ega.

Kompozitsiya munosabati ob'ektlar orasidagi eng kuchli bog'liqlik turlaridan biri hisoblanadi. Bu munosabatda konteyner ob'ekt boshqa ob'ektlarni o'z ichiga oladi va ularning to'liq hayotiy siklini nazorat qiladi. Konteyner ob'ekt yo'q qilinganda, unga tegishli barcha tarkibiy qismlar ham avtomatik ravishda yo'q qilinadi. Bu xususiyat

kompozitsiyani agregatsiyadan ajratib turuvchi asosiy belgi bo'lib, agregatsiyada tarkibiy qismlar mustaqil hayotiy siklga ega bo'lishi mumkin.

Ma'lumotlar bazalari kontekstida kompozitsiya munosabatini to'g'ri modellashtirish bir qator muhim masalalarni hal qilishga imkon beradi. Birinchidan, ma'lumotlar yaxlitligini ta'minlash va referensial bog'liqliklarni saqlash muhim vazifa hisoblanadi. Ikkinchidan, ma'lumotlar bazasida amalga oshiriladigan operatsiyalarning semantik to'g'riligini kafolatlash zarur. Uchinchidan, tizim samaradorligini oshirish va ma'lumotlarga kirish vaqtini optimallashtirish talablari mavjud.

Agregatsiya tushunchasi ma'lumotlar bazalari sohasida nisbatan yangi bo'lsada, u amaliy dasturlashda keng tarqalgan muammolarni hal qilishga qaratilgan. Zamonaviy ilovalarda tez-tez uchraydigan muammo - bu ma'lumotlar bazasiga haddan tashqari ko'p so'rovlar yuborish natijasida yuzaga keladigan samaradorlik muammolari. Masalan, talabalar ro'yxatini ko'rsatish va har bir talaba uchun uning kurslarini alohida so'rov bilan olish juda samarasiz yondashuvdir. Agregatsiya strategiyalari ushbu muammoni hal qilish uchun turli usullarni taklif etadi.

Ma'lumotlar bazalarida ishlashda ikkita asosiy strategiya mavjud: dangasa yuklash va faol yuklash. Dangasa yuklash yondashuvida bog'liq ma'lumotlar faqat ularga murojaat qilinganda yuklanadi. Bu usul xotiradan tejamkor foydalanishga imkon beradi, lekin ko'p hollarda ma'lumotlar bazasiga ko'proq so'rovlar yuborilishiga olib keladi. Faol yuklash strategiyasida esa barcha zarur ma'lumotlar oldindan bitta yoki bir nechta optimallashtirilgan so'rovlar orqali yuklanadi. Bu yondashuv so'rovlar sonini kamaytiradi, ammo ba'zan keraksiz ma'lumotlarni ham yuklashga olib kelishi mumkin.

Ushbu tadqiqot ishining asosiy maqsadi kompozitsiya va agregatsiya konseptsiyalarini chuqur o'rganish, ularning nazariy asoslarini tahlil qilish va amaliy qo'llanish sohasidagi eng yaxshi amaliyotlarni aniqlashdir. Tadqiqot obyekti sifatida klassik misollar - talabalar va kurslar, avtomobillar va dvigatellar - olingan bo'lib, ular kompozitsion munosabatlarning barcha asosiy xususiyatlarini namoyish etishga imkon beradi.

Kompozitsiya ob'ektga yo'naltirilgan modellashtirish va dasturlashning fundamental konseptsiyalaridan biri bo'lib, u ob'ektlar o'rtasidagi kuchli bog'liqlikni ifodalaydi. Ushbu munosabat turida bitta ob'ekt boshqa ob'ektlarni o'zining tarkibiy qismlari sifatida o'z ichiga oladi va ularning mavjudligi konteyner ob'ektga to'liq bog'liq bo'ladi. Bu munosabatni "qism-butun" yoki "has-a" munosabati deb ham ataladi.

Kompozitsiyaning asosiy xususiyatlaridan biri - bu tarkibiy qismlarning mustaqil mavjud bo'la olmasligi. Agar konteyner ob'ekt yo'q qilinsa, uning barcha tarkibiy qismlari ham avtomatik ravishda yo'q qilinishi kerak. Bu xususiyat kompozitsiyani boshqa bog'liqlik turlaridan, xususan agregatsiyadan ajratib turadi.

Agregatsiyada tarkibiy qismlar konteynerdan mustaqil ravishda mavjud bo'lishi mumkin.

Kompozitsiya munosabatining ikkinchi muhim xususiyati - ekskluzivlik tamoyilidir. Tarkibiy qism bir vaqtning o'zida faqat bitta konteynerga tegishli bo'lishi mumkin. Bu tamoyil kompozitsion ierarxiyaning aniqligini ta'minlaydi va ob'ektlarning hayotiy siklini boshqarishni soddalashtiradi. Masalan, avtomobilning dvigateli bir vaqtda faqat bitta avtomobilga tegishli bo'lishi mumkin va u boshqa avtomobilda ishlatilishi mumkin emas.

Relyatsion ma'lumotlar bazalarida kompozitsiya munosabatini amalga oshirish uchun turli xil texnik yechimlar qo'llaniladi. Eng keng tarqalgan yondashuv - tashqi kalitlar va referensial yaxlitlik cheklovlaridan foydalanish. Tashqi kalit tarkibiy qism jadvalida konteyner jadvalga havola qiluvchi maydonni ifodalaydi. Referensial yaxlitlik cheklovlari esa ma'lumotlar izchilligini ta'minlaydi va noto'g'ri havolalarning paydo bo'lishiga yo'l qo'ymaydi.

Kompozitsiya munosabatini ta'riflashda ON DELETE CASCADE cheklovidan foydalanish muhim ahamiyatga ega. Bu cheklov konteyner ob'ekt o'chirilganda barcha bog'liq tarkibiy qismlarning avtomatik o'chirilishini ta'minlaydi. Bu mexanizm kompozitsiya semantikasining to'liq amalga oshirilishini kafolatlaydi va dasturchi tomonidan qo'shimcha kod yozish zaruratini bartaraf etadi.

Ma'lumotlar bazasi dizaynida kompozitsiya munosabatini to'g'ri identifikatsiya qilish muhim vazifa hisoblanadi. Barcha "qism-butun" munosabatlari kompozitsiya bo'lavermaydi. Munosabatni kompozitsiya sifatida tasniflash uchun bir nechta mezonlarni tekshirish zarur. Birinchidan, tarkibiy qism konteynerdan ajralib mavjud bo'la oladimi yoki yo'qligini aniqlash kerak. Ikkinchidan, konteyner yo'q qilinganda tarkibiy qism ham yo'q qilinishi kerakmi degan savolga javob topish zarur. Uchinchidan, tarkibiy qism bir vaqtda bir nechta konteynerga tegishli bo'lishi mumkinmi yoki yo'qligini baholash kerak.

Kompozitsiya munosabatining semantik to'g'riligi nafaqat texnik, balki biznes logikasi nuqtai nazaridan ham muhimdir. Noto'g'ri modellashtirish ma'lumotlar yaxlitligining buzilishiga va tizim xatolariga olib kelishi mumkin. Masalan, agar talabaning kurs bilan bog'liqligini kompozitsiya sifatida modellashtirsak, talaba o'chirilganda kurs ham o'chiriladi, bu esa mantiqan noto'g'ri bo'ladi, chunki kurs boshqa talabalarga ham tegishli bo'lishi mumkin.

Kompozitsiya munosabatining boshqa muhim jihati - bu kardinallik. Bir konteyner bir yoki bir nechta tarkibiy qismga ega bo'lishi mumkin. Masalan, kitobning ko'plab sahifalari bo'lishi mumkin, lekin har bir sahifa faqat bitta kitobga tegishli. Kardinallikni to'g'ri belgilash ma'lumotlar bazasi sxemasining aniqligini ta'minlaydi va noto'g'ri ma'lumotlarning kiritilishini oldini oladi.

Kompozitsiya munosabatini amalga oshirishda normalizatsiya tamoyillariga rioya qilish ham zarur. Kompozitsion ob'ektlar odatda alohida jadvallarda saqlanadi va tashqi kalitlar orqali bog'lanadi. Bu yondashuv ma'lumotlar takrorlanishini kamaytiradi va ma'lumotlar yaxlitligini oshiradi. Ammo ba'zi hollarda, ayniqsa, juda oddiy tarkibiy qismlar uchun denormalizatsiya ham qo'llanilishi mumkin.

Zamonaviy ob'ektga yo'naltirilgan dasturlash tillarida kompozitsiya munosabati sinf strukturalari orqali tabiiy ravishda ifodalanadi. Konteyner sinf tarkibiy qism sinfining instansiyalarini o'z maydonlari sifatida saqlaydi. Ma'lumotlar bazasidan ob'ektli dunyoga o'tishda ushbu munosabatlarni to'g'ri moslash muhim vazifa hisoblanadi. Ob'ektga relyatsion mapping texnologiyalari, masalan, ORM freymvorklari, ushbu vazifani avtomatlashtirish uchun turli vositalarni taqdim etadi.

Kompozitsiya munosabatining to'g'ri modellashtirish tizimning kengaytiriluvchanligiga ham ta'sir qiladi. Yaxshi loyihalangan kompozitsion strukturalar tizimga yangi funksionallik qo'shishni osonlashtiradi va kodning qayta ishlatilishini ta'minlaydi. Shu bilan birga, haddan tashqari murakkab kompozitsion ierarxiyalar tizimni tushunish va qo'llab-quvvatlashni qiyinlashtirishi mumkin.

Talabalar va kurslar munosabati ma'lumotlar bazalarini o'rgatishda eng ko'p ishlatiladigan klassik misollardan biridir. Ushbu misol orqali turli xil munosabat turlarini, ularning xususiyatlarini va to'g'ri modellashtirish prinsiplarini ko'rsatish mumkin. Biroq, bu munosabatni kompozitsiya sifatida tasniflash mantiqan noto'g'ri bo'ladi, chunki u kompozitsiyaning asosiy mezonlariga javob bermaydi.

Talabalar va kurslar o'rtasidagi munosabat aslida ko'pdan-ko'pga munosabat bo'lib, bitta talaba bir nechta kurslarga yozilishi mumkin va bitta kursda ko'plab talabalar ishtirok etishi mumkin. Bu munosabatni amalga oshirish uchun odatda uchta jadval kerak bo'ladi: talabalar jadvali, kurslar jadvali va ro'yxatdan o'tishlar jadvali. Ro'yxatdan o'tishlar jadvali talabalar va kurslar o'rtasidagi bog'liqlikni ifodalaydi va qo'shimcha ma'lumotlarni, masalan, ro'yxatdan o'tish sanasini yoki bahoni saqlashi mumkin.

Agar biz talaba va kurs munosabatini kompozitsiya sifatida modellashtirsak, bu jiddiy muammolarga olib keladi. Talaba o'chirilganda uning kurslari ham o'chirilishi kerak degan mantiqqa kelamiz, bu esa to'g'ri emas, chunki kurslar boshqa talabalarga ham tegishli. Xuddi shunday, kurs o'chirilganda talabalarni o'chirish ham mantiqan noto'g'ri. Bu misoldan ko'rinadiki, talaba va kurs mustaqil ob'ektlar bo'lib, ular o'zaro bog'langan, lekin bir-biriga bog'liq emas.

Talabalarning kurslarga yozilishi agregatsiya munosabatiga yaqinroq bo'lib, bu munosabatda ob'ektlar o'zaro bog'langan, ammo mustaqil hayotiy siklga ega. Talaba universitetdan ketganda, uning ro'yxatdan o'tish yozuvlari o'chirilishi mumkin, lekin kurslarning o'zi saqlanadi. Xuddi shunday, kurs yakunlanganda yoki bekor qilinganda,

ro'yxatdan o'tish yozuvlari o'chirilishi mumkin, ammo talabalar ma'lumotlari saqlanadi.

Biroq, talabalar va kurslar munosabati kontekstida kompozitsion ob'ektlarni ham topish mumkin. Masalan, talabaning shaxsiy ma'lumotlari va kontakt ma'lumotlari o'rtasida kompozitsiya mavjud bo'lishi mumkin. Agar talaba ma'lumotlar bazasidan o'chirilsa, uning barcha shaxsiy ma'lumotlari, telefon raqamlari va elektron pochta manzillari ham o'chirilishi kerak. Bu klassik kompozitsiya munosabatining namunasi.

Kurslar kontekstida ham kompozitsion munosabatlarni topish mumkin. Masalan, kurs va uning mavzulari o'rtasida kompozitsiya mavjud bo'lishi mumkin. Kurs o'chirilganda, uning barcha mavzulari, darslik rejalari va o'quv materiallari ham o'chirilishi kerak. Bu elementlar kursdan tashqarida mustaqil ma'noga ega emas va faqat ma'lum bir kurs kontekstida mavjud bo'lishi mumkin.

Ro'yxatdan o'tishlar jadvali o'zi ham kompozitsion elementlarni o'z ichiga olishi mumkin. Masalan, har bir ro'yxatdan o'tish bilan bog'liq baholar, davomatlar va topshiriqlarni bajarish ma'lumotlari ro'yxatdan o'tish yozuviga kompozitsion bog'liq. Ro'yxatdan o'tish o'chirilganda, barcha bog'liq baholar va davomatlar ham o'chirilishi kerak.

Ma'lumotlar bazasida ushbu munosabatlarni to'g'ri modellashtirish uchun tashqi kalitlar va tegishli referensial yaxlitlik cheklovlaridan foydalanish zarur. Talaba va kurs o'rtasidagi munosabat uchun ON DELETE CASCADE ishlatilmaydi, chunki bu munosabat kompozitsiya emas. Biroq, talaba va uning shaxsiy ma'lumotlari o'rtasida ON DELETE CASCADE ishlatilishi kerak, chunki bu kompozitsiya munosabati.

Amaliy dasturlashda talabalar va kurslar bilan ishlash tez-tez agregatsiya masalalarini keltirib chiqaradi. Masalan, talabalar ro'yxatini va har bir talabaning kurslarini ko'rsatish kerak bo'lsa, dangasa yuklash yondashuvida har bir talaba uchun alohida so'rov yuboriladi. Agar ro'yxatda yuz ta talaba bo'lsa, ma'lumotlar bazasiga yuz birta so'rov yuboriladi: bitta talabalar ro'yxatini olish uchun va yuz ta har bir talabaning kurslarini olish uchun. Bu keng tarqalgan masala "N plus one query problem" deb nomlanadi.

Faol yuklash strategiyasidan foydalanib, barcha zarur ma'lumotlarni ikki yoki uch ta so'rov orqali olish mumkin. Birinchi so'rov bilan talabalar ro'yxatini olamiz, ikkinchi so'rov bilan barcha kerakli kurslarni olamiz va uchinchi so'rov bilan ro'yxatdan o'tishlar ma'lumotlarini olamiz. Keyin dastur darajasida bu ma'lumotlarni birlashtirish mumkin. Bu yondashuv so'rovlar sonini sezilarli darajada kamaytiradi va tizim samaradorligini oshiradi.

Zamonaviy ORM freymvorklari, masalan, Hibernate yoki Entity Framework, talabalar va kurslar kabi ko'pdan-ko'pga munosabatlarni modellashtirish uchun qulay mexanizmlarni taqdim etadi. Ular annotatsiyalar yoki konfiguratsiya fayllar orqali munosabat turini belgilash, yuklash strategiyasini tanlash va kaskadli operatsiyalarni

sozlash imkonini beradi. Dasturchi bu vositalardan to'g'ri foydalanib, samarali va to'g'ri ishlaydigan tizim yaratishi mumkin.

Talabalar va kurslar misolida muhim jihat - bu munosabat semantikasini to'g'ri tushunish va tegishli texnik yechimlarni tanlashdir. Kompozitsiya va agregatsiya o'rtasidagi farqni anglash, munosabat kardinalligini to'g'ri belgilash, yuklash strategiyalarini optimal tanlash - bularning barchasi muvaffaqiyatli tizim yaratishning muhim qismlaridir.

Avtomobil va dvigatel munosabati kompozitsiyaning klassik va to'g'ri misoli hisoblanadi. Bu munosabatda dvigatel avtomobilning ajralmas tarkibiy qismi bo'lib, u avtomobildan tashqarida mustaqil ma'noga ega emas. Dvigatel avtomobilning funktsional asosini tashkil etadi va avtomobil yo'q qilinganda dvigatel ham yo'q qilinishi mantiqiy jihatdan to'g'ri.

Avtomobil va dvigatel o'rtasidagi munosabat kompozitsiyaning barcha asosiy xususiyatlariga javob beradi. Birinchidan, dvigatel avtomobildan mustaqil mavjud bo'lmaydi - ma'lumotlar bazasi kontekstida bu dvigatel yozuvi faqat avtomobil yozuviga bog'liq ravishda mavjud bo'lishini anglatadi. Ikkinchidan, bir dvigatel faqat bitta avtomobilga tegishli bo'lishi mumkin - bu ekskluzivlik tamoyili. Uchinchidan, avtomobil o'chirilganda uning dvigateli ham o'chirilishi kerak - bu hayotiy sikl bog'liqligi.

Ma'lumotlar bazasida ushbu munosabatni modellashtirish uchun ikkita jadval yaratiladi: avtomobillar jadvali va dvigatellar jadvali. Dvigatellar jadvalida avtomobil identifikatoriga havola qiluvchi tashqi kalit mavjud bo'lib, bu kalit ON DELETE CASCADE cheklovi bilan ta'minlanadi. Bu cheklov avtomobil o'chirilganda uning dvigateli ham avtomatik o'chirilishini ta'minlaydi.

Avtomobillar jadvali odatda quyidagi maydonlarni o'z ichiga oladi: avtomobil identifikatori, markasi, modeli, yili, rangi, davlat raqami va boshqa umumiy xususiyatlar. Dvigatellar jadvali esa dvigatel identifikatori, avtomobil identifikatori (tashqi kalit), dvigatel turi, hajmi, quvvati, yoqilg'i turi va boshqa dvigatelga xos xususiyatlarni saqlaydi. Tashqi kalit maydonida UNIQUE cheklovi bo'lishi mumkin, bu bir avtomobilga faqat bitta dvigatel tegishli bo'lishini ta'minlaydi.

Kompozitsion munosabatning afzalligi shundaki, u biznes logikasini ma'lumotlar bazasi darajasida ta'minlaydi. Dasturchi avtomobilni o'chirish operatsiyasini amalga oshirganda, dvigatellni alohida o'chirish kodini yozish shart emas. Ma'lumotlar bazasi mexanizmlari avtomatik ravishda barcha bog'liq yozuvlarni o'chiradi. Bu kodning soddaligini ta'minlaydi va xatolarga yo'l qo'yish ehtimolini kamaytiradi.

Avtomobil misolida boshqa kompozitsion elementlarni ham ko'rsatish mumkin. Masalan, avtomobilning g'ildiraklari, o'rindiqlari, eshiklari va boshqa tarkibiy qismlari ham kompozitsion munosabatda bo'lishi mumkin. Agar ma'lumotlar bazasida bu

elementlar haqida batafsil ma'lumot saqlash zarur bo'lsa, ular uchun alohida jadvallar yaratiladi va avtomobillar jadvaliga kompozitsion bog'lanadi.

Amaliy dasturlashda avtomobil ob'ekti bilan ishlashda agregatsiya muhim ahamiyatga ega. Avtomobilni yuklaganda uning dvigatelini ham yuklash kerak bo'ladi, chunki avtomobil haqidagi to'liq ma'lumot dvigatelsiz yaroqsiz. Bu faol yuklash strategiyasini qo'llash uchun aniq misol. Agar dangasa yuklash qo'llansa, avtomobilni yuklab, keyin uning dvigateliga murojaat qilganda ikkinchi so'rov yuboriladi. Bu avtomobil misoli uchun samarasiz yondashuv hisoblanadi.

ORM freymvorklarida avtomobil va dvigatel munosabatini modellashtirish uchun bir-biriga bog'liq annotatsiyalar ishlatiladi. Masalan, Java Persistence API da CascadeType punktlari orqali qaysi operatsiyalar dvigatellga ham qo'llanilishini belgilash mumkin. CascadeType ALL belgisi barcha operatsiyalar (saqlash, yangilash, o'chirish) dvigatelga ham qo'llanilishini anglatadi.

Avtomobil va dvigatel munosabatida kardinallik muhim omil. Odatda bir avtomobilga bitta dvigatel to'g'ri keladi, bu bir-biriga munosabat. Biroq, ba'zi maxsus avtomobillarda ikki yoki undan ortiq dvigatel bo'lishi mumkin. Bunday holatlarda kardinallik bir-ko'pga o'zgaradi, lekin kompozitsiya xususiyati saqlanadi - dvigatellar hali ham avtomobilga bog'liq va u bilan birga yo'q qilinadi.

Kompozitsion munosabatni to'g'ri amalga oshirish ma'lumotlar yaxlitligini ta'minlash va tizim ishonchliligini oshirish uchun zarur. Avtomobil va dvigatel misoli kompozitsiyaning barcha asosiy tamoyillarini yaqqol namoyish etadi va bu konseptsiyani tushunish uchun ideal modeldir. Real loyihalarda shunga o'xshash munosabatlarni identifikatsiya qilish va to'g'ri modellashtirish tizim sifatiga sezilarli ta'sir qiladi.

Ma'lumotlar bazasi dizayni va kompozitsion munosabatlar tizimning kengaytiriluvchanligiga katta ta'sir qiladi. Yaxshi loyihalangan kompozitsion strukturalar tizimga yangi funksionallik qo'shishni osonlashtiradi va kodni qayta ishlatishni ta'minlaydi. Tizim rivojlanishi davomida kompozitsion munosabatlarni qo'llab-quvvatlash va kengaytirish muhim vazifa hisoblanadi.

Abstraction kompozitsion strukturalarni kengaytiruvchan qilishning asosiy mexanizmi. Interface yoki abstract class orqali tarkibiy qismlar turini umumlashtirish mumkin. Masalan, avtomobilning engine interface ga ega bo'lishi mumkin va turli xil dvigatel turlari bu interface ni amalga oshiradi: petrol engine, diesel engine, electric engine. Bu yangi dvigatel turlarini qo'shishni osonlashtiradi.

Inheritance kompozitsion ierarxiyalarni modellashtirish uchun ishlatilishi mumkin. Biroq, inheritance bilan kompozitsiya ni aralashtirib yubormaslik kerak. Inheritance "is-a" munosabatni ifodalaydi, kompozitsiya esa "has-a" munosabatni. Masalan, electric car avtomobilning subclass i bo'lishi mumkin va u electric engine ga ega bo'lishi mumkin. Bu ikkala konseptsiyani birga qo'llashdir.

Polymorphism turli xil tarkibiy qismlar bilan ishlashni osonlashtiradi. Konteyner kod tarkibiy qismning aniq turini bilmasa ham, umumiy interface orqali u bilan ishlashi mumkin. Bu kodni moslashuvchan va kengaytiruvchan qiladi. Yangi tarkibiy qism turlari qo'shilganda mavjud konteyner kodi o'zgartirilishi shart emas.

Plugin arxitekturasi kompozitsion tizimlarni juda kengaytiruvchan qiladi. Tarkibiy qismlar pluginlar sifatida amalga oshiriladi va dinamik ravishda yuklanadi. Tizimga yangi funksionallik yangi pluginlar qo'shish orqali qo'shiladi. Bu arxitektura ayniqsa katta va murakkab tizimlarda foydali, chunki u modulyarlikni ta'minlaydi.

Configuration-driven approach kompozitsion strukturalarni kodda hardcode qilmasdan configuration orqali boshqarish imkonini beradi. Qaysi tarkibiy qismlar mavjudligi va ular qanday bog'langani configuration fayllarida yoki ma'lumotlar bazasida saqlanadi. Bu tizimni qayta compile qilmasdan o'zgartirish imkonini beradi va deployment ni osonlashtiradi.

Versioning kompozitsion strukturalarni rivojlantirish jarayonida muhim. Agar tarkibiy qism strukturasi o'zgarsa, versiyani boshqarish zarur. API versioning, schema versioning va data migration strategiyalari tizimning barqarorligini ta'minlaydi. Eski versiyalar bilan backward compatibility ni saqlash muhim.

Migration strategiyalari ma'lumotlar bazasi strukturasi o'zgarganda qo'llaniladi. Kompozitsion munosabatlar o'zgarganda migration scriptlar ehtiyotkorlik bilan yozilishi kerak. Zero-downtime migration ni ta'minlash uchun blue-green deployment yoki canary deployment strategiyalari qo'llanilishi mumkin. Migration scriptlarni production ga qo'llashdan oldin test muhitida sinab ko'rish majburiy.

Feature toggles yangi funksionallikni bosqichma-bosqich chiqarish imkonini beradi. Yangi kompozitsion elementlar yoki yangi yuklash strategiyalari feature toggle orqali yoqilishi yoki o'chirilishi mumkin. Bu xavfni kamaytiradi va muammolar yuzaga kelganda tezda orqaga qaytish imkonini beradi.

Monitoring va observability tizim rivojlanishi davomida muhim. Yangi kompozitsion elementlar qo'shilganda yoki mavjudlari o'zgartirilganda, ularning samaradorligi va to'g'ri ishlashi monitoring qilinishi kerak. Metrics, logs va traces tizim holatini tushunish va muammolarni tezda aniqlash uchun zarur.

Documentation tizim rivojlanishi bilan birga yangilanishi kerak. Kompozitsion munosabatlar, ularning semantikasi va texnik amalga oshirilishi aniq dokumentatsiyalangan bo'lishi kerak. Architecture decision records yangi dizayn qarorlarini va ularning sabablarini qayd qiladi. Bu yangi dasturchilar uchun tizimni tushunishni osonlashtiradi.

XULOSA

Kompozitsiya va agregatsiya zamonaviy ma'lumotlar bazalari va dasturiy ta'minot arxitekturasida muhim rol o'ynaydi. Ushbu tadqiqot ishida biz kompozitsiyaning nazariy asoslarini, amaliy qo'llanish usullarini va agregatsiya

strategiyalarini chuqur o'rgandik. Tadqiqot natijalari shuni ko'rsatdiki, kompozitsion munosabatlarni to'g'ri modellashtirish va optimal yuklash strategiyalarini tanlash tizim sifati va samaradorligiga sezilarli ta'sir qiladi.

Kompozitsiya ob'ektlar o'rtasidagi eng kuchli bog'liqlik turi bo'lib, tarkibiy qismlarning hayotiy sikli konteyner ob'ektga to'liq bog'liq. Bu munosabat turini to'g'ri identifikatsiya qilish va amalga oshirish ma'lumotlar yaxlitligini ta'minlaydi va biznes logikasini to'g'ri ifodalaydi. Talabalar va kurslar misolida ko'rsatilganidek, barcha qism-butun munosabatlari kompozitsiya bo'lavermaydi va farqni tushunish muhimdir.

Avtomobil va dvigatel misoli kompozitsiyaning klassik namunasini taqdim etadi. Bu munosabatda dvigatel avtomobilning ajralmas qismi bo'lib, avtomobil yo'q qilinganda dvigatel ham yo'q qilinadi. Ma'lumotlar bazasida ON DELETE CASCADE cheklovi orqali bu semantika avtomatik ta'minlanadi. Bu yondashuv kodni soddalashtiradi va xatolar ehtimolini kamaytiradi.

Foydalanilgan Adabiyotlar

1. Budd T. An Introduction to Object-Oriented Programming. Third Edition. Addison-Wesley Professional. 2002. 592 p.
2. Date C.J. An Introduction to Database Systems. Eighth Edition. Addison-Wesley. 2003. 1024 p.
3. Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley Professional. 2003. 560 p.
4. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley Professional. 2002. 560 p.
5. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional. 1994. 416 p.
6. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O'Reilly Media. 2017. 616 p.
7. Newman S. Building Microservices: Designing Fine-Grained Systems. Second Edition. O'Reilly Media. 2021. 612 p.
8. Ramakrishnan R., Gehrke J. Database Management Systems. Third Edition. McGraw-Hill. 2003. 1065 p.