O'ZBEKISTON RESPUBLIKASI AXBOROT TEXNOLOGIYALARI VA KOMMUNIKATSIYALARINI RIVOJLANTIRISH VAZIRLIGI

MUHAMMAD AL-XORAZMIY NOMIDAGI TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI

X.N.Zaynidinov, J.T.Usmonov,
SH.B.Redjepov, I.Yusupov.

MA'LUMOTLAR BAZASI

Oliy o'quv yurtlari uchun darslik.

O'zbekiston Respublikasi Oliy va O'rta-maxsus ta'lim vazirligi tomonidan Oliy o'quv yurtlari talabalari uchun darslik sifatida tavsiya etilsin.

Toshkent 2019

UDK:000.0(000.0)

BBK:00.000-00

ZOO

X.N.Zaynidinov, J.T.Usmonov, SH.B.Redjepov, I.Yusupov. Ma'lumotlar bazasi // Al-Xorazmiy Nomidagi Toshkent Axborot Texnologiyalari Universiteti. - T.,2019. -120b.

ISBN 978-9943-5487-8-7

Ushbu darslik o'quvchilarga ma'lumotlar bazasining asosiy tushunchalari, ularning tuzilmasi, mohiyat-aloqa diagrammasini qurishni o'rgatadi. Shuningdek, darslikda ma'lumotlar bazasining SQL tili va uning so'rovlarini tuzish, ular ustida amallar bajarish va so'rovlarni amalga oshirish asoslari keltirilgan.

Darslik quyidagi - Kompyuter injiniring ("Kompyuter injiniringi", "AT-Servis", "Multimediya texnologiyalari"), Dasturiy injiniring Telekommunikatsiya texnologiyalari ("Telekommunikatsiyalar", "Teleradioeshittirish", "Mobil tizimlar"), Axborot xavfsizligi (Axborot, kommunikatsiya texnologiyalari va servis), Televizion texnologiyalar ("Audiovizual texnologiyalari", "Telestudiya tizimlari va ilovalari"), AKT sohasida iqtisodiyot va menejment, Pochta aloqasi texnologiyasi, AKT sohasida kasbiy ta'lim, Axborotlashtirish va kutubxonashunoslik kabi yo'nalishlarda taxsil olayotgan talabalar uchun mo'ljallangan.

UDK:000.0(000.0)

BBK:00.000-00

Taqrizchilar:

Donayev S.B. - Ph.D. I.A.Karimov nomidagi Toshkent davlat texnika universiteti "Axborotlarga ishlov berish va boshqarish tizimlari" kafedrasi dotsenti

JumanovJ.X.- t.f.d., professor. Muhammad al-Xorazmiy nomidagi Toshkent Axborot Texnologiyalari Universiteti, "Kompyuter tizimlari" kafedrasi mudiri

MUNDARIJA

Kirish

Hozirgi kunda barcha sohalarda ma'lumotlar bazasi (MB) kerakli axborotlarni saqlash va undan oqilona foydalanishda juda muhim rol o'ynamoqda. Jamiyat taraqqiyotining qaysi jabhasiga nazar solmaylik o'zimizga kerakli ma'lumotlarni olish uchun, albatta, MBga murojaat qilishga majbur bo'lamiz. Demak, MBni tashkil qilish axborot almashuv texnologiyasining eng dolzarb hal qilinadigan muammolaridan biriga aylanib borayotgani davr taqozasidir. Axborot texnologiyalarining rivojlanishi va axborot oqimlarining tobora ortib borishi, ma'lumotlarning tez o'zgarishi kabi holatlar insoniyatni bu ma'lumotlarni o'z vaqtida qayta ishlash choralarining yangi usullarini qidirib topishga undamoqda. Ma'lumotlarni saqlash, uzatish va qayta isMash uchun MBni yaratish, so'ngra undan keng foydalanish bugungi kunda dolzarb bo'lib qolmokda. Moliya, ishlab chiqarish, savdo-sotiq va boshqa korxonalar ishlarini ma'lumotlar bazasisiz tasawur qilib bo'lmaydi.

Ma'lumki MB tushunchasi fanga kirib kelgunga qadar, ma'lumotlardan turli ko'rinishda foydalanish juda qiyin edi. Dastur tuzuvchilar ma'lumotlarini shunday tashkil qilar edilarki, u faqat qaralayotgan masala uchungina o'rinli bo'lardi. Har bir yangi masalani hal qilishda ma'lumotlar qaytadan tashkil qilinar va bu hoi yaratilgan dasturlardan foydalanishni qiyinlashtirar edi.

Har qanday axborot tizimining maqsadi real muhit ob'ektlari haqidagi ma'lumotlarga ishlov berishdan iborat. Keng ma'noda ma'lumotlar bazasi - bu qandaydir bir predmet sohasidagi real muhitning aniq ob'ektlari haqidagi ma'lumotlar to'plamidir. Predmet sohasi deganda avtomatlashtirilgan boshqarishni tashkil qilish uchun o'rganilayotgan real muhitning ma'lum bir qismi tushiniladi. Masalan, korxona, zavod, ilmiy tekshirish instituti, oliy o'quv yurti va boshqalar. Shuni qayd qilish lozimki, MBni yaratishda ikkita muhim shartni hisobga olmoq zarur:

Birinchidan, ma'lumotlar turi, ko'rinishi, ularni qo'llaydigan programmalarga bog'liq bo'lmasligi lozim, Ya'ni MBga yangi ma'lumotlarni kiritganda yoki

Kirish

Hozirgi kunda barcha sohalarda ma'lumotlar bazasi **(MB)** kerakli axborotlarni saqlash va undan oqilona foydalanishda juda muhim rol o'ynamoqda. Jamiyat taraqqiyotining qaysi jabhasiga nazar solmaylik o'zimizga kerakli ma'lumotlarni olish uchun, albatta, MBga murojaat qilishga majbur bo'lamiz. Demak, MBni tashkil qilish axborot almashuv texnologiyasining eng dolzarb hal qilinadigan muammolaridan biriga aylanib borayotgani davr taqozasidir. Axborot texnologiyalarining rivojlanishi va axborot oqimlarining tobora ortib borishi, ma'lumotlarning tez o'zgarishi kabi holatlar insoniyatni bu ma'lumotlarni o'z vaqtida qayta ishlash choralarining yangi usullarini qidirib topishga undamoqda. Ma'lumotlarni saqlash, uzatish va qayta ishlash uchun MBni yaratish, so'ngra undan keng foydalanish bugungi kunda dolzarb bo'lib qolmokda. Moliya, ishlab chiqarish, savdo-sotiq va boshqa korxonalar ishlarini ma'lumotlar bazasisiz tasavvur qilib bo'lmaydi.

Ma'lumki MB tushunchasi fanga kirib kelgunga qadar, ma'lumotlardan turli ko'rinishda foydalanish juda qiyin edi. Dastur tuzuvchilar ma'lumotlarini shunday tashkil qilar edilarki, u faqat qaralayotgan masala uchungina o'rinli bo'lardi. Har bir yangi masalani hal qilishda ma'lumotlar qaytadan tashkil qilinar va bu hol yaratilgan dasturlardan foydalanishni qiyinlashtirar edi.

Har qanday axborot tizimining maqsadi real muhit ob'ektlari haqidagi ma'lumotlarga ishlov berishdan iborat. Keng ma'noda ma'lumotlar bazasi - bu qandaydir bir predmet sohasidagi real muhitning aniq ob'ektlari haqidagi ma'lumotlar to'plamidir. Predmet sohasi deganda avtomatlashtirilgan boshqarishni tashkil qilish uchun o'rganilayotgan real muhitning ma'lum bir qismi tushiniladi. Masalan, korxona, zavod, ilmiy tekshirish instituti, oliy o'quv yurti va boshqalar. Shuni qayd qilish lozimki, **MB**ni yaratishda ikkita muhim shartni hisobga olmoq zarur:

Birinchidan, ma'lumotlar turi, ko'rinishi, ularni qo'llaydigan programmalarga bog'liq bo'lmasligi lozim, Ya'ni **MB**ga yangi ma'lumotlarni kiritganda yoki

ma'lumotlar turini o'zgartirganda, programmalarni o'zgartirish talab etilmasligi lozim.

Ikkinchidan, MBdagi kerakli ma'lumotni bilish yoki izlash uchun biror programma tuzishga hojat qolmasin. Shuning uchun ham **MB**ni tashkil etishda ma'lum qonun va qoidalarga amal qilish lozim. Bundan buyon **axborot** so'zini **ma'lumot** so'zidan farqlaymiz, Ya'ni **axborot** so'zini umumiy tushuncha sifatida qabul qilib, **ma'lumot** deganda aniq bir belgilangan narsa yoki hodisa sifatlarini nazarda tutamiz. Ma'lumotlar bazasini yaratishda, foydalanuvchi axborotlarni turli belgilar bo'yicha tartiblashga va ixtiyoriy belgilar birikmasi bilan tanlanmani tez olishga intiladi.

Ma'lumotlar bazasidan foydalanuvchilar turli amaliy dasturlar, dasturiy vositalari, predmet sohasidagi mutaxassislar bo'lishi mumkin. Ma'lumotlar bazasining zamonaviy texnologiyasida ma'lumotlar bazasini yaratish, uni dolzarb holatda yuritishni va foydalanuvchilarga undan axborot olishini ta'minlovchi maxsus dasturiy vosita, Ya'ni ma'lumotlar bazasini boshqarish tizimi yordami bilan markazlashtirilgan holda amalga oshirishni nazarda tutadi. Ma'lumotlar bazasi - EHM xotirasiga yozilgan ma'lum bir strukturaga ega, o'zaro bog'langan va tartiblangan ma'lumotlar majmuasi bo'lib, u biror bir ob'ektning xususiyatini, holatini yoki ob'ektlar o'rtasidagi munosabatni ma'lum ma'noda ifodalaydi. MB foydalanuvchiga strukturalashtirilgan ma'lumotlarni saqlash va ishlatishda optimal qulaylikni yaratib beradi.

Ma'lumki ma'lumotlarni kiritish va ularni qayta ishlash jarayoni katta hajmdagi ish bo'lib ko'p mehnat va vaqt talab qiladi. MB bilan ishlashda undagi ma'lumotlarning aniq bir strukturagi ega bo'lishi, birinchidan foydalanuvchiga ma'lumotlarni kiritish va qayta ishlash jarayonida undagi ma'lumotlarni tartiblashtirish, ikkinchidan kerakli ma'lumotlarni izlash va tez ajratib olish kabi qulayliklarni tug'diradi. MB tushunchasi fanga kirib kelgunga qadar, ma'lumotlardan turli ko'rinishlarda foydalanish juda qiyin edi. Bugungi kunda turli ko'rinishdagi ma'lumotlardan zamonaviy kompbyuterlarda birgalikda foydalanish

va ularni qayta ishlash masalasi hal qilindi. Kompbyuterlarda saqlanadigan MB maxsus formatga ega bo'lgan muayyan tuzilmali fayl bo'lib, undagi ma'lumotlar o'zaro bog'langan va tartiblangandir.

Demak, ma'lumotlar bazasi deganda ma'lum bir strukturada saqlanadigan ma'lumotlar to'plami tushuniladi. Boshqacha qilib aytganda MB - bu ma'lum berilgan aniq bir strukturaga ega bo'lgan ma'lumotlarni o'z ichiga oluvchi maxsus formatga ega bo'lgan fayldir. Ma'lumotlarni strukturalashtirish - bu shunchaki ma'lumotlarni tasvirlashda qandaydir moslikni kiritish usulidir. Odatda MB ma'lum bir ob'ekt sohasini ifodalaydi va uning ma'lumotlarni o'z ichiga oladi, ularni saqlaydi va foydalanuvchiga ma'lumotlarni qayta ishlashda undan foydalanish imkonini yaratib beradi.

Ma'lumotlar bazasi - bu ma'lum bir predmet sohasiga oid tizimlashtirilgan ma'lumotlarning nomlangan to'plamidir.

Ma'lumotlar bazasi - axborot tmzimlarining eng asosiy tarkibiy qismi bo'lib hisoblanadi. Ma'lumotlar bazasidan foydalanish uchun foydalanuvchi ishini engillashtirish maqsadida ma'lumotlar bazasini boshqarish trizimlari yaratilgan.

Ma'lumotlar bazasini boshqarish tizimi (MBBT) -bu dasturiy va apparat vositalarining murakkab majmuasi bo'lib, ular yordamida foydalanuvchi ma'lumotlar bazasini yaratish va shu bazadagi ma'lumotlar ustida ish yuritishi mumkin.

1-mavzu. Ma'lumotlar bazasining maqsadi, vazifalari va asosiy tushunchalari

Reja

1. Ma'lumotlar bazasi haqida tushuncha.
2. Ma'lumotlar bazasining asosiy terminlari.
3. Ma'lumotlar bazasiga qo'yiladigan talablar.
4. Avtomatlashgan axborot tizimlari.

Tayanch so'zlar: tizilmalashtirish, baza, texnologiya.

Hozirgi kunda inson faoliyatida ma'lumotlar bazasi (MB) kerakli axborotlarni saqlash va undan oqilona foydalanishda juda muhim rol o'ynamoqda. Jamiyat taraqqiyotining qaysi jabhasiga nazar solmaylik o'zimizga kerakli ma'lumotlarni olish uchun MBga murojaat qilishga to'g'ri keladi. Demak, MBni tashkil qilish axborot texnologiyalarining eng dolzarb hal qilinadigan muammolaridan biriga aylanib borayotgani davr taqozasidir.

Axborot texnologiyalarning rivojlanishi va axborot oqimlarini tobora ortib borishi, ma'lumotlarning tez o'zgarishi kabi holatlar insoniyatni bu ma'lumotlarni o'z vaqtida qayta ishlash choralarining yangi usullarini qidirib topishga undamoqda. Ma'lumotlarni saqlash, uzatish va qayta ishlash uchun MBni yaratish, so'ngra undan keng foydalanish bugungi kunda dolzarb bo'lib qolmoqda. Moliya, ishlab chiqarish, savdo-sotiq va boshqa sohalarni ma'lumotlar bazasisiz tasavvur qilib bo'lmaydi.

Ma'lumki, MB tushunchasi fanga kirib kelgunga qadar, ma'lumotlardan turli ko'rinishda foydalanish juda qiyin edi. Dastur tuzuvchilar ma'lumotlarini shunday tashkil qilar edilarki, u faqat qaralayotgan masala uchungina o'rinli bo'lardi. Har bir yangi masalani hal qilishda ma'lumotlar qaytadan tashkil qilinar va bu hol yaratilgan dasturlardan foydalanishni qiyinlashtirar edi.

Har qanday axborot tizimining maqsadi real muhit ob'ektlari haqidagi ma'lumotlarga ishlov berishdan iborat. Keng ma'noda ma'lumotlar bazasi - bu qandaydir bir predmet sohasidagi real muhitning aniq ob'ektlari haqidagi ma'lumotlar to'plamidir.

Predmet sohasi deganda avtomatlashtirilgan boshqarishni tashkil qilish uchun o'rganilayotgan real muhitning ma'lum bir qismi tushiniladi. Masalan, korxona, zavod, ilmiy tekshirish instituti, oliy o'quv yurti va boshqalar.

Shuni qayd qilish lozimki, MBni yaratishda ikkita muhim shartni hisobga olmoq zarur:

Birinchidan, ma'lumotlar turi, ko'rinishi, ularni qo'llaydigan dasturlarga bog'liq bo'lmasligi lozim, Ya'ni MBga yangi ma'lumotlarni kiritganda yoki ma'lumotlar turini o'zgartirganda, dasturlarni o'zgartirish talab etilmasligi lozim.

Ikkinchidan, MBdagi kerakli ma'lumotni bilish yoki izlash uchun biror dastur tuzishga hojat qolmasin.

Shuning uchun ham MBni tashkil etishda ma'lum qonun va qoidalarga amal qilish lozim. Bundan buyon axborot so'zini ma'lumot so'zidan farqlaymiz, Ya'ni axborot so'zini umumiy tushuncha sifatida qabul qilib, ma'lumot deganda aniq bir belgilangan narsa yoki hodisa sifatlarini nazarda tutamiz.

Ma'lumotlar bazasini yaratishda, foydalanuvchi axborotlarni turli belgilar bo'yicha tartiblashga va ixtiyoriy belgilar birikmasi bilan tanlanmani tez olishga intiladi. Buni faqat ma'lumotlar tuzilmalashtirilgan holda bajarish mumkin.

Tizilmalashtirish - bu ma'lumotlarni tasvirlash usullari haqidagi kelishuvni kiritishdir. Agar ma'lumotlarni tasvirlash usuli haqida kelishuv bo'lmasa, u holda ular tuzilmalashtirilmagan deyiladi. Tuzilmalashtirilmagan ma'lumotlarga misol sifatida matn fayliga yozilgan ma'lumotlarni ko'rsatish mumkin.

Ma'lumotlar bazasidan foydalanuvchilar turli amaliy dasturlar, dasturiy vositalari, predmet sohasidagi mutaxassislar bo'lishi mumkin.

Ma'lumotlar bazasining zamonaviy texnologiyasida ma'lumotlar bazasini yaratish, uni dolzarb holatda yuritishni va foydalanuvchilarga undan axborot olishini ta'minlovchi maxsus dasturiy vosita, Ya'ni ma'lumotlar bazasini boshqarish tizimi yordami bilan markazlashtirilgan holda amalga oshirishni nazarda tutadi.

Ma'lumot - bu uni ma'nosiga e'tibor bermay qaraladigan ixtiyoriy simvollar to'plamidir. O'zaro bog'langan ma'lumotlar ma'lumotlar tizimi deyiladi.

Ma'lumotlar bazasi - bu ma'lum bir predmet sohasiga oid tizimlashtirilgan ma'lumotlarning nomlangan to'plami bo'lib, kompyuter xotirasiga yozilgan ma'lum bir strukturaga ega, o'zaro bog'langan va tartiblangan ma'lumotlar majmuasi bo'lib, u biror bir ob'ektning xususiyatini, holatini yoki

ob'ektlar o'rtasidagi munosabatni ma'lum ma'noda ifodalaydi. MB foydalanuvchiga strukturalashtirilgan ma'lumotlarni saqlash va ishlatishda optimal qulaylikni yaratib beradi.

Ma'lumki ma'lumotlarni kiritish va ularni qayta ishlash jarayoni katta hajmdagi ish bo'lib ko'p mehnat va vaqt talab qiladi. MB bilan ishlashda undagi ma'lumotlarning aniq bir strukturaga ega bo'lishi, birinchidan foydalanuvchiga ma'lumotlarni kiritish va qayta ishlash jarayonida undagi ma'lumotlarni tartiblashtirish, ikkinchidan kerakli ma'lumotlarni izlash va tez ajratib olish kabi qulayliklarni tug'diradi. MB tushunchasi fanga kirib kelgunga qadar, ma'lumotlardan turli ko'rinishlarda foydalanish juda qiyin edi. Bugungi kunda turli ko'rinishdagi ma'lumotlardan zamonaviy kompyuterlarda birgalikda foydalanish va ularni qayta ishlash masalasi hal qilindi. Kompyuterlarda saqlanadigan MB maxsus formatga ega bo'lgan muayyan tuzilmali fayl bo'lib, undagi ma'lumotlar o'zaro bog'langan va tartiblangandir.

Demak, ma'lumotlar bazasi deganda ma'lum bir strukturada saqlanadigan ma'lumotlar to'plami tushuniladi. Boshqacha qilib aytganda MB - bu ma'lum berilgan aniq bir strukturaga ega bo'lgan ma'lumotlarni o'z ichiga oluvchi maxsus formatga ega bo'lgan fayldir.

Ma'lumotlarni strukturalashtirish - bu shunchaki ma'lumotlarni tasvirlashda qandaydir moslikni kiritish usulidir. Odatda MB ma'lum bir ob'ekt sohasini ifodalaydi va uning ma'lumotlarni o'z ichiga oladi, ularni saqlaydi va foydalanuvchiga ma'lumotlarni qayta ishlashda undan foydalanish imkonini yaratib beradi.

Ma'lumotlar bazasi - axborot tmzimlarining eng asosiy tarkibiy qismi bo'lib hisoblanadi. Ma'lumotlar bazasidan foydalanish uchun foydalanuvchi ishini engillashtirish maqsadida ma'lumotlar bazasini boshqarish trizimlari yaratilgan. Bu tizimlar ma'lumotlar bazasini amaliy dasturlardan ajratadi.

Ma'lumotlar bazasini boshqarish tizimi (MBBT) -bu dasturiy va apparat vositalarining murakkab majmuasi bo'lib, ular yordamida foydalanuvchi

ma'lumotlar bazasini yaratish va shu bazadagi ma'lumotlar ustida ish yuritishi mumkin. Shu bilan bir qatorda Ma'lumotlar bazasini boshqarish tizimi ma'lumotlar bazasini yaratish, ularni dolzarb holatini ta'minlash va undagi zarur axborotni topish ishlarini tashkil etish uchun mo'ljallangan dasturlar majmui va til vositasidir.

Ma'lumotlar bazasi tushunchasi maydon, yozuv, fayl (jadval) kabi elementlar bilan chambarchas bog'liq .

Maydon - bu ma'lumotlarni mantiqiy tashkil etishni elementar birligi bo'lib, u axborotni eng kichik va bo'linmas birligi bo'lgan rekvizitga mos keladi. Maydonni tasvirlash uchun quyidagi tavsiflardan foydalaniladi:

Maydon nomi, masalan familiyasi, ismi, tug'ilgan sana, lavozimi, ish staji, mutaxassisligi.

1-maydon nomi	**2-maydon nomi**	**3-maydon nomi**	**•••**	**N-maydon nomi**
Yozuv	*Yozuv*	*Yozuv*	*Yozuv*	*Yozuv*

Ma'lumotlar bazasi tuzulmasining asosiy elementlari

Maydon turi - bu kiritilgan yozuvning qaysidir tipga tegishli ekanligi bilan ifodalanadi. masalan, sonli, belgili (simvolli), sana/vaqt, satrli, mantiqiy va hokazo bo'lishi mumkin.

Maydon uzunligi - bu maydondagi yozuvning simvollar yig'indisidir.

Yozuv - bu mantiqiy bog'langan maydonlar to'plami. Yozuv tuzilishi uchun uning tarkibiga kiruvchi maydolar tarkibi va joylashishi ketma-ketligi bilan aniqlanib, ularni har biri ichida elementar yozuvlarning nusxasi deb ataladi. Yozuv ob'ektning biror bir elementi haqida to'liq ma'lumotni ifodalaydi.

Ob'yekt (jadval) - bu bir xil tuzilmaga ega bo'lgan yozuvning nusxalar to'plamidir. U o'zicha har bir maydonda qiymatga ega.

Misol. STUDENT ob'yektdagi yozuvlarning mantiqiy strukturasini tavsiflashga doir misolda ko'rsatilgan. STUDENT faylidagi yozuvning tuzilishi chiziqli bo'lib, u o'zgarmas uzunlikdagi yozuvlardan iborat. Yozuv maydonlari takrorlanuvchi qiymatlar guruhiga ega emas. Maydon qiymatiga murojaat uning nomeri bo'yicha amalga oshiriladi.

Ob'yekt nomi *student*				
Maydonlar		***Kalit belgisi***	***Maydon formati***	
Nomni belgilash	***To 'liq nomlanish (rekvizit)***		***Toifasi***	***Uzunligi***
Raqam	Talaba reyting daftarchasi raqami	*	simvol	10
Familiya	Talaba familiyasi		simvol	10
Ism	Talaba ismi		simvol	8
Otaismi	Talaba otasi ismi		simvol	10
T_kun	Talabaning tug'ilgan sanasi		sana	8
O'rtabaho	Talabaning o'rtacha bahosi		son	3
...	„.		..	..

Har bir MB jadvali o'zining birlamchi kalitiga ega bo'lishi mumkin. Birlamchi kalit deganda yozuvlar kaytarilmasligini ta'minlovchi maydon yoki maydonlar guruxi tushiniladi. Birlamchi kalit sifatida ishlatiladigan maydon yoki maydonlar guruxi, bir xil yozuvga ega bo'lmaslik shartini bajarishi kerak. Boshqa maydonlarida bir xil yozuvlar takrorlanishi mumkin. Shu sabab ular birlamchi kalit bo'la olmaydi. Birlamchi kalit qisqa va sonli maydonlardan tashkil topishi maqsadga muvofiqdir. MB jadvaliga birlamchi kalitni kiritishdan maqsad, jadvaldagi ma'lumotlarni izlash, tartiblashtirish va tanlab olishda qulaylikni beradi.

Birlamchi kalit kiritish yoki kiritmaslik foydalanuvchi tamonidan MB jadvali strukturasini tashkil qilishda aniqlanadi.

Bosh jadval yordamida boshqa jadvaldagi mos ma'lumotlarni chaqirishni ta'minlash uchun boshqa jadvalda tashqi kalit tashkil qilinadi. "Bitta-ko'pga" bog'lanish holatida tashqi kalit bosh jadvalda tashkil qilinadi. Birinchi va ikkinchi kalitlarni aniqlashda MBBT avtomatik ravishda jadvalda indekslarni quradi.

Aniq ma'lumotlarni (masalani) hal qilishda inson real dunyoni u yoki bu sohasi bilan cheklanadi. Bunday hollarda faqat ba'zibir ob'ektlarni o'rganishgina qiziqish o'yg'otadi. Bunday ob'ektlarni majmuasini predmet soxa deyiladi.

Barcha ob'ektlar **atributlari** bilan xarakterlanadi. Masalan, ob'ekt sifatida fakultet, kutubxona, kompyuter va boshqalarni qarash mumkin. Jumladan, kompyuter ob'ektini atributi sifatida hisoblash tezligini, operativ xotira hajmi, o'lchamlari va boshqalarni ko'rish mumkin.

Ma'lumotlarni nomlangan eng kichik birligi **ma'lumot elementidir.** U ko'pincha maydon deb aytiladi va bayt va bitlardan tashkil topadi. Ma'lumotlar agregati ma'lumot elementini nomlangan to'plamidir.

MB administratori deyilganda birorta shaxs yoki bir necha shaxslardan iborat bo'lgan va MBni loyihalash, uzatish va samarador ishlashini ta'minlovchidir.

Ma'lumotlar bazasi tushunchasi bilan **ma'lumotlar banki** tushunchasi ham mavjud. Ma'lumotlar banki (MBn) tushunchasi ikki xil talqin etiladi.

1. Hozirgi kunda ma'lumotlar markazlashmagan holda (ishchi o'rinlarda) ShK yordamida qayta ishlanadi. Ilgari ular alohida xonalarda joylashgan hisoblash markazlarida markazlashgan holda qayta ishlangan. Hisoblash markazlariga axborotlar tashqi qurilmalar orqali kelib to'plangan. Ma'lumotlar bazasi markazlashgani hisobiga ularni ma'lumotlar banki deb atashgan. Bunda ma'lumotlarga murojat etish ishchi stansiyalardan markazlashgan holda tashkil etilgan va shuning uchun ma'lumotlar banki bilan ma'lumotlar bazasi tushunchalari o'rtasida farq qilingan.

Hozirgi kunda ko'p hollarda ma'lumotlar bazasi markazlashmagan holda tashkil qilinmoqda. Shuning uchun ma'lumotlar banki va ma'lumotlar bazasi sinonim so'zlar sifatida ham ishlatiladi.

2. Boshqacha talqinda, ma'lumotlar banki deyilganda ma'lumotlar bazasi uni boshqarish tizimi(MBBT) tushuniladi. Ma'lumot bazasi bilan ishlaydigan dasturni ilova deb ataladi. Bitta ma'lumot bazasi bilan juda ko'p ishlashi mumkin.

MB ni ishlatish afzalliklari.

- Ixchamligi;
- Axborotlarni qayta ishlash tezligini oshishi;
- Kam mehnat sarfi;
- Har doim yangi axborot olish imkoniyati;
- Ma'lumotlarni ortiqchaligini kamayishi.

Nazorat savollari

1. Maydon uzunligi nima?
2. Ob'ekt ga tarif bering?
3. Ob'ektlar attributlariga misollar keltiring?
4. MB administrator deganda nimani tushunasiz?
5. Ma'lumotlar banki tushunchasiga ta'rif bering.
6. Ma'lumotlar Bazasini ishlatish afzalliklari sanang.

2-mavzu. Ma'lumot bazasi tizimining arxitekturasi va uch bosqichli arxitektura

Reja

1. Ma'lumotlar bazasini sinflarga ajratish.
2. Ma'lumotlar bazasini uch bosqichli arxitekturasi.
3. Ma'lumotlarni fizik va mantiqiy tavsifi.
4. Ma'lumotlar bazasini boshqarish tizimini tashkil etuvchilari.

Tayanch iboralar: Ma'lumotlar logik tasviri, arxitekrura, administrator, konseptual, MBBT.

Ob'ektlarni sinflarga ajratish deyilganda, barcha ob'ektlar to'plamini birorta norasmiy belgi bo'yicha qism to'plamlarga ajratishni tushunamiz. MBni murakkabligini hisobga olib, uni sinflarga ajratish belgilari xilma - xil. Hozirgi kunda MBni quyidagi sinflari ko'p ishlatiladi:

1. MB ma'lumotlarni tasvirlash shakliga qarab: video, audio, multimedia guruxlariga ajratish mumkin.

2. Video MB ma'lumotlarini ko'rinishiga qarab o'z navbatida matnli va grafik tasvirli bo'ladi.

3. Matnli MB ma'lumotlarni strukturalanganligaga qarab strukturalangan, qisman strukturalangan va strukturalanmagan MB ga bo'linadi.

4. Strukturalangan MB o'z navbatida ma'lumotlarni modeliga qarab: ierarxik, tarmoqli, relyatsion, ob'ektli relyatsion, ob'ektga yo'naltirilgan MBga bo'linadi. Bundan tashqari strukturalangan MBlari strategik va dinamik shuningdek, markazlashgan va taqsimlangan MBga bo'linadi. MBni foydalanuvchilar soniga qarab: bitta va ko'p foydalanuvchili MBga bo'lamiz va ular ma'lumotlarni saqlanishiga qarab operatsion va analitik bo'ladi.

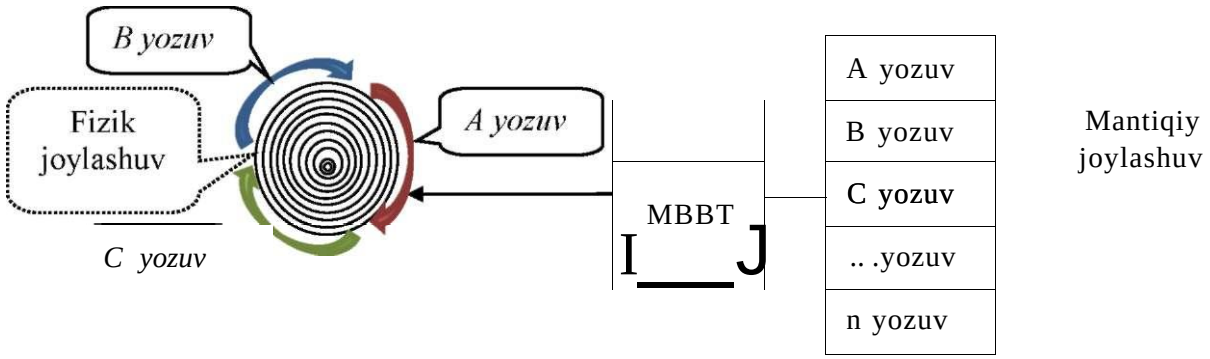
Sanab o'tilgan guruhlardan tashqari iqtisodiy nuqtai nazardan pulli va pulsiz MBga bo'linadi. Shuningdek, murojaat qilish darajasiga qarab: ommabop va murojaati cheklangan MBga bo'linadi.

MBni logik va fizik tasvirlash. Ma'lumotlarni tavsiflash va ular orasidagi munosabat aloqalar o'rnatish 2 xil ko'rinishda bo'ladi:

1. Logik yoki mantiqiy;

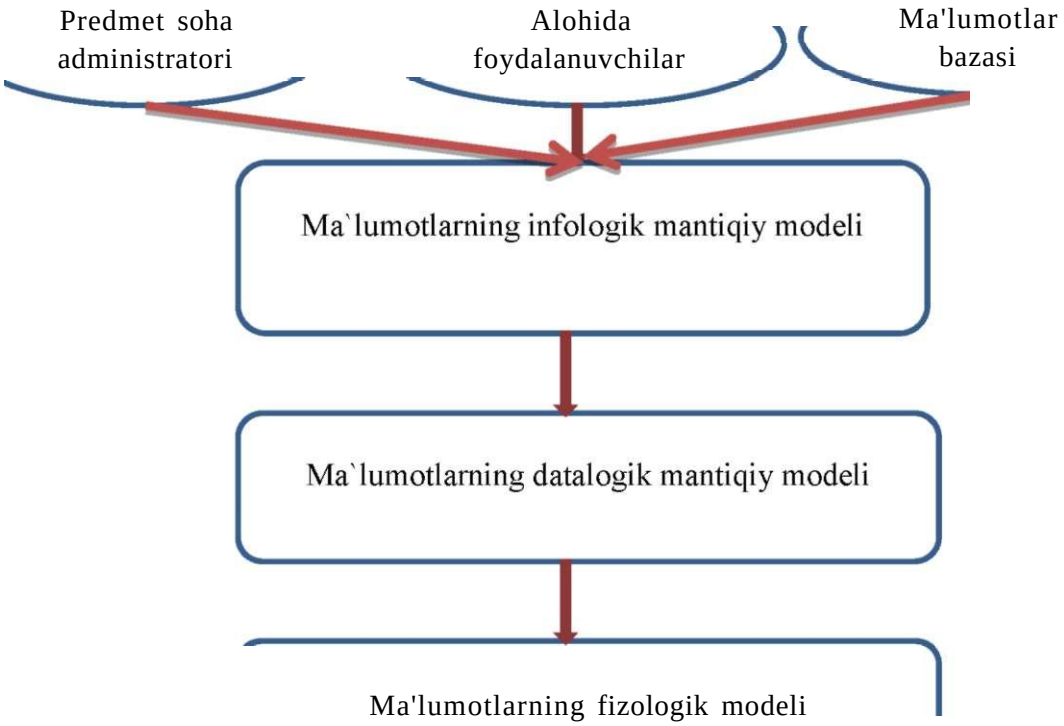
2. Fizik;

Fizik tasvirlashda ma'lumotlar mashinani tashqi xotirasida saqlashi bilan izohlanadi. Mantiqiy tasvirlashda esa amaliy dasturchi yoki foydalanuvchi tomonidan ma'lumotlarni tasvirlash ko'rinishi tushuniladi.



2.1-rasm. Ma'lumotlarni fizik va mantiqiy joylashuvi

Ma'lumotlar bazasini uch bosqichli arxitekturasi. Ma'lumotlar bazasini boshqarish tizimini qanday tashkil etilishini o'rganishdagi ilmiy izlanishlar, ularni amalga oshirishni xilma - xil usullarini ko'rsatib beradi. Bularning ichidan eng optimal varianti Amcrika standartlashtirish qo'mitasi ANSI (American National Standarts Institute) tomonidan taqdim etilgan MBni uch bosqichli tashkil etish usuli hisoblanadi.



2.2-rasm. Ma'lumotlar bazasini boshqarish tizimini uch bosqichli modeli

Tashqi modellar - eng yuqori bosqich, bunda har bir model o'zini ma'lumotlar tasviri qabul qiladi. Har bir ilova, o'ziga kerakli zarur bo'lgan ma'lumotlarni ko'radi va qayta ishlaydi. Masalan, ishchilarni malakasi bo'yicha taqsimlash tizimi xizmatchi malakasi haqidagi ma'lumotlarni ishlatadi. Uni oklad, manzili, telefoni haqidagi axborotlar qiziqtirmaydi va aksincha, oxirgi ma'lumotlar xodimlar bo'limi qism tizimida ishlatiladi.

Konseptual bosqich - markaziy boshqarish zvenosi bo'lib, bunda MB eng ko'p umumiy holda tasvirlanib, u shu MB bilan ishlaydigan barcha ilovalar ishlatiladigan ma'lumotlarni qamrab oladi. Umuman konseptual bosqich MB yaratilgan predmet sohani umumlashgan modelini akslantiradi. Bu model ob'ektlarning muhim xossalarini shakllantirish uchun xizmat qiladi.

Fizik bosqich - fayllarda joylashgan ma'lumotlarni tashqi axborot saqlovchilarida joylashishini belgilaydi. Bu arxitektura ma'lumotlar bilan ishlaganda mantiqiy va fizik mustaqillikni ta'minlab beradi.

Mantiqiy mustaqiliylik bitta ilovani o'zgartirishni, shu baza bilan ishlaydigan boshqa ilovani o'zgartirmasdan amalga oshirishni bildiradi.

Fizik mustaqiliylik, saqlanuvchi ma'lumotlarni bir qattiq diskdan boshqasiga ko'chirganda uni ishlash qobiliyatini saqlab qolgan holda o'tkazishni bildiradi.

2.2-rasmda ko'rsatilgan boshqa modellar kompyuter uchun yo'naltirilgan hisoblanadi. Ular yordamida MBBT dasturlar va foydalanuvchilarga saqlanayotgan ma'lumotlardan foydalanish uchun imkoniyat yaratadi. Bu imkoniyat ma'lumotlarni fizik joylashishini hisobga olmasdan, balki dasturlar va foydalanuv-

chilar nomlari bo'yicha amalga oshiriladi. MBBT kerakli ma'lumotlarni tashqi eslab qolish qurilmasidan ma'lumotlarning fizik modeli bo'yicha izlaydi.

Demak, kerakli ma'lumotlardan foydalanishga ruxsat aniq bir MBBT yordamida bajariladi. Shuning uchun, ma'lumotlar modeli ushbu MBBT ma'lumotlarni tavsiflash tilida tavsiflanishi kerak bo'ladi. Ma'lumotlarning infologik modeli bo'yicha yaratiladigan bunday tafsiviga ma'lumotlarning datalogik modeli deyiladi.

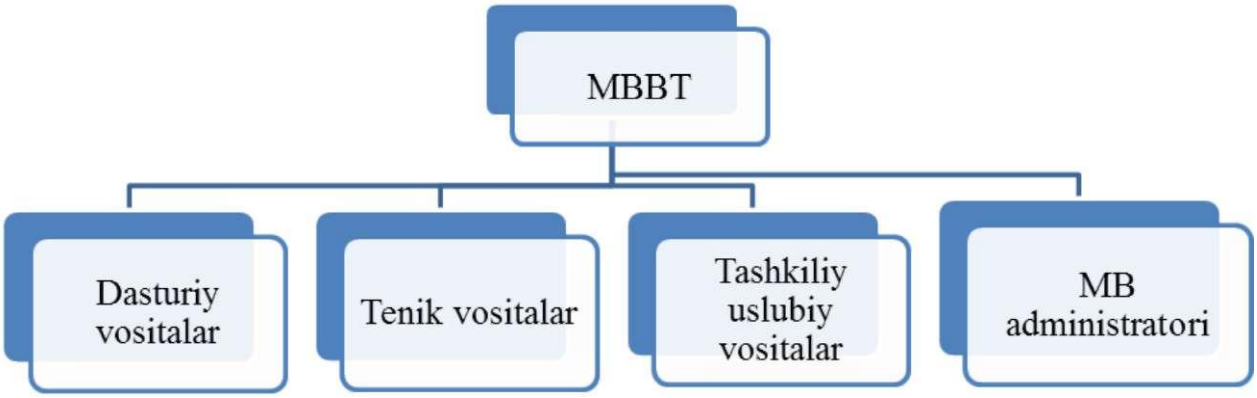
Uch bosqichli arxitektura (infologik, datalogik va fizik bosqich) ma'lumotlarning saqlanishi unga ishlatiladigan dasturga bog'liqmasligini ta'minlaydi. Kerak bo'lganda saqlanayotgan ma'lumotlarni boshqa ma'lumot tashuvchilarga yozib qo'yish va (yoki) ma'lumotlarning fizik modelini o'zgartirish bilan uning fizik strukturasini qayta tashkil etish mumkin. Tizimga istalgan yangi foydalanuvchilarni (yangi ilovalarni) qo'shish mumkin. Agar datalogik model kerak bo'lsa, uni qo'shish mumkin.

MBBTni tarkibi. MBBT shunday dastur qobig'iki, uning yordamida jadvallarni strukturasi, jadvallar orasidagi bog'lanish, jadvallarni ma'lumotlar bilan to'ldirgandan keyin uning yordamida MB yaratiladigan dasturiy vositasidir. Shu munosabat bilan MBBT bir qancha tarkibiy qismlardan iborat.

Dasturiy vositalariga translyatorlar va MBga ma'lumotlarni kiritadigan, qayta ishlaydigan, saqlaydigan, takomillashtiridigan, testdan o'tkazadigan, ma'lumotlarni kiritish chiqarishni ta'minlaydigan boshqarish tizimlari kiradi. Asosiy dasturlash tili sifatida C, C++ kabi tillarni ishlatiladi.

MBBTni paydo bo'lish tarixida 3 ta til umumlashgan holda ishlatilgan:

1. Ma'lumotlarni tavsiflash tili - MTT (YaOD). Uni yordamida MB jadvallarini strukturalari quriladi.
2. Ma'lumotlar bilan ishlaydigan til - MIT (YaMD). Bu til MB sini ma'lumotlar bilan to'ldirish va uni tiklash amallarni (olib tashlash, takomillashtirish va b.) bajarishda ishlatiladi.
3. So'rovlar tili - YaZ. Bu til yordamida qidirish mezonlari asosida kerakli axborotlarni topish va ularni chiqarish uchun hizmat qiladi.



2.3-rasm. MBBT tashkil etuvchilari

Hozirgi kunda barcha aytilgan tillarni vazifasini SQL tili bajaradi.

Texnik vositalar sifatida, asosan, shaxsiy kompyuterlar va super kompyuterlarni ishlatamiz. Uslubiy - metodik vositalar - bu ko'rsatmalar, metodik va me'yoriy materiallarni majmuasi bo'lib, ular yordamida MB va MBBT dan foydalanish yo'llari ko'rsatiladigan vositalaridir.

MBBTdan ikki gurux shaxslari foydalanadi:

1. Chekli yoki oddiy foydalanuvchilar;
2. MB administratori;

MB administratorini xizmat doirasiga quyidagi ishlar kiraddi:

a) Predmet sohani tahlili va foydalanuvchilar va axborotni o'rnini aniqlash;
b) Ma'lumotlarni tuzilishini loyihalash va ularni takomillashtirish;
c) Qo'yilgan topshiriqlar va ma'lumotlarni bir butunligini ta'minlash;
d) MBni yuklash va yuritish;
e) Ma'lumotlarni himoya qilish;
f) MBni tiklashni ta'minlab berish;
g) MBga murojaatlarni yiqish va statistik qayta ishlab berish;
h) MBga ko'p foydalanuvchilar rejimida ishlaganda, ma'lumotlarni o'chib ketishidan ximoya qilish;
i) Texnik vositalar nosoz bo'lib ishdan chiqqanda, ma'lumotlarni saqlash va qayta tiklash ishlarini bajarish;

Nazorat savollari

7. Ma'lumotlar qaysi belgilari bo'yicha sinflarga ajratiladi?
8. Ma'lumotlarni logik (mantiqiy) va fizik tasvirlash nima?
9. Ma'lumot bazasini qanday uch bosqichli arxitekturalari mavjud?
10. Ma'lumotlar bazasi alministratorini asosiy vazifalarini aytib bering.
11. Ma'lumotlar bazasida tasvirlar qanday saqlanadi. Misollar keltiring.
12. Ma'lumotlar bazasini uch bosqichli arxitekturasini chizmasini tasvirlang.
13. So'rovlarni qayta ishlashda MBBTning bajaradigan ishlar ketma-ketligini tavsiflab bering.

3-mavzu. Ma'lumotlar bazasi modellari va mohiyat-aloqa modeli

Reja

1. Ma'lumotlar modeli tushunchasi.
2. Ierarxik (shajara) ma'lumotlar modeli.
3. Tarmoqli ma'lumotlar modeli.
4. Relyatsion ma'lumotlar modeli.
5. Ma'lumotlar bazasini loyihalashda mohiyat - aloqa modeli.
6. Mohiyat aloqa diagrammasini qurish.

Tayanch iboralar: infologik model, datalogik model moslik, Ulman Chen usuli, mohiyat, ma'lumot modeli, ierarxik,tarmoqli model, relyatsion model.

Buguni kun ma'lumotlar bazasini boshqarish tizimlari datalogik bosqichda xilma-xil ma'lumotlar bazasi bilan ishlashni ta'minlaydi. Hozirgi kunda eng ko'p o'rganilgan va keng ishlatiladigan ma'lumot modellari kiradi. Ma'lumotlar modeli bu ma'lumotlar bazasini ma'lumot elementlari to'plami orasidagi bog'lanish tuzulmalarini tasvirlovchi umumiy sxemadir. Ma'lumotlar modeli tushunchasini aniq ta'rifi Kodd tomonidan tushuntirib berilgan. U ma'lumotlar modelini uchta kerakli komponentasini keltirgan:

1. mavjud bo'lgan ma'lumot tuzilmalarini aniqlash vositalari majmuasi;

2. ma'lumotlarni qidirish va qayta ishlash uchun ma'lumotlar bazasi holatiga qo'llaniladigan amallar to'plami;

3. oshkor holda ma'lumotlar bazasi holatini aniqlovchi va bir butunligini ta'minlovchi vositalar to'plami.

Hozirgi kunda klassik hisoblashlarda 3 ta ma'lumotlar modeli ko'p ishlatiladi:

- **Ierarxik ma'lumotlar modeli;**
- **Tarmoqli ma'lumotlar modeli;**
- **Relyatsion ma'lumotlar modeli.**

Har bir ma'lumotlar bazasi u yoki bu model asosida yoritiladi. Har bir MBBT esa u yoki bu ma'lumotlar modelini ta'minlaydi deyiladi. Masalan ierarxik ma'lumotlar modeliga asoslangan tizim - *ines* tizimidir. Tarmoqli modellarda esa - *BANK OS, SETOR,* relyatsion modelga asoslangan tizimlar - *Access, KARAT* va boshqalar.

Ierarxik malumotlar modeli. Ierarxik ma'lumotlar modelida yozuvlar daraxtsimon tuzilmali ko'rinishda bo'ladi. Ma'lumotlar bazasini boshqarish tizimlaridan ba'zi birlari faqat ierarxik tuzilishga ega bo'lganlari bilan ishlatiladi. Ierarxik tuzilmali ma'lumotlar sodda yaratiladi. Bu ko'pincha taddimotlarda qulay, lekin ma'lumotlarni ko'plari daraxtsimon tuzilmali bog'lanish tabiatiga ega emas.

Masalan ikkita firma ishlab chiqargan mahsulotlarning barcha turlarini narxlari berilgan. Shu ma'lumotlarni narxlar ma'lumotnomasi qurilsin va kompyuter xotirasiga joylashtirilsin. Faraz qilamiz, A va B firmalar mos ravishda ikki xil ko'rinishdagi mahsulot ishlab chiqaradi. Har bir mahsulot ko'rinishi har xil texnologiya asosida bajariladi. Bunda uning narxi ham shunga qarab bo'ladi. Bir nechta mahsulot ikkita sxema asosida tayyorlanadi. Ular 01, 02 kabi deb belgilanadi va ular quyidagi narxni belgilaydi. 580, 610 va 1250 maxsulotlar uchta sxema asosida tayyorlanadi. Ular 01, 02, 03 deb belgilanadi va ular quyidagi narxni belgilaydi 380, 345 va 410. B firma uch xil maxsulot ishlab chiqaradi. Ularni kodi mos ravishda 1250, 1640 va 1930 kodga ega bo'lsin. Ular ham o'zlarini ishlab chiqish sxemasi va narxiga ega bo'lsin.

				01	380
	01	580			
			1250	02	345
3980					
	02	610		03	410

B firma uch xil mahsulotni ishlab chiqaradi: 1250, 1640, 1930.

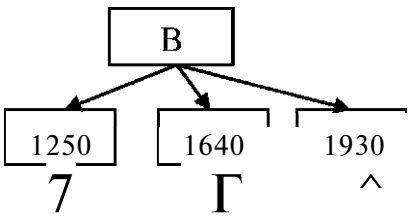
1250

1640 ^

1930

B archa maxsulotlar nomlari, ularni ishlab chiqish sxemasi va narxlarni o'z ichiga oluvchi ma'lumotnoma tuzish va mashina xotirasiga joylash talab etilsin.

Keltirilgan ma'lumotlar tuzilmani ierarxik (daraxtsimon) ko'rinishda tasvirlash qulay.



firma A

mahsulot	3980		1250		
	01	02	01	02	03
	580	610	380	345	410

3.1-rasm. Ma'lumotlarni ierarxik ko'rinishi.

Bunday tuzilmani jadval ko'rinishida ham tasvirlash mumkin.

3.1-jadval. Ma^lumotlarnijadval ko^rinishi.

Format	Mahsulot	Texnologiya	Narx

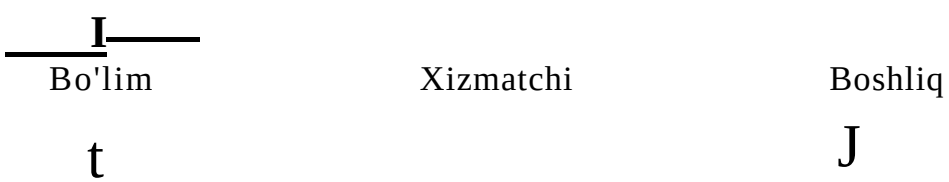
A	3980	01	586
A	3980	02	610
A	1250	01	380
A	1250	02	345
A	1250	03	410

Agar berilgan element bir nechta o'zidan yuqori elementga suyansa tarmoqli ma'lumot elementiga ega bo'lamiz.

Tarmoqli ma'lumotlar modeli.Agar munosabatdagi joriy element bir necha berilgan elementga ega bo'lsa, bunday bogTanishlarni ierarxik strukturalar bilantavsiflab bo'lmaydi.

Bunday tuzilmalar tarmoqli graflar bilan tavsiflanadi. Tarmoqli strukturalarida element ixtiyoriy boshqa element bilan bogTanishi mumkin. Ya'ni, tarmoqli bir necha kichkina ob'ektlardan tashkil topgan yirik ob'ekt deb qarash mumkin.

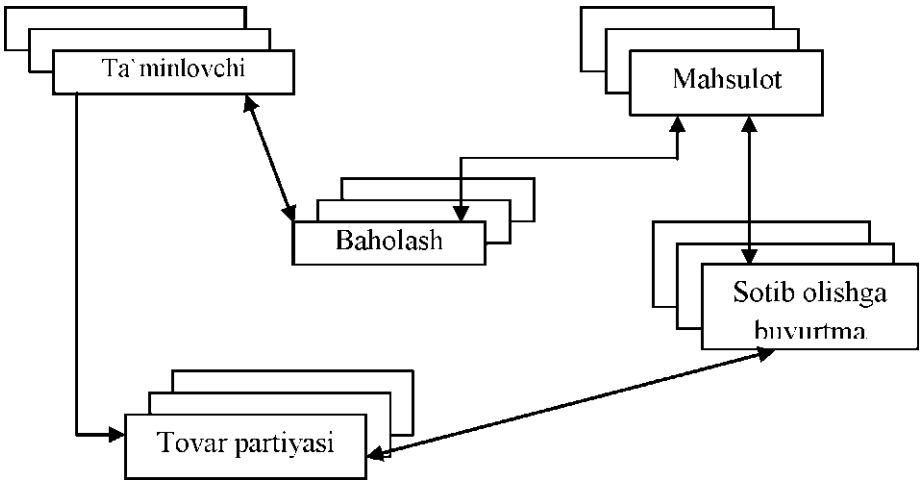
3.2-rasmda bogTanishlarni tarmoqli modelda tasvirlanishi keltirilgan. Shunday qilib, tarmoqli model ma'lumotlar elementlari orasidagi xilma-xil bogTanishlarni ixtiyoriy ko'rinishdagi grafik yordamida akslantiradi. Tarmoqli model yozuvlar to'plami va mos bogTanishlar to'plamidan tashkil topadi. BogTanishlarni yaratish uchun alohida cheklanishlar qo'yilmaydi. Misol tariqasida oddiy tarmoqli ma'lumotlar bazasi sxemasi sifatida quyidagini keltirish mumkin 3.3-rasm.



3.2-rasm. Ma'lumotlarni taqmoqli ko'rinishi

Tarmoqli ma'lumotlar bazasi turida ma'lumotlar bilan quyidagi ishlarni bajarish mumkin.

1. ma'lumotlar bazasi yozuvlarini qidirish;
2. yangi yozuvni yaratish;
3. joriy yozuvni olib tashlash;
4. joriy yozuvni tiklash;
5. yozuvni bog'lanishga qo' shish;
6. yozuvni bog'lanishdan olib tashlash;
7. boglanishlarni o'zgartirish.



3.3-rasm. Tarmoqli ma'lumotlar modeliga misol

Relyatsion ma lumotlar modeli. Ma'lumotlarni relyatsion modeli asosida munosabat tushunchasi yotadi. Munosabatni ikki o'lchamli jadvallar yordamida tavsiflash qulay. Jadval tushunarli ko'rimli va inson uchun oddiy. Munosabatlar to'plami ma'lumotlarni saqlash uchun ishlatilishi mumkin. Shu bilan birga ular orasidagi bogTanishlarni modellashtirish imkonini beradi. Yuqorida ko'rib chiqilgan ierarxik, tarmoqli va boshqa ma'lumotlarni tasvirlash usullarini shunday ikki o'lchamli jadvalga keltirish mumkin. Bunday jadvallar quyidagi xususiyatlarga ega bo'ladi.

1. Jadvalni har bir ma'lumot elementi maydon hisoblanadi va takrorlanuvchi guruhlar bo'lmaydi;
2. Barcha ustunlar bir jinslidir;

3. Har bir ustunga nom tayinlangan;

4. Jadvalda bir xil satr ikki marta uchramaydi;

5. Bunday jadvalda satr va ustunlar ixtiyoriy tartibda qaraladi va ixtiyoriy ketma-ketlikda ishlatilishi mumkin.

Yuqoridagi sanab o'tilgan ma'lumotlar modelidan tashqari hozirgi kunda quyidagi ma'lumotlar modellari ham amaliyotga kirib kelmokda.

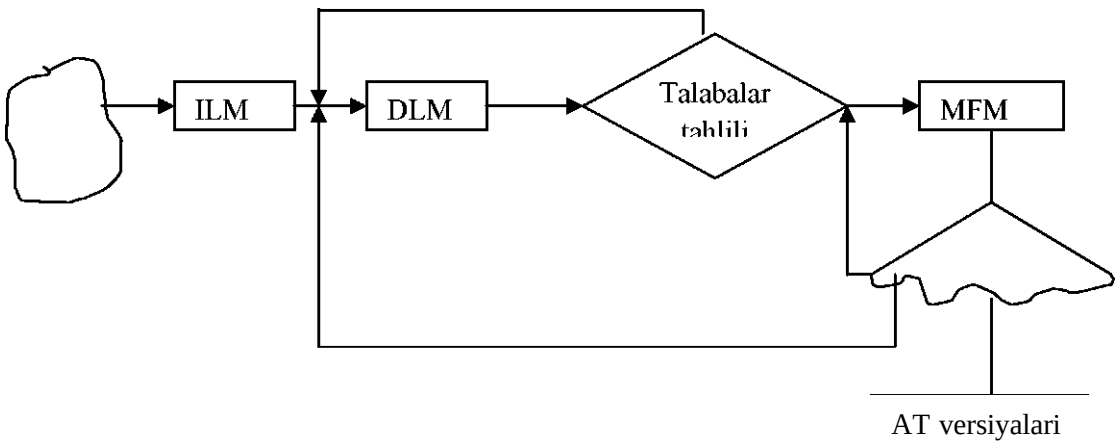
1. Ko'p o'lchamli ma'lumotlar modellari;
2. Ob'ektga yo'naltirilgan ma'lumotlar modellari.

Shuningdek boshqa ma'lumotlar modellariga asoslangan har xil tizimlar ham ishlab chiqilmoqda. Bular qatorida quyidagilarni sanash mumkin.

1. ob'ekt- relyatsion;
2. semantik;
3. yo'naltirilgan;
4. konseptual va boshqalar.

Ulardan ba'zilari bilimlari bazasi va dasturlash tillarini integratsiyalashga xizmat qiladi.

Hozirgi kunda axborot tizimlarini loyihalash xilma-xil usullari mavjud. Umuman olganda, axborot tizimlarini dasturiy ta'minotini yaratish interaktiv xarakterga ega. Axborot tizimlarini loyihalashni asosiy bosqichlari va ular orasidagi bog'lanish 3.4rasmda keltirilgan:



3.4-rasm. Ma'lumotlar bazasini loyihalashning asosiy bosqichlari

PS - predmet soha;

ILM - infologik model;

DLM - datalogik model;

MFM - ma'lumotlarni fizik modeli;

AT - axborot tizimi.

AT loyihalashni 1 - bosqichida predmet sohasida mantiqiy axborot tuzilmasini quramiz. U PS ni va foydalanuvchini talablarini o'zida mujassamlashtiradi. Bunda biz aniq MBBTga bog'lanmagan ravishda bu ishlarni bajaramiz, Ya'ni PSni axborot-logik tavsifi bajariladi. Bu bosqich infologik model qurish bosqichi deb ataladi. MBBT vositasi yordamida ma'lumotlarni mantiqiy bog'lanishlarini tashkil qilish ma'lumotlar bazasini DLM ini bildiradi. Bu model yordamida ma'lumotlar elementlari orasidagi mantiqiy bog'lanishlarni aks ettiradi. DLMni ma'lumotlarni saqlash muhiti bilan bog'laydigan bosqich ma'lumotlarni fizik modeli deyiladi.

Hozirgi kunda PSni tavsiflash uchun ko'p usullar mavjud. Shulardan biri ob'ekt-aloqa usulidir. Bu usulni ba'zan Ulman - CHen usuli ham deyiladi. PSni mohiyat-aloqa usulida tavsiflaganda quyidagi bosqichlarda ish olib boriladi:

1. PS ni ob'ektlari aniqlanadi;
2. Ob'ekt sohalari (atributlari) belgilanadi va uning kalit parametri aniqlanadi. Kalit parametri ob'ektni identifikatsiyalaydi;
3. Ob'ektlar o'rtasida aloqa o'rnatiladi va ular sinflarga ajratiladi;
4. Maxsus belgilar kiritilib, ob'ekt aloqa diagrammasi o'rnatiladi.

Bu diagramma PS ning infologik modeli grafik tasviri hisoblanadi.

ER modeli ma'lumotlarni quyidagicha tavsiflaydi:

1. Ob' ektlar va ob' ektlar maj mui
2. Aloqa va munosabatlar majmui (Relations).
3. Xususiyatlar (Attributes), Ob'ekt va munosabatlarni tavsiflovchi xususiyatlar

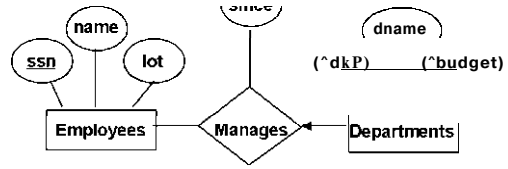
ER modeli ob'ektlar majmuasini ifodalaydi. Lekin ob'ektlar ularning atributlari bo'yicha tavsiflanadi.

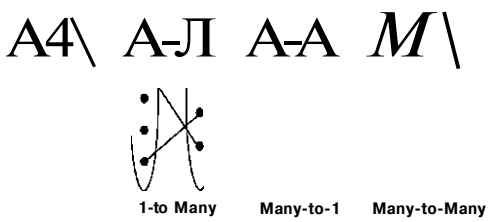
Xususiyatlar(atributlar) kategoriyasi

- Oddiy va kompozitsiyadagi atributlar
- Bir o'lchamli va bir nechta o'lchamli atributlar
- Saqlangan va tanlab olingan atributlar
- Kalit yoki yagona atributlar

Xususiyat qiymatlari ob'yektlar uchun alohida bo'lishi shart.

ER Model asoslari





3.5-rasm ER modeli bazasidagi "sxema"

ER modeli bazasidagi "sxema" ER diagrammalaridan foydanganda rasm sifatida tasvirlangan.

Ob'ekt: Ma'lumotlar bazasida ob'ekt atributlar yig'indisi sifatida tavsiflanadi.

Ob'ektlar yig'indisi: Korxonadagi barcha ob'ektlar bir xil atributlarga ega bo'lsa (ISA ierarxiyasi qo'llanilmasa):

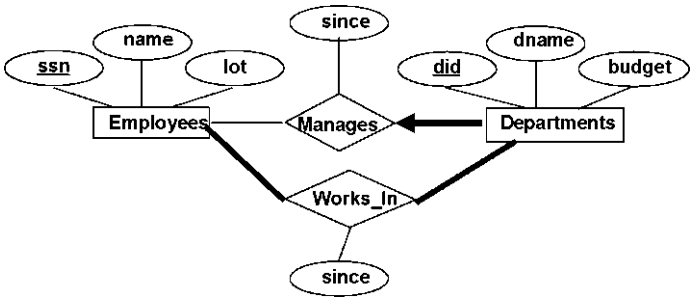
Har bir ob'ektning kaliti mavjud.

Har bir xususiyat domen(nom) ga ega bo'ladi.

Asosiy cheklovlar

Xodimlar ko'plab bo'limlarda ishlashlari mumkin;

Bo'limlarda ko'plab xodimlar bo'lishi mumkin.



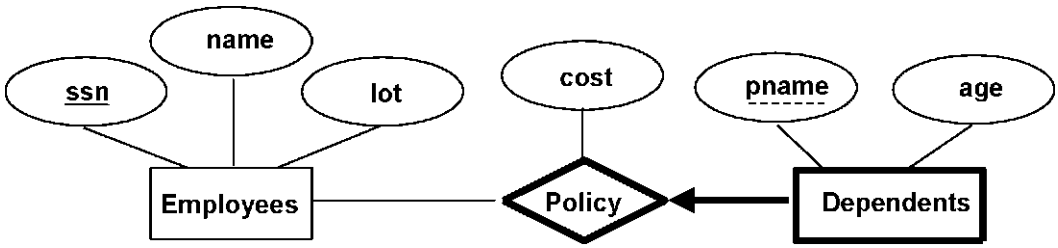
3.6-rasm ER bog'lanish

Aksincha, boshqaruvchiga ko'ra har bir bo'linma ko'pi bilan bitta boshqaruvchiga ega bo'lishi mumkin.

Qo'shimcha cheklovlari

Har bir bo'limda boshqaruvchi bormi?

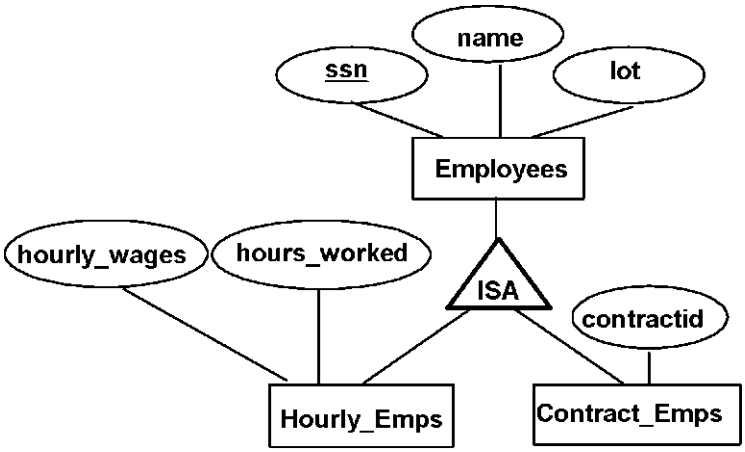
Agar shunday bo'lsa, bu ishtirokchi cheklovidir: "Manages"da bo'linmalarining ishtiroki umumiy deb hisoblanadi (qisman bo'lsada).



3.7-rasm ER ob'ekt

Zaif ob'ekt boshqa (egalik qiluvchi) ob'ektning asosiy kalitini hisobga olgan holda aniqlashi mumkin.

- Mulk egasi tanlangan va zaif ob'ektlar bir-biri bilan ko'pga-ko'p munosabat o'rnatishi kerak.
- Zaif ob'ekt munosabat to'plamini aniqlashda to'liq ishtirok etishi kerak.

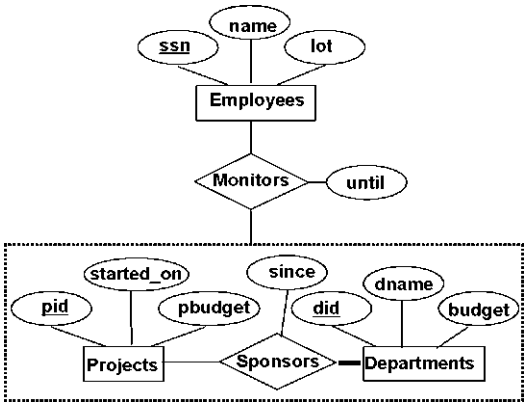


3.8-rasm ER to'plam

Ierarxiyalar

- 2 ta cheklov turlari mavjud:
  - Overlap cheklov
  - Covering cheklov

ISAdan foydalanish uchun sabablar:



3.9-rasm ISA modeli

- Quyi sinfga xos tavsiflovchi atributlarni kiritish.
- Munosabatda ishtirok etuvchi ob'ektlarni aniqlash.

Birlashtirish

- *Ob 'ektlar majmuini* va *munosabat majmuini* o'z ichiga olgan munosabatlarni modellashtirishimiz zarur.
- *Birlashtirish:* aloqalar boshqa munosabatlar majmuasida ishtirok etayotganligini ko'rsatadi.

- *Birlashtirish bizga boshqa munosabatlarda qatnashish uchun belgilangan ob 'ekt sifatida belgilangan munosabatlarni ko'rib chiqishga imkon beradi.*

Birlashtirish va uch tomonlama munosabatlar

- tavsiflovchi atribut bilan alohida munosabatlarni nazorat qiladi
- bundan tashqari, har bir homiylikning ko'pchiligi kamida bitta xodim tomonidan nazorat qilinishini mumkin.

ER Modeli g'oyaviy rejalar

- Reja tanlovi:
  - Agar kontseptsiya ob'ekt yoki xususiyat sifatida modellashtirilsa?
  - Agar kontseptsiya ob'ekt yoki munosabat sifatida modellashtirilsa?
- Aloqalarni aniqlash: ikkilik yoki uch tomonlama? Aggregatsiya(Birlashtirish)?
- ER Modelida cheklovlar:
  - Ko'pgina ma'lumotlar semantikasini olish mumkin.
  - Biroq, ER diagrammalarida ba'zi cheklovlar mavjud emas.

Ob'ekt va Attribut(Xususiyat)

Masalan: Ishchining Manzil atributi bo'lishi kerakmi yoki ob'ektning?

- Ushbu ma'lumot manzil ma'lumotlari va ma'lumotlarning semantikasidan foydalanishga bog'liqdir:
  - Agar biz, har bir xodim uchun bir nechta manzilga ega bo'lsak, manzil ob'ekt bo'lishi shart (chunki atributlar belgilanishi mumkin emas).
  - Agar bino (shahar, ko'cha va hokazo) muhim bo'lsa, **(Masalan: biz ma'lum bir shaharda xodimlarni ishga olishni istaymiz)** u holda manzil, manzil sifatida modellashtirilgan bo'lishi kerak (chunki atribut qiymati atomik).
- *Works_In2* > 2 davr mobaynida ishchiga bo'limda ishlashga ruxsat bermaydi.
- Bu esa xodim bir nechta manzilni yozishni xohlash muammosiga o'xshaydi:

biz ushbu munosabatlarning har bir namunasi uchun tavsiflovchi atributlarning bir nechta qiymatlarini yozib olamiz.

Ob'ekt va Munosabat

- Agar menejer har bir bo'lim uchun alohida byudjet ajratsa ER diagram normal xolatda bo'ladi.
- Agar boshqaruvchi barcha boshqariladigan departamentlarni qamrab oladigan ixtiyoriy byudjetga ega bo'lsachi?
  - *Ortiqcha dbudget,* menejer tomonidan boshqariladigan har bir bo'lim uchun saqlanadi.

ER diagrammalarini kontseptsial sxema bo'yicha bajarish

- Har bir ob'ekt yangi xaritaga jadval qo'shadi.
- Har bir attribut xaritaga yangi jadval ustuni qo'shadi.
- Har bir munosabat xaritalarga yangi ustunlar yoki yangi jadvalga (munosabat turiga qarab)o'rnatish.

Nazorat savollari

1. Ma'lumot baza modeli nima?
2. Ierarxik (shajara) modeli ma'lumot va uning asosiy xarakteristikalari.
3. Tarmoqli model ma'lumot va uning asosiy xarakteristikalari.
4. PS moxiyat aloqa usulida tavsiflaganda qanday ishlar bajariladi?
5. Mosliklarni (munosobatlar) qanday turlari bor? Ularni tavsiflang.
6. Moxiyat - aloqa diagrammasi qanday quriladi?
7. Axborot tizimlarini loyixalashga infologik yondoshishni asosiy qoidalarini tushuntiring.

4-mavzu. Relyatsion ma'lumot bazasi va ma'lumotlar bazasida munosabatlar

Reja

1. Relyatsion ma'lumotlar bazasini asosiy tushunchalari
2. Ma'lumotlarni tasvirlashda jadvallardan foydalanish.
3. Ma'lumotlar bazasida munosobatlar.
4. Kodd ilmiy ishi.

5. Munosabatni ikki o'lchamli jadvallar yordamida tavsiflash.
6. Munosabatlar to'plami ma'lumotlarni saqlash uchun ishlatilish.

Tayanch iboralar: kortej, atribut, domen, relyatsion, munosabat, 1:1, 1:N, M:N, mohiyat, ustun, maydon.

Ma'lumotlarni relyatsion modeli asosida munosabat tushunchasi yotadi. Munosabatni ikki o'lchamli jadvallar yordamida tavsiflash qulay. Jadval tushunarli ko'rimli va inson uchun oddiy. Munosabatlar to'plami ma'lumotlarni saqlash uchun ishlatilishi mumkin. Shu bilan birga ular orasidagi bogTanishlarni modellashtirish imkonini beradi.

4.1-jadval. Ma'lumotlar bazasidagi ikki o'lchamli jadval

Xizmatchi raqami	**F.I.SH**	**Unvoni**	**Tugilgan yili**	**BoTim**	**Mutaxassis kodi**	**Lavozim**	**Maosh**
XN	*FISH*	*UN*	*TY*	*VL*	*MK*	*LM*	*MSH*
2518	Valiev S.P.	t.f.n.	1985	1	PM	Dot.	4 mln
2567	Soliev I.T.	t.f.d.	1987	2	EVM	Prof.	5 mln
3245	Aliev S.I.	t.f.n.	1988	1	PM	Il.xodim	2,8 mln
3267	Buriev A.O.	Akad.	1982	3	ASU	Prorek.	4 mln

4.1-jadvaldagadi birinchi satr atribut nomlari, ikkinchi satr atributlarni qisqacha nomlari va uchinchi, to'rtinchi va beshinchi satrlar esa atribut qiymatlaridir.

Otgan mavzularda korib chiqilgan ierarxik, tarmoqli va boshqa ma'lumotlarni tasvirlash usullarini shunday ikki o'lchamli jadvalga keltirish mumkin. Bunday jadvallarni quyidagi xususiyatlari bo'ladi.

- jadvalni har bir ma'lumot elementi maydon hisoblanadi va takrorlanuvchi

guruxlar bo'lmaydi;

- barcha ustunlar bir jinslidir;
- xar bir ustunga nom tayinlangan;
- jadvalda bir xil satr ikki marta uchramaydi;
- bunday jadvalda satr va ustunlar ixtiyoriy tartibda qaraladi va ixtiyoriy ketma-ketlikda ishlatilishi mumkin.

Bunday xususiyatlarga jadvalar munosabat deyiladi. Munosabat asosida qurilgan ma'lumotlar bazasi relyatsion ma'lumotlar bazasi deyiladi.

4.1-jadvalni sxematik qisqartirilgan ko'rinishida xizmatchi ob'yekti (xiz.nom F.I,SH, unvoni, tugilgan yili, bo'lim, mutaxasiss kodi, lavozim, maosh) ma'lumotlar bazasi sxemasi deyiladi.

Shunday qilib, relyatsion ma'lumotlar bazasi ma'lumot elementlar to'plami asosida quriladi. Munosabat yoki jadvalni kortejlar to'plami deb qarash mumkin.

Agar jadvalda n ta ustun bo'lsa, u n tartibli kortejdan iborat deyiladi va munosabat ham N-darajali deyiladi.

Har bir atribut qiymatlari to'plami domen deyiladi. Munosabatda har bir kortej o'zining kalit identifikatoriga (nomiga) ega bo'lishi kerak va kalit quyidagi xususiyatlarga ega bo'ladi:

- kortej kalit qiymati bilan bir qiymatli ifodalanishi kerak.
- kalitda ortiqchalik bo'lmasligi kerak, Ya'ni hech qanday atributni kalitdan olib tashlash mumkin emas.

Ob'ektlarni identifikatsiyalash uchun yoki boshqacha qilib aytganda kompyuter xotirasida yozuvlarning o'rnini aniqlashda ma'lumot elementi ishlatiladi. Bu element kalit deb ataladi. Agar kalit ob'ektni bir qiymatli identifikatsiyalasa, u birlamchi kalit deyiladi. Aks holda tashqi kalit deyiladi. Agar ob'ektlarni identifikatsiyalash uchun bir nechta atributlar talab etilsa, bunday kalit tugallangan kalit deb ataladi. Agar A va B guruxdagi ob'ektlar berilgan bo'lsa, ular orasidagi quyidagi mosliklar yoki munosabatlarni o'rnatish mumkin:

1. Birga - bir (1:1)munosabat. A va B ob'ektlar to'plami orasida 1:1 munosabat

o'rnatilgan deyiladi, agarda A ob'ektning bir nushasiga B ob'ektning bitta nushasi mos kelsa, va aksincha, B ob'ektning bir nushasiga A ob'ektning bitta nushasi mos kelsa.

2. Birga - ko'p (1:n) munosabat. A va B ob'ektlar to'plamida A ob'ektning bir nushasiga B ob'ektning bir nechta nushasi mos kelsa, shu bilan birga B ob'ektning bir nushasiga A ob'ektning bir nechta nushasi mos kelsa shunday munosabat hosil bo'ladi.

3. Ko'pga - bir (n:1) munosabat A va B ob'ektlar to'plami orasida o'rnatilsa, unda A ob'ektning bir nechta nushasiga B ob'ektning bitta nushasi mos keladi. B ob'ektning nushalari orasida shundaylari mavjudki, ularga A ob'ektning bir nechta nushasi mos keladi.

4. Ko'pga - ko'p (m:n) munosabat. A va B ob'ektlar orasida shunday munosabat o'rnatilgan deyiladi, agarda A ob'ektning bir nechta nushasiga B ob'ektni bir nechta nushasi mos kelsa va aksincha.

Ob'ektlarni tahlil qilib bo'lingandan so'ng, shu ob'ektga qo'yiladigan boshlang'ich so'rovni ishlab chiqish zarur.

Masalan avtovakzalni faoliyati iqtisodiy va texnik ko'rsatgichlar bilan bog'liq bo'lganligi uchun, yo'lovchilarga axborot ma'lumot berganligi uchun yaratayotgan axborot tizimi quyidagi so'rovlarga javob berishi kerak:

- Har bir reys uchun nechta chipta sotilganligi va umumiy sotilgan chiptalarni aniqlash;
- Reys raqami bo'yicha reys haqida ma'lumotlar chiqarish;
- Marshrutlar haqidagi zarur axborotlarni chiqarish;
- Aniq reyslar uchun qaysi haydovchilar tayinlanganligi va ular haqida ma'lumotlarni olish;
- Avtobusni texnik xarakteristikalari haqidagi ma'lumotni olish.

Ko'rib chiqilayotgan predmet sohani ob'ektlari orasida quyidagi tipdagi bog'lanishlar mavjud:

1. 1:1 - chiptalar bilan yo'lovchilar ob'ektlari orasidagi bog'lanish (sotilgan);

2. M:1 - marshrut va reys orasidagi bog'lanish (marshrut munosabati);

3. M:N - marshrut va bekatlar orasidagi bog'lanish (bekatlar);

4. 1:N - reys va haydovchi orasidagi bog'lanish (tayinlash);

5. 1:N -Avtobus va haydovchi orasidagi bog'lanish (haydovchiga ruxsat berish);

Shunday qilib, ko'rilayotgan masalada asosan ob'ektlar aniqlanadi va ular orasidagi bog'lanish topiladi, hamda sinflarga ajratiladi.

4.2-jadval.Ekvivalent (sinonim) tushunchalar

Fayl	**Jadval**	**Munosabat**	**Mohiyat**
Yozuv	Satr	Kartej	Mohiyat nusxasi
Maydon	Ustun	Atribut	Atribut

Relyatsion ma'lumotlar bazasi munosabatlarida tuzilmali va semantik axborotlar saqlanishi mumkin. Tuzilmali axborotlar munosabat sxemalar yordamida aniqlanadi.

Semantik axborotlar esa munosabat sxemalarda ma'lum bo'lgan va hisobga olinadigan va atributlar o'rtasidagi funksional bog'lanishlar bilan ifodalanadi. Ma'lumotlar bazasidagi munosabatlarda atributlarni tarkibi quyidagi talablarga javob berishi kerak.

1. Atributlar o'rtasida funksional bo'lmagan bog'lanishlar bo'lmasligi kerak.

2. Atributlar guruhlanishi ma'lumotlar takrorlanishidan eng kam holatining tahlillash kerak va ular qayta ishlash va tiklashni qiyinchiliksiz amalga oshirilishi kerak.

3. Qo'yilgan ma'lumotlar bazasi munosabatlari normallashadi. Munosabatlarni normalashtirish ma'lumotlar bazasida berilgan munosabatlarni dekompozitsiya (ajratish) jarayoni yordamida sodda va kichik munosabatlar hosil qilishdir.

4.3-jadval. Ma'lumotlar bazasida talaba ob'yekti

Talaba kodi	**Famliyasi**	**Telefon**	**Talaba**

1001	Ashurov	4767777	2341717
1002	Soliev	1365556	2341717
1003	Soliev	1365656	2485888
1004	Amirov	2351717	2485888
1005	Amirov	2381817	
1006	Amirov	2351817	

Har bir munosabatda kortejlar identifikator kalitiga ega bo'lishi kerak. Kalit quyidagi ikkita xossaga ega bo'lishi kerak:

1. Kartej kalit qiymati bilan bir qiymatli ifodalanishi kerak;
2. Kalitda ortiqchalik bo'lmasligi kerak. Bu degani hech qanday atributni kalitdan olib tashlash mumkin emas.

Relyatsion ma'lumotlar bazasida axborotlarni ortiqchaligini normallashtirish yo'li bilan kamaytiriladi. Jadvallar ustida har xil amallar bajarish mumkin. Amallarga quyidagilar kiradi:

- To'plamlar ustida birlashtirish, kesishuv, ayirma, dekart ko'paytma va bo'lish amallari kiradi.
- Maxsus relyatsion amallar, ularga: proeksiya, birlashtirish, ajratish (tanlab olish) amallari kiradi.

Munosabatlar ustida amalni bajarish uchun ishlatiladigan tillarni ikki sinfga ajratish mumkin:

a) Relyatsion algebra tillari;
b) Relyatsion hisoblash tillari.

Munosabatlar o'z mazmuniga qarab ikki sinfga ajratiladi:

a) Ob'ektli munosabatlar;
b) Bog'lanuvchi munosabatlar;

Ob'ektli munosabatlarda ob'ektlar haqidagi munosabatlar saqlanadi. Masalan, talaba munosabati. Bog'lanish munosabatlarida asosan, ob'ektli munosabatlarning kalitlari saqlanadi. Kalit atributlari oddiy va murakkab bo'lishi mumkin. Agar kalit

ikkita va undan ortiq atributdan tashkil topgan bo'lsa, murakkab hisoblanadi.

Nazorat savollari

1. Relyatsion ma'lumotlar modeli qanday farqlanadi?
2. Relyatsion ma'lumotlar bazasining asosiy tushunchalari.
3. Munosobat xossalariga nimalar kiradi?
4. Munosobatlar sxemasiga misollar keltiring.
5. Munosabat turlari nechta?
6. Relyatsion algebra amallarini sanab bering va misol keltiring.

5-mavzu. Relyatsion algebra va relyatsion hisobot elementlari

Reja

1. Munosabatlar ustida amallar.
2. Relyatsion ma'lumotlar bazasini asosiy tushunchalari.
3. Relyatsion algebra va uning amallari.
4. Relyatsion hisoblash elementlari va ulardan foydalanish.

Tayanch so'zlar: dekard, kesishuv, birlashtirish, seleksiya, ayirma, domen, relyatsion algebra, relyatsion hisoblash.

Munosabatlar ustida amallar. Munosabatlar ustida har xil amallarni bajarish imkoniyati mavjud. Relyatsion ma'lumotlar modelini xususiyatlaridan biri ma'lumotlarni qayta ishlashni relyatsion algebra operatorlari (amallari) yordamida amalga oshirishdir. Relyatsion algebrada quyidagi 8 ta operator keng ishlatiladi. Ulardan 4 tasi an'anaviy toplamlar ustida bajarilishi mumkin bo'lgan amallardir. An'anaviy amallarga quyidagilar kiradi.

1. Birlashtirish
2. Kesishuv
3. Ayirma
4. Dekart ko'paytma

Maxsus amallarga esa quyidagilar kiradi.

1. Tanlash (seleksiya)
2. Proeksiya
3. Qo'shish
4. Bo'lish

Munosabatlar ustida bajariladigan birlashtirish, kesishuv, ayiruv amallari operatorlarning tili yoki turi bo'yicha mosligini talab etadi. Ikkita munosabat tipi bo'yicha mos keladi, agarda ularda ekvivalent munosabat sxemasi bulib:

- ulardagi har bir daraja bir xil bo'lsa yoki ular bir xil atribut to'plamiga ega bo'lsa;
- sxema atributlarini shunday tartiblash mumkinki, bir xil o'rinda turib solishtirilayotgan atributlari bir xil domenda aniqlangan bo'lishi kerak.

Birlashtirish amaliga quyidagi misol ko'rib chiqiladi. Ikkita guruh jadvallari berilgan bo'lsin va bu jadvallar o'rtasida birlashtirish amalini bajarish talab etilsin.

5.1-j adval. 1 -guruh haqida ma'lumot

Familiyasi	**Ismi**	**Tug'ilgan yili**	**Manzili**	**Yoshi**	**Kursi**
Karimov	Alisher	1999	Toshkent	20	2
Odilov	Furqat	1998	Xorazm	23	3
Isaev	Qudrat	1997	Andijon	35	2
Aliev	Qosim	2000	Navoiy	49	4

5.2-jadval. 2 -guruh haqida ma'lumot

Familiyasi	**Ismi**	**Tug'ilgan yili**	**Manzili**	**Yoshi**	**Kursi**
Karimov	Alisher	1999	Toshkent	20	2
Ilxomov	Ali	1998	Xorazm	23	3
Eragshev	Yo'lchi	1997	Andijon	35	2
Azizov	Toshmat	2000	Navoiy	49	4

5.1 va 5.2 jadvallarni birlashtirish orqali 5.3 jadval hosil bo'ladi. Hosil bo'lgan

Jadvalning e'tiborli tomoni shundan iboratki, umumlashtirilgan jadvalda qaytarilgan qatorlar bir marta ishlatiladi. Bu jadvallarda birinchi qator ma'lumotlari bir xil bo'lganligi uchun bir marta ishlatilganligini ko'rish mumkin.

6.3-jadval. Birlashtirish amali natijasi

Familiyasi	**Ismi**	**Tug'ilgan yili**	**Manzili**	**Yoshi**	**Kursi**
Karimov	Alisher	1999	Toshkent	20	2
Odilov	Furqat	1998	Xorazm	23	3
Isaev	Qudrat	1997	Andijon	35	2
Aliev	Qosim	2000	Navoiy	49	4
Ilxomov	Ali	1998	Xorazm	23	3
Eragshev	Yo'lchi	1997	Andijon	35	2
Azizov	Toshmat	2000	Navoiy	49	4

Birlashtirish amalida quyidagi shartlar bajarilishi talab etiladi:

- jadvallardagi atributlar soni mos ravishda ustama - ust tushishi shart;
- atributlarning toifalari mos bo'lishi kerak;
- agar atributlar toifalari mos kelmaganda so'rovlar orqali moslartirish talab etiladi.

Relyatsion algebraning keying amali kesishuv amali bo'lib, unda tanlangan jadvallar ma'lumotlarining mos kelganlari aks ettiriladi. Yuqoridagi 5.1 va 5.2 jadvallaridan foydalanib kesishuv amaliga misol keltirilgan (5.4-jadval).

5.4-jadval. Kesishuv amali natijasi

Familiyasi	**Ismi**	**Tug'ilgan yili**	**Manzili**	**Yoshi**	**Kursi**
Karimov	Alisher	1999	Toshkent	20	2

Kesishuv amalida tanlangan jadvallardagi uchragan qatorlardagi ma'lumotlarning moslari ajratib olinadi. Bunda barcha atributlar qiymatlari va ularning toifalari mos kelishi talab etiladi.

Relyatsion algebraning keying amali ayirma amali bo'lib, unda tanlangan birinchi jadvaldagi ma'lumotlardan ikkinchi jadvalga uchraganlari ajratib tashlanadi. Bunda yuqoridagi amallar kabi barcha atribut qiymatlari mos kelishi va atribut toifalari usta - ust tushgan bo'lishi kerak.

Shu bilan bir qatorda natijada faqat birinchi jadval atribut qiymatlari aks ettiriladi. Ikkinchi jadval esa o'z ornida birinchi jadvaldan mos qiymatlarni olib tashlash uchun xizmat qiladi. Agar ikkinchi jadvaldan birinchisini ayirish kerak bo'lsa jadvallar o'rnini almashtirish orqali amalga oshiriladi (5.5-jadval).

5.5-jadval. Ayirish amali natijasi

Familiyasi	**Ismi**	**Tug'ilgan yili**	**Manzili**	**Yoshi**	**Kursi**
Odilov	Furqat	1998	Xorazm	23	3
Isaev	Qudrat	1997	Andijon	35	2
Aliev	Qosim	2000	Navoiy	49	4

Relyatsion algebraning yana bir amali dekart ko'paytma amali bo'lib, unda tanlangan birinchi jadvalning har bir qatoriga ikkinchi jadvalning har bir qatori mos kelishi tushuniladi. E'tiborli tomoni shundan iboratki, boshqa amallarda satrlar birlashtirilgan bo'lsa dekart ko'paytma amalida esa atributlar birlashtiriladi. Dekart ko'paytmada munosabat operatorlari har-xil sxemada bo'lishi mumkin.

5.6-jadval. Talaba jadvali

Familiyasi
Alimov
Ashurov
Oripov

5.7-jadval. Fan jadvali

Fan	Sana
Matematika	09.01.2019
Tarix	14.01.2019

Matematik munosabatlar darajasi operant munosabat darajalarining yig'indisiga teng. Quvvati esa operant quvvatlarini ko'paytmasiga teng. Quyidagi jadvalda 5.6 va 5.7 jadvallardan foydalanib dekart ko'paytma keltirilgan.

5.8-jadval. Dekart ko'paytma natijasi

Familiya	Fan	Sana
Alimov	Matematika	09.01.2019
Alimov	Tarix	14.01.2019
Ashurov	Matematika	09.01.2019
Ashurov	Tarix	14.01.2019
Oripov	Matematika	09.01.2019
Oripov	Tarix	14.01.2019

Seleksiya (tanlash) amali 1 ta munosabat ustida bajariladi. Natija munosabatda biror shart bo'yicha tanlab olingan kortejlar qatnashadi.

Qo'shish amali ikkita operant ustida bajariladi. Har bir munosabat qaysi atribut bo'yicha qo'shish bajarilayotgan bo'lsa, u ajratiladi.

Natija munosabat 1 va 2-munosabatni barcha atributlarini o'z ichiga oladi. Misol tariqasida 5.9 va 5.10- jadvallardan foydalanib seleksiya amali ko'rsatilgan (5.11jadval).

5.9-jadval. Guruh ma'lumotlari

Mutaxassislik	Talaba kodi
Matematika	1
Fizika	3
Ximiya	4

5.10-jadval. Talaba ma'lumotlari

Talab kodi	Familiya	Kurs
1	Diyorov	1
2	Sattorov	1
3	Pulatov	2
4	Ashurov	3

5.11-jadval. Seleksiya amali natijasi

Mutaxassislik	Talaba kodi	Familiya	Kurs
Matem	1	Diyorov	1
Fizika	3	Pulatov	2
Ximiya	4	Ashurov	3

Har bir munosabatda kortejlar identifikator kalitiga ega bo'lishi kerak. Kalit quyidagi ikkita xossaga ega bo'lishi zarur:

3. Kortej kalit qiymati bilan bir qiymatli ifodalanishi kerak;
4. Kalitda ortiqchalik bo'lmasligi kerak. Bu degani hech qanday atributni kalitdan olib tashlash mumkin emas.

Relyatsion ma'lumotlar bazasida malumorlarni ortiqchaligini normallashtirish yo'li bilan kamaytiriladi. Jadvallar ustida har xil amallar bajarish mumkin. Amallarga quyidagilar kiradi:

- To'plamlar ustida birlashtirish, kesishuv, ayirma, dekart ko'paytma va bo'lish amallari kiradi.
- Maxsus relyatsion amallar, ularga: proeksiya, birlashtirish, ajratish (tanlab olish) amallari kiradi.

Munosabatlar ustida amalni bajarish uchun ishlatiladigan tillarni ikki sinfga ajratishimiz mumkin:

c) Relyatsion algebra tillari;
d) Relyatsion hisoblash tillari.

Munosabatlar o'z mazmuniga qarab ikki sinfga ajratiladi:

c) Ob'ektli munosabatlar;
d) Bog'lanuvchi munosabatlar;

Ob'ektli munosabatlarda ob'ektlar haqidagi munosabatlar saqlanadi. Masalan, talaba munosabati.

Bog'lanish munosabatlarida asosan, ob'ektli munosabatlarning kalitlari saqlanadi. Kalit atributlari oddiy va murakkab bo'lishi mumkin. Agar kalit ikkita va undan ortiq atributdan tashkil topgan bo'lsa, murakkab hisoblanadi.

Familiya	Kurs	Mutaxasislik
Sobirov	2	Matematika
Aliev	4	Fizika
Xabirov	3	Ximiya

Talaba	Fan
Sobirov	Algebra
Aliev	Tarix
Aliev	Algebra
Xabirov	Programmairovaniya

Relyatsion algebra va uning amallari.

Relyatsion MBBTda ma'lumotlar bilan ishlash uchun bir qancha tillar yaratilgan. Ba'zi hollarda bu tillarni ma'lumotlarni qism tillari deb ataladi. MB bilan ishlovchilar bu tillarda avtomatlashtirishni 3 bosqichga bo'lishadi:

1. Eng pastki bosqich - kortej deb ataladi. Bunda dasturchi yozuvlar yoki kartijlar bilan ishlaydi.
2. Relyatsion algebra deyiladi. Bunda foydalanuvchi munosabatlar ustida yuqori bosqichli amallar to'plamini kiritadi.
3. Eng yuqori bosqich - hisoblash bosqichi. Bunda foydalanuvchi bevosita kompyuterga maxsus tillarda murojaat qiladi va mashina bu murojaatni qabul qiladi.

Relyatsion algebra amallarini opperandlari sifatida doimiy yoki o'zgarmas va o'zgaruvchan munosabatlar ishlatiladi. Relyatsion algebrada 5 ta amal ishlatiladi:

1. Birlashtirish, R va S munosabatlarni birlashtirish RUS ko'rinishida berilib, bu amalni natijasi R munosabatga tegishli bo'lgan yoki S munosabatga tegishli bo'lgan yoki ikkalasiga ham tegishli bo'lgan kartejlar to'plamidir. Bu amallarni bajarayotganda bir xil tartibda bo'lishi kerak. Natijani tartibi ham operandlar tartibiga teng bo'ladi.
2. Ayirma R va S munosabatlarni ayirmasi R-S ko'rinishida yoziladi va undagi kortejlar to'plami R munosabatga tegishli, lekin S munosabatga tegishli bo'lmagan kortejlardir. Bu amalni bajarganda ham operandlarni tartibi bir xil bo'lishi kerak.
3. Dekart ko'paytma. Bizda R va S munosabat berilgan bo'lsin. R munosabatni tartibi R-R va S munosabatniki S-q ga teng bo'lsin. Unda dekart ko'paytma R*S ko'rinishida yozilib, uning natijasi uzunligi R+q ga teng bo'lgan kortejlar to'plamidan iborat bo'lib, bu kortejlarni birinchi R komponentasi R kortejga teng bo'ladi, qolgan q komponentasi S kortejga teng bo'ladi.
4. Proeksiya, R munosabatga bu amal tadbiq etilganda, R munosabatdan ba'zi bir komponentalar olib tashlanadi. Qolganlari esa qaytadan tartiblanadi.

5. Seleksiya tanlash. Bu amal bajarilganda operandlar sifatida munosabat atributlari ishtirok etadi va solishtirish arifmetik amallari: =, Φ, <, >, <, > va mantiqiy amallar: va (U), yoki (V), not amallari ishlatiladi.

Relyatsion MBBTda ma'lumotlar bilan ishlashda ishlatiladigan 2 ta katta gurux tillari relyatsion hisoblash deyiladi. Relyatsion hisoblash predikatlarni hisoblashga asoslangan bo'lib ifodalarni yozishga mo'ljallangan qiodalar to'plamidan iboratdir. Ular yordamida biz mavjud munosabatlardan yangi munosabatlar yaratishni ta'minlaymiz. Bunday ifodaalrni yozishda solishtirish amallari, mantiqiy amallar va mavjudlik kvanteri va umumiylik kvanteri ishlatiladi.

Hozirgi paytda relyatsion MBBTni taraqqiyotida yangitil QBE tili ishlamoqda. Bu tilda relyatsion algebra va relyatsion hisoblashlarda ko'zda tutilmagan bir qpncha imkoniyatlar kirgan. Bu tilni hususiyati shundan iboratki, u terminallarda ishlashga muljallangan. So'rovlarni yaratish uchun maxsus ekran redaktoridan , munosabat va redaktorlaridan foydalanamiz. QBE tilida foydalanuvchi o'zi olishini mo'ljallagan natijani so'rov ko'rinishida tasvirlaydi va MBBT uni kerakli amallar ketma - ketligiga aylantirib beradi.

Ma'lumot modelini rivojlanish konsepsiyasi 5 ta bosqichni ko'rsatishi mumkin:

1. 60- yillarning 2 - yarmida, bunda asosan ierarxik modellarga e'tibor berilgan;
2. 70- yillarni 1 - yarmi, tarmoqli modellar;
3. 70- yillarning 2 - yarmi, relyatsion modellar;
4. 80- yillarning 1 - yarmi, semantik modellar;
5. 80- yillarning 2 - yarmi, ob'ektga mo'ljallangan sistema.

Nazorat savollari

1. Relyatsion ma'lumotlar bazasini asosiy tushunchalari.
2. Munosobat xossalari qanday?
3. Munosobatlar sxemasiga misollar keltiring.

4. Relyatsion algebra amallarini aytib o'ting.
5. Relyatsion hisoblash amallarini ayting va misol keltiring.

6-mavzu. Ma'lumotlar bazasini rejalashtirish, loyihalash va administrate rlash

Reja

1. Ma'lumotlar bazasini hayot siklini tashkil etish.
2. Ma'lumotlar bazasini rejalashtirish.
3. Ma'lumotlar bazasini loyihalash.
4. Ma'lumotlar bazasini administratorlash.
5. Ma'lumotga samarali murojaatni tashkil qilishda bazalar o'zaro aloqasi fayl tuzilmalaridan foydalanish.
6. Ma'lumotlar bazasida aloqadorlik chegaralari va xavfsizlik choralarini tasvirlash.

Tayanch so'zlar: administratorlash, rejalashtirish, birlashtirish, loyihalash.

MB administratori deyilganda birorta shaxs yoki bir necha shaxslardan iborat bo'lgan va MBni loyihalash, uzatish va samarador ishlashini ta'minlovchidir.Ma'lumotlar bazasi tushunchasi bilan **ma'lumotlar banki** tushunchasi ham mavjud. Ma'lumotlar banki (MBn) tushunchasi ikki xil talqin etiladi.

Hozirgi kunda ma'lumotlar markazlashmagan holda (ishchi o'rinlarda) ShK yordamida qayta ishlanadi. Ilgari ular alohida xonalarda joylashgan hisoblash markazlarida markazlashgan holda qayta ishlangan. Hisoblash markazlariga axborotlar tashqi qurilmalar orqali kelib to'plangan. Ma'lumotlar bazasi markazlashgani hisobiga ularni ma'lumotlar banki deb atashgan. Bunda ma'lumotlarga murojat etish ishchi stansiyalardan markazlashgan holda tashkil etilgan va shuning uchun ma'lumotlar banki bilan ma'lumotlar bazasi tushunchalari o'rtasida farq qilingan.

Hozirgi kunda ko'p hollarda ma'lumotlar bazasi markazlashmagan holda tashkil qilinmoqda. Shuning uchun ma'lumotlar banki va ma'lumotlar bazasi sinonim so'zlar sifatida ham ishlatiladi.

1. Boshqacha talqinda, ma'lumotlar banki deyilganda ma'lumotlar bazasi uni boshqarish tizimi(MBBT) tushuniladi. Ma'lumot bazasi bilan ishlaydigan dasturni ilova deb ataladi. Bitta ma'lumot bazasi bilan juda ko'p ishlashi mumkin.

MB ni ishlatish afzalliklari:

- Ixchamligi;
- Axborotlarni qayta ishlash tezligini oshishi;
- Kam mehnat sarfi;

SQL Server Enterprise Manager dasturi yordamida SQL Server himoya qilish vositalaridan foydalanish mumkin. Bu himoya vositalari haqidagi ma'lumotni SQL Server hujjatlaridan olish mumkin.

Agar shifrlangan tasavvur strukturasini keyinchalik o'zgartish kerak bo'lishi mumkin bo'lsa quyidagi maslahatdan foydalaning. Tasavvurni aniqlovchi SQL yoriqnomani matnli faylda saqlab qo'ying. Ishonchli joyda mustahkam nusxani saqlab qo'ying. Tasavvurni shifrlang. Kerak bo'lsa shifrlangan tasavvur struk-turasini o'zgartiring:

Oldingi shifrlangan tasavvurni o'chirish.

Oldingi tasavvur bilan bir xil nomdagi yangi tasavvur yarating.

Saqlangan matnli fayldagi SQL yoriqnomadan almashish buferiga nusxa oling. Uni yangi tasavvur Konstruktorining SQL yo'l-yo'riq kiritish maydoniga joylashtiring.

Tasavvur strukturasini o'zgartirish.

O'zgartirilgan SQL yoriqnomani matnli faylda saqlang. Bu faylni ishonchli joyga joylashtiring.

SQL Server hisob yozuvlarini boshqarish

MB himoya tizimini boshqarish vazifasini (Tools) menyusidagi (Database Security) buyrug'i yordamida bajarish mumkin. Agar SQL Server loyihasi saqlanayotgan kompyuterda o'rnatilgan bo'lsa bu buyruqqa murojaat qilish

mumkin. Bu vosita yordamida SQL Serverda registratsiya qilish uchun hisob yozuvlarini, ma'lumotlar bazalari foydalanuvchilari hisob yozuvlarini va ularning vazifalarini qo'shish, o'chirish va o'zgartirish mumkin.

SQL Serverda registratsiya qilish uchun qo'llanadigan ikki himoya tizimi mavjud:

- SQL Server o'zining himoya tizimi. Serverda registra- tsiyadan o'tish uchun server foydalanuvchisi nomi va parolini ko'rsatish kerak.
- Windows NT bilan Integratsiyalashgan tizimi foydalanuvchilari hisob yozuvlaridan foydalanadi. Bu holda foydalanuvchi autentifikatsiyasi Windows NT asosida tarmoqda registratsiyadan o'tishda bajariladi.

SQL tilida protseduralardan foydalanish dasturlar tuzish samaradorligini oshiradi. Saqlanuvchi protseduralar (stored protsedure) - bu SQL buyruqlar to'plamidan iborat bo'lib, bu buyruqlar to'plamini SQL SERVER bir marta kompilyatsiya qiladi.

Protseduralarning keyingi ishlatilishida saqlangan protseduralar kompilyatsiya qilinmaydi. Bu protseduralar xuddi algoritmik tillardagi kabi kirish parametrlaridan iborat bo'lishi ham mumkin.

Saqlanuvchi protseduralar SQL tilida quyidagi buyruq yordamida yaratiladi:

CREATE PROCEDURE <protsedura nomi>

[(% birinchi parametr ma'lumoti turi)] ...] AS SQL-operatorlari;

Saqlanuvchi protseduralarning ikki turi mavjud: foydalanuvchi protseduralari va tizimli protseduralar.

Foydalanuvchi protseduralari SQL SERVERlarida qo'llanilib, serverni boshqarish, MB va foydalanuchilar haqidagi ma'lumotlarni olish uchun ishlatiladi. Tizimli protseduralar esa, amaliy dasturlarni bajarish uchun yaratiladi. Amaliy dasturlar hech bo'lmaganda bitta modulni o'zida saqlashi kerak. Modul (MODULE) biror bir algoritmik tilda tuzilgan, uzoq muddat saqlanadigan ob'yektdir.

Modul - modul nomidan (module name), algoritmik til bo'limidan (language clause), modul bo'limi huquqidan (module authorization clause), kursorlarni

tavsiflash (declare cursor) va bir yoki bir nechta protsedura (procedure) lardan tashkil topadi.

MBBTdan ikki gurux shaxslari foydalanadi:

3. Chekli yoki oddiy foydalanuvchilar;
4. MB administratori;

MB administratorini xizmat doirasiga quyidagi ishlar kiraddi:

j) Predmet sohani tahlili va foydalanuvchilar va axborotni o'rnini aniqlash;
k) Ma'lumotlarni tuzilishini loyihalash va ularni takomillashtirish;
l) Qo'yilgan topshiriqlar va ma'lumotlarni bir butunligini ta'minlash;
m) MBni yuklash va yuritish;
n) Ma'lumotlarni himoya qilish;
o) MBni tiklashni ta'minlab berish;
p) MBga murojaatlarni yiqish va statistik qayta ishlab berish;
q) MBga ko'p foydalanuvchilar rejimida ishlaganda, ma'lumotlarni o'chib ketishidan ximoya qilish;
r) Texnik vositalar nosoz bo'lib ishdan chiqqanda, ma'lumotlarni saqlash va qayta tiklash ishlarini bajarish;

Nazorat savollari

1. Relyatsion ma'lumotlar bazasini rejalashtirish tushunchalari.
2. Ma'lumotlar bazasini hayot siklini keltiring.
3. Ma'lumotga samarali murojaatni tashkil qilishga misollar keltiring.
4. Ma'lumotlar bazasini administratorlash usullari.
5. Ma'lumotlar bazasida aloqadorlik chegaralari tarifini keltiring.

7-mavzu. Ma'lumotlar bazasini normallashtirish: 1NF, 2NF, 3NF va Kodd normal formalari

Reja

1. Ma'lumotlar bazasini normallashtirish.

2. Funksional bog'lanishlar va ularning turlari.
3. Birinchi normal forma va uning talablari.
4. Ikkinchi normal forma va uning talablari.
5. Uchinchi normal forma va uning talablari.
6. Kodd normal formasi.
7. Berilgan munosabatni bir necha marta oddiy va kichik munosabatlarga ajratish.

Tayanch iboralar: funksional bog'lanish, tranzitiv bog'lanish, normal forma, normallashtirish, 1 NF, 2 NF, 3 NF.

Ma'lumotlar bazasida axborotlarni ko'payishi hisobiga dinamik ravishda o'zgarib turadi. Unda yangi ma'lumot elementlari qo'shiladi. Ular orasida yangi aloqalar yoki bog'lanishlar o'rnatiladi va ularni qayta ishlashni yangi usullari qo'llaniladi. Bu jarayonda imkoni boricha foydalanuvchi yaratgan MB bilan ishlash uchun yaratilgan dastur ilovasini kam o'zgartirishga harakat qiladi. Bu muammoni hal qilish uchun ma'lumot elementlarini asosli ravishda guruhlarga birlashtirish va ular uchun kalitlarni aniqlash yo'li bilan hal qilinishi mumkin. Hozirgi kunda axborot tizimlari ishlab chiqaruvchilar ma'lumotlarni uchinchi normal formada tasvirlab ishlatishni taklif etadilar.

Funksional bog'lanish tushunchasi. Relyasion MBda ma'lumotlarni tuzilmasidan tashqari ularni sxematik axborotiga ham etibor beriladi. MBni tuzilmasi haqidagi axborot munosabat sxemasi yordamida beriladi. Sxematik axborotlar esa atributlar orasidagi funksional bog'lanishlar orqali ifodalanadi. MB munosabatlarida atributlarni tarkibini quyidagi talablarga javob beradigan qilib guruhlash kerak:

- Atributlar orasidagi zaruriy bo'lmagan takrorlanishlar bo'lmasligi kerak.
- Atributlarni guruhlaganda ma'lumotlar takrorlanishi minimal darajada qilib ta'minlanishi kerak. Bu bevosita ma'lumotlarni tez qayta ishlash imkonini beradi. Bunga normallashtirish jarayoni yordamida erishiladi.

Normallashtirish deganda berilgan munosabatni bir necha marta oddiy va

kichik munosabatlarga ajratish tushuniladi. Bu jarayonda mumkin bo'lgan barcha funksional bog'lanishlar aniqlanadi.

Masalan A va B atributlar berilgan bo'lsin. Agar ixtiyoriy vaqtda A atributni bittadan ortiq bo'lmagan qiymati mos kelsa, unda B atributda funksional bog'langan deyiladi va quyidagicha belgilanadi:

A — B Shaxsiy raqam — Familiya

Bog'lanishlar

Mansabi — Maosh

7.1-jadval. Birinchi normal forma

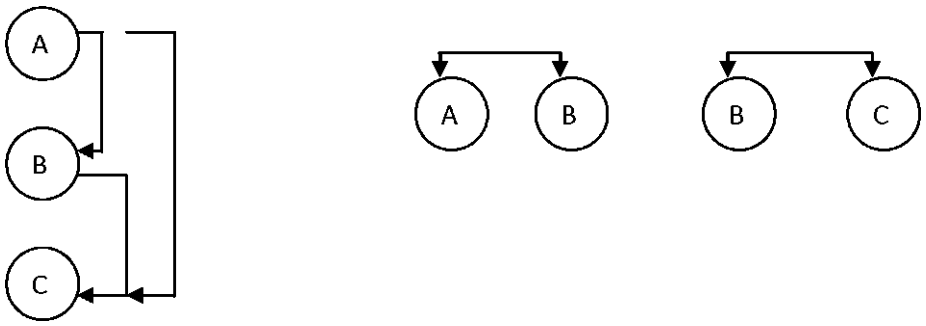
Shaxsiy raqam	Predmet nomi	Soatlar soni	Familiya	Mansabi	Maoshi	Kafedra	Tel.
201	EHM	36	Ergashev	Dots.	70000	EVM	4-89
201	SHK	72	Ergashev	Dots.	70000	EVM	4-89
202	MBBT	48	Komilov	Dots.	70000	EVM	4-89
301	MBBT	48	Babaev	Prof.	100000	ASU	5-19
401	Fizika	52	G'aniev	Ass.	50000	FE	4-12
401	Optika	20	G'aniev	Ass.	50000	FE	4-12

Agar munosabat 1-normal formada bo'lsa - 1NF, unda barcha kalit bo'lmagan atributlar kalit atributga funksional bog'langan. Lekin, bog'lanish darajasi har xil. Agar kalit bo'lmagan atribut kalit atributni qismiga bog'langan bo'lsa, u qisman bog'lanishli deyiladi. Bizning misolda soatlar soni (kalit bo'lmagan atribut) predmetlar nomi atributiga qisman bog'langan. Agar kalit bo'lmagan atribut barcha murakkab kalitga bog'langan bo'lsa va uni qismiga bog'langan bo'lmasa, unda bu atributni murakkab kalitga to'la funksional bog'lanish deyiladi. Agar, A,B,S atributlar berilgan bo'lsa va unda A — B bo'lsa, B—S bo'lsa, unda S A dan tranzitiv bog'langan bo'ladi. Bizni misolda familiya, kafedra, telefon.

Uchinchi normal forma (3 NF). Ma'lumotlar munosabatlarda 2 NFga keltirilganda ham bir qancha noqulayliklar bo'ladi. Jumladan, ma'lumotlarda

axborotlarni ortiqchaligi, amallarni bajarish qiyinligi va boshqalar. Bunday munosabatlarni 3 NFga keltiriladi.

Agar, A,B,S R munosabatini 3 ta atributi yoki atributlar to'plami bo'lsin. Agar B atribut A atributga, S atribut esa B atributga bog'langan bo'lsa, Ya'ni , A — B va B—S. Bunda teskari bog'linishlar bo'lmasa, unda S atribut A atributga tranzitiv bog'langan deyiladi. Uni ko'pincha diagramma ko'rinishida quyidagicha belgilaymiz:



7.1-rasm. 3 NFga keltirishning diagramma ko'rinishi

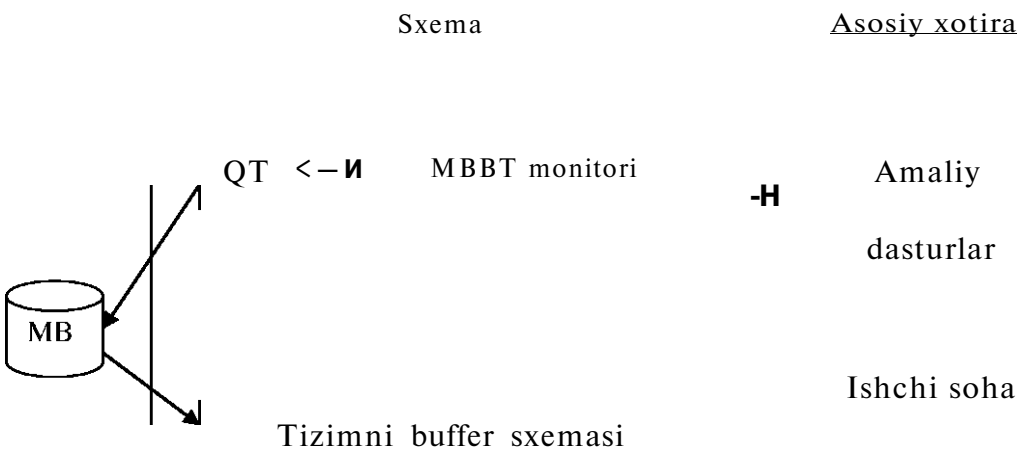
Shunday qilib, R munosabat 3 NFda berilgan deyiladi, agarda, u 2 NFda bo'lsa va R munosabatdagi birlamchi kalit bo'lmagan har bir atribut R munosabatni har har bir mumkin bo'lgan kalit atributiga notranzitiv bog'langan bo'lsa. Umuman olganda normallashtirish jarayoni va munosabatni 3 NFga keltirish quyidagi bosqichlardan iborat bo'ladi:

1. Ma'lumotlarni ixtiyoriy tuzilmasidan oddiy tuzilali ikki o'lchamli jadvallarga o'tish va 1 NFni hosil qilish;
2. Kalit atributlari bilan barcha atributlar orasidagi mumkin bo'lgan to'liqmas funksional bogTanishlarni yo'qotish va 2 NF hosil qilish;
3. Mumkin bo'lmagan kalit atributlari va asosiy bo'lmagan atributlar orasidagi tranzitiv bogTanishlarni yo'qotish va 3 NFni hosil qilish.

Ma'lumotlar bazasi va MBBT ni fizik tashkil etish.

MBBT komponentalari va ularni amaliy dasturlar bilan o'zaro bog'liqligi ma'lumotlarni fizik tasvirlashda muhim o'rin to'tadi. MBBT murakkab til dastur

kompleksidan iborat bo'lib, MBni ishlash imkoniyatini ta'minlaydi. MBBT tarkibiga tizimli dasturlar kompleksi kiradi. Bu kompleksni markaziy komponentasi monitor yoki boshqaruvchi dasturlar hisoblanadi. Komponentalarning fizik tashkil etuvchilari 7.2-rasmda berilgan.



7.2-rasm. MBBTning fizik tashkil etuvchilari

7.2-rasmda amaliy dasturlar tarkibidagi ma'lumotlar bilan ishlash tili (YAMD)ni bitta operatorini bajarishiga tegishli bo'lgan amallar ketma - ketligi ko'rsatilgan.

Masalan, bu MBdan ma'lumotlarni o'qish so'rovini operatori bo'lib xizmat qilsin. Unda strelkalar quyidagi ma'noga ega:

1. Amaliy dasturlar MBga (YAMD) operatori orqali murojaat qilsin. Uni monitor tomonidan tahlil qilinadi.

2. Talqin qilish jarayonida monitor oldindan translyasiya qilib qo'yilgan sxemani ishlatadi.

3. Bu so'rovga tegishli ma'lumotlar aniqlanib bo'lingandan keyin, monitor OTga tashqi xotiraga murojaat qilishni amalga oshirish talabi bilan murojaat qiladi.

4. OT MBga murojaatni bajaradi. Bu xuddi fayllarga murojaat qilish kabi oddiy bajariladi.

5. Talab qilingan ma'lumotlar tashqi xotiradan tiimni bufer sohasiga o'zatiladi.

6. Ma'lumotlar amaliy dasturlarni ishchi sohasiga jo'natiladi.

7. Monitor amaliy dasturga so'rovni bajarish natijalari xabarini beradi.

8. Amaliy dastur MBdan olingan ma'lumotlar ustida kerakli amallarni bajaradi.

Manzillash usullari. Bitta mashina ko'rsatmasi yordamida o'qish mumkin bo'lgan bitlar guruhi fizik yozuvlar deb ataladi. Fizik yozuvlar mashina xotirasining yacheykalarida saqlanadi va mashina adreslari yordamida identifikatsiyalanadi. Dasturlar mantiqiy yozuvlarni kalitlar yordamida aniqlaydi. Dastur uchun zarur bo'lgan ma'lumotni mantiqiy yozuv kalitlari yordamida fizik yozuvlarni adreslarini aniqlaydi. Dastur uchun zarur bo'lgan ma'lumotni mantiqiy yozuv kalitlari yordamida fizik yozuvlarni manzillarini aniqlanadi. Kalit qiymatlari juda ko'p bo'lganligi uchun mashina manzillar bilan munosiblikni aniqlash uchun xilma - xil manzilash usulidan foydalaniladi. Kalit sifatida har bir yozuvda joylashgan piksellangan uzunlikdagi maydonlardan foydalaniladi. Ba'zi hollarda kalit sifatida bir nechta maydon olinadi va bunda ulangan kalitlar hosil qilinadi. Fayllardagi yozuvlarni bir qiymatli aniqlash uchun albatta yagona kalit mavjud bo'lishi kerak va bunday kalitlar birlamchi kalitlar deb ataladi.

Yozuvlarni manzillashning quyidagi usullari mavjud:

1. Fayllarni ketma - ket saqlash usuli. Har bir yozuv kaliti tekshiriladi. Bunday usul ko'p vaqtni talab etadi.

2. Blokli qidirish. Agar yozuvlar kalit bo'yicha tartiblangan bo'lsa, fayllarni skanerlashda har bir yozuvni o'qib chiqish talab etilmaydi. Bunday ho llada kerakli yozuvdarni topish uchun blokli qidirish usulidan foydalaniladi. Bunda yozuvlar bloklarga guruhlanadi va har bir blok bir martadan tekshiriladi, kerakli yozuv qidirib topilguncha.

3. Binar qidirish. Bunda soha o'rtasidagi yozuv topiladi va uning kaliti qidirish tartibi bilan solishtiriladi. So'ngra qidirish sohasi ikkiga ajratiladi va har bir yarmi alohida qidiriladi. Binar qidirish to'g'ridan - to'g'ri murojaat qurilmalarida ishlatib bo'lmaydi.

Nazorat savollari

1. Munosabotlar nima maqsadda normallashtiriladi?
2. Munosabotlar atributlariga qanday talablar qo'yiladi?
3. Funksional bog'lanish turlari ayting.
4. 1 NF va undagi shartlar qanday?
5. Qanday qilib 1 NFdan 2 NFga o'tiladi?
6. Qanday qilib 2 NFdan 3 NFga o'tiladi?

8-mavzu. SQL tili va SQL operatorlarini yozish.

Reja

1. SQL tilining vazifalari.
2. Interaktiv va qurilgan SQL.
3. SQL tilida ma'lumot toifalari va ular bilan ishlash.
4. SQL tilining komandalarini tuzilishi va sintaksisi.
5. SQL tilining SELECT (tanlash) operatori va uning parametrlari.

Tayanch iboralar: SQL, SELECT, DISTINCT, DDL, DML, VAR, VARCHAR, FROM, WHERE, ORDER BY ,GROUP BY ,HAVING , UNION.

Ma'lumotlar bilan ishlash uchun mo'ljallangan MBBT ichki tili ikki qismdan tashkil topgan:

- ma'lumotlarni tasvirlash tili (Data Definition Language (DDL));
- ma'lumotlarni manipulyasiya qilish tili (Data Manipulation Language (DML)).

Ma'lumotlar bazasini boshqarish tizimida DDL tili ma'lumotlar bazasi arxitekturasini tashkil etishda foydalaniladi va ma'lumotlar bazasi tuzilmasini aniqlash va ma'lumotlarga murojaatni boshqarish uchun mo'ljallangan.

Ma'lumotlar bazasini boshqarish tizimida DML tili esa ma'lumotlar bazasida amallar bajarish va ma'lumotlarni qayta ishlash uchun foydalaniladi va ma'lumotlarni ajratish va tiklash uchun mo'ljallangan.

Bu ichki tillar ma'lumotlarni qism tillari va yuqori darajali dasturlash tillari deyiladi, chunki ularni tarkibida barcha hisoblarni bajarish uchun zarur bo'ladigan til kostruksiyalari bo'lmaydi (shartli o'tish amallari).

Ma'lumotlar bazasida ishlaydigan ixtiyoriy til foydalanuvchiga quyidagi imkoniyatlarni berishi kerak:

- ma'lumotlar bazasi arxitekturasi va tafsifini yarata olish;
- ma'lumotlarni manipulyasiya qilishni asosiy amallarini, jumladan jadvalga ma'lumotlarni kiritish, o'zgartirish, o'cherish, tanlash va ularni takomillashtirish, jadvaldan ma'lumotlarni olib tashlash;
- oddiy va murakkab so'rovlar yaratish.

SQL tilining vazifasi. Ma'lumotlar bazasi bilan ishlash tillari belgilangan masalalarni ortiqcha harakatlarsiz hal qilish imkoniyatini berishi kerak. Tilning komandalarini tuzilishi va sintaksisi yetarli darajada sodda va foydalanishga oson bo'lishi kerak. Bundan tashqari u universal bo'lishi va qandaydir standart talablariga javob berishi kerak. Bu esa uni komanda strukturasini va sintaksisini bir qancha MBBTishlatishini imkonini beradi. Bu talablarni barchasiga SQL javob beradi. SQL tili - bu (Strusture Query Language), ya'ni strukturalangan so'rovlar tili hisoblanadi.

SQL tili operatorlarni erkin formatda yozilishini ta'minlaydi. Buning ma'nosi, operatorlar elementlarini yozilishi ekrandan fiksirlangan joylarga bog'liq emas.

Komanda tuzilmasi bir qancha kalit xizmatchi so'zlar bilan beriladi, masalan:

CREATE TABLE - jadval yaratish;

INSERT - ma'lumot kiritish;

SELECT - ma'lumotlarni tanlab olish.

SQL operatori xizmatchi so'zlar va foydalanuvchi qo'llaydigan so'zlardan tashkil topadi.

Xizmatchi so'zlar SQL tilining doimiy qismi bo'lib, ular aniq qiymatga ega. Ularni standartda ko'rsatilgandek yozish kerak va ularni bir satrdan ikkinchisiga ko'chirish mumkin emas. Foydalanuvchi tomonidan aniqlangan so'zlar, foydalanuvchi tomonidan ma'lum sintaksis qoidalari asosida beriladi. SQL tilida operatorlar o'rnatilgan sintaksis qoidalariga moslab joylashtiriladi. Til standartida bu ko'rsatilmagan bo'lsa ham, SQL tilining ko'rinishida matn tugallanganini bildiruvchi belgi, ko'pgina hollarda nuqtali vergul (**;**) ishlatiladi.

SQL operator komponentalarini ko'pchiligi registrga bog'liq emas, Ya'ni ixtiyoriy har qanday katta va kichik harflar ishlatishi mumkin. Bularda bitta istisno bor. Bu istisno simvollarga tegishli. Ularga mos bo'lgan ma'lumotldar bazasidagi qiymatlar qanday saqlansa shunday yozilishi kerak. Masalan agar ma'lumotlar bazasida familiyaning qiymati "Qosimov" ko'rinishida bo'lsa, qidirish shartida "Qosimov" ko'rinishida berilmasa, bunga tegishli yozuv hech qachon topilmaydi.

SQL tili erkin formatga ega bo'lgani uchun SQL alohida operatorlari va ularning ketma-ketligini alohida ajratib yozish yozis mumkin. SQL tilidan foydalanishda quyidagi qoidalarga bo'ysunish talab etiladi:

- operatordagi har bir konstruksiya yangi satrdan boshlanishi kerak;
- har bir konstruksiya boshlanishida tashlab ketiladigan bo'sh pozitsiyalar, boshqa operator konstruksiyalarida ham bo'lishi kerak;
- -agar konstruksiya bir necha qismdan iborat bo'lsa, ularning har biri yangi satrlardan bo'sh o'rinlarni oldingi konstruksiyaga nisbatan siljitib yoziladi.

Amaliyotda ma'lumot bazasi tuzilishini (asosan uni jadvallarini) aniqlash uchun **DDL** operatorlari ishlatadi. Bu jadvallarni ma'lumotlar bilan to'ldirish uchun va ulardan axborotlarni so'rovlar yordamida ajratib olish uchun - **DML** operatorlari ishlatiladi.

Ma'lumotlarni manipulyasiyalash SQL tilini **DML** peratorlari qo'llaniladi.

Interaktiv va qurilgan SQL. SQL tilini ikkita shakli mavjud.

- interaktiv SQL

- qurilgan (kiritilgan) SQL

Interaktiv SQLda foydalanuvchi SQL so'rovlar bevosita MBBT orqali kiritadi va natijalarini interaktiv rejimda olinadi.

Qurilgan SQLda esa so'rovlar SQL komandalaridan tashkil topib, u boshqa birorta dasturlash tili (java, C++, C, Delphi va hokazo)ga yozilgan dastur ichiga joylashtiriladi. Bu shunday tillarni ishlatadigan dasturlarni samaradorligini oshiradi. Ularga relyasion ma'lumotlar bazasi bilan ishlash imkonini beradi.

SQL ma'lumot toifalari. Simvollar satr ma'lumot toifasi SQL standartida matnlarni faqat bitta tavsifi keltiriladi. Uning sintaksisi quyidagicha:

CHARACTER[(uzunligi)] yoki **CHAR**[(uzunligi)]

Jadvalda belgilangan atrubutlar **CHAR** toifasiga tegishli bo'lishi mumkin. Bunda bu atributlar qiymatlari 1 dan 255 tagacha belgidan iborat bo'lishi mumkin. SQL tilini ba'zi birlardagina o'zgaruvchan uzunlikdagi satr toifalari mavjud. Bu toifalar quyidagicha tavsiflanadi:

VARCHAR() yoki **CHARVARYING()**

CHARACTER va **VARCHAR** toifasidagi konstantalar apostrof ichiga yoziladi.

Quyidagi toifalarni barchasi ekvivalent SQLda bir xil vazifa bajaradi:

VARCHAR [(uzunligi)];

CHARVARYING [(uzunligi)];

CHARACTER VARYING[(uzunligi)].

Agar uzunlik oshkor ko'rsatilmasa, u holda 255 ga teng deb qabul qilinadi, Ya'ni barcha hollarda 255 ta simvoldan iborat bo'ladi.

Sonli ma'lumot toifalari SQL standartida quyidagi son toifalarida ishlatiladi.

INTEGER - butun sonlar uchun - 2^{-31}.. ..2^{31};

SMOLLINT - butun sonlar 2^{-15}... 2^{15};

DECIMAL (aniqlik[masshtab]) - fiksirlangan nuqtali son. Aniqlik sondagi qiymatli raqamlar masshtab. Unli nuqtadan undagi raqamlarning maksimal sonini ko'rsatadi;

NUMERIC - butun sonlar uchun - 2^{-31}.. ..2^{31};

FLOAT[(aniqlik)] - haqiqiy sonlar uchun qo'llaniladi.

Bundan tashqari uzgaruvchan uzunlikdagi simvolli satrlar toifasi ham ishlatiladi. Bunda uzgaruvchi toifalar ixtiyoriy uzunlikda bo'ladi. Bunda uzunliklar zarur bo'lmagan parametrlar hisoblanadi. Agar ular ishlatilmasa, unda 255 ta simvolga joy ajratiladi.

Simvolli satrlarni belgilashni yana bir usuli mavjud.

1. **Varchar** [(uzunlik)]

Ma'lumotlarni sonli tiplari

1. **Integer**
2. **Smollint**
3. **Decimal** (aniqlik, masshtab).
4. **Numeric** (aniqlik, (masshtab)..
5. **FLOAT** (aniqlik).

Sana va vaqt toifasidagi ma'lumotlar standard qo'shimcha qilinmagan. Bular yozilishini texnik hujjatlarda ko'rish kerak.

Noaniq va o'tkazib yuborilgan ma'lumotlar. SQLda atribut qiymatlari noma'lum bo'lgan otkazib yuborilgan yoki mavjud bo'lmaganlarini NULL bilan yoziladi. NULL qiymat oddiy tushunchada qiymat hisoblanmaydi. U faqat atributni haqiqiy qiymati tushib qoldirilgan yoki noma'lumligini anglatadi. NULLni ishlatishda quyidagilarga e'tibor berish kerak:

- Agregat funksiyalar ishlatilganda birorta atributni qiymatlar to'plami bo'yicha hisoblashlar bajarilganda aniqlikni ta'minlash maqsadida NULL qiymat hisobga olinmaydi.
- shartli operatorlarda TRUE, FALSEdan tashqari UNKOWN paydo bo'lsa, natija NULL qiymatda bo'ladi.
- bu qiymatni tekshirish uchun IS NULL yoki IS NOT NULLlardan foydalaniladi.
- almashtirish funksiyalari ham argument sifatida NULL bo'lsa natija NULLga

teng buladi.

SELECT operatori.

SELECT (tanlash) operatori SQL tilining eng muhim va ko'p ishlatiladigan operatori hisoblanadi. U ma'lumotlar bazasi jadvallaridan axborotlarni tanlab olish uchun mo'ljallangan.

SELECT operatori sodda holda quyidagi ko'rinishda yoziladi.

SELECT [DISTINCT] <atributlar ro'yxati>

FROM <jadvallar ro'yxati>

[WHERE <tanlash sharti>]

[ORDER BY < atributlar ro'yxati **>]**

[GROUP BY < atributlar ro'yxati **>]**

[HAVING <shart>]

[UNION <ON **SELECT** operatorli ifoda> **];**

Bu yerda kvadrat kavslarda yozilgan elementlar har doim ham yozilishi shart emas. **SELECT** xizmatchi so'zi ma'lumot bazasidan axborotni tanlab olish operatori yozilganini anglatadi. **SELECT** so'zidan keyin bir biridan vergul bilan ajratilib ko'1ksatilgan maydon nomlari (atributlar ro'yxati) yoziladi. **SELECT** so'rov operatorini zarur xizmatchi so'zi **FROM** hisoblanadi. **FROM** so'zidan keyin axborot olinayotgan jadval nomlari bir biridan vergul bilan ajratilib yoziladi. Masalan:

SELECT Name, Surname **FROM** STUDENT;

Ixtiyoriy SQL so'rov operatori nuqtali vergul(;) simvoli bilan tugaydi.

Keltirilgan so'rov STUDENT jadvalidan Name va Surname maydonlarni barcha qiymatlarini ajratib olishni amalga oshiradi. Natijada quyidagi jadval hosil bo'ladi.

8.1-j adval. Student j advali

Name	**Surname**
Ali	Soliyev
Qosim	Telmanov

Temir	Yo'ldashev
Qalandar	Qosimov
Tal'at	Temirov

Nazorat savollari

1. DDL komandalarini ayting.
2. DML vazifasi va uning ishlatilishini ko'rsating.
3. SQLda toifa tushunchasi nima?
4. SQLda toifalar nima maqsadda qo'llaniladi?
5. SQL tilining ikkita asosiy komponentasi va ularning funksiyalarini ko'rsating.
6. **SELECT** operatori asosiy konstruksiyalari va ularga qo'yiladigan cheklanishlarni ayting.

9-mavzu. Ma'lumotlar manipulyatsiya qilishda oddiy so'rovlar yaratish.

Reja

1. Murakkab so'rovlar yaratish.
2. SQLda almashtirish funksiyalari bilan ishlash.
3. Guruhli funksiyalarni so'rovlarda ishlatish.
4. Tasavurlar (View) yaratish.
5. Jadvallar bilan ishlash.
6. Ma'lumotlarni ajratish va tiklash

Tayanch iboralar: Guruhli funksiyalar, so'rovlar, manipulyatsiya, view, tasavur, tranzatsiya.

Ko'pgina amaliy masalarni yechishda ma'lum shartlar asosida axborotlarni ajratib olish talab etiladi. Masalan: "student" jadvalidan "Petrov" familiyali talabalarni chiqarish kerak.

Select Surname, Name from STUDENT Where Surname='Petrov';

Surname	Name
Petrov	Petr
Petrov	Anton

Where shartida solishtirish amallari jumladan =>, <, <=, >=, **Φ** shuningdek mantiqiy amallar "and" "or", "not" amallari ishlashi mumkin. Ular yordamida murakkab shartlar tuzililadi.

Masalan: 3-kurs stipendiya oladigan talabalarni ismi familiyasini chiqaring.

Select Name, Surname From student Where kurs=3 and stipend> 0;

Mantiqiy shartlarni berishda where parametri tarkibida **IN, BETWEEN, Like, is null** amallari ham ishlatiladi.

In yoki **not** parametli ishlatilganda tekshirilayotgan maydon qiymati berilgan ro'yxat bilan solishtiriladi. Bu ro'yxat **in** operatori o'ng tomonidan () ichida yoziladi.

Exam baholari jadvaldan «4» va «5» baholi talabalar ro'yxatini chiqaring.

Select * From exam-marks Where mark in (4,5);

Birorta ham 4 yoki 5 olmagan talaba haqidagi ma'lumotlar olish uchun **not** yoziladi.

Berween amali maydon qiymatini berilgan intervalga kirganligini tekshirish uchun ishlatiladi.

30 va 40 soat doirasida o'qitiladigan fanlarni chiqarish so'rovini berish.

Select * from subject Where hour between 30 and 40;

Between amali maydonlar sonli va simvolli bo'lganda ham ishlatiladi.

Like amali simvolli toifadagi maydonlar uchun ishlatiladi. Bu amal maydonni satrli matnlarni **like**dan so'ng ko'rsatilgan qism satr bilan solishtiradi.

Studentlar jadvalida familiyasi "M" harfi bilan boshlanadigan talaba haqida

ma'lumot chiqaring.

Select* from student Where surname like M%;

Foiz (%) belgisi shu pozitsiyada ixtiyoriy simvollar ketma-ketligi kelishini anglatadi. Bundan tashqari "__" ma'nosi shu pozitsiyada ixtiyoriy 1 ta simvol kelishini anglatadi.

Bu amallarni maydonda o'tkazib yuborilgan qiymatlarni yoki noaniq qiymatlarni topishda ishlatib bo'lmaydi.

SQL tilida ma'lumotlarni almashtiruchi va kiritilgan familiyalar ishlatilishi mumkin. Ular ustun qiymatlari bilan ishlashi uchun yoki **const** sifati ifodalarda keladi. const sifatida simvolli const, sonli constlarni ishlatish mumkin. Ular ustunlar ro'yxatiga kiritiladi va xuddi virtual ustun kabi aniqlanadi. Agar so'rovda ustun o'rnida son kelsa, bu sonli const hisoblanadi. Simvolli const () ichida yoziladi.

Misol: quyidagi so'rov ushbu jadvalni chiqaradi.

Select 'familiya', surname, 'imya', name, 100 From STUDENT

'familiya'	Surname	'imya'	name	100
familiya	Ivanov	imya	Ivan	100
familiya	Petrov	imya	Petr	100

Sonli ma'lumotlarni uzunlik ozgartirish uchun atribut amallardan foydalanamiz. Bunda quyidagi amalla ishlatiladi: "+", "*", "/"

Select surname, name, stipend, kurs, (stipend*kurs)/2 From student

Where kurs ning 4 and stipend >0

Natija:

surname	**name**	**stipend**	**kurs**	**(stipend*kurs)/2**
Sidorov	Vadim	150	4	-300
Petrov	Anton	200	4	-400

Satrlarni ulash amali yordami const 2 ta va undan ko'p simvolli ustun qiymatlari bitta satrga joylashtirib boriladi.

Select surname //'-'// name, stipend Weher kurs ning and stipend >0;

surname //'-'// name	**Stipend**
Sidirov_ Vadim	150
Petrov Anton	200

SQLda almashtirish funksiyalari bilan ishlash

Lower (<satr>)- berilgan satrni kichik harflarga almashtirib beradi.

Upper- (<satr>) - kichik harflarni kata harflarga almashtirib beradi.

Init cap- (<satr>)- satrdagi har bir suzunlikni birinchi harfini bosh harf qilib beradi. Maslan, ularga quyidagi misolni kuramiz.

Select Lower(surname) Upper(name) from student Where kurs=4;

Lower(surname)	**Upper(name)**
sidirov	VADIM
petrov	ANTON

LPAD (<satr>, uzunlikunlik, [< kism satr>) RPAD (<satr>, <uzunlikunlik>, [<kism satr>]).

Berilgan uzunlikunlikdagi qism satrni chapdan unga joylashtiriladi.

Agar kism satr kursatilmagan bulsa, satr sukut bilan, probellar bilan tuldiriladi. Agar uzunlikunlik satr uzunlikunlikdan kerak bulsa berilgan satr kursatilgan uzunlikunlikgacha kirkiladi.

LTRIM (<satr>), [<qism satr>]);

Bu funksiyalarni vazifasi mos ravishda chapdagi chegaraviy simvol olib tashlashdan iborat. Olib tashlangan simollar qism satrda ko'rsatiladi. Agar qism satr ishlamasa probellari olib tashlanadi.

Substr (<satr>, <boshlanish>, [<soni>]

Ko'yidagi bu funksiyalari satrdan berilgan pozitsiyadan boshlab berilgan sondagi simvollari ajratib olinadi. Agar soni ko'rsatilmagan bo'lsa satrni boshidan oxirigacha ajratib olinadi.

Misol: **substr** (Hurmatli do'stim: 10,7)=> do'stim

Length (<satr>) vazifasi satrni uzunligini aniqlab borishdan iborat

Select Lpad(Surname, 10, $), RPad(Name, 10,@) from STUDENT
Where kurs=3 and stipend>0

Lpad(Surname, 10, $)	RPad(Name, 10,@)
$$$$Petrov	Petr@@@@@@
$$$$Pavlov	Andrey@@@@
$$$$$Lukin	Artem@@@@@

Select substr(name, 1,1) //''// Surname, City, length(City) from STUDENT
Where krus in (2,3,4) and stipend>0;

substr(name, 1,1) //''// Surname	City	length(City)
A.Petrov	Kursk	5
S.Sidorov	Moskva	6

Select Surname, Name, Brithday, Tochar(birthday, DD.MM.YY) From STUDENT

Surname	Name	Birthday	Tochar(birthday, DD.MM.YY
Ivanov	Ivan	3/11/992	3.12.92

Guruhli (agregat) funksiyalar

Guruhli funksiyalar jadvaldan yig'ilgan axborotlarni olish uchun xizmat qiladi. Bu funksiyalar jadvaldagi satrlar guruhi bilan amal bajarib, 1 ta natija chiqaradi. Guruhli funksiyalar uchun quyidagi amallarni ishlatiladi.

1. **COUNT** - jadvaldagi satrlar sonini aniqlab beradi.
2. **SUM** - ko'rsatilgan maydon qiymatlarini yig'indisini hisoblaydi.
3. **AVG** - tanlab olingan maydon qiymatlarini o'rta arifmetigini hisoblaydi.
4. **MAX** yoki **MIN** - tanlab olingan maydon qiymatlarini eng kattasini yoki kichigini topib beradi.

Select so'rovida guruhli funksiyalar maydon nomlari kabi ishlatiladi. Maydon nomlari funksiyalar argumentlari sifatida keladi.

Misol uchun jadvaldagi satrlar sonini aniqlash uchun quyidagi so'rovdan foydalanamiz.

Select count (*) From EXAMSMARKS;

Select komandasida group by parametrini ham ishlatish mumkin. Bu paramet bir maydon o'xshash parametrlari (aniqlanayotgan qiymati) bo'yicha guruhlaydi va agregat funksiyalar ishlatilsa ular shu guruhda bo'ladi.

Select student_ID, Max (mark) from exam_marks GROUP BY student_ID;

Guruhlashni bir nechta maydon bo'yicha ham bajarish mumkin.

Select student_ID, subject_ID, Max (mark) From exam-marks
GROUP BY Student_ID, subject_ID;

Guruhlar ichidan kerakli yozuvlarni ajratib olish uchun having ishlatiladi.

Select Subj_name, max (hour) From SUBJECT
Group by Subj_name
Having max (Hour)>= 34;

Ba'zi hollarda natija jadvalidagi ma'lumotlarni tartiblash talab etiladi. Buning uchun **Order by** parametri ishlatiladi. Bu parameter ko'rsatilgan maydon barcha yozuvlarni o'sib borishi tartibida tartiblab beradi. **Order by desc** yozilsa kamayishi tartibida yoziladi. **Order by ASC** bo'lsa usish tartibida yoziladi.

1. **Select * from Subject Order by Subj_name;**
2. **Select*from Subject Order by desc Subj-name;**

Tartiblash bir nechta maydon bo'yicha bajarilishi ham mumkin. Bunda avval tartiblash 1-maydon bo'yicha keyin 2-maydon bo'yicha bajariladi.

Shuningdek **order by** parametri **group by** parametri bilan birga ishlatilishi mumkin. Bunda **group by** so'rovda oxiri keladi va unda guruhni ichidagi yozuvlarni tartiblaydi.

Select * from SUBJECT Order by Semester Group by Subj-name;

SQL tili bitta so'rov ichiga ikkinchi so'rovni joylashtirib ishlatish imkonini beradi. Misol uchun birorta talabani familiyasi bo'yicha uning **id**sini topish talab etilsa va bu talabani barcha baholari hakidagi ma'lumotni ko'rmoqchi bo'lsak quyidagi so'rovni yozish mumkin.

Select * from exam_marks Where student_ID=(select student_I From student where surname= 'Petrov')

Jadvallar bilan ishlaganda ba'zan ustun va jadval nomlarini qayta aniqlashga yoki qayta nomlashga to'g'ri keladi. Bunday masalalar ko'pincha birorta ifodalarni hisoblaganda, virtual ustunga joylashganda unga nom qo'yish yoki ba'zan natija jadvali ustunini nomlashda kerak bo'ladi.

Select name as Name_ talaba, 2* stipend AS yangi St

Name_ talaba	**yangi St**
Ivanov	150
Petrov	200

Xuddi shuningdek biz ustun nomlarini ham o'zgartirishimiz mumkin.

EXITS operatori

SQLda ishlatiladigan EXITS operatori mantiqiy ifoda kabi rost va yolg'on qiymatlar qabul qiladi. Bu operator argument sifatida qism so'rovlarini ishlatadi. Agar qism so'rov birorta qiymati rost, aks holda yolg'on bo'lishi mumkin. Misol uchun imtixonlar jadvalidan hech bo'lmaganda talaba haqidagi ma'lumotni olgan talaba haqidagi ma'lumotni chiqarish uchun quyidagi so'rov yoziladi.

Select distinct student_ID From exam_marks A where Exits (select * from EXAM-MARKS B) where mark<3 and B Student_ID= A student_ID;

Ma'lumotlar bazasi jadvallardan tashkil topadi. Jadvallar alohida fayl ko'rinishida yoki birorta faylni bo'lagi bo'lishi mumkin.

Ma'lumki **Select** operatori yordamida virtual jadvallar yaratish, Ya'ni vaqtinchalik jadvallar yaratish mumkin. Bunday jadvallar vaqtinchalik bo'lib, yaratgan foydalanuvchi o'zi undan foydalanishi mumkin.

Tasavurlar ham vaqtinchalik jadvallar bo'lib, ular ko'p foydalanuvchilar murojat qilishi mumkin va u ma'lumot bazasidan majburan olib tashlanguncha mavjud bo'ladi.

Tasavurlar MB oddiy jadvallariga o'xshash bo'lib, ma'lumotlar saqlovchi fizik ob'ekt hisoblanmaydi. Tasavurlarda ma'lumotlar jadvallardan tanlab olinadi. Tasavurlar foydalanuvchilardan jadvallarni ba'zi ustunlarini yashirish uchun yoki ko'pincha foydalanuvchiga kerakli bo'lgan bir nechta jadvaldan bitta yaratish kerak bo'ladi. Misol sifatida 3 ta jadvaldan tashkil topgan oddiy ma'lumot bazasini qarab chiqildi.

Tovarlar (ID -tovar , nomi, narxi, tavsifi)

Mijozlar(ID - mijoz, ismi, manzili, telefon)

Sotish(ID- tovar,soni, mijoz)

Tashkil qilish nuqtai nazaridan bu ma'lumot bazasi yomon loyihalanmagan. Lekin ba'zi masalarni yechishda foydalanuvchini mijoz va tovar identifikatorlari qiziqtirmaydi. Aniqroq aytganada unga bitta jadval kerak bo'ladi. Masalan bu jadval "sotishtaxlili (tovar, soni,bag'osi, narxi, mijoz)". Bu jadvalni berilgan

uchta jadvaldan quyidagi so'rov yordamida hosil qilish mumkin.

SELECT Tovarlar.Nomi AS Tovar, Sotish.Soni*Tovarlar.Bahosi AS Narxi, Mijoz.Ismi || '.Manzil: ' || Mijoz.Manzil || '. tel. ' || Mijoz.Telefon AS Mijoz FROM Sotish, Tovarlar, Mijozlar WHERE Sotish.ID_mijoz= Mijozlar.ID_mijoz AND Sotish.ID_tovar = Tovarlar. ID_tovar;

Ko'rib chiqilgan so'rov uchta jadvalni birlashtirishidan iborat bo'lib, ularga narx va mijoz ustunlari qo'shilgandir. Agar bu jadval SELECT operatorini natijasi emas, tasavur bo'lganda edi, unga oddiy ma'lumot bazasini oddiy jadvali kabi murojat qilinar edi. Ko'p hollarda esa MB uchta jadvaldan iborat ekanligini hisobga olmay, bitta tasavur bilan ishlanar edi.

Tasavurlar yaratish uchun CREATE VIEW komandasi ishlatiladi

Uni sintaksisi kuyidagicha:

CREATE VIEW «tasavur nomi» AS « select so'rovi» ;

Tasavurlarga ham ma'lumot baza jadvallari kabi nom beriladi. Bu nom birorta ham jadval nomi bilan bir xil bo'lmasligi kerak. AS so'zidan keyin ma'lumotlar tanlashga uchun so'rov iborasi yoziladi.

CREATE VIEW sotish_taxlili AS SELECT Tovarlar.Nomi AS Tovar, Sotish.Soni*Tovarlar.Bahosi AS Narxi, Mijoz.Ismi || '.Manzil: ' || Mijoz.Manzil || tel. ' || Mijoz.Telefon AS Mijoz FROM Sotish, Tovarlar, Mijozlar WHERE Sotish.ID_mijoz= Mijozlar.ID_mijoz AND Sotish.ID_tovar = Tovarlar. ID_tovar;

Natijada sotish taxlili nomli virtual jadval yaratiladi. Unga so'rovlar yordamida murojat qilish mumkin:

Select * from sotish_taxlili where tovar = 'moloko';

Jadvallarni umumlashtirish.

Jamlashtirish relyasion ma'lumotlar bazasi operatsmyalaridan biri bo'lib, jadvallar orasidagi aloqani belgilaydi va ulardan ma'lumotni bitta komanda yordamida ajratishga imkon beradi. Xar xil jadvallarda bir xil nomli ustunlar bo'lishi mumkin bo'lgani uchun, kerakli ustun uchun jadval nomii prefiksi ishlatiladi. Jamlashda jadvallar FROM ifodasidan so'ng ro'yxat sifatida tasvirlanadi. So'rov predikati ixtiyoriy jadval ixtiyoriy ustuniga tegishli bo'lishi mumkin. Jamlash eng soddasi bu dekart ko'paytmasi, uni quyidagicha bajarish mumkin:

SELECT Customers.*, Salepeople.* FROM Salepeople, Customers;*

Lekin bu yerda xosil bo'lgan jadval keraksiz ma'lumotlarga ega. Keraksiz satrlarni olibtashlash uchun WHERE jumlasidan foydalaniladi.

Masalan: beritlgan shaxardagi sotuvchilar va buyurtmachilar ixtiyoriy kombinatsiyasini ko'rish uchun quyidagini kiritish lozim:

SELECT Customers.CName, Salepeople.SName, Salepeople.City FROM Salepeople, Customers WHERE Salepeople.City = Customers.City;

Jamlashda SQL bir necha jadval satrlari kombinatsiyasini predikatlar bmyicha solishtirishdir. Asosoan ma'lumotlar ilovali yaxlitlik asosida tekshirilib, ajratib olinadi.

Misol: xar bir sotuvchiga mos keluvchi buyurtmachilar ro'yxati:

SELECT Customers.CName, Salepeople.SName FROM Customers, Salepeople WHERE Salepeople.SNum=Customers.SNum;

Tenglikka asoslangan predikatlardan foydalanuvchi jamlanmalar, tenglik bo'yicha jamlanma deb atalib, jamlanmalarning eng umuiy ko'rinishidir. Shu Bilan birga ixtiyoriy relyasion operatordan foydalanish mumkin.

Ichki va tashqi jamlashlar

Jamlashlar bir jadval satriga ikkinchi jadval satrlarini mos qo'yishga imkon beradi. Jamlashlar asosiy turi bu ichki jamlashdir. Jadvallarni ichki jamlash ikki jadval usutunlarini tenglashtirishga asoslangandir:

SELECT *book, title, author, name* FROM *author, book* WHERE *book, author = author, id*

MySQL jamlashning kuchliroq tipi Ya'ni chap tashqi jamlash(yoki tashqi jamlash) dan foydalanishga imkon beradi. Jamlash bu turitning ichki jamlashdan farqi shundaki natijaga o'ng jadvalda mos ustunga ega bo'lmagan chap jadval ustunlari qo'shiladi. Agar avtorlar va kitoblar misoliga e'tibor bersangiz atijaga ma'lumotlar bazasida kitoblarga ega bo'lmagan kitoblar kirmagan edi.

Ko'p xollarda o'ng jadvalda mosi bo'lmagan chap jadvaldagi satrlarni chiqarish kerak bo'ladi. Buni tashqi jamlash yordamida amalga oshirish mumkin:

SELECT book.title, author.name FROM author LEFT JOIN book ON book.author = author.id

E'tibor bering tashbii jamlanmada WHERE o'rniga ON kalit so'zi ishlatiladi.

MySQL tabiiy tashqi jamlashdan *(natural outer* y'oznjfoydalanishga imkon beradi. Tabiiy tashqi jamlash ikki jadval ikki ustuni bir xil nom va bir xil tiga ega bo'lgan hamda shu ustundagi qiymatlar teng bo'lgan satrlarni birlashtirishga imkon beradi:

SELECT my_prod.name FROM my_prod NATURAL LEFT JOIN their_prod

Jadvallarni o 'zi bilan jamlash.

Jadvallarni o'zi bilan jamlash uchun xar bir satrning o'zi yoki boshqa satrlar bilan kombinatsiyasini xosil qilishingiz mumkin. So'ngra xar bir satr predikat yordamida baxolanadi. Bu turdagi jamlash boshqa turdagi jamlashdan farq

qilmaydi, farqi ikki jadval bir xildir. Jadvallarni jamlashda qaytariluvchi ustun noilari oldiga jadval nomi qo'yiladi. Bu usutunlarga so'rovlarda murojaat qilish uchun xar xil nrmlarga ega bo'lishi kerak. Buning uchun vaqtinchalik nomlar Ya'ni psevdonimlar qo'llandi. Ular so'rov FROM jumlasida jadval nomidan so'ng bo'shlik qo'yib yoziladi.

Misol: bir xil reytingga ega xamma buyurtmachilar juftlarini topish.

SELECT a.CName, b.CName, a.Rating FROM Customers a, customers b WHERE a.Rating = b.Rating;

Bu holda SQL a va b jadvallarni jamlagandek ish tutadi. Yuqorida keltirilgan misolda ortiqcha satrlar mavjud, xar bir kombinatsiya uchun ikkita qiymat. Birinchi psevdonimdagi A qiymat ikkinchi psevdonimdagi B qiymat bilan kombinatsiyasi olinadi, so'ngra ikkinchi psevdonimdagi A qiymat birinchi psevdonimdagi B qiymat bilan kombinatsiyasi olinadi.

Xar gal satr o'zi bilan solishtiriladi. Buni oldini olish soda usuli ikki qiymatga cheklanish kiritish, toki birinchi qiymat ikkinchisidan kichik bo'lsin yoki alfavit bo'yicha oldin kelsin. Bu predikatni aasimmetrik qiladi, natijada xudi shu qiymatlar teskari tartibda olinmaydi.

Misol:

SELECT a.CName, b.CName, a.Rating FROM Customers a, customers b WHERE a.Rating = b.Rating AND a.CName < b.CName;

Bu misolda agar birinchi kombinatsiya ikkinchi shartni qanoatlantirsa u chiqariladi, lekin teskari kombinatsiya bu shartni qanoatlantirmaydi va aksincha. Siz SELECT ifodasida yoki so'rovning FROM jumlasida keltirilgan xar bir psevdonim yoki jadvalni ishlatishingiz shart emas. Siz xar xil jadvallar, hamda bitta jadval xar psevdonimlaridan iborat jmlanma yaratishingiz mumkin.

Sodda joylashtirilgan ostki so'rovlar. SQL yordamitda so'rovlarni bir birining ichiga joylashtirishingiz mumkin. Odatda ichki so'rov qiymat xosil qiladi

va bu qiymat tashqi predikat tomonidan tekshirilib, to'g'ri yoki noto'g'riligi tekshiriladi.

Misol: bizga sotuvchi nomi ma'lum: Motika, lekin biz SNum maydoni qiymatini bilmaymiz va Buyurtmachilar jadvalidan xamma buyurtmalarni ajratib olmoqchimiz. Buni quyidagicha amalga oshirish mumkin:

SELECT * FROM Orders WHERE SNum = (SELECT SNum FROM Salepeople WHERE SName = 'Motika');

Avval ichki so'rov bajariladi, so'ngra uning natijasi tashqi so'rovni xosil qiligsh uchun ishlatiladi (SNum ostki so'rov natijasi bilan solishtiriladi).

Ostki so'rov bitta ustun tanlashi lozim, bu ustun qiymatlari tipi predikatda solishtiriladigan qiymat tipi bilan bir xil bo'lishi kerak. Siz ba'zi xollarda ostki so'rov bitta qiymat xosil qilishi uchun DISTINCT operatoridan foydalanish mumkin.

Misol: Hoffman (CNum=21) ga xizmat ko'rsatuvchi sotuvchilar xamma buyurtmalarini topish lozim bo'lsin.

SELECT * FROM Orders WHERE SNum = (SELECT DISTINCT SNum FROM Orders WHERE CNum = 21);

Bu holda ostki so'rov faqat bitta 11 qiymat chiqaradi, lekin umumiy xolda bir necha qiymatlar bo'lishi mumkin va ular ichidan DISTINCT faqat bittasini tanlaydi. Ixtiyoriy sondagi satrlar uchun avtomatik ravishda bitta qiymat xosil qiluvchi funksiya turi - agregat funksiya bo'lib, undan ostki so'rovda foydalanish mumkin.

Masalan, siz summasi 4 oktyabrdagi bajarilishi lozim bo'lgan buyurtmalar summasi o'rta qiymatidan yuqori bo'lgan xamma buyurtmalarni ko'rmoqchisiz:

SELECT * FROM Orders WHERE AMT >
(SELECT AVG (AMT) FROM Orders WHERE ODate = '1990/10/04');

Shuni nazarda tutish kerakki guruxlangan agregat funksiyalar GROUP BY ifodasi terminlarida aniqlangan agregat funksiyalar bo'lsa ko'p qiymatlar xosil qilishi mumkin.

Agar ostki so'rov IN operatoridan foydalanilsa, ixtiyoriy sondagi satrlar xosil qilish mumkin.

Misol: Londondagi sotuvchilar uchun xamma buyurtmalarni ko'rsatish.

SELECT * FROM Orders WHERE SNum IN (SELECT SNum FROM Salepeople WHERE City = 'London');

Bu natijani jamlanma orqali xosil qilish mumkin. Lekin odatda ostki snrrovli sщrovlar tezrobi bajariladi. Siz ostki so'rov SELECT jumlasida ustunga asoslangan ifodadan foydalanishingiz mumkin. Bu relyasion operatorlar yordamida yoki IN yordamida amalga oshirilishi mumkin. Siz ostki so'rovlarni HAVING ichida ishlatishingiz mumkin. Bu ostki so'rovlar agar ko'p qiymatlar qaytarmasa xususiy agregat funksiyalaridan yoki GROUP BY yoki HAVING operatorlaridan foydalanishi mumkin.

Misol:

SELECT Rating, COUNT (DISTINCT CNum) FROM Customers GROUP BY Rating HAVING Rating > (SELECT AVG (Rating) FROM Customers WHERE City = 'San Jose');

Bu komanda San Jose dagi baxolari o'rtachadan yuqori bo'lgan buyurtmachilarni aniqlaydi.

Korrellangan (mutanosib) joylashtirilgan ostki so 'rovlar.

SQL tilida ostki so'rovlardan foydalanilganda tashqi so'rov FROM qismidagi ichki so'rovga mutanosib so'rov yordamida murojaat qilishingiz mumkin. Bu xolda ostki so'rov asosiy so'rov xar bir satri uchun bir martadan bajariladi.

Misol: 3 oktyabrda buyurtma bergan xamma buyurtmachilarni toping.

SELECT * FROM Customers a WHERE '1990/10/03' IN (SELECT ODate FROM Orders b WHERE a.CNum = b.CNum);

Bu misolda tashqi so'rovning Cnum maydoni o'zgargani uchun ichki so'rov tashqi so'rovning xar bir satri uchun bajarilishi kerak. Ichki so'rov bajarilishini talab qiladigan tashqi so'rov satri joriy satr -kandidat deyiladi. Mutanosib ostki so'rov bilan bajariladigan baxolash protsedurasi quyidagicha:

1. Tashqi so'rovda nomlangan jadvaldan satrni tanlash. Bu kelajak satr - kandidat.

2. Tashqi so'rov FROM jumlasida nomlangan psevdonimda bu satr - kandidat qiymatlarini saqlab qo'yish.

3. Ostki so'rovni bajarish. Tashqi so'rov uchun berilgan psevdonim topilgan xamma joyda joriy satr-kandidat qiymatidan foydalanish. Tashqi so'rov satr-kandidatlari qiymatlaridan foydalanish, tashqi ilova deyiladi.

4. Tashqi so'rov predikatini 3 qadamda bajariluvchi ostki so'rov natijalari asosida baxolash. U chiqarish uchun satr-kandidat tanlanishini belgilaydi.

5. Jadval keyingi satr-kandidatlari uchun protsedurani qaytarish va shu tarzda toki xamma jadval satrlari teshirilib bo'lmaguncha.

Yuqoridagi misolda SQL quyidagi protsedurani amalga oshiradi:

1. U buyurtmachilar jadvalidan Hoffman satrini tanlaydi.
2. Bu satrni joriy satr-kandidat sifatida a - psevdonim bilan saqlaydi.
3. So'ngra ostki so'rovni bajaradi. Ostki so'rov CNum maydonning qiymati a.Cnum qiymatiga teng satrlarni topish uchun Buyurtmachilar jadvali xamma satrlarini ko'rib chiqadi. Xozir a.CNum qiymati 21 ga Ya'ni Hoffman satrining CNum maydoni qiymatiga teng. Shundan so'ng shu satrlarning ODate maydonlari qiymatlari to'plamini xosil qiladi.
4. Shundan so'ng asosiy so'rov predikatida 3 oktyabrdagi qiymat shu to'plamga tegishliligini tekshiradi. Agar bu rost bo'lsa Hoffman satrini chiqarish uchun tanlaydi.

5. Shundan so'ng u butun protsedurani Giovanni satrini satr -kandidat sifatida foydalanib qaytaradi va saqlab qo'yadi, toki Buyurtmachilar xamma satri tekshirilib bo'lmaguncha.

6. Ba'zida xatolarni topish uchun maxsus yaratilgan so'rovlardan foydalanish kerak bo'ladi.

7. Misol: Quyidagi so'rov Buyurtmachilar jadvalini ko'rib chiqib SNum va CNum mos kelishini tekshiradi va mos bo'lmagan satrlarni chiqaradi.

SELECT * FROM Orders main WHERE NOT SNum =(SELECT SNum FROM Customers WHERE CNum = main.CNum);

Asosiy so'rov asoslangan jadvalga asoslanuvchi mutanosib so'rovdan foydalanishingiz mumkin.

Misol: sotib olishlar buyurtmachilari uchun o'rta qiymatdan yuqori bo'lgan hamma buyurtmalarni topish.

SELECT * FROM Orders a WHERE AMT >(SELECT AVG (AMT) FROM Orders b WHERE b.CNum = a.CNum);

HAVING operatoridan ostki so'rovlarda foydalanilganidek mutanosib ostki so'rovlarda ham foydalanigsh mumkin.

HAVING ifodasida mutanosib ostki so'rovdan foydalanganda HAVING o'zida ishlatilishi mumkin bo'lgan pozitsiyalarga tashqi ilovalarni cheklab qo'yishingiz kerak. Chunki HAVING ifodasida faqat agregat SELECT ifodasida ko'rsatilgan funksiyalardan yoki GROUP BY ifodasida ko'rsatilgan maydonlardan foydalanish mumkin. Ulardan siz tashqi ilova sifatida foydalanishingiz mumkin. Buning sababi shuki HAVING tashqi so'rovdagi satrlar uchun emas guruxlar uchun baxolanadi. Shuning uchun ostki so'rov bir marta satr uchun emas gurux uchun bajariladi.

Misol: Buyurtmalar jadvalidagi sotibolishlar summalarini sanalar bo'yicha guruxlar summasini hisoblash kerak bo'lsin. Shu bilan birga summa maksimal

summadan kamida 2000.00 ga ko'p bo'lmagan sanalarni chiqarib tashlash kerak bo'lsin:

SELECT ODate, SUM (AMT) FROM Orders a GROUP BY ODate HAVING SUM (AMT) > (SELECT 2000.00 + MAX (AMT) FROM Orders b WHERE a.ODate = b.ODate);

Ostki so'rov asosiy so'rovning ko'rilayotgan agregat guruxi sanasiga teng sanaga ega xmma satrlar uchun MAX qiymat xisoblaydi. Bu WHERE ifodasidan foydalanib bajarilishi lozim. Ostki so'rovning o'zi GROUP BY yoki HAVING operatorlarini ishlatmasligi kerak.

EXISTS operatoridan foydalanish.

EXISTS - bu "TRUE" yoki "FALSE" qaytaruvchi operatordir. Bu shuni bildiradiki, u predikatda avtonom yoki mantiqiy operatorlar AND, OR, va NOT yordamida tuzilgan mantiqiy ifodalar bilan kombinatsiya qilingan xolda ishlatilishi mumkin. U ostki so'rovni "TRUE" deb baxolaydi agar u ixtiyoriy natija xosil qilsa va "FALSE" deb baxolaydi xech qanday natija xosil qilmasa.

Misol: Agar buyurtmachilardan ju da bo'lmasa bittasi San Jose shaxrida yashasa, buyurtmachilar jadvalidagi ma'lumotlarni chiqaring.

SELECT CNum, CName, City FROM Customers
WHERE EXISTS
(SELECT * FROM Customers
WHERE City = 'San Jose');

EXISTS ni faqat sodda ostki so'rov bilan emas mutanosib so'rov bilan ishlatish mumkin. Bu holda EXISTS ichki ostki so'rovni tashqining xar bir satri uchun tekshiradi.

ALL, ANY, SOMEoperatorlaridan foydalanish.

ANY, ALL, va SOME ostki so'rovlarni argument sifatida qabul qiluvchi EXISTS operatorni eslatadi, lekin relyasion operatorlar bilan birga ishlatilishi bilan

farq qiladi. Bu tomondan ular ostki so'rovlarga qo'llaniluvchi IN operatorini eslatadi, lekin undan farqli faqat ostki so'rovlar bilan ishlashadi. SOME va ANY operatorlari o'zaro almashinuvchan.

Misol: bir shaxarda joylashgan sotuvchilar bilan buyurtmachilarni topish uchun ANY operatoridan foydalanish.

```
SELECT * FROM Salepeople
WHERE City = ANY (SELECT City FROM Customers);
```

Operator ANY ostkiso'rov chiqargan xamma qiymatlarni oladi, (bu misol uchun - Buyurtmachilar jadvalidagi xamma City qiymatlari), i va rost deb baxolaydi agar ularning ixtiyoriysi (ANY) tashqi so'rov satridagi shaxar qiymatiga tengbo'lsa. ANY operatori o'rniga IN yoki EXISTS ishlatish mumkin, lekin ANY "= " operatordan boshqa relyasionn operatorlarni ishlatishi mumkin. Misol: Xamma sotuvchilarni alfavit bo'yicha kelgan buyurtmachilari bilan birga topish.

```
SELECT * FROM Salepeople
WHERE SName < ANY (SELECT CName FROM Customers);
```

ANY to'la bir qiymatli emas. Misol: Rimdagi buyurtmachilarga ko'ra yuqori reytinga ega buyurtmachilarni topish.

```
SELECT * FROM Customers
WHERE Rating > ANY (SELECT Rating FROM Customers
WHERE City = 'Rome');
```

Ingliz tilida " ixtiyoriysidan katta (bu yerda City = Rome) " baxolash quyidagicha talqin qilinadi, bu baxolash qiymati xar bir City = Rome xoldagi baxolash qiymatidan katta bo'lishi kerak. SQL tilida ANY operatoridan foydalanilganda bunday emas. ANY to'g'ri deb baxolanadi agar ostki so'rov shartga mos keluvchi ixtiyoriy qiymat topsa. Yuqorida ko'rilgan misol 300 va 200

baxoli xamma buyurtmachilarni topadi, chunki 300>200 dlya Rimdagi Giovanni uchun va 200>100 Rimdagi Pereira uchun.

Soddaroq aytganda < ANY ifodasi eng katta tanlangan qiymatdan kichik qiymatni, > ANY - eng kichik tanlangan qiymatdan katta qiymatni bildiradi.

ALL yordamida, predikat rost xisoblanadi, ostki so'rov tanlagan xar bir qiymat tashqi so'rov predikatidagi shartga mos kelsa.

Misol: Rimdagi xar bir buyurtmachidan baxolari yuqori bo'lgan buyurtmachilarni chiqaring.

```
SELECT * FROM Customers
WHERE Rating > ALL (SELECT Rating FROM Customers
WHERE City = 'Rome');
```

Bu operator Rimdagi xamma buyurtmachilar baxolari qiymatlarini tekshiradi. Shundan so'ng Rimdagi xamma buyurtmachilardapn baxosi yuqori bo'lgan buyurtmachilarni topadi. Rimda eng yuqori baxo - Giovanni (200). Demak 200 dan yuqori qiymatlar olinadi.

ANY operatori uchun bo'lgani kabi ALL operatori uchun ham IN va EXISTS yordamida al'ternativ konstruksiyalar yaratish mumkin.

ALL asosan tengsizliklar bilan ishlatiladi, chunki qiymat "xammasi uchun teng " ostki so'rov natijasi bo'lishi mumkin agag xamma natijalar bir xil bo'lsa. SQL da < > ALL ifoda aslida ostki so'rov natijasining " xech qaysisiga teng emas " ma'noni bildiradi. Boshqacha qilib aytganda proedikat rost agar berilgan qiymat ostki so'rov natijalari orasida topilmagan bo'lsa. Agar oldingi misolda tenglik tengsizlikka almashtirilsa, reytingi 300 ga teng bo'lgan xamma buyurtmachilar chiqariladi, chunki ostki so'rov 100 va 200 ga teng reytinglarni topgan.

ALL va ANY - orasidagi asosiy farq, ostki so'rov xech qanday natija qaytarmagan xolatda ko'rinadi. Bu xolda ALL - avtomatik ("TRUE") ga teng, ANY bo'lsa avtomatik ("FALSE") ga teng.

Misol: Buyurtmachilar jadvali xammasini chiqarish

```
SELECT * FROM Customers
WHERE Rating > ALL (SELECT Rating FROM Customers
WHERE City = 'Boston');
```

Ko'rsatilgan operatorlar bilan ishlashda NULL qiymatlar ma'lum muammolarni keltirib chiqaradi. SQL predikatda solishtirayotgan qiymatlardan biri bo'sh (NULL) qiymat bo'lsa, natija noaniqdir. Noaniq predikat, noto'g'ri predikatga o'xshash, shuning uchun satr tashlab yuboriladi.

UNION ifodasidan foydalanish.

UNION ifodasi bir yoki bir necha SQL so'rovlar nitijasini birlashtirishga imkon beradi.

Misol: Londonga joylashgan xamma sotuvchilar va buyurtmachilarni bitta jadvalda chiqaring.

```
SELECT SNum, SName FROM Salepeople
WHERE City = 'London'
UNION
SELECT CNum, CName FROM Customers
WHERE City = 'London';
```

Ikki yoki undan ortiq jadvallar jamlanganda ularning chiqish ustunlari jamlash uchun o'zaro muvofiq bo'lishi kerak. Bu shuni bildiradiki, xar bir so'rov bir xil sondagi ustunlarni ko'rsatib, bu ustunlar mos tartibda kelishi va xar biriga mos tiplarga ega bo'lishi kerak. Sonli maydonlar bir xil tipga va kattalikka ega bo'lishi kerak. Simvolli maydonlar bir xil sondagi simvollarga ega bo'lishi kerak. Moslik ta'minlovchi yana bir shart bo'sh (NULL) qiymatlar jamlanma ixtiyoriy ustunida man etilgan bo'lishi kerak. Bu qiymatlar boshqa jamlovchi so'rovlarda ham man etilgan bo'lishi kerak. Bundan tashqari siz ostki so'rovlarda UNION operatoridan, hamda jamlovchi so'rov SELECT operatorida agregat funksiyalardan foydalanishingiz mumkin emas. Siz individual so'rovlardagi kabi natijani

tartiblash uchun ORDER BY operatoridan foydalanishingiz mumkin. Jamlanma ustunlari chiqarish ustunlari bo'lgani 4chun ulapr nomlarga ega bo'lmaydi, shuning uchun nomeriga qarab aniqlanishi lozim. Demak ORDER BY operatorida ustun nomeri ko'rsatilishi lozim. Foydali jamlanmalardan biri ikki so'rovnni jamlashda ikkinchi so'rov birinchi so'rov chiqarib tashlagan satrlarni tanlashidir. Bu tashqi jamlanma deyiladi.

Misol: O'z shaxarlarida buyurtmachilarga ega yoki ega emasligini ko'rsatgan xolda xamma sotuvchilarni chiqarish.

SELECT Salepeople.SNum, SName, CName, Comm FROM Salepeople, Customers

WHERE Salepeople.City = Customers.City

UNION

SELECT SNum, SName, 'NO MATCH', Comm FROM Salepeople

WHERE NOT City = ANY (SELECT City FROM Customers)

ORDER BY 2 DESC;

Xar gal bir necha so'rovlarni jamlaganda yumaloq qavslar yordamida baxolash mezonini ko'rsatishingiz mumkin. Ya'ni

query X UNION query Y UNION query Z;

o'rniga, yoki

(query X UNION query Y)UNION query Z;

Yoki

query X UNION (query Y UNION query Z);

ko'rsatishingiz mumkin. CHunki UNION bitta dublikatlarni yo'qotib boshqasini qoldirishi mumkin. Quyidagi ikki ifoda

(query X UNION ALL query Y)UNION query Z;

query X UNION ALL(query Y UNION query Z);

bir xil natija qaytarishi shart emas, agar ikkilangan satrlar unda o'chirilgan bo'lsa.

Nazorat savollari

1. SQLda almashtirish funksiyalari nima maqsadda ishlatiladi?
2. Guruhli funksiyalarga qnaday funksiyalar kiradi?
3. Tasavurlar (View)ning afzalliklari nimada?
4. Shartli so'rovlar qanday tashkil qilinadi?
5. Guruhli funksiya vazifalari?
6. Guruhli funksiya ko'rinshlari?
7. Murakkab so'rovlar yaratish qanday yaratiladi?

10-mavzu. SQLtili yordmida ma'lumotlarni tavsiflash

Reja

1. SQL tilida ma'lumotlarni butunligini ta'minlash.
2. Ma'lumot jadvallarini yaratish.
3. Qism so'rovlar bilan ishlash.
4. Ma'lumot bazasi ob'ektlarini yaratish.
5. Ma'lumotlarni aniqlash tili (DLL) operatorlari.
6. CREATE TABLE komandasi.
7. INSERT komandasi.
8. Har bir ustun uchun tip (toifa) va o'lcham.

Tayanch so^zlar: CREATE TABLE, Char, character, Int, Smallint, Dec, Number, Float.

Ma'lumotlar bazasi ob'ektlarini yaratish. Ma'lumotlar bazasi ob'ektlarini yaratish ma'lumotlarni tavsiflash tili **(DLL)** operatorlari yordamida amalga oshiriladi. Ma'lumotlar bazasi jadvallari **CREATE TABLE** komandasi yordamida amalga oshiriladi. Bu komanda bo'sh jadval yaratadi, Ya'ni jadvalda satrlar bo'lmaydi. Bu jadvalga qiymatlar **INSERT** komandasi yordamida kiritiladi. **CREATE TABLE** komandasi jadval nomini va ko'rsatilgan tartibda nomlangan ustunlar to'plamini aniqlaydi. Har bir ustun uchun tip (toifa) va

o'lcham aniqlanadi. Har bir yaratilgan jadval hech bo'lmaganda bitta ustunga ega bo'dishi kerak. **CREATE TABLE** komanda ko'rinishi quyidagicha:

CREATE TABLE <jadval nomi>(<ustun nomi><ma'lumot oifasi>[<o'lchami>]);

CREATE TABLE xususiyati quyidagicha:

SQL ishlatilayotgan ma'lumot toifalariga ANSI standarti berilgan..

- Char(character)
- Int(integer);
- Smallint,
- Dec(detcimal),
- Number,
- Float va hokazo

Albatta ko'rsatilishi zarur bo'lgan ma'lumot toifasi CHAR. Maydonga yozilgan real simvollar soni noldan (agar maydonda NULL qiymati bo'lsa) CREATE TABLEda berilgan maksimal qiymatgacha bo'ladi. Masalan STUDENT1 jadvalini quyidagi komanda bilan yaratish mumkin:

CREATE TABLE Student1 (Student_ID INTEGER, Surname VARCHAR(60), Name VARCHAR(60), Stipend DOUBLE, Kurs INTEGER, City VARCHAR(60), Birthday DATE, Univ_ID INTEGER);

Jadvaldagi ma'lumotlarni maydonlar bo'yicha qidirish tanlash amali yetarli darajada tezlatish uchun ma'lumotlarni berilgan maydon bo'yicha indeksatsiya qilish ishlatiladi. Indekslarni bitta yoki bir nechta maydon bo'yicha bajarish mumkin.

Indeks komandasini ko'rinishi:

CREATE INDEX <indeks nomi> ON <jadval nomi>(<ustun nomi>);

Bu komanda bajarilishi uchun jadval yaratilgan bo'lishi kerak va indeksda ko'rsatilgan ustunlar unda bo'lishi kerak. Masalan, agar ExamMarks jadvalidan talabani StudentID maydoni qiymati bo'yicha bahosini qidirish tez-tez talab

etilsa, unda shu maydon bo'sicha indeks bajariladi.

CREATE INDEX Student_ID_1 ON ExamMarks (Student_ID);

Indeksni olib tashlash uchun (bunda uni nomini albatta bilish kerak) quyidagi komanda ishlatiladi.

DROP INDEX <Indeks nomi>;

Masalan, **DROP INDEX** <Student_ID_1>;

Mavjud jadval tuzilmasi va parametrlari uchun **ALTER TABLE** komandasi ishlatiladi. Jadvalga ustunlar qo'shish **ALTER TABLE** komandasi orqali quyidagicha bo'ladi:

ALTER TABLE <jadval nomi>ADD(<ustun nomi> <ma'lumottipi> <o'lchami>);

Bu komanda orqali mavjud jadval satrlariga yangi ustun qo'shiladi va unga NULL qiymati yoziladi. Jadvalga bir nechta ustun ham ko'shsa bo'ladi. Ular bir biridan vergul bilan ajratiladi.

ALTER TABLE <jadval nomi> MODIFY (<ustun nomi><ma'lumot tipi> <o'lcham/aniqlik>);

Ustun xarakteristikalarini modifikatsiyalashda quyidagi cheklanishlarni hisobga olish kerak:

- ma'lumot toifasini o'zgartirishni, faqat ustun bo'sh bo'lsa bajarish mumkin;
- to'ldirilmagan ustun uchun o'lcham/aniqlik o'zgartirish mumkin;
- to'ldirilgan ustun uchun o'lcham/aniqlik faqat kattalashtirish mumkin;
- NOT NULL o'rnatilishi uchun ustunda birorta ham NULL qiymat boTmasligi kerak;
- sukut bilan o'rnatilgan qiymatni har doim o'zgartirish mumkin.

Ma'lumotlar bazasidan jadvallarni olib tashlash quyidagi komanda bilan bajariladi.

DROP TABLE < jadval nomi >;

Mumkin bo'lgan ma'lumot qiymatlar cheklanishlari bo'lishi mumkin. Unda **CREATE TABLE** komandasi quyidagicha bo'ladi.

CREATE TABLE <jadval nomi> (<ustun nomi> <ma'lumot toifasi> <ustunga cheklanishlar>, <ustun nomi> <ma'lumot toifasi> <ustunga cheklanishlar>, < jadvalga cheklanishlar> (<ustun nomi>,[< ustun nomi >]));

Student jadvalining Student_ID, Surname, Name maydonlarida **NULL** qiymat bo'lishini taqiqlash uchun so'rov quyidagicha yoziladi:

CREATE TABLE STUDENT (Student_ID INTEGER NOT NULL,
Surname CHAR(25) NOT NULL,
Name CHAR(10) NOT NULL,
Stipend INTEGER,
Kurs INTEGER,
City CHAR(15),
Bithday DATE,
Univ_ID INTEGER);

Ba'zi hollarda biror maydonga kiritilayotgan barcha qiymatlar bir biridan farq qilishi kerak. Bunda shu maydon uchun **UNIQUE (yagona)** so'z ishlatiladi. Masalan Student jadvalida StudentID qiymatlari farqli bo'lishi uchun so'rov quyidagicha yoziladi.

CREATE TABLE STUDENT (Student_ID INTEGER NOT NULL UNIQUE,
Surname CHAR (25) NOT NULL,
Name CHAR(10) NOT NULL ,
Stipend INTEGER,
Kurs INTEGER,
City CHAR(15),

Bithday DATE);

Jadvalda kalit maydonlarni ishlatish komandasi quyidagicha yoziladi:

CREATE TABLE Student (Student_ID INTEGER PRIMER KEY,
Surname CHAR (25) NOT NULL,
Name CHAR(10) NOT NULL ,
Stipend INTEGER,
Kurs INTEGER,
City CHAR(15),
Bithday DATE,
Univ_ID INTEGER);

SQL tilida jadvalga ma'lumotlar kiritish, o'zgartirish va o'chirish uchun ma'lumotlarni manipulyasiya qilish **(DML)** tilining uchta komandasi ishlatiladi. Bular **INSERT** (qo'shish), **UPDATE** (tiklash yangilash), **DELETE** (o'chirish) komandalaridir.

INSERT komandasi jadvalga yangi satr qo'shishni amalga oshiradi. Sodda holda uning sintaksisi quyidagicha:

Insert into <jadval nomi> values (<qiymat>,<qiymat>,);

Bunday yozuvda **VALUES** kalit so'zidan keyin qavs ichida ko'rsatilgan qiymatlar jadvaldagi yangi qo'shilgan satrning maydonlariga kiritiladi. Kiritish jadvalini **CREATE TABLE** operatori bilan yaratilish paytidagi ustunlarni ko'rsatish tartibida amalga oshiriladi. Masalan, STUDENT jadvalida yangi satrni qo'shish quyidagicha amalga oshirish mumkin.

Insert into Student Values (101, 'Aliyev', 'Rustam', 200, 3, 'Uzbekistan', '6/10/1979', 15);

Agar birorta maydonga NULL qiymati qo'shish zarur bo'lsa u oddiy qiymat

kabi kiritiladi.

Insert into Student Values (101, 'Aliyev', Null, 200, 3, 'Uzbekistan', '6/10/1979', 15);

Ba'zi hollarda maydonlarning qiymatini **CREATE TABLE** komandasida berilgan tartibdan boshqa tartibda kiritish zaruriyati paydo bo'lsa yoki qiymatlarni ba'zi bir ustunlarga kiritish talab etilmasa, **INSERT** komandasining quyidagi ko'rinishi ishlatiladi.

Insert into Student (Student_ID, City, Surname, Name) Values (101, 'Uzbekistan', 'Aliyev', 'Rustam');

Qavs ichidagi ro'yxatda nomi keltirilmagan ustunlarga avtomatik ravishda sukut bilan jadval tavsiflashda **(CREATE TABLE** komandasida) tayinlangan qiymat yoki **NULL** qiymat tayinlanadi.

INSERT komandasi yordamida, bir jadvaldan qiymat tanlab olib uni boshqa jadvalga joylashtirish mumkin.

Insert into Student1 SELECT * from Student where CITY='Xiva';

Bunda Student1 jadvali **CREATE TABLE** komandasi yordamida yaratilgan bo'lishi kerak va Student jadvali strukturasiga o'xshash bo'lishi kerak.

Jadvaldagi satrlarni o'chirish uchun **DELETE** komandasi ishlatiladi. Quyidagi ifoda Exam_Marks1 jadvalidan barcha satrlarni o'chiradi.

DELETE * FROM Exam_Marks1;

Buning natijasida jadval bo'sh bo'lib qoladi (bundan so'ng jadvalni **DROP TABLE** komandasi bilan o'chirish mumkin).

Jadval bir yo'la birorta shartni qanoatlantiradigan bir nechta satrni olib tashlash uchun **WHERE** parametridan foydalanish mumkin.

DELETE * FROM ExamMarks WHERE STUDENT_ID=103;

UPDATE komandasi jadval satrlari yoki mavjud satrni ba'zi bir yoki barcha maydonlari qiymatini yangilash, Ya'ni o'zgartirish imkonini beradi. Masalan Universitetl jadvalidagi barcha universitetlarni reytingini 200 qiymatga o'zgartirish uchun quyidagi so'rovni ishlatish mumkin:

UPDATE University1 SET Rating=200;

Jadvaldagi maydon qiymatlarini o'zgartirish kerak bo'lgan aniq satrlarni ko'rsatish uchun **UPDATE** komandasi **WHERE** parametrida predikat ishlatish mumkin.

UPDATE UNIVERSITY1 **SET** RATING=200 **WHERE** CITY= 'Moskva';

Bu so'rov bajarilganda faqat Moskvada joylashgan universitetlarning reytingi o'zgartiriladi.

UPDATE komandasi faqat bitta ustun emas balki ustunlar to'plamini o'zgartirish imkonini beradi. Qiymatlari modifikatsiya qilinishi zarur bo'lgan aniq ustunlarni ko'rsatish uchun, SET parametri ishlatidi. Masalan o'qitilayotgan fan nomi "Matematika" (uning uchun SUBJ_ID=43) "Oliy matematika" nomiga o'zgartirish talab etilsa va bunday indetifikatsion nomeri saqlab o'zgarish qoldirish kerak bo'lib, lekin shu bilan birga jadvaldagi mos satr maydonlariga o'qitiladigan fan haqida yangi ma'lumotlar kiritish uchun so'rov quyidagi ko'rinishda bo'ladi.

UPDATE Subject1 SET Subj_Name='Oliy matematika', Hour=36, Semester=1 WHERE SUBJ_ID=43;

UPDATE komandasini **SET** parametrida skalyar ifodalarni ishlatish mumkin. Skalyar ifodada maydon sifatida o'zgartirilayotgan va boshqa maydonlar kiritilib, u maydon qiymatini o'zgarish usulini ko'rsatadi

UPDATE University1 SET Rating=Rating*2;

Student1 jadvaldagi Stipend maydon qiymatini Moskva shahri talablari

uchun 2 marta oshirish uchun quyidagi so'rov yoziladi.

UPDATE student1 SET Stipend=Stipend*2 WHERE City= 'Moskva';

SET predikat hisoblanmaydi. Shuning uchun unda **NULL** qiymatni ko'rsatish mumkin.

UPDATE Student1 SET Stipend=NULL WHERE City= 'Moskva';

INSERTda qism so'rovlarini ishlatish.

INSERT operatorini qism so'rovi bilan ishlatish bitta jadvalga bir vaqtning o'zida bir nechta satr yuklash imkonini beradi. **VALUES** ishlatuvchi **INSERT** operatori bitta satr qo'shsa **INSERT** qism so'rov jadvalga qism so'rov boshqa jadvaldan qancha satr ajratsa shuncha satr jadvalga qo'shadi.

Bu holda qism so'rov bilan olinayotgan ustunlar soni va toifasi bo'yicha, ma'lumotlari qo'shilayotgan jadvaldagi ustun soni va toifasiga mos kelishi kerak. Misol uchun Student1 jadvalini tuzilmasi Student jadval tuzilmasiga to'la mos bo'lsin.

Student jadvalidan Moskva shahri talabalari barchasi haqida yozuvlari bilan Studentl jadvalni to'ldirish imkonini beradigan so'rov ko'rinishi quyidagicha bo'ladi.

Insert INTO Student1 SELECT * FROM Student WHERE City= 'Moskva';
Studentl jadvaliga Moskvada o'qiyotgan barcha talabalar haqidagi ma'lumotlarni qo'shish uchun **WHERE** parametrida mos qism so'rov ishlatish mumkin.

INSERT INTO Student1 SELECT * FROM Student WHERE Univ_ID IN (SELECT Univ_ID FROM University WHERE City= 'Moskva');

Nazorat savollari

1. SQL tilining jadval yaratish operatori sintaksisi qanday?
2. Indeks operatorini ko'rinishi va uning vazifasi?

3. SQL tilining jadvalga yozish va takomillashtirish komandalarini tavsiflang.
4. Insert opretori haqoda ma'lumot bering.
5. IN predikati nima vazifa bajaradi?
6. Drop va Delete operatorlarining farqi nimada?
7. Update operatori sintaksisi qanday yoziladi?

11-mavzu. SQLda jarayonlar va standart funksiyalar

Reja

1. SQL tilida agregat funksiyalar.
2. Agregat funksiyalar argumentlari.
3. Standart funksiyalar orqali so'rovlar yaratish.
4. Standart funksiyalarning SQLda sintaksisi.
5. DISTINCT standart so'zi va undan foydalanib ikki nusxadagi satrlarni o'chirish.
6. ORDER BY komandasidan foydalanib satrlarni tartiblashtirish.
7. Agregatlar va ma'lumotlarni guruhlash.

Tayanch so\lar: Agregat funksiyalar, group by, having

Agregat funksiyalar qo 'llanishi

Agregat (yoki STATIK) funksiyalar, sonli yoki xisoblanuvchi ustunlar Bilan ishlaydi. Agregat funksiya argumenti butun ustun bo'lib, bita qiymat qaytaradi. Bu funksiyalarni ko'rib chiqamiz:

- SUM() - Ustundagi xamma qiymatlar summasini xisoblaydi.
- AVG() - Ustundagi xamma qiymatlar o'rtasi qiymatini xisoblaydi.
- MIN() - Ustundagi xamma qiymatlar eng kichigini aniqlaydi.
- MAX() - Ustundagi xamma qiymatlar eng kattasini aniqlaydi.
- COUNT() - Ustundagi qiymatlar sonini xisoblaydi.
- COUNT(*) - So'rov natijalari jadvalidagi satrlar sonini xisoblaydi.

Komandalar sintaksisi ko'rinishi:

SUM (— ifoda

--DISTINCT - ustun nomi

AVG (--- ifoda

DISTINCT ustun nomi

MIN (ifoda)

MAX (ifoda)

COUNT ustun nomi)

DISTINCT

COUNT(*)

Agregatlash argumenti bo'lib ustun nomidan tashqari ixtiyoriy matematik ifoda xizmat qilishi mumkin. Misol uchun quyidagi so'rovda: Sizni kompaniyangizda reja bajarilishi o'rtacha protsenti qancha?

SELECT AVG(100 * (SALES/QUOTA)) FROM SALESREPS

Yana bir shakl: Sizni kompaniyangizda reja bajarilishi o'rtacha protsenti qancha?

SELECT AVG(100 * (SALES/QUOTA)) PROCENT FROM SALESREPS

Bu xolda ustun nomi ma'noliroq, lekin bu asosiysi emas. Ustunlar summasini xisoblab ko'ramiz. SUM() funksiyasini qo'llaymiz, buning uchun ustun int toifada bo'lishi kerak! Masalan, quyidagicha: Kompaniya xizmatchilari sotuvlar xajmi rejadagi va xaqiqiy o'rta qiymati qanchaga teng?

SELECT SUM(QUOTA), SUM(SALES) FROM SALESREPS

AVG() agregatlash funksiyasiga Yana bir necha sodda misollarni ko'ramiz.
Masalan: "ACI" ishlab chiqaruvchi mollari o'rtacha narxini xisoblang.

SELECT AVG(PRICE)

FROM PRODUCTS
WHERE MFR_ID = 'ACI'

Ekstremumlarni topish funksiyalari yani MIN(), MAX() funksiyalarini ko'ramiz.Bu funksiyalar sonli ustunlar, sanalar va satrli o'zgaruvchilar Bilan ishlaydi. Eng soda qo'llanishi sonlar bilan ishlash.

Masalan quyidagi so'rov yozamiz: Rejadagi eng ko'p va kam sotuvlar xajmi qancha?

SELECT MIN(QUOTA), MAX(QUOTA) FROM SALESREPS

Bu sonlarni o'z ichiga olgan ustunlardir. Yana bir so'rov beramiz: Bazadagi buyurtmalarning ichida eng oldin berilgan so'rov sanasi?

SELECT MIN(ORDERDATE) FROM ORDERS

Satrlar Bilan ishlaganda xar xil SQL serverlardagi kodirovkalar xar xil natija berishi mumkin. YOzuvlar sonini sanash uchun COUNT() qo'llanadi. Bu funksiya son qiymat qaytaradi. Masalan: Kompaniyamiz mijozlari soni nechta?

SELECT COUNT(CUST_NUM) FROM CUSTOMERS

Yana bir so'rov: Qancha xizmatchi rejani ortig'i bilan bajardi?

SELECT COUNT(NAME) FROM SALESREPS WHERE SALES > QUOTA

COUNT(*) funksiyasi qiymatlar sonini emas, satrlar sonini xisoblaydi. Quyidagicha yozish mumkin:

SELECT COUNT(*) FROM ORDERS WHERE AMOUNT > 250

NULL qiymat va agregat funksiyalar

Ustun qiymati NULL bo'lsa AVG(), MIN(), MAX(), SUM(), COUNT() funksiyalari qanday qiymat qaytaradi? ANSI/ISO qoidalariga ko'ra "agregat funksiyalar NULL qiymatni e'tiborga olmaydi"! Quyidagi so'rov ko'ramiz: **SELECT COUNT(*), COUNT(SALES), COUNT(QUOTA) FROM SALESREPS**

Jadval bita lekin so'rovdagi qiymatlar xar xil. CHunki QUOTA maydoni- NULL qiymatni o'z ichiga oladi. COUNT funksiyasi COUNT(maydon) ko'rinishda bo'lsa NULL qiymatni e'tiborga olmaydi, COUNT(*) bo'lsa satrlar umumiy sonini xsoblaydi. MIN(), MAX() funksiyalari xam NULL qiymatni e'tiborga olmaydi, lekin AVG(), SUM() - NULL qiymat mavjud bo'lsa chalkashtiradi. Masalan, quyidagi so'rov:

SELECT SUM(SALES), SUM(QUOTA), (SUM(SALES) - SUM(QUOTA)), (SUM(SALES - QUOTA)) FROM SALESREPS

(SUM(SALES)-SUM(QUOTA)) va (SUM(SALES-QUOTA)) ifodalari agar QUOTA, maydoni NULL qiymatga ega bo'lsa xar xil qiymat qaytaradi. Ya'ni ifoda SUM(ustun qiymati - NULL) Yana NULL qaytaradi! Shunday qilib:

1. Agar ustundagi qiymatlardan biri NULL ga teng bo'lsa, funksiya natijasini xisoblashda ular tashlab yuboriladi!
2. Agar ustundagi xamma qiymatlar NULL ga teng bo'lsa, AVG(), SUM(), MIN(), MAX() funksiyalari NULL qaytaradi! Funksiya COUNT() nolb qaytaradi!
3. Agar ustunda qiymatlar bo'lmasa (Ya'ni ustun bo'sh), AVG(), SUM(), MIN(), MAX() funksiyalari NULL qaytaradi! Funksiya COUNT()nolb qaytaradi!
4. Funksiya COUNT(*) satrlar sonini xisoblaydi va ustunda NULL qiymat bor yo'qligiga bog'liq emas! Agar ustunda satrlar bo'lmasa, bu funksiya nol qaytaradi!
5. DISTINCT funksiyasini agregat funksiyalar bilan birga ishlatish mumkin.

Masalan quyidagi so'rovlarda: Kompaniyamizda qancha xar xil raportlar nomlari mavjud?

SELECT COUNT(DISTINCT TITLE) FROM SALESREPS

DISTINCT va agregatlar ishlashda quyidagi qoidalar mavjud. Agar siz DISTINCT va agregat funksiyani ishlatsangiz uning argumenti faqat ustun nomi bo'lishi mumkin, ifoda argument bo'lolmaydi. MIN(), MAX() funksiyalarida DISTINCT ishlatish ma'nosi yo'q! COUNT() funksiyasida DISTINCT ishlatiladi, lekin kam xollarda. COUNT(*) funksiyasiga umuman DISTINCT qo'llab bo'lmaydi, chunki u satrlar sonini xisoblaydi! Bita so'rovda DISTINCT faqat bir marta qo'llanishi mumkin! Agarda u agregat funksiya argumenti sifatida qo'llanilsa,boshqa argument Bilan qo'llash mumkin emas!

Agregatlar va ma 'lumotlarni guruxlash

Agregat funksiyalar jadval uchun natijaviy satr xosil qiladi. Masalan: Buyurtma o'rtacha narxi qancha?

SELECT AVG(AMOUNT) FROM ORDERS

Masalan, oraliq natijani topish lozim bo'lsin. Bu holda guruxlanishli so'rov yordam beradi. Ya'ni SELECT operatorining GROUP BY ifodasi. Avval GROUP BY ifodasi qatnashgan quyidagi so'rovni ko'ramiz: Xar bir xizmatchi uchun buyurtma o'rtacha narxi qancha?

SELECT REP, AVG(AMOUNT) FROM ORDERS GROUP BY REP

REP maydoni bu xolda guruxlash maydonidir, Ya'ni REP maydonning xamma qiymatlari guruxlarga ajratiladi va xar bir gurux uchun AVG(AMOUNT) ifodasi xisoblanadi. Ya'ni quyidagilar bajariladi:

So'rovlar xar bir xizmatchaga bittadan guruxga ajratiladi.Xar bir guruxda REP maydoni bir xil qiymatga ega. Xar bir gurux uchun guruxga kiruvchi xamma satrlar bo'yicha AMOUNT ustuni o'rta qiymati xisoblanadi va bita natijaviy satr

xosil qilinadi. Bu qator gurux uchun REP ustuni qiymati vash u gurux uchun so'rov o'rta qiymatini o'z ichiga oladi.

Shunday qilib, GROUP BY ifodasi qo'llanilgan so'rov, "GURUXLANISHLI SO'ROV " deb ataladi! SHu ifodadan keyin kelgan ustun "guruxlash ustuni " deyiladi. Yana bir necha guruxlanishli so'rovlarni ko'rib chiqamiz.

Xar bir ofis uchun sotuvlarning rejalashtirilgan xajmi diapazoni qancha?

SELECT REP_OFFICE, MIN(QUOTA), MAX(QUOTA) FROM SALESREPS GROUP BY REPOFFICE

Yana bir so'rov: Xar bir ofisda qancha xizmatchi ishlaydi?

SELECT REPOFFICE, COUNT(*) FROM SALESREPS GROUP BY REP_OFFICE

Yana bir guruxlanishli qiziqarli so'rov: Xar bir xizmatchi nechta mijozga xizmat ko'rsatadi?

SELECT COUNT(DISTINCT CUST_NUM), 'CUSTOMERS FOR SALESREPS', CUST_REP FROM CUSTOMERS GROUP BY CUST_REP

Bu yerda 'CUSTOMERS FOR SALESREPS' psevodomaydonning ishlatilishiga e'tibortbering. So'rov natijalarini bir nechta ustun bo'yicha guruxlash mumkin. Masalan, quyidagicha:

Xar bir xizmatchi uchun xar bir klient bo'yicha buyurtmalar umumiy sonini xisoblash.

SELECT REP, CUST, SUM(AMOUNT) FROM ORDERS GROUP BY REP, CUST

Lekin ikki ustun bo'yicha guruxlashda natijalar ikki darajasiga ega guruxlar va ostki guruxlar yaratish mumkin emas. Lekin tartiblashni qo'llash mumkin. SHu bilan birga GROUP BY ishlatilganda so'rov natijalari avtomatik tartiblanadi.

Quyidagi so'rovni ko'ramiz:

Xar bir xizmatchi uchun xar bir klient bo'yicha buyurtmalar umumiy sonini xisoblash; so'rov natijalarini klientlar va xizmatchilar bo'yicha tartiblash.

SELECT REP, CUST, SUM(AMOUNT) FROM ORDERS GROUP BY REP, CUST ORDER BY REP, CUST

Shunday qilib GROUP BY ifodasi SELECT ni guruxlarni qayta ishlashga majbur qiladi. MS SQL serverida COMPUTE ifodasi mavjud bo'lib relyasion so'rovlar asoslariga zid keladi. Lekin uning yordamida saqlanuvchi protseduralardan foydalanmasdan shunga o'xshash natijalarni olish mumkin. Ruruxlanishli so'rovlar uchun chegaralar mavjud. Satrlarni xisoblanuvchi ifoda asosida guruxlash mumkin emas. Qaytarilao'tgan qiymatlar elementlariga xam chegaralar mavjud. Qaytariluvchi ustun bo'lishi mumkin:

Konstantalar.

A. Guruxga kirgan xamma satrlar uchun bitta qiymat qaytaruvchi agregat funksiya.
B. Gurux xamma satrlarida bir xil qiymatga ega guruxlash ustuni.
C. Ko'rsatilgan elementlarni o'z ichiga oluvchi ifoda.

Odatda guruxlanishli so'rovlar qaytaruvchi ustunlarga guruxlash ustuni va agregat funksiya kiradi. Agar agregat ko'rsatilmasa GROUP BY dan foydalanmasdan DISTINCT ifodasidan foydalanish etarli. Agar so'rovga guruxlash ustuni qo'shilmasa, u yoki bu satr qaysi guruxga tegishliligini aniqlash mumkin emas. Shu kabi SQL92 guruxlanishli so'rovlarni taxlil qilishda birlamchi va ikkilamchi kalitlar xaqidagi ma'lumot ishlatilmaydi.

Xar bir xizmatchi uchun buyurtmalar umumiy sonini xisoblash.

SELECT EMPL_NUM, NAME, SUM(AMOUNT) FROM ORDERS, SALESREPS WHERE REP = EMPL_NUM GROUP BY EMPL_NUM, NAME

Yana soddaroq shakl: Xar bir xizmatchi uchun buyurtmalar umumiy sonini xisoblash.

SELECT NAME, SUM(AMOUNT) FROM ORDERS, SALESREPS WHERE REP = EMPL_NUM GROUP BY NAME

Agar guruxlash maydonlaridan birida NULL qiymat mavjud bo'lsa qaysi guruxga tegishli bo'ladi? WHERE ifodasida NULL va NULL tenglikka solishtirish natijasi Yana NULL beradi. SHuning uchun ANSI/ISO standartida GROUP BY ifodasida NULL qiymatlar teng deb qabul qilingan.

Guruxlash va HAVING yordamida ajratish:

Shart bo'yicha satrlarni ajratish uchun WHERE ifodasidan foydalangan edik. Shart bo'yicha guruxlarni ajratish uchun HAVING operatori mavjuddir. Uning sintaksisi WHERE operatori bilan bir xil va ulardan birgalikda foydalanigsh mumkin. Quyidagi so'rovni ko'ramiz: Buyurtmalar umumiy narxi $300 dan ortiq xizmatchilar uchun buyurtma o'rtacha narxi qanchaga teng?

SELECT REP, AVG(AMOUNT) FROM ORDERS GROUP BY REP HAVING SUM(AMOUNT) > 300

Ko'rinib turibdiki HAVING SUM(AMOUNT) > 300 ifodasi satrlarni guruxlash Sharti sifatida kelmoqda. Agar SUM(AMOUNT) > 300 Sharti yolg'on bo'lsa, bu gurux natijaviy to'plamdan chiqariladi. Agar rost bo'lsa gurux natijaviy to'plamga kiradi! Yana bir misol ko'raylik: Ikki va undan ortiq xizmatchiga ega xar bir ofisning xamma xizmatchilari uchun rejadagi va xaqiqiy sotuvlar umumiy xajmini xisoblash.

SELECT CITY, SUM(QUOTA), SUM(SALESREPS.SALES) FROM OFFICES, SALESREPS WHERE OFFICE = REP_OFFICE GROUP BY CITY HAVING COUNT(*) >= 2

Bu misolda WHERE va HAVING ifodalari o'z funksiyalarini bajaradilar. Na shunga e'tibor berish kerakki HAVING ifodasida agregat funksiyalardan foydalaniladi, So'rov bajarilishini ko'ramiz:

OFFICES va SALESREPS jadvallari xizmatchi yashaydigan gshaxarni topish uchun qo'shiladilar.

Qo'shilgan jadval satrlarlari ofislar bo'yicha guruxlanadilar.

Ikkidan kam satrga ega guruxlar tashlab yuboriladi. Ular HAVING ifodasi talabiga javob bermaydilar. Xar bir gurux uchun xaqiqiy va rejadagi sotuvlar xajmlari xisoblanadi. Murakkabroq misolni ko'ramiz: Xar bir tovar nomi uchun narxi, ombordagi soni va buyurtma berilganlar umumiy sonini ko'rsating, agar uning uchun buyurtma berilganlar umumiy soni ombordagi umumiy soni 75 foizidan ko'p bo'lsa.

SELECT DESCRIPTION, PRICE, QTY_ON_HAND, SUM(QTY) FROM PRODUCTS, ORDERS WHERE MFR = MFR_ID GROUP BY MFR_ID, PRODUCT_ID, DESCRIPTION, PRICE, QTY_ON_HAND HAVING SUM(QTY) > (0.75 * QTY_ON_HAND) ORDER BY QTY_ON_HAND DESC

HAVING uchung qo'shimcha chegaralar mavjuddir. Bu ifoda juda bo'lmasa bita agregat funksiyani o'z ichiga olishi kerak. Chunki WHERE aloxida satrlarga HAVING satrlar guruxlariga qo'llanadi. NULL qiymat uchun WHERE ifodasiga o'xshab quyidagi qoida o'rinli Agar izlash Sharti NULL qiymatga ega bo'lsa satrlar guruxi tashlab yuboriladi. HAVING ifodasini GROUP BY siz qo'llash mumkin. Bu xolda natija xamma satrlardan iborat gurux deb qaraladi, lekin amalda bu kam qo'llanadi.

Nazorat savollari

1. Agregat funksiyalar qo'llanishiga misollar keltiring?
2. Guruxlash komandasi uchun so'rov yozing.
3. Having bilan Where ni farqlarini keltiring.
4. Null qanday qiymat hisoblanadi?
5. Standart funksiyaga misollar keltiring?

12-mavzu. Tranzaksiyalarni boshqarishda so'rovlar yaratish va qayta ishlash.

Reja

1. SQL muhitida tranzaksiya tushunchasi.

2. SQL muhitida tranzaksiyalani boshqarish.
3. Arifmetik jarayonlar.
4. Hisoblash tartibini belgilash.
5. Triggerlar va ulardan foydalanish.
6. POSITION() funksiyalaridan foydalanib pastki satrni qidirish.
7. CASE ifodasini ishlatib shartli qiymatlarni ifodalash.
8. Joriy satsiyasi.

Tayanch so'zlar: tranzaksiva, himoyalanish, ROLLBACK, Commit.

SQL muhitida tranzaksiva tushunchasi. SQL tilida tranzaksiya deb, ma'lumotlami tiklashga nisbatan ajralmas bo'lgan operatorlar ketma - ketligiga aytiladi. SQL tilidagi har bir chaqirish moduli tranzaksiyadir. SQL tili tranzaksiyalari biror-bir modulning protseduralarini bajarishdan boshlanadi. COMMIT yori ROLLBACK operatorining bajarilishi bilan tugaydi. Agar tranzaksiya ROLLBACK operatori bilan tugasa, prtoseduradagi barcha qilingan amallar bekor qilinadi.

Har bir tranzaksiyaning "faqat o'qish" yoki "o'qish va yozish" tartiblari mavjud. Tranzaksiya tartiblari SETTTRANSACTION operatori yordamida o'rnatiladi. Jimlik qoidagisa nusbatan "o'qish va yozish" tartibi o'rnatiladi. "Faqat o'qish" tartibi doimo saqlanadigan bazaviy ma'lumotlarga qo'llaniladi.

Har bir SQL tranzaksiyasi himoyalanish darajasiga ega: READUNCOMMITTED, READCOMMITTED, REPEATABLEREAD yoki SERIALIZABLE. SQL tranzaksiyasi himoyalanish darajalari bajarilayotgan tranzaksiyaning boshqa parallel bajarilayotgan tranzaksiyalarga ta'sir etish daragasini aniqlaydi. Tranzaksiyaning aniq darajasini o'rnatish uchun SETTRANSACTION operatoridan foydalanishadi. Jimlik qoidasiga nisbatan SERIALIZABLE tartibi o'rnatiladi.

Himoyalanish daragalari tranzaksiyalarning parallel bajarilishida yuz berishi mumkin bo'lgan hodisalami aniqlaydi. Qyuidagi ko'rinishda gi hodisalar bo'lishi mumkin:

> PI ("Dirtyread" - "Yomon o'qish"): T1 tranzaksiya qatomi yaratadi. Keyin T2 tranzaksiya T1 COMMIT amalini bajarmasdan bu qatomi o'qiyi. Shundan so'ng T1 ROLLBACK amalini bajarsa, T2 tranzaksiya umuman mavjid bo'lmagan qatomi o'qigan bo'lib chiqadi.

> P2 ("Non-repeaiableread" - "Takrorlanmaidigan o'qish"): T1 tranzaksiya qatomi o'qiydi. Shundan so'ng T2 tranzaksiya bu buyruqlar qatorini o'zgartiradi yoki olib tashlayda va COMMITni bajaradi. Shundan so'ng T1 shu qatomi yana o'qishga harakat qiladi, ammo bu qator birinchi holatdagi qator emas yoki olib tashlangani uchun topolmaydi.

> P3 ("Phantom" - "Fantom"): T1 tranzaksiya biror-bir shartni qanoatlantiradigan N qatomi o'qiydi. Shundan so'ng T2 tranzaksiya bu qatorlar ichidan bir yoki bir nechta qator shartlarini generatsiya qiladi. Agar shu ishlardan keyin T1 o'qishni qaytarsa, u butunlay boshqa qatorlarga ega bo'ladi,

To'rtala himoyalanish daragalari P1, P2 va P3 hodisalarga nisbatan quyidagicha ta'sirga ega:

Himoyalanish PI P2 P3

READUNCOM	Mumkin	Mumkin	Mumkin
READCOMMIT	Mumkin	Mumkin	Mumkin
REPEATABLE	Mumkin	Mumkin	Mumkin
SERIALIZABL	Mumkin	Mumkin	Mumkin

MB bilan ish jarayonida ma'lumotlar butligi muhim o'rin tutadl. Ma'lumotlar butligi deganimizda, ma'lumotlaming to'g'riligi va mazmunan qarma-qarshi ma'noga ega emasligi tushuniladi. Masalan, "O'qituvchi" jadvalidagi har bir

o'zgarish "Yuklama" jadvalida ham qayd etilishi kerak. 0'qituvchining "Yuklama" jadvalida qayd etilmasligi ma'lumotlar butligining buzilishiga olib keladi.

Ko'pchilik hollarada MBning ma'lumotlari butligini saqlashni tashkillashtirish uchun tranzaksiyalardan foydalanishadi.

Umuman olganda tranzaksiya - bu mantiqan bo'linmaydigan ish birligi.

Bu jarayonda:

- yoki tranzaksiyaga kiruvchi barcha amallar MBda aks etadi;
- yoki bu amallar umuman bajarilmaydi.

Tranzaksiyaning bu xususiyati butlik shartining buzilmasligini ta'minlaydi.

Ko'pgina MBBTda tranzaksiyalaming ikkita modeli ishlatiladi:

- Tranzaksiyalaming avtomatik bajarilish modeli.
- Tranzaksiyalaming bajarilishini boshqarish modeli.

Tranzaksiyalaming avtomatik bajarilish modelida, tranzaksiya avtomatik ravishda ishga tushadi va quyidagi usullardan biri bilan tugaydi:

> COMMIT yo'riqnomasi bilan, bunda MBdagi o'zgarishlar doimiy bajariladigan bo'ladi va yangi tranzaksiya COMMIT buyrug'idan so'ng boshlanadi.

> ROLLBACK yo'riqnomasi bilan, bunda tranzaksiyada bajarilgan barcha o'zgarishlar bekor bo'ladi va yangi tranzaksiya ROLLBACK buyrug'idan so'ng boshlanadi.

Tranzaksiyalaming bajarilishini boshqarish modeli SUBD Sysbase dasturida qo'llanilib, quyidagi yoriqnomalardan foydalaniladi:

S BEGIN TRANSACTION yo'riqnomasi, tranzaksiyaning boshlanishini bildiradi.

S COMMIT TRANSACTION yo'riqnomasi, tranzaksiyaning muvaffaqiyatli tuganini bildiradi. Bunda yangi tranzaksiya avtomatik ravishda ishga tushmaydi.

- SAVE TRANSACTION yo'riqnomasi, bunda tranzaksiya ichida saqlash nuqtasi tashkillashtiradi va saqlangan saqlash nuqtasiga nom berish imkoniyati yaratiladi.
- ROLLBACK yo'riqnomasi, bunda barcha tranzaksiyadagi amallar bekor qilinadi va MB holati tranzaksiyadan oldingi holatga qaytariladi.

Shunday qilib, tranzaksiya - bu MBga tugallangan murojaat bo'lib quyidagi to'rtta shartning bajarilishini kafolatlaydi:

- Bo'linmaslik (atomarnost) - tranzaksiya boshi va oxiriga ega bo'lgan boTinmas blok. Bu blok yoki to'liqligicha bajariladi, yoki umuman bajarilmaydi;
- Kelishuvchanlik - tranzaksiya tugaganidan so'ng, hamma ob'yektlar kelishganlik holatini saqlab qoladi;
- Himoyalanganlik - har bir tranzaksiya jarayoni boshqa tranzaksiya ishiga ta'sir ko'rsatmaydi.
- Doimiylik - tranzaksiya jarayonida bajarilgan barcha o'zgarishlar doimiylik xarakteriga ega.

SQL tilida tranzaksiya jarayoniga misol.

SQL tilida har bir tranzaksiya bitta to'liq jarayonni bajarishiga tushuncha hosil qildik. Ana shunday jarayonga misol keltiramiz:

"Ta'minlovchilar" jadvalidagi Sx raqamini Sy raqamiga o'zganirish lozim bo'lsin. Sx va Sy - berilgan aniq parametr.

```
TRANEX: PROC OPTIONS (MAIN); /* tranzaksiyaga misol*/ EXEC SQL
WNENEVER SQLERROR GO TO UNDO; GET LIST (SX,SY);
EXEC SQL UPDATE S
SET TA'MINLOVCHI RAQAMI = SY WHERE TA'MINLOVCHI RAQAMI =
SX;
EXEC SQL UPDATE SP
SET TA'MINLOVCHI RAQAMI = SY WHERE TA'MINLOVCHI RAQAMI =
SX;
EXEC SQL COMMIT;
GO TO FINISH;
UNDO: EXEC SQL ROLLBACK; FINISH: RETURN;
END TRANEX;
```

Misolimizdan ko'rinib turibdiki, bu tranzaksiya jarayonida ikkita jadval ustuda o'zgarishlar amalga oshirilayapti. Demak, tranzaksiya deganimizda bitta amalni emas, balki amallar ketma-ketligini tushunish lozim.

SQL muhitida tranzaksiyalani boshqarish.

SQL tilida tranzaksiyalami maxsus operatorlat yordamida boshqarish imkoniyati mavjud. Shulardan biri tranzaksiya parametrlarini o'matish operatori bo'lib, uniyozilish formati quyidgicha:

<set transaction statement> ::=

SET TRANSACTION transaction mode> [{ <comma> transaction mode> }...]

transaction mode> ::=

<isolation level>

| <transaction access mode>

| ^diagnostics size>

^isolation level> ::=

ISOLATION LEVEL «level of isolation>

<level of isolation> ::=

READ UNCOMMITTED | READ COMMITTED i REPEATABLE READ | SERIALIZAB1 1

transaction access mode>

READ ONLY | READ WRITE

diagnostics size> ::=

DIAGNOSTICS SIZE <number of conditions>

<number of conditions> ::= <simple value specification>

Bu yerda:

- Agar himoya darajalari ko'rsatilmasa, himoya darajasi SERIALIZABLE deb tushumniladi.
- Agar ruxsat tartibi READWRITE kalit so'zi bilan belgilansa, unda himoyalanish darajasi READUNCOMMITTED bo'lmasligi kerak.

> Agar ruzsat tartibi va himoyalanish darajasi READUNCOMMITTED deb ko'rsatilsa, unda beriladigan ruxsat tartibi READONLY deb tushuniladi, aks hollarda ruxsat tartibi READWRITE bo'ladi.

Ko'pchilik hollarda tranzaksiyalaming bajarilish jarayonida MB jadvallari stmkturasi buzilishining oldini olish uchun tranzaksiyalarga faqat o'qish tartibini o'rnatish mumkin. Buning uchun quyidagi operator ishlatiladi:

SET TRANSACTION READ ONLY;

Bu operator tranzaksiya jarayoni boshlanishidan oldin ko'rsatiladi. Masalan,

EXEC SQL SET TRANSACTION READ ONLY;

Buyurtmalarni qabul qilgan sotuvchini aniqlamoqchmiz. Bu ikki so'rovni bir-biridan farq qilishi uchun matn kiritish yo'li bilan tashkillashtirisimiz mumkin:

SELECT a.snum, sname, onum, 'Highest on', odate FROM Salespeople a.
Orders b WHERE a.snum = b.snum AND b.amt = (SELECT MAX (amt) FROM Orders c WHERE c.odate = b,odate)
UNION
SELECT a.snum, sname, onum, 'Lowest on', odate FROM Salespeople a, Orders b WHERE a.snum b.snum AND b.amt = (SEi FCT MIN (amt)FROM Orders c WHERE c.odate ; b.odate);

10 Peel 30 High o 10/05/
10 Peel 30 Low o 10/05/
10 Peel 30 High o 10/06/
10 Serre 30 High o 10/03/
10 Serre 30 Low o 10/04/
10 Serre 30 Low o 10/06/
10 Axel 30 High o 10/04/

Shu paytgacha UNION kalit so'zi yordamida birlashtirilgan so'rov natijalari qanday tartibda matnga chiqarilishi to'g'risida gapirmadik. Birlashtirilgan natijalarni ORDER BY kalit so'zi yordamida tartiblashtirish mumkin. Yoqoridagi misolni tartub raqamlariga nisbatan tartiblashni ko'rib o'tamiz.

SELECT a.snum, sname, onum, 'Highest on', odate FROM Salespeople a. Orders b WHERE a.snum = b.snum AND b.amt = (SELECT MAX (amt) FROM Orders c WHERE c.odate = b.odate)

UNION

SELECT a.snum, sname, onum. 'Lowest on', odate F ROM Salespeople a. Orders b WHERE a.snum = b.snu AND b.amt = (SELECT MIN (amt) FROM Orders c WHERE c.odate = b.odate) ORDER BY 3; Natija quyidagicha:

1007 Rifkin	3001	Lowest on	10/03/199
1002 Serres	3005	Highest on	10/03/199
1002 Serres	3007	lowest on	10/04/199
1001 Peel	3008	Highest on	10/05/199
1001 Peel	3008	Lowest on	10/05/199
1003 Axelrod	3009	Highest on	10/04/199
1002 Serres	3010	Lowest on	10/06/199
1001 Peel	3011	Highest on	10/06/199

Nazorat savollari

1. SQL muhitida tranzaksiyaning vazifasi nimadan iborat?
2. TCL uchun muhim jarayon qaysi?
3. Commit nima vazifani bajaradi? Misol keltiring.
4. Tranzaksiyalarni boshqarishni tushuntiring?
5. Rollback uchun misol keltiring?

13-Mavzu. SQL Serverda ma'lumotlar bazasini administratorlash va xavfsizligini ta'minlash.

Reja

1. SQL Serverda ma'lumotlar bazalari ob'yektlari himoyasi.
2. SQL Server hisob yozuvlarini boshqarish.
3. Protseduralar va ularni yaratish.

4. Ma'lumot bazasini administratori.
5. Ma'lumotlar bazasini loyihalash, uzatish va samaradorligini oshirish.

Tayanch so'zlar: SQL server, protsedura.

SQL Server foydalanuvchilar darajasida ma'lumotlar bazalarining ichki himoya tizimiga ega. SQL Server va undagi ma'lumotlar bazasiga faqat serverda ro'yxatdan o'tgan, mos huquqlarga ega foydalanuvchi ulanishi mumkin.

SQL Serverda ma'lumotlar bazalari ob'yektlari himoyasi:

SQL Serverda saqlanuvchi boshqa ob'yektlarni (jadvallar, tasavvurlar, saqla- nuvchi protseduralar va ma'lumotlar sxemalari) himoya qilish- ning ikki usuli mavjud.

Tasavvurlar, saqlanuvchi protseduralar va triggerlarni shifrlash mumkin. Shifrlangandan so'ng tasavvur strukturasini o'zgartirish mumkin emas. Lekin tasavvurdan manbada ma'lumotlarni tahrirlash uchun foydalaniladi. Tasavvurni qanday shifrlash haqidagi ma'lumotni SQL Server hujjatidan olish mumkin. Saqlanuvchi protsedura yoki triggerni shifr- lash uchun ularni oldin tahrirlash va maxsus Transact-SQL yoriqnomasini yozish kerak, masalan: CREATE PROCE¬ DURE WITH ENCRYPTION.

SQL Server Enterprise Manager dasturi yordamida SQL Server himoya qilish vositalaridan foydalanish mumkin. Bu himoya vositalari haqidagi ma'lumotni SQL Server hujjatlaridan olish mumkin.

Agar shifrlangan tasavvur strukturasini keyinchalik o'zgartish kerak bo'lishi mumkin bo'lsa quyidagi maslahatdan foydalaning. Tasavvurni aniqlovchi SQL yoriqnomani matnli faylda saqlab qo'ying. Ishonchli joyda mustahkam nusxani saqlab qo'ying. Tasavvurni shifrlang. Kerak bo'lsa shifrlangan tasavvur struk- turasini o'zgartiring:

Oldingi shifrlangan tasavvurni o'chirish.

Oldingi tasavvur bilan bir xil nomdagi yangi tasavvur yarating.

Saqlangan matnli fayldagi SQL yoriqnomadan almashish buferiga nusxa oling. Uni yangi tasavvur Konstruktorining SQL yo'l-yo'riq kiritish maydoniga joylashtiring.

Tasavvur strukturasini o'zgartirish.

O'zgartirilgan SQL yoriqnomani matnli faylda saqlang. Bu faylni ishonchli joyga joylashtiring.

SQL Server hisob yozuvlarini boshqarish

MB himoya tizimini boshqarish vazifasini (Tools) menyusidagi (Database Security) buyrug'i yordamida bajarish mumkin. Agar SQL Server loyihasi saqlanayotgan kompyuterda o'rnatilgan bo'lsa bu buyruqqa murojaat qilish mumkin. Bu vosita yordamida SQL Serverda registratsiya qilish uchun hisob yozuvlarini, ma'lumotlar bazalari foydalanuvchilari hisob yozuvlarini va ularning vazifalarini qo'shish, o'chirish va o'zgartirish mumkin.

SQL Serverda registratsiya qilish uchun qo'llanadigan ikki himoya tizimi mavjud:

- SQL Server o'zining himoya tizimi. Serverda registra- tsiyadan o'tish uchun server foydalanuvchisi nomi va parolini ko'rsatish kerak.
- Windows NT bilan Integratsiyalashgan tizimi foydalanuvchilari hisob yozuvlaridan foydalanadi. Bu holda foydalanuvchi autentifikatsiyasi Windows NT asosida tarmoqda registratsiyadan o'tishda bajariladi.

SQL tilida protseduralardan foydalanish dasturlar tuzish samaradorligini oshiradi. Saqlanuvchi protseduralar (stored protsedure) - bu SQL buyruqlar to'plamidan iborat bo'lib, bu buyruqlar to'plamini SQL SERVER bir marta kompilyatsiya qiladi.

Protseduralarning keyingi ishlatilishida saqlangan protseduralar kompilyatsiya qilinmaydi. Bu protseduralar xuddi algoritmik tillardagi kabi kirish parametrlaridan iborat bo'lishi ham mumkin.

Saqlanuvchi protseduralar SQL tilida quyidagi buyruq yordamida yaratiladi:

CREATE PROCEDURE <protsedura nomi>

[(% birinchi parametr ma'lumoti turi)] ...] AS SQL-operatorlari;

Saqlanuvchi protseduralarning ikki turi mavjud: foydalanuvchi protseduralari va tizimli protseduralar.

Foydalanuvchi protseduralari SQL SERVERlarida qo'llanilib, serverni boshqarish, MB va foydalanuchilar haqidagi ma'lumotlarni olish uchun ishlatiladi. Tizimli protseduralar esa, amaliy dasturlarni bajarish uchun yaratiladi. Amaliy dasturlar hech bo'lmaganda bitta modulni o'zida saqlashi kerak. Modul (MODULE) biror bir algoritmik tilda tuzilgan, uzoq muddat saqlanadigan ob'yektdir.

Modul - modul nomidan (module name), algoritmik til bo'limidan (language clause), modul bo'limi huquqidan (module authorization clause), kursorlarni tavsiflash (declare cursor) va bir yoki bir nechta protsedura (procedure) lardan tashkil topadi.

Modul sintaksisi quyidagicha:

<Module> ::= <module name clause>

<language clause>

<module autorization clause> [<declare cursor>...]

< procedure > ...

<language clause>::=LANGUAGE{ COBOL | FORTRAN | PASCAL | PLI }

<module autorization clause> ::= AUTHORIZATION <module autorization identifier>

<module autorization identifier> ::= <autorization identifier>;

Modullarni yaratushda quyidagi sintaktik qoidalarga rioya qilish lozim bo'ladi:

Har bir aniqlangan kursorda (cursor declare) hech bo'lmaganda bitta modul (module) va bu modulda hech bo'lmaganda bitta protsedura (protcedure) mavjud bo'lishi kerak, hamda bu protsedura ochish operatori (open statement) va tavsiflashda (cursor declare) e'lon qilinadigan kursor nomini (cursor name) o'zida aks ettishi ozim. Amaliy dastur bittadan ortig modul bilan ishlamasligi kerak.

Protsedura o'z navbatida potsedura nomidan (procedure name), parametrlar tavsifi (parametrs declaration) va hech bo'lmaganda bitta SQL operatoridan (SQL statment) tashkil topadi.

Moduldan tashkil topgan amaliy dastur potseduraga murojaat qilish uchun CALL operatoridan foydalanadi. CALL operatori potsedura nomidan (procedure name), parametr qiymatalri ketma-ketligidan, son va ma'lumotlar turidan iborat.

Protseduraga murojaat protsedurada mavjud bo'lgan SQL operatorlarini bajarishni ta'minlaydi.

SQL tilida protseduralar quyidagicha yaratiladi.

<procedure> ::=PROCEDURE <procedure name>

<parameter declaration>...

<SQL statment>;

Bu yerda,

<parameter declaration>::=<parameter name> <data type>

| <SQLCODE parameter>

<SQLCODE parameter> ::=SQLCODE

<SQL statement> ::=<close statement>

| <commit statement>

| <delete statement positioned>

| <delete statement searched>

| <fetch statement>

| <insert statement>

| <open statement>

| <rollback statement>

| <select statement>

| <update statement positioned>

| <update statement searched>

Protseduralarni yaratishda quyidagi sintaktik qoidalarga amal qilish lozim:

Protsedura nomi modulda ishtirok etadigan boshqa protsedura nomlaridan farq qilishi lozim.

a) Protsedura paramaetrlari ham boshqa protsedura parametrlaridan farq qilishi lozim.
b) Har bir parametr nomi (parametr declaration) tavsifida ko'rsatilgan bo'lishi ozim.
c) Agar SQL operatoridagi ustun nomi (column names) (parametr declation) tavsifida ko'rsatilgan parametr nomi bilan mos tushsa, bunday ustun nomlari (column specification) oldiga kvalificator (qualifier) qo'yiladi.
d) Til bo'limida (language clause) modulda ishlatiladifan algoritmik til nomi ko'rsatiladi. Har bir algoritmik tilni ishlatishda o'ziga xos qoidalarga rioya qilishga to'g'ri keladi.

1. SQLCODE parametrining turi INTEGER bo'lishi kerak;
2. Har qanday ishlatiladigan ma'lumot turlari (data type) CHAR, INTEGER va REAL bo'lishi talab qilinadi;
3. Agar (parametrs declaration) tavsifida berilgan parametr turi (data type) INTEGER yoki REAL bo'lsa, shu parametrlarga mos keluvchi i- parametr turi ham INTESER yoki REAL bo'lishi kerak.

Tizimli protseduraga misol keltiramiz. MBdagi detallarni hajm jihatidan katta kichikligiga qarab Detallar jadvalidan izlash lozim bo'lsin. Buni quyidagi rekursiv protsedura yordamida amalga oshiramiz.

```
GET LIST(Kiritilayotgandetal);
CALL RECURSION(Kiritilayotgandetal); RETURN;
RECURSION: PROC(Kattadetal) RECURSIVE;
DCL Kattadetal CHAR(30); DCL Kichikdetal CHAR(30);
EXEC SQL DECLARE C KURSOR FOR
SELECT Detalraqami FROM Detallar WHERE Asosiy_detal=Katta_detal
AND Detal_raqami>Kichik detal ORDER BY Asosiydetal;
EXEC SQL CLOSE C;
```

CALL RECURSION (Kichikdetal); END;

END;

Foydalanuvchilar va ularning imtiyozlari.

SQL muxitida har bir foydalanuvchi maxsus identifikatsiton nom, murojjat identifikatoriga (ID) ega. Ma'lumotlar bazasiga yuborilgan komanda ma'lum foydalanuvchi bilan yoki boshqacha aytganda maxsus murojaat identifikatori bilan bog'lanadi. SQL ma'lumotlar bazasida ID ruxsat - bu foydalanuvchi nomi va SQL komanda bilan bog'langan murojaat identifikatoriga ilova qiluvchi maxsus kalit so'z USER dan foydalanishi mumkin.

Registratsiya bu kompyuter tizimiga kirish huquqini olish uchun foydalanuvchi bajarishi kerak bo'lgan protseduradiryu Bu protsedura foydalanuvchi Bilan qaysi murojaat ID si bog'lanishini aniqlaydi. Odatda har bir ma'lumotlar bazasidan foydalanuvchi o'zining ID siga ega bo'lishi kerak va registratsiya jarayonida haqiqiy foydalanuvchiga aylanadi. Lekin ko'p masalalarga ega foydalanuvchilar bir necha murojaat ID lari bilan registratsiyadan o'tishlari 1ki bir necha foydalanuvchi bitta murojaat ID sidan foydalanishlari mumkin.

Imtiyozlar-Har bir foydalanuvchi SQL ma'lumotlar bazasida nima qilish mumkinlgini ko'rsatuvchi imtiyozlarga egadir. Bu imtiyozlar vaqt o'tishi Bilan o'zgarishi Ya'ni eskilari o'chirilib Yangilari qo'shilishi mumkin. SQL imtiyozlar bu ob'ekt imtiyozlaridir. Bu shuni bildiradiki foydalanuvchi berilgan komandani ma'lumotlar bazasining biror ob'ekti ustida bajarishi mumkin. Ob'ekt imtiyozlari bir vaqtning o'zida foydalanuvchilar va jadvallar Bilan bog'liq. Ya'ni imtiyoz ma'lum foydalanuvchiga ko'rsatilgan jadvalda, asos jadvalda yoki tasavvurda beriladi. Ixtiyoriy turdagi jadvalni yaratgan foydalanuvchi shu jadval egasidir. Bush uni bildiradiki foydalanuvchi bu jadvalda hamma imtiyozlarga ega va imtiyozlarini shu jadvalning boshqa foydalanuvchildariga uzatishi mumkin.

Foydalanuvchiga tayinlash mumkin bo'lgan imtiyozlar:

^ SELECT Bu imtiyozga ega foydalanuvchi jadvallarda so'rovlar bajarishi mumkin.

S INSERT Bu imtiyozga ega foydalanuvchi jadvalda INSERT komandasini bajarishi mumkin.

- UPDATE Bu imtiyozga ega foydalanuvchi jadvalda UPDATE komandasini bajarishi mumkin. Bu imtiyozni jadvalning ayrim ustunlari uchun cheklab qo'yishingiz mumkin.
- DELETE Bu imtiyozga ega foydalanuvchi jadvalda DELETE komandasini bajarishi mumkin.
- REFERENCES Bu imtiyozga ega foydalanuvchi jadvalning ustunidan (yoki ustunlaridan) ajdod kalit sifatida foydalanuvchi tashqi kalit aniqlashi mumkin. Siz bu imtiyozni ayrim ustunlar uchun berishingiz mumkin.

Bundan tashqari siz ob'ekt nostandart imtiyozlarini uchratasiz, masalan INDEX (INDEKS) -jadvalda indeks yaratish huquqini beruvchi, SYNONYM (SINONIM)- ob'ekt uchun sinonim yaratish huquqini beruvchi va ALTER (IZMENIT)- jadvalda ALTER TABLE komandasini bajarish xquqini beruvchi. SQL Mexanizm foydalanuvchilarga bu imtiyozlarni GRANT komandasi yordamida beradi.

GRANT Komandasi- GRANT komandasining 4 formati mavjud bo'lib, ulardan biri konkret ob'ekt ustidan, konkret foydalanuvchilarga konkret imtyozlar berish bo'lib, quyidagi ko'rinishga ega:

GRANT privilege ON [creator.]tablename TO userid, ... [WITH GRANT OPTION]

Bu yerda

- privilege - tayinlanayotgan imtiyozlar ro'yxati,
- tablename - jadval nomi,
- userid - imtyozlar olgan foydalanuvchilar ro'yxati.

Masalan: GRANT SELECT, INSERT ON Orders TO Adrian, Diane;

Ma'lum foydalanuvchilarga imtiyozlarni SQL Central da ikki usul bilan tayinlash mumkin.

Birinchidan: Users & Groups papkasini tanlash va ma'lum foydalanuvchi xossalarini ro'yxatdan chaqirish (sichqoncha o'ng klavishasini bosish va menyu

Properties punktini tanlash). So'ngra Permissions qo'shimcha sahifasida kerakli jadvalni tanlab imtiyozni o'rnatish.

Ikkinchidan: Tables yoki Views papkasida ma'lum jadval yoki tasavvur xossalar oynasini chaqirish, so'ngra Permissions qo'shimcha sahifasiga o'tish va GRANT tugmasi yordamida kerakli foydalanuvchini tanlab, imtiyozni o'rnatish.

GRANT UPDATE (City, Comm) ON Salespeople TO Diane; - bu Diane ga Salepeople jadvalining City va Comm ustunlari qiymatlarini o'zgartirish huquqini beradi yoki GRANT REFERENCES (CName, CNum) ON Customers TO Stephen; - bu komanda Stephen ga CNum va CName ustunlarini o'zining jadvallaridagi ixtiyoriy tashqi kalitlarga nisbatan ajdod kalit sifatida ishlatish huquqini beradi. Stephen (CName, CNum) yoki (CNum, CName) usutunlarni, jadvalarining ikki ustuni Bilan tashqi kalit yordamida mos kelgan ikki -ustunli ajdod kalit sifatida aniqlashi mumkin. Yoki u maydonga individual murojaat qilish uchun ajratilgan tashqi kalitlar yaratishi mumkin.

Nazorat savollari

1. Ma'lumotlar bazalarining ichki himoya vazifasini keltiring?
2. SQL Server himoya vositasiga deganda nimani tushunasiz?
3. GRANT privilege nima vazifani bajaradi? Misol keltiring.
4. GRANT UPDATE uchun misollar yozing?
5. GRANT REFERENCES uchun misol keltiring?

14-mavzu. Ma'lumotlar bazasiga murojaatni tashkil etishda ODBC va ob'ektga yo'naltirilgan dasturlar foydalanish

Reja

1. Ma'lumotlar bazasiga murojaatni tashkil etishda C# dasturi.
2. Ma'lumotlar bazasiga murojaatni tashkil etishda C++ dasturi.
3. ODBS va C++ dasturlash tili yordamida ma'lumotlar bazasiga murojaatlarni tashkil etish usullari.

4. SQL so'rovlardan foydalanish, interfeysni va malumotlar bazasi aloqasini ta'minlash.

Tayanch sozlar: MySqlConnection , MySql.Data , C#, C++.

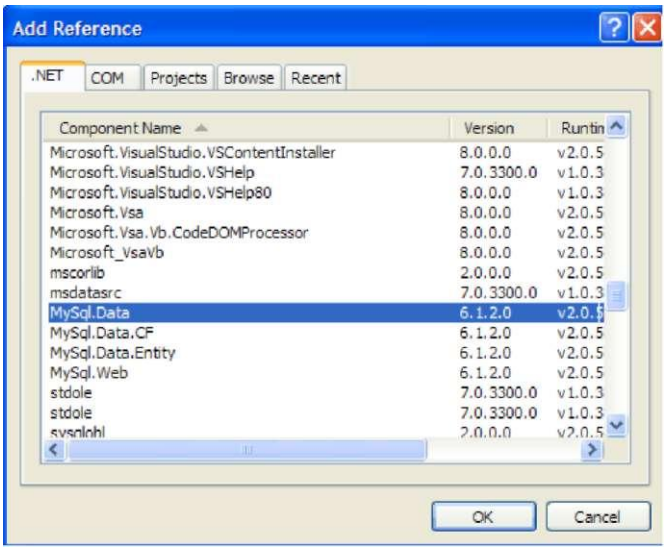
C # ni MySQLga bog'lash. MySQL ma'lumotlar bazasini C # dasturiga ulash uchun MySQL MySQL ulagichi / tarmog'ida bir qator kutubxonalarni qo'shishimiz zarur. C# ilova va MySQL serveri o'rtasidagi barcha aloqa MySqlConnection ob'ekti orqali yuboriladi.

Shunday qilib, dastur server bilan bog'lanishdan oldin, u MySqlConnection ob'ekti, sozlashi va ochishi kerak.

Mysql dasturini quyidagi veb-saytidan MySQL Connector / Netni bepul yuklab olishingiz mumkin.

Havolani bosing: https://dev.mysql.com/downloads/connector/net

Ulanish uchun oldin MySQL kutubxonasini qo'shishingiz kerak, mysql Reference bo'limidan qo'shiladi. Buni amalga oshirish uchun proekt nidan kelib chiqib o'ng tugmasini bosing va "qo'shish"ni tanlang, so'ng ro'yxatdan " MySql.Data " ni tanlang .



14.1 rasm MySQL.Data qo'shish

Keyinchalik, C # loyihangizga MySql Libraryni qo'shishingiz kerak

C# MySQL ulanish quyidagi ko'rinishda bo'ladi:

string myConnectionString = "server=localhost; database=testDB; uid=root; pwd=abc123;";

Quyidagi C# dasturi MySqlConnection ob'ektini yaratish, ulanish satrini tayinlash va ulanishni ochish uchun ishlatiladi.

```
using System;
using System.Windows.Forms;
using MySql.Data.MySqlClient;
namespace WindowsApplication1
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      private void button1_Click(object sender, EventArgs e)
      {
         string connetionString = null;
         MySqlConnection cnn ;
                              connetionString                                    =
"server=localhost;database=testDB;uid=root;pwd=abc123;";
         cnn = new MySqlConnection(connetionString);
         try
         {
            cnn.Open();
            MessageBox.Show ("Connection Open ! ");
            cnn.Close();
         }
         catch (Exception ex)
         {
            MessageBox.Show("Can not open connection ! ");
         }
      }
   }
}
```

Serverlar

Replikatsiya qilingan server konfiguratsiyasida serverga ulanish uchun quyidagi server ulanishi kerak bo'ladi.

myConnectionString = Server=server1, server2; database=testDB; uid=root; pwd=abc123;";

TCP portini ko'rsatish

myConnectionString="Server=myServerAddress;Port=1234;Database=testDB;Uid=root;Pwd=abc 123;

MySQL Connector / Net, Microsoft mahsulotlarining, shu jumladan Microsoft Visual Web Developer-ning Express versiyalarini qo'llab-quvvatlamaydi.

Ma'lumotlar bazasini yaratish:

Ma'lumotlar bazasini va keyinchalik dasturda foydalanadigan jadval yaratamiz.

Buyruq satridan biz ma'lumotlar bazasini yaratishni boshlaymiz:

ConnectCsharpToMysql ma'lumotlar bazasini yaratish ;

Keyin jadval yaratilishidan oldin foydalanish uchun ma'lumotlar bazasini tanlaymiz:

ConnectCsharpToMysqldan foydalanish;

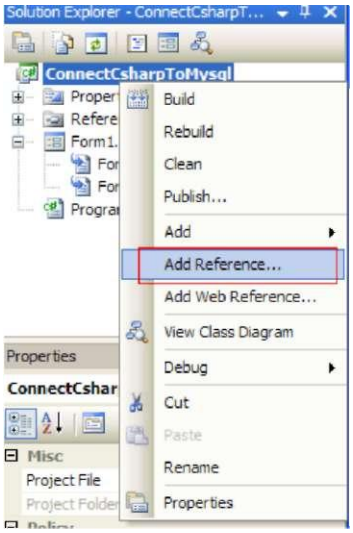
Biz dasturimizdan so'rashimiz mumkin bo'lgan jadvalni yaratamiz:

```
Create Table TableInfo
(
id AVTO AVTOMOZALARNI NULL EMAS,
nomi VARCHAR (30),
yosh INT
PRIMARY KEY ( id)
);
```

Kodlardan foydalanish.

Ma'lumot qo'shish va loyihadan MySQL ulagichi DLL yaratish.

Kodni yozishni boshlashdan oldin , bizning loyihamizga mysql Reference- ni qo'shishimiz kerak . Buning uchun biz loyihamiz nomini o'ng tugmasini bosing va "Add Reference" ni tanlang



14.2 rasm Reference qo'shish

Keyin biz ro'yxatdan MySql.Data- ni tanlaymiz : AddReference 2

Ilovani ulagich o'rnatilmagan boshqa kompyuterlarda ishlatish uchun biz ma'lumotnomadan DLL yaratishimiz kerak. Buning uchun biz loyihamizdagi mos yozuvlar nomini sichqonchaning o'ng tugmasi bilan bosamiz va uning nusxasida lokal ravishda haqiqiyligini o'rnatamiz:

- S Dll qo'shiladi.
- ^ Sinfni yaratiladi.

Ma'lumotlar bazasiga ulanish uchun yangi sinf yaratish va ma'lumotlar bazasiga kiradigan koddan ajratish har doim yaxshiroq. Bu bizning kodimizni toza, o'qishi oson va samaraliroq bo'lishiga yordam beradi.

Biz MySql Connector kutubxonasini qo'shishdan boshlaymiz :

// MySql kutubxonasini qo'shish

MySql.Data.MySqlClient-dan foydalanish;

Keyin biz foydalanadigan o'zgaruvchini e'lon qilish va ishga tushirish:

- > ulanish : ma'lumotlar bazasiga ulanishni ochishda foydalaniladi.
- > server : bizning serverimiz joylashgan joyni ko'rsatadi, bizning holimizda , bu localhost .

> Ma'lumotlar bazasi : bu biz foydalanadigan ma'lumotlar bazasining nomi, bizning holimizda biz oldinroq yaratgan ma'lumotlar bazasi, bu connectcsharptomysql .

> uid : bu bizning MySQL foydalanuvchi nomimiz.

> parol : bu bizning MySQL parolimiz.

> connectionString : ma'lumotlar bazasiga ulanish uchun ulanish satrini o'z ichiga oladi va ulanish o'zgaruvchisiga tayinlanadi.

```
class DBConnect
{
   private MySqlConnection connection;
   private string server;
   private string database;
   private string uid;
   private string password;
   //Constructor
   public DBConnect()
   {
      Initialize();
   }
   //Initialize values
   private void Initialize()
   {
      server = "localhost";
      database = "connectcsharptomysql";
      uid = "username";
      password = "password";
      string connectionString;
      connectionString = "SERVER-' + server + ";" + "DATABASE-' +
```

```
            database + ";" + "UID=" + uid + ";" + "PASSWORD-' + password +
»».»».
  ;  ;
        connection = new MySqlConnection(connectionString);
    }
    //open connection to database
    private bool OpenConnection()
    {
    }
    //Close connection
    private bool CloseConnection()
    {
    }
    //Insert statement
    public void Insert()
    {
    }
    public void Update()
    {
    }
    public void Delete()
    {
    }
    //Select statement
    public List <string> [] Select()
    {
    }
    //Count statement
    public int Count()
    {
```

```
        }
        //Backup
        public void Backup()
        {
        }
        //Restore
        public void Restore()
        {
        }
    }
```

Ulanishni ochish va yopish

Biz doimo jadvallarimizga murojaat qilishdan oldin ulanishni ochishimiz va uni tugatgandan so'ng darhol o'chirib qo'yishimiz kerak, bu resurslarni bo'shatish va bu ulanish endi kerak emasligini bildiradi.

Ma'lumotlar bazasiga ulanishni ochish va yopish juda oson, ammo har doim ulanishni ochmasdan yoki yopmasdan oldin istisnolardan foydalanish, xatolarni ko'rish va ular bilan shug'ullanish yaxshidir.

```
// ma'lumotlar bazasiga ochiq ulanish
private bool OpenConnection()
{
   try
   {
      connection.Open();
      return true;
   }
   catch (MySqlException ex)
   {
      switch (ex.Number)
      {
```

```
            case 0:
                MessageBox.Show("Cannot connect to server.  Contact administrator");
                break;
            case 1045:
                MessageBox.Show("Invalid username/password, please try again");
                break;
        }
        return false;
    }
}
private bool CloseConnection()
{
    try
    {
        connection.Close();
        return true;
    }
    catch (MySqlException ex)
    {
        MessageBox.Show(ex.Message);
        return false;
    }
}
```

DML bilan ishlash (qo'shish, yangilash, tanlash, o'chirish).

Odatda, kiritish, yangilash va o'chirish ma'lumotlar bazasida ma'lumotlarni yozish yoki o'zgartirish uchun ishlatiladi, Select esa ma'lumotlarni o'qish uchun ishlatiladi.

Shu sababli, biz ushbu so'rovlarni bajarish uchun har xil usullarga egamiz. Usullari quyidagilar:

ExecuteNonQuery : misol uchun, har qanday ma'lumotlarni qaytarib bo'lmaydi buyruqni bajarishi uchun ishlatiladi Insert , yangilash yoki o'chirish.

ExecuteReader : 0 yoki undan ortiq yozuvlarni qaytaradigan buyruqni bajarish uchun foydalaniladi, masalan Select .

ExecuteScalar : Faqat 1 qiymatni qaytaradigan buyruqni bajarish uchun foydalaniladi, masalan, Hisoblashni tanlang (*).

Ma'lumotlar bazasiga ochiq ulanish.

MySQL buyrug'ini yarating.

Buyruqqa ulanish va so'rovni tayinlang. Buni konstruktor yordamida yoki MySqlCommand sinfidagi Connection

va CommandText usullaridan foydalanib bajarish mumkin .

```
public void Insert()
{
   string query = "INSERT INTO tableinfo (name, age) VALUES('John Smith', '33')";
   if (this.OpenConnection() == true)
   {
      //create command and assign the query and connection from the constructor
      MySqlCommand cmd = new MySqlCommand(query, connection);
       cmd.ExecuteNonQuery();
      this.CloseConnection();
   }
}
public void Update()
{
   string query = "UPDATE tableinfo SET name='Joe', age='22' WHERE name='John Smith'";
   if (this.OpenConnection() == true)
   {
```

```
        //create mysql command
        MySqlCommand cmd = new MySqlCommand();
        //Assign the query using CommandText
        cmd.CommandText = query;
        //Assign the connection using Connection
        cmd.Connection = connection;
        //Execute query
        cmd.ExecuteNonQuery();
        //close connection
        this.CloseConnection();
    }
}
//Delete statement
public void Delete()
{
    string query = "DELETE FROM tableinfo WHERE name-John Smith'";
    if (this.OpenConnection() == true)
    {
        MySqlCommand cmd = new MySqlCommand(query, connection);
        cmd.ExecuteNonQuery();
        this.CloseConnection();
    }
}
```

Endi MySQLni C ++ ga ulashni ko'rib chiqamiz. Ma'lumotlar bazasiga yordamida C / C ++ ulanish.

SQL (Structured Query Language) - bu to'rtinchi avlod tili (4GL) bo'lib, u MBBTni aniqlash, boshqarish va boshqarish uchun ishlatiladi (ma'lumotlar bazasini boshqarish tizimi).

Kod:

C: \ SQLAPI \ lib \ libsqlapiddll.a

C: \ Program Files \ CodeBlocks \ MinGW \ lib \ libuser32.a

C: \ Program Files \ CodeBlocks \ MinGW \ lib \ libversion.a

C: \ Program Files \ CodeBlocks \ MinGW \ lib \ liboleaut32.a

C: \ Program Files \ CodeBlocks \ MinGW \ lib \ libole32.a

Ular sizning SQLAPI ++ da topiladi (Agar siz C: drayverida ajratib olmagan bo'lsangiz, tegishli joyni tanlang va bog'langan sozlamalarga ushbu fayllarni qo'shing).

Yuqoridagi kod C / C ++ dasturini SQLAPI bilan ulash uchun kutubxona fayllarini qo'shish uchun ishlatiladi.

Asosan, ikkita bosqich mavjud:

1. Ma'lumotlar bazasiga ulanish (va xatolar bilan ishlov berish)

```
// Ma'lumotlar bazasiga ulanish uchun C ++ pgroram (va xatolar bilan ishlash)
#include<stdio.h>
#include<SQLAPI.h>              // main SQLAPI++ header
int main(int argc, char* argv[])
{
    // create connection object to connect to database
    SAConnection con;
    try
{
con.Connect ("test", // database name
"tester", // user name
"tester", // password
SA_Oracle_Client); //Oracle Client
printf("Ulandi!\n");
con.Disconnect();
printf("Uzildi!\n");
```

```
        }
    catch(SAException & x)
    {
    try
    {
    con.Rollback ();
    }
    catch(SAException &)
    {
    }
    printf("%s\n", (const char*)x.ErrText());
        }
        return 0;
}
```

Ulandi !

Uzildi !

SQL oddiy buyrug'ini bajarish.

Endi biz sodda SQL so'rovini bajaramiz. Dastlab ma'lumotlar bazasi uchun jadval yaratamiz :

create table tb1(id number, name varchar(20);

Endi, so'rovni ma'lumotlar bazasiga yuborish uchun cmd.setCommandText usulidan foydalanish kerak, u quyidagicha ko'rsatiladi:

con.Connect("test", "tester", "tester", SA_Oracle_Client);

cmd.setCommandText("create table tb1(id number, name varchar(20));");

and now, to execute the query we have to use the following command:

cmd.Execute();

Full Code:

```
#include<stdio.h>
#include <SQLAPI.h> // main SQLAPI++ header
```

```
int main(int argc, char* argv[])
{
    SAConnection con; // connection object to connect to database
    SACommandcmd; // create command object
    try
{
    // connect to database (Oracle in our example)
    con.Connect("test", "tester", "tester", SA_Oracle_Client);
    // associate a command with connection
    // connection can also be specified in SACommand constructor
    cmd.setConnection(&con);
    // create table
    cmd.setCommandText("create table tbl(id number, name varchar(20));");
    cmd.Execute();
    // insert value
    cmd.setCommandText("Insert into tbl(id, name) values (1,"Vinay")");
    cmd.setCommandText("Insert into tbl(id, name) values (2,"Kushal")");
    cmd.setCommandText("Insert into tbl(id, name) values (3,"Saransh")");
    cmd.Execute();
    // commit changes on success
    con.Commit();
    printf("Table created, row inserted!\n");
        }
    catch(SAException &x)
        {
    try
    {
        // on error rollback changes
        con.Rollback();
```

```
        }
        catch(SAException &)
        {
        }
        printf("%s\n", (const char*)x.ErrText());
        }
        return 0;
}
```

```
con.Commit ( );
```

Yuqorida biz ma'lumotlar bazasiga murojaatni tashkil etishda C++,C# dasturlariga bog'lash va ularni kod jixatdan taxlilini keltirib o'tdik.

Nazorat savollari

1. Ma'lumotlar bazalarining ichki himoya vazifasini keltiring?
2. SQL Server himoya vositasiga deganda nimani tushunasiz?
3. GRANT privilege nima vazifani bajaradi? Misol keltiring.
4. GRANT UPDATE uchun misollar yozing?
5. GRANT REFERENCES uchun misol keltiring?

15 - mavzu. XML va ma'lumotlar bazalari

Reja

1. XML xaqida umumiy tushunchalar.
2. XML ning vazifasi.
3. Native XML ma'lumotlar bazasida ma'lumotlarni saqlash.
4. XML(Extended Markup Language) kengaytirilgan hoshiyalash tili sifatida.
5. Hujjatlarga nisbatan ma'lumotlar.
6. Shablon asosida so'rovlar tillari.
7. XML so'rovlari tillari.

Tayanch so'zlar: XML, XSLT, MBBT.

Ushbu bo'lim ma'lumotlar bazalari bilan XMLdan qanday foydalanish xaqida yuqori darajadagi ma'lumot beradi. Unda ma'lumot markazlashtiruvchi va markazidagi hujjatlar o'rtasidagi farq ularning ma'lumotlar bazalaridan foydalanishga qanday ta'sir qilishi, qanday qilib XML ma'lumotlar bazasi relatsion ma'lumotlar bazasida ishlatilishi va mahalliy XML ma'lumotlar bazalari nima va ulardan qachon foydalanish kerakligi tasvirlangan.

Izoh: Ushbu hujjatda muhokama qilingan ma'lumotlar (asosan) zamonaviy bo'lsa-da, XML va ma'lumotlar bazasini ma'lumotlar-markazli / hujjat-markaziy bo'linish orqali ko'rish mumkinligi haqidagi fikr ma'lum darajada eskirgan. Shunday qilib, ma'lumotlarga asoslangan / hujjatlarga asoslangan bo'linish qulay boshlang'ich nuqtasi bo'lsada, XMLga asoslangan ma'lumotlar bazalari va mahalliy XML ma'lumotlar bazalari o'rtasidagi farqlarni tushunish va ishlov berish ehtiyojlariga qarab tegishli ma'lumotlar bazasini tanlash yaxshidir.

XML ma'lumotlar bazasimi?

XML va ma'lumotlar bazalari haqida gapirishni boshlashdan oldin, biz ko'p odamlarda paydo bo'lgan savolga javob berishimiz kerak: "XML ma'lumotlar bazasimi?" XML faqat atamaning qat'iy ma'nosida ma'lumotlar bazasidir. Ya'ni, bu ma'lumotlar to'plami. Ko'p jihatdan, bu uni boshqa fayllardan farq qilmaydi - axir barcha fayllarda biron-bir ma'lumot mavjud. "Ma'lumotlar bazasi" formati sifatida XML ba'zi afzalliklarga ega. Masalan, u o'zini o'zi tavsiflaydi (bu belgilar semantikani emas, balki ma'lumotlarning tuzilishini va turlarini nomlarini tavsiflaydi), ko'chma (Unicode) va daraxt yoki grafik tuzilmalardagi ma'lumotlarni tavsiflashi mumkin. Bundan tashqari, ba'zi bir kamchiliklari bor. Masalan, matnni tahlil qilish va matnni konvertatsiya qilish tufayli u juda ko'p va ma'lumotlarga kirish sust.

XML va uning atrofidagi texnologiyalar "atamalar bazasini" atamaning bo'sh ma'nosida, ya'ni ma'lumotlar bazasini boshqarish tizimida (MBBT) tashkil etadimi, degan savol yanada foydali bo'ladi. Boshqa tomondan, XML ma'lumotlar bazalarida topilgan ko'p narsalarni taqdim etadi: saqlash (XML hujjatlari),

sxemalar (DTD, XML sxemalari, RELAX NG va boshqalar), so'rovlar tillari (XQuery, XPath, XQL, XML-QL, QUILT) va boshqalar), dasturiy interfeyslar (SAX, DOM, JDOM) va boshqalar. Boshqa tomonida, u haqiqiy ma'lumotlar bazalarida mavjud bo'lgan ko'p narsalarga ega emas: samarali saqlash, indekslar, xavfsizlik, tranzaktsiyalar va ma'lumotlarning yaxlitligi, ko'p foydalanuvchiga kirish, triggerlar, bir nechta hujjatlar bo'yicha so'rovlar va boshqalar.

Shunday qilib, oz miqdordagi ma'lumotlar, oz sonli foydalanuvchilar va kam ishlash talablariga ega bo'lgan muhitda XML hujjati yoki hujjatlaridan ma'lumotlar bazasi sifatida foydalanish mumkin bo'lsa-da, bu ko'p foydalanuvchilarga ega bo'lgan ko'pgina ishlab chiqarish muhitida muvaffaqiyatsiz bo'ladi.

XML mos keladigan "ma'lumotlar bazasi" turining yaxshi namunasi .ini faylidir, Ya'ni dastur konfiguratsiyasi ma'lumotlarini o'z ichiga olgan fayldir. Kichik XML tilini ixtiro qilish va ushbu tilni izohlash uchun SAX dasturini yozish, vergul bilan ajratilgan fayllar uchun sintaktik yozuv yozishdan ko'ra osonroqdir. Bunga qo'shimcha ravishda, XML sizga ichkariga kirishga imkon beradi, buni vergul bilan ajratilgan fayllarda qilish qiyinroq narsa.

XML ma'lumotlar bazasi sifatida mos bo'lishi mumkin bo'lgan yanada murakkab ma'lumot to'plamlariga misol bo'la oladi. Masalan: shaxsiy aloqa ro'yxatlari (ismlar, telefon raqamlari, manzillar va boshqalar), brauzer xatcho'plari va Napster yordamida o'g'irlangan MP3-larning tavsiflari. Ammo, dBASE va Access kabi ma'lumotlar bazalarining arzonligi va ulardan foydalanish qulayligi hisobga olinsa, hatto XML holatlarida ham XML ma'lumotlar bazasi sifatida foydalanish uchun asos yo'q. XML-ning yagona haqiqiy afzalligi shundan iboratki, ma'lumotlar ko'chma bo'ladi va bu XML kabi ma'lumotlar bazalarini seriyalashtirish vositalarining keng ko'lamda mavjudligi sababli ko'rinmaydi.

XML ning vazifasi?

XML va ma'lumotlar bazalari haqida o'ylashni boshlaganingizda, o'zingizdan so'rashingiz kerak bo'lgan birinchi savol, nima uchun avvalo ma'lumotlar bazasidan foydalanishni xohlaysiz. Eskirgan ma'lumotingiz

bormi? Veb-sahifalaringizni saqlash uchun joy qidiryapsizmi? Bormi bir elektron tijorat dastur tomonidan ishlatilayotgan bazasi qaysi XML ma'lumot transport sifatida ishlatiladi? Ushbu savollarga berilgan javoblar sizning ma'lumotlar bazangizni va o'rta dasturlarni (agar mavjud bo'lsa) tanlashingizga, shuningdek, ushbu ma'lumotlar bazasidan qanday foydalanishingizga ta'sir qiladi. Masalan, sizda ma'lumot uzatish sifatida XML-dan foydalanadigan elektron tijorat dasturi mavjud deylik. Ma'lumotlaringiz yuqori darajadagi muntazam tuzilishga ega bo'lishi va XML bo'lmagan dasturlar tomonidan ishlatilishi yaxshi xisoblanadi. Bundan tashqari, XML tomonidan ishlatiladigan ob'ektlar va kodlashlar kabi narsalar, ehtimol siz uchun ahamiyatli emas - axir siz ma'lumotni XML qanday saqlanishiga emas, balki ma'lumotga qiziqasiz. Bunday holda, ehtimol XML va ma'lumotlar bazasi o'rtasida ma'lumotlarni uzatish uchun sizga aloqador ma'lumotlar bazasi va dasturiy ta'minot kerak bo'ladi. Agar sizning ilovalaringiz ob'ektga yo'naltirilgan bo'lsa, siz hatto ushbu ob'ektlarni ma'lumotlar bazasida saqlashi yoki ularni XML sifatida tartiblashi mumkin bo'lgan tizimni xohlashingiz mumkin.

Boshqa tomondan, sizda nasarga yo'naltirilgan XMLdan iborat veb-saytingiz bor deylik. Siz nafaqat saytni boshqarishni xohlamaysiz, balki foydalanuvchilarga uning tarkibini qidirish usulini taqdim etmoqchisiz. Hujjatlaringiz odatdagi tuzilishga ega emas va ob'ektlardan foydalanish kabi narsalar, ehtimol siz uchun muhimdir, chunki ular hujjatlaringiz tuzilishining asosiy qismi hisoblanadi. Bunday holda, siz mahalliy XML ma'lumotlar bazasi yoki tarkibni boshqarish tizimi kabi mahsulotni xohlashingiz mumkin. Bu sizga hujjatning jismoniy tuzilishini saqlashga, hujjat darajasidagi tranzaktsiyalarni qo'llab-quvvatlashga va XML so'rovlari tilida so'rovlarni bajarishga imkon beradi.

Hujjatlarga nisbatan ma'lumotlar

Ehtimol ma'lumotlar bazasini tanlashda eng muhim omil bu *ma'lumotlar* yoki *hujjatlarni* saqlash uchun *ma'lumotlar* bazasidan foydalanayotganligingizdir. Masalan, XML oddiygina ma'lumotlar bazasi va ilova

o'rtasida ma'lumot uzatish sifatida foydalaniladimi? Yoki XHTML va DocBook hujjatlarida bo'lgani kabi, undan foydalanish yaxlitmi? Odatda bu niyat masalasidir, lekin bu juda muhim, chunki barcha *ma'lumotlarga asoslangan hujjatlar* , barcha *hujjat markazidagi hujjatlar* kabi bir qator xarakteristikalarni taqsimlaydi va bu ma'lumotlar bazasida XML qanday saqlanishiga ta'sir qiladi. Keyingi ikkita bo'limda ushbu xususiyatlar ko'rib chiqiladi.

Ma'lumotga oid hujjatlar

Ma'lumotlar markazlashtiruvchi hujjatlar - bu XML ma'lumotlarini uzatish sifatida ishlatadigan hujjatlardir. Ular mashina iste'moli uchun mo'ljallangan va XML umuman ishlatilmasligi odatda ortiqcha. Ya'ni, dastur yoki ma'lumotlar bazasi uchun ma'lumot uzoq vaqt davomida XML hujjatida saqlanishi muhim emas. Ma'lumotga oid hujjatlarga misollar: savdo buyurtmalari, parvozlar jadvali, ilmiy ma'lumotlar va birja narxlari.

Masalan, quyidagi buyurtma hujjatlari ma'lumotlarga asoslangan:

```
<SalesOrder SONumber = "12345">
  <Customer CustNumber = "543">
    <CustName> ABC Industries </CustName>
    <Street> 123 Asosiy ko'chasi < / ko'chasi>
    <Shahar> Chikago </City>
    <State> IL </State>
    <PostCode> 60609 </PostCode>
  </Mijoz>
  <OrderDate> 981215 </OrderDate>
  <ElementNumber = "1 ">
    <Part PartNumber = "123">
      <Izoh>
        <p> <b> Turkiya kaliti: </b> <br />
        Zanglamaydigan po'latdan yasalgan buyumlar,
        umr bo'yi kafolat. </p>
      </sign>>
      <Narx> 9.95 </Price>
    </part>
    <Santity> 10 </Quantity>
  </Item>
  <ElementNumber = "2">
    <Part PartNumber = "456">
```

```
        <Izoh>
          <p> <b> to'ldiruvchi ajratuvchi: <b> <br />
          Alyuminiy, bir yillik kafolati . < /p>
        </sign>>
        <Narx> 13.27 </Price>
      </part>
      <Qadriyat> 5 </Quantity>
    </Item>
  </SalesOrder>
```

Masalan, parvozni tavsiflovchi quyidagi hujjatni ko'rib chiqing:

```
  <FlightInfo>
    <Airline> ABC Airways </Airline> <Count> uchta </Count> ta'minlaydi
    har kuni <Origin> Dallas </Origin> dan doimiy reyslar
    <Belgilangan joy> Fort-Uert </Destination>. Jo'nash vaqti
    <Departure> 09:15 </Departure>, <Departure> 11:15 </Parture>,
    va <Departure> 13:15 </Departure>. Kelish vaqtlari bir necha daqiqadan so'ng.
  </FlightInfo>
```

Buni quyidagi XML hujjati va oddiy uslublar jadvalidan qurish mumkin:

```
  <Yoritgichlar>
    <Airline> ABC Airways </Airline>
    <Origin> Dallas </Origin>
    <Belgilangan joy> Fort-Uert </Destination>
    <Yorug'>
      <Departure> 09:15 </Departure>
      <Obriv> 09:16 </Ariv>>
    </Flight>
    <Yorug'>
      <Departure> 11:15 </parture>
      <Obriv> 11:16 </Ariv>>
    </Flight>
    <Yorug'>
      <Departure> 13:15 </parture>
      <Obriv> 13:16 </Ariv>>
    </Flight>
  </live>
```

Ma'lumotlarni saqlash va olish

Ma'lumotni XML va ma'lumotlar bazasi o'rtasida uzatish uchun, XML hujjat sxemasini (DTD, XML Schemas, RELAX NG va boshqalar) ma'lumotlar bazasi sxemasiga xaritalash kerak. Keyin ma'lumotlar uzatish dasturi ushbu xaritaning tepasida qurilgan. Dastur XML so'rovi tilidan (masalan, XPath, XQuery yoki

mulkiy til) foydalanishi yoki shunchaki xaritaga muvofiq ma'lumotlarni uzatishi mumkin (SELECT * FROM jadvalining XML ekvivalenti).

Ikkinchi holda, hujjatning tuzilishi xaritada kutilgan tuzilishga to'liq mos kelishi kerak. Ko'pincha bunday emasligi sababli, ushbu strategiyani ishlatadigan mahsulotlar ko'pincha XSLT-dan foydalaniladi. Ya'ni, ma'lumotlar bazasiga ma'lumotlarni uzatishdan oldin, hujjat avval xaritada kutilgan tuzilishga o'tkaziladi;

So'rovlar tillari

Ko'pgina mahsulotlar to'g'ridan-to'g'ri ma'lumotni ular tuzilgan modelga muvofiq uzatadilar. XML hujjatining tuzilishi ko'pincha ma'lumotlar bazasining tuzilishidan farq qiladiganligi sababli, ushbu mahsulotlar ko'pincha XSLT-ni o'z ichiga oladi yoki ishlatiladi. Bu foydalanuvchilarga hujjatlarni ma'lumotlar bazasiga o'tkazmasdan oldin, shuningdek, teskari tartibda ma'lumotlarni model tomonidan talab qilingan tuzilishga o'zgartirishga imkon beradi. XSLTni qayta ishlash qimmatga tushishi mumkinligi sababli, ba'zi mahsulotlar cheklangan miqdordagi o'zgarishlarni o'zlarining xaritalariga qo'shib yuboradi.

Ushbu muammoning uzoq muddatli echimi XML-ni qaytaradigan so'rovlar tillarini amalga oshirishdir. Hozirgi vaqtda, ushbu tillarning aksariyati andozalarga kiritilgan SELECT ko'rsatmalariga tayanadi. XQuery va SQL / XML tugallanganda, bu holat o'zgarishi kutilmoqda, chunki ma'lumotlar bazasining yirik sotuvchilari allaqachon amalga oshirish ustida ishlamoqda. Afsuski, deyarli barcha XML so'rovlari tillari (shu jumladan XQuery 1.0 va SQL / XML ning dastlabki versiyasi) faqat o'qish uchun mo'ljallangan, shuning uchun yaqin kelajakda ma'lumotlarni kiritish, yangilash va yo'q qilish uchun turli xil vositalar kerak bo'ladi. (Uzoq muddatda XQuery va SQL / XML bu imkoniyatlarni qo'shadi.)

Shablon asosida so'rovlar tillari

Relativ ma'lumotlar bazasidan XML-ni qaytaradigan eng keng tarqalgan so'rov tillari shablonga asoslangan. Ushbu tillarda hujjat va ma'lumotlar bazasi o'rtasida oldindan aniqlangan xarita mavjud emas. Buning o'rniga SELECT ko'rsatmalari shablonga joylashtirilgan va natijalar ma'lumotlarni uzatish dasturi

tomonidan qayta ishlangan. Masalan, quyidagi shablon (biron bir real mahsulot tomonidan ishlatilmaydi) natijalarni qaerga joylashtirish kerakligini aniqlash uchun SELECT va $ ustun nomidagi qiymatlarni qo'shish uchun <SelectStmt> elementlaridan foydalanadi:

```
<? xml version = "1.0"?>
<FlightInfo>
  <Kirish> Quyidagi reyslar bor mavjud: </ Kirish>
  <SelectStmt> SELECT aviakompaniyasi, FltNumber,
        Jo'nash, FROM reyslariga etib borish </SelectStmt>
    <Yorug'>
      <Airline> $ Aviakompaniya </Airline>
      <FltNumber> $ FltNumber </FltNumber>
      <Depart> $ jo'nash </Depart>
      <Kelish> $ Kelish </Arive>
    </Flight>
  <Xulosa> Biz bu biri sizning ehtiyojlarini </ Xulosa> javob umid
</FlightInfo>
```

Bunday shablonni qayta ishlash natijasi quyidagicha bo'lishi mumkin:

```
<? xml version = "1.0"?>
<FlightInfo>
  <Kirish> Quyidagi reyslar bor mavjud: </ Kirish>
  <Yoritgichlar>
    <Yorug'>
      <Airline> ACME </Airline>
      <FltNumber> 123 </FltNumber>
      <Depart> 2017 yil 12-dekabr 13:43 </Depart>
      <Avval> 2018 yil 13-dekabr 01:21 </Arive>

    </Flight>

  </live>
  <Xulosa> Biz bu biri sizning ehtiyojlarini javob umid . < / Xulosa>
</FlightInfo>
```

Andoza asosidagi so'rovlar tillari juda moslashuvchan bo'lishi mumkin. Xususiyatlar to'plami mahsulotdan mahsulotga farq qilishi mumkin, ammo ba'zi bir tez-tez uchraydigan xususiyatlar quyidagilar:

- Natija to'plamining qiymatlarini chiqish hujjatining har qanday joyiga, shu jumladan keyingi SELECT ko'rsatmalarida parametrlar sifatida joylashtirish imkoniyati.

- Dasturlash uchun konstruktsiyalar, masalan looplar va if operatorlari.
- O'zgaruvchilar va funktsiya ta'riflari.
- HTTP parametrlari orqali SELECT ko'rsatmalarini parametrizatsiya qilish.

Shablonlarga asoslangan so'rovlar tillari deyarli ma'lumotni nisbiy ma'lumotlar bazasidan XML hujjatlariga o'tkazish uchun ishlatiladi. Garchi shablonga asoslangan so'rov tillaridan foydalanadigan ba'zi mahsulotlar ma'lumotlarni XML hujjatlaridan aloqador ma'lumotlar bazalariga uzatishi mumkin bo'lsa-da, ular shu maqsadda to'liq shablon tilidan foydalanmaydilar. Buning o'rniga, yuqorida aytib o'tilganidek, jadvalga asoslangan xaritadan foydalanadilar.

XML so'rovlari tillari

Shablonlarga asoslangan so'rovlar tillari va SQL-ga asoslangan so'rovlar tillaridan farqli o'laroq, ular faqat nisbiy ma'lumotlar bazalarida ishlatilishi mumkin, XML so'rovlari tillari har qanday XML uchun ishlatilishi mumkin. Bularni nisbiy ma'lumotlar bazalarida ishlatish uchun ma'lumotlar bazasidagi ma'lumotlar XML sifatida modellashtirilgan bo'lishi kerak va shu bilan virtual XML hujjatlari bo'yicha so'rovlarni amalga oshirishga imkon beradi.

XQuery yordamida jadval asosida yoki ob'ektga nisbatan xaritalashdan foydalanish mumkin. Agar jadvalga asoslangan xarita ishlatilsa, har bir jadval alohida hujjat sifatida ko'rib chiqiladi va SQL-da bo'lgani kabi so'rovning o'zida jadvallar (hujjatlar) o'rtasida birlashtiriladi. Ob'ektga nisbatan xaritalash ishlatilgan bo'lsa, jadvallar ierarxiyalari bitta hujjat sifatida ko'rib chiqiladi va qo'shilishlar xaritalashda ko'rsatiladi. Ko'pgina amaliy dasturlarda jadvalga asoslangan xaritalar relyatsion ma'lumotlar bazalariga nisbatan qo'llaniladi, chunki ularni amalga oshirish sodda va SQL foydalanuvchilariga tanishroq ko'rinadi.

XPath yordamida bir nechta jadvallar bo'yicha so'rovlarni bajarish uchun ob'ektga nisbatan xaritalashdan foydalanish kerak. Buning sababi, XPath hujjatlarni birlashtirishni qo'llab-quvvatlamaydi. Shunday qilib, agar jadvalga asoslangan xarita ishlatilgan bo'lsa, bir vaqtning o'zida faqat bitta jadvalga murojaat qilish mumkin edi.

Native XML ma'lumotlar bazasida ma'lumotlarni saqlash

Ma'lumotni XML ma'lumotlar bazasida saqlash mumkin. Buning bir necha sabablari bor. Bularning birinchisi, sizning ma'lumotlaringiz yarim tuzilgan bo'lsa. Ya'ni, u muntazam tuzilishga ega, ammo bu struktura o'zgaruvchan bo'lib, uni nisbiy ma'lumotlar bazasida xaritalash natijasida nol qiymatlarga ega bo'lgan ustunlar sonining ko'payishi (bo'sh joyni isrof qiladigan) yoki ko'p sonli jadvallar (samarasiz) bo'ladi. Yarim tuzilgan ma'lumotlar ob'ektga yo'naltirilgan va ierarxik ma'lumotlar bazalarida saqlanishi mumkin bo'lsa ham, siz uni XML hujjati shaklida mahalliy XML ma'lumotlar bazasida saqlashni tanlashingiz mumkin.

Ma'lumotni XML ma'lumotlar bazasida saqlashning ikkinchi sababi bu izlash tezligi. Tarkibiy XML ma'lumotlar bazasi ma'lumotlarning qanday saqlanishiga qarab, ma'lumotlarga nisbatan ma'lumotlarga nisbatan tezroq olinishi mumkin. Buning sababi shundaki, mahalliy XML ma'lumotlar bazalari tomonidan ishlatiladigan ba'zi saqlash strategiyalari butun hujjatlarni jismonan birga saqlaydi yoki hujjat qismlari o'rtasida jismoniy (balki mantiqiy) ko'rsatkichlarni ishlatadi. Bu hujjatlarni birikmalarsiz yoki jismoniy birikmalar bilan olish imkonini beradi, ikkalasi ham relyatsion ma'lumotlar bazalarida ishlatiladigan mantiqiy birikmalarga qaraganda tezroq. Masalan: Relatsion ma'lumotlar bazasida bu to'rtta jadvalda - SalesOrders, Elements, Customers and Parts - saqlanishi kerak va hujjatni olish uchun ushbu jadvallarda birlashtirish kerak bo'ladi. To'liq XML ma'lumotlar bazasida butun hujjat diskdagi bitta joyda saqlanishi mumkin, shuning uchun uni yoki uning bir qismini olish ma'lumotlarni olish uchun bitta indeksni qidirishni va bitta o'qishni talab qiladi. Aloqador ma'lumotlar bazasi to'rtta indekslarni qidirishni talab qiladi va ma'lumotlarni olish uchun kamida to'rt marta o'qishni talab qiladi. Bunda yaqqol kamchilik shundaki, ortib borayotgan tezlikni faqat diskda saqlangan tartibda ma'lumotlarni olishda qo'llash mumkin. Agar siz ma'lumotlarning boshqacha ko'rinishini, masalan, xaridorlar ro'yxati va ularga nisbatan qo'llaniladigan savdo buyurtmalarini olishni istasangiz, unumdorlik bazaga nisbatan yomonroq bo'lishi mumkin. Shunday qilib, ma'lumotni mahalliy

XML ma'lumotlar bazasida ishlash sabablariga ko'ra saqlash faqat sizning arizangizda ma'lumotlarning bitta ko'rinishi ustunlik qilgan taqdirdagina qo'llaniladi.

Ma'lumot turlari

XML ma'lumotlarning har qanday ma'nosida ma'lumot turlarini qo'llab-quvvatlamaydi. Tayyorlanmagan ob'ektlar bundan mustasno, XML hujjatidagi barcha ma'lumotlar matndir, hatto boshqa ma'lumotlar turini, masalan sana yoki butun sonni. Odatda, ma'lumotlarni uzatish dasturi ma'lumotni matndan (XML hujjatidagi) boshqa turlarga (ma'lumotlar bazasida) o'zgartiradi va aksincha.

Dastur qanday konversiyani amalga oshirishni mahsulotga xosligini qanday aniqlaydi, ammo ikkita usul keng tarqalgan. Birinchi usul shundaki, dasturiy ta'minot ma'lumotlar turini ma'lumotlar bazasi sxemasidan aniqlaydi, chunki bu har doim ish vaqtida mavjud. (XML sxemasi ish vaqtida majburiy bo'lishi shart emas va hatto mavjud bo'lmasligi ham mumkin.) Ikkinchi usul, foydalanuvchi xaritada ma'lumot olish kabi ma'lumotlar turini aniq etkazib beradi. Bu foydalanuvchi tomonidan yozilishi yoki ma'lumotlar bazasi yoki XML sxemasidan avtomatik ravishda yaratilishi mumkin. Avtomatik ravishda yaratilganda ma'lumotlar turlarini ma'lumotlar bazasi sxemalaridan va ba'zi XML sxemalaridan (XML Schemas, RELAX NG) olish mumkin.

O'zgartirishlar bilan bog'liq yana bir muammo bu qanday format formatlari tan olinganligi (ma'lumotlarni XML-dan uzatishda) yoki yaratilishi mumkin (ma'lumotlarni XML-ga o'tkazishda). Ko'pgina hollarda, ma'lum bir ma'lumot turi uchun qo'llab-quvvatlanadigan matn formatlari soni cheklangan bo'lishi mumkin, masalan, bitta, maxsus format yoki berilgan JDBC drayveri tomonidan qo'llab-quvvatlanadigan formatlar uchun. Xurmo, ehtimol, muammolarga olib kelishi mumkin, chunki mumkin bo'lgan formatlar oralig'i juda katta. Turli xil xalqaro formatlarga ega raqamlar ham muammolarga olib kelishi mumkin.

Ma'lumotni saqlash

Ba'zida aralash ma'lumotlarsiz elementlarni ma'lumotlar bazasida keyingi tahlil qilinmasdan saqlash foydalidir. Bu tugallangandan so'ng, markirovka belgilar bilan belgilanadi. Biroq, ajratish uchun ishlatilmaydigan belgilarni qanday saqlash kerakligi bilan bog'liq muammo yuzaga keladi. XML dabular lt, gt, amp, quot va apos ob'ektlari yordamida saqlanadi. Buni ma'lumotlar bazasida ham amalga oshirish mumkin. Masalan, quyidagi tavsif:

< tavsif >

 Noto'g'ri misol: & lt; foo / & gt;

</description>

ma'lumotlar bazasida quyidagicha saqlanishi mumkin :

 Noto'g'ri misol: & lt; foo / & gt;

Bu bilan muammo shundaki, SQL kabi XML bo'lmagan so'rovlar tillari, ustunlik qiymatlarini belgilash va ob'ektni ishlatish uchun tekshirmaydi va ularni mos ravishda izohlaydi. Shunday qilib, agar siz "<foo />" satrini SQL bilan qidirishni xohlasangiz, aslida "& lt ; foo / & gt;" qatorini qidirishingiz kerakligini bilishingiz kerak .

Boshqa tomondan, agar mantiqiy ob'ektlarning havolalari kengaytirilsa, ma'lumotlar uzatish dasturi mantiqiy ob'ektni ishlatishni ajratib ko'rsatolmaydi. Masalan, yuqoridagi tavsif bazada quyidagicha saqlanadigan bo'lsa, dasturiy ta'minot , va <foo /> belgilar yoki matn ekanligini aniqlay olmaydi.

 Noto'g'ri misol: <foo />

Ushbu muammoni uzoq muddatli hal qilish - bu XML bilan ishlaydigan ma'lumotlar bazasi bo'lib, unda haqiqiy belgilash faqat belgilashga o'xshash narsalardan farq qiladi. XML bo'lmagan dasturlar XML bilan ishlayotganda, ehtimol har doim ham muammolar bo'ladi.

Relational Schemas va Vitsa Versiyalaridan XML sxemalarini yaratish

XML va ma'lumotlar bazasi o'rtasida ma'lumotlarni uzatishda keng tarqalgan savol bu ma'lumotlar bazasi sxemasidan XML sxemasini yaratish va aksincha. Buni qanday qilish kerakligini tushuntirishdan oldin, bu dizayn vaqtini talab qiladigan ish ekanligini ta'kidlash kerak. Buning sababi shundaki , ko'p ma'lumotlarga asoslangan dasturlar va deyarli barcha vertikal dasturlar ma'lum XML sxemalari va ma'lumotlar bazasi sxemalari bilan ishlaydi. Shunday qilib, ular ish vaqtida sxemalarni yaratishi shart emas. Bundan tashqari, quyida ko'rib o'tilganidek, sxemalarni yaratish protseduralari mukammal emas. Tasodifiy XML hujjatlarini ma'lumotlar bazasida saqlashi kerak bo'lgan dasturlar, ehtimol ish vaqtida sxemalar yaratish o'rniga mahalliy XML ma'lumotlar bazasidan foydalanishi kerak. O'zaro aloqador sxemalarni XML sxemalaridan va aksincha, yaratishning eng oson usuli - bu bir qator qo'shimcha funktsiyalarga ega bo'lgan ob'ektga nisbatan xaritalash orqali yo'lni kodlash. Ob'ektga yo'naltirilgan ma'lumotlar bazasidan foydalanish uchun shunga o'xshash protseduralar mavjud.

XML sxemasidan o'zaro bog'liqlik sxemasini yaratish uchun:

> Har bir *murakkab element turi* uchun jadval va boshlang'ich kalit ustunlarini yarating.
> Aralash tarkibga ega har bir element turi uchun ota-ona jadvalining boshlang'ich kaliti orqali ota-ona jadvaliga ulangan PCDATA-ni saqlash uchun alohida jadval yarating.
> Ushbu element turining har bir qiymatli atributi uchun va bitta oddiy bola elementi uchun ushbu jadvalda ustun yarating. Agar XML sxemasida ma'lumotlar turi ma'lumotlari mavjud bo'lsa, unda ustun turidagi ma'lumot turlarini tegishli turga o'rnating. Aks holda, ma'lumotlar turini CLOB yoki VARCHAR (255) kabi oldindan belgilangan turga o'rnating. Agar bola elementi yoki atributi ixtiyoriy bo'lsa, ustunni bekor qiling.

O'zaro bog'liqlik sxemasidan XML sxemasini yaratish uchun:

> Har bir jadval uchun element turini yarating.

- Jadvaldagi har bir ma'lumot (kalit bo'lmagan) ustunlar uchun, shuningdek asosiy kalit ustunlari (elementlari) uchun element turiga atribut yoki tarkibiy modelga PCDATA-faqat bolalar elementini qo'shing.
- Asosiy kalit eksport qilinadigan har bir jadval uchun tarkibiy qismga kichik element qo'shing va jadvalni rekursiv ravishda qayta ishlang.
- Har bir tashqi kalit uchun tarkibiy modelga bolalar elementini qo'shing va tashqi kalit jadvalini rekursiv ravishda qayta ishlang.

Ushbu protseduralarning bir qator kamchiliklari mavjud. Ularning aksariyati qo'l bilan tuzatilishi oson, masalan, nomlarning to'qnashuvi va ustunlar ma'lumotlari turlari va uzunliklari. (DTD-larda ma'lumotlar turi ma'lumotlari mavjud emas, shuning uchun ma'lumotlar bazasida qanday turdagi ma'lumotlardan foydalanish kerakligini oldindan aytib bo'lmaydi. Esda tutingki, ma'lumotlar turlarini va ularning uzunligini XML sxemasi hujjatidan bashorat qilish mumkin.)

Yana jiddiy muammo shundaki, XML hujjatida foydalaniladigan ma'lumotlar "modeli" ma'lumotlar bazasida ma'lumotlarni saqlash uchun eng samarali modelga qaraganda ko'pincha farq qiladi (va odatda murakkabroq). Masalan, XML ning quyidagi parchasini ko'rib chiqing:

```
<Iste'molchi>
  <Ism> ABC Industries </Name>
  <Manzil>
    <Street> 123 Asosiy ko'chasi < / ko'chasi>
    <Shahar> Fooville </City>
    <State> CA </State>
    <Mamlakat> AQSh </devropa>
    <PostCode> 95041 </PostCode>
  </Address>
</Mijoz>
```

O'zaro bog'liqlikni XML sxemasidan yaratish tartibi bu yerda ikkita jadvalni yaratadi: biri mijozlar uchun, ikkinchisi manzillar uchun. Biroq, aksariyat hollarda manzilni alohida jadvalda emas, balki mijozlar jadvalida saqlash ma'qulroq.

<Manzil> elementi *o'rash elementining* yaxshi namunasidir . Sharf elementlari odatda ikkita sababga ko'ra ishlatiladi. Birinchidan, ular hujjatni tushunishni

osonlashtiradigan qo'shimcha tuzilmani taqdim etishga yordam beradi. Ikkinchidan, ular odatda ma'lumotlarni yozish shakli sifatida ishlatiladi. Masalan, <Manzil> elementini qayerda bo'lishidan qat'iy nazar barcha manzillarni manzil ob'ektlariga o'zgartiradigan odatiy holatga o'tkazish mumkin.

Qoplash elementlari XML-da foydali bo'lsa-da, ular odatda ma'lumotlar bazasida joylashtirilganda qo'shimcha tuzilish shaklida muammolarga olib keladi. Shu sababli, aloqador sxemani yaratmasdan oldin ular odatda XML sxemasidan chiqarib tashlanishi kerak. XML sxemasini doimiy ravishda o'zgartirishi yoki o'zgartirishi dargumon bo'lganligi sababli, amaldagi hujjatlar va ma'lumotlar uzatish dasturi tomonidan kutilgan hujjatlar o'rtasida nomuvofiqlik yuzaga keladi, chunki o'rash elementlari xaritaga kiritilmagan. Buni hujjatlarni ish vaqtida o'zgartirish orqali hal qilish mumkin, masalan XSLT: o'rash elementlari ma'lumotlar bazasiga o'tkazilishidan oldin yo'q qilinadi va ma'lumotlar bazasidan ma'lumotlarni uzatgandan so'ng kiritiladi.

Ushbu kamchiliklarga qaramay, yuqoridagi protseduralar XML sxemalarini relyatsion sxemadan va aksincha, ayniqsa katta tizimlardan yaratish uchun foydali boshlang'ich nuqtani beradi.

Hujjatlar to'plami

Ko'pgina mahalliy XML ma'lumotlar bazalari to'plam tushunchasini qo'llab-quvvatlaydi. Bu relyatsion ma'lumotlar bazasidagi jadvalga yoki fayl tizimidagi katalogga o'xshash rol o'ynaydi. Masalan, siz buyurtmalarni saqlash uchun mahalliy XML ma'lumotlar bazasidan foydalanayotgansiz deylik. Bunday holda, savdo buyurtmalari bo'yicha so'rovlar ushbu to'plamdagi hujjatlar bilan cheklanishi uchun siz savdo buyurtmalari to'plamini belgilashni xohlashingiz mumkin.

Boshqa bir misol sifatida, siz kompaniyaning barcha mahsulotlari uchun qo'llanmalarni mahalliy XML ma'lumotlar bazasida saqlamoqdasiz deylik. Bunday holda siz to'plamlarning ierarxiyasini aniqlashni xohlashingiz mumkin. Masalan, sizda har bir mahsulot uchun to'plam bo'lishi mumkin va ushbu to'plam ichida har bir qo'llanmadagi barcha boblar uchun to'plamlar bo'lishi mumkin.

To'plamlarning joylashtirilishi ma'lumotlar bazasiga bog'liq.

So'rovlar tillari

Deyarli barcha mahalliy XML ma'lumotlar bazalari bir yoki bir nechta so'rovlar tillarini qo'llab-quvvatlaydi. Ularning eng mashhurlari XPath (bir nechta hujjatlar ustida so'rovlar uchun kengaytmalar bilan) va XQuery, ammo ko'p sonli mulkiy so'rovlar tillari ham qo'llab-quvvatlanadi. O'zingizning XML ma'lumotlar bazangizni ko'rib chiqayotganda, ehtimol so'rovlar tili sizning ehtiyojlaringizga mos kelishini tekshirishingiz kerak, chunki bu to'liq matnli qidiruvlardan tortib bir nechta hujjatlardan parchalarni qayta to'plash ehtiyojlariga qadar bo'lishi mumkin. Kelajakda ko'pgina mahalliy XML ma'lumotlar bazalari W3C-dan XQuery-ni qo'llab-quvvatlaydi.

Ilova dasturlash interfeysi (API)

Deyarli barcha mahalliy XML ma'lumotlar bazalari dasturiy API-larni taklif qiladi. Bular odatda ODBC-ga o'xshash interfeys shaklida bo'lib, ma'lumotlar bazasiga ulanish, metadata o'rganish, so'rovlarni bajarish va natijalarni olish usullari mavjud. Odatda natijalar XML satr, DOM daraxti yoki SAX tahlil qiluvchi yoki XMLReader sifatida qaytarilgan hujjat sifatida qaytariladi. Agar so'rovlar bir nechta hujjatlarni qaytarishi mumkin bo'lsa, natijalar to'plami orqali iteratsiya usullari ham mavjud. Ko'pgina mahalliy XML ma'lumotlar bazalari mulkiy API-larni taklif qilishsa-da, ikkita sotuvchi neytral XML ma'lumotlar bazasi API-lari ishlab chiqilgan.

- XML : DB API dan XML: DB.org til-neytral dasturlash, uning so'rovlar tili sifatida XPath'i foydalanadi va XQuery qo'llab-quvvatlash uchun kengaytirilgan qilinmoqda hisoblanadi. Bu bir qator mahalliy XML ma'lumotlar bazalari tomonidan amalga oshirilgan va mahalliy bo'lmagan ma'lumotlar bazalarida ham bajarilgan bo'lishi mumkin.
- JSR 225: Java uchun XQuery API (XQJ) JDBC-ga asoslangan va so'rov tili sifatida XQuery-dan foydalanadi. Ushbu dastur Sun Java Jamiyat jarayoni (JCP) orqali ishlab chiqilmoqda va qoralama versiyasi mavjud. Ko'pgina

XML ma'lumotlar bazalari so'rovlarni bajarish va HTTP orqali natijalarni qaytarish imkoniyatini taklif etadi.

Masofaviy ma'lumotlar: Ba'zi mahalliy XML ma'lumotlar bazalari ma'lumotlar bazasida saqlanadigan hujjatlardagi masofaviy ma'lumotlarni o'z ichiga olishi mumkin.

Odatda, bu ma'lumotlar ODBC, OLE DB yoki JDBC bilan aloqador ma'lumotlar bazasidan olinadi va jadvalga asoslangan xarita yoki ob'ektga nisbatan xaritalash yordamida modellashtiriladi. Ma'lumotlar jonli bo'ladimi-yo'qmi, Ya'ni XML ma'lumotlar bazasidagi yangilanishlar uzoqdagi ma'lumotlar bazasida aks etadimi - bu mahalliy XML ma'lumotlar bazasiga bog'liq. Oxir oqibat, ko'pgina mahalliy XML ma'lumotlar bazalari jonli masofaviy ma'lumotlarni qo'llab-quvvatlaydi.

Ko'rsatkichlar(Index): Barcha mahalliy XML ma'lumotlar bazalari so'rovlar tezligini oshirish uchun indekslarni qo'llab-quvvatlaydi. Bular indekslarning uch turi. *Qiymat indekslari* matn va atributlarning *ko'rsatkichlarini* indekslaydi va "Santa Cruz" qiymati bo'lgan barcha elementlarni yoki atributlarni toping "kabi savollarni hal qilish uchun ishlatiladi.

Tuzilmaviy indekslar elementlar va atributlarning joylashuvini *indekslaydi* va "Manzilning barcha elementlarini topish" kabi savollarni hal qilish uchun ishlatiladi. Qiymati va tarkibiy ko'rsatkichlari "Santa Cruz" bo'lgan shaharning barcha elementlarini toping "kabi savollarni hal qilish uchun birlashtirilgan. Va nihoyat, *to'liq matnli indekslar* indikatorlarni va atributlar qiymatlarini *indekslaydi* va "Santa Cruz" so'zlarini o'z ichiga olgan barcha hujjatlarni topish ", yoki tarkibiy indekslar bilan birgalikda" Barcha hujjatlarni toping "kabi savollarni hal qilish uchun ishlatiladi. unda manzil elementidagi "Santa Cruz" so'zlari mavjud. "

Ko'pgina mahalliy XML ma'lumotlar bazalari ham qiymat, ham tarkibiy indekslarni qo'llab-quvvatlaydi. Ba'zi mahalliy XML ma'lumotlar bazalari to'liq matnli indekslarni qo'llab-quvvatlaydi.

Nazorat savollari

1. XML ning vazifasi nima?
2. Ma'lumotlar markazlashtiruvchi hujjatlar
3. XML hujjatining tuzilishi ma'lumotlar bazasining tuzilishidan nimasi bilan farq qiladi?
4. Shablon asosida so'rovlar tillariga misol keltiring?
5. Tuzilmaviy indekslar nima?

Glossariy

Termin	O'zbek tilidagi sharhi	Ingliz tilidagi sharhi
SQL	Strukturalangan so'rovlar tili	Structured Query Language
cess	Microsoft Office RMBBT dasturi	The Microsoft Office RDBMS application
Append	Jadval oxirigacha ma'lumotlarni qo'shish	Adding data to the end of table
Accending order	Eng past va eng yuqori uchun sanada asoslangan matn sohasida alifbo tartibi	In order from lowest to highest. Also called alphabetical order, when a sort is based on a text field, and chronological, when a sort is based on a date field
Autonumber field	Yozishga qaraganda katta maydonga qo' shimcha ravishda avtomatik saqlash	A field that automatically stores a numeric value ,that is one greater than that in the last record added
Database	Tegishli ma'lumotlarni yig'ish	An organized collection of related data
Database schema	Ma'lumotlar bazasida jadvallar yacheykasiga ma'lumotlarning bayoni va ma'lumotlarni uzatishni tashkil etish	A description of the data, and the organization of the data, into tables in a relational database
Datasheet	Ma'lumotlar uchun satrlar ustunlar sohalarda va yozuvlar bilan tashkil etish	The data for a table organized with fields in columns and records in rows

Datasheet view	Satrlar ustunlar sohalarda va yozuvlar bilan, bir ma'lumot sahifasida bir stol asosiy tuzilishini ko'rish uchun ishlatiladi	Used to display the basic structure of a table in a datasheet, with fields in columns and records in rows
Entry	Jadval uchun ma'lumotlar	The data for a field
Field	Jadvaldagi maydonlarni belgilaydi	A column in a table. Used to store data
OLAP	Хакикий вактда маълумотларга аналитик ишлов бериш	On-Line Analytical Processing
OLTP	Хакикий вактда транзакцияларга ишлов бериш	On-Line Transaction Processing
Form	So'rovlar yordamida ma'lumotlarni ko'rishda ishlatiladi	A database object used for entering records into a table, and for viewing existing records
Long integer	Uzun butun toifa	A field size that indicates a whole number
Lookup field	Maydondagi ma'lumotlarni saqlaydi	A field that stores data; retrieved from a field in another table
Name	Jadval nomi bo'lib, soz orqali ifodalanadi	Word or words, used to describe the data stored in a field
Primary key	Birlamchi kalit hisoblanadi	A field in a table that is designated to contain unique data.
RDBMS	Relatsion ma'lumotlar bazasini boshqarish tizimi	(Relational Database Management System) A software application that

		contains tools to manage data, answer queries, create user-friendly forms for data entry, and generate printed reports.
Date/time field	Maydon sana yoki vaqtni saqlaydi	A field that stores a date or time
Decending order	Oliy maqsadidan eng past uchun	In order from highest to lowest
Design view	Jadvallar uchun maydon ta'riflar ko'rsatadi	The table view that shows the field definitions for a table
Yes/No field	ha / yo'q, to'g'ri / noto'g'ri, yoki / off vakillik qilish.	A field that is either selected or not selected to represent yes/no, true/false, or on/off.
ERP	Korxona resurslarini rejalashtirish	Enterprise Resource Planning
Record	Jadvaldagi maydonlarni uchun ma'lumotlar majmui	A set of data for fields in a table
Updating	Yozuvni o'zgartirish	Modifying a record
Table	Ma'lumotlar bazasi obyekti. Satr va ustunlar ichiga tashkil etilgan tegishli ma'lumotlarni saqlaydi	A database object that stores related data organized into rows and columns.
Text field	Jadvallarda belgilar (harflar, belgilar, so'zlar, harflar va raqamlar kombinatsiyasini) hisob talab qilmaydigan va	A field that stores characters (letters, symbols, words, a combination of letters and numbers) and numbers

	sonlar saqlaydi	that do not require calculations.
CRM	Мижозлар билан узаро муносабатларни бопщариш	Customer Relations Management
LAN	Локал хисоблаш тармоги	Local Area Network
MAN	Махаллий хисоблаш тармоги	Metropolitan Area Network
WAN	Худудий хисоблаш тармоги	Wide Area Network
ISO	Х,алкаро стандартлаштириш ташкилоти	International Organization for Standardization
WWW	Умумжахон ургамчак тури	World Wide Web
ASCII	Ахборот алмашишнинг Америка стандарти	American Standard Code for Information Interchange

Adabiyotlar ro'yxati

1. В.П. Базы данных. Книга 2 распределенные и удаленные базы данных: учебник.// Москва ИД «ФОРУМ» - ИНФРА-М. - 2018. - С 261.
2. Голицына О.Л. Базы данных: учеб. Пособие // - 4-е изд., перераб. И доп. - М.: ФОРУМ: ИНФРА-М, 2018. - 400 с.
3. Мартишин С.А. Базы данных. Практическое применение СУБД SQL -и NoSQL - типа для проектирования информационных систем: учеб. Пособие // - Москва: ИД «ФОРУМ» - ИНФРА-М, 2019, - 368 с.
4. Rahul Batra. SQL Primer An Accelerated introduction to SQL Basics.// Gurgaon, India. 2019. -P 194.

5. Поликов А.М. Безопасность Oracle глазами аудитория: нападение и защита. -Москва. 2017. -336 с.
6. Usmonov J.T., Xujaqulov T.A. Ma'lumotlar bazasini boshqarish tizimi// o'quv qo'llanma. - T. : Aloqachi, 2018. - 96 b.
7. Usmonov J. T., Xo'jaqulov T. A. Ma'lumotlar bazasini boshqarish tizimi fanidan laboratoriya ishlarini bajarish bo'yicha uslubiy ko'rsatma - T. : TATU, 2016. - 55 b.
8. Eric Redmond, Jim R. Wilson. A Guide to Modern Databases and the NoSQL MovementAQSH, 2015. - 347 c.
9. Elmasri, R., S. B. Navathe: Fundamentals of Database Systems (5th Ed.)// Addison Wesley, 2015. - 671 p.

Qo'shimcha adabiyotlar

1. Узбекистан Республикаси президентининг 2017 йил 7 февралдаги ПФ-4947-сон "Узбекистан Республикасини янада ривожлантириш буйича Хдракатлар стратегияси тугрисида"ги Фармони.
2. Мирзиёев Ш.М. Буюк келажагимизни мард ва олижаноб халкимиз билан бирга курамиз. Тошкент. «Узбекистан», НМИУ, 2017. - 488 б.
3. Fundamentals of database systems sixth edition. Ramez Elmasri. Department of Computer Science and Engineering The University of Texas at Arlington. 2011. - 261 c.
4. Введение в Oracle 10g. Перри Джеймс, Пост Джеральд. 697 стр 2013
5. Диго С.М. Базы данных Проектирование и использование. издательство "Финансы и статистика". 2005 г. - 592 с.
6. Конноли Т., Брегк К. Базы данных, проектирование, реализация и сопровождения, теория и практика, Университет Пейсли, Шотландия, изд. М.- СПБ.- Киев. 2003. - 264 с.

Internet saytlari

1. www.intiut.ru;