Sh.Sh.Allamova

MA’LUMOTLAR BAZASI

O’ZBEKISTON RESPUBLIKASI OLIY TA’LIM, FAN VA INNOVATSIYALAR VAZIRLIGI

TOSHKENT IQTISODIYOT VA PEDAGOGIKA INSTITUTI

Sh.Sh.Allamova

MA’LUMOTLAR BAZASI

O‘zbekiston Respublikasi Oliy ta’lim, fan va innovatsiyalar vazirligi tomonidan oliy ta’lim muassasalari talabalari uchun o‘quv qo‘llanma sifatida tavsiya etilgan.

Chirchiq-2025

UDK:
BBK:

Ma’lumotlar bazasi. (O‘quv qo‘llanma). –2025. – 237 b.

ISBN

O‘quv qo‘llanma ma’lumotlar bazasining maqsadi, vazifalari, asosiy tushunchalari, ma’lumot bazasining uch bosqichli arxitekturasi, ma’lumotlar bazasi modellari, ma’lumotlar bazasida munosabatlar, relyatsion algebra va relyatsion hisoblash elementlari, ma’lumotlar bazasini rejalashtirish, loyihalashlash, administratorlash, ma’lumotlar bazasini normallashtirish, SQLtili va SQL operatorlarini yozish, ma’lumotlar bazasiga murojaatni tashkil etishda ODBC va turli dasturlardan foydalanish, XML va ma’lumotlar bazasi bilan ishlash ko‘nikmalarini hosil qilishga qaratilgan.

O‘quv qo‘llanma 60610500 – Kompyuter injiniringi(Kompyuter injiniringi) yo‘nalishida tahsil olayotgan talabalar uchun mo‘ljallangan.

Taqrizchilar:

N.A.Gulbayev	Toshkent iqtisodiyot va pedagogika instituti, Axborot texnologiyalari va aniq fanlar kafedrasi t.f.n., dotsenti
M.A.Qo‘chqarov	Muhammad al-Xorazmiy nomidagi Toshkent axborot texnologiyalari universiteti, “Suniy intellekt” kafedrasi dosenti, PhD

KIRISH

Mamlakatimizda raqamli iqtisodiyotni faol rivojlantirish, barcha tarmoqlar va sohalarda, ayniqsa ta’lim sohasida zamonaviy axborot-kommunikatsiya texnologiyalarini keng joriy etish bo‘yicha kompleks chora-tadbirlar amalga oshirilmoqda. Xususan, elektron hukumat tizimini takomillashtirish, dasturiy mahsulotlar va axborot texnologiyalarining mahalliy bozorini yanada rivojlantirish, respublikaning barcha hududlarida IT-parklarni tashkil etish, shuningdek, sohani malakali kadrlar bilan ta’minlashni ko‘zda tutuvchi —Raqamli O‘zbekiston — 2030 strategiyasi loyihalarini amalga oshirish boshlangan.

Ma’lumotlar bazalari zamonaviy axborot texnologiyalarining muhim tarkibiy qismlaridan biridir va ular turli sohalarda ma’lumotlarni saqlash, boshqarish va ishlov berish uchun keng qo‘llaniladi. O‘quv qo‘llanmasi, ayniqsa, ma’lumotlar bazasi tizimlarini o‘rganishda talabalarga yordam berishni maqsad qilgan bo‘lib, ularning ma’lumotlar bazasi nazariyasi, dizayni va amaliy qo‘llanilishi bo‘yicha keng qamrovli tushunchalar berishga qaratilgan.

Ma’lumotlar bazalari turli formatlarda mavjud, masalan, relatsion ma’lumotlar bazalari (RDBMS), no-relatsion ma’lumotlar bazalari va XML yoki NoSQL kabi boshqa tipdagi tizimlar. Ular turli maqsadlar uchun qo‘llaniladi: kompaniyalar, ta’lim muassasalari, tibbiyot sohalari va hatto onlayn xizmatlar uchun ma’lumotlar saqlash, tartibga solish va tahlil qilishda asosiy vosita hisoblanadi.

Ushbu qo‘llanma ma’lumotlar bazasi tizimlarining nazariy asoslari va amaliy yondashuvlarini o‘rgatishga qaratilgan bo‘lib, u SQL tilidan foydalanish, ma’lumotlar bazasini normallashtirish va boshqarish hamda xavfsizligini ta’minlash kabi mavzularni o‘z ichiga oladi. Qo‘llanma nafaqat nazariy bilimlarni, balki real hayotdagi misollarni ham o‘z ichiga oladi, bu esa talabalar uchun ma’lumotlar bazalarini ishlab chiqish va samarali boshqarish bo‘yicha ko‘nikmalarni rivojlantirishga yordam beradi.

Mazkur o‘quv qo‘llanmada fan dasturida keltirilgan barcha mavzularning uzviy ketma-ketlikdagi batafsil bayoni, o‘quv kursi bo‘yicha test savollari, qisqartmalar va foydalanilgan adabiyotlar ro‘yxatini o‘z ichiga oladi.

Ushbu o‘quv qo‘llanmani yaratishda ayrim xato va kamchiliklar bo‘lishi mumkin, bu kamchiliklarni bartaraf etishda o‘z takliflari bilan o‘rtoqlashgan talabalarga minnaddorchilik bildiramiz. Barcha fikr va takliflarni mamnuniyat bilan qabul qilamiz.

1 BOB. Ma’lumotlar bazasining maqsadi, vazifalari va asosiy tushunchalari

Tayanch so‘zlar: *ma’lumotlar bazasi, ma’lumotlar bazasini boshqarish tizimi, birlamchi kalit, tashqi kalit, fayl, atribut, jadval.*

1.1. Ma’lumotlar bazasi haqida tushuncha

Ma’lumotlar bazalari va ma’lumotlar bazasi texnologiyasi kompyuterlardan foydalanuvchilar sonining ortishiga katta ta’sir ko‘rsatdi. Aytish joizki, ma’lumotlar bazalari kompyuterlar qo‘llaniladigan deyarli barcha sohalarda, jumladan biznes, elektron tijorat, ijtimoiy media, muhandislik, tibbiyot, genetika, huquq, ta’lim va kutubxonachilikda muhim rol o‘ynaydi. Ma’lumotlar bazasi so‘zi juda keng tarqalgan tushunchalardan biri bo‘lib, biz ma’lumotlar bazasiga quyidagicha umumiy ta’rif berishimiz mumkin: “Ma’lumotlar bazasi - bu o‘zaro bog‘liq ma’lumotlar yig‘indisidir”. Ma’lumotlar deganda biz yozilishi mumkin bo‘lgan va yashirin ma’noga ega bo‘lgan ma’lum faktlarni tushunamiz. Ya’ni ma’lumotlar faqat fikr yoki tushunchalar bo‘lishi mumkin emas, balki ularni qog‘ozga, kompyuterga yoki boshqa vositalarga yozib olish mumkinligini va ma’lumotlarning faqat raqam yoki so‘zlar emas, balki ular ma’lum bir kontekstda muhim ma’noga ega ekanligini ifodalaydi. Misol uchun, “25” raqami oddiygina bir raqam bo‘lishi mumkin, ammo u ma’lum bir kontekstda “yosh”, “daraja” yoki “mijoz raqami” bo‘lishi mumkin. Yoki siz bilgan odamlarning ismlari, telefon raqamlari va manzillar bo‘lishi mumkin.

Hozirgi kunda, ko‘pgina ma’lumotlar odatda mobil telefonlarda saqlanadi, ularda o‘zlariga xos oddiy ma’lumotlar bazasi dasturlari mavjud.

Mobil telefonlarda mavjud bo‘lgan oddiy ma’lumotlar bazasi dasturlari quyidagilarni o‘z ichiga oladi:

1. **SQLite**: Mobil qurilmalarda eng keng tarqalgan ma’lumotlar bazasi tizimlaridan biri. SQLite ko‘plab mobil ilovalarda, masalan, iOS va Android dasturlarida ma’lumotlarni saqlash uchun ishlatiladi.

2. **Realm**: Bu mobil qurilmalar uchun mo‘ljallangan ma’lumotlar bazasi. U yuqori tezlikda ishlaydi va ma’lumotlarni real vaqtda yangilash imkoniyatini taqdim etadi. Realm juda intuitiv interfeysga ega bo‘lib, dasturchilar uchun qulaydir.
3. **Firebase Realtime Database**: Google tomonidan taqdim etilgan bulut asosidagi ma’lumotlar bazasi. Ushbu dastur mobil ilovalarga real vaqtda ma’lumotlar almashish imkoniyatini beradi.
4. **Couchbase Lite**: Mobil qurilmalar uchun mo‘ljallangan NoSQL ma’lumotlar bazasi. U ko‘p platformali qo‘llab-quvvatlashni ta’minlaydi va offline holatda ishlash imkoniyatiga ega.
5. **Core Data**: Apple tomonidan ishlab chiqilgan, iOS va macOS ilovalari uchun ma’lumotlarni boshqarish uchun mo‘ljallangan ma’lumotlar bazasi tizimi. Bu dastur foydalanuvchilarga obyektga yo‘naltirilgan ma’lumotlarni boshqarish imkonini beradi.
6. **PouchDB**: Bu JavaScript bilan yozilgan, mobil qurilmalar va veb ilovalar uchun mo‘ljallangan NoSQL ma’lumotlar bazasi. U offline rejimda ishlay oladi va CouchDB bilan sinxronizatsiya qilinishi mumkin.

Bu dasturlar mobil telefonlarda ma’lumotlarni saqlash, boshqarish va ularga kirish uchun foydalaniladi, bu esa foydalanuvchilarga qulaylik va samaradorlikni ta’minlaydi.

Ma’lumotlar, shuningdek, Microsoft Access yoki Excel kabi shaxsiy kompyuter va dasturlardan foydalangan holda qattiq diskda saqlanishi mumkin.

Odatda, ma’lumotlar bazasi atamasidan umumiy foydalanish ko‘proq cheklangan. Ma’lumotlar bazasi quyidagi yashirin xususiyatlarga ega:

- Ma’lumotlar bazasi real dunyoning ba’zi jihatlarini ifodalaydi. Ya’ni ma’lumotlar bazasi muayyan predmetlar, hodisalar yoki vaziyatlar haqidagi ma’lumotlarni saqlash uchun ishlatiladi. O‘zgarishlar ma’lumotlar bazasida aks ettiriladi. Masalan, yangi ma’lumotlar qo‘shilishi, mavjud ma’lumotlarning

o‘zgartirilishi yoki o‘chirilishi ma’lumotlar bazasida yangilanadi va aks etadi. Bu, ma’lumotlar bazasining har doim haqiqiy holatni aks ettirishini ta’minlaydi.

- Ma’lumotlar bazasi - bu ma’lum bir mantiqiy tartibga ega bo‘lgan va o‘zida biror ma’noni saqlovchi ma’lumotlar to‘plamidir. Ya’ni Ma’lumotlar bazasi ichidagi ma’lumotlar tartiblangan va bir-biri bilan mantiqiy ravishda bog‘langan bo‘lishi kerak. Tasodifiy ma’lumotlar yig‘indisini ma’lumotlar bazasi deb atash mumkin emas.
- Ma’lumotlar bazasi ma’lum bir maqsad uchun ishlab chiqilgan, tuzilgan va ma’lumotlar bilan to‘ldirilgan bo‘lishi kerak.

Ma’lumotlar bazasi har qanday o‘lcham va murakkablikda bo‘lishi mumkin.

Masalan, fayldagi ismlar va manzillar ro‘yxati atigi bir necha yuz yozuvdan iborat bo‘lishi mumkin va ularning tuzilishi oddiydir. Boshqa tomondan, katta kutubxonaning kompyuterlashtirilgan katalogi turli toifalar bo‘yicha tashkil etilgan bo‘lishi mumkin: masalan, - asosiy muallifning familiyasi, kitob nomi, kitob nashriyoti va h.k; har bir toifa alifbo tartibida tartiblangan holda bo‘lishi mumkin.

Katta tijorat ma’lumotlar bazasiga Amazon.com misol bo‘la oladi. U 60 milliondan ortiq faol foydalanuvchi va millionlab kitoblar, CD, videolar, DVD, o‘yinlar, elektronika, kiyim-kechak va boshqa tovarlar ma’lumotlarini o‘z ichiga oladi. Undagi ma’lumotlar bazasi 42 terabaytdan ortiq joyni egallaydi va yuzlab kompyuterlarda (serverlarda) saqlanadi. Har kuni millionlab foydalanuvchilar Amazon.com saytiga kirishadi va xaridlar qilish uchun ma’lumotlar bazasidan foydalanadilar. Yangi kitoblar va boshqa narsalar inventarizatsiyaga qo‘shilganda va sotib olishlar amalga oshirilganda, ma’lumotlar bazasi doimiy ravishda yangilanadi.

Ma’lumotlar bazasi qo‘lda yoki kompyuter yordamida yaratilishi va saqlanishi mumkin.

Ma’lumotlar bazasining asosiy maqsadini quyidagicha ifodalashimiz mumkin:

1. *Ma'lumotlarni samarali boshqarish:* Ma'lumotlar bazasi turli sohalarda katta hajmdagi ma'lumotlarni tartibli saqlash, boshqarish va ularga tezkor kirish imkonini berish uchun mo'ljallangan. Bu ma'lumotlarni avtomatlashtirilgan tarzda qayta ishlash va tahlil qilish imkonini beradi.
2. *Ma'lumotlarning yaxlitligini saqlash:* Ma'lumotlar yagona manbada saqlanadi va ko'p foydalanuvchi ulardan bir vaqtda foydalanganda ham ularning to'g'riligi va yaxlitligi saqlanishi kerak.
3. *Takrorlanishni bartaraf qilish:* Ma'lumotlar bazasi bir xil ma'lumotlarning bir necha joyda takrorlanishini oldini olishga yordam beradi, bu esa resurslarni tejash va ma'lumotlarning to'g'riligini ta'minlagan holda saqlash imkonini beradi.
4. *Tezkor kirish va tahlil qilish:* Ma'lumotlarga tezkor kirishni ta'minlab, ularni osongina tahlil qilish va ma'lumotlarga asoslangan qarorlar qabul qilish jarayonlarini tezlashtiradi.
5. *Xavfsizlikni ta'minlash:* Foydalanuvchilarning ruxsatsiz kirishini oldini olish uchun ma'lumotlar bazasida ruxsatlar va xavfsizlik siyosati qo'llaniladi, bu esa ma'lumotlarni xavfsiz saqlashni ta'minlaydi.

Ma'lumotlar bazasining asosiy vazifalari:

1. *Ma'lumotlarni saqlash:* Ma'lumotlarni aniq va tartibli saqlash, ularning takrorlanishini bartaraf etish va qayta ishlashni osonlashtirish.
2. *Ma'lumotlarni qayta ishlash:* Ma'lumotlarga kirish va ularni qayta ishlash uchun samarali vositalarni taqdim etish, shu jumladan so'rovlarni bajarish, ma'lumotlarni tahlil qilish va hisob-kitoblar qilish.
3. *Ma'lumotlarni yangilash va boshqarish:* Ma'lumotlarni muntazam yangilab turish, ma'lumotlar bazasini kengaytirish, ma'lumotlarning dolzarbligini va mosligini ta'minlashni o'z ichiga oladi.

4. *Xavfsizlik va maxfiylikni ta'minlash:* Foydalanuvchilarga kirishni nazorat qilish, shaxsiy va tijorat ma'lumotlarining maxfiyligini ta'minlash uchun xavfsizlik choralarini o'rnatish.
5. *Ko'p foydalanuvchanlik:* Bir vaqtning o'zida bir nechta foydalanuvchi tomonidan ma'lumotlarga kirishni ta'minlash va ma'lumotlarni o'zaro baham ko'rish imkoniyatini yaratish.

1.2. Ma'lumotlar bazasining asosiy terminlari

Ma'lumotlar bazasi (Database) tushunchasi: Ma'lumotlarni tizimli ravishda saqlash, boshqarish va ulardan samarali foydalanish imkonini beradigan tuzilmalar to'plami. Bu ma'lumotlar bir yoki bir nechta jadvallarda saqlanadi va ular o'rtasidagi bog'lanishlar orqali tashkil etiladi.

Jadval (Table) tushunchasi: Ma'lumotlar bazasida ma'lumotlarni saqlashning asosiy birligi. Har bir jadval ma'lum bir obyekt yoki hodisa haqidagi ma'lumotlarni o'z ichiga oladi. Har bir jadvalning ustunlari (columns) va satrlari (rows) mavjud. Ustunlar (fields) ma'lum bir ma'lumot turini (masalan, **ism**, **familiya** yoki **telefon raqami**) belgilaydi, satrlar esa ushbu ustunlar bo'yicha individual yozuvlarni ifodalaydi.

Misol: Agar *"Talabalar"* jadvali quyidagicha ko'rinishda bo'lsa:

1.1-jadval. Talabalar jadvali

Talaba raqami	Ism	Familiya	Kurs
12345	Shohista	Karimova	2
12346	Alisher	Rahmonov	3

Yuqoridagi jadvalda:

Jadval – bu *"Talabalar"* jadvali.

Har bir **satr** bitta **yozuv** hisoblanadi. Masalan, **"Shohista Karimova"** ga tegishli ma'lumotlar bitta yozuv sifatida ifodalanadi.

Demak, jadval yozuvlar to'plamidan iborat va har bir yozuv jadvaldagi ma'lumotlarning bir individual qatorini tashkil etadi.

Fayl: Bu atama ko‘proq ma’lumotlarni saqlash uchun ishlatiladigan fizik saqlash tizimini anglatadi. Fayl tizimi bilan bog‘liq tushunchalarda fayllar ichida turli xil yozuvlar yoki ma’lumotlar saqlanadi.

Agar biz ma’lumotlarni saqlashning mantiqiy tuzilmasi haqida gapirayotgan bo‘lsak, **jadval** atamasi to‘g‘ri bo‘lishi mumkin. Agar ma’lumotlarning fizik joylashuvi haqida, ya’ni ularning qanday fayllarda saqlanayotgani haqida gapirilsa, **fayl** so‘zi mos keladi.

Yozuv (Record) tushunchasi: Jadvalning qatori bo‘lib, har bir yozuv ma’lum bir obyektning (masalan, bir talabani) haqidagi barcha ma’lumotlarni o‘z ichiga oladi. Har bir yozuvda jadvalning ustunlari bilan bog‘liq ma’lumotlar mavjud. **Misol**: Bir talabaga oid ismi, manzili, kurslari kabi ma’lumotlar.

Ustun (Column/Field) tushunchasi: Jadvaldagi ma’lumotning turi yoki atributini ifodalaydi. Ustunlar ma’lumotlarning turini (masalan, ism, tug‘ilgan sana) belgilaydi va ular bo‘yicha qatorlar (yoki yozuvlar) tashkil etiladi.

Misol: Talaba jadvalidagi ustunlar: “Ism”, “Yosh”, “Kurs”.

Atribut tushunchasi: Jadvaldagi ustunlarni ifodalaydi. Har bir atribut ma’lumotlar bazasida saqlanadigan ma’lumotlarning bir turini tasvirlaydi. Atributlar quyidagi xususiyatlarga ega:

1. **Nom**: Har bir atributning o‘ziga xos nomi bo‘lishi kerak, bu esa ma’lumotlar bazasida uning ma’nosini anglatadi (masalan, “ism”, “familiya”, “tug‘ilgan sana”).
2. **Ma’lumotlar turi**: Atributning ma’lumotlar turi uning qanday ma’lumotlarni saqlashi mumkinligini belgilaydi (masalan, matn, raqam, sana va h.k.).
3. **Xususiyatlar**:
   - ✓ **Majburiy atributlar(mandatory attribute (NOT NULL)) -** Bu shuni anglatadiki, bu ustun uchun ma’lumot bo‘sh qolishi mumkin emas. Har safar

yangi yozuv qo‘shilganda yoki mavjud yozuv yangilanganda, ushbu ustun albatta to‘ldirilishi kerak.

- ✓ **Takrorlanmas atributlar (UNIQUE)** – Ba’zi ustunlarda qiymatlarning takrorlanishi mumkin emas. Masalan, “Pasport raqami” yoki “Email” kabi ustunlarda har bir yozuv uchun noyob qiymat kiritilishi talab qilinadi, shunda ma’lumotlar bazasida bir xil qiymatlar takrorlanmaydi.
- ✓ **Tashqi kalit (FOREIGN KEY) -** Bu atributlar boshqa jadvallardagi asosiy kalit ustuniga ishora qiladi. Bu bog‘lanish orqali ma’lumotlar bazasidagi jadvallar o‘zaro biri biri bilan bog‘lanadi. Masalan, “Talabalar” jadvalida “Guruh_ID” atributi “Guruhlar” jadvalidagi asosiy kalit bo‘lgan “ID” ustuniga bog‘lanishi mumkin.

Ma’lumotlar bazasidagi har bir ustun bir atributni ifodalaydi. Ular o‘rtasida asosiy farq, “atribut” ko‘proq nazariy tushuncha bo‘lsa, “ustun” esa jadval strukturasida mavjud bo‘lgan elementni ifodalaydi.

Asosiy kalit (Primary Key) tushunchasi: Har bir jadvalda mavjud bo‘lgan noyob identifikator bo‘lib, har bir yozuvni boshqalaridan ajratib turadi. Bu kalit orqali ma’lumotlar bazasida bir xil yozuvlarni aniqlash mumkin. **Misol**: Talaba jadvalida “Talaba ID” asosiy kalit sifatida ishlatilishi mumkin.

Tashqi kalit (Foreign Key) tushunchasi: Bir jadvaldagi kalitni boshqa jadval bilan bog‘laydi va bu bog‘lanishlar orqali jadvallar o‘rtasida munosabatlar yaratiladi. Tashqi kalit bir jadvaldagi asosiy kalitga ishora qiladi. **Misol**: “Talaba ID” *talabalar* jadvalidan *kurslar* jadvaliga bog‘lanishi mumkin.

Indeks (Index) tushunchasi: Jadvaldagi qatorlarga tezkor kirish imkonini beradigan ma’lumotlar tuzilmasi. Indeks yordamida ma’lumotlarni qidirish va saralash tezlashadi. Misol: “Ism” ustuni bo‘yicha indeks yaratish, ma’lumotlarni ismlar bo‘yicha tez topishga yordam beradi.

So‘rov (Query) tushunchasi: Ma’lumotlar bazasidan kerakli ma’lumotlarni olish uchun yoziladigan buyruqlar yoki so‘rovlardir. So‘rovlar orqali ma’lumotlar qidiriladi, yangilanadi yoki o‘chiriladi. Misol: SQL (Structured Query Language) orqali talabalar ro‘yxatini kurslari bilan olish mumkin.

Ma’lumotlar turlari (Data Types) tushunchasi: Har bir ustunda saqlanadigan ma’lumotlar turi. Bu ma’lumotlar matn, butun son, kasr son, sana kabi turli xil bo‘lishi mumkin. Misol: “Yosh” ustuni butun son (integer), “Tug‘ilgan sana” esa sana (date) sifatida saqlanadi.

Normallashtirish (Normalization) tushunchasi: Ma’lumotlarni takrorlanishni kamaytirish va samaradorlikni oshirish maqsadida jadvallarni tuzish jarayoni. Bu jarayon orqali katta jadvallarni kichikroq va bog‘langan jadvallarga bo‘lish orqali ma’lumotlarning takrorlanishi oldini olinadi.

Ma’lumotlar bazasini boshqarish tizimi (MBBT) foydalanuvchilarga ma’lumotlar bazasini yaratish va saqlash imkonini beruvchi kompyuterlashtirilgan tizimdir. DBMS - bu turli foydalanuvchilar va ilovalar o‘rtasida ma’lumotlar bazalarini aniqlash, qurish, manipulyatsiya qilish va almashish jarayonlarini osonlashtiradigan umumiy maqsadli dasturiy ta’minot tizimidir. Ma’lumotlar bazasini aniqlash ma’lumotlar bazasida saqlanadigan ma’lumotlar turlarini, tuzilmalarini va cheklovlarini belgilashni o‘z ichiga oladi.

Ma’lumotlar bazasi(MB) va **ma’lumotlar bazasini boshqarish tizimi** (MBBT) tushunchalari o‘rtasidagi farqlar bo‘lib, biz ularni quyidagicha izohlashimiz mumkin:

1. Database (Ma’lumotlar bazasi):

- **Tushunchasi:** Ma’lum bir sxema asosida saqlanuvchi ma’lumotlarning strukturalashgan majmuasi. Bu yerda ma’lumotlar fayllar, jadvallar yoki yozuvlar ko‘rinishida saqlanadi.

- **Misol:** Universitetda talabalar, fanlar, o‘qituvchilar va ularning reytinglari haqidagi ma’lumotlar bir-biri bilan bog‘liq holda saqlanishi mumkin.
- **Asosiy maqsadi:** Ma’lumotlarni markazlashtirilgan holda saqlash va ularni keyinchalik qayta ishlash, tahlil qilish yoki yangilash uchun foydalanish.
- **Tarkibi:** Jadvallar (tables), yozuvlar (records), ustunlar (columns) va bog‘lanishlardan (relationships) tashkil topgan.

2. DBMS (Database Management System):

- **Tushunchasi:** DBMS — bu dasturiy ta’minot bo‘lib, u ma’lumotlar bazasini boshqaradi. Bu tizim foydalanuvchilarga ma’lumotlar bazasiga kirish, ma’lumotlarni saqlash, tahrirlash, qayta ishlash va xavfsizligini ta’minlash imkonini beradi.
- **Misol:** MySQL, Oracle, PostgreSQL kabi DBMS tizimlari foydalanuvchilarga ma’lumotlar bazasini boshqarishda yordam beradi.
- **Asosiy maqsadi:** Ma’lumotlar bazasini boshqarish, foydalanuvchi so‘rovlarini bajarish, ma’lumotlar xavfsizligini ta’minlash va ma’lumotlarni yangilash uchun dasturiy vositalarni taqdim etish.
- **Funksiyalari:** Ma’lumotlarni saqlash, qidirish, yangilash, o‘chirish, kirish huquqlarini boshqarish, hamda ko‘p foydalanuvchilar bilan bir vaqtning o‘zida ishlashni ta’minlash.

Asosiy farqlari:

Vazifasi:

- **MB**: Ma’lumotlar bazasi faqatgina ma’lumotlarni saqlaydi.
- **MBBT**: DBMS esa ma’lumotlar bazasini boshqarish, ularga kirish va tahlil qilish uchun vositalarni taqdim etadi.

Ko‘p foydalanuvchilik:

MB: Faqat ma’lumotlar joylashgan joy. **MBBT**: Ma’lumotlar bilan bir vaqtning o‘zida bir necha foydalanuvchi ishlashini boshqaradi.

Ma’lumotlar bazasini aniqlash jarayoni ma’lumotlarni saqlash uchun mo‘ljallangan ma’lumot turlari, tuzilmalari va cheklovlarini belgilashni o‘z ichiga oladi. Ma’lumotlar bazasi yoki tavsifiy ma’lumotlar MBBT tomonidan ma’lumotlar bazasi katalogi yoki lug‘at shaklida saqlanadi, bu esa meta-ma’lumot deb ataladi.

Ma’lumotlar bazasini qurish — bu MBBT tomonidan boshqariladigan ma’lumotlarni saqlash uchun qandaydir saqlash vositasiga ma’lumotlarni joylashtirish jarayonidir.

Ma’lumotlar bazasini manipulyatsiya qilish ma’lumotlarni olish uchun ma’lumotlar bazasiga so‘rov yuborish, o‘zgarishlarini aks ettirish uchun ma’lumotlarni yangilash va ma’lumotlardan hisobotlar yaratishni o‘z ichiga oladi.

Ma’lumotlar bazasini ulashish bir vaqtning o‘zida bir nechta foydalanuvchilar va dasturlarni ma’lumotlar bazasiga kirish imkonini beradi.

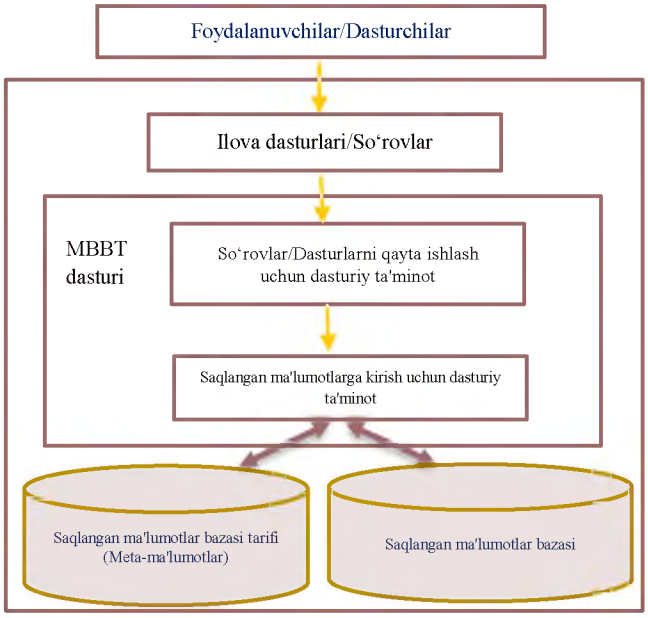
Ilova dasturi ma’lumotlar bazasiga MBBTga ma’lumotlar uchun so‘rov yuborish orqali murajat qiladi. So‘rov orqali ma’lum natijalarni olish mumkin; tranzaksiya esa ma’lumotlarni o‘qish va ba’zi ma’lumotlarni ma’lumotlar bazasiga yozishga imkon beradi. Himoya tizimi apparat yoki dasturiy ta’minot nosozliklaridan (yoki qulflashlardan) himoya qilishni va ruxsatsiz yoki zararli kirishdan xavfsizlikni ta’minlashni o‘z ichiga oladi. Odatda, katta ma’lumotlar bazasining hayot sikli ko‘plab yillarga cho‘zilishi mumkin.

Biz ma’lumotlar bazasini va MBBT dasturiy ta’minotini birgalikda ma’lumotlar bazasi tizimi deb ataymiz. 1.1-rasmda siz bilan muhokama qilingan ba’zi tushunchalar tasvirlangan.

Quyidagi rasmda ma’lumotlar bazasi tizimining qanday ishlashini tasvirlangan. Unga ko‘ra:

1. **Foydalanuvchilar/Dasturchilar** ma’lumotlar bazasiga kirishadi yoki dasturlar orqali so‘rovlar yuborishadi. Ular ilova dasturlari yoki to‘g‘ridan-to‘g‘ri so‘rovlar orqali tizim bilan muloqot qilishadi.

2. **Ma’lumotlar bazasini boshqarish tizimi dasturi (DBMS Software)** — bu tizim foydalanuvchilarning ma’lumotlar bazasiga kirishlarini boshqaradi va ularni samarali ishlashini ta’minlaydi. MBBT ikki asosiy vazifani bajaradi:
   - **So‘rovlar/Dasturlarni qayta ishlash dasturi** — foydalanuvchilar tomonidan yuborilgan so‘rovlarni yoki dasturlarni bajaradi ya’ni ma’lumotlarni tahlil qilish, yangilash yoki qidirish kabi amallarni amalga oshiradi.
   - **Saqlangan ma’lumotlarga kirish dasturi** — bu dastur saqlangan ma’lumotlar bazasidagi real ma’lumotlarga murajat qilishni ta’minlaydi.
3. **Saqlangan ma’lumotlar bazasi ta’rifi(Meta-ma’lumotlar)** — bu ma’lumotlar bazasining tuzilishi va tashkil etilishi haqidagi ma’lumotlardir. Bu meta-ma’lumotlar ma’lumotlar bazasidagi jadvallar, ustunlar, ma’lumot turlari va boshqa muhim elementlarni ifodalaydi.
4. **Saqlangan ma’lumotlar bazasi** — bu haqiqiy ma’lumotlar saqlanadigan joydir. Ma’lumotlar foydalanuvchilar so‘rovlariga asoslangan holda o‘qiladi, yangilanadi yoki o‘chiriladi.



1.1-rasm. Ma’lumotlar bazasi tizimining ishlash muhiti

Demak, bu rasmda ma’lumotlar bazasi tizimining umumiy ishlash jarayonini ko‘rsatadi, ya’ni foydalanuvchi ilova yoki so‘rov orqali tizimga murojaat qiladi, so‘ngra DBMS dasturi ushbu so‘rovni qayta ishlaydi va kerakli ma’lumotlarni beradi.

Ma’lumotlar bazasining asosiy maqsadi – katta hajmdagi ma’lumotlarni tizimli ravishda saqlash, ularga tezkor va samarali tarzda murajat qilish, ulardan foydalanish, tahlil qilish va boshqarish imkonini berishdir. Bu maqsad orqali foydalanuvchilar, tashkilotlar va dasturlar ma’lumotlarni ishonchli, izchil va xavfsiz tarzda boshqarish imkoniyatiga ega bo‘ladi.

1.3. Ma’lumotlar bazasiga qo‘yiladigan talablar

Ma’lumotlar bazasiga qo‘yiladigan talablar quyidagilardan iborat:

1. Ma’lumotlarning yaxlitligi va to‘g‘riligi (Data Integrity and Accuracy)

Ma’lumotlar bazasi foydalanuvchilarga aniq va ishonchli ma’lumotlarni taqdim qilishi kerak. Yaxlitlikni ta’minlash uchun uchta asosiy tamoyil qo‘llaniladi:

Entity Integrity (Ob’ekt yaxlitligi): Har bir jadvalda asosiy kalitlar (primary key) noyob va NULL qiymatga ega bo‘lmasligi kerak.

Referential Integrity (Bog‘lanish yaxlitligi): Jadval o‘rtasidagi bog‘lanishlar to‘g‘ri ishlashi kerak.

Domain Integrity (Domen yaxlitligi): Har bir ustun belgilangan ma’lumot turiga mos kelishi va qiymatlar doirasidan chiqmasligi lozim.

2. Ishonchlilik (Reliability)

Ma’lumotlar bazasi har qanday texnik nosozliklar yoki foydalanuvchilar xatosidan himoyalangan bo‘lishi lozim.

Zaxiralash va tiklash (Backup and Recovery): Ma’lumotlar bazasi muntazam ravishda zaxiralangan va favqulodda holatda tiklanishi zarur.

Nosozlikdan himoya (Fault Tolerance): Nosozlik paydo bo‘lsa ham, ma’lumotlar bazasi operatsiyalarni davom ettirishi yoki iloji boricha tez tiklanishi kerak.

3. Ma’lumotlarning xavfsizligi (Data Security)

Ma’lumotlar bazasi ma’lumotlarning maxfiyligi, yaxlitligi va mavjudligini ta’minlashi kerak.

Kirishni boshqarish (Access Control): Ma’lumotlarni faqat ruxsat etilgan foydalanuvchilar o‘zgartira oladi yoki ko‘ra oladi.

Shifrlash (Encryption): Ma’lumotlarning tarmoq orqali uzatilishida ularning himoyalanganligiga ishonch hosil qilinadi.

4. Moslashuvchanlik va kengayuvchanlik (Scalability and Flexibility)

Ma’lumotlar bazasi yangi foydalanuvchilar yoki katta hajmdagi ma’lumotlarga moslashishi va ishlash unumdorligini saqlab qolishi kerak.

Kengaytirilish imkoniyati: Yangi ustunlar yoki jadval qo‘shishda tizimda o‘zgarish talab qilinmasligi kerak.

Ko‘p foydalanuvchilarni qo‘llab-quvvatlash: Bir vaqtning o‘zida bir nechta foydalanuvchi ishlaganda tizim barqaror bo‘lishi lozim.

5. Ishlash tezligi va samaradorlik (Performance and Efficiency)

Ma’lumotlar bazasi katta hajmdagi so‘rovlar va ma’lumotlar bilan tez ishlashi lozim.

Indekslar: So‘rovlarni tezlashtirish uchun ma’lumotlar indekslangan bo‘lishi kerak.

Optimallashtirilgan so‘rovlar (Query Optimization): So‘rovlar bajarilish tezligini oshirish uchun SQL optimallashtirish amalga oshirilishi kerak.

6. Muvofiqlik (Consistency)

Bir vaqtda bir nechta foydalanuvchilar tizimdan foydalanganda ma’lumotlar bir xil saqlanishi kerak. Bu, ayniqsa, tranzaksiyalar bilan ishlashda muhim.

ACID tamoyillari (Atomicity, Consistency, Isolation, Durability): Tranzaksiyalarni ishonchli amalga oshirish tamoyillari.

Yuqoridagi talablarni bajarish ma’lumotlar bazasining mustahkamligi va ishonchliligini oshiradi.

1.4. Avtomatlashgan axborot tizimlari: axborotni qayta ishlaydigan ixtiyoriy tizim, tadbiq etish sohasiga qarab ATlar ishlab chiqarish sohasi

Avtomatlashgan axborot tizimlari (AAT) – bu kompyuter texnologiyalari asosida ma'lumotlarni yig'ish, saqlash, tahlil qilish va qayta ishlash jarayonlarini avtomatlashtiruvchi tizimdir. Ushbu tizimlar ishlab chiqarish jarayonlarini samarali boshqarish va nazorat qilishda keng qo'llaniladi.

Ishlab chiqarish sohasidagi avtomatlashgan axborot tizimlari

AATlar ishlab chiqarish sohasida quyidagi asosiy yo'nalishlarda qo'llaniladi:

1. Texnologik jarayonlarni boshqarish tizimlari (Process Control Systems)

Ishlab chiqarishdagi texnologik jarayonlarni nazorat qilish va avtomatlashtirish uchun qo'llaniladigan tizimlar. Masalan:

- SCADA tizimlari: Rejalashtirish, nazorat va boshqaruvni ta'minlaydi.
- PLC tizimlari: Mahsulot ishlab chiqarish liniyalaridagi asbob-uskunalarni boshqarish uchun dasturlanadigan mantiqiy kontrollerlar.

2. Resurslarni rejalashtirish va boshqarish tizimlari (ERP - Enterprise Resource Planning)

ERP tizimlari ishlab chiqarish resurslarini optimal taqsimlash va rejalashtirish uchun qo'llaniladi. Ular quyidagilarni o'z ichiga oladi:

- Moddiy resurslar (xomashyo, jihozlar).
- Inson resurslari (ishchilar).
- Moliyaviy resurslar (xarajatlar va daromadlarni boshqarish).

Misol: SAP ERP tizimi korxona darajasida resurslarni integratsiyalashda ishlatiladi.

3. Ta'minot zanjiri boshqaruvi (Supply Chain Management - SCM)

Ishlab chiqarish uchun xomashyolarni yetkazib berishdan tortib, tayyor mahsulotni mijozga yetkazib berishgacha bo'lgan barcha jarayonlarni avtomatlashtirish.

Maqsad: Tezkorlik va samaradorlikni oshirish, xarajatlarni kamaytirish.

Ilova: Katta omborxonalarda inventarizatsiya va logistika jarayonlarini boshqarish.

Misol: Amazon'ning avtomatlashtirilgan logistika tizimi.

4. Mahsulot sifatini nazorat qilish tizimlari (Quality Control Systems)

Mahsulotlarning sifatini real vaqt rejimida kuzatish va tekshirish uchun qo'llaniladi.

Sensorlar va kameralar: Mahsulotning o'lchamlari va parametrlarini avtomatik tahlil qiladi.

AI yordamida sifat nazorati: Sun'iy intellekt algoritmlari nosoz mahsulotlarni aniqlashga yordam beradi.

Misol: Elektronika ishlab chiqarishda nosozliklarni aniqlovchi AI tizimlari.

Ishlab chiqarish sohasida avtomatlashgan axborot tizimlarining afzalliklari:

- Samaradorlikni oshirish: Inson xatosini kamaytirib, ishlab chiqarish jarayonlarini optimallashtiradi.
- Real vaqt rejimida nazorat: Ishlab chiqarishning barcha bosqichlarini kuzatish imkonini beradi.
- Qaror qabul qilishni yaxshilash: Tahliliy ma'lumotlar asosida tezkor va aniq qarorlar qabul qilishni ta'minlaydi.
- Xarajatlarni qisqartirish: Resurslardan samarali foydalanish va texnik nosozliklarning oldini olish imkoniyatini beradi.

1.5. Ma'lumotlar bazasida jadvallar yaratish bo'yicha amaliy qism

Keling, ko'pchilik uchun yaxshi tanish bo'lgan oddiy namunani ko'rib chiqamiz.

Predmet soha UNIVERSITET bo'lsin.

UNIVERSITET ma'lumotlar bazasi, bu universitet muhitida talaba, kurslar, bo'limi, baholar va h.klar haqidagi ma'lumotlarni saqlash uchun mo'ljallangan. Quyida ma'lumotlar bazasining tuzilmasi va bir nechta ma'lumot yozuvlari ko'rsatilgan.

Ma’lumotlar bazasi 4 ta fayl sifatida tashkil etilgan. **Talabalar** fayli har bir talaba haqidagi ma’lumotlarni, **Kurs** fayli har bir kurs haqidagi ma’lumotlarni, **Bo‘lim** fayli har bir kurs bo‘limi haqidagi ma’lumotlarni, **Baholash natijalari** fayli talabalar turli bo‘limlarda olgan baholar ma’lumotlarini saqlaydi.

Ushbu ma’lumotlar bazasini aniqlash uchun har bir fayldagi yozuvlarning tuzilishini belgilashimiz kerak. Quyida, **Talabalar** jadvali talabaning ismi, talaba_raqami, kurs (masalan, birinchi kurs yoki ‘1’, ikkinchi kurs yoki ‘2’ va hokazo) va mutaxassisligini (masalan, matematika, kompyuter injiniring va h.k) ifodalash uchun ma’lumotlarni o‘z ichiga oladi, Kurs jadvali kurs_nomi, kurs_raqami, kredit_soatlari va kursni taklif qiladigan bo‘limni o‘z ichiga oladi va h.k.

1-jadval. Talaba jadvali

Ismi	talaba_raqami	kurs	mutaxassislik
Shoxruh	8	1	KI
Aziza	14	1	DI

2-jadval.Kurs jadvali

Kurs_nomi	kurs_raqami	Kredit_soatlari	Bo‘lim
Ma’lumotlar bazasi	Mb202412	6	KI
Dasturlash	Das1306	4	KI

3-jadval. Bo‘lim jadvali

Bo‘lim identifikatori	kurs_raqami	semestr	yil	instruktor
85	Mb202412	kuzgi	2021	Alimov
95	Das1306	Baxorgi	2022	Alimov

4-jadval. Baholash_natijalari jadvali

Talaba_raqami	bo‘lim identifikatori	Baho
8	85	A
8	95	B

Har bir ma’lumot elementi uchun ma’lumot turini ko‘rsatishimiz kerak. Masalan, **STUDENT**ning ismi alfavit belgilari qatoridan iborat bo‘lishini, student_number raqamli bo‘lishini va **Baholash natijalari** yozuvidagi baho A, B, C, D, F, I kabi belgilar to‘plamidan tanlanishini belgilashimiz mumkin. Shuningdek, ba’zi ma’lumot elementlari qiymatlarini **kodlash sxemasidan** foydalanishimiz mumkin. Masalan, 1-jadvalda **Talaba**ning kursini 1 (birinchi kurs), 2 (ikkinchi kurs), 3 (uchinchi kurs), 4 (to‘rtinchi kurs) sifatida ifodalaymiz.

Ma’lumotlar bazasida kodlash sxemasi (coding scheme) ma’lumotlar elementlarining qiymatlarini ifodalash uchun foydalaniladigan usuldir. Bu sxema ma’lumotlar bazasida ma’lumotlarni saqlash va qayta ishlashni osonlashtiradi, chunki u elementlarning qiymatlarini qisqa va aniq shaklda ko‘rsatadi. Misol uchun, talaba kursini ifodalashda kodlash sxemasi yordamida “1” raqami birinchi kurs, “2” - ikkinchi kurs, “3” - uchinchi kurs, “4” - to‘rtinchi kurs va h.k. ko‘rinishida ifodalanishi mumkin.

UNIVERSITET ma’lumotlar bazasini yaratish uchun har bir talaba, kurs, bo‘lim, baho hisobotlarini tegishli faylga yozuv sifatida kiritamiz. E’tibor bering, turli fayllardagi yozuvlar bir-biriga bog‘liq bo‘lishi mumkin. Masalan, **Talaba** faylidagi Shoxruhning yozuvi **Baholash_natijalari** faylidagi ikki yozuvga bog‘langan, bu esa **Shoxruh**ning ikki bo‘limdagi baholarini ko‘rsatadi. O‘rta va katta hajmdagi ma’lumotlar bazalari ko‘plab yozuvlarni o‘z ichiga oladi va yozuvlar o‘rtasida ko‘plab bog‘lanishlar mavjud bo‘ladi.

Nazorat savollari:

1. Jadval va faylning nima farqi bor?
2. Ma’lumotlar bazasi nima?
3. MBBT nima?
4. MBBT va MB ning asosiy farqli jihatlari?
5. Atribut nima?
6. Birlamchi va ikkilamchi kalitlar nima?

2 BOB. MA'LUMOT BAZASINING ARXITEKTURASI VA UCH BOSQICHLI ARXITEKTURA

Tayanch so'zlar: *abstraktsiyalash, ma'lumotlar modeli, mohiyat, atribut, munosabat, ma'lumotlar bazasi sxemasi, ichki daraja, konseptual daraja, tashqi daraja*

2.1. Ma'lumotlar bazasini sinflarga ajratish

Ma'lumotlar bazasi yondashuvining asosiy xususiyatlaridan biri ma'lumotlarni bir darajada abstraktsiyalashni ta'minlashidir. Ma'lumotlarni abstraktsiyalash, odatda, ma'lumotlarni tashkil etish va saqlash holatlarini yashirishni va ma'lumotlarni yaxshiroq tushunish uchun asosiy xususiyatlarni ta'kidlashni anglatadi. Ya'ni ma'lumotlarni abstraktsiyalash — bu murakkab ma'lumotlarni soddalashtirish jarayonidir. Buning ma'nosi shundaki, ma'lumotlar qanday saqlanayotganini va tashkil etilayotganini emas, balki ularning muhim xususiyatlari va ma'nosini ko'rsatishdir. Abstraktsiyalash yordamida biz ma'lumotlarning asl ko'rinishini, detallarini unchalik ko'rmasdan, faqat kerakli va asosiy ma'lumotlarni ko'rishimiz mumkin. Bu foydalanuvchilarga ma'lumotlarni yaxshiroq tushunishga va ulardan samarali foydalanishga yordam beradi. Misol uchun, sizga biror avtomobil haqida ma'lumot kerak bo'lsa, siz uning markasi, rangi va narxi haqida bilishni xohlaysiz, lekin qanday mexanik qismlar ishlayotganini bilishingiz shart emas.

Ma'lumotlar modeli — bu ma'lumotlar bazasining tuzilishini tasvirlash uchun ishlatilishi mumkin bo'lgan tushunchalar to'plami. Ushbu model ma'lumotlar bazasining abstraktsiyasini yaratish uchun zarur bo'lgan vositalarni taqdim etadi. Ma'lumotlar modeli, ma'lumotlar bazasining qanday ishlashini va foydalanuvchilar qanday qilib ma'lumotlardan foydalanishini tushunish uchun zaruriy vositalarni taqdim etadi. Shu tarzda, ma'lumotlar modeli bazaning umumiy tuzilishi va uning komponentlari haqida aniq tasavvur beradi. Ma'lumotlar bazasi tuzilishi deganda, ma'lumotlar uchun qo'llaniladigan ma'lumotlar turlari, munosabatlar va cheklovlar tushuniladi.

Ma’lumotlar modellari, ma’lumotlar bazasini boshqarish va undan foydalanish uchun zarur bo‘lgan asosiy amallarni belgilaydi. Bu amallar ma’lumotlarni qidirish va yangilash kabi funksiyalarni o‘z ichiga oladi.

1. **Qidiruv**: Bu amal yordamida foydalanuvchilar ma’lumotlar bazasidan kerakli ma’lum otlarni topishlari mumkin. Masalan, siz ma’lumotlar bazasidan muayyan xodimning ismini yoki maoshini qidirishingiz mumkin.
2. **Yangilash (Update)**: Bu operatsiya orqali foydalanuvchilar ma’lumotlar bazasidagi mavjud ma’lumotlarni o‘zgartirishlari mumkin. Misol uchun, agar bir xodimning maoshi oshsa, siz bu ma’lumotni yangilashingiz kerak.

Asosiy amallar to‘plami ma’lumotlar bazasi bilan ishlashda foydalanuvchilarga qulaylik yaratadi va ularning ehtiyojlariga mos keladigan ma’lumotlarni olish va yangilash imkonini beradi.

Ma’lumotlar modelida oddiy qidirish va yangilash operatsiyalaridan tashqari, ma’lumotlar bazasi ilovalari qanday ishlashini va ularning xatti-harakatlarini boshqarish uchun qo‘shimcha tushunchalarni kiritish tobora keng tarqalmoqda. Bu shuni anglatadiki, ma’lumotlar bazasi faqat ma’lumotlarni saqlash bilan cheklanib qolmay, balki ma’lumotlar bilan qanday amallar bajarilishi mumkinligini ham belgilaydi.

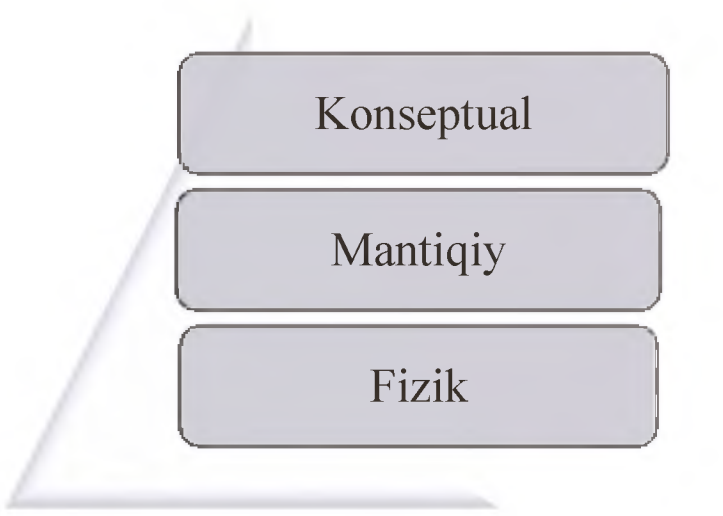
Misol uchun, ba’zi ma’lumotlar bazalarida foydalanuvchi tomonidan ruhsat berilgan amallarni kiritish mumkin. Masalan, **COMPUTE_GPA** operatsiyasi yordamida bir talabaga tegishli GPA (umumiy baho o‘rtachasi) ni hisoblash mumkin. Bu oddiygina ma’lumotlar qidirish yoki o‘zgartirishdan ko‘ra ko‘proq jarayonni o‘z ichiga oladi, ya’ni ma’lumotlar bazasining dinamik jihatini ham o‘z ichiga oladi.

Ushbu yondashuv, ayniqsa, ob’ektga yo‘naltirilgan va ob’ekt-relyatsion ma’lumotlar modellarida qo‘llaniladi. U yerda ma’lumotlar nafaqat obyektlar va ularning xususiyatlarini saqlash, balki ushbu obyektlar bilan qanday ishlar bajarilishi (masalan, operatsiyalar va hisob-kitoblar) kerakligini ham belgilash imkonini beradi.

Natijada, ma’lumotlar bazasi nafaqat statik ma’lumotlar yig‘indisi bo‘lib qolmaydi, balki dinamik, foydalanuvchi ehtiyojlariga moslashadigan tizimga aylanadi.

Bu ma’lumotlar bazasi dizayneriga ma’lumotlar bazasi ob’ektlari ustida ruxsat berilgan foydalanuvchi tomonidan belgilangan amallar to‘plamini aniqlash imkonini beradi.

Ko‘plab ma’lumotlar modellari taklif etilgan bo‘lib, ularni ma’lumotlar bazasi tuzilishini tasvirlash uchun ishlatiladigan tushunchalar turiga ko‘ra tasniflash mumkin. **Yuqori darajadagi yoki kontseptual ma’lumotlar modellari** ko‘plab foydalanuvchilar ma’lumotni qanday qabul qilishi bilan yaqin tushunchalarni taqdim etadi. **Past darajadagi yoki fizik ma’lumotlar modellari** esa ma’lumotlar kompyuter xotira vositalarida, odatda magnit disklarida qanday saqlanishi haqida tafsilotlarni tasvirlaydi. **Fizik ma’lumotlar modellari** tomonidan taqdim etilgan tushunchalar asosan kompyuter mutaxassislari uchun mo‘ljallangan, foydalanuvchilar uchun emas.



2.1-rasm. Ma’lumotlar modellari kategoriyalari

1. Konseptual ma’lumotlar modeli (Conceptual Data Model): Bu model ma’lumotlar tuzilishini yuqori darajada tasvirlaydi va foydalanuvchilarga ma’lumotlarning umumiy tuzilishi va munosabatlarini tushunishga yordam beradi. U odatda ob’ektlar, atributlar va ular o‘rtasidagi munosabatlarni o‘z ichiga oladi.

Mohiyat -bu ma’lumotlari ma’lumotlar bazasida saqlanishi kerak bo‘lgan biror real yoki tasavvur qilingan obyektdir.

Atribut mohiyatni yanada batafsil tavsiflovchi xususiyatdir. Masalan, xodimning ismi yoki maoshi uning atributlari hisoblanadi. Bu xususiyatlar mohiyat haqida qo‘shimcha ma’lumot beradi.

Ikki yoki undan ortiq mohiyat o‘rtasidagi **munosabat** bu mohiyatlar o‘rtasidagi bog‘lanishni ifodalaydi. Misol sifatida “talaba” va “kurs” o‘rtasidagi munosabatni keltirishimiz mumkin.

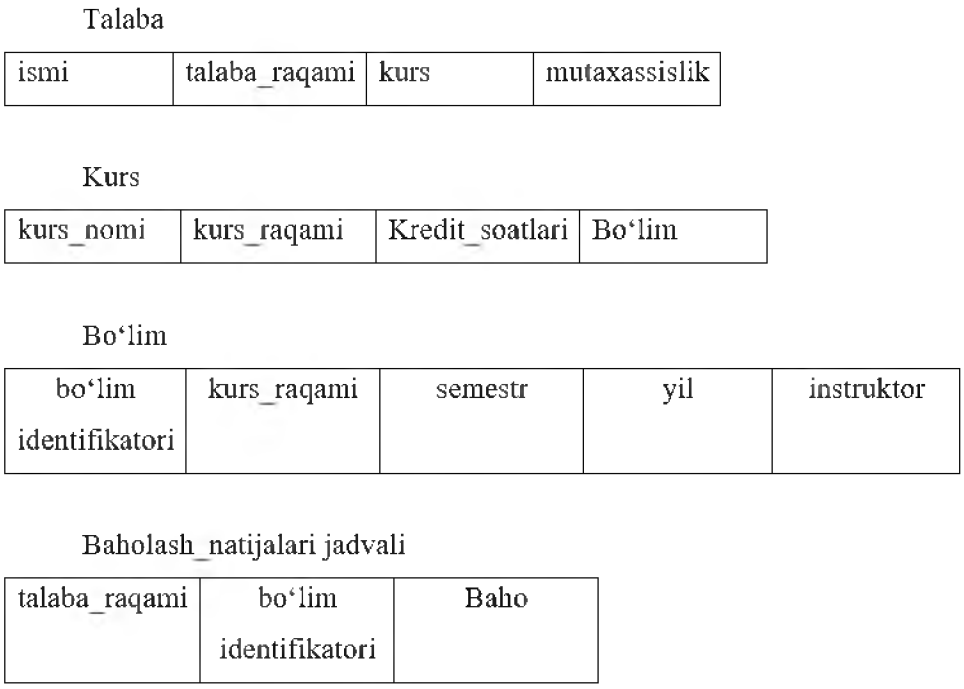
2. Mantiqiy ma’lumotlar modeli (Logical Data Model): Bu model ma’lumotlarni qanday tashkil qilish va saqlashni belgilaydi, lekin ma’lumotlar qanday saqlanishi haqida aniqroq tafsilotlarni o‘z ichiga olmaydi. Mantiqiy model ma’lumotlarning tuzilishini va ularning o‘zaro munosabatlarini, shuningdek, ma’lumotlar turlarini (masalan, raqamli, matnli) aniqlaydi.

Mantiqiy model: Ma’lumotlar va ularning tuzilishi haqida aniqroq va batafsil ma’lumotlar beradi, shu bilan birga, ma’lumotlar bazasining qanday tashkil etilishi va ishlashi haqida strukturalar taqdim etadi.

3. Fizik ma’lumotlar modeli (Physical Data Model): Bu model ma’lumotlar qanday saqlanadi va kompyuterda qanday tashkil etilishini ko‘rsatadi. U ma’lumotlarni saqlash joylarini, ma’lumotlar formati, indekslar va xotira tuzilmalarini o‘z ichiga oladi.

Fizik ma’lumotlar modellari ma’lumotlar kompyuterda qanday saqlanishini fayllar sifatida tasvirlaydi va yozuv formatlari, yozuv tartiblari va kirish yo‘llari kabi ma’lumotlarni ifodalaydi. Kirish yo‘li ma’lum ma’lumotlar bazasi yozuvlarini qidirishni samarali amalga oshirish uchun qidiruv strukturasidir, masalan, indekslash yoki xeshlash. Indeks — ma’lumotlarga indeks yoki kalit so‘z orqali bevosita kirishni ta’minlovchi kirish yo‘lining misolidir.

Ma’lumotlar modelida ma’lumotlar bazasi tavsifi va ma’lumotlar bazasining o‘zini farqlash muhimdir. Ma’lumotlar bazasi tavsifi **ma’lumotlar bazasi sxemasi** deb ataladi, u ma’lumotlar bazasini loyihalashda belgilanadi va tez-tez o‘zgarmaydi. Ko‘pgina ma’lumotlar modellarida sxemalarni **diagrammalar** sifatida ko‘rsatish uchun ma’lum qoidalar mavjud. 2.2-rasmda ko‘rsatilgan sxema 1.2-rasmda ko‘rsatilgan ma’lumotlar bazasi uchun **sxema diagrammasi** deb ataladi; diagramma har bir yozuv turining tuzilishini aks ettiradi, lekin yozuvlarning haqiqiy nusxalarini emas.



2.2-rasm. 1-mavzudagi ma’lumotlar bazasi uchun sxema diagrammasi

Biz sxemadagi har bir ob’ektni, masalan, TALABA yoki KURSni **sxema konstruktsiyasi** deb ataymiz.

Sxema diagrammasi faqat sxemaning ayrim jihatlarini ko‘rsatadi, masalan, yozuv turlari va ma’lumotlar elementlarining nomlari va ayrim turdagi cheklovlarni. Boshqa jihatlar sxema diagrammada ko‘rsatilmagan; masalan, 2.2-rasmda na har bir ma’lumot elementining ma’lumotlar turi, na turli fayllar orasidagi munosabatlar ko‘rsatilgan. Cheklovlarning ko‘p turlari sxema diagrammalarida ko‘rsatilmagan.

Ma’lumotlar bazasidagi haqiqiy ma’lumotlar juda tez o‘zgarishi mumkin. Masalan, 1.2-rasmda ko‘rsatilgan ma’lumotlar bazasi har safar yangi talabani qo‘shganimizda yoki yangi baho qo‘shganimizda o‘zgaradi. Ma’lumotlar bazasidagi ma’lumotlar **ma’lumotlar bazasi holati** deb ataladi. Berilgan ma’lumotlar bazasi holatida har bir sxema konstruktsiyasi o‘zining joriy to‘plamiga ega; masalan, TALABA konstruksiyasi uning misollari sifatida alohida talaba ob’ektlari (yozuvlari) to‘plamini o‘z ichiga oladi. Ko‘pgina ma’lumotlar bazasi holatlari ma’lum bir ma’lumotlar bazasi sxemasiga mos keladigan tarzda tuzilishi mumkin. Har safar yozuvni qo‘shganimizda yoki o‘chirganimizda yoki yozuvdagi ma’lumotlar elementining qiymatini o‘zgartirganimizda, biz ma’lumotlar bazasining bir holatini boshqa holatga o‘zgartiramiz.

Ma’lumotlar bazasi sxemasi va ma’lumotlar bazasi holati o‘rtasidagi farq juda muhimdir.

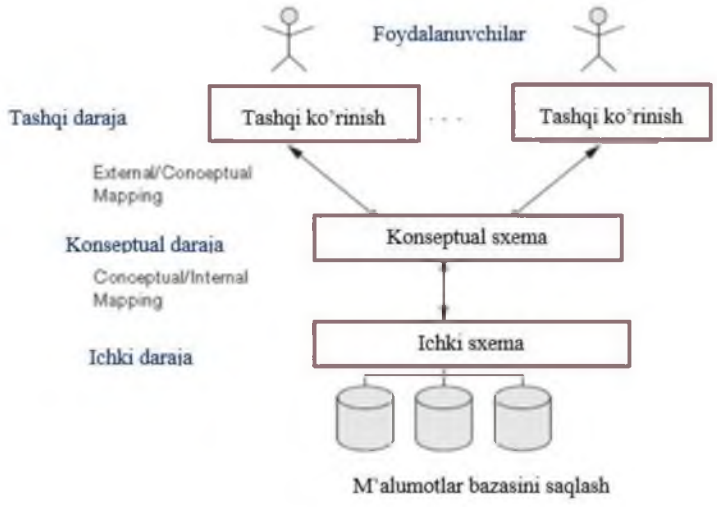
Yangi ma’lumotlar bazasini aniqlaganimizda, biz uning ma’lumotlar bazasi sxemasini faqat MBBTga belgilaymiz. Bu vaqtda tegishli ma’lumotlar bazasi holati ma’lumotlarsiz bo‘sh holatdir. Biz ma’lumotlar bazasining dastlabki holatini ma’lumotlar bazasi birinchi marta to‘ldirilganda yoki dastlabki ma’lumotlar bilan yuklanganda olamiz. Shu vaqtdan boshlab, har safar ma’lumotlar bazasiga yangilash operatsiyasi qo‘llanilganda, biz boshqa ma’lumotlar bazasi holatini olamiz. Vaqtning istalgan nuqtasida ma’lumotlar bazasi joriy holatga ega bo‘ladi. Shunday qilib, MBBT uchun to‘g‘ri sxemani ko‘rsatish juda muhim va sxema juda ehtiyotkorlik bilan ishlab chiqilishi kerak. MBBT sxema konstruksiyalari va cheklashlarning tavsiflarini saqlaydi, ular MBBT katalogidagi meta-ma’lumotlar deb ham ataladi, shuning uchun MBBT dasturiy ta’minoti kerak bo‘lganda sxemaga murojaat qilishi mumkin.

2.2. Ma’lumotlar bazasini uch bosqichli arxitekturasi

3 bosqichli arxitekturasi, ma’lumotlar bazasining turli darajalardagi tavsiflarini taqdim etadi. U foydalanuvchilar va dasturlar uchun ma’lumotlarni tushunishni oson-

lashtirishga yordam beradi.

Shaklda ko‘rsatilgan uch sxemali arxitekturaning maqsadi foydalanuvchi ilovalarini jismoniy ma’lumotlar bazasidan ajratishdir. Ushbu arxitekturada sxemalar quyidagi uchta darajada aniqlanishi mumkin:



2.3-rasm. 3-bosqichli arxitektura

1. Ichki daraja(fizik daraja) — bu daraja ma’lumotlar bazasining fizik saqlashini va ichki tuzilishini aks ettiradi. U fizik ma’lumotlar modelini, ma’lumotlarni qanday saqlanishi va ularga qanday kirish yo‘llari borligini batafsil bayon qiladi. Bu daraja ma’lumotlar bazasining qanday ishlashini tushunishga yordam beradi.

2. Konseptual daraja - ma’lumotlar bazasining umumiy tuzilishini taqdim etadi, bu foydalanuvchilar uchun muhimdir. Konseptual sxema, fizik saqlash tizimlari haqida tafsilotlarni yashiradi va ma’lumotlar bazasidagi ob’ektlar (entities), ma’lumot turlari (data types), ularning o‘zaro munosabatlari (relationships), foydalanuvchi tomonidan bajariladigan operatsiyalar (user operations) va cheklovlar (constraints) haqida ma’lumot beradi. Bu ma’lumotlar bazasining murakkab jihatlarini oson tushunarli shaklda ko‘rsatadi.

3. Tashqi daraja (foydalanuvchi darajasi) - ma’lumotlar bazasidagi foydalanuvchilarning turli ko‘rinishlarini ifodalaydi. Ushbu darajada bir necha tashqi sxemalar yoki foydalanuvchi ko‘rinishlari mavjud. Har bir tashqi sxema ma’lumotlar

bazasining foydalanuvchilar guruhiga qiziqarli bo‘lgan qismiga taalluqli bo‘lib, ushbu guruhga boshqa qismni ko‘rsatmaydi.

Misol uchun, bir tashkilotda xaridlar bo‘limi ma’lumotlar bazasida faqat o‘zlariga kerakli ma’lumotlarni ko‘rishi mumkin, masalan, mahsulotlar va narxlar haqida. Boshqa bo‘limlar esa o‘zlariga kerakli ma’lumotlarni ko‘rishadi, lekin har bir bo‘lim boshqa bo‘limlarning ma’lumotlarini ko‘rmaydi.

Tashqi sxemalar odatda ma’lumotlar bazasining konseptual darajasidan olingan yuqori darajadagi ma’lumotlar modelidan foydalanib amalga oshiriladi. Bu sxemalar foydalanuvchilarni ma’lumotlarni qulay va tushunarli tarzda ko‘rishga imkon beradi.

Uch-sxemali arxitektura — bu foydalanuvchiga ma’lumotlar bazasi tizimidagi sxema darajalarini tasavvur qilish uchun qulay vositadir. Ko‘pgina MBBTlari uchta darajani to‘liq va aniq ajratmaydi, lekin ular uch-sxemali arxitekturaning ayrim jihatlarini qo‘llab-quvvatlaydi. Ba’zi eski MBBTlari fizik darajadagi tafsilotlarni konseptual sxemada o‘z ichiga olishi mumkin. Uch darajali ANSI arxitekturasi ma’lumotlar bazasi texnologiyasi rivojlanishida muhim ahamiyatga ega, chunki u foydalanuvchilarning tashqi darajasini, ma’lumotlar bazasining konseptual darajasini va ichki darajasini aniq ajratadi. Bu, hatto bugungi kunda ham MBBTlarini loyihalashda keng qo‘llaniladi. Foydalanuvchi ko‘rinishlarini qo‘llab-quvvatlaydigan ko‘pgina MBBTlarida tashqi sxemalar konseptual darajadagi ma’lumotni tavsiflaydigan bir xil ma’lumotlar modelida belgilangan (masalan, Oracle yoki SQLServer kabi relatsion DBMS SQL-dan foydalanadi).

Haqiqiy ma’lumotlar faqat fizik darajada saqlanadi. Uch-sxemali arxitekturadagi har bir foydalanuvchilar guruhi o‘zining tashqi sxemasiga murojaat qiladi. Shuning uchun, MBBTlari tashqi sxemada ko‘rsatilgan so‘rovni konseptual sxemaga, keyin esa ichki sxemaga o‘tkazish jarayonini bajarishi kerak. Agar so‘rov ma’lumotlar bazasidan ma’lumotlarni olish bo‘lsa, saqlangan ma’lumotlardan chiqarilgan ma’lumotlar foydalanuvchining tashqi ko‘rinishiga mos keladigan tarzda qayta formatlanishi kerak.

Darajalar o‘rtasida so‘rovlar va natijalarni o‘zgartirish jarayonlari "mapping" deb ataladi. Ushbu mapping jarayonlari vaqt talab qilishi mumkin, shuning uchun ba’zi MBBTlar — ayniqsa kichik ma’lumotlar bazalarini qo‘llab-quvvatlash uchun mo‘ljallanganlari — tashqi ko‘rinishlarni qo‘llab-quvvatlamaydi. Biroq, bunday tizimlarda ham konseptual va ichki darajalar o‘rtasida so‘rovlarni o‘zgartirish zarurati mavjud.

2.3. Ma’lumotlarni fizik va mantiqiy tavsifi

Uch bosqichli arxitekturani ma’lumotlar mustaqilligi tushunchasini tushuntirish uchun qo‘llash mumkin. Ma’lumotlar mustaqilligi, ma’lumotlar tizimining biror darajadagi sxemasini o‘zgartirganda, undan yuqoridagi darajadagi sxemani o‘zgartirishga hojat bo‘lmagan imkoniyat sifatida ta’riflanadi. Biz ma’lumotlar mustaqilligini ikki turga bo‘lib tushuntirishimiz mumkin:

1. **Mantiqiy ma’lumotlar mustaqilligi** - bu kontseptual (mantiqiy) sxemani o‘zgartirgan holda, tashqi sxemalar yoki dasturiy ilovalarni o‘zgartirishga hojat bo‘lmagan imkoniyatdir. Kontseptual sxemani kengaytirish uchun (yangi yozuv turi yoki ma’lumotlar elementi qo‘shish), cheklovlarni o‘zgartirish yoki bazani kamaytirish uchun (yozuv turi yoki ma’lumotlar elementini olib tashlash) o‘zgartirishlar qilinishi mumkin. Agar faqat qolgan ma’lumotlarga murojaat qiladigan tashqi sxemalar mavjud bo‘lsa, ular ta’sir qilmaydi.
2. **Fizik ma’lumotlar mustaqilligi** - bu ichki sxemani o‘zgartirgan holda, kontseptual sxemani o‘zgartirishga ehtiyoj bo‘lmagan imkoniyatdir. Shu bilan birga, tashqi sxemalar ham o‘zgartirilmasligi kerak.

2.4. Ma’lumotlar bazasini boshqarish tizimini tashkil etuvchilari

Ma’lumotlar bazasini boshqarish tizimi (DBMS) – bu ma’lumotlarni tashkil etish, saqlash, boshqarish va ularga tezkor kirishni ta’minlovchi dasturiy vositalar majmuasidir. DBMSning samarali ishlashi uchun uning bir necha asosiy tarkibiy qismlari mavjud. Quyida DBMSni tashkil etuvchi asosiy komponentlar keltiriladi:

1. Ma’lumotlar bazasi (Database)
2. Ma’lumotlar bazasi boshqaruv tizimi (DBMS Engine)
3. Ma’lumotlar bazasining foydalanuvchi interfeysi (User Interface)
4. Ma’lumotlarni aniqlash tillari (Data Definition Language - DDL)
5. Ma’lumotlarni boshqarish tillari (Data Manipulation Language - DML)
6. Xavfsizlik va ruxsatnoma mexanizmi (Security and Authorization)
7. Ma’lumotlar so‘rov tili (Query Language)
8. Ma’lumotlar ombori (Data Storage and Retrieval Subsystem)
9. Tranzaksiya boshqaruvi (Transaction Management)

2.5. Amaliy qism

Ma’lumotlar bazasining uch bosqichli arxitekturasi (MB uch darajali arxitekturasi) real hayotdagi bir nechta holatlarda namuna sifatida ko‘rsatib berilishi mumkin. Quyida ushbu arxitektura bosqichlariga oid oddiy misolni keltiramiz:

Misol: Universitet ma’lumotlar bazasi

1. Tashqi (Foydalanuvchi) daraja:

Bu daraja foydalanuvchilar (talabalar, o‘qituvchilar) ko‘radigan ma’lumotlarni ifodalaydi. Har bir foydalanuvchi guruhi faqat o‘zlariga kerakli ma’lumotlarni ko‘radi.

- Masalan, **talabalar** o‘z baholari, kurslari va ro‘yxatdan o‘tgan darslar va h.k haqida ma’lumotni ko‘radi.
- **O‘qituvchilar** esa o‘zlariga bog‘liq kurslardagi talabalar ro‘yxati, imtihon natijalari kabi va h.k ma’lumotlarni ko‘radilar.
- **Universitet administratori** esa barcha fakultetlar va talabalar haqida to‘liq ma’lumotga ega bo‘lishi mumkin.

Har bir foydalanuvchi o‘zining tashqi sxemasini ko‘radi va boshqa foydalanuvchilar foydalanadigan ma’lumotlarga kirish imkoniyatiga ega emas.

2. Kontseptual (Conceptual) daraja:

Bu daraja ma’lumotlar bazasining umumiy tuzilishini, ya’ni butun universitet tizimini ifodalaydi. Bunda barcha ob’ektlar va ularning o‘zaro munosabatlari saqlanadi.

Masalan, universitetda **talabalar**, **kurslar**, **fakultetlar**, **o‘qituvchilar** va **imtihon natijalari** kabi asosiy ma’lumotlar mavjud bo‘lib, ularning barchasi bog‘lanadi. Talabalar o‘qituvchilarga, kurslarga va fakultetlarga bog‘langan, imtihon natijalari esa talabalar va kurslar o‘rtasidagi munosabatni aks ettiradi.

Bu darajada ma’lumotlar fizik saqlash tafsilotlarisiz umumiy tuzilmasi ko‘rsatiladi.

3. Ichki (Fizik) daraja:

Bu daraja ma’lumotlar qanday saqlanishini, fizik darajasidagi tafsilotlarni ko‘rsatadi. Bu darajada ma’lumotlar bazasi fayllari, indekslar va saqlash joylari haqida ma’lumotlar beriladi.

Masalan, talabalar haqida ma’lumotlar qaysi fayllarda, qanday indekslar orqali tezkor qidirish va olish mumkinligi, ma’lumotlar qanday zichlanishi (compress), qanday saqlanishi va joylanishi ko‘rsatiladi.

Ushbu arxitektura qanday ishlaydi?

Agar bir talaba o‘z baholarini ko‘rmoqchi bo‘lsa, so‘rovni tashqi darajadan boshlaydi. DBMS bu so‘rovni kontseptual darajaga tarjima qiladi va talabalar, baholar va kurslar o‘rtasidagi munosabatlarni aniqlaydi. So‘ngra, ichki darajaga o‘tadi va ma’lumotlar bazasidan bu ma’lumotlarni olib beradi. Shu orqali talaba faqat o‘ziga tegishli bo‘lgan baholarni ko‘radi, lekin boshqa talabalar yoki fakultet ma’lumotlarini ko‘ra olmaydi.

Nazorat savollari:

1. Uch bosqichli arxitektura nima?
2. Konseptual daraja nima?
3. Foydalanuvchi daraja nima?
4. Fizik daraja nima?

5. Konseptual bilan fizik daraja o‘rtasida qanday farqlar bor?

6. Ma’lumotlar modeli nima?

3 BOB. MA'LUMOTLAR BAZASI MODELLARI VA MOHIYAT-ALOQA MODELI

Tayanch so'zlar: *mohiyat, atribut, munosabat, ma'lumotlar bazasi sxemasi, ichki daraja, konseptual daraja, tashqi daraja*

3.1. Ma'lumotlar bazasi modellari. Ierarxik ma'lumotlar modeli

Ma'lumotlar bazasi modellari — bu ma'lumotlar qanday tashkil etilganini, saqlanishini va boshqarilishini tavsiflaydigan konseptual tuzilmalar yoki usullardir. Ular ma'lumotlar tuzilishini, ularning o'zaro bog'liqligini va ularga qanday kirish va ishlov berish kerakligini aniqlab beradi. Har bir model ma'lumotlar bazasini yaratish va undan foydalanishda alohida yondashuv va mexanizmlarni taqdim etadi. Ma'lumotlar bazasi modellari asosan quyidagi turlarga bo'linadi:

Ierarxik ma'lumotlar modeli dastlab Cobol tilining ma'lumotlar tuzilmalarini umumlashtirish natijasida paydo bo'lgan. Iyerarxik modellarda ma'lumotlarni taqdim etishning asosiy tuzilishi daraxt shaklida bo'ladi. Iyerarxiyaning eng yuqori (birinchi) darajasi daraxtning ildizi deb ataladi. Ushbu ildiz ikkinchi sathdagi tugunlar bilan, ikkinchi sathdagi tugunlar uchinchi sathdagi tugunlar bilan ulanishlarga ega va hokazo. Bir xil sathdagi tugunlar o'rtasida hech qanday aloqa yo'q. Shunday qilib, iyerarxik tuzilishdagi ma'lumotlar teng emas - ba'zilari boshqalarga qat'iy ravishda bo'ysunadi. Ushbu model **ildiz, sath, tugun, bog'lanish** kabi parametrlar bilan tavsiflanadi. Uning ishlash tamoyili shundayki, quyi sathdagi bir nechta tugunlar bog'lanish yordamida yuqoriroq sathdagi bitta tugun bilan bog'langan bo'ladi.

Ierarxik ma'lumotlar bazasi modelida quyidagi tushunchalar mavjud:

- ✓ **Ildiz (Root)**: Ierarxik modeldagi eng yuqori darajadagi tugun. Ildiz tuguni butun tuzilmaning boshlanish nuqtasi hisoblanadi va boshqa barcha tugunlar undan tarqaladi.
- ✓ **Sath (Level)**: Ierarxik modelda sath yoki daraja ma'lumotlarning joylashuvini ko'rsatadi. Ildiz tuguni birinchi sathda joylashgan bo'lsa, undan keyingi tugunlar

pastki sathlarda joylashgan bo‘ladi. Masalan, ikkinchi sathda “bolalar” tugunlari, uchinchi sathda esa ularning “nabiralari” joylashgan bo‘lishi mumkin.

- ✓ **Tugun (Node)**: Ierarxik modelning har bir elementi tugun deb ataladi. Har bir tugun ma’lum bir ma’lumotni ifodalaydi va boshqa tugunlar bilan bog‘lanishi mumkin.
- ✓ **Bog‘lanish (Relationship/Connection)**: Bog‘lanishlar tugunlar orasidagi aloqalarni ifodalaydi. Ierarxik modelda har bir tugun faqat bitta yuqori darajadagi tugun bilan bog‘langan bo‘lishi mumkin, bu esa daraxt strukturasini hosil qiladi.

Misol uchun, internet-provayder ma’lumotlar bazasi haqida o‘ylaylik. Bunda eng yuqori darajadagi ma’lumot “Mijozlar” bo‘lishi mumkin. Mijozlar haqida ma’lumotlar, masalan, “Ism”, “Manzil”, “Telefon” kabi maydonlar orqali ifodalanadi. Shu ma’lumotlar ostida esa “Xizmatlar” va “Arizalar” degan yozuvlari joylashgan bo‘lishi mumkin. “Xizmatlar” sathi provayder tomonidan mijozlarga ko‘rsatilgan xizmatlar haqida ma’lumot saqlaydi, “Arizalar” sathi esa mijozlar tomonidan kiritilgan arizalar haqida ma’lumotlarni o‘z ichiga oladi.

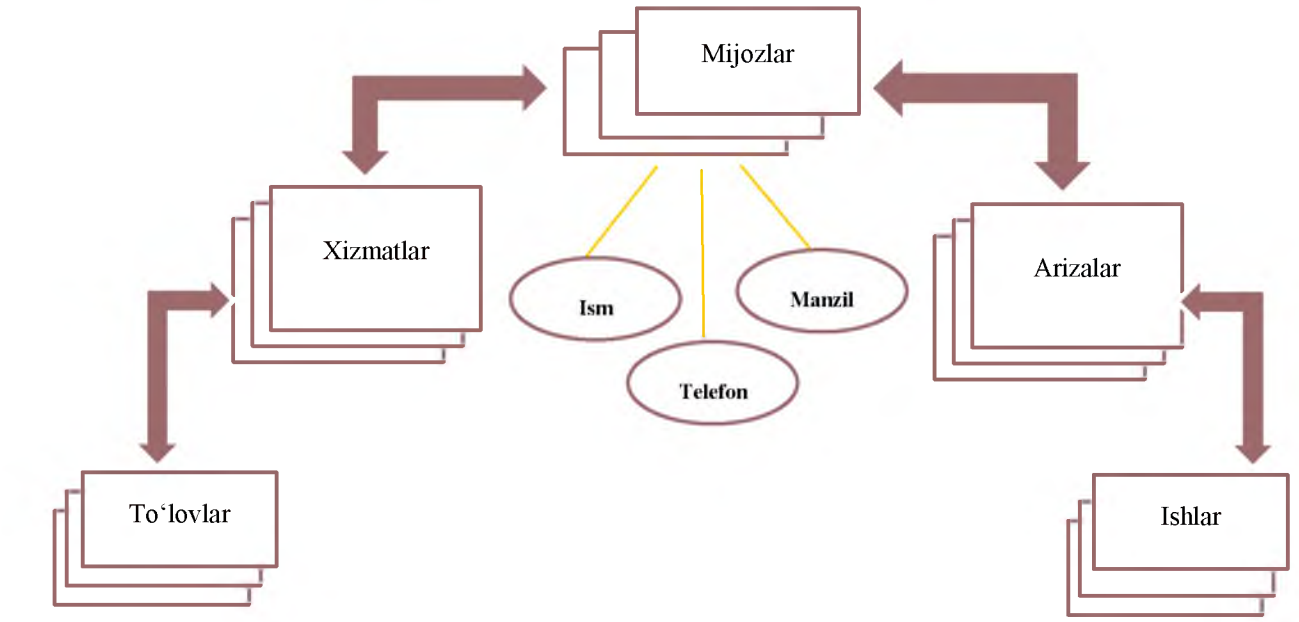
Afzalliklari:

- **Oddiylik va tushunarli tuzilma**: Daraxt shakli juda intuitiv va aniq.
- **Tabiiy ravishda ierarxik munosabatlar**: Masalan, texnik qurilmaning qismlari yoki internet-provayder mijozlari va ularning xizmatlari o‘rtasidagi munosabatlar.

Kamchiliklari:

- **Moslashuvchan emas**: Ierarxik tuzilma ko‘plab real dunyo vaziyatlariga mos emas.
- **Yangilanish murakkab**: Agar ma’lumotlar tuzilmasi o‘zgartirilsa, butun daraxtni qayta qurish kerak bo‘lishi mumkin.

Shunday qilib, bu model o‘ziga xos tabiiy ierarxik munosabatlarga ega sohalarda yaxshi ishlaydi, lekin moslashuvchanlik va murakkab strukturalarga moslashish qobiliyati cheklangan.



3.1-rasm. Mijozlar maydonini ierarxik tuzulishi

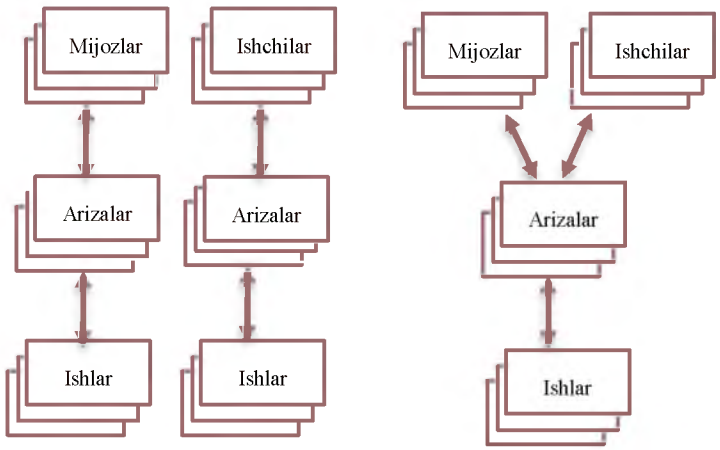
Shu bilan bir qatorda, “Xizmatlar” sathi “To‘lovlar” sathi bilan bog‘lanishi mumkin, bunda har bir nusxa provayder tomonidan mijozga ko‘rsatilgan xizmat uchun olingan to‘lovlar haqidagi ma’lumotlarni o‘z ichiga oladi. “Arizalar” sathi esa mijozlarning so‘rovlariga ko‘ra provayder tomonidan bajarilgan ishlarni ifodalovchi “Ishlar” sathi bilan bog‘langan.

Ierarxik modelning asosiy afzalligi — qidiruv algoritmlarining yuqori samaradorligi hisoblanadi. Ma’lumotlarni daraxt tuzilmasida izlash juda tez amalga oshadi, chunki har bir qadam daraxtning chuqurligiga bog‘liq. Daraxt tuzilmasining chuqurligi deganda, ierarxik modelda ma’lumotlarni qidirishda yoki ishlov berishda eng yuqori darajadagi tugundan (ildiz) eng pastki darajadagi tugungacha (barg tugunlari) bo‘lgan qadamlar soni nazarda tutiladi. Daraxtning chuqurligi qanchalik

kichik bo‘lsa, ma’lumotlarni qidirish yoki ularga murojaat qilish jarayoni shuncha tez bo‘ladi. Shu sababli, ierarxik modelda qidiruv algoritmlarining samaradorligi, asosan, daraxtning chuqurligi bilan bog‘liq.

Biroq, bu modelning kamchiliklari ham bor: ma’lumotlarni qo‘shish va o‘chirish jarayonlari juda murakkab va qimmat, chunki ular daraxt tuzilmasini o‘zgartirishni talab qiladi. Bundan tashqari, ba’zi hollarda ma’lumotlarni saqlashda ularni takrorlashga to‘g‘ri keladi. Ierarxik modelda ma’lumotlarni takrorlash deganda, bir xil ma’lumotlarning bir necha joyda saqlanishi nazarda tutiladi. Masalan, agar internet-provayderning ierarxik ma’lumotlar bazasida xodimlar o‘rtasida ishlarni taqsimlash kerak bo‘lsa, u holda bir xil ma’lumotlarni bir nechta joyda takrorlash kerak bo‘ladi. Bu esa ma’lumotlar bazasida dublikatlarni keltirib chiqaradi.

Soddaroq qilib aytganda, ierarxik modelda ma’lumotlarni saqlash va yangilashning qiyinligi va ma’lumotlarni takrorlash zarurati asosiy muammolardir.



a) Ierarxik model b) Ierarxiyadan voz kechish

3.2-rasm. Ierarxik modelda ma’lumotlarning takrorlanishi

Agar ierarxik model bilan ishlashda ma’lumotlar tuzilmasida muammolar paydo bo‘lsa, masalan, bitta ma’lumot bir nechta joyda takrorlanishi kerak bo‘lsa, unda radikalroq yechim sifatida ierarxik modeldan voz kechib, tarmoq modelidan foydalanish tavsiya etiladi. Tarmoq modeli ma’lumotlarni saqlashda yanada

moslashuvchanroq, chunki unda bitta element bir nechta yuqoridagi sathlar bilan bog‘lanishi mumkin.

Bu shuni anglatadiki, tarmoq modeli yordamida ma’lumotlar tuzilmasini optimallashtirish va takrorlanishni kamaytirish imkoniyati mavjud.

3.2. Tarmoqli ma’lumotlar modeli

1969-yilda ma’lumotlar tizimlari tillari konferensiyasi (Conference on Data Systems Languages — CODASYL) tomonidan tarmoq modeli ishlab chiqildi. CODASYL modeli ma’lumotlar bazasini boshqarishda ko‘plab murakkab aloqalarni ifodalash imkonini beradi, lekin shu bilan birga, bunday tuzilmani saqlash va qidirish usullarini dasturiy ta’minlash ham qiyin.

CODASYL spetsifikatsiyasi ma’lumotlarning kontseptual, mantiqiy va fizik modellarini birlashtiradi. Ushbu spetsifikatsiyada ma’lumotlar bazasining mantiqiy sxemasini aniqlash tili, ma’lumotlar bilan ishlash tili va tashqi ma’lumot tashuvchilarini boshqarish tili mavjud. Ma’lumotlar bazasining mantiqiy sxemasini aniqlash tili, ma’lumotlarning strukturasini, ularning o‘zaro aloqalarini va qanday qilib saqlanishi kerakligini belgilashga imkon beradi. Bu til foydalanuvchilarga ma’lumotlar bazasining logic ko‘rinishini yaratishga yordam beradi.

Ma’lumotlar bilan ishlash tili esa, foydalanuvchilarga ma’lumotlarga kirish, o‘zgartirish, o‘chirish va saqlash kabi operatsiyalarni bajarishga imkon beradi. Ushbu til yordamida foydalanuvchilar ma’lumotlarni samarali va tezkor tarzda boshqarishi mumkin.

Tashqi ma’lumot tashuvchilarini boshqarish tili, ma’lumotlar bazasi bilan tashqi tizimlar o‘rtasida ma’lumotlar almashinuvi va o‘zaro aloqalarni boshqarish uchun mo‘ljallangan. Ushbu til ma’lumotlarni import qilish, eksport qilish va tashqi manbalardan ma’lumotlarni olish jarayonlarini avtomatlashtirish imkoniyatini yaratadi.

Bu spetsifikatsiya CODASYL modeli doirasida ma’lumotlar bazasining samarali boshqarilishi va uzluksiz o‘zgarishlarga moslashuvchanligini ta’minlaydi, shuningdek,

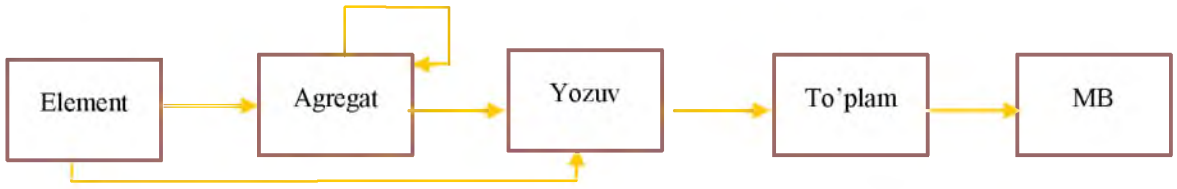
foydalanuvchilarga turli xil vazifalarni bajarishda qulaylik yaratadi. Bunda ushbu til va funksiyalar ma'lumotlar bazasining mustahkamligi va foydalanuvchilar uchun qulayligini ta'minlashga yordam beradi.

Ma'lumotlar bazasining mantiqiy sxemasini fayl tuzilmasi bilan bog'lash uchun "ma'lumotlar bazasi sohasi" tushunchasi qo'llaniladi. Shuningdek, foydalanuvchi ko'rinishlarini aniqlash uchun "subsxema" tushunchasi qo'llaniladi.

CODASYL modeli ikki asosiy komponentga ega:

1. **Yozuv** — bu ma'lumotlar obyektlarini tavsiflovchi xususiyatlar yig'indisini ifodalaydi.
2. **Yozuvlar to'plami** — ikki darajali grafik bo'lib, u haqiqiy obyektlar orasidagi ba'zi ierarxik munosabatlarni ifodalaydi.

Yozuvning maydoni ham muhim komponent hisoblanadi va u oddiy element yoki boshqa elementlar yig'indisidan iborat bo'lishi mumkin. CODASYL modeli murakkab ma'lumotlarni samarali boshqarishga yordam beradi.



3.3-rasm. CODASYL modeliga namuna

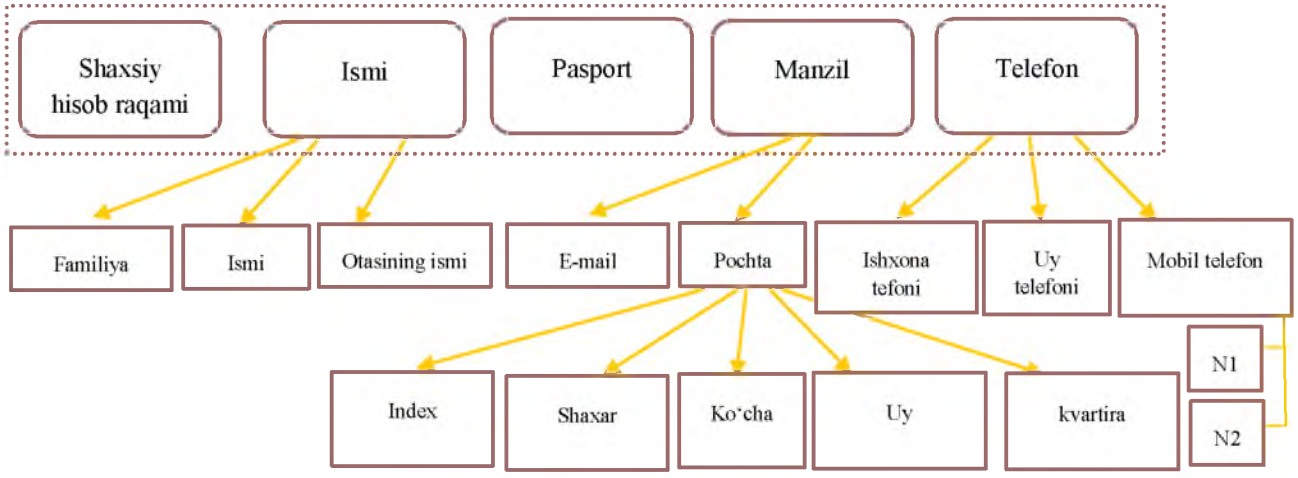
Quyidagi sxemada ko'rsatilgan ma'lumotlar bazasi tuzilishi **CODASYL** (Conference on Data Systems Languages) tarmoq modeliga asoslanadi. Bu modelning tuzilishi quyidagi asosiy komponentlardan iborat:

1. **Element**: Oddiy ma'lumotlar, xususiyatlar yoki qiymatlar. Ular ma'lumotlar bazasining asosiy birligi sifatida xizmat qiladi.
2. **Agregat**: Elementlar to'plami. Agregat bir yoki bir nechta elementni o'z ichiga oladi va ma'lum bir ob'ekt yoki xususiyatni ifodalaydi.

3. **Yozuv (Запись)**: Agregatlardan tashkil topgan ma’lumotlar blokidir. Har bir yozuv biron bir ma’lum bir ob’ekt yoki sub’ekt haqida ma’lumot saqlaydi.
4. **To‘plam (Набор)**: Bir nechta yozuvlar to‘plami. U ma’lumotlar bazasidagi ma’lumotlarning yig‘indisini ifodalaydi.
5. **Ma’lumotlar bazasi (База данных)**: Barcha ma’lumotlar va yozuvlar bir joyda to‘plangan joy. U yuqoridagi barcha komponentlarning yig‘indisidir.

Bu tuzilish orqali ma’lumotlar bazasidagi ma’lumotlarni boshqarish va saqlash jarayonlari osonlashadi va samaradorlik oshadi. CODASYL modeli, shuningdek, ma’lumotlarning murakkab aloqalarini tashkil etishga imkon beradi, bu esa ma’lumotlarni yanada samarali saqlash va olish imkonini beradi.

Quyida, internet-provayderlar ma’lumotlar bazasida “Mijoz” yozuvi quyidagi beshta maydondan iborat: “Shaxsiy hisob raqami” va “Pasport” — atomar elementlar, “Ism”, “Manzil” va “Telefon” esa agregat turidagi takrorlanadigan guruhlar.



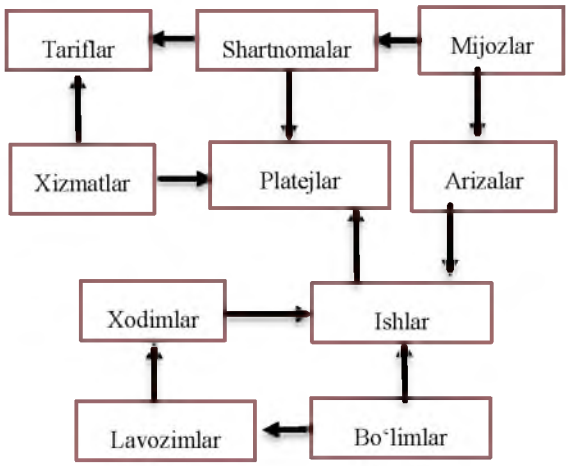
3.4-rasm. CODASYL modeliga asoslangan Mijoz yozuvi

Bu yozuv strukturasi, ma’lumotlar bazasiga oddiy so‘rov yuborish orqali mijozning barcha ma’lumotlarini olish imkonini beradi va agregatlangan maydonlarning alohida elementlariga kirishni ta’minlaydi. Masalan, ma’lum bir shahar yoki ko‘chada yashovchi mijozlar ro‘yxatini olish yoki qarzi bor mijozlarga avtomatik telefon qo‘ng‘irog‘i qilish imkoniyatiga ega bo‘lish mumkin.

Shuningdek, ushbu mijozlarga birgalikda pochta xatlarini yuborish, xatlarda har bir mijozga ismi va otasining ismi bilan murojaat qilish kabi funksiyalarni amalga oshirish mumkin. Misol uchun, to‘lovlarni kechiktirgan mijozlarga yuboriladigan xatlarda, majburiyatlarni bajarish zarurligini va eslatib o‘tish mumkin.

“Pasport” maydoni atomar element sifatida ko‘rsatilgan, lekin haqiqiy pasport ma’lumotlari ko‘proq murakkab tuzilmani talab qiladi. Bu holda, “seriya - raqam - berilgan sana - beruvchi joy” kabi agregatlarni yaratish mumkin edi. Bu maydon atomar sifatida tanlangan bo‘lishi mumkin, chunki dasturchi foydalanuvchilar so‘rovlarini tahlil qilganida, pasport ma’lumotlari faqat mijozning shartnomasi uchun kerak bo‘lgan ma’lumot sifatida olingan.

Charlz Bakhman (C. W. Bachman) tomonidan CODASYL spetsifikatsiyasi asosida ma’lumotlarning tarmoq modelini grafik tarzda ko‘rsatish uchun “Bakhman diagrammasi” ishlab chiqildi. Bu diagramma dasturchiga yozuvlar tuzilmasini, parametrlarini va ob’ektlar o‘rtasidagi munosabatlarni ko‘rish imkonini beradi.
Ushbu matnda CODASYL modeli va uning diagrammalari, jumladan, internet-provayderlari ma’lumotlar bazasining tarmoq modelining oddiy ko‘rinishi haqida batafsil ma’lumot berilgan.



3.5-rasm. Bakhman diagrammasiga namuna

“Xizmatlar→Tariflar” internet-provayderining narxlar ro‘yxatini ifodalaydi. Har bir “Xizmatlar” yozuv egasi bir yoki bir nechta “Tariflar” yozuv a’zolari bilan

bog‘lanadi. Provayderning mijozlar bazasi “Mijozlar→Shartnomalar” to‘plami orqali aks ettiriladi, shartnomalarning mazmuni esa “Shartnomalar→Tariflar” to‘plamida joylashgan. “Tariflar” yozuvi esa ikki to‘plamning a’zosi bo‘lib, u biror shartnoma doirasida taqdim etilayotgan xizmat tarifini bildiradi.

“Mijozlar→Arizalar” to‘plami mijozlardan qabul qilingan arizalar jurnalini taqdim etadi, “Arizalar→Ishlar” to‘plami esa arizalarni bajarish uchun zarur bo‘lgan ishlar tarkibini belgilaydi. “Bo‘limlar→Lavozimlar” va “Lavozimlar→Xodimlar” to‘plamlari bo‘limlar uchun shtat jadvalini aks ettiradi, “Xodimlar→Ishlar” to‘plami esa arizalar bo‘yicha ishlarni xodimlar o‘rtasida taqsimlaydi.

“To‘lovlar” yozuvi esa uchta turli to‘plamning a’zosi bo‘lib, bu yozuv mijozga taqdim etilgan xizmatlar yoki arizalar bo‘yicha bajarilgan ishlar uchun to‘lovni aks ettiradi.

CODASYL modelida yozuvlarni saqlash va olish uchun ikkita metod mavjud: to‘g‘ridan-to‘g‘ri kirish (xesh algoritmidan foydalangan holda) va bog‘langan ro‘yxatlar metodi (ko‘rsatkichlar tizimidan foydalanib). Yozuvlarning formatini boshqarish va ma’lumotlarga kirishni tezlashtirish imkoniyatlari mavjud.

1970-yilda CODASYL tarmoq modelida IDMS (Integrated Database Management System) ma’lumotlar bazasi boshqaruv tizimi ishlab chiqilgan. Ushbu tizim ko‘plab korporativ ma’lumotlarni qayta ishlash tizimlari uchun platforma vazifasini o‘ynadi. CODASYL modeliga asoslangan ma’lumotlar bazalari o‘sha davrda rivojlanayotgan relatsion tizimlarga nisbatan samaradorlik va ma’lumotlarni saqlash jihatidan yuqori ko‘rsatkichlarga ega edi.

3.3. Relyatsion ma’lumotlar modeli

Relyatsion ma’lumotlar modeli haqida nazariy tadqiqotlar 1960–1970-yillarda Edgar F. Codd tomonidan olib borilgan va keyinchalik Chris Date va Hugh Darwen tomonidan davom ettirilgan. Ushbu mavzu bo‘yicha relyatsion modelni qo‘llash nazariyasi va amaliyotiga oid ko‘plab maqolalar mavjud. Keling, relyatsion

yondashuvning asosiy g‘oyalariga to‘xtalib o‘tamiz. Ushbu g‘oyalar dastlab relyatsion tizimlarning ierarxik va tarmoqli ma’lumotlar bazalariga jiddiy raqobat qilishiga va keyinchalik ularni deyarli to‘liq siqib chiqarishga sabab bo‘ldi.

CODASYL modeli dasturchilik yechimlariga asoslangan bo‘lsa, relyatsion model aniq matematik asosga — matematik mantiq va to‘plamlar nazariyasiga ega. Coddning fundamental tayyorgarligi va bilimlari unga ma’lumotlar bazasini munosabatlar (relationships) to‘plami sifatida ko‘rish imkonini berdi. Bu munosabatlarga to‘g‘ridan-to‘g‘ri matematik mantiq tillari va tushunchalari qo‘llanib, ular ustida relyatsion algebra operatsiyalarini bajarish mumkin edi (relyatsion algebra ham Codd tomonidan ishlab chiqilgan va to‘plam nazariyasiga oid operatsiyalarni o‘z ichiga oladi).

Relyatsion ma’lumotlar bazasi o‘zaro bog‘langan, nomlangan munosabatlardan tashkil topgan to‘plamdir. Munosabat — bu haqiqiy ob’ektning (“entity”) axborot modelidir, u bir turdagi tuzilmali yozuvlar to‘plami sifatida taqdim etiladi. Munosabatning yozuvi modellashtirilayotgan ob’ektning misolini ifodalaydi, uning xususiyatlari mos keluvchi atributlar (maydonlar) qiymatlari bilan aniqlanadi.

Agar munosabat yozuvlari o‘rtasida bog‘liqlik mavjud bo‘lsa, bu bog‘liqlik “tashqi kalitlar” mexanizmi orqali amalga oshiriladi. Tashqi kalitlar aslida bir necha munosabatlar yozuvlari atributlariga bog‘lanadigan havolalardir.

Relyatsion modelda munosabatning yozuvi CODASYL modelidagi yozuvga o‘xshash, atribut esa yozuvning maydoniga o‘xshaydi. Farqi shundaki, relyatsion modelda atributlarning ichki tuzilmasiga yo‘l qo‘yilmaydi — ular barchasi atomar bo‘lishi kerak. Bu soddalashtirish relyatsion ma’lumotlar bazalarini to‘g‘ri burchakli jadval sifatida tasavvur qilishga imkon beradi, yozuvlarni esa ushbu jadvallarning qatorlari sifatida ko‘rish mumkin, bu esa ma’lumotlar bazasining tuzilishini tushunishni oddiy foydalanuvchilarga ham oson qiladi.

Relyatsion ma’lumotlar modeli ierarxik va tarmoqli modellar bilan taqqoslaganda informativlik va ma’lumotlarga murajaat qilish tezligi bo‘yicha sekin ishlashi mumkin bo‘lsa-da, uning asosiy afzalligi — ma’lumotlar bazasining tuzilishini soddaligi va yuqori darajadagi texnologikligi, shuningdek, ma’lumotlar ustida o‘zgartirish amallarini samarali bajarishdir.

Dastlabki relyatsion ma’lumotlar bazasi boshqaruv tizimi (MBBT) IBM korporatsiyasi tomonidan ishlab chiqilgan System R edi. U 1970-yillarning o‘rtalarida tadqiqot loyihasi natijasida yaratildi. System R tarkibidagi ko‘plab texnologik yechimlar va algoritmlar keyinchalik tijoriy relyatsion MBBTlarni yaratishda ishlatildi. Shuningdek, System R birinchi bo‘lib SQL tilini qo‘llab-quvvatlagan MBBT edi.

Kompyuterlarning apparat ta’minoti kuchayishi, mini-kompyuterlarga o‘tish va hisoblash tizimlarida mijoz-server arxitekturasi qo‘llanilishi tufayli 1980-yillarning o‘rtalariga kelib relyatsion MBBTlarning yuqori texnologikligi ularni yanada ommalashtirdi va ularga nisbatan ilgari aytilgan tanqidlar ahamiyatsiz bo‘lib qoldi. Shu sababli relyatsion tizimlar dasturlashda noqulay va ekspluatatsiyada moslashuvchan bo‘lmagan ierarxik va tarmoqli CODASYL tizimlarini bozordan siqib chiqardi.

Relyatsion ma’lumotlar modeli to‘liq aniqlangan deb hisoblanadi, agar uning uchta asosiy komponenti aniqlangan bo‘lsa: tuzilmaviy, yaxlitlik va manipulyatsiya komponentlari.

1. **Tuzilmaviy komponent**: Ushbu komponent ma’lumotlarning mantiqiy darajasidagi tuzilmasini belgilaydi, ya’ni ER-modelidagi ob’ektlar va ularning o‘zaro bog‘liqligini tasvirlash uchun qanday ma’lumotlar tuzilmalaridan foydalanish mumkinligini aniqlaydi. Ushbu ma’lumotlar tuzilmalari til darajasida qo‘llab-quvvatlanishi va ularga nisbatan manipulyatsion komponentning metodlari qo‘llanilishi kerak.
2. **Yaxlitlik komponenti**: Bu komponent ma’lumotlar bazasi serverlari tomonidan qo‘llab-quvvatlanadigan va ma’lumotlar bazasi ob’ektlarining yangilanishi

vaqtida ularning mantiqiy izchilligi va mosligini avtomatik nazorat qilishni ta'minlaydigan yaxlitlik cheklovlarini o'z ichiga oladi.

3. **Manipulyatsiya komponenti**: Ushbu komponent foydalanuvchilarning ma'lumotlar bazasiga bo'lgan so'rovlarini amalga oshirish uchun ma'lumotlar tuzilmalariga nisbatan qo'llanilishi mumkin bo'lgan operatsiyalar to'plamini belgilaydi. Bu operatsiyalar orqali foydalanuvchilar ma'lumotlarni o'zgartirish, qo'shish yoki olish imkoniyatiga ega bo'ladilar.

3.4. Ma'lumotlar bazasini loyihalashda mohiyat - aloqa modeli. Mohiyat aloqa diagrammasini qurish

Ma'lumotlarning relyasion bazasi - bu o'zaro bog'langan munosabatlar, ya'ni jadvallar to'plamidir. Har qanday munosabat (jadval) kompyuterlarning xotirasida fayl ko'rinishda joylashtiriladi.

Muayyan tashkilot (korxona, bank, universitet, kutubxona va boshqalar) muammolarini hal qilish uchun axborot tizimi (AT) yaratiladi. AT yaratish va ishlatish uchun uning tavsifi talab qilinadi. ATning to'liq, keng qamrovli tavsifi nafaqat ATning o'zi, balki atrof-muhitni ham o'z ichiga olishi kerak, ya'ni predmet sohasining tavsifi bo'lishi kerak.

Umumiy mavzuni batafsil tavsifi umuman erkin shaklda berilishi mumkin. UML (Unified Modeling Language) mo'ljallangan tizimning mavhum modelini grafik tasvirlash uchun ishlatiladi. Biz bu tilni haddan tashqari mavhumligi va murakkabligi tufayli o'rganmaymiz.

Ma'lumotlar bazasini loyihalashtirishda relyasion model bilan ishlash ancha noqulayliklarga olib keladi. Shu sabab ma'lumotlar bazasini loyihalashda har xil semantik modellar ham ishlatiladi. Ulardan eng ko'p tarqalganlaridan biriga – **ER** modeli deyiladi. Bu model ingilizcha **"Entity-relation"** deyilib, ma'nosi **"Mohiyat-aloqa"** demakdir. Bu model 1976-yil Piter Chen tamonidan kiritilgan bo'lib u o'ziga bir qator grafik diagrammalarini oluvchi bir necha har xil turdagi komponentalarni

birlashtirgan. Piter Chen mohiyatlar to‘plami va ular orasida bog‘lanish sifatida relyasion ma’lumotlar strukturasini interpritatsiya qilishni taklif qildi. ER -modelining asosiy komponentalari mohiyat, bog‘lanish va atribut (xossa) bo‘lib hisoblanadi.

“Mohiyat – aloqa” modeli predmet sohani tashkil etuvchi uchta asosiy komponentalardan foydalanib quriladi: **mohiyat, atribut, aloqa.**

Mohiyat — bu real hayotdagi obyektning axborot modeli bo‘lib, loyiha doirasida muhim hisoblangan obyektning xususiyatlarini aks ettiradi. **Atribut** esa bu obyektning biror-bir xususiyatini ifodalovchi ma’lumot. **Aloqa** — bu obektlar orasidagi aloqa yoki bog‘lanish. **Relationship turi** obekt turlari o‘rtasidagi bog‘lanishni ifodalaydi.

Obyektlar to‘plami o‘xshash obyektlarning bir nechta namunalarini o‘z ichiga oladi va har bir obyekt ER-modelda bitta obyekt namunasi sifatida ko‘rsatiladi. Obyektning atributi esa shu obyektning xususiyatlariga mos keladigan qiymatlar to‘plamini ifodalaydi.

ER-modellar takrorlanishni oldini olish uchun har bir obyekt uchun ma’lumotlarni bir martagina saqlashni talab qiladi. Bu obyektlarning atributlari foydalanuvchilar uchun baza so‘rovlarini amalga oshirishda ko‘rsatiladigan natijalar sifatida xizmat qiladi.

Atributlar turlari

Atributlar ikki asosiy toifaga bo‘linadi:

1. **Tavsiflovchi atributlar** — bu obyektning xususiyatlarini ifodalovchi atributlar bo‘lib, ular so‘rov natijalarida foydalanuvchiga ko‘rsatiladi.
2. **Identifikatsion (kalit) atributlar** — bu obyektning namunalarini aniqlash uchun ishlatiladigan atributlar. Ular so‘rov natijalari uchun natija emas, balki ob’ektlarni tanlash vositasi sifatida ishlatiladi.

Birlamchi kalitlar

Har bir obyekt kamida bitta **birlamchi kalit**ga ega bo‘lishi kerak, bu obyektning barcha namunalarining unikal ekanligini ta’minlaydi. Birlamchi kalitlar har bir obyekt

namunasi uchun yagona va qaytarilmas qiymatga ega bo‘lib, bu qiymatlar orqali obyektlar aniqlanadi.

Tashqi(ikkilamchi) kalitlar

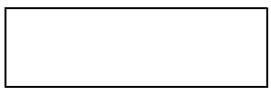
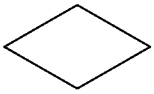
Tashqi kalitlar ma’lumotlar bazasida parametrik qidiruv uchun xizmat qiladi, ya’ni obyektlarning ba’zi atributlari bo‘yicha guruhlash va izlashni amalga oshirish imkonini beradi. Bu kalitlar foydalanuvchilar tomonidan amalga oshiriladigan so‘rovlar asosida aniqlanadi.

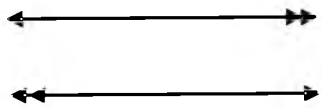
Misol uchun, bir tashkilotning xodimlari uchun ER-modelda quyidagi kalitlar bo‘lishi mumkin:

- **Birlamchi kalit:** Employee_ID — xodimni identifikatsiya qiluvchi kalit.
- **Tavsiflovchi atributlar:** Pasport, INN (ularning unikal xususiyatlari mavjud bo‘lsa-da, asosiy kalit sifatida tanlanmagan).
- **Tashqi (ikkilamchi) kalitlar:** Jinsi, ish boshlagan sana va ish staji kombinatsiyasi (xodimlarning malakasini oshirish rejasini tuzish uchun).

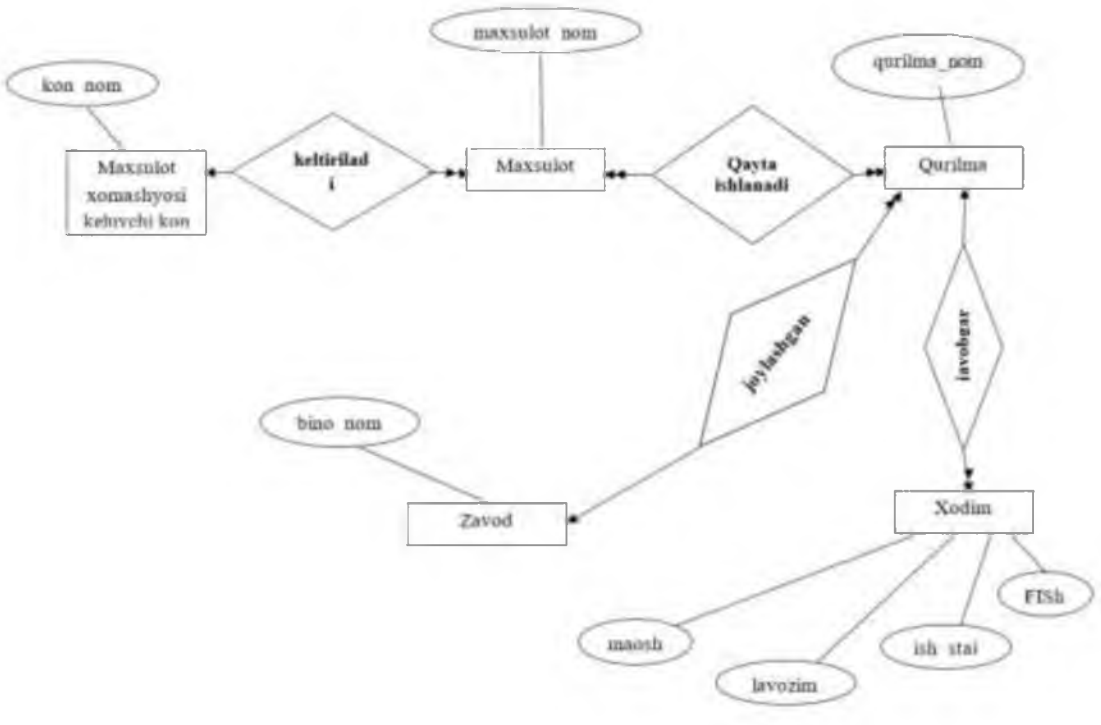
Bu kalitlar va atributlarning ma’lumotlar bazasidagi muhimligini to‘liq tasvirlash uchun ER-model hujjatlarida batafsil keltirilishi kerak. Bu hujjatlar ma’lumotlar bazasining fizik modeli ustida ishlaydigan ishlab chiqaruvchilarga indekslar yaratish va ma’lumotlarni qidirishni optimallashtirishga yordam beradi.

Ulman-Chen notasiyasining mohiyat-aloqa modeli elementlari.

	Yordamida ob’ektlar belgilanadi.
	Yordamida ob’ekt atributlari belgilanadi. Ular ob’ektlar bilan yo`nalishsiz chiziqlar yordamida birlashtiriladi
	Yordamida ob’ektlar orasidagi aloqalarni belgilaymiz.

	Bunda birga ko`p bog`langan A va V orasida V ga qaratib yo`nalgan yo`nalishli chiziq bilan ko`rsatiladi, ya'ni 1:N. Agar A va V ob'ektlar o`rtasida N:1 bog`lanish bo`lsa, strelka A ga qarab yo`naltiriladi.
	Ko`pga – ko`p bog`lanish N:M A va V ob'ektlar o`rtasida M: N bo`lsa, ularni ulovchi chiziq orqa li bog`lanadi.
	A va V orasida 1:1 bog`lanish bo`lsa, yo`nalishsiz chiziq bilan bog`laymiz.

Namuna sifatida. Foydali qazilma boyitish zavodi predmet sohasi uchun mohiyat – aloqa modelini keltiramiz.



3.5-rasm. Foydali qazilma boyitish zavodi Er-modeli

Nazorat savollari

1. Ma’lumotlar bazasi modellari nechta turga bo‘linadi?
2. Ma’lumotlar bazasining ierarxik modeli qanday ishlaydi?
3. Ma’lumotlar bazasining tarmoqli modeli qanday ishlaydi?
4. Ma’lumotlar bazasining relyatsion modeli qanday ishlaydi?
5. Mbning tarmoqli modeli va uning asosiy harakteristikalari.
6. PS mohiyat aloqa usulida tavsiflaganda qanday ishlar bajariladi?

4 BOB. RELYATSION MA'LUMOTLAR BAZASI MODELI VA MA'LUMOTLAR BAZASIDA MUNOSABATLAR

Tayanch so'zlar: *Relyatsion ma'lumotlar bazasi, birga-bir bog'lanish, birga-ko'p bog'lanish, ko'pga-ko'p bog'lanish, domen, relyatsion sxema*

4.1. Relyasion ma'lumotlar bazasini asosiy tushunchalari

Relyatsion ma'lumotlar modeli ilk bor E.F.Codd tomonidan 1970-yilda IBM tadqiqot markazida taklif qilingan va uning soddaligi hamda matematik asoslari tufayli katta e'tibor qozongan. **Edgar Frank Codd** 1923-yilda Angliyada tug'ilgan va 2003-yilda vafot etgan. Edgar Codd — mashhur informatika olimi bo'lib, u **relatyatsion ma'lumotlar bazasi** nazariyasining asoschisi hisoblanadi. 1970-yilda u o'zining relatsion ma'lumotlar modeli haqidagi maqolasini chop etdi, bu hozirgi zamonaviy ma'lumotlar bazasi tizimlarining asosi bo'lib xizmat qiladi. Shu maqolada u **normalizatsiya** va **ma'lumotlar bazasi dizayni** kabi asosiy konsepsiyalarni kiritgan. Ushbu modelning asosiy qurilishida matematik munosabatlar tushunchasi ishlatiladi, ya'ni qiymatlar jadvali ko'rinishida bo'lib, u o'zining nazariy asosini to'plamlar nazariyasi va birinchi darajali mantiqiy mulohaza (predicate logic) bilan bog'laydi.

Relyatsion modelning ilk tijorat dasturlari 1980-yillarda yaratilgan bo'lib, IBM tomonidan ishlab chiqilgan SQL/DS tizimi va Oracle ma'lumotlar bazasi tizimlari misol bo'la oladi. Hozirda relyatsion ma'lumotlar bazasi boshqaruv tizimlari (RMBBT) orasida DB2 (IBM), Oracle (Oracle korporatsiyasi), Sybase (SAP), SQLServer va Microsoft Access (Microsoft) kabi mashhur dasturlar mavjud. Shuningdek, MySQL va PostgreSQL kabi ochiq kodli tizimlar ham keng qo'llaniladi.

Relyatsion model ma'lumotlar bazasini munosabatlar yig'indisi sifatida taqdim etadi. Har bir munosabat qiymatlar jadvali yoki oddiy yozuvlar fayliga o'xshaydi. Jadvalning har bir qatori ma'lumotlar to'plamini ifodalaydi, bu esa ko'pincha real dunyo ob'ekti yoki munosabati bilan mos keladi. Har bir ustun jadvaldagi qiymatlar uchun qanday ma'noga ega bo'lishini aniqlaydi.

Domen — bu bir nechta atomik qiymatlar to‘plami. “Atomik” deganda, ushbu qiymatlarni relyatsion model uchun bo‘linmas va kichik qismlarga ajratib bo‘lmaydigan deb tushuniladi. Domenni belgilashning umumiy usuli uning ma’lumot turi (data type) bilan aniqlashdan iborat bo‘lib, bu qiymatlar ushbu to‘plamdan olinadi. Domen qiymatlarini tushunish uchun unga nom berish ham foydali bo‘ladi.

Quyida domenlarga ba’zi misollar keltirilgan:

- **Usa_phone_numbers**: Qo‘shma Shtatlardagi o‘n xonali telefon raqamlari to‘plami.
- **Local_phone_numbers**: AQShning muayyan hududiy kod doirasidagi yetti xonali telefon raqamlari to‘plami (biroq, bu raqamlar o‘n xonali raqamlar bilan almashtirilmoqda).
- **Social_security_numbers**: To‘qqiz xonali ijtimoiy sug‘urta raqamlari to‘plami (bu Qo‘shma Shtatlarda ish, soliqlar va imtiyozlar uchun har bir kishiga berilgan noyob identifikator).
- **Names**: Odamlarning ism-ma’lumotlarini ifodalaydigan qatorlar to‘plami.
- **Grade_point_averages**: 0 dan 4 gacha bo‘lgan sonlar ko‘rinishidagi o‘rtacha baholar qiymatlari.
- **Employee_ages**: Kompaniyada ishlovchilar yoshi; bu sonlar 15 dan 80 gacha bo‘lgan butun sonlar bo‘lishi mumkin.
- **Academic_department_names**: Universitetdagi akademik bo‘limlar nomlari to‘plami (masalan, Kompyuter fanlari, Iqtisodiyot, Fizika va boshqalar).
- **Academic_department_codes**: Akademik bo‘limlarning kodlari, masalan, ‘CS’ (Kompyuter fanlari), ‘ECON’ (Iqtisodiyot) va ‘PHYS’ (Fizika).

Yuqoridagi misollar domenlarning mantiqiy ta’riflari sifatida qaraladi. Domenning ma’lumot turi yoki formatini ham belgilash muhim. Masalan:

- **Usa_phone_numbers** domeni uchun ma’lumot turi (ddd)ddd-dddd formatida belgilanadi, bu yerda har bir **d** raqam bo‘lib, birinchi uchta raqam esa amal qiladigan telefon kodi hisoblanadi.
- **Employee_ages** domeni uchun ma’lumot turi 15 dan 80 gacha bo‘lgan butun sonlar ko‘rinishida bo‘ladi.
- **Academic_department_names** domenining ma’lumot turi barcha akademik bo‘limlarning haqiqiy nomlarini ifodalovchi qatorlar to‘plami bo‘lishi kerak.

Domenlarga qo‘shimcha ravishda, ularning qiymatlarini to‘g‘ri talqin qilish uchun boshqa ma’lumotlar ham keltirilishi mumkin. Masalan, **Person_weights** (og‘irlik) domeni uchun birliklar, masalan, kilogramm yoki funt ko‘rsatilishi lozim.

Domenning to‘liq tavsifi uning nomi, ma’lumot turi va formati bilan aniqlanadi.

Relyatsion sxema (schema) — bu ma’lumotlar bazasidagi munosabat (relation) nomidan va atributlar ro‘yxatidan tashkil topgan tuzilma. Relyatsion sxema odatda quyidagicha ifodalanadi: **R(A1, A2, ..., An)**, bu yerda **R** — munosabat nomi, **A1, A2, ..., An** esa atributlardir.

Har bir atribut bir domen (domain) bilan bog‘liq bo‘ladi. Domen — bu ma’lum bir ustundagi qiymatlar to‘plamini ifodalaydi. Masalan, agar biz **STUDENT** (talaba) munosabatiga ega bo‘lsak, unda quyidagi atributlarni ko‘rishimiz mumkin:

STUDENT(Name, Ssn, Home_phone, Address, Office_phone, Age, Gpa)

Bu yerda:

- **Name** (Ism) — talabaning ismi.
- **Ssn** (Ijtimoiy sug‘urta raqami) — shaxsni identifikatsiyalovchi raqam.
- **Home_phone** va **Office_phone** — telefon raqamlari.
- **Address** — yashash manzili.
- **Age** (Yosh) — talabaning yoshi.
- **Gpa** (Baholar o‘rtachasi) — o‘quv baholarining o‘rtacha ko‘rsatkichi.

Atributlar turlarini aniqlash uchun ba’zan turli ma’lumot turlaridan (masalan, string, integer) foydalaniladi. Masalan:

- **Name: string** — ism uchun matnli qiymatlar.
- **Age: integer** — yosh uchun butun sonlar.
- **Gpa: real** — baholar uchun haqiqiy sonlar.

Relyatsion sxemaning darajasi — bu atributlar sonidir. Masalan, yuqorida keltirilgan **STUDENT** sxemasining darajasi yettita, chunki u yettita atributga ega.

Relyatsion holat (relation state) — bu munosabatning ayni vaqtdagi qiymatlari to‘plamidir. Har bir qator yoki yozuv (tuple) munosabatda ma’lum bir ob’ekt yoki hodisani ifodalaydi. Bu qatorlar qiymatlar ketma-ketligi shaklida yoziladi va har bir qiymat atributning domeniga mos keladi yoki o‘ziga xos bo‘lgan **NULL** qiymatga ega bo‘lishi mumkin (agar ma’lumot noma’lum yoki mavjud bo‘lmasa).

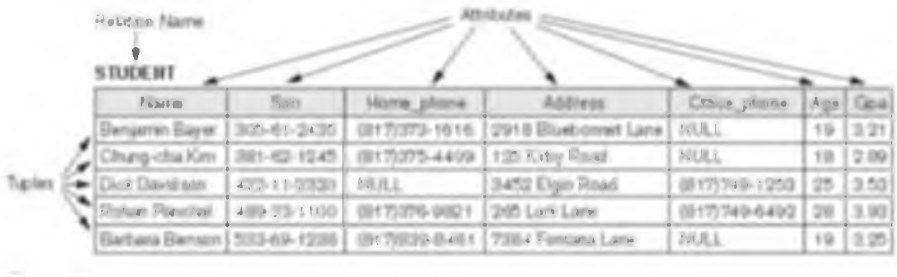
4.2. Ma’lumotlarni tasvirlashda jadvallardan foydalanish

Munosabat sxemasi va munosabat holati ma’lumotlar bazasini tuzish va boshqarishda juda muhim tushunchalardir. Har bir qator yoki yozuv(tuple) ma’lum bir ob’ektga tegishli ma’lumotlarni ifodalovchi qator sifatida jadvalda aks ettiriladi va atributlar jadvaldagi ustunlar sarlavhasi sifatida ishlaydi.

4.1-rasmda **STUDENT** munosbatining misoli keltirilgan bo‘lib, u ilgari ko‘rsatilgan **STUDENT** sxemasiga mos keladi. Ushbu munosabatda har bir yozuv (tuple yoki qator) ma’lum bir talabaga tegishli ma’lumotni ifodalaydi.

Munosabatni jadval shaklida ko‘rsatamiz, bunda har bir tuple qator sifatida ko‘rsatiladi, har bir atribut esa ustun sarlavhasi sifatida beriladi, bu ustundagi qiymatlarning ma’nosini tushuntirishga yordam beradi.

NULL qiymatlar, ba’zi talabalar uchun noma’lum yoki mavjud bo‘lmagan atribut qiymatlarini ifodalaydi. Masalan, agar talabaga oid ma’lumotlarda uning manzili noma’lum bo‘lsa, **Address** atributi uchun **NULL** qiymat ko‘rsatiladi.



4.1-rasm. Talaba munosabatining atributlari va qatorlari

Ma’lumotlar bazasidagi munosabatlar (relation) — bu ma’lumotlarni saqlash, tashkil etish va boshqarish uchun asosiy tuzilmalardir. Ular ma’lumotlar bazasining asosiy elementlarini tashkil etadi va qator atributlardan (ustunlardan) iborat bo‘lgan karteshlar (qatorlar) to‘plamini ifodalaydi. Har bir munosabat o‘z ichiga olgan atributlar va karteshlar orqali ma’lumotlarni ifodalaydi.

4.3. Ma’lumotlar bazasida munosobatlar

Relyatsion ma’lumotlar bazasi modeli asosida munosabat tushunchasi markaziy o‘rinni egallaydi. Bu model birinchi marta 1970-yilda **E. F. Codd** tomonidan taqdim etilgan bo‘lib, u ma’lumotlarni qat’iy strukturada – jadvallar shaklida saqlashni ta’minlaydi.

Munosabatlarning asosiy tushunchalari

- ✓ **Munosabat (Relation)**:
  - Munosabat — bu ma’lumotlar bazasidagi jadvalga o‘xshaydi. Har bir munozara ma’lum bir ob’ekt yoki hodisaga oid ma’lumotlarni saqlaydi.
  - **Misol**: talaba munosabati talabalar haqidagi ma’lumotlarni saqlashi mumkin:

4.1-jadval. Talaba munosabati

Name	Ssn	Age	Major
John	123-45-6789	20	Computer Science
Alice	987-65-4321	22	Mathematics

- ✓ **Birlamchi kalit (Primary Key)**:

- Asosiy kalit — bu munosbatdagi har bir karteshni noyob tarzda identifikatsiyalovchi atribut yoki atributlar to‘plami.
- **Misol**: Ssn (ijtimoiy xavfsizlik raqami) Talaba munosabatida asosiy kalit sifatida ishlatilishi mumkin, chunki har bir talabada noyob raqam mavjud.

✓ **Tashqi kalit (Foreign Key)**:

- Tashqi kalit — bu boshqa munosabat bilan bog‘lanishni ta’minlovchi atribut. U o‘zining asosiy kalitiga mos kelishi kerak.
- **Misol**:

4.2-jadval. O‘qituvchi jadvali

Id_uqituvchi	Ismi	Bo‘limi
1	Muxtor Karimov	Informatika
2	Dilshod Murodov	Matematika

4.3-jadval.Kurslar jadvali (courses)

kurs_id	kurs_nomi	uqituvchi_id
CSE101	Dasturlash asoslari	1
MAT201	Algebra	2

Tashqi kalitlar: **Kurslar** jadvalidagi uqituvchi_id ustuni **O‘qituvchi** jadvalidagi id_uqituvchi ustuniga tashqi kalit sifatida bog‘lanadi.

✓ **Kartesh (Tuple(qator))**:

- Kartesh — bu munozaradagi bir qator ma’lumotlar. Har bir kartesh atributlar bo‘yicha tashkil etilgan qiymatlar to‘plamidir.
- **Misol**: Talaba jadvalidagi bir kartesh: (John, 123-45-6789, 20, Computer Science)

✓ **Atribut (Attribute)**:

- Atribut — bu munozarada saqlanadigan ma’lumotning turini ifodalaydi.
- **Misol**: Name, Ssn, Age, Major atributlar STUDENT munozarasi uchun atributlardir.

4.4. Kodd ilmiy ishi. Munosabatni ikki o‘lchamli jadvallar yordamida tavsiflash

1983-yil oktyabrida E.F. Kodd ikki maqola chop ettirdi. Birinchi maqolada to‘liq relyatsion ma’lumotlar bazasi qoniqtirishi lozim bo‘lgan 12 qoida asoslandi. Ikkinchi maqolada dasturiy ta’minotlar shu qoidalarga mosligi taxlil etildi.

Qoida 1: Axborot qoidasi (The Information Rule): Relyatsion bazada hamma ma’lumot mantiqiy darajada bir usulda taqdim etilishi lozim: jadvallardagi qiymatlar bilan.

Qoida 2: Kafolatlangan murajat qoidasi (Guaranteed Access Rule):

Relyatsion ma’lumotlar bazasida har bir ma’lumot elementiga mantiqiy jadval nomi, birlamchi kalit qiymati va ustun nomi orqali murojaat qilish mumkin.

Qoida 3: NULL qiymatni tizimli qo‘llash (Systematic Treatment of Null Values): Har qanday ma’lum qiymatdan farqli NULL qiymatlar, hamma ma’lumotlar turlari uchun hamma amallar bajarilganda qo‘llanishi lozim. Masalan, sonli qiymatlar uchun 0 qiymat yoki belgili qiymatlar uchun bo‘sh satrdan farq qilishi lozim.

Qoida 4: Relyatsion model asosida dinamik aktiv lug‘at (Active On-Line Catalog Based on the Relational Model): Lug‘at relyatsion jadval shaklida saqlanishi lozim va ma’lumotlar bazasini boshqarish tizimlari bu jadvalga foydalanuvchi ma’lumotlari saqlanuvchi jadvalga murojjaat qilish uchun qo‘llanadigan standart vositalarni qo‘llash imkonini berishi lozim.

Qoida 5: Til ostki to‘plami to‘liqligi (Comprehensive Data Sublanguage Rule): Relyatsion ma’lumotlar bazasini boshqarish tizimlari juda bo‘lmasa bitta quyidagi xossalarga ega tilni qo‘llashi lozim: (a) chiziqli sintaksisga ega, (b) ham interaktiv, ham amaliy dasturlarda qo‘llanishi mumkin, (v) ma’lumotlar va tasvirlarni ta’riflash, ma’lumotlarni modifikatsiyalash (interaktiv yoki dasturiy), yaxlitlik cheklanishlarini ta’riflash, murojaatni boshqarish va tranzaksiya amallarini boshqarish (begin, commit va rollback).

Qoida 6: Tasvirlarni molifikatsiyalash qoidasi (View Updating Rule): Nazariy jixatdan modifikatsiyalanuvchi tasvirlarni tizim ham modifikatsiyalashi lozim. Har bir modifikatsiyalanuvchi tasvirga relyatsion jadvallarga qo‘llaniladigan amallar qo‘llash imkoniyati mavjud bo‘lishi lozim: tanlash, joylash, o‘zgartirish va ma’lumotlarni o‘chirish.

Qoida 7: Yuqori darajada joylash, yangilash va o‘chirish (High-Level Insert, Update, and Delete): Asosiy yoki hosila munosabat bilan bitta operand sifatida foydalanish faqat o‘qish uchun emas joylash, yangilash va o‘chirish amallariga ham qo‘llanishi lozim. Joylash, yangilash va o‘chirish amallari relyatsion jadvallar faqat bitta emas bir nechta satrlariga qo‘llanishi lozim.

Qoida 8: Ma’lumotlar fizik mustaqilligi (Physical Data Independence): Ilovalar va terminal amallar ma’lumotlarni saqlash va murojaat usullariga bog‘liq bo‘lmasligi lozim.

Qoida 9: Ma’lumotlar mantiqiy mustaqilligi (Logical Data Independence): Ilovalar va terminal amallar nazariy jixatdan ma’lumotlarni saqlovchi ixtiyoriy o‘zgarishlarga bog‘liq bo‘lmasligi lozim. Agar normallashtirish jarayonida jadval ajratilsa ilovaga ta’sir etmasligi lozim.

Qoida 10: Yaxlitlik mustaqilligi (Integrity Independence): Konkret ma’lumotlar bazasi yaxlitligi cheklanishlari relyatsion til ostki to‘plamida tasvirlash imkoniyati mavjud bo‘lishi va bu tasvir ilovalarda emas ma’lumotlar lug‘atida saqlanishi lozim. Ma’lumotlar bilan ishlaydigan til kiritilayotgan ma’lumotlarni tekshirishi va ma’lumotlarning yaxlitligini saqlashi lozim.

Qoida 11: Tarqatish mustaqilligi (Distribution Independence): Relyatsion ma’lumotlar bazasida taqsimlash mustaqildir. Ma’lumotlar bazasi tarmoqlangan ya’ni bir nechta kompyuterlar xotirasida saqlanishi mumkin, lekin bu ilova ishiga ta’sir qilmasligi lozim.

Qoida 12: Til darajalari muvofiqligi (The Nonsubversion Rule):Agar ma'lumotlar bilan ishlash uchun quyi darajadagi til ishlatilsa, yuqori darajadagi til qo'llovchi xavfsizlik va yaxlitlik qoidalarni qo'llashi lozim.Quyi darajadagi til yordamida ma'lumotlar yaxlitligini buzish imkoniyati mavjud bo'lmasligi lozim.

Relyatsion ma'lumotlar bazasi modelida **munosabat** ikki o'lchamli jadval sifatida ifodalanadi. Ushbu yondashuv real dunyo ob'ektlari va ularning o'zaro bog'lanishlarini sodda va tushunarli shaklda tasvirlash imkonini beradi. Quyida ikki o'lchamli jadvallar yordamida munosabatni tavsiflashning asosiy jihatlari keltiriladi.

4.5. Munosabatlar to'plami ma'lumotlarni saqlash uchun ishlatilishi va ular orasidagi bog'lanishlarni modellashtirish

Munosabatlar to'plami – bu relyatsion ma'lumotlar bazasida bir-biriga bog'langan bir nechta jadvallar. Ushbu jadvallar ma'lumotlarni qayta ishlash va ularga kirish imkoniyatini ta'minlaydi.

Ma'lumotlar saqlash: Ma'lumotlar birlamchi kalit (primary key) orqali bir-biriga bog'lanadi.

Misol: Talabalar va kurslar haqidagi ma'lumotlar ikki munosabatda saqlanishi mumkin:

1. Talabalar: Talaba_ID, Ism, Familiya.
2. Kurslar: Kurs_ID, Talaba_ID, Kurs_Nomi.

- **Birga-bir bog'lanish(One-to-One Relationship)**: Ikki mohiyat orasida birga bir bog'lanish mavjud deyiladi agar birinchi mohiyatning har bir nusxasiga ikkinchi mohiyatning bitta nusxasi mos kelishi mumkin bo'lsa, va aksincha.

  Misol.

4.4-jadval. Talabalar jadvali

id	Name	Email
1	Ali Karimov	ali.k@example.com
2	Gulnora Ahamova	gulnora.a@example.com

4.5-jadval. Talaba profil jadvali (profiles)

ID_profil	talaba_id	Address	Phone
1	1	Tashkent, Uzbekistan	+998901234567
2	2	Samarkand, Uzbekistan	+998911234567

Bog‘lanish

Talabalar jadvalidagi ID ustuni **Talaba Profil** jadvalidagi Student_ID ustuniga tashqi kalit sifatida bog‘langan. Har bir talabaning shaxsiy profili bo‘lishi zarur va bu profil faqat bitta talaba bilan bog‘liq.

- **Bir ko‘pga bog‘lishlar(One-to-Many Relationship)**: Ikki mohiyat orasida birga ko‘p bog‘lanish mavjud deyiladi. Agar birinchi mohiyatning har bir nusxasiga ikkinchi mohiyatning bir nechta nusxasi mos kelishi mumkin bo‘lsa va ikkinchi mohiyatning har bir nusxasiga birinchi mohiyatning bittadan ko‘p bo‘lmagan nusxasi mos kelishi mumkin bo‘lsa.

  Misol:

4.6-jadval. Doktorlar jadvali:

doktor_id	Ismi
1	Dr. Ahmed
2	Dr. Malika
3	Dr. Asad

4.7-jadval. Mijozlar jadvali (Patients) (tashqi kalit qo‘shildi):

Id_bemor	Name	doktor_id
1	Zafar Karimov	1
2	Sara Qodirova	1
3	Jamshid Nurmurodov	2

- **Ko‘p ko‘pga bog‘langan munosabatlar (Many-to-Many Relationship)**: Ikki mohiyat orasida ko‘pga ko‘p bog‘lanish mavjud deyiladi, agar birinchi mohiyatning har bir nusxasiga ikkinchi mohiyatning bir nechta nusxasi mos kelishi mumkin bo‘lsa va aksincha.

  Misol.

4.8-jadval. Talabalar Jadvali (Students)

student_id	Name
1	Aisha Karimova
2	Jamshid Abdullayev
3	Sanjar Yuldashev

4.9-jadval. Fanlar Jadvali (Subjects)

subject_id	Subject_Name
1	Mathematics
2	Physics
3	Chemistry
4	Biology
5	Geography

4.10-jadval. Talaba-Fanlar Munosabatlari Jadvali (Student_Subjects)

student_id	subject_id
1	1
1	2
1	3
2	2
2	3
2	4
3	1
3	5

Talabalar Jadvali (Students): Talabalar haqidagi ma’lumotlar saqlanadi.

Fanlar Jadvali (Subjects): Fanlar haqidagi ma’lumotlar saqlanadi.

Talaba-Fanlar munosabatlari jadvali (Student_Subjects): Bu jadval ko‘pga ko‘p munosabatni ifodalaydi. Har bir talaba bir nechta fanlarni o‘qishi mumkin va har bir fan bir nechta talabalar tomonidan o‘qilishi mumkin.

Ushbu munosabatlar yordamida ma’lumotlar bazasidagi ma’lumotlarni osonlik bilan saqlash, boshqarish va ularga murojaat qilish mumkin. Har bir munosabat o‘zaro bog‘lanishlar bilan birga kelishi, ma’lumotlar bazasining funksional va samarali ishlashini ta’minlaydi.

Nazorat savollari:

1. Munosobat xossalariga nimalar kiradi?
2. Munosobatlar sxemasiga misollar keltiring.
3. Munosabat turlari nechta?
4. Birga-bir munosabatga misol keltiring?
5. Birga – ko‘p munosabatga misol keltiring?

5 BOB. RELYATSION ALGEBRA VA RELYATSION HISOBOT ELEMENTLARI

Tayanch so‘zlar: *relyatsion algebra, relyatsion hisoblash, birlashtirish (union), kesishuv (intersection), ayirma (set difference), dekart ko‘paytma (cross product), tanlash (select), proeksiya (project), qo‘shish (union), bo‘lish (difference)*

5.1. Munosabatlar ustida amallar

Bu mavzuda biz relyatsion model uchun ikkita rasmiy tilni muhokama qilamiz: relyatsion algebrasi va relyatsion hisoblash. Tarixan, relyatsion algebrasi va relyatsion hisoblash SQL tilidan oldin ishlab chiqilgan. SQL asosan relyatsion hisoblash tushunchalariga asoslangan va unga relyatsion algebrasi tushunchalari ham qo‘shilgan. Ko‘pgina relyatsion ma’lumotlar bazasi boshqaruv tizimlari (MBBT) SQL tilidan foydalanganadi.

Relyatsion algebra elementlari foydalanuvchiga asosiy qidiruv so‘rovlarini belgilash imkonini beradi. Qidiruv natijasi yangi munosabat (relation) hisoblanadi. Shunday qilib, algebra operatsiyalari yangi munosabatlarni hosil qiladi, ularni yana shu algebra operatsiyalari bilan qayta ishlash mumkin. Munosabatlar algebra operatsiyalarining ketma-ketligi relyatsion algebra ifodasini tashkil etadi va uning natijasi ham munosabat bo‘lib, bu ma’lumotlar bazasi so‘rovining (yoki qidiruv so‘rovining) natijasini ifodalaydi.

Relyatsion algebra juda muhimdir, chunki u so‘rovlarni qayta ishlash va optimallashtirish modullari uchun asos bo‘lib xizmat qiladi, bu modullar munosabatli ma’lumotlar bazasi boshqaruv tizimlarining ajralmas qismlaridir. Shuningdek, uning ba’zi tushunchalari SQL standart so‘rov tiliga kiritilgan. Garchi bugungi kunda aksariyat relyatsion ma’lumotlar bazasi boshqaruv tizimlari foydalanuvchilar uchun relyatsion algebra so‘rovlari interfeyslarini taqdim etmasa ham, ichki modullarda ishlatiladigan asosiy operatsiyalar va funksiyalar ko‘pincha relyatsion algebra operatsiyalariga asoslanadi.

5.2. Relyasion ma’lumotlar bazasini asosiy tushunchalari

Relyatsion ma’lumotlar bazasi (RDBMS) — bu ma’lumotlarni saqlash, boshqarish va qayta ishlashda relyatsion modelga asoslangan tizimdir. Relyatsion model, 1970-yillarda E. F. Codd tomonidan taklif qilingan va u ma’lumotlarni ikki o‘lchamli jadvallar (yoki munosabatlar) shaklida taqdim etadi. Quyida relyatsion ma’lumotlar bazasining asosiy tushunchalari keltirilgan:

1. Munosabat (Relation)

Munosabat (jadval) ma’lumotlar bazasining asosiy tarkibiy qismidir. Bu ikki o‘lchamli jadval bo‘lib, satrlar (qatorlar) va ustunlar (atributlar)dan tashkil topadi.

2. Atribut (Attribute)

Atribut jadvalning ustunini tashkil qiladi. Har bir atribut ma’lumot turini va unga tegishli qiymatlarni belgilaydi.

Misol: Talaba_ID, Ism, Yoshi, Kurs — bu talabalar jadvalidagi atributlar.

3. Satr (Row)

Jadvaldagi har bir satr munosabatdagi ma’lumotning bitta yagona yozuvini tashkil qiladi. Har bir satr atributlar qiymatlari to‘plamidan iborat bo‘ladi.

Misol: 1-satr: (1, Ali, 20, Matematika) — bu bir talaba haqida ma’lumotdir.

4. Kalit (Key)

Kalit — bu atribut yoki atributlar to‘plamidir, u orqali jadvalda har bir satrni noyob tarzda identifikatsiya qilish mumkin.

Birlamchi kalit (Primary Key): Jadvaldagi har bir satrni noyob identifikatsiya qilish uchun mo‘ljallangan kalit. Har bir jadvalda faqat bitta birinchi kalit bo‘lishi mumkin.

Tashqi kaliti (Foreign Key): Boshqa jadvaldagi birinchi kalitga ishora qiluvchi atribut. Bu ikki jadvalni bog‘laydi.

Misol: Talabalar jadvalidagi Talaba_ID atributi birinchi kalit bo‘lib xizmat qiladi.

5. Relyatsion algebra (Relational Algebra)

Relyatsion algebra— bu ma’lumotlar bazasida ma’lumotlarni boshqarish uchun qo‘llaniladigan matematik operatsiyalar to‘plamidir. U, odatda, ma’lumotlarni tanlash, proeksiya qilish, qo‘shish, kesishuv va boshqa amallarni o‘z ichiga oladi.

5.3. Relyasion algebra va uning amallari

Relyatsion algebra relyatsion modelning ajralmas qismi hisoblanadi. Uning operatsiyalari ikkita guruhga bo‘linadi. Birinchi guruhga matematik to‘plam nazariyasidan kelib chiqqan to‘plam amallari kiradi (An’anaviy amallar); bu operatsiyalar relyatsion ma’lumotlar bazasida qo‘llaniladi, chunki har bir munosabat (relation) to‘plam sifatida belgilangan.

To‘plam(An’anaviy amallar)amallariga:

- Birlashtirish (union)
- Kesishuv (intersection)
- Ayirma (set difference)
- Dekart ko‘paytma (cross product) kiradi.

Ikkinchi guruhga relyatsion ma’lumotlar bazalari uchun maxsus ishlab chiqilgan amallar-maxsus amallar kiradi:

- Tanlash (select)
- Proeksiya (project)
- Qo‘shish (Union)
- Bo‘lish (Difference)

Relyatsion hisoblash esa relyatsion ma’lumotlar bazasi uchun yuqori darajadagi deklarativ tilni taqdim etadi. Relyatsion hisoblash ifodasida operatsiyalar tartibini belgilash shart emas — faqat natijada qanday ma’lumot bo‘lishi kerakligi ko‘rsatiladi. Bu reltaysion algebrasi va relyatsion hisoblash o‘rtasidagi asosiy farqdir. Relyatsion hisoblash matematik mantiq asosida qurilgan va uning bir turi SQL tilining asoslaridan biridir.

Keling, ***birlashtirish (union), keseshuv (intersection), ayirma (set difference) va dekart ko'paytma (cartesian product)*** operatsiyalarini to'plamlar nazariyasi va jadval ko'rinishida umumlashtiramiz.

To'plamlar nazariyasi ko'rinishida

Faraz qilaylik bizda ikkita to'plam mavjud:

To'plam A={1,2,3}

To'plam B={2,3,4}

Jadval ko'rinishida esa quyidagi A va B jadvallarimiz bo'lsin:

5.1-jadval. A jadvali:

ID	Ism
1	Ali
2	Vali
3	Salim

5.2-jadval.B jadvali:

ID	Ism
2	Vali
3	Karim
4	Husan

1. BIRLASHMA (UNION)

To'plamlar nazariyasida BIRLASHTIRISH ikki to'plamdagi barcha elementlarni birlashtiradi, lekin takrorlanadigan elementlarni bir marta kiritadi.

$A \cup B = \{1,2,3,4\}$

Jadval ko'rinishida:

Natija: 5.3-jadval. A UNION B

ID	Ism
1	Ali
2	Vali
3	Salim
3	Karim
4	Husan

2. KESISHMA (INTERSECTION).

To‘plamlar nazariyasida KESESHISH ikki to‘plamdagi umumiy elementlarni oladi, ya’ni ikkala to‘plamda ham mavjud bo‘lgan elementlarni.

To‘plamlar nazariyasi ko‘rinishida:

A∩B={2,3}

5.4-jadval. Kesishma

ID	Ism
2	Vali
3	Karim

3. AYIRMA (SET DIFFERENCE)

AYIRMA operatsiyasi birinchi jadvaldagi, lekin ikkinchi jadvalda mavjud bo‘lmagan yozuvlarni oladi.

To‘plamlar nazariyasi ko‘rinishida: A−B={1}

Bu yerda faqat A to‘plamida mavjud bo‘lib, B to‘plamida yo‘q bo‘lgan elementlar ko‘rsatilgan.

5.5-jadval. Ayirma

ID	Ism
1	Ali

4. DEKART KO‘PAYTMA (CARTESIAN PRODUCT)

To‘plamlar nazariyasida DEKART KO‘PAYTMA ikki to‘plamdagi har bir element juftligini hosil qiladi.

5.6-jadval. Dekart ko‘paytma(To‘plamlar nazariyasi ko‘rinishida)

A.ID	A.Ism	B.ID	B.Ism
1	Ali	2	Vali
1	Ali	3	Karim
1	Ali	4	Husan
2	Vali	2	Vali
2	Vali	3	Karim

2	Vali	4	Husan
3	Salim	2	Vali
3	Salim	3	Karim
3	Salim	4	Husan

$A \times B = \{(1,2),(1,3),(1,4),(2,2),(2,3),(2,4),(3,2),(3,3),(3,4)\}$

Bu yerda A to‘plamidagi har bir element B to‘plamidagi har bir element bilan juftlik hosil qiladi.

Yuqoridagi to‘plamlar nazariyasi va jadval ko‘rinishidagi misollar BIRLASHTIRISH, KESESHUV, AYIRMA va DEKART KO‘PAYTMA operatsiyalarini birga ko‘rsatadi. Bu orqali siz to‘plamlar nazariyasi asosida ma’lumotlarni qanday qo‘llash va ularni jadval ko‘rinishida tasvirlash mumkinligini tushunishingiz mumkin.

Ikkinchi guruhga relyatsion ma’lumotlar bazasi uchun maxsus ishlab chiqilgan operatsiyalarni yoritamiz:

Quyida relyatsion ma’lumotlar bazalari uchun maxsus ishlab chiqilgan relyatsion algebra amallari uchun misollar keltirilgan. Bu misollarni tushunish uchun ikki jadvalni ko‘rib chiqamiz:

5.7-jadval. Employee jadvali

EmployeeID	Name	Department
1	Ali	IT
2	Bahrom	HR
3	Diyor	IT
4	Elbek	Finance

5.8-jadval. Department jadvali

Department	Location
IT	Toshkent
HR	Samarqand
Finance	Buxoro

1. TANLASH (SELECT(σ))

Ta'rif: Tanlash operatsiyasi jadvaldan ma'lum bir shartga mos keladigan satrlarni ajratib oladi.

Misol: IT bo'limida ishlayotgan xodimlarni tanlaymiz.

So'rov: SELECT * FROM Employee WHERE Department = "IT";

Relyatsion algebra ko'rinishi: σ Department = 'IT' (Employee)

5.9-jadval. Tanlash operatori natijasi

ID	Name	Department
1	Ali	IT
3	Diyor	IT

2. PROEKSIYA (PROJECT(π))

Ta'rif: Proeksiya operatsiyasi jadvaldan faqat kerakli ustunlarni tanlaydi.

Misol: Xodimlarning faqat ismi va bo'limini ko'rsatamiz.

So'rov: SELECT Name, Department FROM Employee;

Relyatsion algebra ko'rinishi: *π Name, Department (Employee)*

5.10-jadval. Proeksiya natija

Name	Department
Ali	IT
Bahrom	HR
Diyor	IT
Elbek	Finance

Bu misollar orqali TANLASH, PROEKSIYA operatsiyalarining qanday ishlashini tushunishingiz mumkin.

5.4. Relyasion xisoblash elementlari va ulardan foydalanish

Relatsion hisoblash (relational calculus) — bu relatsion ma'lumotlar bazalarida ma'lumotlarni so'rash uchun ishlatiladigan matematik va formal til. Ushbu til yordamida biz ma'lumotlarni qanday olishimiz kerakligini belgilamay, faqat qaysi

ma’lumotlarni olishimiz kerakligini aniqlaymiz. Relatsion hisoblashning asosiy elementlari quyidagilar:

Bizda EMPLOYEE jadvali yaratilgan bo‘lsin:

5.11-jadval. EMPLOYEE jadvali

ID	Fname	Lname	Department	Salary
1	Ali	Ibragimov	IT	60000
2	Bahrom	Qodirov	HR	45000
3	Diyor	Toshpulatov	IT	70000
4	Elbek	Sardorov	Finance	80000
5	John	Smith	HR	55000

1. Qator o‘zgaruvchilari (Tuple Variables)

Qator o‘zgaruvchilari — bu ma’lumotlar jadvalidagi qatorlarni ifodalovchi o‘zgaruvchilar. Har bir qator o‘zgaruvchisi ma’lum bir jadvaldagi (relation) har qanday qatorni qabul qilishi mumkin.

Misol uchun **t** EMPLOYEE jadvalidagi har qanday qatorni ifodalaydi.

{t | EMPLOYEE(t)}

SQl so‘rovi: SELECT * FROM Employee AS t;

2. Shartlar (Conditions)

Shartlar qator o‘zgaruvchisi uchun mantiqiy ifodalar bo‘lib, ular qator o‘zgaruvchisi qanday qatorlarni qamrab olishini belgilaydi. Shartlar TRUE (rost) yoki FALSE (yolg‘on) qiymatini qabul qiladi.

Misol: {t | EMPLOYEE(t) AND t.Salary > 50000} — bu shart, agar **t** o‘zgaruvchisi maoshi 50000 dan yuqori bo‘lgan xodimni ifodalasa, TRUE ga teng bo‘ladi.

SQL so‘rovi: SELECT * FROM Employee WHERE Salary > 50000;

3. Shartlar to‘plami (Condition Set)

Shartlar to‘plami — bu bir yoki bir nechta shartlarni o‘z ichiga olgan ifoda. Bu shartlar birgalikda bajarilishi kerak bo‘lgan shartlarni ifodalaydi.

Misol: {t | EMPLOYEE(t) AND t.Salary > 50000 AND t.Department = ‘IT’}

- ✓ **AND t.Salary > 50000**: Bu shart, t qator o‘zgaruvchisi (ya’ni, xodim) maoshi 50000 dan yuqori bo‘lishi kerakligini belgilaydi.
- ✓ **AND t.Department = ‘IT’**: Bu shart, xodimning bo‘limi “IT” ekanligini ko‘rsatadi.

SQL so‘rovi:

SELECT * FROM Employee WHERE Salary > 50000 AND Department = ‘IT’;

4. So‘rov Tuzilishi (Query Structure)

Relatsion hisoblashdagi so‘rovlar quyidagi shaklda yoziladi:

{t | COND(t)}

Bu yerda **t** — qator o‘zgaruvchisi, **COND(t)** esa shartlar ifodasi. Ushbu so‘rov **COND(t)** shartini qanoatlantiruvchi barcha qatorlarni qaytaradi.

Misol: **{t | EMPLOYEE(t) AND t.Salary > 50000}** — bu so‘rov **EMPLOYEE** jadvalidagi maoshi 50000 dan yuqori bo‘lgan barcha xodimlarni chiqaradi.

Sql so‘rov: SELECT * FROM Employee AS t WHERE t.Salary > 50000;

5. Natijalar (Results)

Relatsion hisoblashda so‘rov natijasi, shartlarni qanoatlantiruvchi qatorlar to‘plamidan iborat bo‘ladi. Ushbu natijalar so‘rovda ko‘rsatilgan atributlar bo‘yicha qaytariladi.

Misol: Agar biz faqat xodimlarning ismini va familiyasini olishni xohlasak:

{t.Fname, t.Lname | EMPLOYEE(t) AND t.Salary > 50000}

Ushbu so‘rov natijasi maoshi 50,000 dan yuqori bo‘lgan xodimlarning ismlari va familiyalari to‘plami bo‘ladi.

Sql so‘rov: SELECT Fname, Lname FROM Employee AS t WHERE t.Salary > 50000;

6. Mantiqiy Operatorlar

Relatsion hisoblashda mantiqiy operatorlar (AND, OR, NOT) yordamida shartlar birlashtirilishi mumkin.

Misol: **EMPLOYEE(t) AND t.Salary > 50000 OR t.Department = ‘HR’** — bu ifoda, xodimlarning maoshi 50000 dan yuqori bo‘lgan yoki bo‘limi "HR" bo‘lgan xodimlarni qidiradi.

SELECT * FROM Employee AS t WHERE t.Salary > 50000 OR t.Department = ‘HR’;

7. O‘zgaruvchilar doirasi (Range of Variables) - bu qator o‘zgaruvchilarining qaysi jadvalda (relation) ishlashini belgilaydi. Bu doira, o‘zgaruvchilar qaysi qatorlarni qamrab olishi va ular ustida qanday shartlar bajarilishini aniqlashga yordam beradi.

Misol uchun 20,000 dan 50,000 oraliqdagi maoshga ega xodimlarni tanlash uchun relatsion hisoblash ifodasini quyidagicha yozamiz:

{t | EMPLOYEE(t) AND t.Salary >= 20000 AND t.Salary <= 50000}

Sql so‘rovi: SELECT * FROM Employee WHERE Salary BETWEEN 20000 AND 50000;

Demak, Relyatsion hisoblash elementlari, relyatsion ma’lumotlar bazalarida matematik amallarni amalga oshirish uchun ishlatiladigan elementlardir. Ular, jadvaldagi ma’lumotlar ustunlari (sutunlar) va qatorlari (satrlar) orqali amal qiladi. Relyatsion hisoblash elementlari orqali, ma’lumotlarning agregat (umumiy) qiymatlarini hisoblash, ma’lumotlarni guruhlash, statistik ma’lumotlarni olish va boshqalar kabi hisoblash amallari amalga oshiriladi. Misol uchun, AVG(), SUM(), COUNT() kabi funksiyalar ma’lumotlarni hisoblash uchun foydalaniladi.

Relyatsion hisoblash, shartlar va qator o‘zgaruvchilari orqali ma’lumotlarni qidirishga qaratilgan. Bu til yordamida ma’lumotlar bazasidan ma’lum shartlar asosida ma’lumotlarni olish mumkin. Relatsion hisoblash o‘zining matematik asoslari bilan yuqori darajadagi so‘rov tillari, masalan, SQL, uchun asos hisoblanadi.

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 BOB. MA'LUMOTLAR BAZASINI REJALASHTIRISH, LOYIHALASH VA ADMINISTRATSIYALASH

Tayanch so'zlar: *rejalashtirish, loyihalash, konseptual loyihalash, logik loyihalash, fizik loyihalash, administrator, administratsiyash*

6.1. Ma'lumotlar bazasini hayot siklini tashkil etish

Dasturiy ta'minot ishlab chiqishning strukturaviy yondashuvi axborot tizimlarining hayot sikli deb ataladi. Hayot siklning bosqichlari:

Bosqich	Mazmuni
Ma'lumotlar bazasini ishlab chiqishni rejalashtirish	**Tizim talablarini aniqlash**. Ma'lumotlar bazasi dasturining amal qilish doirasini, chegaralarini, foydalanuvchilar tarkibini va foydalanish sohalarini aniqlash.
	Foydalanuvchi talablarini yig'ish va tahlil qilish. Yaratilayotgan ma'lumotlar bazasi dasturi yordamida qo'llab-quvvatlanadigan tashkilotning ish faoliyati haqida ma'lumotlarni yig'ish va tahlil qilish, shuningdek, foydalanuvchilar talablarini aniqlash uchun ushbu ma'lumotlardan foydalanish.
Ma'lumotlar bazasini loyihalash	Ma'lumotlar bazasini loyihalash jarayoni - tashkilotning funksiyalarini avtomatlashtirish va uning maqsadlariga erishishga yordam beradigan jarayondir. Ushbu bosqich quyidagi uch bosqichga bo'linadi: **Konseptual ma'lumotlar bazasini loyihalash**: Ma'lumotlar bazasining umumiy ko'rinishi va ob'ektlar o'rtasidagi munosabatlar aniqlanadi. ER-diagrammalar yordamida ma'lumotlar strukturasi belgilanadi. **Logik ma'lumotlar bazasini loyihalash**: Ma'lumotlar bazasining mantiqiy strukturasini aniqlanadi. Normalizatsiya jarayonlari o'tkaziladi va jadval strukturalari yaratiladi. **Fizik ma'lumotlar bazasini loyihalash**: Ma'lumotlar qanday saqlanadi, indekslar va saqlash qurilmalari aniqlanadi.
	Ma'lumotlar bazasini boshqarish tizimi yaratiladi. Yaratilayotgan ma'lumotlar bazasi dasturini qo'llab-quvvatlash uchun mos turdagi MBBT tanlanadi.

Bosqich	Mazmuni
	Dasturlarni ishlab chiqish. Ma'lumotlar bazasi bilan ishlash uchun foydalanuvchi interfeysini va qo'shimcha dasturlarni loyihalash amalga oshiriladi.
	Prototip yaratish. Ma'lumotlar bazasi va ma'lumotlar bazasi dasturining ishchi modelini yaratiladi.
	Amalga oshirish. Ma'lumotlar bazasi asosida ishlab chiqilgan dastur amalga oshiriladi
	Sinovdan o'tkazish. Xatolarni topish maqsadida dastur sinovdan o'tkaziladi.
	Ishlatish va qo'llab-quvvatlash. Tizimning ishlash jarayonini kuzatiladi va uning normal ishlashini ta'minlanadi.

6.2. Ma'lumotlar bazasini rejalashtirish

1. Tayyorlov ishlari: Bu bosqichda loyiha jarayonini rejalashtirish va tashkil etish uchun barcha kerakli tayyorgarlik ishlari amalga oshiriladi.

Ma'lumotlar bazasining tuzilishi: Loyihaning muvaffaqiyati uchun zarur bo'lgan resurslar (moliyaviy., vaqt, inson kuchi) aniqlanadi.

Tizimga doir tushunchalar: Tizimning maqsadi, foydalanuvchilar va uning natijalari aniqlanadi, bu jarayonda loyiha jamoasi o'zaro fikr almashadi.

2. Tizim talablarini aniqlash

Talablar aniqlanadi**:** Bu bosqichda ma'lumotlar bazasi dasturining imkoniyatlari va cheklovlari aniqlanadi.

Tizimning ishlash doirasi**:** Qanday ma'lumotlar saqlanadi, foydalanuvchilar kimlar va tizimdan qanday foydalanish kerakligi belgilanadi.

Ishlab chiqish hujjati: Talablar to'g'risida batafsil hujjat tayyorlanadi, bu keyingi bosqichlar uchun muhim asos bo'ladi.

3. Foydalanuvchi talablarini yig'ish va tahlil qilish

6.3. Ma’lumotlar bazasini loyihalash

4.Ma’lumotlar bazasini loyihalash jarayoni: Bu bosqichda tashkilotning funksiyalarini avtomatlashtirish va uning maqsadlariga erishishga yordam beradigan ma’lumotlar bazasini loyihalash amalga oshiriladi. Ushbu bosqich uchta asosiy qismga bo‘linadi:

Konseptual ma’lumotlar bazasini loyihalash: Tizimda ishlatiladigan asosiy ma’lumotlar ob’ektlari va ularning o‘zaro munosabatlari aniqlanadi. ER-diagrammalar yordamida ma’lumotlar strukturasi belgilanadi.

Logik ma’lumotlar bazasini loyihalash: Ma’lumotlar bazasining logik strukturasini aniqlash va normalizatsiya jarayonlari o‘tkaziladi. Har bir jadval va ustunlar aniqlanadi.

Fizik ma’lumotlar bazasini loyihalash: Ma’lumotlar qanday saqlanishi, indekslar (partitioning) va saqlash qurilmalari ko‘rib chiqiladi.

5. Ma’lumotlar bazasini boshqarish tizimini tanlash

MBBT tanlovi: Ma’lumotlar bazasi dasturini qo‘llab-quvvatlash uchun eng mos MBBT (Ma’lumotlar basazini boshqaruv tizimi) tanlanadi.

Kriteriyalar: Tanlov jarayonida turli mezonlar ko‘rib chiqiladi: funksionallik, ishlash tezligi, qo‘llab-quvvatlash,narx va h.k.

6. Dasturlarni ishlab chiqish

Foydalanuvchi interfeysi: Foydalanuvchilar uchun intuitiv va qulay interfeys loyihalanadi.

Qo‘shimcha dasturlar: Ma’lumotlar bazasi bilan ishlash uchun kerakli dasturlar va funksiyalar ishlab chiqiladi.

Kod yozish: Dasturlash tillaridan foydalanib, dastur kodlari yoziladi.

7. Prototip yaratish

Prototip tayyorlash: Yaratilayotgan tizimning ishchi modeli tayyorlanadi. Bu model orqali tizimning funksiyalari va interfeysi sinovdan o‘tkaziladi.

Foydalanuvchi fikri: Prototip yordamida foydalanuvchilarning fikr-mulohazalari olinadi va tizimga yangiliklar kiritiladi.

8. Amalga oshirish

Tizimni ishga tushirish: Tayyor tizim foydalanuvchilarga taqdim etiladi va ishga tushiriladi.

Ma'lumotlarni ko'chirish: Mavjud ma'lumotlar yangi tizimga o'tkaziladi va barcha kerakli konfiguratsiyalar amalga oshiriladi.

Dasturlarni yuklash va modifikatsiya qilish: Mavjud dasturlar yangi ma'lumotlar bazasi bilan birgalikda ishlashini ta'minlash uchun yangilanadi.

10. Sinovdan o'tkazish

Dasturlarni bajarish: Xatolarni topish maqsadida dasturlar ishga tushiriladi. Tizimning barcha funksiyalari sinovdan o'tkaziladi.

Xato tuzatish: Sinov jarayonida aniqlangan xatolar tuzatiladi va tizim ishlashga tayyorlanadi.

11. Ishlatish va qo'llab-quvvatlash

Tizim monitoringi: Tizimning normal ishlashini ta'minlash, foydalanuvchilarga yordam ko'rsatish va muammolarni hal qilish maqsadida tizim kuzatib boriladi.

Yangilash: Dasturiy ta'minotni yangilash, yangi funksiyalarni qo'shish va o'zgarishlarga muvofiq tizimni moslashtirish amalga oshiriladi.

12. Takomillashtirish

Foydalanuvchi fikrlari: Foydalanuvchilarning fikr-mulohazalari asosida tizimni takomillashtirish amalga oshiriladi.

O'zgarishlarni kiritish: Yangi talablar va o'zgarishlarga muvofiq dasturiy ta'minot yangilanadi.

Har bir bosqichda to'g'ri yondashuv va rejalashtirish juda muhim, chunki bu tizimning muvaffaqiyatli ishlashini ta'minlaydi!

6.4. Ma’lumotlar bazasini administratsiyalash

Ma’lumotlar bazasini administratsiyalash (yoki ma’lumotlar bazasini boshqarish) — bu ma’lumotlar bazasining doimiy ravishda ishlashini, xavfsizligini va samaradorligini ta’minlash uchun amalga oshiriladigan texnik va boshqaruv amaliyotlari 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 (MB) tushunchasi ikki xil talqin etiladi.

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. Ayrim hollarda, ma’lumotlar banki deyilganda ma’lumotlar bazasi uni boshqarish tizimi(MBBT) tushuniladi. Ma’lumot bazasi bilan ishlaydigan dasturni ilova deb ataladi.

6.5. Ma’lumotga samarali murojaatni tashkil qilishda bazalar o‘zaro aloqasi fayl tuzilmalaridan foydalanish

Ma’lumotlar bazasini samarali boshqarish va ma’lumotga tezkor murojaatni tashkil qilishda fayl tuzilmalarining roli juda katta. Fayl tuzilmalari ma’lumotlarni saqlash va unga tezkor kirishni ta’minlashda muhim ahamiyatga ega. Bunda ma’lumotlar bazasining o‘zaro aloqasi va fayl tuzilmalaridan foydalanish tizimning samaradorligini oshiradi.

1. Fayl tuzilmalarining asosiy turlari

Fayl tuzilmalari ma’lumotlarni saqlash va ularga murojaat qilish uchun ishlatiladigan strukturalar bo‘lib, ular quyidagicha bo‘lishi mumkin:

1.1. Ketma-ket (Sequential) fayl tuzilmasi**.** Ma’lumotlar ketma-ket tartibda saqlanadi. Bu tuzilma oddiy va arzon, ammo ma’lumotlarga murojaat qilish vaqtida samarali emas, chunki barcha yozuvlarni qidirish kerak.

1.2. Indekslangan fayl tuzilmasi. Ma’lumotlar faylda indekslar yordamida joylashtiriladi. Indekslar qidiruvni tezlashtirishga yordam beradi, chunki ular ma’lumotlarni tezkor topish imkoniyatini yaratadi.

1.3. Hashlash fayl tuzilmasi. Hashlash texnologiyasi yordamida ma’lumotlar ma’lum bir kalitga asoslanib saqlanadi. Har bir ma’lumotga biror xesh-kod tayinlanadi va bu kod yordamida ma’lumot tezda topiladi.

2. Ma’lumotlar bazasi va fayl tuzilmalari o‘rtasidagi aloqalar

Relyatsion ma’lumotlar bazasi (RDBMS) tizimlarida ma’lumotlar bazasi va fayl tuzilmalari o‘rtasidagi aloqalar juda muhim ahamiyatga ega. Ularning o‘zaro aloqasi quyidagicha amalga oshiriladi:

2.1. Indekslar va ma’lumotlar bazasidagi o‘zaro aloqalar. Ma’lumotlar bazasida indekslar ma’lumotlarga tezkor murojaat qilish uchun ishlatiladi. Indekslar ma’lumotlar bazasi tomonidan yaratiladi va ular ma’lumotlar fayl tuzilmasi bilan bog‘lanadi.

Masalan: Agar ma’lumotlar bazasida talabalar jadvali bo‘lsa va talabaning ID raqami bo‘yicha qidiruv amalga oshirilsa, bu ID raqami bo‘yicha indeks yaratish orqali qidiruvni tezlashtirish mumkin.

2.2. Fayl tuzilmalarining ma’lumotlarni saqlash va qayta ishlashdagi roli. Fayl tuzilmalari ma’lumotlarni saqlash va qayta ishlash jarayonini boshqaradi. Har bir ma’lumot bazasi operatsiyasi (masalan, SELECT, INSERT, UPDATE) o‘z navbatida fayl tuzilmalariga ta’sir qiladi.

Masalan: Ma’lumotlar bazasida yangi ma’lumot qo‘shilganda, bu ma’lumot to‘g‘ridan-to‘g‘ri fayl tuzilmasiga qo‘shiladi yoki indeksga kiritiladi.

6.6. Ma’lumotlar bazasida aloqadorlik chegaralari va xavfsizlik choralarini tasvirlash

Ma’lumotlar bazalarining samarali ishlashi, ma’lumotlar xavfsizligi va tizimning yaxlitligini ta’minlash uchun aloqadorlik chegaralari (referential integrity constraints)

va xavfsizlik choralarining ahamiyati juda katta. Bu ikki element ma’lumotlarni saqlashda va unga murojaat qilishda muhim rol o‘ynaydi.

1. Aloqadorlik chegaralari (Referential Integrity Constraints)

Aloqadorlik chegaralari — bu ma’lumotlar bazasidagi bir jadvalda kiritilgan ma’lumotlar boshqa jadvaldagi ma’lumotlar bilan aloqada bo‘lishi kerakligini ta’minlovchi qoidalar to‘plamidir. Bu chegaralar ma’lumotlar orasidagi bog‘lanishlarni tasvirlaydi va ularni yangilashda yoki o‘chirishda tizim tomonidan ma’lumotlarning yaxlitligini ta'minlaydi. Aloqadorlik chegaralari asosan quyidagi turlarga bo‘linadi:

1.1. Foreign Key Constraints**.** Bu chegaralar bir jadvaldagi ustun (yoki ustunlar) boshqa jadvaldagi bir ustun (yoki ustunlar) bilan bog‘lanishini belgilaydi. Bu aloqada kiritilgan yoki yangilangan ma’lumotlar, tegishli boshqa jadvalda mavjud bo‘lishi kerak. Bu, masalan, talabalar va kurslar jadvali o‘rtasidagi bog‘lanishlarda qo‘llaniladi.

Qoidalar:

ON DELETE CASCADE: Agar birinchi jadvaldagi ma’lumot o‘chirilsa, unga bog‘langan ikkinchi jadvaldagi ma’lumotlar ham avtomatik o‘chiriladi.

ON UPDATE CASCADE: Agar birinchi jadvaldagi ma’lumot yangilansa, unga bog‘langan ikkinchi jadvaldagi ma’lumotlar ham yangilanadi.

Misol:

Talaba (id, ism, kurs_id)

Kurs (kurs_id, kurs_nomi)

Talaba jadvalidagi kurs_id ustuni Kurs jadvalidagi kurs_id ustuni bilan bog‘langan bo‘ladi.

1.2. ON DELETE CASCADE**.** Agar asosiy jadvaldagi yozuv o‘chirilsa, unga bog‘langan boshqa jadvaldagi yozuvlar ham o‘chiriladi. Bu chegara ma’lumotlarning yaxlitligini ta’minlaydi va bog‘langan ma’lumotlarni avtomatik ravishda olib tashlashga yordam beradi.

1.3. ON DELETE RESTRICT**.** Agar asosiy jadvaldagi yozuvni o‘chirishga urinsangiz, tizim bu operatsiyani bloklaydi va sizga avval boshqa bog‘langan yozuvlarni o‘chirishni talab qiladi.

Xavfsizlik choralarini tasvirlash

Ma’lumotlar bazalarida xavfsizlik choralari tizimning ma’lumotlar xavfsizligini ta’minlash uchun juda muhim ahamiyatga ega. Ma’lumotlar bazasining xavfsizligi to‘rtta asosiy choraga bo‘linadi: identifikatsiya, autentifikatsiya, avtorizatsiya va audit.

Identifikatsiya. Identifikatsiya foydalanuvchining tizimga kirishini aniqlash jarayonidir. Bu jarayon, foydalanuvchining tizimga kirishi uchun uning foydalanuvchi nomi va boshqa ma’lumotlari talab etiladi.

Autentifikatsiya. Autentifikatsiya foydalanuvchining haqiqiyligini tekshirish jarayonidir. Odatda, autentifikatsiya foydalanuvchi nomi va parol yordamida amalga oshiriladi, ammo ilgari samarali autentifikatsiya usullari quyidagicha bo‘lishi mumkin:

Ikki faktorli autentifikatsiya (2FA): Foydalanuvchi faqatgina parolni kiritish bilan tizimga kirish huquqiga ega bo‘lmaydi, balki qo‘shimcha tekshiruv (masalan, telefon orqali yuborilgan kod) talab etiladi.

Avtorizatsiya. Avtorizatsiya foydalanuvchiga ma’lumotlar bazasida qanday amallarni bajarishga ruxsat berilganligini aniqlash jarayonidir. Masalan, foydalanuvchiga faqatgina o‘qish (SELECT), o‘zgartirish (UPDATE), qo‘shish (INSERT) yoki o‘chirish (DELETE) huquqlari berilishi mumkin.

Audit va monitoring. Audit tizimga kirish va ma’lumotlarga murojaat qilish jarayonlarini nazorat qilishni ta’minlaydi. Bunda foydalanuvchining barcha faoliyatlari qayd etiladi va ma’lumotlar bazasining xavfsizligi buzilgan taqdirda tizim administratorlari xabardor bo‘ladi.

Misol: Ma’lumotlar bazasidagi barcha kirishlar, ma’lumotlar yangilanishi va o‘zgartirilgan yozuvlar qayd etilib boriladi.

Xavfsizlik choralarini amalga oshirish:

Parolni kuchli qilish

Ma’lumotlar bazasiga kirishda kuchli va murakkab parollarni qo‘llash, ularni muntazam ravishda yangilab borish xavfsizlikni oshiradi. Shuningdek, parollarni shifrlashda hashing algoritmlarini qo‘llash kerak (masalan, SHA-256).

Xavfsiz tarmoq aloqalari

Ma’lumotlar bazasi bilan aloqa tarmog‘ida shifrlashni amalga oshirish. SSL/TLS protokollari yordamida ma’lumotlarni shifrlash, ma’lumotlar bazasi serveriga uzatilayotgan ma’lumotlarning xavfsizligini ta'minlaydi.

Foydalanuvchi rollari va huquqlarini boshqarish

Har bir foydalanuvchiga tegishli ruxsatnomalarni belgilash. Misol uchun, “Admin” foydalanuvchisi barcha amallarni bajarishi mumkin, “User” foydalanuvchisi faqat ma’lumotlarni ko‘rishi mumkin.

SQL injeksiyasiga qarshi himoya

SQL injeksiyasi (SQL Injection) hujumlarini oldini olish uchun ma’lumotlar bazasiga kirishni tekshiruvchi maxsus filtrlar va parametrli so‘rovlar (prepared statements) qo‘llanilishi kerak.

Nazorat savollari:

1. Ma’lumotlar bazasini ishlab chiqishni rejalashtirish bosqichida qanday jarayonlar amalga oshiriladi.
2. Ma’lumotlar bazasini loyihalash jarayoning asosiy uch bosqichin tavsiflang?
3. Konseptual, logik va fizik loyihalash bosqichlarida qaysi asosiy elementlar aniqlanadi?
4. Ma’lumotlar bazasi administratorining (DBA) asosiy vazifalari qaysilar?
5. Ma’lumotlar bazasi administratsiyalash nima?

7 BOB. MA'LUMOTLAR BAZASINI NORMALLASHTIRISH VA 1NF, 2NF, 3NF VA KODD NORMAL FORMALARI

Tayanch so'zlar: *Normallashtirish, funksional bog'lanish, 1NF, 2NF, 3NF, tranzativ bog'lanish, to'liq bog'lanish, qisman bog'lanish*

7.1. Ma'lumotlar bazasini normallashtirish

Ma'lumotlar bazasini normallashtirish jarayoni birinchi marta Edgar F. Codd tomonidan 1970-yillarning boshlarida taklif qilingan. Codd, o'zining "Relational Model of Data for Large Shared Data Banks" maqolasida, ma'lumotlar bazalarini loyihalashda normallashtirish jarayonini tavsiflagan. U ma'lumotlar bazalari nazariyasini rivojlantirishda katta hissa qo'shdi va normallashtirishni ma'lumotlar bazasining dizaynini yaxshilash va takrorlanishlarni kamaytirish uchun muhim vosita sifatida belgiladi. Coddning normallashtirish jarayoni ma'lumotlar bazasidagi aloqalar orasidagi funksional bog'liqliklarni o'rganishga asoslangan edi.

1972-yilda Codd tomonidan birinchi, ikkinchi va uchinchi normal shakllar (1NF, 2NF, 3NF) taklif qilingan va ular ma'lumotlar bazasini normallashtirish jarayonida qo'llaniladi. Keyinchalik, Boyce va Codd (1974) 3NFning kuchliroq versiyasini taklif qilishdi, bu Boyce-Codd normal shakli (BCNF) deb ataladi. Bularning barchasi ma'lumotlar bazasida takrorlanishlarni kamaytirish va ma'lumotlarni kiritish, o'chirish va yangilash bilan bog'liq anomaliyalarni minimallashtirishga qaratilgan.

Normallashtirishning maqsadi — ma'lumotlar bazasining strukturasi samarali, izchil va kengaytirilishi oson bo'lishini ta'minlashdir. Ushbu jarayon quyidagi muammolarni hal qilishga yordam beradi:

Takrorlanishni kamaytirish: Ma'lumotlar bazasida takroriy ma'lumotlar bo'lishining oldini olish.

Anomaliyalarni kamaytirish: Qo'shish, o'chirish va yangilash amallari bilan bog'liq muammolarni (anomaliyalarni) kamaytirish.

Ma’lumotlar yaxlitligini ta’minlash: Ma’lumotlar o‘zaro bog‘lanishlar va munosabatlar orqali izchil bo‘lishini ta’minlash.

Normallashtirish jarayoni funksional bog‘lanishlar (functional dependencies) orqali amalga oshiriladi. Bu bog‘lanishlar ma’lumotlar bazasining tuzilishini tushunishda muhim ahamiyatga ega.

7.2. Funsional bog‘lanishlar va ularning turlari

Funksional bog‘lanishlar (functional dependencies) ma’lumotlar bazasida atributlar o‘rtasidagi bog‘lanishlarni ifodalaydi. Agar jadvaldagi bir atribut (yoki atributlar to‘plami) qiymati boshqa bir atributning qiymatini **aniqlab bersa**, ular orasida funksional bog‘lanish mavjud deb hisoblanadi. Bu bog‘lanish quyidagicha ifodalanadi:

$A \rightarrow B$

Bu yerda:

A – determinant yoki bog‘lanishni aniqlovchi atribut (yoki atributlar to‘plami).

B – aniqlanayotgan atribut.

Yuqoridagi ifoda shuni bildiradiki, **jadvaldagi A qiymati har doim B qiymatini aniqlaydi**. Ya’ni, agar A bir xil qiymatga ega bo‘lsa, u holda B qiymati ham doim bir xil bo‘lishi kerak.

Ular bir atributning (yoki atributlar to‘plamining) qiymatiga asoslangan holda boshqa atributning qiymatini belgilash mumkinligini ko‘rsatadi. Funksional bog‘lanishlar, ma’lumotlar bazasining normalizatsiya jarayonida muhim rol o‘ynaydi, chunki ular ma’lumotlarning tuzilishini tushunishda va normallashtirishni amalga oshirishda yordam beradi.

Funksional bo‘g‘lanishlar quyidagi turlarga bo‘linadi.

To‘liq funksional bog‘lanish (Full Functional Dependency): Bir atribut Y boshqa atribut to‘plami X ga to‘liq bog‘liq bo‘lsa, unda Y ning qiymati X ning

qiymatiga bog‘liq. Bu shuni anglatadiki, agar X dan biror bir atribut olib tashlansa, Y ga bog‘lanish yo‘qoladi.

Namuna.

X Atributlar To‘plami

Bizning **X** atributlar to‘plamimiz quyidagicha bo‘ladi:

Talaba_ID (A) - Talabaning unikal identifikatori

Yunalish (B) - Talabaning o‘qiyotgan yo‘nalishi

Y Atributi

Talaba_Ismi (Y) - Talabaning ismi

7.1-jadval. Talabalar jadvali

Talaba_ID (A)	Yo‘nalish (B)	Talaba_Ismi (Y)
1	Informatika	Azamat Abdullayev
2	Matematika	Laylo Abdurahmonova
3	Informatika	Asqar Qodirov
4	Kimyo	Davron Murodov
5	Fizika	Durdona Rahmonova

Bu jadvalda **Talaba_Ismi (Y)** atributi, **Talaba_ID (A)** va **Yo‘nalish (B)** atributlar to‘plamiga to‘liq bog‘liq.

Agar **Talaba_ID (A)** dan biror bir atribut olib tashlansa (masalan, **Yo‘nalish (B)**), **Talaba_Ismi (Y)** atributining qiymatini aniqlash uchun kerakli ma’lumotlar yo‘qoladi.

Misol uchun, agar **Talaba_ID** 1 bo‘lsa, **Talaba_Ismi** faqat shu **Talaba_ID** ga bog‘liq bo‘ladi. Agar biz **Yo‘nalish** atributini olib tashlasak, **Talaba_Ismi** (Azamat Abdullayev) haqida hech qanday ma’lumotga ega bo‘lmaymiz, chunki ismi faqat **Talaba_ID** orqali aniqlanadi.

Qisman funksional bog‘lanish (Partial Functional Dependency): Agar bir atribut Y to‘plamdagi bir qismi X ga bog‘liq bo‘lsa, lekin X ning barcha atributlariga bog‘liq bo‘lmasa, bu qisman funksional bog‘lanish hisoblanadi.

Bu holat, bir atribut Y ning qiymati atributlar to‘plami X ga qisman bog‘liq bo‘lganda yuzaga keladi. Ya’ni, agar X atributlar to‘plamidan bir yoki bir necha atribut olib tashlanganda, Y atributining qiymati hali ham aniqlanadi.

Bu holatda, Y faqat X ning bir qismini (ya’ni, X ning bir qismini) bilish orqali aniqlanadi. Qisman funksional bog‘lanish ko‘pincha murakkab kalit (yani, bir necha atributlardan iborat kalit) mavjud bo‘lganda uchraydi.

Qisman funksional bog‘lanishni ko‘rsatish uchun quyidagi misolni keltiramiz. Keling, talabalar haqidagi bir jadvalni ko‘rib chiqamiz.

Atributlar to‘plami

Talaba_ID (A) - Talabaning unikal identifikatori

Fan (B) - Talaba o‘qiyotgan kurs

Talaba_Ismi (C) - Talabaning ismi

Professor (D) - Talabaga dars beradigan professor ismi

7.2-jadval. Talabalar jadvali

Talaba_ID (A)	**Fan (B)**	**Talaba_Ismi (C)**	**Professor (D)**
1	Informatika	Azamat Abdullayev	Abdulla Yuldashev
1	Matematika	Azamat Abdullayev	Oleg Ibragimov
2	Matematika	Laylo Abdurahmonova	Oleg Ibragimov
2	Informatika	Laylo Abdurahmonova	Abdulla Yuldashev
3	Kimyo	Asqar Qodirov	Yevgeniy Petrov

Bu jadvalda **Professor (D)** atributi **Fan (B)** atributiga qisman bog‘liq. Agar biz **Talaba_ID (A)** ni bilsak, biz **Talaba_Ismi (C)** ni bilamiz, lekin **Professor (D)** faqat **Kurs (B)** orqali aniqlanadi.

Misol uchun, agar **Kurs** “Informatika” bo‘lsa, bu kursga dars beradigan professor “Abdulla Yuldashev” bo‘lishi mumkin, lekin **Talaba_ID** orqali professor ismini aniqlash imkoniyati yo‘q.

Professor (D) atributi **Kurs (B)** ga qisman bog‘liq, chunki **Professor (D)** faqat **Fan (B)** ga qarab aniqlanadi, lekin **Talaba_ID (A)** orqali bu bog‘lanish aniq emas.

Boshqacha aytganda, agar biz **Fan** atributini bilsak, **Professor** ni aniqlashimiz mumkin, lekin agar **Fan** atributini olib tashlasak, **Professor** atributi haqida hech qanday ma’lumotga ega bo‘lmaymiz.

Transitiv funksional bog‘lanish (Transitive Functional Dependency): Agar X dan Y ga bog‘lanish mavjud bo‘lsa va Y dan Z ga bog‘lanish mavjud bo‘lsa, lekin X dan bevosita Z ga bog‘lanish mavjud bo‘lmasa, bu transitiv bog‘lanish hisoblanadi. Masalan, agar A→B va B→C lekin A dan C ga bog‘lanish yo‘q bo‘lsa, bu transitiv bog‘lanishdir.

Misol.

Tasavvur qilaylik, talabalar haqidagi quyidagi jadval bor:

7.3-jadval. Talabalar jadvali

Talaba_ID (A)	**Fakultet (B)**	**Dekan (C)**
1	Informatika	Akmal Ismoilov
2	Matematika	Shahnoza Karimova
3	Kimyo	Diyor Murodov

Bog‘lanishlar

A → B: Talaba_ID (A) Fakultet (B)ga bog‘liq, ya’ni har bir talabaning fakulteti bor.

B → C: Fakultet (B) esa Dekan (C)ga bog‘liq, ya’ni har bir fakultetga bir dekan biriktirilgan.

Bu yerda **A → C** ga bevosita bog‘lanish yo‘q, chunki Talaba_ID orqali bevosita Dekan aniqlanmaydi. Lekin **A → B** va **B → C** bog‘lanishlar orqali **A dan C ga** transitiv bog‘lanish mavjud.

Bu holatda, **Talaba_ID (A)** orqali fakultet (B) topiladi, va fakultet orqali dekan (C) aniqlanadi. Shunday qilib, A dan C ga bog‘lanish bevosita emas, balki tranzitiv (vositali) bog‘lanish hisoblanadi**.**

7.3. Birinchi normal forma va uning talablari

Ma’lumotlar bazalarini **normallashtirish** turli **normal formalarga** ajratib, ular orasidagi bog‘lanishlarni tahlil qilish orqali **ortiqcha ma’lumotlar** (redundant) va **anomaliyalarni** kamaytirish imkonini beradi.. Quyida 1NF, 2NF, 3NF va Boyce-Codd normal formalari (BCNF) haqida tushuncha beriladi.

Birinchi Normal Forma (1NF)

1NF jadvalda barcha ma’lumotlarning atomar qiymatlarda bo‘lishini ta’minlaydi. Bu bir nechta qiymatlar yoki yig‘ma tuzilmalar (masalan, massiv yoki boshqa jadvallar) mavjudligini rad etadi. Ya’ni har bir katakda faqat bitta qiymat bo‘lishi kerak, ko‘p qiymatlar yoki yig‘ma ma’lumotlar (masalan, ro‘yxatlar, massivlar) saqlanmasligi kerak, jadvalning har bir ustuni faqat bitta turdagi (masalan, ism, telefon raqami) qiymatlarni ifodalashi kerak va bu qiymatlar atomar bo‘lishi lozim. Shunday qilib, har bir katakda yagona qiymat bo‘lishi talab qilinadi. 1NF nafaqat takrorlanishni kamaytiradi, balki qidiruv va yangilash kabi jarayonlarni ham soddalashtiradi.

1NF ga misol.

Talabalar jadvalida quyidagi ustunlar mavjud bo‘lsin:

Student_ID (Talabaning identifikatsion raqami)

Name (Ismi)

Phone_Numbers (Telefon raqamlari)

7.4-jadval. Talaba jadvali

Student_ID	Name	Phone_Numbers
1	Azamat	123-456, 987-654
2	Laylo	456-789

Muammo: 1NF qoidalari talab qiladiki, har bir katakda **atomar qiymat** bo‘lsin, lekin “Phone_Numbers” ustunida bir nechta qiymatlar mavjud.

1NF holatiga keltirish uchun jadvalni qayta tuzib chiqamiz:

7.5-jadval. 1NFga keltirlgan talaba jadvali

Student_ID	Name	Phone_Number
1	Azamat	123-456
1	Azamat	987-654
2	Laylo	456-789

Natija: Bu jadvalda har bir katak atomar qiymatlarga ega.

7.4. Ikkinchi Normal Forma (2NF)

Munosabat yoki jadval ikkinchi normal formada deyiladi, qachonki agar u birinchi normal formada bo‘lsa va har bir birlamchi kalitga kirmagan atribut birlamchi kalit atributlariga to‘liq funsional bog‘liq bo‘lsa.

2NF (Ikkinchi Normal Forma) ga misol.

Misol: O‘qituvchilar va kurslar haqida jadval:

Teacher_ID (O‘qituvchi identifikatori)

Course_ID (Kurs identifikatori)

Teacher_Name (O‘qituvchi ismi)

Course_Name (Kurs nomi)

7.6-jadval. O‘qituvchilar va kurslar haqida jadval

Teacher_ID	**Course_ID**	**Teacher_Name**	**Course_Name**
1	101	Abdulla	Informatika
1	102	Abdulla	Matematika
2	101	Laylo	Informatika
2	103	Laylo	Fizika

Muammo: Bu jadval 1NFda, lekin 2NF talablariga javob bermaydi, chunki unda **qisman funksional bog‘lanishlar** mavjud. Masalan, Teacher_Name faqat Teacher_ID ga, Course_Name esa faqat Course_ID ga bog‘liq.

2NF ga keltirish uchun jadvalni quyidagicha bo‘lish kerak:

7.7-jadval. Teachers jadvali:

Teacher_ID	Teacher_Name
1	Abdulla
2	Laylo

7.8-jadval. Courses jadvali:

Course_ID	Course_Name
101	Informatika
102	Matematika
103	Fizika

Bu tarzda jadvaldagi qisman bog‘lanishlar bartaraf etiladi.

7.5. Uchinchi Normal Forma (3NF)

3NF 2NF shartlariga rioya qilishi kerak va unda transitiv funksional bog‘lanishlar bo‘lmasligi talab qilinadi. Bu shuni anglatadiki, agar A → B va B → C bo‘lsa, lekin A → C bo‘lmasa, bu tranzitiv bog‘lanish hisoblanadi. Ushbu normal forma takrorlanadigan ma’lumotlar miqdorini kamaytiradi va tranzitiv bog‘lanishlarni bartaraf etadi, bu esa ma’lumotlar bazasi izchilligini oshiradi.

3NF (Uchinchi Normal Forma) ga misol

Masala: Quyidagi talaba jadvalini ko‘rib chiqamiz:

Student_ID (Talabaning identifikatsion raqami)

Student_Name (Talabaning ismi)

Advisor_ID (Maslahatchi identifikatori)

Advisor_Name (Maslahatchining ismi)

7.9-jadval. Talaba jadvali

Student_ID	Student_Name	Advisor_ID	Advisor_Name
1	Azamat	501	Akmal
2	Laylo	502	Rashid

Muammo: Ushbu jadval 2NFda bo‘lsa ham, unda **transitiv funksional bog‘lanish** mavjud, chunki Student_ID → Advisor_ID va Advisor_ID → Advisor_Name, lekin Student_ID dan bevosita Advisor_Name ga bog‘lanish mavjud emas.

3NF holatiga keltirish uchun jadvalni bo‘lish kerak:

7.10-jadval. Students jadvali**:**

Student_ID	Student_Name	Advisor_ID
1	Azamat	501
2	Laylo	502

7.11-jadval. Advisors jadvali**:**

Advisor_ID	Advisor_Name
501	Akmal
502	Rashid

Bu tarzda tranzitiv bog‘lanish bartaraf qilinadi.

7.6. Kodd normal formasi

Boyce-Codd normal formasi, yoki BCNF, 3NF ni kuchaytirilgan shakli bo‘lib, **kalitdan tashqari bo‘lgan barcha funksional bog‘lanishlarni yo‘q qiladi**. Agar jadval 3NF shartlariga javob bersa-yu, lekin asosiy bo‘lmagan atribut kalit sifatida ishlatilgan bo‘lsa, u BCNF shartlariga mos kelmaydi. Shuning uchun BCNF barcha funksional bog‘lanishlar kalitlar orqali ifodalanishini talab qiladi.

BCNF Codd va Boyce tomonidan ishlab chiqilgan bo‘lib, E. F. Coddning “Further Normalization of the Relational Database Model” (1972) ilmiy maqolasida tahlil qilingan. Bu normal forma murakkabroq tuzilmalar va qo‘shimcha takrorlanadigan bog‘lanishlarni bartaraf etish uchun qo‘llaniladi.

4NF, 3NF dan keyin keladi va ko‘p qiymatli bog‘lanishlarni bartaraf etish uchun ishlatiladi. Agar bir jadvalda ikkita yoki undan ortiq ko‘p qiymatli bog‘lanishlar mavjud bo‘lsa, bu 4NF ga olib kelinishi kerak.

5NF, bu normal forma har bir atribut o‘zining atributlari bilan bog‘liq bo‘lishini ta’minlaydi. Agar ma’lumotlar bir-biriga bog‘liq bo‘lsa, bu bog‘lanishni alohida jadvalga ajratish orqali bartaraf etiladi.

6NF, vaqt o‘tishi bilan o‘zgaruvchi ma’lumotlarni saqlashda foydalaniladi. Bu normal forma vaqt va fazoviy o‘zgarishlar kabi dinamik ma’lumotlar uchun foydalidir. 6NF, 5NF dan kelib chiqadi va u juda aniq maqsadga ega bo‘lgan tizimlar uchun qo‘llaniladi.

Shunday qilib, ma’lumotlar bazasining normallashtirish jarayoni birinchi normal formadan to‘rtinchi normal formagacha va hatto 6-normal forma gacha davom etishi mumkin. Har bir normal forma o‘ziga xos qoidalar va talablarni o‘z ichiga oladi, bu esa ma’lumotlarning redundansini kamaytirish imkonini beradi.

7.7. Berilgan munosabatni bir necha marta oddiy va kichik munosabatlarga ajratish

Munosabatni oddiy va kichik munosabatlarga ajratish normalizatsiya jarayoni bilan bog‘liq bo‘lib, bu amaliyot ma’lumotlar bazalarining samaradorligini oshirish va mantiqiy mustahkamligini ta’minlash uchun muhim ahamiyatga ega.

Demak, normalizatsiya – bu ma’lumotlar bazasini kichik, mantiqan bog‘langan jadvallarga ajratish va ular orasidagi munosabatlarni o‘rnatish jarayoni. Ushbu jarayon orqali ma’lumotlarning ortiqchaligini kamaytirish va ma’lumotlar bir-biriga zid kelishining oldini olish maqsad qilingan.

Amaliy misol.

Berilgan munosabat:

Talabalar(T_ID, Ism, Guruh, Fan, Baho)

Bu jadvalda:

- **T_ID** – talabalar jadvali uchun birlamchi kalit.
- Ushbu jadvalda **Guruh**, **Fan**, va **Baho** ortiqcha ma’lumotni saqlashi mumkin.

Ajratish:

1-bosqich. 1NF ga keltiramiz.

1-Jadval:

Talaba(T_ID, Ism, Guruh)

2-Jadval:

Baholar(T_ID, Fan, Baho)

2-bosqich. 2NF ga keltiramiz.

Agar guruh va fan orasida qisman bog‘liqlik aniqlansa, jadval quyidagicha bo‘linadi:

1-jadval.

Talabalar(T_ID, Ism, Guruh)

2-jadval.

Fanlar(Fan, Guruh)

3-jadval.

Baholar(T_ID, Fan, Baho)

3NF: Agar boshqa mantiqiy bog‘liqliklar aniqlansa, ular alohida jadvalga ajratiladi.

Normalizatsiya orqali ma’lumotlarning ortiqchaligi kamayadi, yozuvlar oson yangilanadi, va mantiqiy bog‘liqliklar tushunarli bo‘ladi. Ushbu jarayon ma’lumotlar bazasini yanada samarali boshqarish imkonini beradi.

Nazorat savollari:

1. Funksional bog‘lanishlar nima?
2. Funksional bog‘lanish turlarini misollar yordamida tushuntiring?
3. Normallashtirish nima?
4. Normalashtirishning qanday turlari bor?
5. Normal formalarning biri-biridan farqli jihatlari.

8 BOB. SQLTILI VA SQL OPERATORLARINI YOZISH

Tayanch so‘zlar: *System R, SEQUEL, SQL, SELECT, DISTINCT, DDL, DML, VAR, VARCHAR, DATE, FROM, WHERE, ORDER BY, GROUP BY, HAVING, UNION.*

8.1. SQL tilining vazifalari

1970-yillarda IBM tadqiqotchisi Edgar F. Codd tomonidan ishlab chiqilgan munosabatli model, SQL uchun nazariy asos bo‘lib xizmat qildi. Bu modelda ma’lumotlar jadvallar shaklida ifodalanadi va ular orasidagi bog‘lanishlar aniq belgilangan. Coddning 1970-yilda chop etilgan *“A Relational Model of Data for Large Shared Data Banks”* maqolasi SQL tilining rivojlanishida muhim ahamiyatga ega bo‘ldi. Bu maqolada u ma’lumotlarni boshqarishning yangi tizimini taklif qilgan bo‘lib, keyinchalik ushbu tizim asosida SQL tili yaratilgan.

SQLning boshlanishi: 1970-yillar

SQL ning rivojlanishining boshlanishi 1970-yillarga borib taqaladi. 1970-yillarda IBM tadqiqotchisi Edgar F. Codd tomonidan ishlab chiqilgan relyatsion model, SQL uchun nazariy asos bo‘lib xizmat qildi. Bu davrda **Edgar F. Codd** tomonidan **relatsion ma’lumotlar modeli** (Relational Model) taklif qilindi. Codd 1970-yilda **“A Relational Model of Data for Large Shared Data Banks”** nomli maqolasini e’lon qilgan va unda ma’lumotlarni tashkil etishning yangi, eng yaxshi usulini taqdim etgan. Bu model iyerarxik va tarmoq modellaridan farq qilgan. Bu modelda ma’lumotlar jadvallar shaklida ifodalanadi va ular orasidagi bog‘lanishlar aniq belgilangan. Bu maqolada u ma’lumotlarni boshqarishning yangi tizimini taklif qilgan bo‘lib, keyinchalik ushbu tizim asosida SQL tili yaratilgan.

SQLning paydo bo‘lishi: 1974-1975 yillar

SQLning dastlabki versiyalari 1974-yilda IBM kompaniyasida ishlab chiqilgan System R nomli tadqiqot loyihasi asosida paydo bo‘ldi. Bu tizimda dastlab SQL ishlatilgan va tizimning maqsadi foydalanuvchilarga ma’lumotlar bazasiga so‘rovlar

yuborish imkoniyatini berish edi. System R — bu IBM kompaniyasining 1970-yillarda amalga oshirilgan tadqiqot loyihasi bo‘lib, ushbu loyiha SQL tilining rivojlanishi uchun katta ahamiyatga ega bo‘lgan dastlabki tizimlardan biri hisoblanadi. System R loyihasi IBM tomonidan 1974-yilda ishga tushirildi. Uning maqsadi relatsion model asosida ishlovchi tizim yaratish edi. Edgar F. Coddning relyatsion ma’lumotlar modeli hamda SQL (Structured Query Language) tilining dastlabki versiyalarini amaliyotda sinovdan o‘tkazish uchun bu loyiha ishlab chiqildi.

Coddning Relyatsion Modeli: Codd 1970-yilda o‘zining mashhur **“A Relational Model of Data”** maqolasini e’lon qilgan va unda ma’lumotlarni tasvirlashda yangi, samarali modelni taqdim etgan. System R loyihasida aynan shu modelga asoslanib tizim yaratildi.

System R loyiha doirasida **SQL** (Structured Query Language) tilining dastlabki versiyasi ishlab chiqildi. SQL dastlab **SEQUEL** (Structured English Query Language) deb nomlangan, ammo keyinchalik SQL nomi qabul qilindi. Bu til foydalanuvchilarga ma’lumotlar bazasidan ma’lumotlarni so‘rov qilish va boshqarish imkoniyatini berdi.

SQL tilining rivojlanishi Oracle kabi kompaniyalar tomonidan ham qo‘llab-quvvatlandi. Oracle 1979-yilda SQL asosida birinchi tijorat ma’lumotlar bazasi dasturini chiqardi. Shu tariqa, SQL tezda mashhurlikka erishdi va turli kompaniyalarning ma’lumotlar bazasi tizimlarida ishlatiladigan standart tilga aylandi.

SQLning Standartlashuvi: 1980-yillar

SQL dastlab faqat oddiy so‘rovlar va oddiy ma’lumotlarni boshqarish uchun mo‘ljallangan bo‘lsada, vaqt o‘tishi bilan unga murakkabroq funksiyalar, tranzaksiyalarni boshqarish va boshqa imkoniyatlar qo‘shildi. Bu kengayishlar SQLning bugungi zamonaviy dasturlar bilan moslashuvchan bo‘lishiga va keng ko‘lamli tizimlarda ishlatilishiga imkon berdi.

SQL dastlab ma’lumotlar bazalarida ma’lumotlarni kiritish, yangilash, o‘chirish va qidirish uchun ishlatilgan. SQLning rivojlanishi esa uning **murakkab so‘rovlar**

(complex queries), **tranzaksiya boshqaruvi** (transaction management) va **ma'lumotlar integratsiyasi** (data integration) kabi imkoniyatlar bilan kengayishiga olib keldi.

1986-yilda ANSI (American National Standards Institute) SQLni standart tili sifatida qabul qildi. SQLning rivojlanishida ISO (International Organization for Standardization) ham muhim rol o'ynadi, chunki u SQLning xalqaro miqyosda standartlashgan bo'lishini ta'minladi. 1986-yildan boshlab SQL rasmiy standart sifatida qabul qilindi va o'sha davrda SQL-86 deb nomlangan versiya chiqdi.

SQLning yirik o'zgarishlari va rivojlanishi: 1990-yillar

1990-yillarda SQL ko'plab rivojlanish bosqichlarini boshdan kechirdi. Bu davrda SQL standarti ba'zi o'zgarishlar va yangi imkoniyatlar bilan boyitildi, shu jumladan:

SQL92 (yoki SQL2) standarti: Yangi versiyalarni qo'llab-quvvatlashni, yangi sintaksis va xususiyatlarni, masalan, tashqi kalitlar (foreign keys), indekslar va boshqalarni o'z ichiga oldi.

SQL99 (yoki SQL3): Yangi ma'lumot turlari, masalan, ob'ektga yo'naltirilgan xususiyatlar va triggerlarni qo'llab-quvvatlash.

SQLni bugungi kundagi rivojlanishi

Bugungi kunda SQL ko'plab rivojlangan versiyalarga ega. Asosiy rivojlanishlar quyidagilarni o'z ichiga oladi:

NoSQL va SQL integratsiyasi: SQLning NoSQL ma'lumotlar bazalari bilan integratsiyasi, bu SQLni yangi turdagi ma'lumotlar va ilovalar bilan ishlash uchun ko'proq moslashuvchan qiladi.

Cloud Computing: SQLni bulutda ishlashga tayyorlash, masalan, Amazon Web Services (AWS), Google Cloud Platform (GCP) va Microsoft Azure kabi bulutli xizmatlar SQL ma'lumotlar bazalarini taqdim etmoqda.

Big Data: Big Data va SQL aloqasi ya'ni SQLni katta hajmdagi ma'lumotlar bilan ishlash uchun moslashtirish.

SQLning asosiy funksiyalari va rivojlanishi, ma'lumotlar bazalarini boshqarishda turli xil tizimlarning moslashuvchanligini va samaradorligini oshirdi.

SQLning bugungi zamonaviy versiyalari, ma'lumotlar bazasining turli turlari va tizimlari bilan ishlash uchun zarur bo'lgan ko'plab funksiyalarni qo'llab-quvvatlaydi.

SQLni yangi texnologiyalar, masalan, **big data** va **cloud computing** bilan moslashtirish uchun bir qator yangi imkoniyatlar ham qo'shildi.

SQL tili bugungi kunda ma'lumotlar bazasini boshqarishning asosiy tiliga aylangan va ko'plab tizimlar, jumladan MySQL, PostgreSQL, Oracle va Microsoft SQL Server kabi tijorat va ochiq manbali tizimlarda ishlatiladi.

SQLning Asosiy Tamoyillari:

Ma'lumotlarning jadvallar shaklida tashkil etilishi: SQL relyatsion ma'lumotlar bazasida ishlaydi, ya'ni ma'lumotlar turli jadvallarda saqlanadi. Har bir jadval qatorlar va ustunlar (atributlar) shaklida tashkil etilgan.

Deklarativ tildan foydalanish: SQL deklarativ til bo'lib, foydalanuvchi ma'lumotni qanday olish kerakligini belgilaydi, ammo SQL tizimi bu jarayonni qanday amalga oshirishi kerakligini biladi. Masalan, SELECT so'rovi yordamida ma'lumotni qanday qidirish kerakligi belgilanadi, lekin SQL bu jarayonni qanday optimallashtirishni o'zi hal qiladi.

Indekslar va optimizatsiya: SQL ma'lumotlar bazasining samarali ishlashi uchun indekslar yaratish imkonini beradi. Bu indekslar so'rovlar tezligini oshiradi va ma'lumotlarni izlashni tezlashtiradi.

Tranzaksiya boshqaruvi: SQL tranzaksiya boshqaruvi uchun qulayliklar taqdim etadi. Tranzaksiyalar bir nechta operatsiyalarni bir vaqtda amalga oshirishga imkon beradi va agar biron bir xatolik yuz bersa, barcha o'zgarishlar orqaga qaytariladi.

Bu tamoyil **ACID** (Atomicity, Consistency, Isolation, Durability) deb ataladi va SQLda har bir tranzaksiya shu tamoyilga muvofiq amalga oshiriladi.

Integratsiya va kengaytirilish: SQL boshqa tizimlar va formatlar bilan ishlash uchun kengaytirilgan imkoniyatlarni taklif qiladi. Masalan, **XML** va **JSON** ma'lumotlarini ishlash, **OLAP (On-Line Analytical Processing)** kabi analitik so‘rovlarni amalga oshirish uchun SQL kengaytirilgan.

SQL tilining asosiy vazifalari - so‘rov yaratish, ma'lumotlarni kiritish, yangilash, o‘chirish, jadval tuzish va boshqarish, ma'lumotlar bazasini boshqarish, cheklovlar o‘rnatish, optimallashtirish va transaksiyalarni boshqarishdan iborat. Bu vazifalar SQLni ma'lumotlar bazasini boshqarishda samarali vositaga aylantiradi. SQLning rivojlanishi va uning so‘nggi versiyalari ma'lumotlar bazalarini boshqarishda keng qo‘llaniladi.

8.2. Interaktiv va qurilgan SQL

Interaktiv SQL — bu foydalanuvchi tomonidan ma'lumotlar bazasiga yuborilgan so‘rovlar natijasini darhol olishga imkon beruvchi usul. Foydalanuvchi so‘rov yuboradi, tizim esa javobni tezda ko‘rsatadi. Bu usulda foydalanuvchi SQL buyrug‘ini bevosita tizimga kiritadi va natijalarni interaktiv tarzda ko‘radi. Interaktiv SQL ko‘pincha ma'lumotlar bazasida tezkor testlar o‘tkazish, ma'lumotlarni so‘rov qilish va boshqarish uchun ishlatiladi. Bu usul o‘qituvchilar, tadqiqotchilar va ma'lumotlar bazasi administratorlari tomonidan ma'lumotlarni tezda tekshirish va o‘zgartirish uchun foydalidir.

Interaktiv SQLga misol: Foydalanuvchi quyidagi **SELECT** so‘rovini tizimga kiritadi va natijani darhol ko‘radi:

```
SELECT * FROM Students WHERE Age > 18;
```

Bu so‘rov foydalanuvchiga 18 yoshdan katta bo‘lgan barcha talabalarning ma'lumotlarini ko‘rsatadi.

Interaktiv SQL ko‘proq ma’lumotlarni tezkor ko‘rish va tahlil qilish uchun mo‘ljallangan.

Qurilgan SQL (Embedded SQL) — bu dasturlash tillarida SQL buyruqlarini bevosita kiritish imkoniyatini beruvchi usul. O‘rnatilgan SQL ma’lum bir dasturlash tilida (masalan, C, C++, Java) SQL buyruqlarini o‘z ichiga oladi. Bu usulda SQL so‘rovlari dastur ichida joylashadi va ishlov berilgan so‘rovlar ma’lumotlar bazasi bilan aloqada bo‘ladi.

O‘rnatilgan SQL dasturlash tilida dinamik SQL buyruqlarini yaratish imkoniyatini beradi. Bu usulda dasturchi SQL kodlarini o‘zgaruvchilarga va shartlarga asoslangan holda yaratishi mumkin. Qurilgan SQL ko‘pincha katta hajmdagi tizimlarda yoki ma’lumotlar bazalari bilan integratsiya qilishda ishlatiladi.

Qurilgan SQL Misol: Quyidagi kodda SQL buyruqlari **C** dasturlash tilida ishlatilgan:

```
EXEC SQL SELECT COUNT(*) INTO :count FROM Students WHERE Age > 18;
```

Bu misolda, **EXEC SQL** direktivasi yordamida SQL so‘rovi C dasturida ishlatilgan. **SELECT COUNT(*)** so‘rovi bazadan 18 yoshdan katta bo‘lgan talabalarning sonini olish uchun ishlatilgan va natija dasturdagi **count** o‘zgaruvchisiga saqlanadi.

Qurilgan SQL dasturchilarga SQL kodlarini dastur bilan birlashtirish, ma’lumotlarni dinamik tarzda boshqarish va tizimlarga bog‘langan ma’lumotlarni boshqarishni ta’minlash imkoniyatini beradi.

Farqlar va qo‘llanilishi

Interaktiv SQL:

- Foydalanuvchi tizimga bevosita SQL so‘rovlarini yuboradi.
- Asosan tezkor ma’lumotlar so‘rovlarini amalga oshirish uchun ishlatiladi.
- Ko‘pincha ma’lumotlar bazasi administratori yoki tahlilchi tomonidan ishlatiladi.

Qurilgan SQL:

- Dasturlash tili ichida SQL buyruqlari mavjud bo‘lib, ular dastur ishga tushirilganda bajariladi.
- Katta tizimlarda yoki integratsiyalashgan ma’lumotlar bazalari bilan ishlash uchun ishlatiladi.
- Dasturchilar tomonidan, ko‘proq tizimni SQL bilan bog‘lash uchun ishlatiladi.

Demak, **Interaktiv SQL** tizimning SQL buyruqlarini tezkor va oson amalga oshirishni ta’minlaydi, foydalanuvchiga ma’lumotlarni so‘rov qilishda samarali bo‘lish imkonini beradi. **Qurilgan SQL** esa dasturlash tizimlarida SQL so‘rovlarini bajarish va tizimni ma’lumotlar bazasi bilan integratsiya qilishda ishlatiladi, bu dasturchilarga tizimlarni yanada moslashuvchan va dinamik qilish imkoniyatini beradi.

8.3. SQL ma’lumot toifalari va ular bilan ishlash

SQL tilida bir nechta turdagi ma’lumot toifalari mavjud. Ular asosan quyidagi asosiy toifadagi ma’lumotlarni o‘z ichiga oladi: **raqamli**, **matnli**, **sana va vaqt**. Har bir ma’lumot turi ma’lum bir maqsad uchun ishlatiladi.

Raqamli ma’lumotlar toifalari

Raqamli ma’lumotlar toifalari sonlarni saqlash uchun ishlatiladi. Ular butun sonlar, o‘nlik sonlar va boshqa raqamli qiymatlarni saqlash imkonini beradi.

INT: Butun sonlar (masalan, -10, 0, 100).

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

FLOAT yoki **REAL**: haqiqiy sonlar uchun ishlatiladi.

Matnli ma’lumotlar toifalari

Matnli ma’lumotlar toifalari, asosan, so‘zlar, jumlalar yoki boshqa matnlar uchun ishlatiladi.

CHAR(n): Belgilangan uzunlikdagi matn (masalan, CHAR(10) – 10 belgidan iborat).

VARCHAR(n): Uzunligi o‘zgaruvchan matn (masalan, VARCHAR(255) – maksimal 255 belgi).

TEXT: Katta hajmdagi matn (bu juda uzun matnlar uchun) uchun ishlatiladi.

Sana va vaqt ma’lumotlar toifalari

Sana va vaqt turlari vaqtni saqlash va ishlash uchun ishlatiladi.

DATE: Sana (masalan, 2024-11-06).

TIME: Vaqt (masalan, 12:45:00).

DATETIME yoki **TIMESTAMP**: Sana va vaqt birikmasi (masalan, 2024-11-06 12:45:00).

Boshqa ma’lumot turlari

Bundan tashqari, ba’zi boshqa maxsus ma’lumot turlari ham mavjud:

BOOLEAN: Ha/Yo‘q (masalan, TRUE, FALSE).

BLOB: Katta hajmdagi binar ma’lumotlar (masalan, rasm, audio yoki video fayllar) uchun ishlatiladi.

8.4. SQL tilining komandalarini tuzilishi va sintaksisi

SQL komandalarining asosiy sintaksisi quyidagicha:

```
COMMAND [OPTIONAL PARAMETERS] [CONDITIONS];
```

COMMAND: SQL komandasining o‘zi (masalan, SELECT, INSERT, UPDATE, DELETE).

OPTIONAL PARAMETERS: Parametrlar yoki argumentlar (masalan, ustun nomlari, qiymatlar).

CONDITIONS: Komanda natijalariga qo‘shilgan shartlar (masalan, WHERE sharti).

SQL tilidagi barcha komandalar o‘zining tuzilishi va sintaksisi bilan aniq belgilab qo‘yilgan. Komandalarni bajarishda asosiy tamoyil - ma’lumotlarni to‘g‘ri manipulyatsiya qilishdir.

SQL komandalarini bir nechta kategoriyalarga bo‘lish mumkin. Ularning har biri ma’lumotlar bazasi bilan ishlashning turli vazifalarini bajaradi. Quyida asosiy SQL komandalari turlari va ularning sintaksisi keltirilgan.

DML (Data Manipulation Language) – bu ma’lumotlarni manipulyatsiya qilish uchun ishlatiladigan SQL komandalar to‘plamidir. DML yordamida ma’lumotlarni qo‘shish, o‘zgartirish, o‘chirish va olish (so‘rash) mumkin. DML komandalarini ishlatish orqali foydalanuvchi ma’lumotlar bazasidagi ma’lumotlar bilan ishlaydi.

DML komandalarining asosiy turlari:

1. **INSERT**: Yangi ma’lumotni jadvalga kiritish.
2. **UPDATE**: Jadvaldagi mavjud ma’lumotlarni yangilash.
3. **DELETE**: Jadvaldagi ma’lumotlarni o‘chirish.
4. **SELECT**: Jadvaldagi ma’lumotlarni so‘rash (dastlabki qiymatlar bilan ishlash).

DDL (Data Definition Language) – bu ma’lumotlar bazasining tuzilishini aniqlash, yaratish va o‘zgartirish uchun ishlatiladigan SQL komandalar to‘plamidir. DDL komandalarining yordamida foydalanuvchilar ma’lumotlar bazasi strukturasi, jadval, indekslar, ko‘rinishlar va boshqa ob’ektlarni yaratish, o‘zgartirish yoki o‘chirish mumkin. DDL SQL tilining asosiy qismlaridan biridir va odatda ma’lumotlar bazasining tuzilishini boshqarish uchun ishlatiladi.

DDL Komandalari:

CREATE: Yangi ob’ekt (masalan, jadval, indeks, ko‘rinish) yaratish uchun ishlatiladi.

ALTER: Mavjud ob’ektni o‘zgartirish uchun ishlatiladi.

DROP: Ob’ektni (masalan, jadvalni yoki indeksni) o‘chirish uchun ishlatiladi.

TRUNCATE: Jadvaldagi barcha ma’lumotlarni o‘chirish uchun ishlatiladi, lekin jadvalning tuzilishi saqlanadi.

RENAME: Mavjud ob’ektning nomini o‘zgartirish uchun ishlatiladi.

DCL (Data Control Language) – bu SQL tilining bir qismi bo‘lib, ma’lumotlar bazasidagi foydalanuvchilar uchun ruxsatlar va huquqlarni boshqarish uchun ishlatiladi. DCL yordamida foydalanuvchilarga ma’lumotlar bazasiga kirish huquqi beriladi yoki olinadi. Bu, asosan, xavfsizlik va huquqni ta’minlash uchun zarur bo‘lgan amallarni bajaradi.

DCL Komandalari:

GRANT – Foydalanuvchiga yoki foydalanuvchi guruhiga ma’lum huquqlarni berish uchun ishlatiladi.

REVOKE – Foydalanuvchidan yoki foydalanuvchi guruhidan oldin berilgan huquqlarni olish uchun ishlatiladi.

1. GRANT Komandasi

GRANT komandasining yordamida foydalanuvchilarga ma’lum bir operatsiyani amalga oshirish huquqi beriladi. Bu komanda ma’lum bir foydalanuvchiga yoki guruhga ruxsatnoma (permissions) beradi. **GRANT** komandasini ishlatish orqali, administratorlar quyidagi huquqlarni foydalanuvchilarga taqdim etishlari mumkin:

- ✓ Jadval yaratish
- ✓ Ma’lumotlarni qo‘shish, o‘zgartirish yoki o‘chirish
- ✓ Ma’lumotlarni so‘roq qilish (SELECT)
- ✓ Jadvalga indekslar qo‘shish

Sintaksis:

```
GRANT privilege_type ON object TO user;
```

Misol:

```
GRANT SELECT, INSERT ON Students TO John;
```

Bu misolda, **John** foydalanuvchisiga **Students** jadvalidan ma’lumotlarni o‘qish (SELECT) va qo‘shish (INSERT) huquqlari beriladi.

2. REVOKE Komandasi

REVOKE komandasining yordamida, ilgari **GRANT** komandasini ishlatib, foydalanuvchiga berilgan huquqlarni olib tashlash mumkin. Agar biron bir foydalanuvchidan huquq olinishi kerak bo‘lsa, **REVOKE** komandasidan foydalaniladi. Bu komanda yordamida, aynan belgilangan foydalanuvchidan yoki guruhdan ma’lum bir huquq olib tashlanadi.

Sintaksis:

```
REVOKE privilege_type ON object FROM user;
```

Misol:

```
REVOKE INSERT ON Students FROM John;
```

Bu misolda, **John** foydalanuvchisidan **Students** jadvalidan ma’lumot qo‘shish (INSERT) huquqi olib tashlanadi.

DCLning Asosiy Xususiyatlari:

Xavfsizlikni ta’minlash: DCL komandalarining asosiy vazifasi ma’lumotlar bazasiga kirishni boshqarish va foydalanuvchilar uchun kerakli huquqlarni taqdim etishdir. Bu tizimni noxush vaziyatlardan (masalan, ma’lumotlarni o‘chirish yoki o‘zgartirishdan) himoya qilishga yordam beradi.

Foydalanuvchi huquqlarini boshqarish: DCL yordamida ma’lum foydalanuvchilarga ma’lumotlar bazasining turli bo‘limlariga kirish ruxsatlari beriladi. Masalan, ma’lumotlar bazasining ba’zi qismlariga faqat ko‘rish (SELECT) huquqi, ba’zi qismlariga esa qo‘shish (INSERT) yoki o‘zgartirish (UPDATE) huquqlari berilishi mumkin.

Ma’lumotlar bazasi xavfsizligini oshirish: **GRANT** va **REVOKE** komandalarini bajarish orqali, tizim administratori ma’lumotlar bazasining

xavfsizligini oshirishi mumkin, bu esa tizimning foydalanuvchilariga cheklovlarni o‘rnatish va shaxsiy ma’lumotlarni himoya qilishga yordam beradi.

Dinamik boshqaruv: DCL komandalarini ishlatish juda tez-tez talab qilinadi, chunki ma’lumotlar bazasiga ruxsat berish yoki olib tashlash juda dinamik jarayon hisoblanadi. Foydalanuvchi yangi bo‘lsa yoki mavjud foydalanuvchi o‘zgarib borsa, tezkor ravishda huquqlarni boshqarish kerak bo‘ladi.

8.5. SQL tilining SELECT (tanlash) operatori va uning parametrlari

SQL tilining SELECT operatori – bu SQL tilining eng muhim va ko‘p ishlatiladigan komandalaridan biridir. SELECT operatori ma’lumotlar bazasidan ma’lumotlarni olish uchun ishlatiladi. Bu komanda orqali ma’lum bir jadvaldan yoki bir nechta jadvaldan kerakli ma’lumotlar tanlab olinadi.

Select operatorining sintaksisi:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition
GROUP BY column
HAVING condition
ORDER BY column;
```

SELECT: Bu operator tanlanadigan ustun yoki ustunlarni belgilash uchun ishlatiladi.

FROM: Ma’lumotlar olinadigan jadval nomini belgilaydi.

WHERE: Ma’lumotlar tanlashda qo‘llaniladigan shartni belgilaydi.

GROUP BY: Ma’lumotlarni guruhlash uchun ishlatiladi.

HAVING: Guruhlashdan so‘ng shartlarni qo‘llash uchun ishlatiladi.

ORDER BY: Natijani tartiblash uchun ishlatiladi.

SELECT operatorining asosiy parametrlari

Column (ustunlar) – Tanlanadigan ma'lumotlarni belgilaydi. Agar barcha ustunlar kerak bo'lsa, * (yulduzcha) ishlatiladi.

SELECT * FROM Students; – Bu komanda Students jadvalidan barcha ustunlarni tanlaydi.

SELECT name, age FROM Students; – Bu komanda faqat name va age ustunlarini tanlaydi.

FROM (Jadval) – Ma'lumotlar olinadigan jadval nomi.

SELECT * FROM Employees; – Bu komanda Employees jadvalidan barcha ma'lumotlarni tanlaydi.

WHERE (Shart) – Natija ma'lumotlarini filtrlaydigan parametr.

SELECT * FROM Students WHERE Age > 18; – Bu komanda Students jadvalidan yoshini 18 dan katta bo'lgan talabalarni tanlaydi.

GROUP BY – Guruhlashni amalga oshiradi, ko'pincha agregat funksiyalari bilan birga ishlatiladi (masalan, COUNT, SUM, AVG).

SELECT department, COUNT(*) FROM Employees GROUP BY department; – Bu komanda Employees jadvalidagi xodimlarni department bo'yicha guruhlaydi va har bir bo'limdagi xodimlar sonini hisoblaydi.

HAVING – GROUP BY bilan ishlatilganda shart qo'yish uchun ishlatiladi.

SELECT department, COUNT() FROM EMPLOYEES GROUP BY DEPARTMENT HAVING COUNT() > 5; – Bu komanda, faqat 5 va undan ko'proq xodimi bo'lgan bo'limlarni ko'rsatadi.

ORDER BY – Ma'lumotlarni tartiblash uchun ishlatiladi. Bu parametr **ASC** (ko'tarilish tartibida) yoki **DESC** (pasayish tartibida) bilan birga ishlatiladi.

SELECT * FROM Employees ORDER BY salary DESC; – Bu komanda **Employees** jadvalidan barcha ma'lumotlarni salary ustuni bo'yicha pasayish tartibida tartiblaydi.

DISTINCT – Duplicatlarni (takroriy qiymatlarni) olib tashlash uchun ishlatiladi.

SELECT DISTINCT department FROM Employees; – Bu komanda faqat har bir bo‘limni bir marta ko‘rsatadi.

LIMIT/OFFSET – Tanlanadigan natijalar sonini chegaralash uchun ishlatiladi.

SELECT * FROM Employees LIMIT 10; – Bu komanda faqat birinchi 10 ta xodimni tanlaydi.

SELECT * FROM Employees OFFSET 10 LIMIT 5; – Bu komanda 10 ta xodimni o‘tkazib yuborib, keyingi 5 ta xodimni tanlaydi.

SELECT operatori SQL tilining ajralmas qismi bo‘lib, ma’lumotlar bazasida saqlangan ma’lumotlarni so‘rash, tanlab olish uchun ishlatiladi. SQL 1970-yillarda IBM tomonidan ishlab chiqilgan va "System R" loyihasining natijasi sifatida paydo bo‘lgan. SQLning ilk versiyasida faqat oddiy SELECT so‘rovlari mavjud bo‘lgan bo‘lsa, keyinchalik tilda yangi imkoniyatlar kiritildi, masalan, GROUP BY, HAVING va ORDER BY kabi ko‘plab kengaytmalar paydo bo‘ldi.

Nazorat savollari:

1. Sqlning rivojlanish bosqichlarini tushuntirib bering?
2. DML ning asosiy vazifalarini ayting?
3. DDL ning asosiy vazifalarini ayting?
4. DCL ning asosiy vazifalarini ayting?
5. DML, DDL va DCL ning farqlarini asoslang?
6. Select operatorining sintaksisi qanday?

9 BOB. MA'LUMOTLAR MANIPULYATSIYA QILISHDA ODDIY SO'ROVLAR YARATISH

Tayanch so'zlar: *insert into, dml, update, guruhli funksiyalar, manipulyatsiya, view, tranzatsiya*

9.1. Ma'lumotlar manipulyatsiya qilishda oddiy so'rovlar yaratish

Ma'lumotlar manipulyatsiyasini amalga oshirish uchun SQL (Structured Query Language) tili DML (Data Manipulation Language) yordamida ma'lumotlarni qo'shish, yangilash, o'chirish va so'rash kabi operatsiyalarni bajaradi. DML so'rovlarining asosiy vazifasi – ma'lumotlar bazasidagi ma'lumotlarni o'zgartirish, qo'shish yoki olishdir. Quyida DML buyruqlari yordamida oddiy so'rovlarni yaratishga oid ma'lumotlar keltirilgan.

1. INSERT INTO

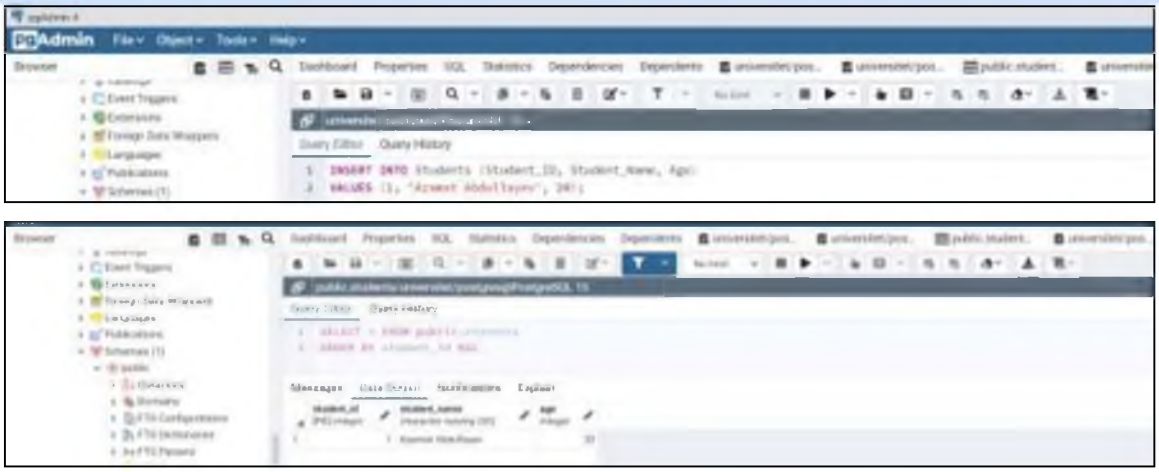
INSERT INTO buyrug'i jadvalga yangi yozuvlarni qo'shish uchun ishlatiladi.

Sintaksis:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

Misol: Talabalar jadvaliga yangi talabani qo'shish:

```
INSERT INTO Students (Student_ID, Student_Name, Age)
VALUES (1, 'Azamat Abdullayev', 20);
```



9.1-rasm. Insert into buyrug'i va natijasi

Bu so‘rov “Students” jadvaliga yangi yozuv qo‘shadi, bunda student_id = 1, student_name = ‘Azamat Abdullayev’ va age = 20.

2. SELECT (Tanlash)

SELECT buyrug‘i ma’lumotlarni bazadan olish uchun ishlatiladi. Bu buyruq yordamida ma’lum shartlar asosida ma’lumotlarni tanlash mumkin.

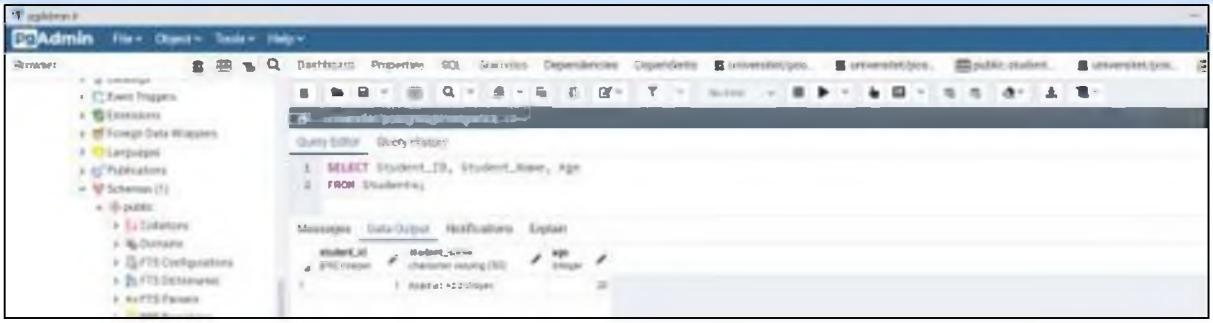
Sintaksis:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Misol:

Talabalar jadvalidan barcha talabalar haqida ma’lumot olish:

```
SELECT Student_ID, Student_Name, Age
FROM Students;
```



9.2-rasm. Select buyrug‘i ishlashi

3. UPDATE (Yangilash)

UPDATE buyrug‘i ma’lumotlarni yangilash uchun ishlatiladi. Bu buyruq yordamida mavjud yozuvlarni o‘zgartirish mumkin.

Sintaksis:

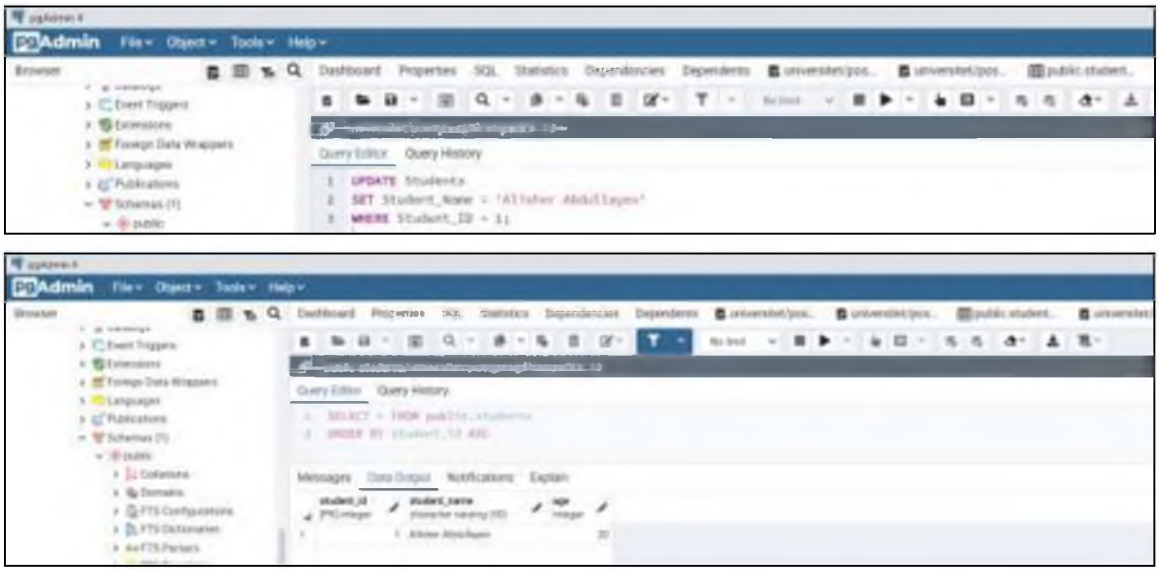
```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Misol:

Talaba ismini yangilash:

```
UPDATE Students
SET Student_Name = 'Alisher Abdullayev'
WHERE Student_ID = 1;
```

Bu so‘rov Student_ID = 1 bo‘lgan talabani nomini yangilaydi.



9.3-rasm. Update buyrug‘I va natijasi

*4. **DELETE (O‘chirish)***

DELETE buyrug‘i ma’lumotlarni o‘chirish uchun ishlatiladi. Bu buyruq yordamida mavjud yozuvlar o‘chiriladi.

Sintaksis:

```
DELETE FROM table_name
WHERE condition;
```

Misol:

Biror talabani jadvaldan o‘chirish:

```
DELETE FROM Students
WHERE Student_ID = 1;
```

Bu so‘rov Student_ID = 1 bo‘lgan talabaning yozuvini o‘chiradi.

Ma’lumotlar manipulyatsiya qilishda quyidagi xavfsizlik masalalarini e’tiborga olish kerak:

WHERE shartini ishlatish: UPDATE va DELETE buyruqlarida har doim **WHERE** shartini qo‘llash zarur, aks holda barcha yozuvlar o‘zgartiriladi yoki o‘chiriladi.

BACKUP: Ma’lumotlar manipulyatsiya qilinayotganda, bazaning zaxira nusxasini olish tavsiya etiladi.

LIMIT: Ba’zi tizimlarda **LIMIT** buyrug‘ini ishlatish orqali faqat ma’lum sonli yozuvlar ustida ish olib borish mumkin.

9.2. Murakkab so‘rovlar yaratish

Murakkab so‘rovlar SQLda bir nechta jadval va ma’lumotlar bilan ishlashni talab qiladi. Bunday so‘rovlar ko‘pincha JOIN, GROUP BY, HAVING kabi operatorlar yordamida amalga oshiriladi.

JOIN Operatori

JOIN operatori bir nechta jadvallarni birlashtirish uchun ishlatiladi. Murakkab so‘rovlar yaratishda ko‘pincha INNER JOIN, LEFT JOIN, RIGHT JOIN yoki FULL JOIN ishlatiladi.

Misol:

1. Mahsulot_Turi jadvali

Bu jadvalda mahsulotlarning turlari haqida ma’lumotlar saqlanadi.

9.1-jadval. Mahsulot turi jadvali

Mahsulot_Turi_ID	Mahsulot_Turi_Nomi
1	Taom
2	Ichimlik
3	Desert

Mahsulot_Turi_ID: Mahsulot turi identifikatori (asosiy kalit).

Mahsulot_Turi_Nomi: Mahsulot turining nomi.

2. Mahsulotlar jadvali

Bu jadvalda turli mahsulotlar va ularning tavsiflari saqlanadi.

9.2-jadval. Mahsulotlar jadvali

Mahsulot_ID	Mahsulot_Turi_ID	Mahsulot_Nomi	Narx
1	1	Pitsa	15.00
2	1	Burger	10.00
3	2	Pepsi	3.00
4	2	Fanta	2.50
5	3	Shokolad	5.00
6	3	Ice Cream	4.00

Mahsulot_ID: Mahsulot identifikatori (asosiy kalit).

Mahsulot_Turi_ID: Mahsulot turi identifikatori (foreign key, **Mahsulot_Turi** jadvalidan).

Mahsulot_Nomi: Mahsulot nomi.

Narx: Mahsulotning narxi.

INNER JOIN ga asoslangan SQL so‘rovi:

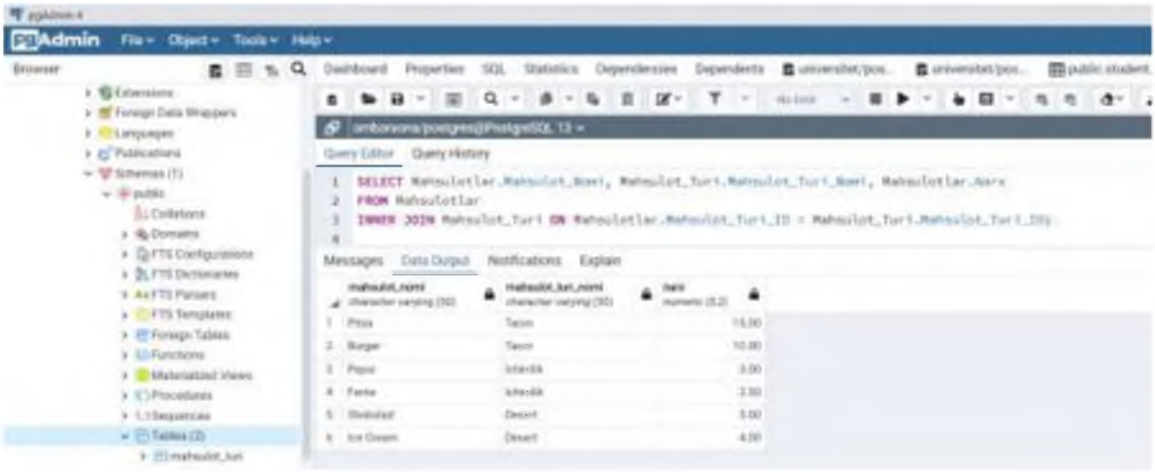
Biz **Mahsulotlar** va **Mahsulot_Turi** jadvallarini **Mahsulot_Turi_ID** bo‘yicha birlashtirib, har bir mahsulotning turi va narxini ko‘rsatuvchi so‘rovni yozamiz.

```
SELECT Mahsulotlar.Mahsulot_Nomi, Mahsulot_Turi.Mahsulot_Turi_Nomi, Mahsulotlar.Narx
FROM Mahsulotlar
INNER JOIN Mahsulot_Turi ON Mahsulotlar.Mahsulot_Turi_ID = Mahsulot_Turi.Mahsulot_Turi_ID;
```

So‘rovning ishlashi:

Mahsulotlar jadvali va Mahsulot_Turi jadvalini Mahsulot_Turi_ID bo‘yicha birlashtiradi. Natijada har bir mahsulotning nomi, uning turi va narxi ko‘rsatiladi.

Natija:



9.4-rasm. INNER JOIN ga asoslangan SQL so‘rov natijasi

Bu yerda har bir mahsulotning nomi, turi va narxi ko‘rsatilgan. Bu so‘rov orqali mahsulotlar va ularning turlari o‘rtasidagi bog‘lanishlar ko‘rsatiladi.

GROUP BY operatori

GROUP BY operatori, ma’lumotlarni guruhlash uchun ishlatiladi. Bu operator bir nechta qatorlarni birlashtiradi va har bir guruhga umumiy hisob-kitoblar (masalan, agregat funksiyalar: COUNT, SUM, AVG, MAX, MIN) bajarishga imkon beradi. **GROUP BY** operatori ko‘pincha agregat funksiyalar bilan birga ishlatiladi.

Misol: Biz **Mahsulotlar** jadvalidagi har bir **Mahsulot_Turi** bo‘yicha mahsulotlarning narxining o‘rtacha qiymatini hisoblamoqchimiz.

SQL so‘rovi:

```
SELECT Mahsulot_Turi.Mahsulot_Turi_Nomi, AVG(Mahsulotlar.Narx) AS Urtacha_Narx
FROM Mahsulotlar
INNER JOIN Mahsulot_Turi ON Mahsulotlar.Mahsulot_Turi_ID = Mahsulot_Turi.Mahsulot_Turi_ID
GROUP BY Mahsulot_Turi.Mahsulot_Turi_Nomi;
```

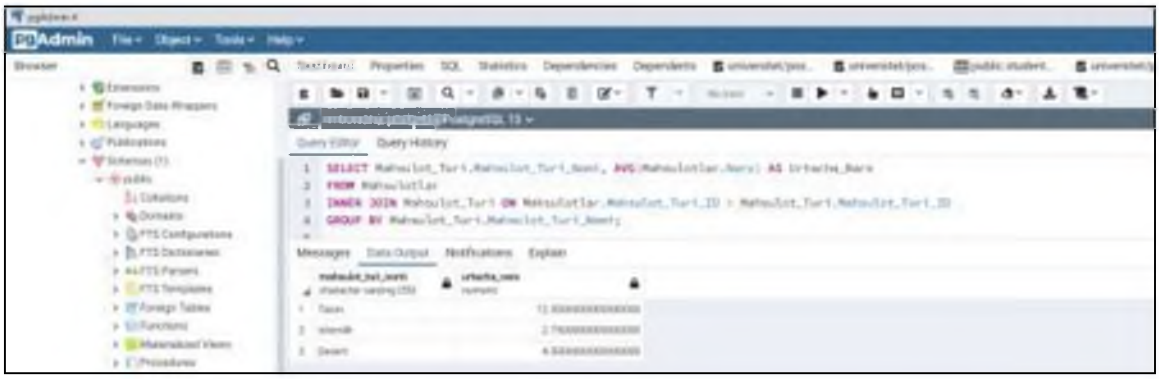
So‘rovning ishlashi:

INNER JOIN yordamida **Mahsulot_Turi** va **Mahsulotlar** jadvallari **Mahsulot_Turi_ID** bo‘yicha birlashtiriladi.

GROUP BY Mahsulot_Turi.Mahsulot_Turi_Nomi qismi har bir mahsulot turiga ko‘ra guruhlashni amalga oshiradi.

AVG(Mahsulotlar.Narx) yordamida har bir guruh uchun o‘rtacha narx hisoblanadi.

Natija:



9.5-rasm. Group byga asoslangan SQL so‘rov natijasi

Bu yerda har bir **Mahsulot_Turi** bo‘yicha o‘rtacha narx ko‘rsatilgan.

HAVING operatori

HAVING operatori, **GROUP BY** bilan guruhlangan natijalarni filtrlash uchun ishlatiladi. Agar siz guruhlangan natijalar ustida shartlarni qo‘llamoqchi bo‘lsangiz, unda **HAVING** operatoridan foydalanasiz. **WHERE** operatori esa faqat individual satrlarni filtrlashda ishlatiladi, lekin **HAVING** guruhlangan ma’lumotlarga nisbatan ishlaydi.

Misol: Endi, biz faqat o‘rtacha narxi 5 dan yuqori bo‘lgan mahsulot turlarini ko‘rsatmoqchimiz.

SQL so‘rovi:

```
SELECT Mahsulot_Turi.Mahsulot_Turi_Nomi, AVG(Mahsulotlar.Narx) AS Urtacha_Narx
FROM Mahsulotlar
INNER JOIN Mahsulot_Turi ON Mahsulotlar.Mahsulot_Turi_ID = Mahsulot_Turi.Mahsulot_Turi_ID
```

```
GROUP BY Mahsulot_Turi.Mahsulot_Turi_Nomi
HAVING AVG(Mahsulotlar.Narx) > 5;
GROUP BY Mahsulot_Turi.Mahsulot_Turi_Nomi;
```

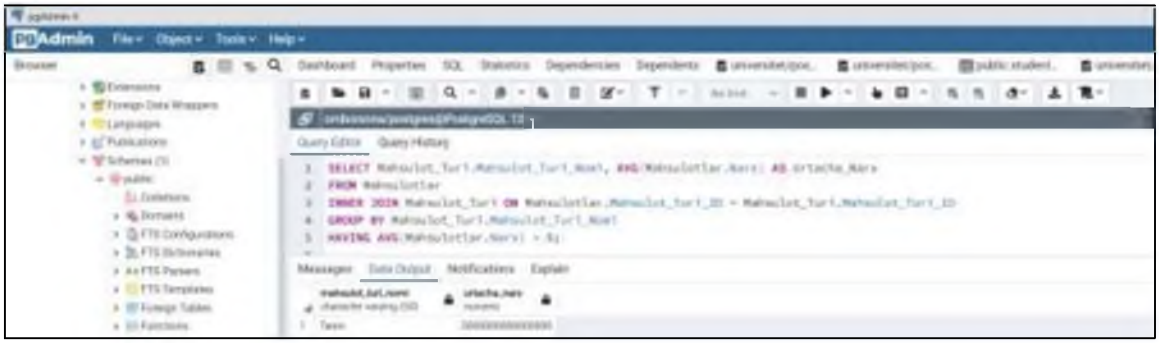
So‘rovning ishlashi:

GROUP BY Mahsulot_Turi.Mahsulot_Turi_Nomi qismi har bir mahsulot turiga ko‘ra guruhlashni amalga oshiradi.

AVG(Mahsulotlar.Narx) yordamida har bir guruh uchun o‘rtacha narx hisoblanadi.

HAVING AVG(Mahsulotlar.Narx) > 5 qismi esa faqat o‘rtacha narxi 5 dan yuqori bo‘lgan guruhlarni filtrlashni amalga oshiradi.

Natija:



9.6-rasm. HAVING operatori natijasi

Bu yerda faqat **Taom** turining o‘rtacha narxi 5 dan yuqori bo‘lgani uchun ko‘rsatilgan.

LIKE operatori. SQLda **LIKE** operatori matnli qiymatlarni qidirish va shunga o‘xshash qiymatlarni topish uchun ishlatiladi. Bu operator yordamida belgilangan shablonlarga mos keladigan satrlarni tanlash mumkin. **LIKE** operatori ko‘pincha **WHERE** bilan birga ishlatiladi.

LIKE operatorining umumiy sintaksisi:

```
SELECT column_name
FROM table_name
WHERE column_name LIKE pattern;
```

Bu yerda:

column_name — izlanayotgan ustun.

pattern — matnni qidirish uchun belgilangan shablon. Shablon **%** va **_** belgilari bilan kengaytirilishi mumkin:

% — nol yoki bir nechta harfni o‘z ichiga oladi.

_ — faqat bitta harfni o‘z ichiga oladi.

Misollar:

LIKE operatoridan foydalanib, ismning boshlanishini qidirish:

Agar siz “John” bilan boshlanuvchi barcha xodimlarni topmoqchi bo‘lsangiz:

```
SELECT *
FROM Employees
WHERE First_Name LIKE ‘John%’;
```

LIKE operatori yordamida ismning ma’lum qismidan qidirish:

Agar siz “a” bilan tugaydigan xodimlarni qidirmoqchi bo‘lsangiz, quyidagicha yozasiz:

```
SELECT *
FROM Employees
WHERE First_Name LIKE ‘%a’;
```

Natija:

Bu so‘rov “First_Name” ustunida “a” bilan tugaydigan barcha xodimlarni ko‘rsatadi. Masalan, “Maria” va “Sara”.

LIKE va _ operatoridan foydalanib, faqat bitta harf bilan qidirish:

Agar siz xodimlarning ismida ikkinchi harf sifatida “o” bo‘lgan xodimlarni qidirmoqchi bo‘lsangiz, quyidagicha so‘rovni yozing:

```
SELECT *
FROM Employees
WHERE First_Name LIKE ‘_o%’;
```

Access dasturida Like operatori bilan ishlashda “%” belgisi o‘rniga “*” belgisi ishlatiladi

9.3. SQLda almashtirish funksiyalari bilan ishlash

SQLda **Almashtirish funksiyalari (String Manipulation Functions)**, matnli ma’lumotlar bilan ishlashda foydalaniladigan bir qator funksiyalarni o‘z ichiga oladi.

Bu funksiyalar yordamida matnlarni manipulyatsiya qilish, o‘zgartirish, tahlil qilish va qayta formatlash mumkin. Sqlda almashtirish funksiyalaridan quyidagilar keng qo‘llaniladi:

9.3-jadval. Almashtirish funksiyalari

Funksiya	Tavsif	Sintaksis	Misol	Natija
UPPER()	Matndagi barcha harflarni katta qilish	UPPER(string)	SELECT UPPER(‘hello world’);	HELLO WORLD
LOWER()	Matndagi barcha harflarni kichik qilish	LOWER(string)	SELECT LOWER(‘HELLO WORLD’);	hello world
INITCAP()	Har bir so‘zni boshlanishini katta qilish (ba’zi tizimlarda)	INITCAP(string)	SELECT INITCAP(‘hello world’);	Hello World
CONCAT()	Matnlarni birlashtirish	CONCAT(string1, string2)	SELECT CONCAT(‘Hello’, ‘World’);	Hello World
SUBSTRING() / SUBSTR()	Matndan qismini olish	SUBSTRING(string, start, length)	SELECT SUBSTRING(‘Hello World’, 1, 5);	Hello
TRIM()	Matnning boshidagi va oxiridagi bo‘sh joylarni olib tashlash	TRIM(string)	SELECT TRIM(‘ Hello World ‘);	Hello World

REPLACE()	Matndagi so‘z yoki belgilarni almashtirish	REPLACE(string, old_string, new_string)	SELECT REPLACE(‘Hello World’, ‘World’, ‘SQL’);	Hello SQL
LEFT()	Matnning chap tomonidan belgilangan sonli belgilarni olish	LEFT(string, length)	SELECT LEFT(‘Hello World’, 5);	Hello
RIGHT()	Matnning o‘ng tomonidan belgilangan sonli belgilarni olish	RIGHT(string, length)	SELECT RIGHT(‘Hello World’, 5);	World
LEN() / LENGTH()	Matn uzunligini hisoblash	LEN(string) yoki LENGTH(string)	SELECT LEN(‘Hello World’);	11
INSTR()	Matndagi belgilangan substring ning boshlanish joyini topish	INSTR(string, substring)	SELECT INSTR(‘Hello World’, ‘World’);	7

Microsoft Accessda SQL rejimida **LOWER()**, **UPPER()**, **INITCAP()** va ayrim funksiyalar mavjud emas. Biroq, Access SQLda o‘zining ba’zi almashtirish funksiyalari bor, ularni quyidagi tarzda ishlatish mumkin:

Bizda talaba jadvali bo‘lsin.

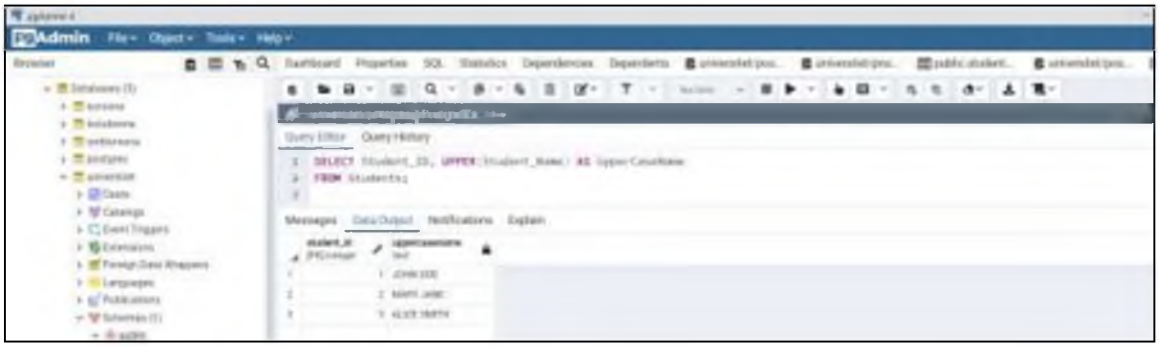
9.4-jadval. Talaba jadvali

Student_ID	Student_Name
1	john doe
2	mary jane
3	alice smith

1. UCASE() – Katta harflarga aylantirish (UPPER ekvivalenti): Bu so‘rov talabalar ismlarini katta harflarga aylantiradi.

```
SELECT Student_ID, UPPER(Student_Name) AS UpperCaseName
FROM Students;
```

Natija:

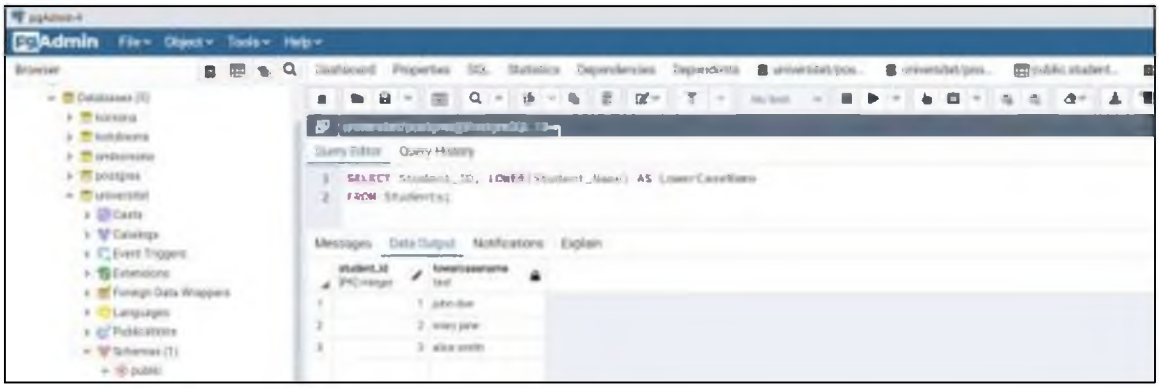


9.7-jadval. UPPER() funksiyasi natijasi

2. LCASE() – Kichik harflarga aylantirish (LOWER ekvivalenti): Bu so‘rov talabalar ismlarini kichik harflarga aylantiradi.

```
SELECT Student_ID, LOWER(Student_Name) AS LowerCaseName
FROM Students;
```

Natija:



9.8-jadval. LOWER() funksiyasi va natijasi

3. StrConv() – To‘liq formatlash (INITCAP ekvivalenti):

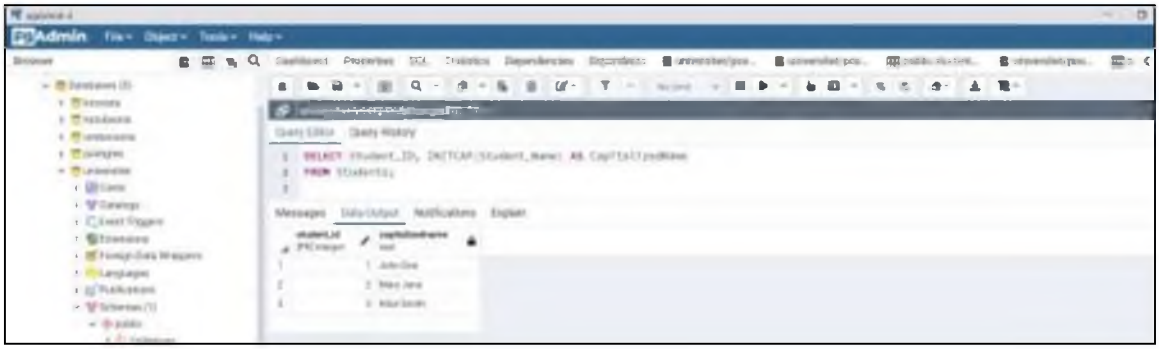
Microsoft Accessda StrConv() funksiyasining ikkinchi parametri uchun to‘g‘ri qiymatni ishlatish kerak. Access SQLda StrConv() yoki INITCAP() funksiyasi uchun matnni to‘g‘ri formatlash (ya’ni, har bir so‘zning birinchi harfini katta qilish) uchun ikkinchi parametr sifatida 3 ishlatiladi, lekin ba’zan bu Access versiyasiga bog‘liq holda ishlamasligi mumkin.

Microsoft Access SQLda har bir so‘zning birinchi harfini katta qilish uchun StrConv() yoki INITCAP() funksiyasidan foydalanamiz.

SELECT Student_ID, INITCAP(Student_Name) AS CapitalizedName

FROM Students;

Natija:



9.9-jadval. INITCAP() funksiyasi va natijasi

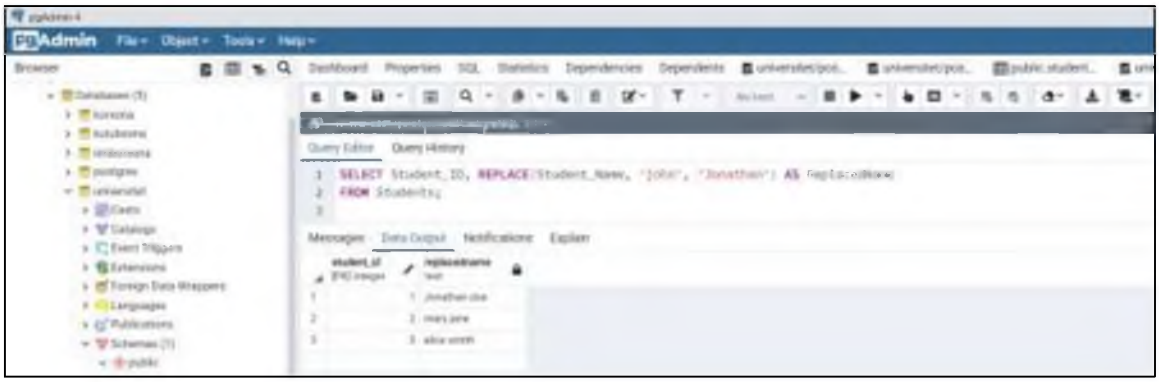
Bu SQL so‘rovi Access SQLda to‘g‘ri ishlashi kerak, chunki Access SQLda 3 parametri “Proper Case” (har bir so‘zning bosh harfini katta qilish) uchun ishlatiladi.

4. Replace() – Belgilarni almashtirish:

Bu so‘rovda “john” so‘zi “Jonathan” bilan almashtiriladi.

```
SELECT Student_ID, Replace(Student_Name, ‘john’, ‘Jonathan’) AS ReplacedName
FROM Students;
```

Natija:



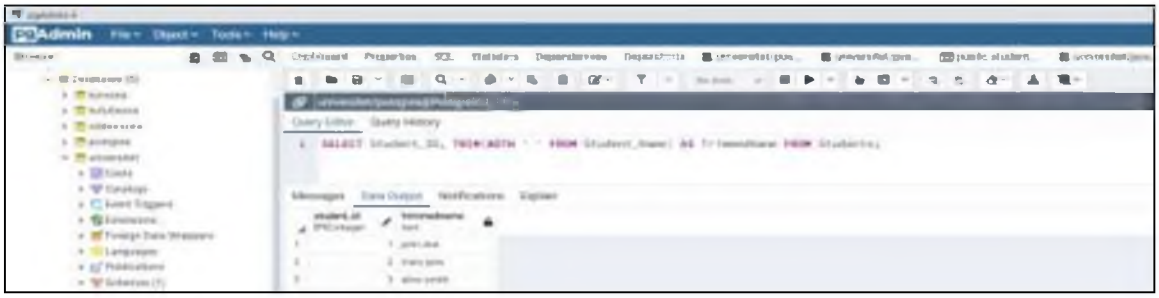
9.10-rasm. Replace() funksiyasi va natijasi

5. Trim() – Bo‘sh joylarni olib tashlash:

Bu so‘rov talabalar ismlaridan ortiqcha bo‘sh joylarni olib tashlaydi.

```
SELECT Student_ID, Trim(Student_Name) AS TrimmedName
FROM Students;
```

Natija:



9.11-rasm. Trim() funksiyasi va natijasi

Microsoft Access SQL-da yuqoridagi funksiyalarni ishlatish orqali matnlarni o‘zgartirish mumkin. Access SQL-da UCASE() va LCASE() funksiyalari yordamida katta va kichik harflar bilan ishlash, StrConv() yordamida INITCAPni simulyatsiya qilish, Replace() yordamida matnni almashtirish va Trim() yordamida bo‘sh joylarni olib tashlash mumkin. Bu funksiyalar talabalar jadvalida amalda ishlatildi.

9.4. Guruhli funksiyalarni so‘rovlarda ishlatish

Guruhli funksiyalar SQLda ma’lumotlar to‘plamini (yoki guruhlarini) qayta ishlash va tahlil qilish uchun ishlatiladi. Bu funksiyalar yordamida ma’lumotlarni guruhlarga ajratish va shu guruhlar ustida hisoblash ishlarini amalga oshirish mumkin. SQLda guruhli funksiyalar DQL (Data Query Language) so‘rovlarida ishlatiladi va ularning asosiy vazifasi bir nechta yozuvlar asosida yagona natijalar olishdir.

Guruhli funksiyalar quyidagilardir:

1. **COUNT()** — guruhdagi yozuvlar sonini hisoblaydi.
2. **SUM()** — guruhdagi raqamli qiymatlarning yig‘indisini hisoblaydi.
3. **AVG()** — guruhdagi raqamli qiymatlarning o‘rtacha qiymatini hisoblaydi.
4. **MAX()** — guruhdagi maksimal qiymatni qaytaradi.
5. **MIN()** — guruhdagi minimal qiymatni qaytaradi.

9.5. SQLda tasavvurlar va jadvallar bilan ishlash

SQLda tasavvur (view) — bu real jadval emas, balki ma’lum bir so‘rov natijasida paydo bo‘lgan virtual jadvaldir. Tasavvur yordamida ma’lumotlar bazasidagi murakkab

so‘rovlarni saqlash, ko‘rinishlarni soddalashtirish va foydalanuvchilarga qulayroq interfeys yaratish mumkin.

Tasavvurlar (Views) nima?

Tasavvur — bu foydalanuvchilarga faqat kerakli ma’lumotlarni ko‘rsatadigan, ma’lumotlarni so‘rashni soddalashtiradigan virtual jadvaldir. Tasavvurlar ma’lumotlar bazasida saqlanmaydi, balki ular SQL so‘rovi yordamida dinamik tarzda yaratilib, vaqtinchalik natijalar sifatida ishlatiladi.

Tasavvurlarni yaratish quyidagi holatlarda foydalidir:

Murakkab so‘rovlarni soddalashtirish: Murakkab va uzun so‘rovlarni takroriy ishlatishda ularni tasavvur sifatida saqlash mumkin.

Xavfsizlik: Ba’zi ma’lumotlarni to‘liq ko‘rinishini cheklash uchun tasavvurlarni ishlatish mumkin. Masalan, faqat muayyan ustunlar yoki yozuvlar ko‘rsatiladi.

Yangi ma’lumotlarni saqlash: Ba’zi tizimlar tasavvurlarni ma’lumotlarni yozish yoki yangilash uchun ham ishlatishi mumkin.

Tasavvurlarni yaratish sintaksisi:

Tasavvur yaratish uchun **CREATE VIEW** operatori ishlatiladi.

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Bizda quyidagi jadvallar bo‘lsin:

9.5-jadval. Students jadvali

Student_ID	Student_Name	Age
1	John Doe	20
2	Mary Jane	22
3	Alice Smith	21

9.6-jadval Courses jadvali

Course_ID	Course_Name
1	Mathematics
2	Physics
3	Computer Science

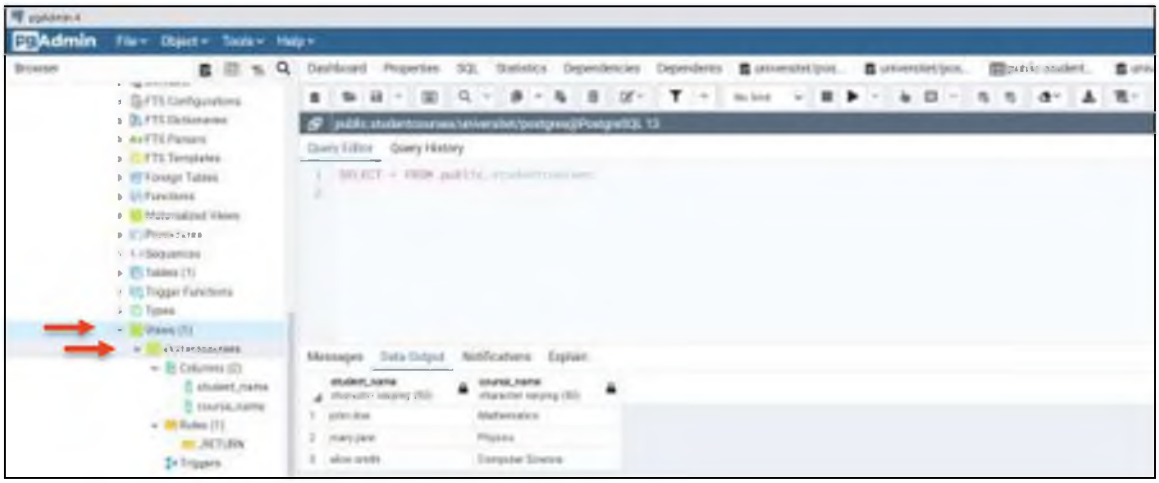
9.7-jadval. Enrollments jadvali(ro‘yxatdan o‘tish jadvali)

Enrollment_ID	Student_ID	Course_ID	Enrollment_Date
1	1	1	2024-01-15
2	2	2	2024-01-17
3	3	3	2024-01-20
4	1	3	2024-02-01

Misol: Agar bizda Students va Courses jadvallari bo‘lsa va talabalar bilan kurslar orasidagi munosabatni ko‘rsatadigan tasavvur yaratmoqchi bo‘lsak, quyidagicha so‘rov yozishimiz mumkin:

```
CREATE VIEW StudentCourses AS
SELECT Students.Student_Name, Courses.Course_Name
FROM Students
JOIN Enrollments ON Students.Student_ID = Enrollments.Student_ID
JOIN Courses ON Enrollments.Course_ID = Courses.Course_ID;
```

Ushbu tasavvur **StudentCourses** nomi bilan yaratiladi va unda har bir talabani tegishli kursi bilan bog‘laydigan ma’lumotlar saqlanadi.



9.12-jadval. StudentCourses tasavvuri natijasi

Tasavvurlardan foydalanish:

Tasavvurlarni yaratganimizdan so‘ng, ularni oddiy jadvalga o‘xshab ishlatish mumkin.

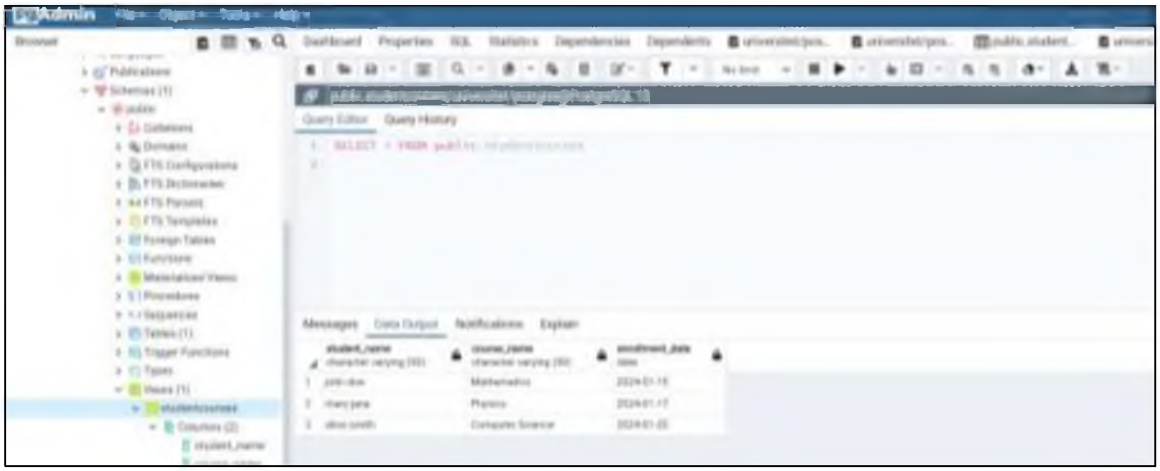
```
SELECT * FROM StudentCourses;
```

Bu so‘rov **StudentCourses** tasavvuridan barcha ma’lumotlarni qaytaradi. Bu juda qulay, chunki biz har safar kompleks so‘rovni qayta yozishimizning o‘rniga tasavvurni ishlatishimiz mumkin.

Tasavvurlarni yangilash va o‘chirish:

Tasavvurni yangilash: Tasavvurlarni yaratgandan so‘ng, ularni yangilash mumkin. SQL da tasavvurni yangilash uchun **CREATE OR REPLACE VIEW** ishlatiladi.

```
CREATE OR REPLACE VIEW StudentCourses AS
SELECT Students.Student_Name, Courses.Course_Name, Enrollments.Enrollment_Date
FROM Students
JOIN Enrollments ON Students.Student_ID = Enrollments.Student_ID
JOIN Courses ON Enrollments.Course_ID = Courses.Course_ID;
```



9.13-jadval. StudentCourses tasavvuri (yangilangan)

Tasavvurni o‘chirish: Agar tasavvurning endi zarurati bo‘lmasa, u o‘chirilishi mumkin.

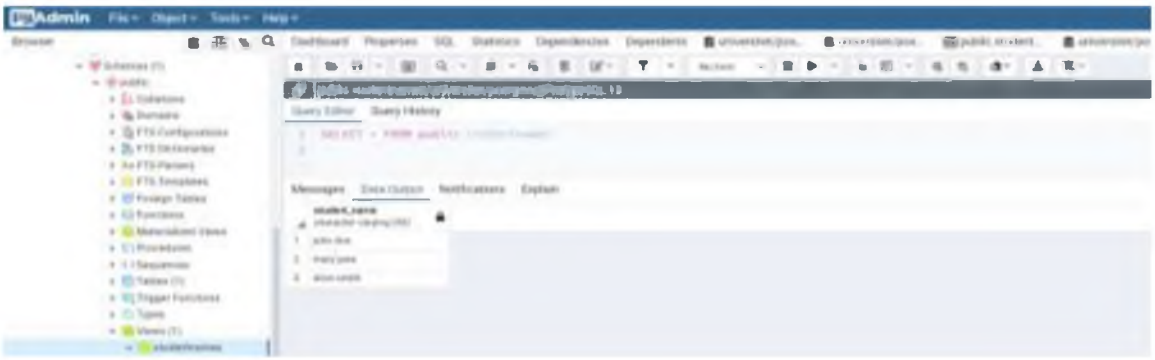
```
DROP VIEW StudentCourses;
```

Tasavvurlarni cheklovlar va xavfsizlik:

Tasavvurlar xavfsizlikni ta’minlashda yordam beradi. Masalan, agar ma’lum bir foydalanuvchi faqat talabalar ismlarini ko‘rishi kerak bo‘lsa, faqat shu ustunlar mavjud bo‘lgan tasavvur yaratilishi mumkin:

```
CREATE VIEW StudentNames AS
SELECT Student_Name
FROM Students;
```

Bu tasavvur faqat **Student_Name** ustunini ko‘rsatadi va boshqa barcha ma’lumotlar (masalan, talabalar raqami, yoshi) yashiriladi.



9.14-jadval. Student_Name ustuni uchun yaratilgan tasavvur

9.6. Ma’lumotlarni ajratish va tiklash

Ma’lumotlarni ajratish jarayoni odatda, normalizatsiya bilan bog‘liq bo‘lib, katta va murakkab munosabatni mantiqan bog‘liq kichik munosabatlarga ajratishni anglatadi. Bu jarayon orqali:

- ✓ Ma’lumotlarning ortiqchaligi kamayadi.
- ✓ Ma’lumotlar ziddiyatining oldi olinadi.
- ✓ Ma’lumotlarni yangilash va saqlash samaradorligi oshadi.

Ajratish tamoyillari

Yo‘qotishlarsiz ajratish (Lossless Decomposition): Ajratilgan jadvallarni birlashtirganda dastlabki ma’lumotlar yo‘qolmasligi kerak.

Bog‘liqlikni saqlash (Dependency Preservation): Ajratish jarayonida munosabatlardagi barcha funktsional bog‘liqliklar saqlanib qolishi lozim.

Ma’lumotlarni tiklash (Recomposition) jarayoni kichik munosabatlarni dastlabki katta munosabatga qaytarishdir. Bu jarayon yo‘qotishlarsiz tiklash tamoyiliga asoslanadi, ya’ni birlashtirilgan natija asl ma’lumotlarga mos kelishi kerak.

Tiklash usullari

Natural join: Kichik jadvallarni qo‘shish uchun asosiy operatorlardan biri. Ushbu operator ustunlar bo‘yicha mos keluvchi qiymatlarni birlashtiradi.

Ma’lumotlarni saqlashdagi samaradorlik: Kichik jadvallar ma’lumotlar ziddiyatini kamaytiradi va ma’lumotlarni yangilashni osonlashtiradi.

Moslashuvchanlik: Ajratilgan jadvallarni kerak bo‘lganda qayta birlashtirish orqali tahliliy jarayonlarni osonlashtirish mumkin.

Ajratish va tiklash ma’lumotlar bazalarini boshqarishda muhim tamoyillardir. Ular orqali ma’lumotlarning ortiqchaligini kamaytirish, ma’lumotlarni boshqarish qulayligini oshirish va bazaning ishonchliligini ta’minlash mumkin. Bu jarayonlarni tushunish uchun normalizatsiya bosqichlari va JOIN operatorlari bilan amaliy ishlashni o‘rganish zarur.

Nazorat savollari:

1. SQLda almashtirish funksiyalari nima maqsadda ishlatiladi?
2. Guruhli funksiyalarga qanday 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 qanday yaratiladi?

10 BOB. SQLTILI YORDAMIDA MA'LUMOTLARNI TAVSIFLASH

Tayanch so'zlar: *butunlik, ALTER, create table, create index, protsedura, funksiya, view, trigger, drop*

10.1. SQL tilida ma'lumotlarni butunligini ta'minlash

Ma'lumotlar bazasida butunlik (inglizcha: *integrity*) deganda, undagi ma'lumotlarning to'g'ri va ishonchli bo'lishi, ya'ni ma'lumotlarning yaxlitligi, mantiqan to'g'ri saqlanishi tushuniladi. Bu, ma'lumotlarning har qanday o'zgarishlarga qarshi himoya qilinishini va ular noto'g'ri yoki mantiqan noto'g'ri bo'lib qolmasligini anglatadi.

Ma'lumotlar butunligini ta'minlash ma'lumotlar bazasining ishonchliligini saqlashda juda muhim va u quyidagi asosiy usullar bilan ta'minlanadi:

1. **Mohiyat bo'yicha butunlik** (Entity Integrity): Bu butunlik asosiy kalitlar (birlamchi kalitlar) bilan bog'liq bo'lib, asosiy kalit qiymatlari takrorlanmas va NULL bo'lmagan qiymatlarga ega bo'lishini ta'minlaydi. Shu bilan har bir yozuv boshqa yozuvlardan farqlanadi.
2. **Murojaat bo'yicha butunlik** (Referential Integrity): Bu butunlik talabiga ko'ra, tashqi kalitlar boshqa jadvallarda mavjud bo'lgan asosiy kalit qiymatlariga muvofiq bo'lishi kerak. Bu qoidalar jadvallar o'rtasidagi bog'lanishlarni mustahkamlaydi va noto'g'ri murojaatlar paydo bo'lishidan saqlaydi.
3. **Foydalanuvchi aniqlaydigan butunlik** (User-defined Integrity): Bu, ma'lumotlar bazasi uchun qo'shimcha cheklovlar va qoidalardan iborat bo'lib, ularni foydalanuvchilar yoki ma'lumotlar bazasini boshqarish tizimi ishlab chiquvchilari belgilaydi. Masalan, ma'lum qiymatlar diapazoni yoki atributlarning alohida qiymatlari kabi qo'shimcha qoidalar.

Bu butunlik qoidalari ma'lumotlarning noto'g'ri kiritilishi, noto'g'ri saqlanishi yoki ma'lumotlar bazasida mantiqan noto'g'ri holatlarning paydo bo'lishidan himoya qiladi.

Butunlikni aniqlash usullari:

1. Birinchi kalitda qatnashuvchi atributlarga aniqlanmagan qiymatlar qabul qilinishiga ruxsat etilmaydi.

2. Tashqi kalit qiymati quyidagilar biri bo‘lishi kerak:

- ✓ Birinchi kalit qiymatiga teng;
- ✓ To‘liq aniqlanmagan, ya’ni tashqi kalitda qatnashadigan har bir atribut qiymati aniqlanmagan bo‘lishi kerak.

3. Har qanday aniq bir ma’lumotlar bazasi uchun qo‘shimcha qoidalar spetsifikatsiyalari mavjud. Ular ishlab chiquvchilar yordamida aniqlanadi. Ko‘p hollarda tekshiriladi:

- ✓ U yoki bu atributning unikalligi;
- ✓ Qiymatlar diapazoni;
- ✓ Qiymatlar to‘plamining aloqadorligi.

10.2. Ma’lumot jadvallarini yaratish

Ma’lumotlar bazasi obyektlarini yaratish, ya’ni turli ma’lumot bazasi komponentlarini shakllantirish, SQL tilining **Data Definition Language (DDL)** orqali amalga oshiriladi. DDL yordamida jadvallar, indekslar, ko‘rinishlar (VIEWS), triggerlar va boshqa obyektlar yaratiladi. Ushbu mavzu ma’lumotlar bazasi tuzilmalarini samarali boshqarish va tashkil qilishda asosiy qadam hisoblanadi.

Jadvallarni yaratish, o‘chirish, o‘zgartirish

Jadvallar — ma’lumot bazasining asosiy obyektlari bo‘lib, ma’lumotlarni qator va ustun shaklida saqlaydi. SQLda jadval yaratish uchun **CREATE TABLE** operatoridan foydalaniladi. **Masalan:** “Talabalar” jadvalini yaratamiz. Bu jadvalda talabaning ID raqami, ismi va tug‘ilgan sanasi kabi ma’lumotlar saqlanadi.

```
CREATE TABLE Students (
Student_ID AUTOINCREMENT PRIMARY KEY,
Student_Name TEXT(50),
```

```
  Birthdate DATE
);
```

Bu SQL kodi Students nomli jadval yaratadi. Jadvalda:

Student_ID – Talabaning avtomatik raqamlanadigan birlamchi kalit ustuni.

Student_Name – Talabaning ismi (50 belgidan iborat matn).

Birthdate – Talabaning tug‘ilgan sanasi (sana formati).

Demak, bu yerda CREATE TABLE SQL buyruği yordamida yangi jadval yaratiladi va u o‘z ichiga birlamchi kalit, xossa nomlari va ularning turlarini oladi.

ALTER Operator

ALTER operatori mavjud ma’lumotlar bazasi ob’ektini ya’ni jadvalni o‘zgartirish uchun ishlatiladi. Bu operator yordamida jadvalga yangi ustunlar qo‘shish, mavjud ustunlarni o‘zgartirish yoki ularning turini yangilash mumkin.

```
ALTER TABLE table_name
ADD column_name datatype;
```

Yoki mavjud ustunni o‘zgartirish:

```
ALTER TABLE table_name
MODIFY COLUMN column_name datatype;
```

Yoki ustunni o‘chirish:

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

Misol:

Yangi ustun qo‘shish

```
ALTER TABLE Students
ADD DateOfBirth DATE;
```

Ustunni o‘zgartirish (misol uchun, o‘lchamni kengaytirish)

MODIFY COLUMN buyrug‘i, SQLda ustunlarning ma’lumot turi, o‘lchami yoki boshqa xususiyatlarini o‘zgartirish uchun ishlatiladi. Bu buyruq ko‘proq MySQL va boshqa ba’zi SQL tizimlarida mavjud.

```
ALTER TABLE Students
MODIFY COLUMN Student_Name VARCHAR(100);
```

Ustunni o‘chirish

```
ALTER TABLE Students
DROP COLUMN Address;
```

DROP Operator

DROP operatori, mavjud ma’lumotlar bazasidagi ob’ektlarni, jumladan jadvallarni, indekslarni va boshqa obyektlarni o‘chirish uchun ishlatiladi. Bu operatorni ishlatish orqali ob’ektning butun tarkibi bazadan butunlay o‘chiriladi va qayta tiklash mumkin emas.

```
DROP TABLE table_name;
```

Yoki indeksni o‘chirish:

```
DROP INDEX index_name;
```

Misol:

Jadvalni o‘chirish

```
DROP TABLE Students;
```

10.3. Qiyom so‘rovlari bilan ishlash

Qiyom so‘rovlari (yoki **aggregat so‘rovlar**) ma’lumotlar bazasida ustunlar qiymatlari ustida umumlashtiruvchi hisob-kitoblar (masalan, SUM, AVG, COUNT, MAX, MIN)ni bajarishga imkon beruvchi so‘rovlar hisoblanadi. Ular, asosan, ma’lumotlarni tahlil qilish va statistik ma’lumotlarni olishda qo‘llaniladi.

Bir nechta qiymatni bitta qiymatga keltiradi: Masalan, bir ustundagi barcha raqamlarning yig‘indisini olish.

Foydali statistik ma'lumot beradi: Masalan, o'rtacha qiymat, eng yuqori yoki eng past qiymatni hisoblash.

Asosiy qiyom funksiyalari

SUM() – Ustundagi qiymatlarning yig'indisini hisoblaydi.

AVG() – Ustundagi qiymatlarning o'rtacha qiymatini hisoblaydi.

COUNT() – Qatorlar sonini hisoblaydi.

MAX() – Ustundagi eng katta qiymatni topadi.

MIN() – Ustundagi eng kichik qiymatni topadi.

10.4. Ma'lumot bazasi obyektlarini yaratish

Indekslarni yaratish, o'chirish

Indekslar jadvallardan ma'lumotlarni tez va samarali qidirish imkonini beradi. **CREATE INDEX** SQL operatori yordamida indeks yaratish mumkin. Masalan:

Talabalar jadvalidagi Student_Name ustuniga indeks yaratish uchun quyidagi SQL kodni ishlatamiz:

```
CREATE INDEX idx_student_name ON Students (Student_Name);
```

Bu kod Students jadvalidagi Student_Name ustuniga idx_student_name nomli indeksni yaratadi. Bu indeks bilan Student_Name ustunidagi ma'lumotlar qidirilganda tezlik oshadi.

Bir nechta ustunlarga indeks yaratish

Agar bir nechta ustunlar asosida indeks yaratish kerak bo'lsa, masalan, Student_Name va Birthdate ustunlarida, Access SQLda quyidagicha yozish mumkin:

```
CREATE INDEX idx_student_name_birthdate ON Students (Student_Name, Birthdate);
```

Bu Student_Name va Birthdate ustunlari kombinatsiyasiga asoslangan idx_student_name_birthdate nomli indeksni yaratadi. Bu indeks ko'p holatlarda qidiruv va tartiblash tezligini oshirish uchun ishlatiladi.

Indeksni o'chirish

Indeks kerak bo‘lmasa, uni o‘chirish ham mumkin. Quyidagi kod idx_student_name indeksini o‘chiradi:

```
DROP INDEX idx_student_name ON Students;
```

Indekslar qidiruvlar va so‘rovlarni tezlashtirish uchun foydali, lekin ortiqcha indekslar jadvalga kiritilayotgan va yangilanayotgan ma’lumotlarni sekinlashtirishi mumkin. Shu sababli, indekslarni faqat kerak bo‘lganda yaratish tavsiya etiladi.

Ko‘rinishlar (Views) yaratish, o‘chirish

Ko‘rinishlar yoki VIEWS — bu mavjud jadvallardagi ma’lumotlarning mantiqiy ko‘rinishini ta’minlaydi va ma’lumotlarga cheklangan kirish imkonini beradi. CREATE VIEW operatoridan foydalanib, kerakli maydonlarni tanlab olish mumkin:

```
CREATE VIEW Student_Info AS
SELECT Student_ID, Student_Name
FROM Students;
```

Bu so‘rov orqali faqat Student_ID va Student_Name ustunlaridan iborat ko‘rinish yaratiladi.

Triggerlarni yaratish

Ma’lumotlar bazasida **triggerlar** — bu ma’lum bir voqea yoki amal sodir bo‘lganda avtomatik ravishda ishga tushadigan **SQL buyruqlari** yoki **so‘rovlar**dir. Triggerlar ma’lumotlar bazasining ma’lum qoidalariga rioya qilinishini ta’minlash, ma’lumotlar bazasida ma’lum amallarni avtomatik ravishda bajarish uchun ishlatiladi.

Triggerlar ma’lumotlar bazasining **strukturali va ma’lumotlar yaxlitligini** saqlashga yordam beradi. Ular ma’lum bir **INSERT**, **UPDATE**, yoki **DELETE** amallari amalga oshirilganda avtomatik tarzda ishlashadi.

Funksiyalar va Protseduralarni yaratish

Ma’lumotlar bazasida funksiyalar va protseduralarni yaratish ma’lum bir vazifalarni avtomatlashtirish va murakkab ma’lumotlarni qayta ishlash uchun juda muhimdir.

Funksiyalar va Protseduralar Nima?

Funksiya — bu ma'lum bir natija qaytaradigan va parametrlarni qabul qiladigan kod blokidir. Funksiya biror hisoblash, tekshirish yoki boshqa amallarni bajaradi va natija sifatida biror qiymatni qaytaradi.

Protsedura — bu ma'lum bir vazifani bajaradigan kod blokidir, lekin protsedura qiymat qaytarmaydi. Protsedura asosan operatsiyalarni bajarishda ishlatiladi (masalan, ma'lumotlar qo'shish, o'chirish va h.k.).

Funksiyalar va Protseduralar Yaratish (Microsoft SQL Server)

SQL Serverda funksiyalar va protseduralarni yaratish uchun T-SQL (Transact-SQL) tilidan foydalaniladi

Funksiya yaratish (SQL Serverda)

SQL Serverda **user-defined function (UDF)** yaratish uchun CREATE FUNCTION ishlatiladi. Misol:

```
CREATE FUNCTION dbo.CalculateTotal(@price DECIMAL, @quantity INT)
RETURNS DECIMAL
AS
BEGIN
   RETURN @price * @quantity;
END;
```

Bu funksiya mahsulot narxi va miqdorini ko'paytirib, jami narxni qaytaradi. So'rovda quyidagicha ishlatilishi mumkin:

```
SELECT dbo.CalculateTotal(100, 5);
```

Protsedura yaratish (SQL Serverda)

SQL Serverda protsedura yaratish uchun CREATE PROCEDURE ishlatiladi. Misol:

```
CREATE PROCEDURE dbo.InsertProduct
   @ProductName NVARCHAR(100),
```

```
    @Price DECIMAL
AS
BEGIN
    INSERT INTO Products (ProductName, Price)
    VALUES (@ProductName, @Price);
END;
```

Bu protsedura yangi mahsulotni Products jadvaliga qo‘shadi. Uni chaqirish uchun quyidagi so‘rovni ishlatish mumkin:

```
EXEC dbo.InsertProduct ‘New Product’, 50.00;
```

Funksiyalar va Protseduralarni Yaratish (Microsoft Access)

Microsoft Accessda **VBA (Visual Basic for Applications)** yordamida funksiyalar va protseduralarni yaratish mumkin. Accessda funksiya va protseduralar yordamida foydalanuvchi ma’lumotlar bilan ishlashni avtomatlashtirish, murakkab hisob-kitoblarni amalga oshirish va ma’lumotlarni qayta ishlash imkoniyatlariga ega bo‘ladi.

Funksiya yaratish (Accessda VBA yordamida)

Accessda VBA yordamida funksiya yaratish uchun quyidagi amallarni bajarish kerak:

VBA Editorni oching (Alt + F11).

Insert > **Module** orqali yangi modul yarating.

Quyidagi funksiya kodini yozing:

```
Public Function CalculateTotal(price As Currency, quantity As Integer) As Currency
    CalculateTotal = price * quantity
End Function
```

Bu funksiya price va quantity parametrlarini qabul qiladi va ular ko‘paytirilgan jami narxni qaytaradi. So‘rov yoki forma elementlarida uni ishlatish mumkin:

```
Private Sub CalculateButton_Click()
    MsgBox "Jami narx: " & CalculateTotal(100, 5)
End Sub
```

Protsedura yaratish (Accessda VBA yordamida)

Accessda protsedura yaratish uchun quyidagi kodni ishlatishingiz mumkin:

```
Public Sub ShowMessage()
    MsgBox "Bu Accessdagi protsedura!"
End Sub
```

Bu protsedura ekranda xabar ko‘rsatadi. Formadagi biror hodisa, masalan, tugma bosilganda uni chaqirish mumkin:

```
Private Sub ShowMessageButton_Click()
    ShowMessage
End Sub
```

Funksiyalar va Protseduralarni Sinovdan O‘tkazish

SQL Serverda sinovdan o‘tkazish

SQL Serverda funksiya yoki protsedura yaratganingizdan so‘ng, uni **Query Analyzer** yoki **SQL Server Management Studio (SSMS)** yordamida sinab ko‘rishingiz mumkin. Misol:

Funksiya sinovi

```
SELECT dbo.CalculateTotal(100, 5);
```

Protsedura sinovi

```
EXEC dbo.InsertProduct ‘New Product’, 50.00;
```

Accessda sinovdan o‘tkazish

Accessda VBA yordamida yaratilgan funksiyalar va protseduralarni **Immediate Window** yoki forma elementlarida sinab ko‘rish mumkin. Misol:

```
Debug.Print CalculateTotal(100, 5)
```

Bu Immediate Windowda natijani ko‘rsatadi. Agar funksiya yoki protsedura formaning hodisalariga ulanadigan bo‘lsa, foydalanuvchi interfeysi orqali ham tekshirish mumkin.

Xatoliklarni Tuzatish

Funksiyalar va protseduralarni yaratishda xatoliklar paydo bo‘lishi mumkin. SQL Serverda xatoliklarni tuzatish uchun **TRY...CATCH** blokidan foydalanish mumkin:

```
BEGIN TRY
   -- Kod blokini yozing
END TRY
BEGIN CATCH
   -- Xatolikni qayta ishlash
   PRINT ERROR_MESSAGE()
END CATCH
```

10.5. Ma’lumotlarni aniqlash tili (DLL) operatorlari

Ma’lumotlarni aniqlash tili (Data Definition Language, **DDL**) — ma’lumotlar bazasi obyektlarini (jadval, indeks, ko‘rinish va boshqalar) yaratish, o‘zgartirish va o‘chirish uchun ishlatiladigan SQL buyrug‘idir. **DDL** operatorlari yordamida ma’lumotlar bazasining tuzilmasi bilan ishlanadi.

DDL operatorlari quyidagilardan iborat:

CREATE – Yangi obyektlar (jadval, indeks, ko‘rinish) yaratadi.

ALTER – Mavjud obyektlar tuzilmasini o‘zgartiradi.

DROP – Obyektlarni butunlay o‘chiradi.

TRUNCATE – Jadvaldagi barcha ma’lumotlarni o‘chiradi, lekin jadval tuzilmasini saqlab qoladi.

10.6. CREATE TABLE komandasi

CREATE operatori — bu **DDL (Data Definition Language)** buyrug‘i bo‘lib, ma’lumotlar bazasida yangi obyektlarni yaratish uchun ishlatiladi. Ushbu buyruq

yordamida jadvallar, indekslar, ko‘rinishlar (views), ma’lumotlar bazalari va boshqa obyektlar yaratiladi.

***CREATE** operatorining umumiy sintaksisi*

Jadval yaratish

Jadval yaratish uchun sintaksis:

```
CREATE TABLE jadval_nomi (
    ustun1 ma'lumot_turi [constraint],
    ustun2 ma'lumot_turi [constraint],
    ...
);
```

Parametrlar:

jadval_nomi: Yaratilayotgan jadvalning nomi.

ustun: Jadvaldagi ustun nomi.

Ma’lumot_turi: Ustun uchun ma'lumot turi (masalan, INT, VARCHAR, DATE).

constraint (ixtiyoriy): Ustun uchun cheklovlar (PRIMARY KEY, NOT NULL, UNIQUE va boshqalar).

Misol: Jadval yaratish:

```
CREATE TABLE Talabalar (
    ID INT PRIMARY KEY,
    Ism VARCHAR(50),
    Guruh VARCHAR(10)
);
```

Bu kod **Talabalar** nomli jadvalni yaratadi.

10.7. INSERT komandasi

INSERT operatori — SQL tilida **DML (Data Manipulation Language)** buyruqlaridan biri bo‘lib, ma’lumotlar bazasidagi jadvalga yangi ma’lumotlar qo‘shish

uchun ishlatiladi. Ushbu buyruq yordamida foydalanuvchi bir yoki bir nechta qatorni jadvalga kiritishi mumkin.

INSERT operatorining vazifalari

Yangi qatorlarni qo‘shish: Jadvaldagi mavjud tuzilmani buzmagan holda yangi ma’lumotlarni qo‘shadi.

Aniqlangan ustunlarga ma’lumot kiritish: Faqat kerakli ustunlarga qiymatlarni kiritish imkoniyatini beradi.

Bir nechta qatorlarni qo‘shish: Bitta buyruqda bir nechta qatorlarni jadvalga kiritishi mumkin.

SELECT bilan birgalikda foydalanish: Boshqa jadvaldan ma’lumotlarni tanlab, yangi jadvalga qo‘shish imkoniyatini beradi.

INSERT operatorining sintaksisi:

1. Oddiy INSERT operatori

```
INSERT INTO jadval_nomi (ustun1, ustun2, ...)
VALUES (qiymat1, qiymat2, ...);
```

Izoh:

jadval_nomi: Ma’lumotlarni qo‘shish kerak bo‘lgan jadvalning nomi.

ustun: Jadvalning ustunlari. Ustun nomlarini keltirish ixtiyoriy, lekin tavsiya etiladi.

VALUES: Yangi qatorning qiymatlari.

Misol:

Talabalar jadvali uchun yangi ma’lumot qo‘shish:

```
INSERT INTO Talabalar (ID, Ism, Yosh)
VALUES (1, 'Alisher', 20);
```

Bu buyruq **Talabalar** jadvaliga yangi qator qo‘shadi:

2. Barcha ustunlarga ma’lumot qo‘shish

Agar jadvaldagi barcha ustunlarga qiymat berilayotgan bo‘lsa, ustunlar ro‘yxatini keltirmaslik mumkin:

```
INSERT INTO jadval_nomi
VALUES (qiymat1, qiymat2, ...);
```

Misol:

```
INSERT INTO Talabalar
VALUES (2, 'Madina', 19);
```

3. Bir nechta qatorlarni qo‘shish

Bir vaqtning o‘zida bir nechta qatorlarni kiritish uchun quyidagi sintaksis ishlatiladi:

```
INSERT INTO jadval_nomi (ustun1, ustun2, ...)
VALUES (qiymat1, qiymat2, ...),
    (qiymat3, qiymat4, ...);
```

Misol:

```
INSERT INTO Talabalar (ID, Ism, Yosh)
VALUES (3, 'Doston', 21),
    (4, 'Zarina', 22);
```

10.8. Xar bir ustun uchun tip (toifa) va o‘lcham

Ma’lumotlar bazasi jadvallarida **ustunlar** uchun toifa (ma’lumot turi) va o‘lchamni aniqlash muhim ahamiyatga ega. Har bir ustun faqat ma’lum bir turdagi va o‘lchamdagi ma’lumotlarni qabul qilishi mumkin. Bu ma’lumotlar bazasi samaradorligi va ma’lumotlarning yaxlitligini ta’minlashda muhim rol o‘ynaydi.

Ma’lumot turi — bu ustun qiymatlarini qanday ma’lumotlarni (sonlar, matn, sanalar va h.k.) saqlashini belgilovchi xususiyatdir. Ma’lumot turlarining tanlovi quyidagilarni boshqaradi:

Xotira sarfi: Har bir tur ma’lumotlarni saqlash uchun o‘ziga xos hajm talab qiladi.

Aniqlik va chegaralar: Ba’zi turlar faqat belgilangan diapazondagi qiymatlarni qabul qiladi.

Foydalanuvchi amaliyoti: Toifalar turli so‘rovlarda (masalan, arifmetik yoki matnli operatsiyalar) foydalanishga imkon beradi.

Asosiy ma’lumot turlari va ularning o‘lchami

1. Raqamli turlar
2. Matnli turlar
3. Sana, vaqt turlari va h.k.

Toifa va o‘lchamni belgilash bo‘yicha tavsiyalar

Matnli ma’lumotlar uchun CHAR yoki VARCHAR tanlashda uzunlikni aniq belgilash muhim.

Raqamlar uchun qabul qilinadigan qiymat diapazoniga mos tur tanlanadi.

Xotiradan samarali foydalanish: Katta o‘lchamli turlarni faqat zarurat bo‘lganda ishlatish.

Nazorat savollari:

1. SQL tilining jadval yaratish operatori sintaksisi qanday?
2. Indeks operatorini ko‘rinishi va uning vazifasi?
3. Drop va Delete operatorlarining farqi nimada?
4. Trigger nima?
5. Funksiya va protseduralar qanday yaratiladi?

11 BOB. SQLda jarayonlar va standart funksiyalar

Tayanch so‘zlar: *Standart funksiyalar, agregat funksiyalar, Concat(), Upper(), Lower(), Round()*

11.1. SQL tilida standart va agregat funksiyalar

SQL tilida standart va agregat funksiyalar ma’lumotlar bazasida tahlil qilish, manipulyatsiya qilish va natijalarni olishda muhim vositalardir.

SQL (Structured Query Language) – ma’lumotlar bazasidagi ma’lumotlar bilan ishlash uchun ishlatiladigan standart tildir.

SQLda **standart funksiyalar** va **agregat funksiyalar** ma’lumotlar bazasini boshqarishda juda muhim vositalardir. **Standart funksiyalar** ma’lumotlarni tahlil qilish, manipulyatsiya qilish va optimallashtirishga yordam beradi. **Agregat funksiyalar** esa ma’lumotlarni guruhlash va umumlashtirishda ishlatiladi. Bu funksiyalarning har biri SQL standartlariga muvofiq ishlaydi va barcha ma’lumotlar bazasida qo‘llanilishi mumkin.

11.2. Agregat funksiyalar argumentlari

Agregat funksiyalar SQLda bir nechta satrlar ustida hisoblashlar amalga oshiradi. Ular ma’lumotlar to‘plamini umumlashtirish uchun ishlatiladi.

COUNT() –Bu funksiya ma’lumotlar bazasidagi satrlar sonini hisoblash uchun ishlatiladi.

Misol:

```
SELECT COUNT(*) AS TotalStudents FROM Students;
```

Natija:

TotalStudents
100

Bu yerda Students jadvalidagi jami talabalarning soni 100.

SUM() – Bu funksiya ma’lum ustundagi qiymatlarni qo‘shib chiqadi.

Misol:

```
SELECT SUM(Salary) AS TotalSalaries FROM Employees;
```

Natija:

TotalSalaries
500000

Bu yerda Employees jadvalidagi barcha xodimlarning maoshlarining yig‘indisi hisoblanadi.

AVG() – O‘rtacha qiymatni hisoblash

Bu funksiya ma’lum ustundagi qiymatlar o‘rtachasini hisoblaydi.

Misol:

```
SELECT AVG(Grade) AS AverageGrade FROM Students;
```

Natija:

AverageGrade
85.5

Bu yerda Students jadvalidagi talabalar o‘rtacha ballari 85.5 ga teng.

MIN() – Eng kichik qiymatni olish

Bu funksiya ma’lum ustundagi eng kichik qiymatni topadi.

Misol:

```
SELECT MIN(Salary) AS MinimumSalary FROM Employees;
```

Natija:

MinimumSalary
25000

Bu yerda Employees jadvalidagi eng kam maosh 25000.

MAX() – Bu funksiya ma’lum ustundagi eng katta qiymatni topadi.

Misol:

```
SELECT MAX(Salary) AS MaximumSalary FROM Employees;
```

Natija:

MaximumSalary

120000

Bu yerda Employees jadvalidagi eng katta maosh 120000.

Agregat funksiyalar va GROUP BY

Agregat funksiyalar odatda GROUP BY operatori bilan birga ishlatiladi, bu ma'lum bir guruh bo'yicha umumlashtirish imkonini beradi.

Misol:

```
SELECT Department, COUNT(*) AS NumberOfEmployees, AVG(Salary) AS AverageSalary
FROM Employees
GROUP BY Department;
```

Natija:

Department	NumberOfEmployees	AverageSalary
HR	5	55000
Sales	8	60000
Marketing	3	65000

Bu yerda Employees jadvalidagi har bir bo'lim uchun xodimlar soni va o'rtacha maosh ko'rsatilgan.

11.3. Standart funksiyalar orqali so'rovlar yaratish

1. Standart SQL funksiyalari

Standart funksiyalar SQL tilida ma'lumotlarni tahlil qilish uchun keng qo'llaniladi. Ular orasida matnli, sanali, sana va vaqtga oid funksiyalar, shuningdek shartli funksiyalar mavjud.

1.1. Matnli funksiyalar (String Functions)

CONCAT(): Matnni birlashtirish uchun ishlatiladi.

Misol:

```
SELECT CONCAT('Hello', ' ', 'World') AS Greeting;
```

UPPER(): Matndagi barcha harflarni katta qilish.

Misol:

```
SELECT UPPER('hello') AS UppercaseText;
```

LOWER(): Matndagi barcha harflarni kichik qilish.

Misol:

```
SELECT LOWER('HELLO') AS LowercaseText;
```

1.2. Sonli funksiyalar (Numeric Functions)

ROUND(): Sonni ma'lum o'nlik raqamga yaxlitlash.

Misol:

```
SELECT ROUND(123.4567, 2) AS RoundedNumber;
```

Natija:

RoundedNumber
123.46

ABS(): Sonning modulini olish.

Misol:

```
SELECT ABS(-123.45) AS AbsoluteValue;
```

Natija:

AbsoluteValue
123.45

1.3. Sanaga oid funksiyalar (Date Functions)

CURRENT_DATE: Hozirgi sanani olish.

Misol:

```
SELECT CURRENT_DATE AS TodayDate;
```

Natija:

TodayDate
2024-11-07

DATEADD(): Sana yoki vaqtga ma'lum bir qiymat qo'shish.

Misol:

```
SELECT DATEADD(DAY, 10, '2024-11-01') AS NewDate;
```

Natija:

NewDate
2024-11-11

11.4. Standart funksiyalarning SQLda sintaksisi

Standart funksiyalarni quyidagilarga ajratish mumkin: **matn funksiyalari**, **sanalar bilan ishlash funksiyalari** va **mantiqiy funksiyalar**.

Matn funksiyalari

Matn funksiyalari SQLda matn (string) ma'lumotlarni tahrirlash va boshqarishda qo'llaniladi. Quyidagi matn funksiyalari eng ko'p ishlatiladi:

CONCAT

Sintaksis:

CONCAT(string1, string2, ...)

Bu funksiya bir nechta matnlarni birlashtiradi.

LENGTH

Sintaksis:

LENGTH(string)

Bu funksiya matnning uzunligini (belgilar sonini) qaytaradi.

UPPER

Sintaksis:

UPPER(string)

Bu funksiya matnni katta harflar bilan qaytaradi.

LOWER

Sintaksis:

LOWER(string)

Bu funksiya matnni kichik harflar bilan qaytaradi.

SUBSTRING

Sintaksis:

SUBSTRING(string, start, length)

Bu funksiya matndan belgilangan joydan boshlab, kerakli uzunlikdagi qismni ajratib beradi.

2. Sana va vaqt funksiyalari

Sana va vaqt funksiyalari SQLda sanalar va vaqtlar bilan ishlash uchun qo‘llaniladi. Quyidagi funksiyalar eng ko‘p ishlatiladi:

CURRENT_DATE

Sintaksis:

CURRENT_DATE

Hozirgi sanani qaytaradi.

CURRENT_TIME

Sintaksis:

CURRENT_TIME

Hozirgi vaqtni qaytaradi.

NOW

Sintaksis:

NOW()

Hozirgi sana va vaqtni qaytaradi.

DATE_ADD

Sintaksis:

DATE_ADD(date, INTERVAL value unit)

Sana yoki vaqtga belgilangan intervalni qo‘shadi.

4. Mantiqiy funksiyalar

SQLda mantiqiy funksiyalar ma’lumotlarni mantiqiy ifodalash uchun ishlatiladi. Ular shartli tekshiruvlar va filtrlarni bajarishda qo‘llaniladi:

COALESCE

Sintaksis:

COALESCE(expression1, expression2, ...)

Bu funksiya birinchi **NOT NULL** qiymatni qaytaradi.

NULLIF

Sintaksis:

NULLIF(expression1, expression2)

Bu funksiya agar **expression1** va **expression2** teng bo‘lsa, **NULL** qiymatini qaytaradi.

11.5. DISTINCT standart so‘zi va undan foydalanib ikki nusxadagi satrlarni o‘chirish

DISTINCT va **ORDER BY** SQL tilida keng tarqalgan operatorlardir. Ular ma’lumotlarni filtrlash va tartiblashda qo‘llaniladi.

1. DISTINCT operatori

DISTINCT SQL operatori bir xil qiymatlarni takrorlashni oldini oladi va faqat noyob (unique) qiymatlarni qaytaradi. U ko‘pincha SELECT operatori bilan birga ishlatiladi va so‘rovning natijasida faqat bir marta takrorlanadigan satrlarni ko‘rsatadi.

```
SELECT DISTINCT column_name
FROM table_name;
```

Misol:

Agar sizda **Employees** jadvali bo‘lsa va xodimlar shaharlarini bilmoqchi bo‘lsangiz, lekin takrorlanadigan shaharlarni chiqarishni xohlamasangiz:

```
SELECT DISTINCT City
FROM Employees;
```

Natija:

City
New York

Los Angeles
Chicago

Bu so‘rovda, **DISTINCT** operatori **Employees** jadvalidagi har bir noyob shaharni qaytaradi. Takrorlangan shaharlar (agar mavjud bo‘lsa) chiqarilmaydi.

DISTINCT yordamida ko‘plab takrorlanadigan ma’lumotlar soni kamaytiriladi va ma’lumotlar bazasining samaradorligi yaxshilanadi.

11.6. ORDER BY komandasidan foydalanib satrlarni tartiblashtirish

ORDER BY SQL operatori natijaviy satrlarni tartiblash uchun ishlatiladi. Bu operator yordamida natijalar **katta kichik (ASC)** yoki **kichik katta (DESC)** tartibda ko‘rsatilishi mumkin.

Sintaksis:

```
SELECT column_name(s)
FROM table_name
ORDER BY column_name [ASC|DESC];
```

ASC — bu **ascending (o‘sish tartibi)**, ya’ni qiymatlar kichikdan kattaga qarab tartiblanadi. Bu default tartib.

DESC — bu **descending (kamayish tartibi)**, ya’ni qiymatlar kattadan kichikgacha qarab tartiblanadi.

Misol:

Agar sizda **Employees** jadvali bo‘lsa va xodimlarni oylik maoshiga ko‘ra tartiblashni istasangiz:

```
SELECT EmployeeName, Salary
FROM Employees
ORDER BY Salary DESC;
```

Natija:

EmployeeName	Salary
John	100000

Alice	85000
Bob	50000

Bu so‘rovda **ORDER BY** operatori orqali xodimlar oylik maoshlari bo‘yicha kamayish tartibida (kattadan kichikka) tartiblanadi.

DISTINCT va **ORDER BY** operatorlarini birgalikda ishlatish mumkin. Misol uchun, agar siz faqat noyob shaharlarni olishni va ularni nomiga ko‘ra tartiblashtirishni xohlasangiz:

```
SELECT DISTINCT City
FROM Employees
ORDER BY City ASC;
```

Bu so‘rovda:

DISTINCT operatori har bir shaharni faqat bitta marta qaytaradi.

ORDER BY operatori natijalarni shahar nomiga ko‘ra tartiblaydi.

Demak, **DISTINCT** operatori ma’lumotlar to‘plamida takrorlangan qiymatlarni chiqarib tashlaydi va faqat noyob satrlarni ko‘rsatadi.

ORDER BY operatori esa natijalarni ko‘rsatilgan tartibda, ya’ni o‘sish yoki kamayish tartibida tartiblaydi.

Ushbu operatorlar SQL tilining **standart funksiyalari** bo‘lib, ma’lumotlarni tahlil qilish va tartiblashda keng qo‘llaniladi.

11.7. Agregatlar va ma’lumotlarni guruhlash

WHERE va **HAVING** SQL operatorlari ma’lumotlarni filtrlashda ishlatiladi, ammo ular bir-biridan ba’zi jihatlari bilan farq qiladi.

1. WHERE operatori

WHERE operatori SQL so‘rovlarida ma’lumotlarni filtrlashda ishlatiladi. Bu operator **SELECT**, **UPDATE**, **DELETE** so‘rovlarida qo‘llaniladi va faqat ma’lum shartlarga javob beradigan satrlarni chiqaradi. **WHERE** operatori natijalar ustunlari

yoki agregat funksiyalarini hisobga olmaydi; ya'ni, bu operator faqat ustunlar darajasida ma'lumotlarni filtrlash uchun ishlatiladi.

Sintaksis:

```
SELECT column_name(s)
FROM table_name
WHERE condition;
```

Misol:

Agar siz **Employees** jadvalidan faqat oyligi 50,000 dan yuqori bo'lgan xodimlarni olishni istasangiz:

```
SELECT EmployeeName, Salary
FROM Employees
WHERE Salary > 50000;
```

Bu so'rov faqat **Salary** ustunidagi qiymati 50,000 dan katta bo'lgan xodimlarni chiqaradi.

WHERE operatori asosan individual ustunlarga qo'llaniladi.

Agregat funksiyalar va guruhlashdan avval **WHERE** operatori ishlatiladi.

WHERE operatori yordamida matematik ifodalar, satrlarni solishtirish va boshqa shartlarni qo'llash mumkin.

2. HAVING operatori

HAVING operatori **guruhlangan (grouped) ma'lumotlar** uchun filtrlashda ishlatiladi. **HAVING** operatori, odatda, **GROUP BY** operatori bilan birgalikda ishlatiladi va faqat agregat funktsiyalari (masalan, COUNT(), SUM(), AVG()) bilan hisoblangan natijalarni filtrlaydi. Agar **WHERE** operatori shartlarni filtrlasa, **HAVING** operatori esa guruhlangan ma'lumotlar bo'yicha shart qo'yadi.

Sintaksis:

```
SELECT column_name(s), aggregate_function(column_name)
FROM table_name
```

```
WHERE condition
GROUP BY column_name
HAVING aggregate_function(column_name) condition;
```

Misol:

Agar siz **Employees** jadvalidan har bir bo‘limning o‘rtacha oyligini hisoblab, o‘rtacha oylik 60,000 dan yuqori bo‘lgan bo‘limlarni olishni istasangiz:

```
SELECT Department, AVG(Salary) AS AverageSalary
FROM Employees
GROUP BY Department
HAVING AVG(Salary) > 60000;
```

Bu so‘rov:

Avval **Department** ustuni bo‘yicha guruhlaydi.

Har bir guruh uchun **AVG(Salary)** orqali o‘rtacha oylikni hisoblaydi.

So‘ngra, **HAVING** operatori yordamida o‘rtacha oyligi 60,000 dan yuqori bo‘lgan guruhlarni qaytaradi.

HAVING operatori agregat funktsiyalarni hisoblab chiqqan natijalarni filtrlash uchun ishlatiladi.

HAVING operatori faqat **GROUP BY** bilan ishlaganda qo‘llaniladi.

HAVING bilan ishlashda shartlar agregat funktsiyalariga tegishli bo‘ladi.

WHERE va HAVING operatorlari o‘rtasidagi farqlar

Funksiya darajasi:

WHERE operatori ma’lumotlar bazasidan satrlarni tanlab olishda ishlatiladi va faqat ustunlar darajasida shartlar qo‘yadi.

HAVING operatori esa guruhlangan natijalarga qo‘llaniladi va agregat funktsiyalar orqali hisoblangan qiymatlar bo‘yicha shartlar qo‘yadi.

Guruhlash bilan aloqasi:

WHERE operatori guruhlanmagan satrlarni filtrlashda ishlatiladi.

HAVING operatori esa **GROUP BY** operatori bilan ishlaydi va faqat guruhlangan ma'lumotlarga taalluqlidir.

WHERE va HAVING operatorlarining qo'llanilishiga misollar:

1. WHERE operatori:

Misol uchun, xodimlarning 50,000 dan yuqori bo'lgan oyliklarini olish:

```
SELECT EmployeeName, Salary
FROM Employees
WHERE Salary > 50000;
```

2. HAVING operatori:

Har bir bo'limning o'rtacha oylik maoshini hisoblash va o'rtacha oylik 60,000 dan yuqori bo'lgan bo'limlarni ko'rsatish:

```
SELECT Department, AVG(Salary) AS AverageSalary
FROM Employees
GROUP BY Department
HAVING AVG(Salary) > 60000;
```

Har ikkala operator ham SQL so'rovlarida ma'lumotlarni filtrlashda muhim rol o'ynaydi, ammo ularning ishlatilish maydoni va vazifalari turlicha.

GROUP BY operatori SQL tilida ma'lumotlarni guruhlash uchun ishlatiladi. Bu operator yordamida bir yoki bir nechta ustunlarga asoslanib ma'lumotlar guruhlanadi. Guruhlash natijasida har bir guruh uchun agregat funktsiyalari (masalan, COUNT(), SUM(), AVG()) yordamida hisoblashlar amalga oshiriladi. **GROUP BY** operatori ko'pincha **HAVING** operatori bilan birgalikda ishlatiladi.

GROUP BY operatorining asosiy vazifasi:

Ma'lumotlarni guruhlash: Bir yoki bir nechta ustunlar bo'yicha ma'lumotlarni guruhlash.

Agregat funktsiyalari: Guruhlangan ma'lumotlarga nisbatan agregat funksiyalarini (masalan, SUM(), COUNT(), AVG()) qo'llash.

Sintaksis:

```
SELECT column_name(s), aggregate_function(column_name)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

Misollar:

1. GROUP BY bilan ishlash

Misol 1: Agar biz **Employees** jadvalidan har bir bo‘limdagi xodimlar sonini bilmoqchi bo‘lsak, quyidagi so‘rovni yozamiz:

```
SELECT Department, COUNT(EmployeeID) AS NumberOfEmployees
FROM Employees
GROUP BY Department;
```

Bu so‘rov har bir bo‘limdagi xodimlar sonini hisoblaydi. Natijada quyidagicha jadval olinadi:

Department	NumberOfEmployees
HR	10
IT	15
Sales	8

2. GROUP BY va agregat funksiyalar

Misol 2: Agar biz har bir bo‘lim uchun o‘rtacha oylikni hisoblamoqchi bo‘lsak, quyidagi so‘rovni ishlatamiz:

```
SELECT Department, AVG(Salary) AS AverageSalary
FROM Employees
GROUP BY Department;
```

Bu so‘rov har bir bo‘limdagi o‘rtacha oylikni hisoblaydi. Natija quyidagicha bo‘lishi mumkin:

Department	AverageSalary
HR	55000
IT	75000
Sales	48000

3. GROUP BY va HAVING bilan ishlash

Misol 3: Agar biz har bir bo‘limning o‘rtacha oylik maoshini hisoblab, o‘rtacha oylik 60,000 dan yuqori bo‘lgan bo‘limlarni ko‘rmoqchi bo‘lsak, quyidagi so‘rovni ishlatamiz:

```
SELECT Department, AVG(Salary) AS AverageSalary
FROM Employees
GROUP BY Department
HAVING AVG(Salary) > 60000;
```

Bu so‘rov har bir bo‘limdagi o‘rtacha oylikni hisoblaydi va faqat 60,000 dan yuqori bo‘lgan bo‘limlarni chiqaradi. Natija quyidagicha bo‘lishi mumkin:

Department	AverageSalary
IT	75000

4. GROUP BY bilan bir nechta ustunlar

Misol 4: Agar biz har bir bo‘limdagi o‘rtacha oylikni va xodimlar sonini bilmoqchi bo‘lsak, ikki ustunni ham guruhlashimiz mumkin:

```
SELECT Department, Gender, COUNT(EmployeeID) AS NumberOfEmployees, AVG(Salary) AS AverageSalary
FROM Employees
GROUP BY Department, Gender;
```

Bu so‘rov har bir bo‘lim va jins bo‘yicha xodimlar sonini va o‘rtacha oylikni hisoblaydi. Natija quyidagicha bo‘lishi mumkin:

Department	Gender	NumberOfEmployees	AverageSalary
HR	Male	5	60000

HR	Female	5	50000
IT	Male	9	78000
IT	Female	6	71000

GROUP BY operatorining ishlash tartibi:

Filtrlash (WHERE operatori): Dastlab ma'lumotlar filtrlash uchun **WHERE** operatori ishlatiladi (agar kerak bo'lsa).

Guruhlash (GROUP BY operatori): So'ngra ma'lumotlar guruhlanadi.

Agregat funktsiyalari: Guruhlangan ma'lumotlarga **COUNT()**, **SUM()**, **AVG()** kabi agregat funksiyalar qo'llaniladi.

Filtrlash (HAVING operatori): Agar kerak bo'lsa, guruhlangan natijalar **HAVING** operatori yordamida filtrlanadi.

GROUP BY operatori SQL so'rovlarida guruhlangan ma'lumotlarni olish uchun ishlatiladi va agregat funksiyalar bilan birgalikda ishlatiladi.

Bu operator **HAVING**, **COUNT()**, **AVG()**, **SUM()** kabi funktsiyalar bilan ishlatilganda qulaylik yaratadi va katta hajmdagi ma'lumotlar ustida tahlil qilishni osonlashtiradi.

Amaliy qism.

Bizda Employees jadvali bo'lsin. U xodimlar haqida ma'lumotlarni saqlash uchun ishlatiladigan jadval bo'ladi. Unda xodimlar bilan bog'liq ma'lumotlar (masalan, ism, lavozim, maosh, bo'lim, va boshqalar) saqlanadi. Quyida **Employees** jadvalining mumkin bo'lgan tuzilishi va ma'lumotlar ko'rsatilgan:

Employees Jadvalining Tuzilishi

Field Name	Data Type	Description
Employee_ID	INT	Xodimning noyob identifikatori
First_Name	VARCHAR(50)	Xodimning ismi
Last_Name	VARCHAR(50)	Xodimning familiyasi
Job_Title	VARCHAR(50)	Xodimning lavozimi
Department	VARCHAR(50)	Xodim ishlaydigan bo'lim

Hire_Date	DATE	Xodimni ishga qabul qilish sanasi
Salary	DECIMAL(10, 2)	Xodimning maoshi
Manager_ID	INT	Xodimning rahbari (agar mavjud bo‘lsa)

Employees Jadvalini yaratish

```
CREATE TABLE Employees (
    Employee_ID INT PRIMARY KEY,
    First_Name VARCHAR(50),
    Last_Name VARCHAR(50),
    Job_Title VARCHAR(50),
    Department VARCHAR(50),
    Hire_Date DATE,
    Salary DECIMAL(10, 2),
    Manager_ID INT
);
```

Misol Ma’lumotlar

Employee_ID	First_Name	Last_Name	Job_Title	Department	Hire_Date	Salary	Manager_ID
1	John	Doe	Software Dev	IT	2022-01-15	5000	NULL
2	Mary	Smith	Data Analyst	IT	2021-03-25	5000	1
3	Alice	Johnson	HR Manager	HR	2020-05-10	5000	NULL
4	Bob	Brown	System Admin	IT	2019-07-11	5000	1
5	Carol	Williams	Marketing Lead	Marketing	2023-06-19	0000	3

Misol: Employees Jadvalidan Ma’lumotlar So‘rovi

Barcha xodimlarning ma’lumotlarini olish:

```
SELECT * FROM Employees;
```

IT bo‘limida ishlaydigan xodimlarni ko‘rsatish:

```
SELECT First_Name, Last_Name, Job_Title FROM Employees
```

```
WHERE Department = 'IT';
```

Maoshi 80000 dan katta bo'lgan xodimlarni ko'rsatish:

```
SELECT First_Name, Last_Name, Salary FROM Employees
WHERE Salary > 80000;
```

Xodimlarni ishga qabul qilish sanasi bo'yicha saralash:

```
SELECT * FROM Employees
ORDER BY Hire_Date DESC;
```

Manager_ID bo'yicha guruhlash va ularning o'rtacha maoshini ko'rsatish:

```
SELECT Manager_ID, AVG(Salary) AS Average_Salary
FROM Employees
GROUP BY Manager_ID;
```

Xodimlarning umumiy maoshini hisoblash:

```
SELECT SUM(Salary) AS Total_Salary
FROM Employees;
```

Bu so'rov **Employees** jadvalidagi barcha xodimlarning maoshlarini yig'adi va umumiy maoshni **Total_Salary** deb nomlangan ustunda ko'rsatadi.

Xodimlarning o'rtacha maoshini hisoblash:

```
SELECT AVG(Salary) AS Average_Salary
FROM Employees;
```

Nazorat savollari:

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

12 BOB. TRANZAKSIYALARNI BOSHQARISHDA SO‘ROVLAR YARATISH VA QAYTA ISHLASH

Tayanch so‘zlar: *tranzaksiya, TCL, Atomiklik, Izolyatsiya, trigger, before trigger, after trigger, Commit, Begin, Rollback.*

12.1. SQL muhitida tranzaksiya tushunchasi

TCL (Transaction Control Language) — bu SQLning bir bo‘limi bo‘lib, ma’lumotlar bazasida tranzaksiyalarni boshqarish uchun ishlatiladi. TCL operatorlari ma’lumotlar bazasining bir yoki bir nechta jadvallariga ta’sir qiladigan operatsiyalarni guruhlash va boshqarishga imkon beradi. Bu operatorlar ma’lumotlarning yaxlitligi va butunligini ta’minlashda yordam beradi.

SQL muhitida *tranzaksiya* tushunchasi — bu ma’lumotlar bazasida bir qancha operatsiyalarni bir vaqtning o‘zida bajarish jarayonini anglatadi. SQL tilidagi har bir chaqirish moduli tranzaksiyadir. SQL tranzaksiyalari biror-bir modulning protseduralarini bajarishdan boshlanadi. Tranzaksiya muvaffaqiyatli bajarilsa, u bilan bog‘liq barcha operatsiyalar bir vaqtda amalga oshadi; agar ulardan birortasi xato qilsa, barcha o‘zgarishlar bekor qilinadi. Tranzaksiya butunligini ta’minlash uchun u quyidagi xususiyatlarga amal qiladi, bular *ACID* (Atomicity, Consistency, Isolation, Durability) xususiyatlari deb ataladi:

1. **Atomiklik (Atomicity)**: Tranzaksiya butunlay bajarilishi yoki hech qanday o‘zgarishsiz bekor qilinishi kerak. Har qanday holatda, tranzaksiya to‘liq tugatiladi yoki umuman amalga oshmaydi.
2. **Izchillik (Consistency)**: Tranzaksiya boshlanishidan oldin va keyin ma’lumotlar bazasi izchillikni saqlashi kerak. Tranzaksiya davomida qilingan barcha o‘zgarishlar ma’lumotlar bazasi qoidalari va cheklovlariga mos bo‘lishi kerak.
3. **Izolyatsiya (Isolation)**: Bir vaqtning o‘zida bajarilayotgan bir nechta tranzaksiyalar bir-biriga ta’sir qilmasligi kerak. Tranzaksiya davomida

kiritilayotgan o‘zgarishlar boshqa tranzaksiyalar tomonidan ko‘rilmasligi ta’minlanadi.

4. **Barqarorlik (Durability)**: Tranzaksiya tugagach, o‘zgarishlar saqlanib qoladi, hatto tizimda nosozliklar yuz bersa ham, bu o‘zgarishlar ma’lumotlar bazasida mavjud bo‘lib qoladi.

12.2. SQL muhitida tranzaksiyalarni boshqarish

SQLda tranzaksiyalarni boshqarish uchun asosan quyidagi operatorlardan foydalaniladi:

- BEGIN TRANSACTION: Tranzaksiyani boshlaydi. (POSTGRESQL da Tranzaksiya boshlash uchun BEGIN komandasidan foydalaniladi).
- COMMIT: Tranzaksiyani yakunlaydi va barcha o‘zgarishlarni saqlaydi.
- ROLLBACK: Tranzaksiyani bekor qiladi va dastlabki holatga qaytaradi.
- **SAVEPOINT**: Tranzaksiyani belgilangan nuqtada saqlaydi. Agar tranzaksiya xato bo‘lsa, ROLLBACK yordamida faqat shu nuqtaga qadar bo‘lgan o‘zgarishlar qaytariladi.
- **SET TRANSACTION**: Tranzaksiya izolyatsiya darajasini belgilaydi. Bu operator tranzaksiyalarni qanday ko‘rish va boshqarish kerakligini aniqlashda yordam beradi, masalan, ular izolyatsiya qilinganmi yoki boshqalar tomonidan ko‘rinishi mumkinmi.

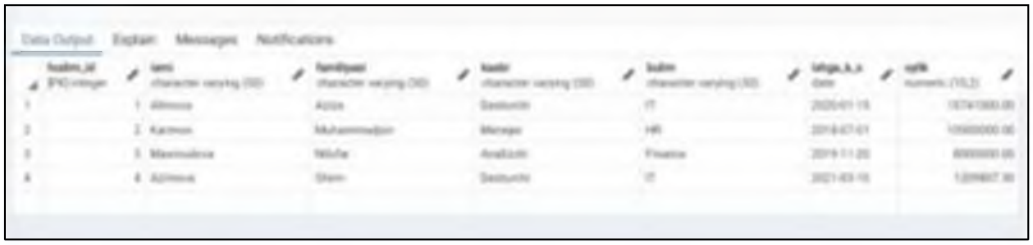
Bu tranzaksiya boshqaruvi xususiyatlari, ayniqsa ko‘p foydalanuvchilar bir vaqtning o‘zida ishlaganda, muhim bo‘lib, ular ma’lumotlarning izchilligini ta’minlashda yordam beradi.

Misol: *IT bulimda ishlaydigan hodimlarning oyiligini 10% ga oshirish tranzaksiyasi.*

```
BEGIN;
UPDATE hodimlar
SET oylik = oylik * 1.10
```

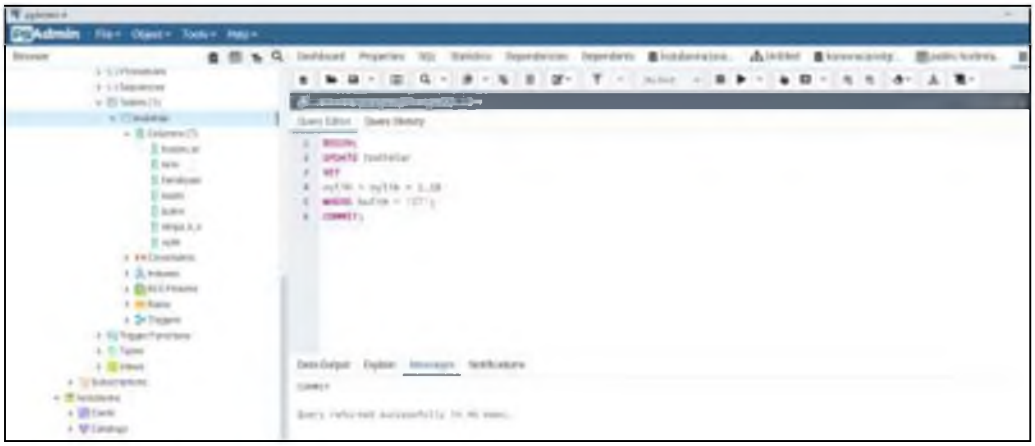
```
WHERE bulim = 'IT';
COMMIT;
```

Jadvalning dastlabki holati quyidagicha:



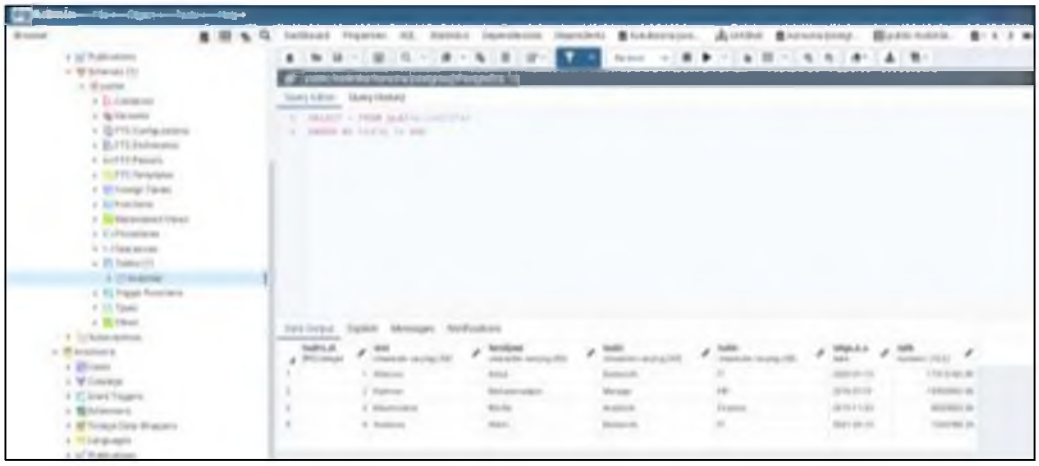
12.1-rasm. Jadvalning dastlabki holati

Yuqoridagi tranzaksiya kodini Postgre SQL dasturida yozamiz.



12.2-rasm. Postgre SQL dasturida tranzaksiya yozilishi

Natijada tranzaksiya muaffaqiyatli bajarilganligi haqidagi ma'lumotga ega bo'ldik. Agar jadvalda o'zgarish bo'lmasa yangilash (refresh)ni amalga oshiring.



12.3-rasm. Tranzaksiya natijasi

Agar tranzaksiya davomida xatolik bo‘lsa, **ROLLBACK** orqali barcha o‘zgarishlar bekor qilinadi va tranzaksiya boshlangandagi holatga qaytariladi.

```
BEGIN;
UPDATE hodimlar
SET
oylik = oylik * 1.10
WHERE bulim = 'IT';
 ROLLBACK;
```

SAVEPOINT - SAVEPOINT tranzaksiya ichida biror nuqtani belgilash imkonini beradi. Agar tranzaksiya xato bo‘lsa, faqat shu nuqtaga qadar bo‘lgan o‘zgarishlar qaytariladi, boshqalar esa saqlanadi.

```
BEGIN;
SAVEPOINT savepoint1;
UPDATE hodimlar SET oylik = oylik * 1.10
 WHERE bulim = 'IT';
 -- Xato bo'lsa, faqat savepoint1 ga qaytamiz
ROLLBACK TO SAVEPOINT savepoint1;
COMMIT;
```

SET TRANSACTION operatori tranzaksiyaning izolyatsiya darajasini belgilash imkonini beradi.

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
BEGIN;
UPDATE hodimlar SET oylik = oylik * 1.10
WHERE bulim = 'IT';
COMMIT;
```

Tranzaksiyalarning izolyatsiya darajalari

SQL muhitida tranzaksiyalarning izolyatsiya darajalarini belgilash orqali ulardan biri davom etayotganda boshqa tranzaksiyalarning ta'sirini cheklash mumkin. Bu quyidagi darajalarni o'z ichiga oladi:

- **SERIALIZABLE**: Eng yuqori izolyatsiya darajasi. Boshqa tranzaksiyalar o'zgarishlarni ko'rmaydi va ular yakunlangandan keyin barcha o'zgarishlar amalga oshiriladi.
- **REPEATABLE READ**: Tranzaksiya davomida ma'lumotlar o'zgarmaydi, lekin yangi satrlar qo'shilishi mumkin.
- **READ COMMITTED**: Tranzaksiya bajarilayotgan vaqtda boshqa tranzaksiyalar tomonidan o'zgartirilgan ma'lumotlarni ko'rish mumkin.
- **READ UNCOMMITTED**: Tranzaksiya davomida boshqa tranzaksiyalar tomonidan yaratilgan o'zgarishlar ko'rinishi mumkin.

12.3. Arifmetik jarayonlar va hisoblash tartibini belgilash

SQLda tranzaksiyalarni amalga oshirishda arifmetik amallar ketma-ketligini aniqlash tartibi ham muhimdir. SQL hisoblash jarayonlarini qavslar va arifmetik operatorlar yordamida bajaradi, bu esa tranzaksiya doirasida hisoblashlar qanday amalga oshirilishini boshqarishga imkon beradi. Masalan, kiritilgan **(Price * Quantity) - Discount** kabi hisoblashlarda tranzaksiya davomida o'zgarishlarni to'g'ri ketma-ketlikda saqlash orqali hisoblashlar aniqligi ta'minlanadi.

Misol:

```
BEGIN TRANSACTION;
  UPDATE Accounts SET balance = balance - 100 WHERE account_id = 1;
  UPDATE Accounts SET balance = balance + 100 WHERE account_id = 2;
COMMIT;
```

Yuqoridagi misolda tranzaksiya boshlanganda hisob raqamlar o'rtasida pul o'tkazilishi amalga oshiriladi. Agar tranzaksiya davomida xatolik bo'lsa,

ROLLBACK orqali barcha o‘zgarishlar bekor qilinadi va tranzaksiya boshlangandagi holatga qaytariladi.

Tranzaksiyaga namuna:

Quyida hodimlar jadvali ustida tranzaksiyalarni amalga oshiramiz. Ushbu misolda bir nechta hodimning maoshlarini o‘zgartiramiz, agar operatsiyada xatolik yuz bermasa o‘zgarishlar tasdiqlanadi, aks holda, barcha o‘zgarishlar bekor qilinadi.

Avval hodimlar haqida ma’lumot saqlovchi **hodimlar** jadvalini yaratamiz:

```
CREATE TABLE hodimlar (
    hodim_ID SERIAL PRIMARY KEY,
    Ismi VARCHAR(50),
    Familiyasi VARCHAR(50),
    Kasbi VARCHAR(50),
    Bulim VARCHAR(50),
    Ishga_k_s DATE,
    Oylik DECIMAL(10, 2)
);
```

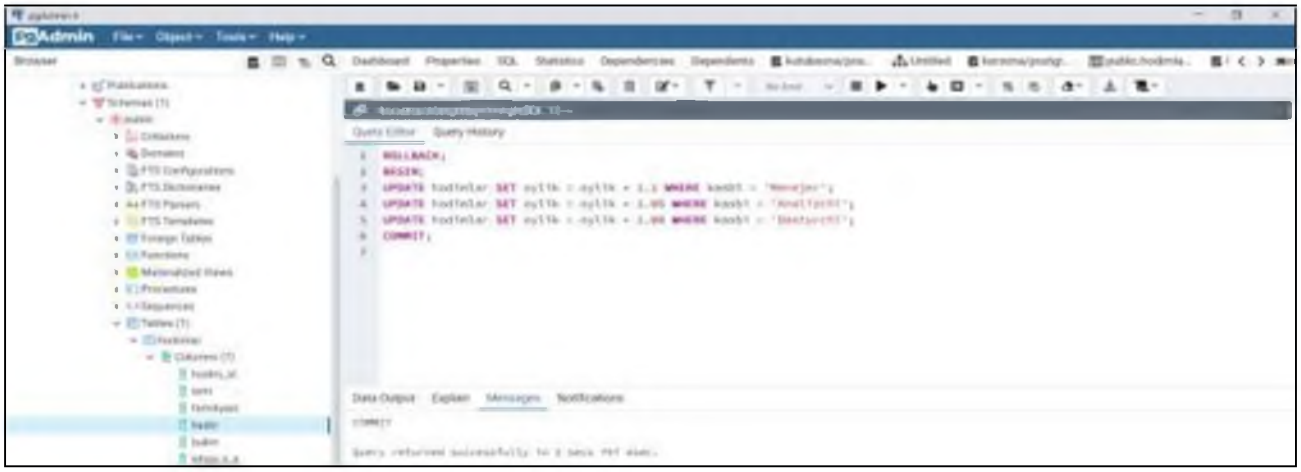
Jadval tuzilgandan so‘ng, unga ba’zi ma’lumotlar kiritamiz:

```
INSERT INTO hodimlar ( ismi, familiyasi, kasbi, bulim, ishga_k_s, oylik)
VALUES
('Alimova', 'Aziza', 'Dasturchi', 'IT', '2020-01-15', 10000000.00),
('Karimov', 'Muhammadjon', 'Menejer', 'HR', '2018-07-01', 9500000.00),
('Maxmudova', 'Nilufar', 'Analizchi', 'Finance', '2019-11-20', 7000000.00),
('Azimova', 'Shirin', 'Dasturchi', 'IT', '2021-03-10', 75000.00);
```

Endi maoshlarni oshirish misolida **tranzaksiya** amalga oshiramiz. Bu misolda agar barcha maosh o‘zgarishlari muvaffaqiyatli bo‘lsa, ular tasdiqlanadi. Aks holda, barcha o‘zgarishlar bekor qilinadi.

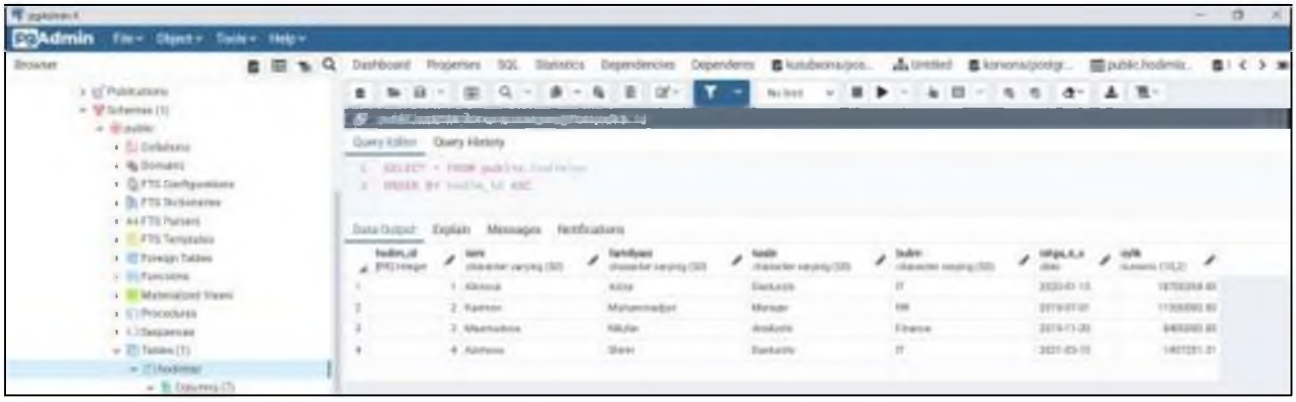
```
BEGIN;
UPDATE hodimlar SET oylik = oylik * 1.1 WHERE kasbi = 'Menejer';
UPDATE hodimlar SET oylik = oylik * 1.05 WHERE kasbi = 'Analizchi';
UPDATE hodimlar SET oylik = oylik * 1.08 WHERE kasbi = 'Dasturchi';
COMMIT;
```

Yuqoridagi tranzaksiya kodini Postgre SQL dasturida yozamiz.



12.4-rasm. Postgre SQL dasturida yozilgan tranzaksiya

Natijada barcha hodimlarning oyligi oshganligini ko‘rishimiz mumkin:



12.5-rasm. Tranzaksiya natijasi

Tranzaksiyaning muhimligi:

Agar yuqoridagi operatsiyalardan birortasi xato bilan tugasa, barcha o‘zgarishlarni **ROLLBACK** yordamida bekor qilish mumkin, bu esa **ma’lumotlar yaxlitligini** ta’minlaydi. Bu tranzaksiyaning ahamiyatini ko‘rsatadi, agar biror qadam

muvaffaqiyatsiz bajarilsa, bazaning holati oldingidek qoladi va noto‘g‘ri ma’lumotlar kiritilmaydi.

12.4. Triggerlar va ulardan foydalanish

Triggerlar (Triggers) — bu ma’lumotlar bazasida avtomatik ravishda ishga tushadigan maxsus dasturlar yoki buyruqlar to‘plamidir. **Triggerlar** ma’lumotlar bazasining **strukturali va ma’lumotlar yaxlitligini** saqlashga yordam beradi. Triggerlar ma’lumotlar bazasidagi jadvalga ma’lumot qo‘shilganda(INSERT), o‘zgartirilganda (UPDATE) yoki o‘chirilganda (DELETE) avtomatik ishga tushadi va belgilangan qoidalar yoki amallarni bajaradi.

Triggerning sintaksisi (SQL):

```
CREATE TRIGGER trigger_name
{BEFORE | AFTER} {INSERT | UPDATE | DELETE} ON table_name
FOR EACH ROW
BEGIN
  -- Trigger amallari
END;
```

Triggerning sintaksisi (PostgreSQL):

```
CREATE TRIGGER trigger_name
 {BEFORE | AFTER} {INSERT | UPDATE | DELETE} ON table_name
FOR EACH ROW
 EXECUTE FUNCTION function_name();
```

Izohlaymiz:

1. **{BEFORE | AFTER}**: Bu triggerning qachon ishlashini belgilaydi.
   BEFORE: Ma’lumotlar bazasiga yozilishdan oldin triggerni ishga tushiradi.
   AFTER: Ma’lumotlar bazasiga yozilishdan so‘ng triggerni ishga tushiradi.
2. **{INSERT | UPDATE | DELETE}**: Trigger qaysi operatsiyaga javoban ishlashini belgilaydi.

INSERT: Yangi yozuv qo‘shilganda.

UPDATE: Yozuv yangilanganida.

DELETE: Yozuv o‘chirilganida.

3. **FOR EACH ROW**: Trigger har bir qator uchun ishlaydi. Har bir qatorga tegishli bo‘lgan o'zgarishlarni NEW va OLD prefikslarini ishlatib olish mumkin.

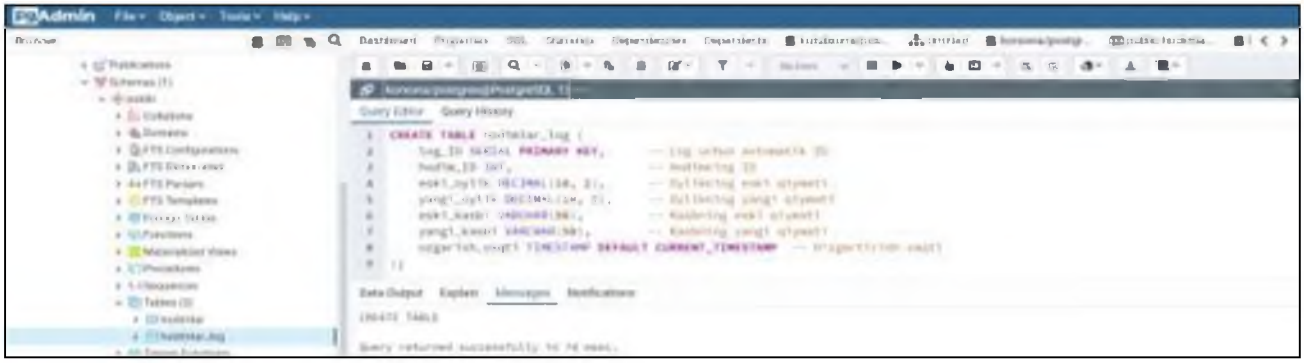
   NEW: Yangi qiymatlarni ifodalaydi (masalan, NEW.column_name).

   OLD: Eski qiymatlarni ifodalaydi (masalan, OLD.column_name).

4. **EXECUTE FUNCTION function_name();**: Bu yerda trigger tomonidan chaqiriladigan funksiya ko‘rsatiladi. Trigger ishlaganda, shu funksiya bajariladi.

Misol. Yuqorida hodimlar jadvali yaratilgan. Triggerlarni hodimlar jadvaliga tadbiq etgan holda misollar bilan tushuntirib boramiz. Oylik yoki kasbi o‘zgartirilgan hodimlar haqidagi ma’lumotlarni loglash uchun hodimlar_log jadvalini yaratamiz:

Jadvalni postgreSQL da yaratib olamiz:



12.6-jadval. PostgreSQl muhida hodimlar_log jadvalini yaratish

Loglash (inglizcha **logging**) — bu tizimning yoki dasturiy ta’minotning ichki ishlash jarayonlarini va foydalanuvchi amallarini qayd etish jarayonidir. Ma’lumotlar bazasida loglash asosan tizimdagi o‘zgarishlar, xatoliklar, foydalanuvchi harakatlari va boshqa muhim hodisalarni yozib borish uchun ishlatiladi.

Triggerlarning asosiy turlari:

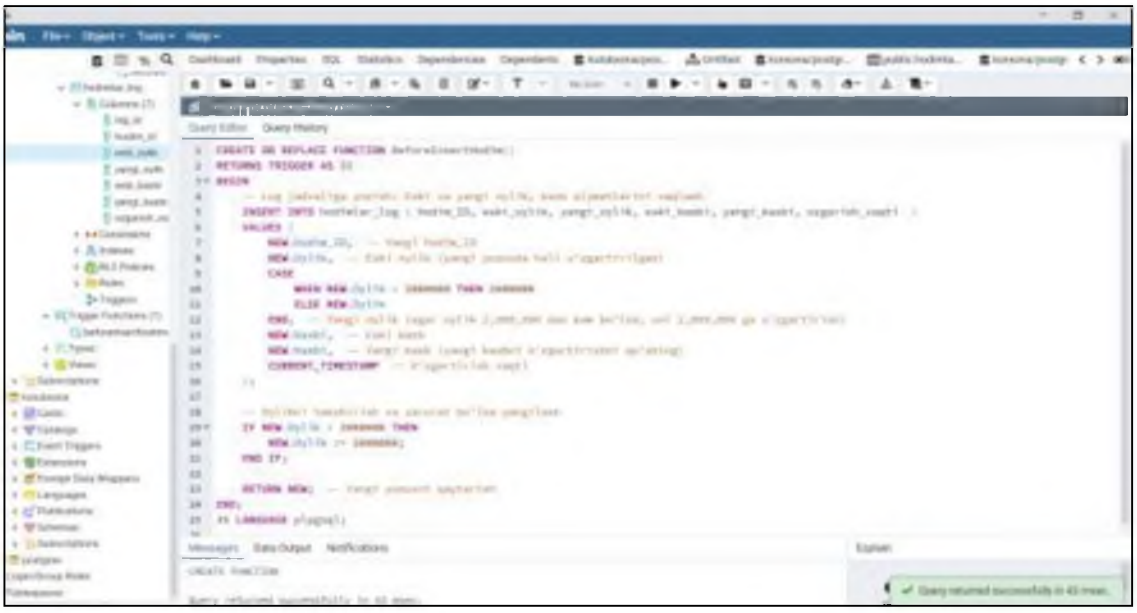
Odatda triggerlar quyidagi asosiy turlarga bo‘linadi:

1. BEFORE Trigger

- Amallar (INSERT, UPDATE, DELETE) bajarilishidan oldin ishga tushadi.
- Amaldan oldin ma’lumotlarni tekshirish, o‘zgartirish yoki cheklovlarni o‘rnatish uchun ishlatiladi.

Misol. Minimal oylik 2,000,000 bo‘lishi kerak. Agar pastroq bo‘lsa, avtomatik tuzatilishi kerak.

1. Dastlab trigger funksiyasini yozib olamiz:



12.7-jadval. PostgreSQl muhida trigger funksiyasini yaratish

✓ **CREATE FUNCTION**: Yangi funksiyani yaratadi.

✓ **OR REPLACE**: Agar funksiya mavjud bo‘lsa, uni yangilaydi. Bu funksiya nomi bilan boshqa funksiya borligini tekshiradi, agar bunday funksiya mavjud bo‘lsa uni almashtiradi. Agar mavjud bo'lmasa, yangi funksiya yaratadi.

✓ **RETURNS TRIGGER AS $$:**

RETURNS: Bu so‘z funksiya yoki triggerning qaytaradigan natijasining turini belgilaydi. Funksiya yoki trigger qaysi turdagi ma’lumotni qaytarishini ko‘rsatadi.

TRIGGER: bu qaytariladigan tur (return type) ni bildiradi.

Triggerlar ma'lum bir voqea (INSERT, UPDATE, DELETE) sodir bo'lganda ishlaydi va ular yangi yoki eskicha qatorni (yangi yoki eski yozuvni) qaytaradi.

RETURNS TRIGGER trigger funksiyasining tipini belgilaydi, ya’ni bu funksiya **TRIGGER** turida qiymat qaytaradi.

AS $$ — Bu qism funksiya yoki trigger tanasini boshlash uchun ishlatiladi.

$$ — bu **dolar qavslar** yoki delimitatorlardir, ular SQL bloklarini ajratib turish uchun ishlatiladi. Bu belgilar orasida funksiya yoki triggerning kodlari yoziladi.

✓ **$$ LANGUAGE plpgsql**; — Bu PostgreSQLdagi funksiya yaratishning yakuniy qismidir.

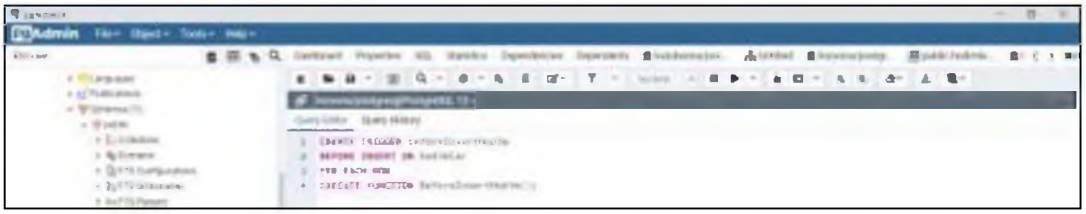
LANGUAGE plpgsql:

LANGUAGE — Bu SQL so‘rovi va funksiyalarni yozishda ishlatiladigan tilni belgilash uchun ishlatiladi.

plpgsql — Bu PostgreSQLning maxsus tilidir, ya’ni **Procedural Language** (PL/pgSQL). **PL/pgSQL** — bu PostgreSQL ma’lumotlar bazasida protsedural kod yozish uchun ishlatiladigan tildir.

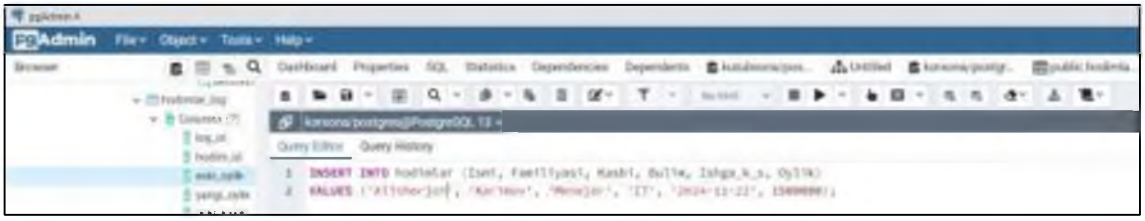
plpgsql yordamida biz SQL so‘rovlarini, o‘zgaruvchilarni, shartli konstruktsiyalarni (masalan, IF), sikllarni (masalan, LOOP) va boshqa jarayonlarni yozish imkoniyatiga ega bo‘lamiz.

2. **Trigerni yaratamiz** va yuqoridagi trigger funksiyaga murajat qilamiz:



12.8-jadval. PostgreSQl muhida trigger yaratish

3. **Triggerni tekshirish.** Triggerni yaratganingizdan so‘ng, triggerning to‘g‘ri ishlashini tekshirish uchun yangi yozuv kiritishingiz kerak. Masalan:



12.9-jadval. PostgreSQl muhida triggerni tekshirish

4. Natijalarni tekshirish. Kiritilgan yozuvni tekshirish uchun quyidagi **SELECT** so‘rovini ishlatishingiz mumkin:

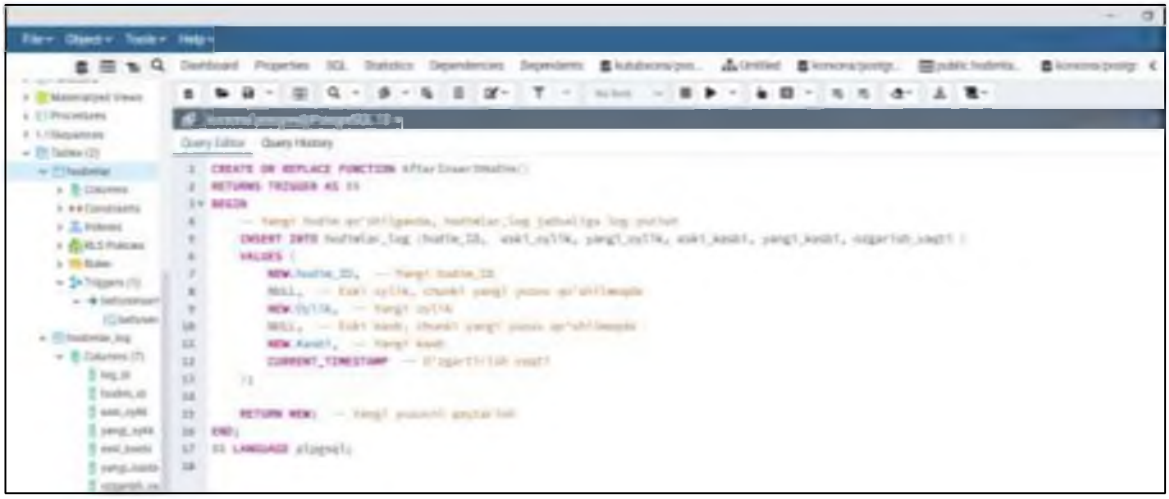
12.10-jadval. PostgreSQl muhida trigger natijalarini tekshirish

AFTER trigger SQLda ma’lum bir amal (INSERT, UPDATE, DELETE) bajarilgandan keyin ishlaydigan triggerdir. Bu trigger yordamida siz biron bir amal o‘tkazilgandan so‘ng ma’lumotlar bilan ishlash, masalan, log yozish yoki boshqa jadvalga ma’lumot kiritish kabi operatsiyalarni bajarishingiz mumkin. Amal bajarilgandan so‘ng ishga tushadi. Masalan, ma’lumotlar bazasida yangi ma’lumot qo‘shilganidan keyin boshqa jadvalga yangilanish kiritish uchun ishlatiladi.

Misol. Hodimlar jadvali uchun AFTER INSERT triggerini yarataylik, bu trigger yangi yozuvlar qo‘shilganidan so‘ng **hodimlar_log** jadvaliga o‘zgarishlarni loglashni amalga oshiradi.

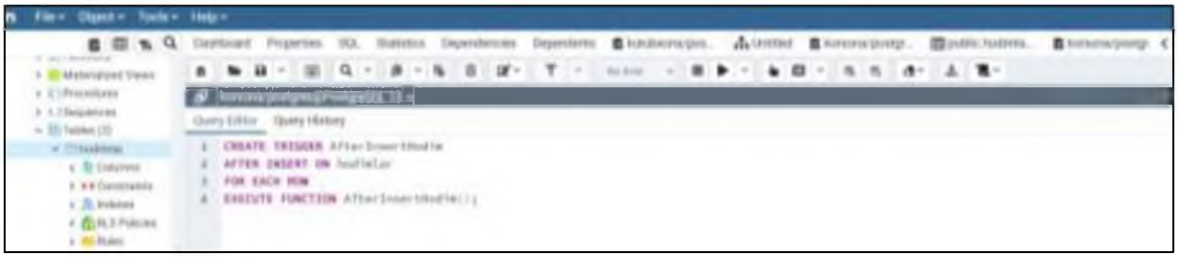
1. AFTER INSERT Trigger funksiyasini yaratish

Trigger, yangi hodim qo‘shilgandan so‘ng, **hodimlar_log** jadvaliga eski_oylik va yangi_oylik qiymatlarini log qilish uchun quyidagicha yaratiladi.



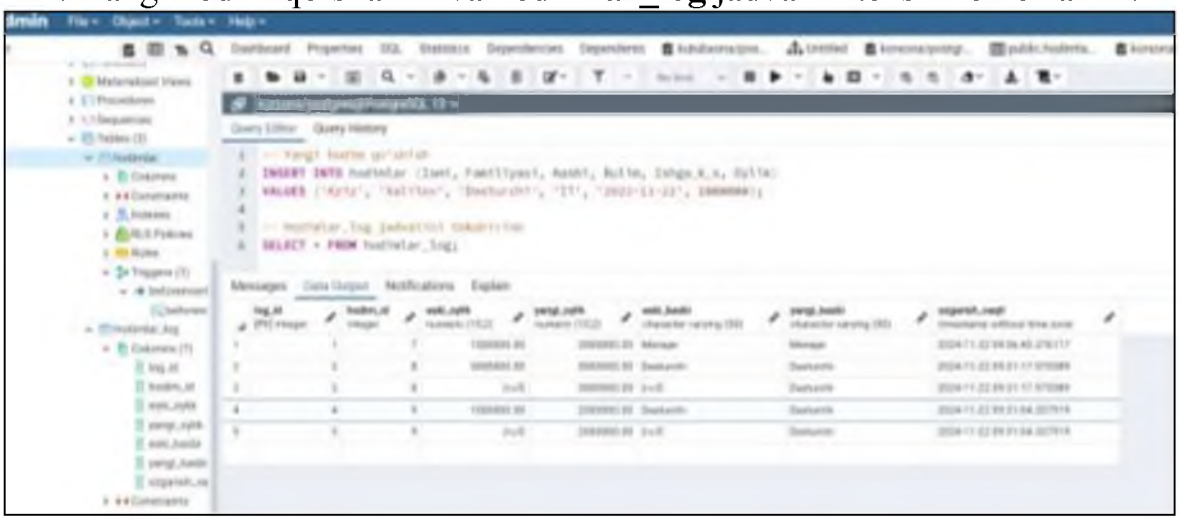
12.10-jadval. PostgreSQl muhida trigger funksiyasini yaratish

2. Keyin **AFTER INSERT** triggerini yaratamiz, bu trigger har safar yangi yozuv qo‘shilganidan so‘ng **hodimlar_log** jadvaliga ma’lumotlarni qo‘shadi.



12.11-jadval. PostgreSQl muhida trigger yaratish

3. Yangi hodim qo‘shamiz va **hodimlar_log** jadvalini tekshirib ko‘ramiz:



12.12-jadval. PostgreSQl muhida triggerni test qilish

Trigerni o‘chirish:

```
DROP TRIGGER IF EXISTS BeforeInsertHodim ON hodimlar;
```

Triggerlar qayerda ishlatiladi?

- ✓ Ma'lumotlar yaxlitligini ta'minlash: Ma'lumotlar bazasida noto'g'ri yoki keraksiz ma'lumotlarning kiritilishiga yo'l qo'ymaslik.
- ✓ Auditing va monitoring: Ma'lumotlar bazasida qaysi amallar amalga oshirilganini tekshirish.
- ✓ Ma'lumotlar uzatish: Ma'lumotlar bazasining bir jadvalidan boshqasiga avtomatik ravishda ma'lumot uzatish.
- ✓ Hisoblash: Dasturda hisoblashlarni avtomatlashtirish.

Triggerlar yordamida bajarilishi mumkin bo'lgan vazifalar:

Avtomatik hisoblash: Masalan, talabalar jadvallariga yangi ma'lumotlar qo'shilganda, avtomatik tarzda ballarni hisoblash.

Audit qilish: Ma'lumotlar bazasida har qanday o'zgarishlarni qayd etish va tekshirish.

Ma'lumotlar bazasi cheklovlarini yaratish: Masalan, biror jadvalda faqat belgilangan shartlarga mos keladigan qiymatlar kiritilishini ta'minlash.

Accessda triggerlar yo'q, ammo **PostgreSQL**, **SQL Server** yoki **MySQL**, **Oracle** kabi tizimlarda triggerlarni yaratish va ishlatish mumkin.

Demak, Triggerlar ma'lumotlar bazasining yaxlitligini ta'minlash, audit qilish, va avtomatik tarzda ma'lumotlar bazasida voqealar sodir bo'lganida turli amallarni bajarish imkoniyatini beradi. Bu asosan katta ma'lumotlar bazalarida ma'lumotlarni boshqarishni osonlashtirish va xavfsizligini ta'minlash uchun ishlatiladi.

Triggerlardan foydalanishning afzalliklari:

- ✓ Ma'lumotlar bazasidagi avtomatik tekshiruvlarni tashkil qilish.
- ✓ Yaxlitlikni ta'minlash.
- ✓ O'zgarishlarni nazorat qilish va audit qilish.

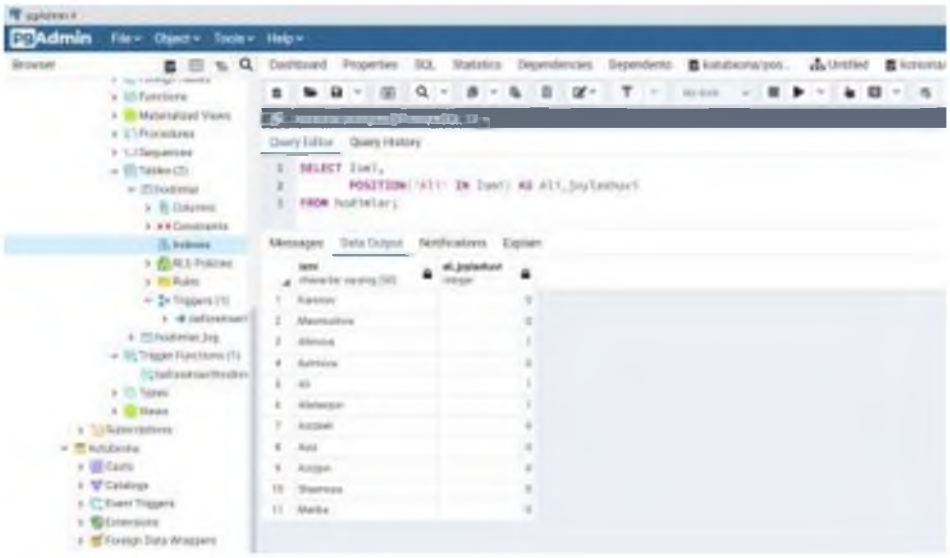
12.5. POSITION() funksiyasidan foydalanib pastki satrni qidirish

SQL da POSITION() funksiyasi belgilar satrida boshqa belgilarni qidirish uchun ishlatiladi. Bu funksiya satrdagi biror belgining pozitsiyasini qaytaradi.

Sintaksisi:

```
POSITION(substring IN string)
```

Misol: **Hodimlar** jadvalidagi **Ismi** ustunida "Ali" so‘zi qayerda joylashganini ko‘rsatadigan so’rov tuzing.



12.13-jadval. PostgreSQl muhida position() funksiyasi

Natijada, agar "Ali" so‘zi ismda mavjud bo‘lsa, 1 ni qaytaradi, agar "Ali" so‘zi mavjud bo‘lmasa, natija 0 bo‘ladi.

POSITION() funksiyasi shuningdek, so‘zdagi yoki satrdagi belgilarning holatini aniqlash uchun ishlatiladi, bu esa pastki satrni qidirish uchun foydalidir.

12.6. Joriy sessiyasi. CASE ifodasini ishlatib shartli qiymatlarni ifodalash

Joriy sessiya SQLda har bir foydalanuvchining alohida ish jarayonini anglatadi. Joriy sessiya, masalan, ma’lumotlar bazasiga kirgan vaqtdan boshlab bajarilayotgan barcha operatsiyalarni o‘z ichiga oladi. Sessiya davomida bajarilgan amallar va qiymatlar faqat ushbu sessiya uchun amal qiladi. Joriy sessiya ko‘pincha tranzaksiyalarni boshqarish va qo‘llab-quvvatlashda ishlatiladi.

Misol:

```
SELECT SESSION_USER;
```

Bu so‘rov joriy sessiyada kim tomonidan ishlashini ko‘rsatadi.

SQL da **CASE** ifodasi shartli mantiqni ifodalashda ishlatiladi. Bu ifoda, ma’lum shartlarga qarab bir nechta qiymatlarni qaytarishga imkon beradi.

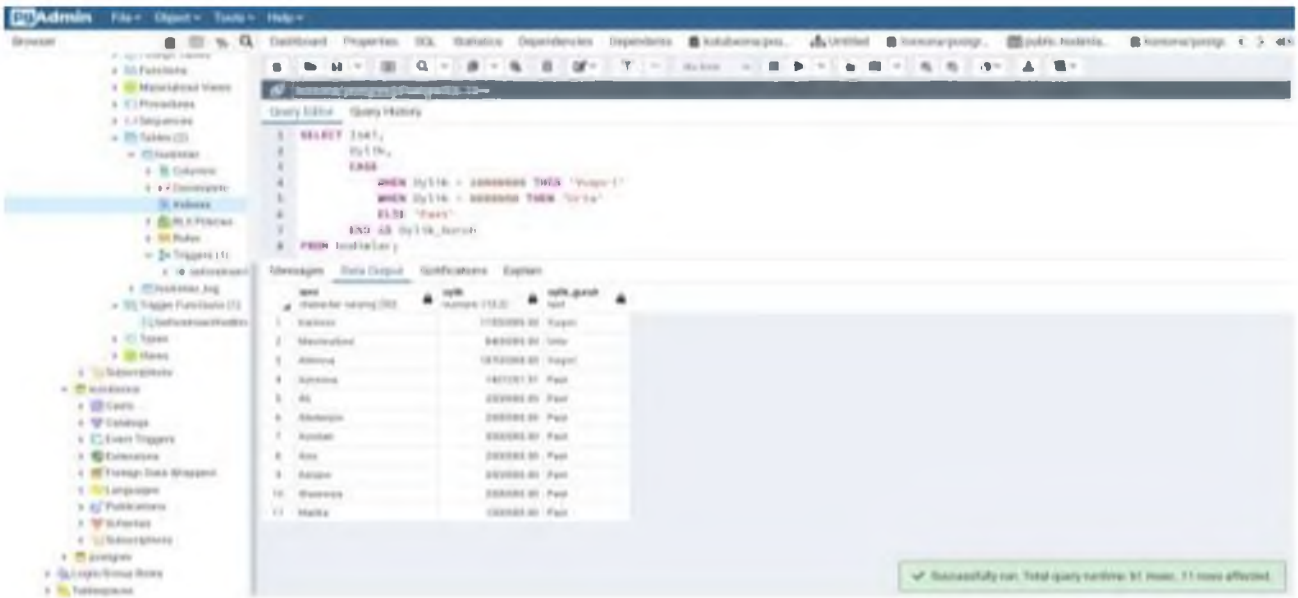
CASE ifodasini ishlatib shartli qiymatlarni ifodalash sintaksisi:

```
CASE
WHEN condition1 THEN result1
WHEN condition2 THEN result2
ELSE result3
END
```

WHEN condition: Bu yerda shartni ifodalaydi. Agar shart rost bo‘lsa, tegishli **THEN** qiymati qaytariladi.

ELSE result: Agar hech bir **WHEN** shartiga to‘g‘ri kelmasa, **ELSE** qismidagi natija qaytariladi.

Misol: Hodimlar jadvalida hodimlarning oylik maoshiga qarab ularni guruhlaylik. Agar oylik 10000000 dan yuqori bo‘lsa, **“Yuqori”**, 8000000 dan yuqori bo‘lsa **“O‘rta”** va 8000000 dan past bo‘lsa **“Past”** deb guruhlasin.



12.14-jadval. PostgreSQl muhida case ifodasi

Bu yerda CASE ifodasi ishchilarning oyligiga qarab “Low Salary”, “Medium Salary”, yoki “High Salary” kabi kategoriya qiymatlarini qaytaradi.

CASE ifodasi SQL da shartli mantiqni amalga oshirishda ishlatiladi va ko‘plab maqsadlar uchun foydalidir, masalan:

Qarorlar qabul qilish.

Shartli qiymatlarni aniqlash.

Ma’lumotlarni tahlil qilish va tasniflash.

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 BOB: MA'LUMOTLAR BAZASINI ADMINISTRATORLASH VA XAVFSIZLIGINI TA'MINLASH

Tayanch so'zlar: *administratorlash, ACL, Encryption, protsedura*

13.1. SQL serverda ma'lumotlar bazalari obyektlari himoyasi

SQL Serverda ma'lumotlar bazasi obyektlari himoyasi (yoki "obyektlarni himoya qilish") deganda, ma'lumotlar bazasidagi ma'lumotlar va obyektlarning maxfiyligi, yaxlitligi va mavjudligini saqlash uchun turli xavfsizlik choralarini qo'llash tushuniladi. Bu himoya choralariga foydalanuvchi huquqlari va rollarni boshqarish, ma'lumotlarni shifrlash, auditi, xavfsizlik siyosatlari va boshqa xavfsizlik yondashuvlari kiradi.

SQL Serverda ma'lumotlar bazasi obyektlari himoyasining asosiy elementlari quyidagilardan iborat:

1. **Foydalanuvchilar va rollar boshqaruvi**: Har bir foydalanuvchiga kerakli darajada huquqlar berish orqali SQL Serverda mavjud ma'lumotlarga kirishni boshqarish. Bu yerda foydalanuvchilar uchun alohida yoki umumiy rollar o'rnatilib, ma'lumotlar bazasidagi obyektlarga ruxsat darajasi belgilanadi. Bu esa foydalanuvchilarni cheklangan huquqlarga ega qilish orqali ma'lumotlarning himoyasini ta'minlaydi.
2. **Obyektlarga kirish huquqlarini boshqarish (ACL)**: Access Control List (ACL) yordamida har bir obyektga kirish huquqi belgilanadi. Har bir foydalanuvchi yoki guruhga o'ziga xos huquqlar o'rnatiladi, masalan, o'qish (SELECT), yozish (INSERT, UPDATE), o'chirish (DELETE) va boshqalar. Bu yondashuv orqali obyektlarga ruxsatsiz kirishni cheklash mumkin.
3. **Ma'lumotlarni shifrlash (Encryption)**: SQL Server shifrlash orqali ma'lumotlarni himoyalash imkonini beradi. Bu yondashuvda ma'lumotlar va ularning uzatish kanallari shifrlanadi, shunday qilib, ularni faqat ruxsat etilgan foydalanuvchilar ko'ra oladi. SQL Serverda Transparent Data Encryption (TDE)

kabi shifrlash texnologiyalari mavjud bo‘lib, ular ma’lumotlarni disklarda ham himoya qiladi.

4. **SQL Server Auditlari**: SQL Server’da auditi ma’lumotlar bazasiga kim, qachon va qanday amallar bajarganini nazorat qilish imkonini beradi. Bu jarayon ma’lumotlar xavfsizligini saqlash va xavfsizlik siyosatlari buzilgan taqdirda, vaziyatni qayta tiklash uchun zaruriy choralar ko‘rishga yordam beradi. Auditlar foydalanuvchi faolligini kuzatishga va xavfsizlik siyosatini nazorat qilishga xizmat qiladi.

13.2. SQL server xisob yozuvlarini boshqarish

SQL Server hisob yozuvlarini boshqarish — bu ma’lumotlar bazasiga kirishni boshqarish uchun foydalanuvchi va rollarga tegishli ruxsatlarni belgilash va ularni boshqarishni nazorat qilish jarayoni hisoblanadi. SQL Server hisob yozuvlarini boshqarish quyidagilarga asoslanishi mumkin:

1. **Foydalanuvchi va rollar tushunchasi**: SQL Server foydalanuvchi hisob yozuvlari (accounts) va rollar (roles) orqali boshqariladi. Har bir foydalanuvchi uchun alohida hisob yozuvi yaratiladi va unga kerakli ruxsatlar beriladi. Rollar esa foydalanuvchilar guruhlarini yaratish imkonini beradi, shunda bir xil ruxsatlarga ega foydalanuvchilarni birgalikda boshqarish osonlashadi. Rollar o‘z ichiga quyidagi asosiy turlarni oladi: **Server darajasi rollari** (masalan, sysadmin, securityadmin), **ma’lumotlar bazasi darajasi rollari** (masalan, db_owner, db_datareader) va **mavjud foydalanuvchi yarata oladigan maxsus rollar**
2. **Ruxsatlarni (Permissions) belgilash**: SQL Serverda foydalanuvchilarga ma’lumotlar bazasi obyektlariga (jadval, ko‘rinish, saqlanadigan prosedura va boshqalar) kirish huquqlari belgilanishi kerak. Ruxsatlar ob’ektlarga kirish darajasida boshqariladi va quyidagi asosiy huquqlar berilishi mumkin: o‘qish (SELECT), yozish (INSERT, UPDATE, DELETE) va ma’lumotlarni bajarish

(EXECUTE). Ushbu ruxsatlarni to‘g‘ri o‘rnatish ma’lumotlarning xavfsizligi va maxfiyligini ta’minlashda muhim hisoblanadi.

3. **Audit va Monitoring**: SQL Server foydalanuvchi harakatlarini qayd qilish va monitoring qilish imkoniyatiga ega. Audit xususiyatlari orqali administratorlar foydalanuvchilarning ma’lumotlar bazasida qanday o‘zgarishlar qilganini kuzatishi mumkin. Masalan, **SQL Server Audit** va **Change Data Capture** kabi funksiyalar mavjud bo‘lib, ular ma’lumotlarni o‘zgartirish va unga kirish holatlarini qayd etib boradi.
4. **Ma’lumotlarni Shifrlash va Himoya Qiluvchi Funksiyalar**: SQL Serverda “Transparent Data Encryption” (TDE) va *Always Encrypted* kabi shifrlash usullari mavjud bo‘lib, ular ma’lumotlarni shifrlab, faqat tegishli huquqlarga ega foydalanuvchilargina ko‘ra olishiga imkon beradi. Bu ma’lumotlarni shifrlash orqali ruxsatsiz kirishning oldini olishda yordam beradi.

Ma’lumotlar bazasini administratori (DBA[1]) — bu ma’lumotlar bazasi tizimlarining xavfsizligi, samaradorligi va uzluksiz ishlashini ta’minlash uchun mas’ul bo‘lgan mutaxassis hisoblanadi. MBAning asosiy vazifalari:

Ma’lumotlar bazasi texnik ishlashini boshqarish: MBAlar ma’lumotlar bazasini yaratish, texnik xizmat ko‘rsatish, optimallashtirish va zaxira nusxalarini tayyorlash uchun javobgardir. Ular SQL buyruqlarini chuqur tushunishi va ma’lumotlarni qayta ishlashda samaradorlikni oshiradigan optimallashtirish usullarini qo‘llashi kerak;

Xavfsizlik va ruxsatlarni boshqarish: MBAlar foydalanuvchi hisoblari va ruxsatlar bilan ishlaydi, ya’ni ma’lumotlarni himoyalash uchun turli foydalanuvchi rollarini o‘rnatish va boshqarish orqali tizim xavfsizligini ta’minlaydi. MBAlar SQL Server va boshqa tizimlarda foydalanuvchilarga alohida kirish huquqlarini taqdim etish orqali xavfsizlik darajasini boshqaradilar.

[1] DBA – database administration

Monitoring va Audit: MBAlar tizimga yuklangan vazifalarni kuzatadi va ma'lumotlar bazasi ishlashini tahlil qiladi. Ma'lumotlar o'zgarishlarini monitoring qilish va qaydlarni tahlil qilish ma'lumotlar xavfsizligi va tizim samaradorligini oshirishga yordam beradi. Monitoring vositalari yordamida tizimning resurs iste'molini nazorat qilish va kerakli o'zgarishlarni amalga oshirish DBAning mas'uliyatiga kiradi.

Zaxira va Qayta tiklash: DBAlar ma'lumotlar bazasi uchun zaxira nusxalarini tayyorlaydi va halokat sodir bo'lgan taqdirda ma'lumotlarni tezkor qayta tiklash imkoniyatini ta'minlaydi. Zaxira nusxalari va avariya tiklash jarayoni tizimning ishonchliligini ta'minlash uchun juda muhim hisoblanadi.

13.3. Proseduralar va ularni yaratish

Proseduralar — ma'lumotlar bazasida bajarilishi kerak bo'lgan bir qator SQL buyruqlarini yagona bir birlik sifatida tashkil qilishga yordam beradigan dasturiy obyektlardir. Proseduralar, odatda, ko'p marta bajarilishi kerak bo'lgan murakkab operatsiyalarni avtomatlashtirish va ma'lumotlar bazasini boshqarish jarayonini soddalashtirish uchun yaratiladi.

Proseduralar ma'lumotlar bazasida saqlanadigan va kerak bo'lganda qayta ishlatiladigan buyruqlar to'plamidir. Prosedura yaratish uchun odatda CREATE PROCEDURE buyrug'i ishlatiladi. Misol uchun, agar bizda EMPLOYEE jadvali bo'lib, har bir xodimning ma'lumotlarini o'z ichiga olsa, quyidagi prosedura xodimlarning oylik maoshlarini hisoblash uchun ishlatilishi mumkin:

```
CREATE PROCEDURE CalculateSalary
AS
BEGIN
UPDATE Employee
SET Salary = BaseSalary * BonusPercentage
WHERE Active = 1;
END;
```

Yuqoridagi kodda CalculateSalary nomli prosedura yaratilgan. Bu prosedura faqat faol xodimlarning oylik maoshlarini yangilaydi. Bu tarzda proseduralar ma'lumotlarni avtomatlashtirishda qo'l keladi va murakkab amaliyotlarni qayta-qayta bajarish imkonini beradi.

2. Proseduranı ishga tushirish

Yaratilgan prosedurani bajarish uchun EXEC yoki EXECUTE buyrug'i ishlatiladi. Misol:

```
EXEC CalculateSalary;
```

Bu buyruq CalculateSalary proseduraSini ishga tushiradi va EMPLOYEE jadvalidagi xodimlarning maoshlarini belgilangan formula asosida yangilaydi.

3. Parametrli prosedura yaratish

Ko'pincha proseduralarni yanada moslashuvchan qilish uchun ularga parametrlar qo'shiladi. Quyidagi misolda, ma'lum bir xodimning ID raqamini kiritish orqali faqat shu xodimning maoshini yangilaydigan prosedura yaratiladi:

```
CREATE PROCEDURE UpdateEmployeeSalary
@EmployeeID INT,
@NewSalary DECIMAL(10, 2)
AS
BEGIN
   UPDATE Employee
   SET Salary = @NewSalary
   WHERE EmployeeID = @EmployeeID;
END;
```

Yuqoridagi prosedura @EmployeeID va @NewSalary parametrlarini qabul qiladi. Bu xususiyat orqali prosedura aniq xodim ma'lumotlarini yangilash uchun ishlatilishi mumkin. Proseduranı chaqirish esa quyidagicha amalga oshiriladi:

```
EXEC UpdateEmployeeSalary @EmployeeID = 101, @NewSalary = 5000.00;
```

Bu buyruq EmployeeID qiymati 101 ga teng bo‘lgan xodimning maoshini 5000.00 ga o‘zgartiradi.

4. Proseduralar orqali ma’lumotlarni qaytarish

Proseduralar SELECT bayonoti yordamida ma’lumotlarni qaytarishi mumkin. Masalan, quyidagi prosedura ma’lum bir bo‘limdagi barcha xodimlar ro‘yxatini qaytaradi:

```
CREATE PROCEDURE GetEmployeesByDepartment
    @DepartmentID INT
AS
BEGIN
    SELECT *
    FROM Employee
    WHERE DepartmentID = @DepartmentID;
END;
```

Bu prosedura bo‘lim ID sini qabul qilib, shu bo‘limdagi barcha xodimlarni qaytaradi. Masalan:

```
EXEC GetEmployeesByDepartment @DepartmentID = 2;
```

Proseduralarning afzalliklari:

Takrorlanuvchi vazifalarni avtomatlashtirish: Proseduralar yordamida murakkab yoki takrorlanuvchi operatsiyalarni bitta buyruq sifatida bajarsa bo‘ladi.

- **Xavfsizlik va boshqaruv**: Ma’lumotlar bazasidagi operatsiyalarni yanada ko‘proq nazorat qilish imkoniyatini beradi. Foydalanuvchilarga proseduraga ruxsat berish orqali ularni to‘g‘ridan-to‘g‘ri ma’lumotlar bazasiga kirishdan cheklash mumkin.
- **Samaradorlikni oshirish**: Proseduralar bir marta kompilyatsiya qilinadi, shuning uchun ular tezroq ishlaydi va tizim resurslarini samarali foydalanadi.

13.4. Ma’lumot bazasini administratori

Ma’lumot bazasini administrator (DBA) — bu malumotlar bazasini boshqarish, saqlash, xavfsizlikni ta’minlash va uning samarali ishlashini nazorat qilish uchun mas’ul bo‘lgan mutaxassis. Ma’lumotlar bazasi administratori (DBA) tashkilotda ma’lumotlar bazasining barcha texnik jihatlarini boshqaradi, shu jumladan uning dizayni, konfiguratsiyasi, ko‘p foydalanuvchi tizimlarida ishlashni ta’minlash va umumiy tizim xavfsizligini ta’minlash.

Ma’lumotlar bazasi administratori quyidagi asosiy vazifalarni bajaradi:

1. Ma’lumotlar bazasini o‘rnatish va konfiguratsiya qilish

Ma’lumotlar bazasi dasturini o‘rnatish va kerakli parametrlar bo‘yicha sozlash.

Tizim uchun kerakli resurslarni (xotira, disk, tarmoqlar) ajratish va optimallashtirish.

2. Zaxira nusxalarini yaratish va tiklash

Ma’lumotlarning yo‘qolishini oldini olish uchun ma’lumotlar bazasining muntazam zaxira nusxalarini yaratish.

Favqulodda holatlarda zaxira nusxasidan ma’lumotlarni qayta tiklash jarayonini boshqarish.

3. Ma’lumotlar xavfsizligini ta’minlash

Ma’lumotlar bazasiga kirishni nazorat qilish, foydalanuvchilarni autentifikatsiyadan o‘tkazish va ularning ruxsatlarini belgilash.

Maxfiy ma’lumotlarni himoya qilish va xavfsizlik siyosatlarini amalga oshirish.

4. Ma’lumotlar bazasi monitoringi va samaradorligini oshirish

Ma’lumotlar bazasining ish faoliyatini kuzatish, sekin ishlash sabablarini aniqlash.

Ishlash samaradorligini oshirish uchun indekslar, kechiktirilgan qidiruvlar (query caching) yoki boshqa optimallashtirish usullarini qo‘llash.

5. Muammolarni aniqlash va bartaraf etish

Xatolarni va nosozliklarni aniqlash, log fayllarini tahlil qilish.

Texnik muammolarni hal qilish va tizimning uzluksiz ishlashini ta’minlash.

6. Yangilash va texnik xizmat ko‘rsatish

Ma’lumotlar bazasi dasturini muntazam ravishda yangilash, yangi versiyalarni o‘rnatish. Tizimning har doim oxirgi xavfsizlik va funksional talablariga mos kelishini ta’minlash.

7. Ma’lumotlarni zaxiralash va arxivlash

Kamdan-kam foydalaniladigan yoki eski ma’lumotlarni arxivlash va asosiy ma’lumotlar bazasidan ajratib olish.

Bu ma’lumotlar bazasining tezligini oshirish va joyni tejash imkonini beradi.

8. Foydalanuvchilarni qo‘llab-quvvatlash va foydalanuvchi ruxsatlarini boshqarish.

Ma’lumotlar bazasi foydalanuvchilarini qo‘llab-quvvatlash, ularga yordam berish va kerakli ruxsat darajalarini belgilash.

Bunda maxsus huquqlar asosida har bir foydalanuvchining ma’lumotlar bazasidan foydalanish imkoniyatlari boshqariladi.

13.5. Ma’lumotlar bazasini loyihalash, uzatish va samaradorligini oshirish

Ma’lumotlar bazasini loyihalash: Ma’lumotlar bazasini loyihalash jarayoni ma’lumotlarni samarali tashkil qilish, saqlash va boshqarishga qaratilgan. Bu jarayonda ma’lumotlar normalizatsiyasi, bog‘liqliklar, jadval tuzilmalari va indekslar yaratiladi. Maqsad, ma’lumotlarni saqlash va qidirishni samarali va tez amalga oshirishdir.

Ma’lumotlarni uzatish (Migratsiya): Ma’lumotlar bazasini uzatish jarayoni mavjud tizimdan yangi tizimga ma’lumotlarni ko‘chirishni nazarda tutadi. Bu jarayonda ma’lumotlarning yaxlitligi, aniqligi va xavfsizligini saqlash juda muhimdir. Ma’lumotlarni uzatishda qiyinchiliklarga yo‘l qo‘ymaslik uchun ehtiyotkorlik bilan rejalashtirish talab etiladi.

Samaradorlikni oshirish: Ma’lumotlar bazasi samaradorligini oshirish uchun indekslash, keshlash, shuningdek, so‘rovlarni optimallashtirish kabi texnikalar qo‘llanadi. Samaradorlikni oshirish ma’lumotlar bazasiga tushadigan yukni kamaytirib, qidiruv jarayonini tezlashtiradi.

Nazorat savollari:

1. Prosedura nima?
2. ma’lumotlar bazalari obyektlari himoyasi nima?
3. xisob yozuvlarini boshqarish nima?
4. Ma’lumot bazasini administratori vazifalari?
5. Ma’lumotlar bazasini loyihalash, uzatish va samaradorligini oshirish haqida ma’lumot bering.
6. Prosedura qanday ishga tushiriladi?

14 BOB. MA'LUMOTLAR BAZASIGA MUROJAATNI TASHKIL ETISHDA ODBC VA TURLI DASTURLARDAN FOYDALANISH

Tayanch so'zlar: *ODBC, C++ Builder, Adotable*

14.1. Ma'lumotlar bazasiga murojaatni tashkil etishda C# dasturi

ODBC (Open Database Connectivity) — bu turli xil ma'lumotlar bazalariga ulanish va ularga murojaat qilishni ta'minlaydigan standart dasturiy interfeys. ODBC Microsoft tomonidan ishlab chiqilgan bo'lib, bu orqali dastur va ma'lumotlar bazasi o'rtasida muvofiqlikni ta'minlash imkoniyati yaratiladi. ODBC yordamida dasturlar bir xil ma'lumotlar bazasi buyruqlarini ishlatib, turli xil ma'lumotlar bazasi boshqaruv tizimlari (DBMS) bilan muloqot qilish imkoniyatiga ega bo'ladi, masalan, MySQL, SQL Server, Oracle va boshqalar bilan.

ODBC interfeysining asosiy vazifalari quyidagilar:

- Dasturlar bir xil ODBC API orqali turli DBMSlarga ulanishi mumkin, shu bilan ularning har biri uchun alohida kod yozishni talab qilmaydi.
- ODBC turli dasturiy platformalarda ma'lumotlarni boshqarishga yordam beradi va o'zaro ishlashni soddalashtiradi.
- ODBC SQL buyruqlari orqali ma'lumotlarni olish va manipulyatsiya qilish imkonini beradi.

C# dasturlash tilidan foydalanib, ma'lumotlar bazalari bilan ishlashni, ularni yaratish, o'qish, yangilash va o'chirish kabi operatsiyalarni bajarishni o'z ichiga oladi. C# dasturida ma'lumotlar bazasiga murojaat qilish uchun asosan **ADO.NET**, **Entity Framework**, va **LINQ to SQL** kabi vositalar ishlatiladi.

1. C# orqali ma'lumotlar bazasiga murojaat qilishning asosiy usullari

C# dasturida ma'lumotlar bazasiga murojaat qilish uchun bir nechta yondashuvlar mavjud, ammo eng keng tarqalganlari quyidagilar:

1.1. ADO.NET (ActiveX Data Objects for .NET)

ADO.NET — bu .NET platformasida ma'lumotlar bazalari bilan ishlash uchun ishlatiladigan asosiy vosita. Bu kutubxona ma'lumotlar bazasiga ulangan dasturlarni yaratish imkonini beradi. ADO.NET yordamida ma'lumotlar bazasiga murojaat qilishning asosiy usuli **SqlConnection**, **SqlCommand**, **SqlDataReader** kabi obyektlarni ishlatishdan iborat.

SqlConnection: Bu obyekt ma'lumotlar bazasi bilan ulanishni ta'minlaydi.

SqlCommand: SQL so'rovlarini bajarish uchun ishlatiladi.

SqlDataReader: Ma'lumotlar bazasidan olingan natijalarni o'qish uchun ishlatiladi.

1.2. Entity Framework (EF)

Entity Framework — bu C# uchun ORM (Object-Relational Mapping) tizimi bo'lib, u obyektlar va ma'lumotlar bazasidagi jadvallar o'rtasidagi bog'lanishni ta'minlaydi. Entity Framework yordamida SQL so'rovlarini yozish zarurati kamayadi, chunki bu tizim avtomatik ravishda obyektlar bilan ishlash uchun kerakli SQL kodlarini yaratadi.

EF yordamida ma'lumotlar bazasiga murojaat qilish uchun **DbContext** va **LINQ** so'rovlarini ishlatish mumkin.

1.3. LINQ to SQL

LINQ to SQL — bu SQL so'rovlarini C# kodidan foydalanib yozishga imkon beruvchi usuldir. Bu yondashuvda SQL so'rovlari LINQ (Language Integrated Query) orqali bajariladi. LINQ to SQL yordamida ma'lumotlar bazasiga murojaat qilish ancha osonlashadi, chunki C# kodi va SQL so'rovlari bir xil kodda ishlatiladi.

14.2. Ma'lumotlar bazasiga murojaatni tashkil etishda C++ dasturi

C++ dasturlash tili, odatda, tizim dasturlari va ishlov berish jarayonlari uchun ishlatiladi, ammo ma'lumotlar bazalari bilan ishlashda ham o'zining o'rnini topdi. C++ dasturida ma'lumotlar bazasiga murojaat qilish uchun asosan **ODBC (Open Database Connectivity)** va **MySQL Connector/C++** kabi kutubxonalar ishlatiladi.

1. C++ da ma’lumotlar bazasiga murojaat qilishning asosiy usullari

C++ dasturida ma’lumotlar bazasiga murojaat qilish uchun quyidagi usullar keng qo‘llaniladi:

1.1. ODBC (Open Database Connectivity)

ODBC — bu platformalararo ma’lumotlar bazasiga murojaat qilish uchun ochiq standarti hisoblanadi. ODBC yordamida C++ dasturlarida ma’lumotlar bazalariga ulanish va SQL so‘rovlarini bajarish mumkin. ODBC protokoli turli ma’lumotlar bazalari bilan ishlashga imkon beradi, shu jumladan SQL Server, MySQL, PostgreSQL va boshqalar.

1.2. MySQL Connector/C++

MySQL Connector/C++ — bu MySQL ma’lumotlar bazasi bilan ishlash uchun mo‘ljallangan rasmiy C++ kutubxonasi. Bu kutubxona yordamida C++ dasturida MySQL ma’lumotlar bazasiga ulanish, so‘rovlar yuborish va natijalarni olish mumkin.

14.3. ODBS C++ dasturlash tili yordamida ma’lumotlar bazasiga murojaatlarni tashkil etil usullari

C++ Builderda ma’lumotlar bazasi (MB) bilan ishlaganda dastlab Alias hosil qilinadi Alias ma’lum bir guruhga tegishli bo‘lgan jadvallarni qaysi toifaga tegishliligini (jadvallar hosil qilinadigan drayverni), jadvallar yo‘lini o‘zida saqlaydi. C++ Builderda Alias hosil qiliashning bir necha usullari mavjud.

1. Menyuning DataBase bo‘limidan SQL Explorer tanlanadi.

2. Object -»New ... [CTRL+N] tanlanadi; (yoki SQL Explorer darchasining o‘ng qismida sichqonchaning o‘ng tugmasi bosiladi va New tanlanadi)

14.1-rasm. Alias hosil qilish

3. Yuqoridagi darchadan drayver nomi tanlanadi(Standart holatda Paradox tanlanadi);

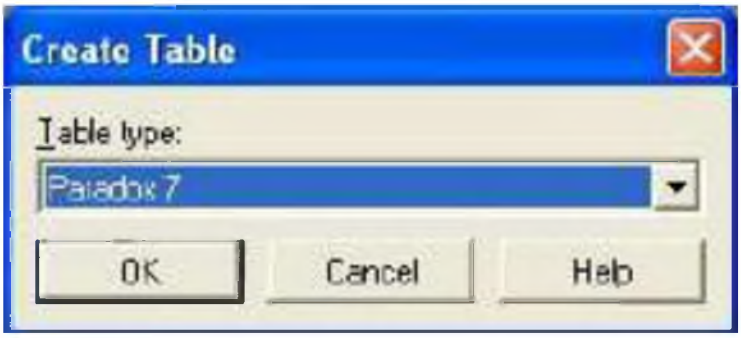
4. Path bo‘limiga jadvallar saqlanadigan katalog ko‘rsatiladi.

5. Hosil qilingan Alias ni saqlash uchun sichqonchaning o‘ng tugmasi bosiladi va Apply [CTRL+A] tanlanadi.

Jadval hosil qilish uchun quyidagi amallar ketma ketligi bajariladi.

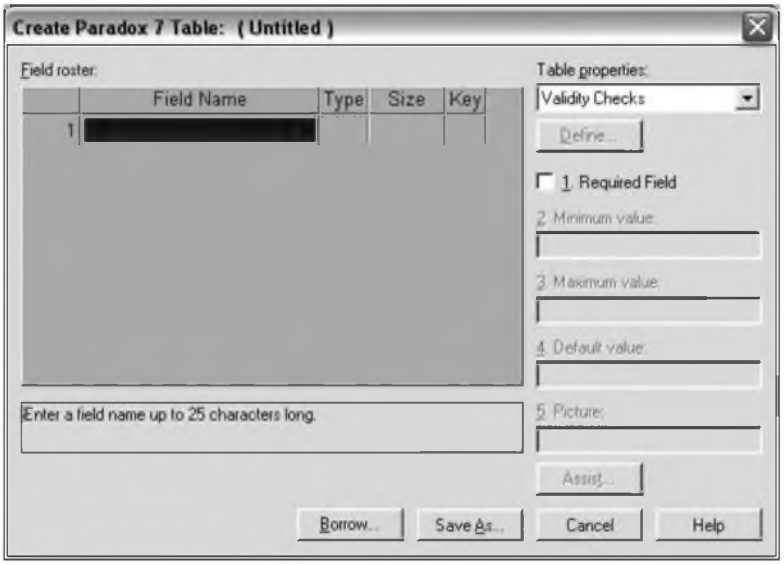
1. C++ Builder menyusining Tools bo‘limidan Database Desktop tanlanadi;

2. Database Desktop dan File—>New—>Table tanlanadi;



14.2-rasm. Jadval yaratish

3. Jadval toifasi tanlanadi;



14.3-rasm. Jadval toifasini tanlash

4. Field Name - jadval maydoni nomi; Type - maydon toifasi; Size - hajmi; Key - kalit maydonini bildirish bo‘limi;

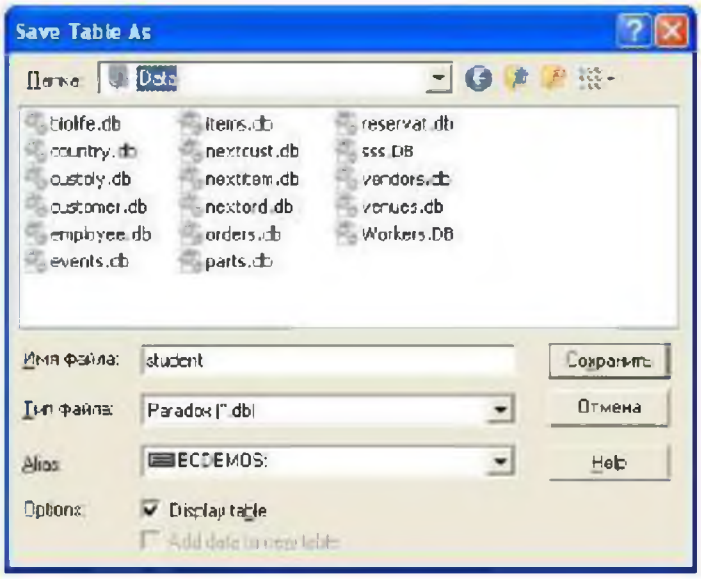
5. Jadval maydonlari kiritilgandan keyin Save As… tugmasi bosiladi;

6. Alias bo‘limidan o‘zimiz hosil qilgan alias nomi tanlanadi;

7. Jadvalga nom beriladi;

8. Options bo‘limida Display Table ga bayroqcha o‘rnatilsa, save tugmasi bosilgandan keyin Database Desktop da jadval ochiladi.

9. Bayroqcha o‘rnatilmasa (yoki ixtiyoriy jadvalni ochish uchun) File —> Open -> Table tanlanadi. Alias bo‘limidan jadval saqlangan alias tanlanadi. So‘ngra ochilichi lozim bo‘lgan jadval nomi tanlanadi va Открыт tugmasi bosiladi.



14.4-rasm. kerakli jadvalni tanlash

Maydonlar haqida ma’lumot

Maydon nomi 25 ta simvoldan iborat bo‘lishi mumkin. Birinchi simvol probel bo‘lishi mumkin emas.

Eslatma: Maydon nomini yozishda probeldan umuman foydalanmagan ma’qul. Chunki, SQL so‘rovlardan foydalanganda muammo chiqishi mumkin. Zarurat bo‘lsa, shift minus belgisidan foydalaning. Apostrovdan (‘) ham foydalanmang!

Maydon toifasini tanlash uchun Type maydoniga o‘tib, sichqonchaning o‘ng tugmasi bosiladi yoki probel bosiladi. Paradox uchun maydon toifalari quyidagicha bo‘lishi mumkin.

1. A1-255 Alpha Satrli maydon. ASCII kodini barcha simvollarda qabul qiladi.

2. N-number- 10307 -10308 butun son

3. $ - money - pul birligini bildiruvchi musbat yoki manfiy son.

4. S - short -32767 ... 32767 oralig‘idagi butun sonlar.

5. I - long integer -2147483648 ... 2147483648

6. # - 0 - 32 BCD Binary Coded Decimal formatdagi son.

7. D - Date - sanani bildiruvchi maydon.

8. T- Time - vaqt.

9. @ - Time Stamp - vaqt va sanani bildiruvchi maydon.

10. M -1-240 Memo cheklanmagan satr ma’lumotlarini saqlash uchun mo‘lallangan.

11. F - 0-240 Formatted cheklanmagan format. Memo ulangan satrni saqlash uchun mo‘ljallangan.

12. G - graphic tasvir ma’lumotlarini saqlash.

13. O - OLE tasvir, ovoz, dokument va xakozolarni saqlash.

14. L - logical mantiqiy maydon.

15. + Autoincrement avtomat ravishda yozuvni bittaga oshirib boradi. Yozuvlar o‘chirilsa oldindagilar o‘zgarishsiz qoladi.

Misol: Lesson nomli alias hosil qiling. Quyidagi maydonlarni o‘z ichiga oluvchi talaba nomli jadval hosil qiling.

14.1-jadval.Lesson jadvali xususiyatlari

№	Field Name	Type	Size
1	ID	+	
2	Familiya	A	20
3	Ism	A	20
4	Otasi	A	20
5	Guruh	A	5
6	Tug_kun	D	

Bu jadvalni C++ Builder bilan bog‘lashni ko‘rib chiqamiz.

1. File→New→Application tanlanadi;

2. Komponentalarning BDE bo‘limidan Table komponentasi formaga qo‘yiladi;

3. Table komponentasining DatabaseName xususiyatida Alias ko‘rsatiladi. (ya’ni Lesson);

4. TableName xususiyatida jadval nomi keltiriladi. (talaba);

5. Active xususiyati true ga o‘zgartiriladi;

6. Komponentalarning Data Access bo‘limidan DataSource komponentasi formaga qo‘yiladi;

7. DataSet xususiyati Table1 qilinadi;

8. Komponentalarning Data Controls bo‘limidan DBNavigator komponentasi formaga qo‘yiladi;

9. DataSource xususiyati DataSource qilinadi;

10. Komponentalarning Data Controls bo‘limidan DBGrid komponentasi formaga qo‘yiladi;

11. DataSource xususiyati DataSource qilinadi;

12. Formaga Edit va Button komponentalari qo‘yiladi Button komponentasi ustida sichqoncha ikki marta bosiladi va quyidagilar kiritiladi:

TLocateOptions qidiruv_turi; qidiruv_turi « loPartialKey « loCaseInsensitive; if (!table1->Locate("Familiya",Editl->Text,qidiruv_turi)) ShowMessage("Bunday yozuv yo‘q");

Bu yerda qidiruv_turi turidagi loPartialKey qidirilayotgan familiya qisman kiritilsa ham qidiruvni amalga ochirishni bildiradi. Masalan: Abdurahimov familiyasini qidirish lozim bo‘lsa, Abdu deb yozilsa ham qidiruv amalga oshirilishini bildiradi. loCaseInsensitive esa, qidirilayotgan ma’lumotning katta yoki kichik yozilganiga axamiyat bermaslikni bildiradi. Masalan: Abdurahimov familiyasini qidirish lozim bo‘lsa, abdurahimov deb yozilsa ham qidiruv amalga oshirilishini bildiradi.

Bu parametrlarni alohida ishlatilsa ham bo‘ladi. qidiruv_turi « loCaselnsensitive; yoki qidiruv_turi « loPartialKey;

Maydonlar bo‘yicha filtrlash

1. Formaga Button va Edit komponentalarini qo‘ying
2. Button komponentasiga quyidagilar yoziladi

```
Table1->Filtered = false;
Table1->FilterOptions << foCaselnsensitive;
Table1->Filter = "Familiya=‘ " + Editl->Text +, “*’ ”;
Table1->Filtered = true;
```

3. Table ning FilterOptions xususiyatining foCaselnsensitive xususiyati True qilinadi.

Yoki Edit komponentasining Onchange hodisasiga quyidagilar yoziladi:

```
Table1->Filtered = false;
// Editl bo‘sh bo‘Isa funksiyadan chiqib ketiladi
if (Editl->Text == "")
return;
Table1->FilterOptions « foCaselnsensitive;
Table1->Filter = "Familiya=‘ " + Editl->Text  + “ *’ ”
Table1->Filtered = true;
```

DIQQAT: Filterlashni bu turi faqat Paradox uchun o‘rinli.

Maydonlarga quyidagicha murojaat qilish mumkin.

```
Table1->FieldByName("Maydon_nomi")->Value;
Table1->FieldByName("Narxi")->AsCurrency;
Table1->FieldByName("Soni")->AsInteger;
```

Maydonga biror qiymatni o‘zlashtirish quyidagicha amalga oshiriladi:

```
Table1->Edit; // Jadvalni o‘zgartirishga ruxsat berish
Table1->FieldByName("Ism")->AsString = Editl->Text;
```

Table1->Post(); // Saqlash

14.4. SQL so‘rovlardan foydalanish, interfeysni va ma’lumotlar bazasi aloqasini ta’minlash

SQL tili (Structured Query Language - strukturalashgan so‘rovlar tili) sintaksisi juda oson. SQL tilida katta - kichik harflar farqlanilmaydi. Ya’ni Select operatorini SELECT, Select, select shakllarida yozish mumkin. Agar bir nechta operatordan foydalanilsa operator oxirida qo‘yiladi. Faqat bitta operator ishlatilsa operator oxirida qo‘yish shart emas. Izoh yozish ba’zi sistemalarda / * */ shaklida, ba’zi sistemalarda esa {} shaklida bo‘ladi.

SQL so‘rovlar tilidan deyarli barcha (MS SQL Server, MySQL, PostgreSQL, Oracle, Informix, Paradox, Interbase, FireBird) ma’lumotlar bazasida foydalanish mumkin. SQL so‘rovlari ma’lumotlar bazasiga qarab qisman o‘zgarishi mumkin.

Select tanlash operatori

Select operatorining umumiy ko‘rinishi

SELECT maydon nomlari ro‘yhati>

FORM

WHERE

GROUP BY

HAVING

ORDER BY maydon nomlari ro‘yhati>;

Quyidagi maydonlarni o‘zida saqlovchi namuna nomli jadval hosil qiling va uni Lesson aliasiga saqlang (14.1-jadval).

1. Bu jadvaldagi ma’lumotlarni ko‘rish quyidagicha bo‘ladi:

```
SELECT * FROM namuna
```

2. Jadvaldagi ba’zi maydonlarni ko‘rsh uchun shu maydon nomlari ro‘yhati select dan keyin keltiriladi:

```
SELECT Familiya, Ism, Otasi FROM namuna
```

3. Familiya bo‘yisha tartiblash

Select familiya, ism, otasi From namuna Order By Familiya, Ism

4. Tug‘ilgan yili bo‘yicha kamayib borish tartibida tartiblash

```
Select familiya, ism, otasi,tug_yil
FROM namuna
Order By tug_yil DESC
```

5. Select operatoridan keyin faqat maydon nomi emas, ixtiyoriy arifmetik amal (+, *, /) ishlatish mumkin. Misol uchun, yuqoridagi jadvaldagi tug‘ilgan yilni joriy yildan ayirib, talabaning yoshini aniqlaymiz:

```
Select familiya, ism, otasi, (2013-tug_yil) AS Yoshi FROM namuna
```

6. Agar maydon nomini ruscha shriftda chiqarish talab etilsa quyidagicha so‘rov yoziladi: (Maydon nomidan keyin AS xizmatchi so‘zi bilan maydon sarlavhasi keltirilladi. Faqat "YO" xarfida muammo bo‘lishi mumkin)

```
Select familiya AS Familiya, ism AS Imya FROM namuna
```

6.1. Familiya va Ism maydonlarini bitta ustunda chiqarish uchun quyidagicha yozish mumkin.

```
Select familiya + ‘ ‘ + ism AS "Familiya Imya"
FROM namuna
```

7. Dastlabki 6 ta amal tushunarli. Misol uchun jinsi ayol bo‘lgan, 1990 yildan keyin tug‘ilganlar ro‘yhatini chiqaramiz.

```
Select Familiya, Ism, Otasi, tug_yil, (2013-tug_yil) AS yoshi
FROM namuna
Where jinsi=, "Ayol" and tug_yil>1990
```

7. Like amali sintaksisi quyidagicha:

<maydon> LIKE "<belgilar ketma - ketligi>"

Bu amal satrli maydonlar uchun qo‘llaniladi va izlanayongan ma’lumot topilsa true qiymat qaytaradi. Belgilar ketma - ketligini to‘liq kiritish yoki “%” belgisi bilan

tugatish mumkin. Bu belgi ixtiyoriy belgilar ketma - ketligini bildiradi(Accessda % belgisi o‘rniga * ishlatiladi).

Misol:

```
Select familiya, ism, otasi
FROM namuna
Where Familiya LIKE "A%"
```

Bu misol "A" harfidan boshlanuvchi barcha familiyalarni bildiradi. (Adambaev, Abdurahimov, Alimov, Azamov, Asqarov, Azizov,...)

```
Select familiya, ism, otasi FROM namuna
Where Familiya LIKE "Abdu%"
```

Bu misol esa "Abdu" harflaridan boshlanuvchi barcha familiyalarni bildiradi.

```
Select familiya, ism, otasi
FROM namuna
Where Ism LIKE "%im%"
```

Bu misolda ketma - ketlikda "im" qismi bor bo‘lgan ismlarni bildiradi.

(Karimboy, Abdukarim, Rahimjon, Hakimboy, Salimjon,...)

8. between ... and amali sintaksisi quyidagicha:

<maydon> between <qiymat> and <qiymat>

Misol. 1985 va 1990 yillar oralig‘ida tug‘ilganlar ro‘yhatini chiqarish

```
Select familiya, ism, otasi
FROM namuna
WHERE tug_yil BETWEEN 1985 AND 1990
```

10. IN amali sintaksisi quyidagicha: IN (< to‘plam >)

Maydondagi ma’lumot qiymati, to‘plam elementlariga tegishli bo‘lganlarini chiqarib beradi. Masalan, ismi Azamat, Qudrat va Ne’mat bo‘lgan talabalar haqidagi ma’lumotlarni chiqarish talab etilsin:

```
SELECT *
```

```
FROM namuna
WHERE Ism IN ("Azamat","Qudrat","Ne'mat")
```

Misol. Tug‘ilgan yili 1980,1982,1989 bo‘lganlar ro‘yxati esa quyidagicha:

```
SELECT familiya, ism, otasi, tug_yil
FROM namuna
WHERE tug_yil IN (1980,1982, 1989)
```

11. DISTINCT xizmatchi so‘zi ko‘rsatilgan maydon bo‘yicha bir xil qiymatli ma’lumotlardan faqat bittasini olishni bildiradi

```
SELECT DISTINCT familiya FROM namuna
```

9. Count ko‘rsatilgan maydon bo‘yicha yoki butun jadvaldagi yozuvlar sonini aniqlash uchun ishlatiladi

```
Select count (*)
FROM namuna
```

Misol. 95-99 guruhidagi talabalar sonini aniqlash:

```
Select count (*) as soni
FROM namuna
WHERE guruh="95-99"
```

10. min(), max(), avg(), sum() funksiyalari mos ravishda maydonning eng kichik qiymatini (min), eng katta qiymatini (max), o‘rtacha qiymatini (avg) va yig‘indisini (sum) aniqlaydi.

```
Select min(tug_yil), max(tug_yil), avg(2010-tug_yil)
FROM namuna
```

11. Quyida yoshi eng katta bo‘lgan talabani aniqlovchi so‘rov keltirilgan

```
Select familiya, ism, tug_yil FROM namuna
WHERE tug_yil=(select min(tug_yil) FROM namuna)
```

12. Group by

Misol. Guruhgagi talabalar sonini aniqlovchi so‘rov

```
Select guruh, count(guruh) as soni
From namuna
Group by guruh
```

13. Having bo‘limi, group by bo‘limi bo‘lgan xollarda ishlatiladi. Having ning ishlatilishi deyarli Where bilan bir hil. Quyida birdan ortiq takrorlanuvchi familiyalar va ularning takrorlanishlar sonini chiqaruvchi so‘rov keltirilgan.

```
select familiya, count(familiya) as soni
from namuna
group by familiya
having count(familiya) >= 2
```

Jadvalga yangi yozuv qo‘shish INSERT operatori orqali amalga oshiriladi

INSERT INTO <jadval nomi> (<maydonlar ro‘yhati>)

VALUES (< qiymatlar ro‘yhati>)

Misol:

```
INSERT INTO namuna (familiya, ism)
VALUES ("Abdurahimov", "Ne’mat")
```

Mavjud yozuvni tahrirlash (qiymatini yangilash) Update operatori orqali amalga oshiriladi:

UPDATE <jadval nomi> SET <maydon> =<qiymat>

WHERE <shart>

Namuna: Abdurahimov Ne’matning tug‘ilgan yilini 1989 ga o‘zgartirish

```
Update namuna SET tug_yil=1989
WHERE familiya="Abdurahimov" AND ism="Ne’mat"
```

Yozuvni o‘chirish quyidagicha amalga oshiriladi:

DELETE FROM <jadval nomi>

WHERE <shart>

```
DELETE FROM namuna
```

WHERE familiya="Abdurahimov"

Bu so‘rov orqali, familiyasi "Abdurahimov" bo‘lgan barcha ma’lumotlar o‘chiriladi.

Yuqorida keltirilgan so‘rovlarni C++Builder da qo‘llanilishini ko‘rib chiqamiz.

1. File -> New-> Application tanlanadi;

2. Komponentalarning BDE bo‘limidan Queryl komponentasi formaga qo‘yiladi;

3. Queryl komponentasining DatabaseName hususiyatida Alias ko‘rsatiladi. (yani Lesson);

4. Queryl komponentasining SQL hususiyatiga quyidagilar kiritiladi: select * from namuna

5. Queryl komponentasining Active hususiyati true ga o‘zgartiriladi;

6. Komponentalarning Data Access bo‘limidan DataSource komponentasi formaga qo‘yiladi;

7. DataSet hususiyati Queryl qilinadi;

8. Komponentalarning Data Controls bo‘limidan DBNavigator komponentasi formaga qo‘yiladi;

9. DataSource hususiyati DataSourcel qilinadi;

10. Komponentalarning Data Controls bo‘limidan DBGrid komponentasi formaga qo‘yiladi;

11. DataSource hususiyati DataSourcel qilinadi;

12. Formaga Label, Edit va Button komponentalari qo‘yiladi. Button komponentasiga quyidagilar kiritiladi:

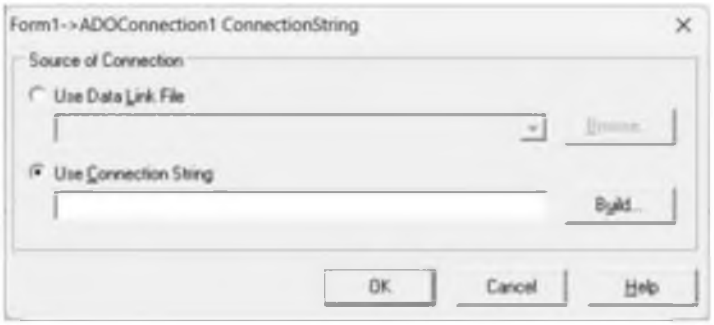
```
Queryl->Close;
Queryl->SQL->Clear() ;
Queryl->SQL->Add("Select * from namuna");
Queryl->SQL->Add("Where Familiya=\""+Editl.Text+"\""); Queryl->Open();
```

Endi tayyor ma'lumotlar bazasi bilan BuilderC++ ni ulanishini ko'rib chiqamiz (Access misolida).

C++Builderda Microsoft Accessga ulanish uchun ADO texnologiyasidan foydalanamiz.

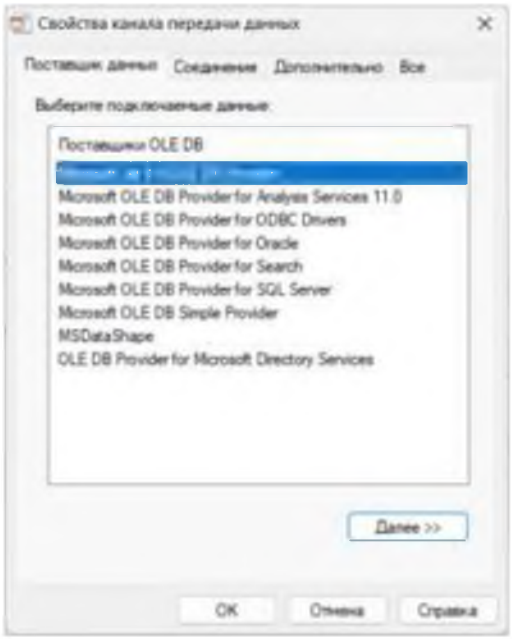
Buning uchun siz quyidagilarni ketma - ket bajaring:

1. C++Builderning komponentalar palitrasidan ADO bo'limini tanlang.
2. Formaga ADOConnectionl va ADOTablel komponentalarini qo'ying.
3. ADOConnectionl ning ConnectionString hususiyati tanlanadi
4. Build tugmasi bosiladi.



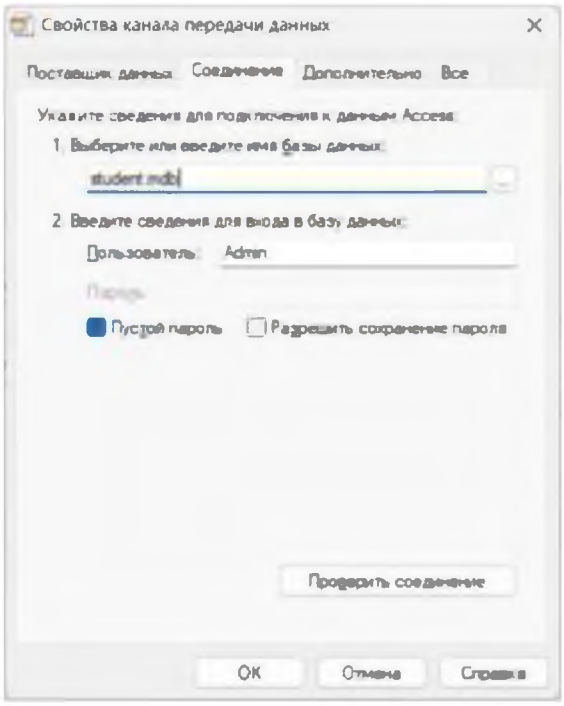
14.5-rasm. ADOConnectionni bog'lash

5. Misrosoft Jet 4.0 Ole DB Provider tanlanadi.



14.6-rasm. Ma'lumotlarni ulash

6. Jadvalga yo‘l ko‘rsatiladi.



14.7-rasm. Jadvalga yo‘l ko‘rsatish

7. Data Source=Baza_nomi.mdb; Data Source bo‘limida faqat fayl nomi qolishi kerak. Shunda programma faylni o‘zi turgan katalogdan izlaydi.

8. Ok tugmasi bosiladi;

9. ADOTablel komponentasining Connection hususiyatiga ADOConnectionl tanlanadi;

10. TableName tanlanadi. User Name va Password so‘ralsa Ok tanlanadi;

11. O‘zimizga kerak bo‘lgan jadvalni tanlashimiz mumkin;

12. Formaga DATA ACCESS bo‘limidan DataSourcel qo‘yiladi va DATASET ga ADOTablel tanlanadi;

13. Data Controls bo‘limidan formaga DBGridl qo‘yiladi va DataSource hususiyatiga DataSourcel tanlanadi;

14. ADOTablel komponentasining Active hususiyati true qilinadi;

15. Dasturni har ish tushurganda parol so‘ramsligi ushun ADOConnectionl komponentasining LoginPrompt hususiyati false qilinadi.

Jadvalning biror maydoni qiymatlarini ComboBox komponentasiga qo‘shish.

Jadvalning biror maydoni qiymatlarini ComboBox komponentasiga qo‘shishda, shu maydon qiymatiga mos ID qiymatlarini ham qo‘shish kerak. Bu ishni ma’lumotlami OBJECT shakldida qo‘shish orqali amalga oshiramiz.

Yuqorida keltirilgan dasturni davom etamiz.

1. Formaga Button va ComboBox komponentalarini qo‘ying.

2. Button komponentasi ustida sichqonchani 2 marta bosing va quyidagilarni kiriting:

```
void __ fastcall TForml::ButtonlClick(TObject *Sender)
{
ADOTablel->First();
ComboBoxl->Clear();
while( !ADOTablel->Eof )
{
ComboBoxl->Items->
AddObject(ADOTablel->FieldByName("g_nomi")->AsString,
(TObject*)ADOTablel->FieldByName("id")->AsInteger);
ADOTablel->Next(); }}
```

3. Formaga Label komponentasini qo‘ying.

4. ComboBoxdagi Object shaklida qo‘shishgan ma’lumotlarni o‘qib olish quyidagicha. ComboBox komponentasi ustida sichqonchani 2 marta bosing va quyidagilarni kiriting:

```
void ___ fastcall TForml::ComboBoxlChange(TObject *Sender)
{
int id;
id = (int)ComboBoxl->Items->Objects[ComboBoxl->ItemIndex];
Labell->Caption = IntToStr(id);
```

}

ComboBox ga ma'lumotlarni Object shaklida qo'shish:

ComboBoxl->Items->AddObject(ADOTablel->FieldByName("g_nomi")->AsString,

(TObject*)ADOTablel->FieldByName("id")->AsInteger);

ComboBoxdagi Object shaklida qo'shishgan ma'lumotlarni o'qib olish:

id = (int)ComboBoxl->Items->Objects[ComboBoxl->ItemIndex]

Nazorat savollari:

1. ODBC (Open Database Connectivity) nim?
2. ODBC (Open Database Connectivity)ning vazifalari?
3. Borland C++ Builder 6 integrallashgan sohasida ma'lumotlar bazasi qanday yaratiladi?
4. Adotable kompanentasining vazifasi?
5. Builder C++ dasturida jadval qanday yaratiladi?

15 BOB. XML VA MA'LUMOTLAR BAZASI

Tayanch so'zlar: *XML, XSLT, XML Editor, XML tahlilchi, element, atribut, Native XML*

15.1. XML haqida umumiy tushunchalar

XML (Extensible Markup Language) — bu ma'lumotlarni saqlash, almashish, va ularga ishlov berish uchun yaratilgan standartlashtirilgan belgilash tili bo'lib, ma'lumotlarning strukturasini aniq ko'rsatish imkonini beradi. XML dastlab 1996-yilda World Wide Web Consortium (W3C) tomonidan ishlab chiqila boshlangan. Uning bosh muallifi Tim Bray bo'lib, Jon Bosak boshchiligidagi guruh bilan birgalikda ishlab chiqilgan va 1998-yilda standart sifatida rasmiylashtirilgan. Ushbu tilni yaratishning asosiy sababi turli tizimlar o'rtasida ma'lumot almashishni soddalashtirish va bu jarayonni standartlashtirish edi. XML o'z tuzilishiga ko'ra HTMLga o'xshash bo'lib, uning asosiy farqi — ma'lumotlarning ko'rinishini emas, balki ularning mazmunini va ma'lumotlar orasidagi bog'liqlikni aniqlash uchun ishlatilishidir. Shu bilan birga, XML dasturlash tillaridan foydalanib, ma'lumotlarga kirish va ularni boshqarish imkonini yaratadi.

XML yaratilish bosqichlari

- ✓ Jon Bosak va Tim Bray boshchiligidagi guruh dastlab SGML (Standard Generalized Markup Language) tilidan ilhom olgan holda XMLni ishlab chiqishga kirishdi. Ular ma'lumotlarni belgilashni osonlashtiradigan va universal bo'lgan formatga ehtiyoj sezishdi.
- ✓ XMLning birinchi rasmiy ko'rinishi 1998-yilda e'lon qilindi va W3C tomonidan rasmiylashtirildi. Bu hujjatda XMLning asosiy qoidalari va imkoniyatlari belgilandi.
- ✓ XML tez orada turli dasturiy ta'minotlarda, ayniqsa veb-xizmatlarda keng qo'llanila boshlandi. Uning osonlik bilan o'qilishi va mashina tomonidan

ishlatilishi uni ma’lumotlar integratsiyasining asosiy vositalaridan biriga aylantirdi .

15.2. XML ning vazifasi

XML ning asosiy xususiyatlari va vazifalari

Ma’lumotlar mustaqilligini ta’minlash: XML yordamida saqlangan ma’lumotlar har qanday dastur va platformada bir xil ko‘rinishda o‘qilishi va ishlatilishi mumkin. Bu esa tizimlar o‘rtasida ma’lumot almashishni soddalashtiradi.

Tuzilishi oson bo‘lgan ma’lumot: XMLda ma’lumotlar daraxt ko‘rinishida (iyerarhiyalashgan) tuziladi. Bu esa ma’lumotlarni kategoriyalar yoki bo‘limlarga ajratish imkoniyatini beradi va ma’lumotlarni qayta ishlashni osonlashtiradi.

Ma’lumotlarning moslashuvchanligi: XML moslashuvchan struktura yaratish uchun juda qulay. Yangi elementlar qo‘shish, mavjudlarini o‘zgartirish yoki qayta nomlash uchun XML moslashuvchan va kengaytiriladigan qoidalar asosida ishlaydi.

Ma’lumotlar integratsiyasi: XML yordamida ma’lumotlar integratsiyasi boshqa formatlar va dasturlar bilan muammosiz ishlash imkonini beradi. Bu xususiyat tufayli XML ma’lumotlar integratsiyasi uchun muhim texnologiyalardan biri hisoblanadi. Ayniqsa, **SOAP** (Simple Object Access Protocol) va **RESTful** APIlar orqali web-servislar bilan ishlashda XML keng qo‘llaniladi. SOAP protokoli XML formatidan foydalanib, turli dasturlar o‘rtasida strukturalangan ma’lumotlarni almashishni ta’minlaydi. Shu bilan birga, **XSLT** (Extensible Stylesheet Language Transformations) yordamida XML ma’lumotlarini boshqa formatlarga (masalan, HTML yoki JSON) o‘tkazish mumkin.

Bundan tashqari, ma’lumotlar bazasi bilan ishlashda **XML DBMS** (Database Management Systems) dan foydalanish ham keng tarqalgan. XML DBMSlar katta hajmdagi XML hujjatlarini samarali saqlash va izlash imkonini beradi. **Microsoft SQL Server** va **Oracle** kabi ko‘plab ma’lumotlar bazasi boshqaruv tizimlari XML bilan

ishlashni qo‘llab-quvvatlaydi va ularni ma’lumotlar bazasida saqlash imkoniyatini yaratadi.

XML bilan bog‘liq texnologiyalar quyidagilarni o‘z ichiga oladi:

- **XSL (Extensible Stylesheet Language)** — XML hujjatlarni boshqa formatlarga (HTML, PDF) aylantirish uchun qo‘llaniladi.
- **XPath va XQuery** — XML hujjatlardan ma’lumotlarni olish uchun ishlatiladigan so‘rov tillari.
- **DOM (Document Object Model)** — XML hujjatning tuzilishini daraxt ko‘rinishida ifodalaydi va unga dasturlash tillari orqali kirish imkonini beradi.

XML ma’lumotlar bazasi XML formatida katta hajmdagi ma’lumotlarni saqlash uchun ishlatiladi. XMLdan foydalanish barcha sohalarda ko‘payib borishi bilan XML hujjatlarini saqlash uchun xavfsiz joy talab qilinadi. Ma’lumotlar bazasida saqlanadigan ma’lumotlar **XQuery** yordamida so‘rov yaratish mumkin va kerakli formatga eksport qilinishi mumkin.

XML MBning tiplari

XML MBning asosiy ikkita tipi mavjud:

- Kiritilgan XML
- Shaxsiy XML (NXD)

Kiritilgan XML ma’lumotlar bazasi

XML bilan ishlaydigan ma’lumotlar bazasi XML hujjatini o‘zgartirish uchun taqdim etilgan kengaytmadan boshqa narsa emas. Bu ma’lumotlar qatorlar va ustunlar jadvallarida saqlanadigan relyatsion ma’lumotlar bazasi. Jadvallar yozuvlar to‘plamini o‘z ichiga oladi, ular o‘z navbatida maydonlardan iborat.

Shaxsiy XML ma’lumotlar bazasi

Shaxsiy XML ma’lumotlar bazasi jadvalga emas, balki konteynerga asoslangan. U katta miqdordagi XML hujjati va ma’lumotlarni saqlashi mumkin. Mahalliy XML ma’lumotlar bazasi XPath ifodalari bilan so‘raladi.

Shaxsiy XML ma’lumotlar bazasi kiritilgan XML ma’lumotlar bazasidan ustun turadi.

Misol. Quyidagi misol XML ma’lumotlar bazasini namoyish etadi:

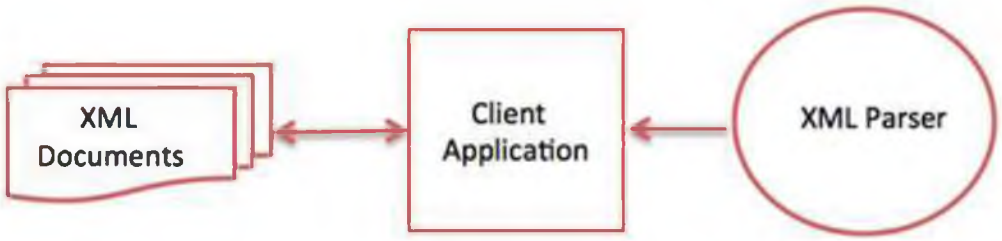
```
<?xml version = "1.0"?>
<contact-info>
  <contact1>
    <name>Tanmay Patil</name>
    <company>TutorialsPoint</company>
    <phone>(011) 123-4567</phone>
  </contact1>
  <contact2>
    <name>Manisha Patil</name>
    <company>TutorialsPoint</company>
    <phone>(011) 789-4567</phone>
  </contact2>
</contact-info>
```

Bu yerda kontaktlar jadvali tuziladi, unda kontakt yozuvlari (contact1 va contact2) mavjud bo‘lib, ular o‘z navbatida uchta obyektdan - name, company va phone iborat.

XML Editor - bu belgilash tili muharriri. XML hujjatlari Notepad, WordPad yoki shunga o‘xshash har qanday matn muharriri kabi mavjud tahrirlovchilar yordamida tahrir qilinishi yoki yaratilishi mumkin. Bundan tashqari, Internetda yuklab olish yoki tahrirlashning yanada kuchli xususiyatlariga ega bo‘lgan professional XML muharririni topishingiz mumkin.

XML-tahlilchi - bu dasturiy ta’minot kutubxonasi yoki paket, bu mijoz dasturlari uchun XML hujjatlari bilan ishlash uchun interfeysni ta’minlaydi. U XML hujjatining to‘g‘ri formatini tasdiqlaydi. Zamonaviy brauzerlar XML tahlilchilariga o‘rnatilgan.

Quyidagi diagrammada XML-parserning XML hujjati bilan o‘zaro aloqasi qanday bo‘lishi ko‘rsatilgan:



Tahlilchining maqsadi XMLni o‘qiladigan kodga aylantirishdir.

Sinov jarayonini soddalashtirish uchun XML hujjatini tahlil qilishni osonlashtiradigan va yanada ishonchli natijalarni ta’minlaydigan bir nechta tijorat mahsulotlari mavjud.

XML teglari ma’lumotlarni aniqlaydi va ma’lumotlarni ko‘rsatish uchun ishlatiladigan HTML teglari sifatida qanday ko‘rsatilishini emas, balki ma’lumotlarni saqlash va tartibga solish uchun ishlatiladi. Yaqin kelajakda XML HTML-ning o‘rnini bosmoqchi emas, lekin HTML-ning ko‘plab muvaffaqiyatli funksiyalaridan foydalanish orqali yangi imkoniyatlar ochadi.

XMLni turli xil tizimlarda va yechimlarda foydali qiladigan uchta muhim xususiyati mavjud:

- XML kengaytirilishi mumkin - XML tavsiflovchi teglaringizni yoki ilovangizga mos keladigan tilni yaratishga imkon beradi.
- XML ma’lumotlar olib yuradi, ularni ifodalaydi - XML qanday taqdim etilishidan qat’iy nazar ma’lumotlarni saqlashga imkon beradi.

- XML - bu keng tarqalgan standart - XML World Wide Web Consortium (W3C) deb nomlangan tashkilot tomonidan ishlab chiqilgan va ochiq standart sifatida mavjud.

XMLdan foydalanish

- ✓ XML katta veb-saytlar uchun HTML hujjatlarni yaratishni osonlashtirish uchun ichki qismda ishlashi mumkin.
- ✓ XML tashkilot va tizimlar o‘rtasida ma’lumot almashish uchun ishlatilishi mumkin.
- ✓ XML ma’lumotlar bazalarini ishga tushirish va qayta yuklash uchun ishlatilishi mumkin.
- ✓ XML ma’lumotlarni saqlash va tartibga solish uchun ishlatilishi mumkin, bu sizning ma’lumotlaringizni qayta ishlash ehtiyojlarini sozlashi mumkin.
- ✓ XML deyarli har qanday kerakli natijani yaratish uchun uslublar jadvallari bilan osonlikcha birlashtirilishi mumkin.

Belgilash tili nima?

XML - bu hujjatlarni inson tomonidan o‘qiladigan va mashinada o‘qiladigan formatda kodlash qoidalarini belgilaydigan belgilash tili. Xo‘sh, belgilash tili nima? Belgilash - bu hujjatga qo‘shilgan, uning ma’nosini ma’lum bir tarzda kuchaytiradigan ma’lumot, chunki bu qismlarni va ularning bir-biri bilan qanday bog‘liqligini aniqlaydi. Aniqrog‘i, belgilash tili - bu hujjat qismlarini chegaralash va yorliqlash uchun hujjat matniga joylashtiriladigan belgilar to‘plami.

Quyidagi misol XML formatlashi matnning bir qismiga o‘rnatilganda qanday ko‘rinishini ko‘rsatadi:

```
<message>
  <text>Hello, world!</text>
</message>
```

Ushbu fragment <message>… </ message> va <text>… </ text> kabi belgilar yoki teglarni o‘z ichiga oladi. <message> va </ message> teglari XML kodining boshi va oxirini belgilaydi. <text> va </ text> teglari Salom, dunyo! matnini o‘z ichiga oladi.

XML - sintaksisi

XML hujjatini yozish uchun oddiy sintaksis qoidalarini muhokama qilamiz. Quyida to‘liq XML hujjati keltirilgan –

```
<?xml version = "1.0"?>
<contact-info>
   <name>Tanmay Patil</name>
   <company>TutorialsPoint</company>
   <phone>(011) 123-4567</phone>
</contact-info>
```

XMLni e’lon qilish

XML hujjati ixtiyoriy ravishda XML deklaratsiyasiga ega bo‘lishi mumkin. Bu shunday yozilgan.

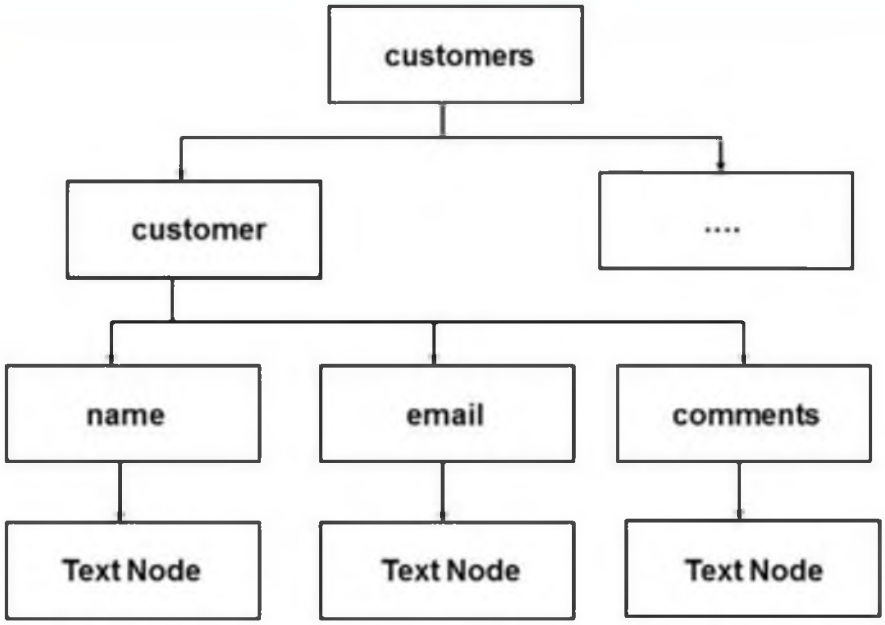
```
<?xml version = "1.0" encoding = "UTF-8"?>
```

XML hujjatining qismlari

XMLga misol.

```
<?xml version="1.0"?>
<customers>
   <customers ID="C001">
      <name>Nancy Davolio</name>
```

```
        <email>nancy@localhost</email>
        <comments>
            <![CDATA[Regular customer since 1995]]>
         </comments>
       </customer>
    <customer ID="C002">
      <name>Steven Buchanan</name>
      <email>steven@gmail.com</email>
      <comments>
<![CDATA[New customer interested in multiple servies]]>
      </comments>
    </customer>
  <customers>
```



Bu misolda daraxtning ildizi customers hisoblanadi. Shuningdek, u boshqa barcha elementlar kelib chiqadigan ildiz (ildiz) elementdir. *Har bir XML fayli faqat bitta ildiz elementiga ega bo'lishi mumkin. U xml fayl deklaratsiyasidan* keyin e'lon qilinishi kerak (misoldagi birinchi qator) va boshqa barcha elementlarni o'z ichiga olishi kerak. Deklaratsiya *majburiy* va hujjatni XML sifatida aniqlash uchun kerak. U uchta psevdo-atributga ega (oldindan belgilangan maxsus atributlar): versiya (1.0 standartiga muvofiq), kodlash (kodlash) va mustaqil. **Elementlar** - boshqa elementlar

va atributlar yordamida ma’lumotlarni saqlaydigan ob’ektlar. **Atributlar** - element qo‘shilganda ko‘rsatiladigan element haqida qo‘shimcha ma’lumot.

15.3. Native XML ma’lumotlar bazasida ma’lumotlarni saqlash. XML(Extended Markup Language) kengaytirilgan xoshiyalash tili sifatida

Native XML ma’lumotlar bazasi (NXMLDB) — bu XML formatidagi ma’lumotlarni tabiiy shaklda, ya’ni ma’lumotlar bazasi ichida **strukturaviy daraxt** yoki **ierarxik ko‘rinishda** saqlaydigan maxsus ma’lumotlar bazasi turidir. Bunday bazalarda XML hujjatlari asosiy birlik sifatida ishlatiladi, ularning tuzilishi va tarkibi saqlanadi.

NXD-lar ikki asosiy turga bo‘linadi: “schema-dependent” (sxemaga bog‘liq) va “schema-independent” (sxemaga bog‘liq bo‘lmagan). Biroq, schema-independent modellari ko‘proq mashhur bo‘lib, ular ma’lumotlarni sxemasiz saqlashni va ular ustida so‘rovlar o‘tkazishni qo‘llab-quvvatlaydi, bu esa tizimni yanada moslashuvchan va qobiliyatli qiladi. Shuningdek, NXD-lar XPath va XQuery kabi so‘rov tillari bilan ishlaydi, bu esa XML hujjatlariga oid ma’lumotlarni qidirish va manipulatsiya qilishda samarali bo‘ladi.

Native XML ma’lumotlar bazalarining xususiyatlari:

XML hujjatlarini to‘liq saqlash: XML fayllari ma’lumotlar bazasida o‘z strukturasi va ma’lumotlari bilan birga saqlanadi.

XML daraxt strukturasi: Ma’lumotlar daraxt tuzilmasida saqlanib, tezkor qidiruv va tahlil qilish imkonini beradi.

XPath va XQuery kabi standart tillar: Ma’lumotlarni so‘rash va manipulyatsiya qilish uchun XMLga mos bo‘lgan tillar ishlatiladi.

Relyatsion tuzilmani talab qilmaydi: XML ma’lumotlari o‘zining tabiiy strukturasi bilan ishlatiladi, bu esa relyatsion modelga moslashni talab qilmaydi.

Ma’lumotlarni saqlash jarayoni:

1. **XML Hujjatning strukturasi:** XML hujjat daraxt tuzilmasiga ega bo‘lib, tugunlar (nodes) va atributlardan iborat.
2. **Saqlash formati:**
   - Ba’zi NXMLDBlarda XML hujjatlar **binarga** o‘giriladi va saqlanadi (tezkor qidiruv uchun).
   - Boshqalarda esa to‘liq **XML hujjat** shaklida saqlanadi.
3. **Indeksatsiya:** XPath so‘rovlari orqali tezkor qidiruvni amalga oshirish uchun tugunlar va atributlar indeksi yaratiladi.

Misol: eXist-db dan foydalanish

eXist-db — bu mashhur Native XML ma’lumotlar bazasi. Quyida uning yordamida XML hujjatlarni qanday saqlash va ularga qanday murojaat qilish haqida misol keltiriladi.

1. XML hujjat:

```
<kitoblar>
  <kitob id="1">
    <nom>O'tkan kunlar</nom>
    <muallif>Abdulla Qodiriy</muallif>
    <janr>Roman</janr>
    <narx>20000</narx>
  </kitob>
  <kitob id="2">
    <nom>Mehrobdan chayon</nom>
    <muallif>Abdulla Qodiriy</muallif>
    <janr>Roman</janr>
    <narx>25000</narx>
  </kitob>
</kitoblar>
```

2. XMLni saqlash:

XML hujjatni **eXist-db** ma'lumotlar bazasiga quyidagicha saqlaymiz:

- Bazaga kirib, **"Upload"** funksiyasi orqali hujjatni yuklaymiz.
- Hujjat daraxt strukturasida saqlanadi.

3. XML so'rovlari:

XQuery yordamida ma'lumotlarni olish:

- **Muallif bo'yicha qidiruv:**

xquery

```
for $kitob in /kitoblar/kitob[muallif = 'Abdulla Qodiriy']
return $kitob/nom
```

- **Narxi 20000 dan yuqori kitoblar:**

```
for $kitob in /kitoblar/kitob[narx > 20000]
return $kitob
```

15.4. Shablon asosida so'rovlar tillari

Relativ ma'lumotlar bazasidan XMLni 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 qayerga joylashtirish kerakligini aniqlash uchun SELECT va $ ustun nomidagi qiymatlarni qo'shish uchun <SelectStmt> elementlaridan foydalanadi:

```
<? xml version = "1.0"?>
<FlightInfo> <Kirish> Quyidagi reyslar mavjud: </ Kirish>
<SelectStmt> SELECT aviakompaniyasi, FltNumber,
Jo'nash, FROM reyslariga yetib borish </SelectStmt>
< Flight >
<Airline> $ Aviakompaniya </Airline>
```

<FltNumber> $ FltNumber </FltNumber>

<Depart> $ jo‘nash </Depart>

< Arive > $ Kelish </Arive>

</Flight>

<Xulosa> ……. </ Xulosa>

</FlightInfo>

Bunday shablonni qayta ishlash natijasi quyidagicha bo‘lishi mumkin:

<? xml version = "1.0"?>

<FlightInfo>

<Kirish> Quyidagi reyslar mavjud: </ Kirish>

< Flight >

< Flight >

<Airline> ACME </Airline>

<FltNumber> 123 </FltNumber>

<Depart> 2017 yil 12-dekabr 13:43 </Depart>

< Arive > 2018 yil 13-dekabr 01:21 </Arive>

</Flight>

...

</live>

<Xulosa> ……< / Xulosa>

</FlightInfo>

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.

15.5. 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’yektga 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’yektga 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.

XPath yordamida bir nechta jadvallar bo‘yicha so‘rovlarni bajarish uchun ob’yektga 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.

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?

IZOHLI LUG‘AT

Termin	O‘zbek tilidagi sharhi	Ingliz tilidagi sharhi
Append	Jadval oxirigacha ma’lumotlarni qo‘shish	Adding data to the end of table
SQLite	Mobil qurilmalarda eng keng tarqalgan ma’lumotlar bazasi tizimlaridan biri.	One of the most common database systems on mobile devices.
Realm	Bu mobil qurilmalar uchun mo‘ljallangan ma’lumotlar bazasi	This is a database designed for mobile devices.
Jadval	Ma’lumotlar bazasida ma’lumotlarni saqlashning asosiy birligi	The basic unit of data storage in a database
Fayl	Bu atama ko‘proq ma’lumotlarni saqlash uchun ishlatiladigan fizik saqlash tizimini anglatadi	This term refers to a physical storage system used to store more data.
Ustun (Column/Field)	Jadvaldagi ma’lumotning turi yoki atributini ifodalaydi	Represents the type or attribute of data in a table
Indeks (Index)	Jadvaldagi qatorlarga tezkor kirish imkonini beradigan ma’lumotlar tuzilmasi	A data structure that allows quick access to rows in a table
So‘rov (Query) tushunchasi	Ma’lumotlar bazasidan kerakli ma’lumotlarni olish uchun yoziladigan buyruqlar yoki so‘rovlardir	These are commands or queries written to retrieve the required information from the database.
Ma’lumotlar turlari (Data Types)	Har bir ustunda saqlanadigan ma’lumotlar turi.	The type of data stored in each column.

Normallashtirish (Normalization)	Ma’lumotlarni takrorlanishni kamaytirish va samaradorlikni oshirish maqsadida jadvallarni tuzish jarayoni	The process of creating tables to reduce data duplication and increase efficiency
Ma’lumotlar modeli	bu ma’lumotlar bazasining tuzilishini tasvirlash uchun ishlatilishi mumkin bo‘lgan tushunchalar to‘plami	This is a set of concepts that can be used to describe the structure of a database.
Mantiqiy ma’lumotlar modeli (Logical Data Model):	Bu model ma’lumotlarni qanday tashkil qilish va saqlashni belgilaydi	Bu model ma’lumotlarni qanday tashkil qilish va saqlashni belgilaydi
Accending order	Eng past va eng yuqori uchun sanada asoslangan matn sohasida alifbo tartibi	Eng past va eng yuqori uchun sanada asoslangan matn sohasida alifbo tartibi
SQL	Strukturalangan so‘rovlar tili	Structured Query Language
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
Entry	Jadval uchun ma’lumotlar	The data for a field
Field	Jadvaldagi maydonlarni belgilaydi	A column in a table. Used to store data

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
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
Design view	Jadvallar uchun maydon ta’riflar ko‘rsatadi	The table view that shows the field definitions for a table
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 sonlar saqlaydi	A field that stores characters (letters, symbols, words, a combination of letters and numbers) and numbers that do not require calculations.

MA’LUMOTLAR BAZASI FANIDAN TEST SAVOLLARI

№	Savollar	Javoblar			
1.	Ma’lumotlar bazasiga ta’rif bering	*ma’lum bir sxema asosida saqlanuvchi ma’lumotlarni ng strukturalashga n majmuasi	uzatish va qayta ishlash uchun yaroqli shakllantirilga n faktlar va fikrlarning taqdim etilishi	qiymatlar majmuasi, operatsiyalar to‘plami bo‘lib, xuddi shunday qiymatlarga qo‘llash mumkin bo‘lgan, shuningdek qiymatlarni saqlashni amalga oshirish va operatsiyalarni bajarish usuli	ma’lumot strukturasi ni yaratish vositalari majmuasi
2.	Ko‘pchilik foydalanuvchilar tomonidan MBni yaratish, to‘ldirish va birgalikda foydalanish uchun mo‘ljallangan dasturiy vositalar majmuasi nima deyiladi?	*MBBT	Ma’lumotlar bazasi	Ma’lumotlar lug‘ati	Hisoblash tizimi
3.	Ma’lumotlar bazasi adminstratori	*bitta yoki bir nechta ma’lumotlar bazasi xaqida to‘liq tasavvurga ega mutaxassis bo‘lib, ushbu ma’lumotlar bazasini loyihalash va qullanilishini nazorat qilish bilan shug‘ullanadi	bitta yoki bir nechta ma’lumotlar bazasi xaqida to‘liq tasavvurga ega mutaxassis bo‘lib, ushbu ma’lumotlar bazasiga xizmat ko‘rsatish bilan shug‘ullanadi	bitta yoki bir nechta ma’lumotlar bazasi xaqida to‘liq tasavvurga ega mutaxassis bo‘lib, ushbu ma’lumotlar bazasiga texnik xizmat ko‘rsatish bilan shug‘ullanadi	bitta yoki bir nechta ma’lumotl ar bazasi xaqida to‘liq tasavvurga ega mutaxassis bo‘lib, ushbu ma’lumotl ar bazasiga tarmoqda xizmat ko‘rsatish bilan

					shug‘ullan adi
4.	MBBT dagi MBning barcha mantiqiy strukturasini ko‘rsatuvchi abstraksiya bosqichini ko‘rsating	*konseptual	tashqi	ichki	jismoniy
5.	Ma’lumotlarni taqdim etish modellari bo‘yicha klassifikatsiya qaysi variantda tasvirlangan	*ierarxik, tarmoqli, relyatsion, ob’ektga yo‘naltirilgan	lokal, tarmoq, bo‘lingan	hujjatli, faktografik, leksikografik	faylli va katalogli
6.	Ma’lumotlarni taqdim etishning relyatsion modeli: foydalanuvchiga ma’lumotlar qaysi ko‘rinishda uzatiladi	*jadvallar	ro‘yxatlar	Daraxt turidagi graf	ixtiyoriy graf
7.	Ma’lumotlarni taqdim etish modelining qaysi biri eng keng tarqalgan xisoblanadi	*relyatsion	tarmoqli	ob’ektga yo‘naltirilgan	ierarxik
8.	Ma’lumotlarni taqdim etishning ierarxik modeli: ma’lumotlar nima orqali taqdim etilgan	*daraxt turidagi graf	ixtiyoriy graf	ro‘yxatlar	jadvallar
9.	Moxiyat-aloqa modelining asosiy tushunchalari	*moxiyat, atribut, aloqa	atribut, jadval, aloqa	ob’ekt, xususiyat, munosabat	kalit, qiymat, assotsiatsiya
10.	Ob’ektlar orasidagi munosabat turini aniqlang: talaba va reyting daftarchasi	*1:1	m:1	n:m	Aloqa yo‘q
11.	Talaba va reyting daftari orasidagialoqa modelini aniqlang	*1:1	1:m	m:n	m:1
12.	Binar aloqaning uch turi to‘g‘ri berilgan javobni ko‘rsating	*Birga-ko‘p, birga-bir, ko‘pga-ko‘p.	Birga-uch, birga-bir, ko‘pga-ko‘p.	Birga-ko‘p, birga-bir, ko‘pga-noaniq.	Birga-aniq, birga-bir,

					ko‘pga-ko‘p.
13.	Talaba va guruh ob’ektlari orasidagi aloqa modelini aniqlang	*n:1	1:1	n:m	bog‘lanmagan
14.	Moxiyat-aloqa modelini kim taklif qilgan	*Piter Chen	Tyuring	Edgar Kodd	Eyler
15.	Talaba va auditoriya ob’ektlari orasidagi munosabat turini aniqlang	*n:m	n:1	1:1	bog‘lanmagan
16.	Daraxt ko‘rinishida qaysi ma’lumotlar bazasi tasvirlanadi?	*ierarxik	tarmoqli	realasion	inververtlangan ro‘yxat asosiga
17.	Kortrej bu ....?	* qator	ustun	jadval	katakcha
18.	Edigan Kod kim bo‘lgan?	*matematik	fizik	ekonomist	tarixchi
19.	Munosobat nima?	*jadval	ustun	qator	katakcha
20.	Relyatsion bazaga o‘xshash juda bo‘lgan analogini ko‘rsating?	*ikki o‘lchamli jadval	vektor	genealogik daraxt	tartiplanmaydigan ma’lumotlar to‘plami
21.	Domen bu ....?	* ustun	jadval	qator	katakcha
22.	Nuqtalar o‘rniga to‘g‘ri keladigan so‘zni ko‘rsating …….ustunlar to‘plami berilgan qatorlar guruhini ko‘rsating	* Jadval	Ma’lumot turlari	Cheklanish	Ma’lumotlar ombori
23.	Jadval ma’lumot kaliti bu	* Jadval ma’lumotlar yig‘indisi bo‘lib, uning	Jadval qatori bo‘lib, o‘zida unikal	Ona jadval kaliti	Jadval ma’lumotlar modeli bo‘lib,

		har bir qatorini aniqlaydi	ma’lumot saqlaydi		uning har bir ustunini belgilaydi
24.	Realyatsion ma’lumotlar bazasida ma’lumotlarni saqlashning asosiy formasi	*Jadval	Yozuv	Domen	Atribut
25.	Realyatsion maydonda mohiyat aloqa diagrammasini o‘zgartirganda atribut nimaga o‘zgaradi?	*atributga	jadvalga	indeksga	ikkilamchi kalitga
26.	Unikal identifikator nima	* Bir qatorni boshqa qatordan ajratib turadigan qiymatga ega ustun	Jadval nomi	Ustun nomi	Qator va ustunlarning mosligi
27.	Jadvalning har xil qatorlari bir xil qiymatdagi kalitga ega bo‘ladimi?	*Yo‘q	Ha	Agar jadval ikkilamchi kalitga ega bo‘lsa	Agar jadval birlamchi kalitga ega bo‘lsa
28.	Qator bu?	* yozuv, atribut, ekzempleyar,borliq	atribut, maydon	fayl	annorgamma
29.	Realyatsion maydonda mohiyat aloqa diagrammasini o‘zgartirganda mohiyat nimaga o‘zgaradi?	* jadvalga	atributga	Ikkilamchi kalitga	indeksga

30.	Ustun bu?	*maydon, atribut	domen, kortej	kortej	jadval, domen
31.	Realyatsion algebrada qanday operatsiya turlari mavjud	*An'anaviy va noan'anaviy	Algebrik va mantiqiy	Kon'yunksiya va diz'yunksiya	Standart va nostandart
32.	Noan'anaviy realyatsion operatsiyalar	* Bog'lanish, tanlash, proeksiya, bo'lish	Impllikatsiya, bog'lanish, har hillik, ulanish	Rad etish, konyuksiya, dezyuksiya	Ketma-ketli bog'lanish, har xilli bog'lanish
33.	Realyatsion ma'lumotlar bazasida qaysi so'rov tillari qo'llaniladi	*SQL	Objective C	Haskell	Basic
34.	Ma'lumotlar bazasi jadvali nima uchun kerak:	*ma'lumotlarni saqlashga;	ma'lumotlarni qayta ishlashga	ma'lumotlar bazasini kiritish va ularni ko'zdan kechirish	qiyin dasturlar yaratishga
35.	An'anaviy realyatsion operatsiyalarga nimalar kiradi	*kesib olish, umumlashtirish,farqlanish,dekart ko'paytma	implikatsiya, bog'lanish, farqlanish, ulanish	ulanish	rad etish
36.	Normallashtirish nimaga kerak	*Anomaliyadan holi bo'lish uchun	Jadval sonin kamaytirish uchun	Jadval sonini ko'paytirish uchun	Foydali ma'lumotlarni ko'paytirish uchun
37.	Ikkinchi normal formada quyidagicha talab qo'yiladi:	* Jadvalning hamma maydonlari birinchi kalitga bog'liq	Hamma maydonlar mustaqil	Hamma maydonlar kalitsiz maydonlarga bog'liq	Hamma maydonlar ikkilamchi kalitga bog'liq
38.	Nechta normal forma mavjud?	*6	5	4	3

39.	SQL kengaytmasi nimani anglatadi?	*Sutrukturalashgan so‘rov tili	Ketma-ket so‘rov tili	Tezkor so‘rov tili	Ma’lumotlar so‘rov tili
40.	Qaysi SQL operatorlari jadvallar sxemasini boshqarishi mumkin?	*CRATE, ALTER, DROP	GRANT, REVOKE	SELECT, UPDATE, INSERT, DELETE	MODIFY, TRUNCATE
41.	Qaysi SQL operatorlari ma’lumotlar ustidan murakkab amallarni bajaradi?	*SELECT, UPDATE, INSERT, DELETE	MODIFY, TRUNCATE	GRANT, REVOKE	CRATE, ALTER, DROP
42.	Obektning nomi ma’lumotlar jadvalida qanday nomlanadi?	*identifikatorlar	Ketma ketliklar	indekslar	konstantalar
43.	Sana vaqt toifasi	*TIMESTAMP	NUMERIC	BOOLEAN	INTEGER
44.	+, -, *, / operatorlari qanday nomlanadi.	*Arifmetik amallar	Mantiqiy amallar	Solishtirish amallari	O‘zlashtirish amallari
45.	Jadvaldagi ustunga qo‘yilgan qanday cheklanish ustun qiymatlarining bo‘sh bo‘lmasligini ko‘rsatadi	*NOT NULL	FORGN KEY	UNIQUE	ChECK
46.	VARChAR	* O‘zgaruvchan toifadagi satr tipi	Butun tip	Sana va vaqt	Moddiy son
47.	TRUE va FALSE Qiymatini qabul qiluvchi toifalar qanday nomlanadi?	*Bul tipli	Butun tipli	Sana va vaqt tipli	Qator
48.	Butun toifa	*NUMERIC	INTEGER	TIMESTAMP	VARChAR

49.	NOT, AND, OR operatorlari vazifasi nimadan iborat?	*Mantiqiy amallar	Solishtirish ammallar	O‘zlashtirish amallar	Arfmetik amallar va operatorlar
50.	Jadvaldagi ustunga qo‘yilgan qanday cheklanish ustun qiymatlarini bog‘langan jadvaldagi birlamchi kalit qiymatlaridan oladi	*FORGN KEY	NOT NULL	ChECK	ChECK
51.	CREATE operatori vazifasi?	* Ma’lumotlar bazasidan obekt yaratish	Ma’lumotlar bazasidan obektni o‘chirish	Ma’lumotlar bazasidan obektni o‘zgartirish	Jadvalga qator qo‘shish
52.	ALTER operatori vazifasi?	* Ma’lumotlar bazasidan obektni o‘zgartirish	Ma’lumotlar bazasidan obektni o‘chirish	Ma’lumotlar bazasidan obektni o‘zgartirish	Jadvalga qator qo‘shish
53.	DROP operatori vazifasi?	*Ma’lumotlar bazasidan obektni o‘chirish	Ma’lumotlar bazasidan obektni o‘zgartirish	Jadvalga qator qo‘shish	Jadvalga ob’ektni o‘chirish
54.	Agar siz ustun rezultat qaytaruvchi jadvalga kirishni xohlasnagiz qanaqa kalit so‘zdan so‘ng SQL so‘rovi ko‘rsatilishi kerak?	*SELECT	WHERE	GROUP BY	FROM
55.	Qidiruv so‘rovlarini tezlashtirish uchun qaysi MBBT mexanizmi ishlatiladi	*indekslar	partisirlangan	svertka	so‘rovlarni bajarilishini tezlashtirish mumkin emas
56.	Keltirilgan qaysi MBBT tekin GNU lisenziyasibilan tarqatiladi	*MySQL	MICROSOFT SQL Server	IBM DB2	ORACLE
57.	So‘rovlarda qanday elementlarga	*Jadval va qatorlarga	Faqat qatorlarga	Faqat jadvallarga	Shartlarga

	psevdonim belgilash mumkin				
58.	Qaysi korxona relyatsion MBBT yaratish bilan shug‘ullanmaydi	*GOOGLE	MICROSOFT	IBM	ORACLE
59.	SELECT operatorini vazifasi?	*Jadvaldan ma’lumotlarni tanlash	Jadvalga satr qo‘shish	Jadvalni o‘zgartirish	Jadvaldan tanlash
60.	UPDATE operatorini vazifasi?	*Jadvalda qatorni o‘zgartirish	Jadvalga satr qo‘shish	Jadvalni o‘zgartirish	Jadvaldan tanlash
61.	DELETE operatorini vazifasi?	*Qatorni o‘chirish	Jadvalga satr qo‘shish	Jadvalni o‘zgartirish	Jadvaldan tanlash
62.	INSERT operatorini vazifasi?	*Jadvalga satr qo‘shish	Jadvalni o‘zgartirish	Jadvaldan tanlash	Qatorni o‘chirish
63.	Qaysi korxona relyatsion MBBT yaratish bozorida yetakchi xisoblanadi	*ORACLE	GOOGLE	MICROSOFT	IBM
64.	Qaysi peredikat Guruhlash uchun ishlatiladi?	*GROUP BY	WHERE	HAVING	ORDER BY
65.	SELECT * FROM STUDENT WHERE SURNAME=‘P’;	*P harfiga teng bo‘lgan familyalar chiqadi.	P bilan boshlanuvchi familyalar chiqadi.	Hech nima chiqmaydi	P harfiga teng bo‘lmagan familyalar chiqadi
66.	Bir nechta shartlardan foydalanishda WHERE operatorida shartlar orasi qanday ajratiladi?	*Kalit so‘zlar, AND yoki OR operatorlari	Nuqtali vergul	FROM kalit so‘zi	Vergul
67.	SELECT operatoridan FROM so‘zidan keying yozuv nimani bildiradi?	*jadvalning nomini	ustunning nomini	shart	1 kalit

68.	Qaysi peredikat berilgan sharni qanoatlantiruvchi qidruvni amalga oshiradi?	*WHERE	GROUP BY	ORDER BY	HAVING
69.	Qays peredikat Saralash uchun ishlatiladi?	* ORDER BY	WHERE	HAVING	GROUP BY
70.	To‘gri yozilgan SELECT operatorini ko‘rsating.	*SELECT * FROM <jadval nomi>	SELECT * <jadval nomi>	SELECT FROM Table <jadvallar nomi>	FROM
71.	INSERT, UPDATE, DELETE quyidagilarning biriga ishlamaydi?	* DML	DDL	DQL	DSL
72.	Qaesi operatorda WHERE ishlatib bo‘lmaydi?	*INSERT	UPDATE	SELECT	DELETE
73.	Ikkilamchi kalit nimani ko‘rsatadi	* birlamchi kalitni	unikal maydonni	indekslangan maydonni	hech nimani ko‘rsatmaydi
74.	SELECT * FROM STUDENT WHERE SURNAME=‘P%’;	* P bilan boshlanuvchi familyalar chiqadi	P harfiga teng bo‘lgan familyalar chiqadi.	Hech nima chiqmaydi	P harfiga teng bo‘lmagan familyalar chiqadi
75.	CREATE TABLE, ALTER TABLE, DROP TABLE Komandalarini qaysilarda ishlatib bo‘lmaydi	*DDL	DML	DQL	DSL
76.	SELECT operatoridan keying * nimani bildiradi?	*hamma ustunlarni belgilashni	hamma satrlarni belgilash	satr qatorlari yulduzcha bilan shifrlanganini	bu belgidan foydalanib bo‘lmaydi

77.	Ichki bog‘lanish peratori – bu	*INNER JOIN	FULL OUTER JOIN	RIGHT OUTER JOIN	LEFT OUTER JOIN
78.	INTERSECT operatori nimaga mo‘ljallangan	*Ikkala tanlovda mavjud umumiy natijalarni chiqarish uchun	Birinchi tanlovda mavjud, lekin ikkinchisida yo‘q bo‘lgan natijalarni chiqarish uchun	Ikki so‘rov tanlovining natijalarini birlashtirish uchun	Ikki jadvaldan olingan natijalarni birlashtirish uchun
79.	SELECT COUNT(id) FROM STUDENT so‘rovini nima qaytaradi	* STUDENTlarning miqdori	STUDENTning maksimal indentifikatori	STUDENTning oxirgi identifikatori	STUDENTning birinchi identifikatori
80.	Qaysi operator yordamida «Raznostь» amali bajariladi	* MINUS	INTERSECT	UNION	DEVIDE
81.	Jadval ustunidagi yig‘indi qiymati qanday hisoblanadi	* SUM funksiyasi yordamida	COUNT funksiyasi yordamida	MIN funksiyasi yordamida	MAX funksiyasi yordamida
82.	JOIN operatori nimaga mo‘ljallangan	* Ikki jadvaldan olingan natijalarni bitta jadvalda birlashtirish uchun	Ikkala tanlovda mavjud umumiy natijalarni chiqarish uchun	Birinchi tanlovda mavjud, lekin ikkinchisida yo‘q bo‘lgan natijalarni chiqarish uchun	Ikki so‘rov tanlovining natijalarini birlashtirish uchun
83.	SELECT MAX(id) FROM STUDENT so‘rovini nima qaytaradi	*STUDENTning maksimal indentifikatori	STUDENTning oxirgi identifikatori	STUDENTning birinchi identifikatori	STUDENTlarning miqdori
84.	SELECT AVG(age) FROM STUDENT so‘rovi nimani qaytaradi	*STUDENTlarning o‘rtacha yoshini	Maksimal STUDENT	Minimal STUDENT	STUDENTlar yoshining yig‘indisini
85.	Quyidagi savol qanday javob qaytaradi SELECT	* groups va students jadvalidagi	groups va students jadvalidagi	groups va students jadvalidagi s.group_id = g.id	groups va students jadvalidagi

	* FROM students s, Groups g	dekart ko‘paytmani chiqaradi	barcha qatorlarni chiqaradi	tengligi orqali bog‘lanadigan barcha qatorlarni chiqaradi	s.group_id = g.id tengligini qanoatlantiruvchi qatorlardan tashqari barcha qatorlarni chiqarib beradi
86.	Jadvaldagi qaydlar miqdori qanday hisoblanadi	* COUNT funksiyasi yordamida	AVG funksiyasi yordamida	MIN funksiyasi yordamida	MAX funksiyasi yordamida
87.	Qaysi operatorlar bilan MIN, MAX, AVG, SUM agregatnix funksiyalar bajarilishi mumkin	*faqatgina SELECT bilan	SELECT va DELETE	UPDATE	INSERT va UPDATE
88.	MINUS operatori nimaga mo‘ljallangan	*Birinchi tanlovda mavjud, lekin ikkinchisida yo‘q bo‘lgan natijalarni chiqarish uchun	Ikki so‘rov tanlovining natijalarini birlashtirish uchun	Ikki jadvaldan olingan natijalarni birlashtirish uchun	Ikkala tanlovda mavjud umumiy natijalarni chiqarish uchun
89.	Quyidagi savol qanday javob qaytaradi SELECT * FROM students s INNER JOIN Groups g ON s.group_id <> g.id	* groups va students jadvalidagi s.group_id = g.id tengligini qanoatlantiruvchi qatorlardan tashqari barcha qatorlarni chiqarib beradi	groups va students jadvalidagi barcha qatorlarni chiqaradi	groups va students jadvalidagi dekart ko‘paytmani chiqaradi	groups va students jadvalidagi s.group_id = g.id tengligi orqali bog‘lanadigan barcha qatorlarni chiqaradi
90.	SELECT MIN(id) FROM STUDENT	* STUDENTnin	STUDENTning oxirgi identifikatori	STUDENTning birinchi identifikatori	STUDENTlarning miqdori

	so‘rovi nimani qaytaradi	g minimal indentifikatori			
91.	Qaysi operatorlardan biri faqatgina bitta qator ustida operatsiya bajarishi mumkin	*INSERT	UPDATE	DELETE	SELECT
92.	Quyidagi savol qanday javob qaytaradi SELECT * FROM students s INNER JOIN Groups g ON s.group_id = g.id	* groups va students jadvalidagi s.group_id = g.id tengligi orqali bog‘lanadigan barcha qatorlarni chiqaradi	groups va students jadvalidagi barcha qatorlarni chiqaradi	groups va students jadvalidagi dekart ko‘paytmani chiqaradi	groups va students jadvalidagi s.group_id = g.id tengligini qanoatlantiruvchi qatorlardan tashqari barcha qatorlarni chiqarib beradi
93.	Quyidagi savol qanday javob qaytaradi SELECT * FROM students s FULL JOIN Groups g ON s.group_id = g.id	* NULL qiymatiga ega bo‘lmagan s.group_id = g.id tengligi orqali bog‘lanadigan groups va students jadvalidagi barcha qatorlarni chiqaradi	groups va students jadvalidagi barcha qatorlarni chiqaradi	groups va students jadvalidagi s.group_id = g.id tengligi orqali bog‘lanadigan barcha qatorlarni chiqaradi	groups va students jadvalidagi dekart ko‘paytmani chiqaradi
94.	Ma’lumotlarni qaysi tipiga MIN, MAX, AVG, SUM agregat funksiyalarni qo‘llash mumkin emas	*Qatorlar	Sonlar	Sanalar	Barcha ma’lumotlarga qo‘llash mumkin
95.	Tizimdagi to‘xtalishlarda	*Backup	Tables	Config	Memory

	ma’lumotlarni qayta tiklash uchun nima zarur bo‘ladi				
96.	Qaysi buyruqlardan biri tranzaksiyadagi o‘zgarishlarni saqlash uchun xizmat qiladi	*COMMIT	FLUSh	ROLLBACK	BEGIN TRANSACTION
97.	Qaysi buyruqlardan biri tranzaksiyani boshlanishini e’lon qilish uchun xizmat qiladi	*BEGIN TRANSACTION	COMMIT	ROLLBACK	COMMIT
98.	Tranzaksiya bu.....?	* Ma’lumotlar bilan ishlashda o‘zining mantiqiy birligiga ega bo‘lgan ma’lumotlar bazasi jarayonlarining ketma-ket bajarilish guruxi	Fizik va mantiqiy rad etish hollarida oldingi holatdagi ma’lumotlar bazasini tiklash uchun ma’lumotlar saqlanadigan MBBT funksiyasi	MBBT ga bo‘lgan asosiy talablardan biri bu tashqi xotirada ma’lumotlarni ishonchli saqlanishidir	MBning asosiy qismi bo‘lib hisoblanadi va barcha rivojdagi MBBTda pratokol Write Ahead Log – WAL deb nomlanadi
99.	Qaysi buyruqlardan biri tranzaksiya otkati uchun xizmat qiladi	*ROLLBACK	BEGIN TRANSACTION	COMMIT	FLUSh
100.	Taqsimlangan MB dan foydalanayotgan vaqtda foydalanuvchi uni qanday ko‘ra oladi	* Yagona MB sifatida ko‘radi	Bir nechta MB to‘plami sifatida ko‘radi	Faqatgina MB ni bir qismini ko‘radi	Ruxsat olish imkoniyatiga ega bo‘lmaydi
101.	Har bir tugun o‘zinig xususiy ma’lumotlar bazasi tizimiga ega bo‘lib va bu tugunlar o‘zaro kelishilgan	*Taqsimlangan	Tarmoqli	Lokalli	Ko‘pfodalanuvchili

	holda ishlaydigan tizim nima deb ataladi				
102.	XML ma'lumotlar validatsiyasi uchun nimalar talab qilinadi	*DTD	Jadval sxemalari	Predikatlar qoidasi	XMLvalidatsiyalanmaydi
103.	Extensible Markup Language ning to‘g‘ri abbreviaturasini ko‘rsating	*XML	SGML	HTML	EML
104.	Qaysi variantlardan biri XML ga tegishli emas	*SQL	XCL va XSLT	Xpath	XHTML
105.	XML uchun daraxt ko‘rinishiga asoslangan API - interfeys nima deb nomalanadi	*DOM	XPATH	JSON	GSerring
106.	Qaysi so‘rovlar tilini XML ga qo‘llash mumkin emas	*QBE	XQUERY	XPATH	XLINKS
107.	Deklarativ so‘rov tillari XML uchun nima deb yuritiladi	* XPath	SQL	QBE	XSLT
108.	XML ning aniq konkret tuzilishini o‘zida aks ettiruvchi tuzilma nima deb yuritiladi	*XML SHEMA	XSLT	SQL	QBE

Foydalanilgan adabiyotlar

1. Sh.Nazirov, A.Ne'matov, R.Qobulov, N.Mardonova. Ma'lumotlar bazasi. T.: "Sharq nashriyoti", 2007. – 200b
2. X.N.Zaynidinov, J.T.Usmonov, SH.B.Redjepov, I.Yusupov. Ma'lumotlar bazasi // Al-Xorazmiy Nomidagi Toshkent Axborot Texnologiyalari Universiteti. – T.,2020. -136 b.
3. Волк В. К. Базы данных. Проектирование, программирование, управление и администрирование : учебник / В. К. Волк. — СанктПетербург : Лань, 2020. — 244 с. : ил.—(Учебники для вузов. Специальная литература).
4. Ramez Elmasri., Shamkant B. Navathe. FUNDAMENTALS OF Database Systems SEVENTH EDITION. Copyright © 2016
5. Silberschatz, A., Korth, h. F., & Sudarshan, S. Database system concepts. 6th edition. Mcgraw-hill. - 2010
6. Date, C. J. (2004). An introduction to database systems. 8th edition. Pearson.
7. Itzik Ben-Gan. Querying Data with Transact-SQL. Microsoft Press, 2017 г. –C 352
8. Dusan Petkovic. Microsoft SQL Server 2019: A Beginner's Guide, Seventh Edition 7th. -2020
9. Michael Alexander, Richard Kusleika. Access 2016 Bible 1st Edition, Wiley Publisher. - 2015
10. J. D. Ullman, "Principles of database systems," 3rd edition, computer science press.

Qo'shimcha adabiyotlar

1. Ўзбекистон Республикаси президентининг 2017 йил 7 февралдаги ПФ-4947-сон "Ўзбекистон Республикасини янада ривожлантириш бўйича Ҳаракатлар стратегияси тўғрисида"ги Фармони.

2. Мирзиёев Ш.М. Буюк келажагимизни мард ва олижаноб халқимиз билан бирга қурамиз. Тошкент. «Ўзбекистон», НМИУ, 2017. – 488 б.
3. Joe Fawcett., Liam R.E. Quin., Danny Ayers. BEGINNING XML. Published by John Wiley & Sons, Inc.
4. Usmonov J.T., Xujaqulov T.A. Ma'lumotlar bazasini boshqarish tizimi// o'quv qo'llanma. - T. : Aloqachi, 2018. – 96 b.
5. 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.
6. Sodiqova Sh.Sh., Ashuraliyev A.A., Sodiqova F.B. «Ma'lumotlar bazasini boshqarish va dasturlash texnologiyalari» fanidan laboratoriya ishlarini bajarish uchun uslubiy ko'rsatmalar. – Toshkent, ToshDTU. 2022. - 82 b.
7. Sodiqova Sh.Sh., Ashuraliyev A.A., Sodiqova F.B. «Ma'lumotlar bazasini boshqarish va dasturlash texnologiyalari» fanidan amaliy mashg'ulotlarini bajarish uchun uslubiy ko'rsatmalar. – Toshkent: ToshDTU. 2022. - 54 b.