O‘ZBEKISTON RESPUBLIKASI OLIY VA O‘RTA MAXSUS TA’LIM VAZIRLIGI

ISLOM KARIMOV NOMIDAGI TOSHKENT DAVLAT TEXNIKA UNIVERSITETI

MA’LUMOTLAR BAZASINI BOSHQARISH VA DASTURLASH TEXNOLOGIYALARI

laboratoriya ishlarini bajarish uchun

USLUBIY KO‘RSATMALAR

I - qism

TOSHKENT – 2022

Tuzuvchilar: 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.

Uslubiy ko‘rsatmada «Ma’lumotlar bazasini boshqarish va dasturlash texnologiyalari» fani bo‘yicha laboratoriya ishlarini bajarishga doir ko‘rsatmalar hamda ma’lumotlar bazasini yaratish va ma’lumotlar bazasini boshqarish tizimi bilan ishlashga oid misollar keltirilgan.

Ushbu uslubiy ko‘rsatmalar oliy o‘quv yurtlarining 60610200 - «Axborot tizimlari va texnologiyalari» (tarmoqlar va sohalar bo‘yicha) yo‘nalishi bakalavr talabalari hamda unga turdosh mutaxassislik talabalari uchun mo‘ljallangan.

Islom Karimov nomidagi ToshDTU ilmiy-uslubiy kengashining qaroriga muvofiq chop etildi. (5-sonli bayonnoma, 26.01.2022 y.)

Taqrizchilar:

Nazirova E.Sh.	– Al-Xorazmiy nomidagi TATU «Multimedia texnologiyalari» kafedrasi mudiri, t.f.d., professor;
Sagatov M.V.	– ToshDTU «Axborot tizimlari» kafedrasi mudiri, texnika fanlari doktori, professor.

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.

Bugungi kunda talabalarga dasturiy mahsulotlar yaratish va axborot texnologiyalariga o‘qitish, apparat-dasturiy vositalarni ishlab chiqish va realizatsiya qilish, robototexnika, internet tarmog‘i orqali axborot xizmatlarini eksport qilish, shuningdek, ma’lumotlarni saqlash va hisoblab qayta ishlash sohasida xizmat ko‘rsatish uchun har bir fandan nazariy bilimlarni amaliyotga tatbiq etishni mukammal o‘rgata oladigan o‘quv qo‘llanmalarni taqdim etish muhim masalalardan biridir.

Ma’lumotlar bazasini boshqarish va dasturlash texnologiyalari fani sanoatning barcha sohalarini raqamli texnologiya asosida yangilashga o‘tish davrida dolzarb ahamiyat kasb etib kelmoqda.

Har bir sanoat korxonalarida turli xildagi ma’lumotlarga ishlob berishga to‘g‘ri keladi. Ma’lumotlarni saqlash, ularga to‘g‘ri ishlov berish va ulardan foydalanish uchun «Ma’lumotlar bazasini boshqarish va dasturlash texnologiyalari» fanini chuqur o‘zlashtirish maqsadga muvofiqdir.

Mazkur uslubiy ko‘rsatmada ma’lumotlar bazasi jadvallarini yaratishning dastlabki qadamlaridan boshlab to so‘rovlar yaratib, ular asosida kerakli axborotga ega bo‘lish jarayonlari batafsil yoritib berilgan. Har bir laboratoriya ishini bajarish uchun namuna sifatida misollar keltirilgan va yechish usullari rasmlar orqali ko‘rsatib berilgan.

1-laboratoriya ishi
Ma‘lumotlar bazasini boshqarish tizimlarni o‘rnatish va sozlash

Ishdan maqsad: ma’lumotlar bazasini boshqarish dasturiy vositalari bilan ishlash, sozlash va tahrirlash ko‘nikmalariga ega bo‘lish.

Masalani qo‘yilishi: ma’lumotlar bazasini boshqarish vositalarini o‘rnatish va sozlashni o‘rganish.

Uslubiy ko‘rsatmalar: Zamonaviy MB texnologiyasida MBni yaratish, unga xizmat ko‘rsatish va foydalanuvchilarni MB bilan ishlashiga imkon yaratish maxsus dasturiy uskunalar yordamida amalga oshiriladi. Bunday dasturiy uskunalar majmuasi ma’lumotlar bazasini boshqarish tizimlari (MBBT) deb ataladi.

MBBT — MBni yaratish, uni dolzarb holatda ushlab turish, kerakli axborotni topishni tashkil etish va boshqa xizmat ko‘rsatish uchun zarur bo‘ladigan dasturiy va til vositalari majmuasidir.

MBBT misoli sifatida quyidagilarni keltirish mumkin:

- DBASE dasturi;
- Microsoft Access;
- Microsoft Fox Pro Mac OS va Linux uchun dasturlar;
- Microsoft Fox Pro WINDOWS uchun dastur;
- Microsoft MySQL;
- Paradox Mac OS va Linux uchun dasturlar;
- Paradox WINDOWS uchun dastur.
- PostgreSQL

MB bilan ishlashga kirishishdan oldin ma’lumotlarni tasvirlash modelini tanlab olish kerak. U quyidagi talablarga javob berishi lozim:

- ✓ axborotni ko‘rgazmali tasvirlash;
- ✓ axborotni kiritishda soddalik;
- ✓ axborot bazaga kiritilgan ma’lumotdan foydalanish imkoniyatining mavjudligi;
- ✓ MBning ochiqligini ta’minlash (yangi ma’lumotlar va maydonlar qo‘shish, ularni olib tashlash imkoniyatlari va hokazo).

MB bitta yoki bir nechta modellarga asoslangan bo‘lishi mumkin. Ular qanday modelga o‘zining xossalari (parametrlari) bilan tavsiflanuvchi obyekt sifatida qarash mumkin. Shunday obyekt ustida biror amal (ish) bajarsa bo‘ladi. MB modellarining uchta asosiy turlari mavjud: ***relyatsion, ierarxik*** va ***semantik*** tarmoq.

Relyatsion (lotin tilidagi relatio — munosabat so‘zidan olingan) modelda ma’lumotlarni saqlash uni tashkil etuvchi qismlari orasidagi

munosabatlarga asoslangan. Eng sodda holda u ikki o‘lchovli massiv yoki jadvaldan iborat bo‘ladi. Murakkab axborot modellari ana shunday jadvallarning o‘zaro bog‘langan to‘plamidan iborat.

MBning ***ierarxik*** modeli pastki pog‘onadagi yuqori pog‘onadagiga bo‘ysinish tartibida joylashgan elementlar to‘plamidan iborat bo‘ladi va ag‘darilgan daraxt(graf)ni tashkil etadi. Ushbu model 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. ***Tugun*** — bu ierarxiyaning berilgan sathida joylashgan elementning axborot modelidir.

MBning ***semantik*** tarmoq modeli ierarxik modelga o‘xshashdir. U ham tugun, sath, bog‘lanish kabi asosiy parametrlarga ega. Lekin semantik tarmoq modelida turli sathdagi elementlar orasida «erkin», ya’ni «har biri hamma bilan» ma’noli bog‘lanish qabul qilingan.

Ko‘pchilik MBlar jadval tuzilmasiga ega. Unda ma’lumotlar adresi satr va ustunlar kesishmasi bilan aniqlanadi. MBda ustunlar — ***maydonlar***, satrlar esa ***yozuvlar*** deb ataladi. ***Maydonlar*** MBning tuzilmasini, ***yozuvlar*** esa, unda joylashgan ma’lumotlarni tashkil etadi.

Maydonlar — MB tuzilmasining asosiy elementlaridir. Ular ma’lum xususiyatlarga ega bo‘ladilar. Maydon uzunligi undagi belgilar soni bilan ifodalanadi.

Maydonning yana bir xususiyati, uning nomidir. Maydonda uning nomidan tashkari yana imzo xususiyati ham mavjud. ***Imzo*** — ustunning sarlavhasida aks ettiriladigan axborotdir. Uni maydon nomi bilan aralashtirib yubormaslik lozim. Agar imzo berilmagan bo‘lsa sarlavhada maydon nomi yozib qo‘yiladi. Turli tipdagi maydonlar turli maqsadlarda ishlatiladi va turli xossalarga ega bo‘ladi.

Ma’lumotlar bazasi haqida bilimlarga ega bo‘ldik endigi navbatda dasturiy vositalarni o‘rnatish va ular bilan ishlash ko‘nikmalarini hosil qilamiz.

Yaratrayotgan bazamizning tuzilishi haqida, jadval maydonlari xususiyatlari haqida to‘xtalsak. Dasturiy vosita sifatida PostgreSQLni tanlaymiz. PostgreSQL dasturi yordamida biz bazamizning umumiy strukturasini hosil qilishimiz mumkin. PostgreSQL - eng mashhur ma’lumotlar bazasini boshqarish tizimlaridan biri. Postgresql loyihasining o‘zi Ingres deb nomlangan boshqa loyihadan rivojlangan. Postgresql rasmiy ravishda 1986 yilda ishlab chiqilgan. Keyin u POSTGRES deb nomlangan. Va 1996 yilda loyiha PostgreSQL deb o‘zgartirildi, bu SQL-

ga ko‘proq e'tibor qaratilishini aks ettiradi. Va aslida 1996 yil 8 iyulda mahsulotning birinchi chiqarilishi bo‘lib o‘tdi.

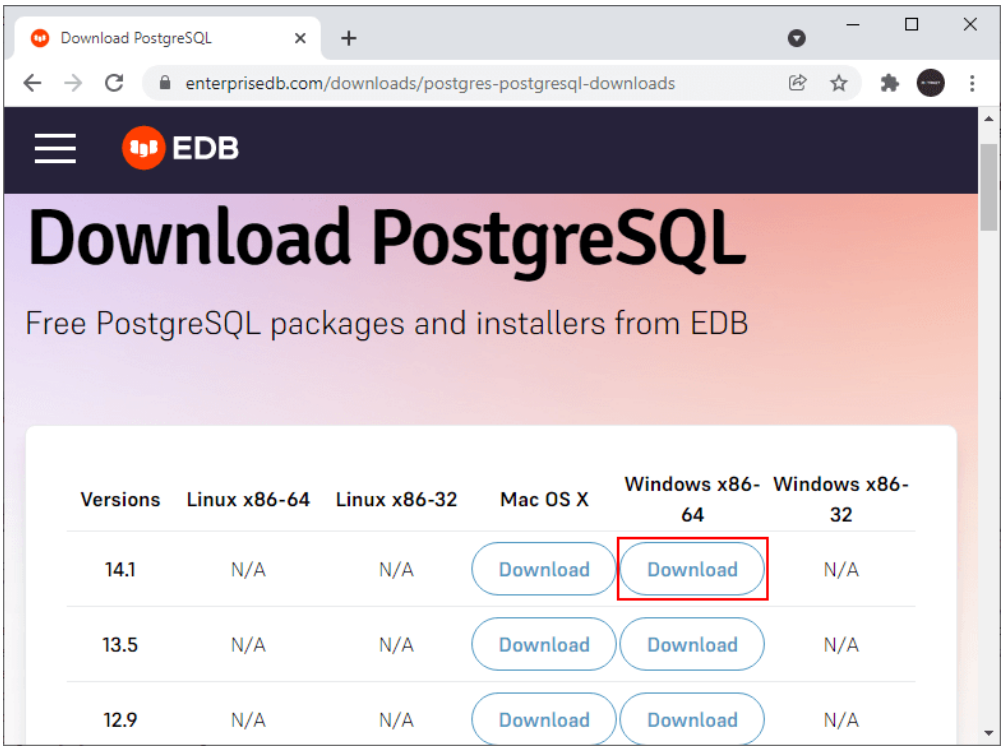
O‘shandan beri PostgreSQL ning ko‘plab versiyalari chiqarildi. Joriy versiya 14-versiya. Biroq, subversiyalar ham muntazam ravishda chiqariladi.

PostgreSQL barcha asosiy operatsion tizimlar - Windows, Linux, MacOS uchun qo‘llab-quvvatlanadi.

Loyihaning rasmiy sayti: https://www.postgresql.org/.

PostgreSQL ochiq manba sifatida rivojlanmoqda. Loyihaning manba kodini https://github.com/postgres/postgres manzilidagi github bazasidan topish mumkin.

Dastlab PostgreSQL dasturiy ta’minotini o‘rnatib olishimiz kerak bo‘ladi. Buning uchun https://www.enterprisedb.com/downloads/postgres-postgresql-downloads havolaga murojaat qilib, dasturiy ta’minotni yuklab olamiz (1.1-rasmda keltirligan oyna ochiladi).



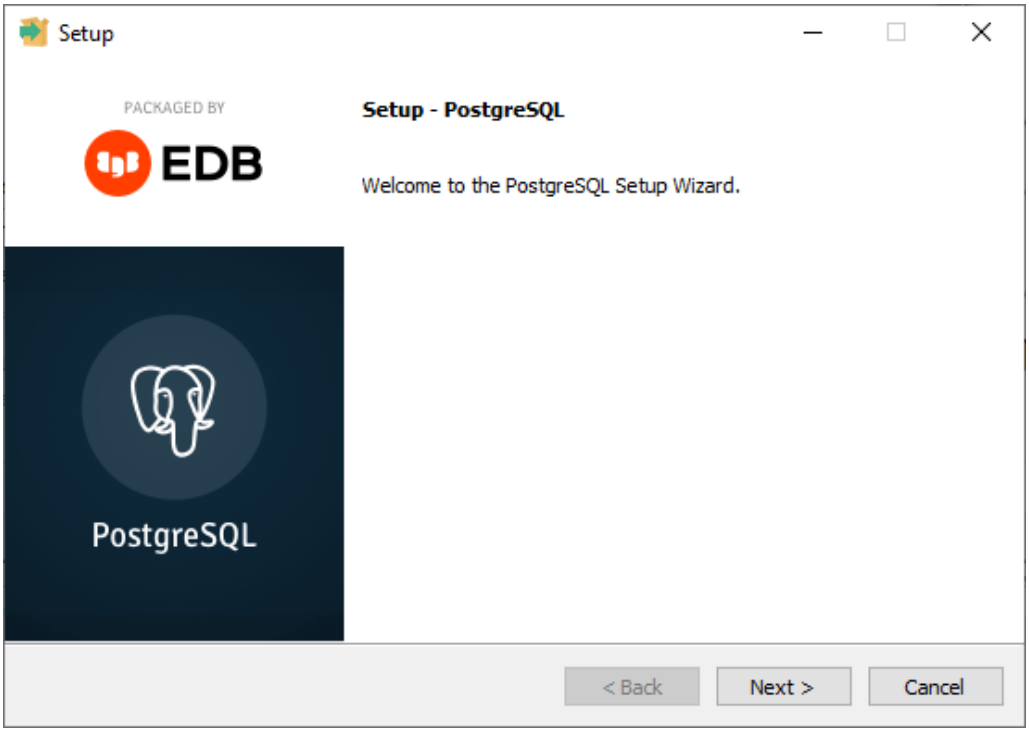
1.1-rasm. PostgreSQL dasturi havolasi

PostgreSQL – bu vizual ma’lumotlar bazasini yaratish uchun vosita bo‘lib, integratsiyalashgan dizaynda PostgreSQL ma’lumotlar bazasi tizimini yagona muhitda modellashtirish, yaratish tizimi.

Dastur xususiyatlari:

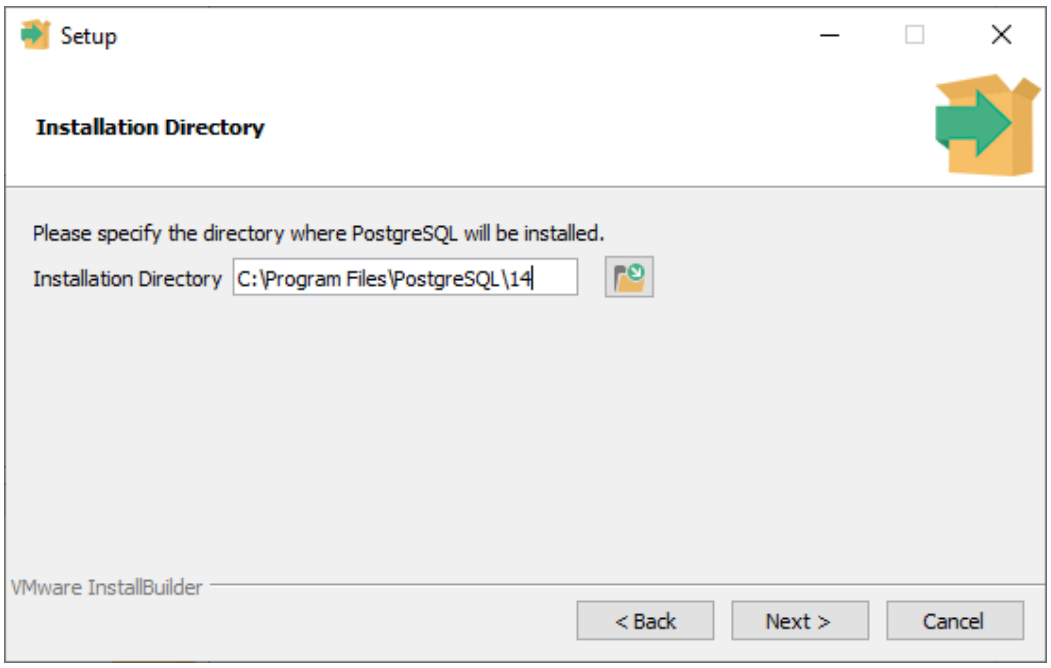
1. Ma’lumotlar bazasi modelini grafik shaklida tasvirlash.
2. Jadvallar o‘rtasidagi munosabatlarni yaratish bilan bog‘liq bo‘lgan ko‘plab munosabatlarni o‘rnatish aniq va funksional mexanizmga ega.
3. SQL so‘rov muharriri ularni darhol serverga yuborishga va jadval shaklida javob olishga imkon beradi.
4. Vizual rejimda jadvaldagi ma’lumotlarni tahrirlash imkoniyati.
5. Reverse Engineering - mavjud ma’lumotlar bazasi serveridan jadvallar tuzilishini tiklash.

Dastlab dasturni o‘rnatish uchun rasmiy saytdan yuklab olish kerak. Dasturning quyidagi versiyasini o‘rnatamiz: « **postgresql-14.1-1-windows-x64** ».



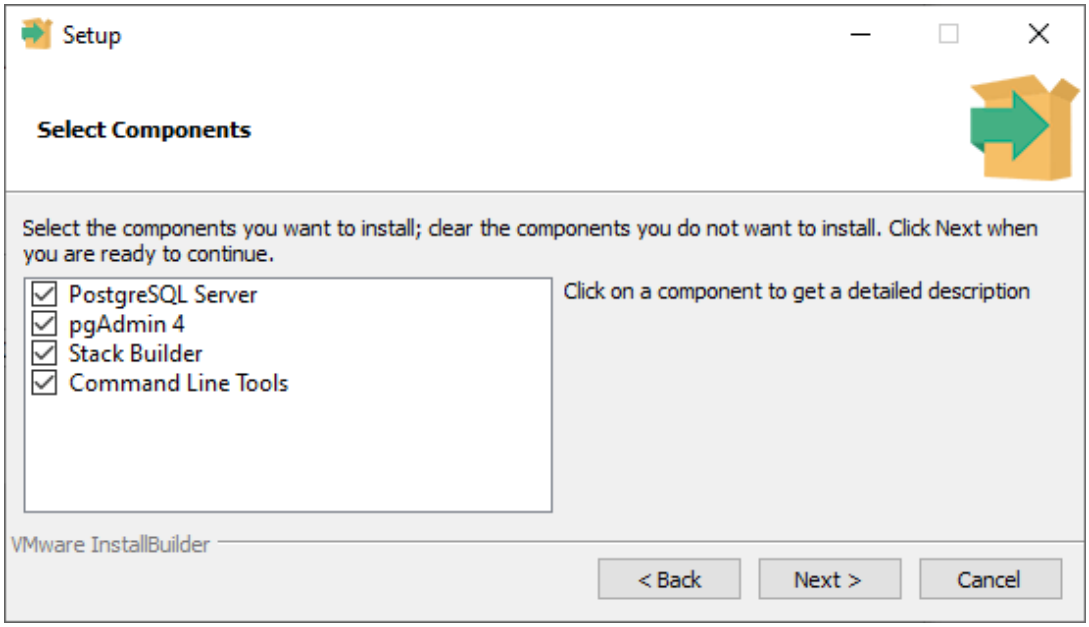
1.2-rasm. PostgreSQL 14.1 versiyasi

Keyingi ekranda siz o‘rnatish papkasini ko‘rsatishingiz kerak bo‘ladi. Keling, standart jildni qoldirib, keyingi bosqichga o‘tamiz:



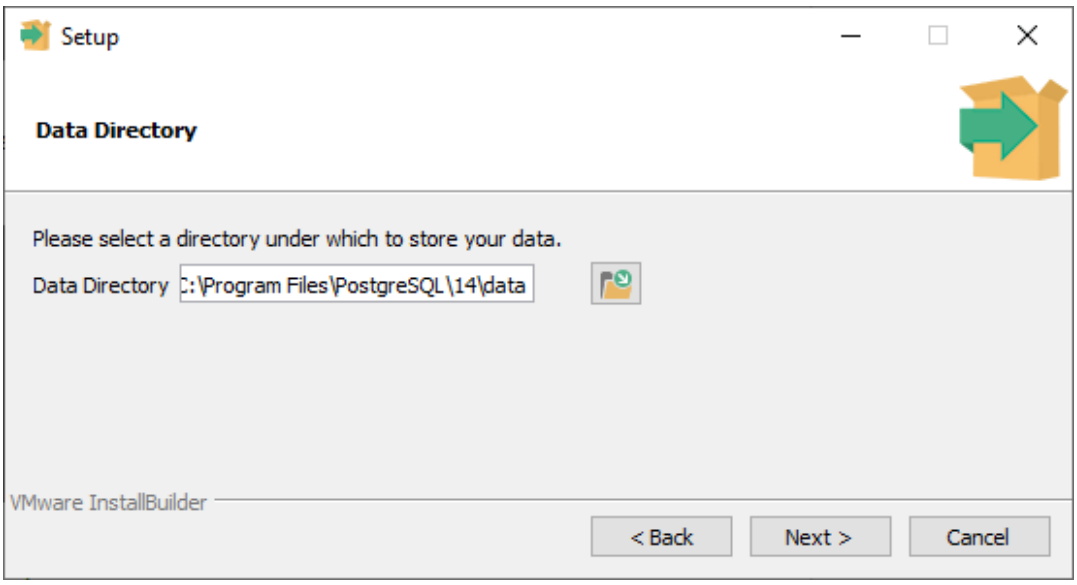
1.3-rasm. PostgreSQL o‘rnatiladigan joyni tanlash

Keyin sizdan o‘rnatish uchun komponentlarni tanlashingiz so‘raladi:



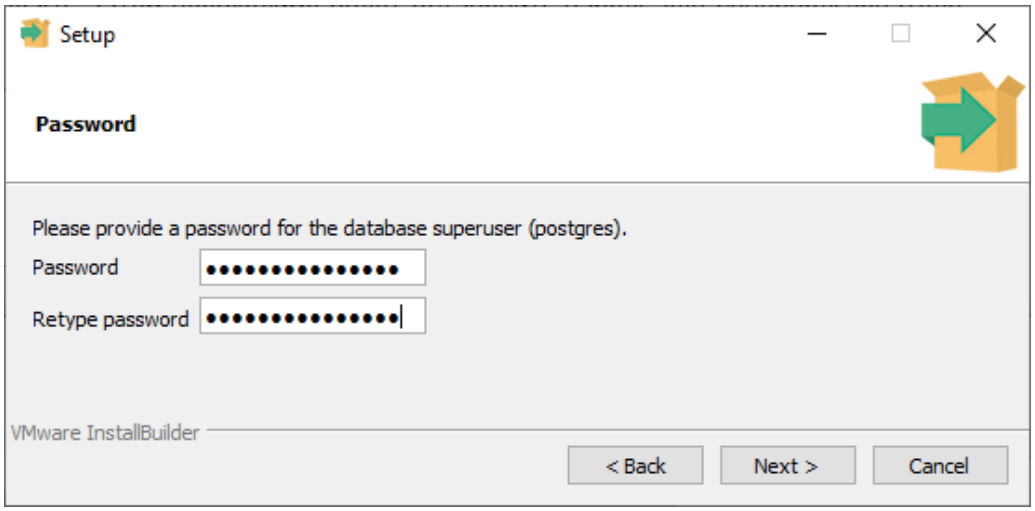
1.4-rasm. PostgreSQL komponentlarni tanlash

Barcha komponentlarni o‘z holida qoldiring va keyingi bosqichga o‘ting. Keyin sizdan ma’lumotlar bazalari saqlanadigan papkani tanlashingiz so‘raladi:



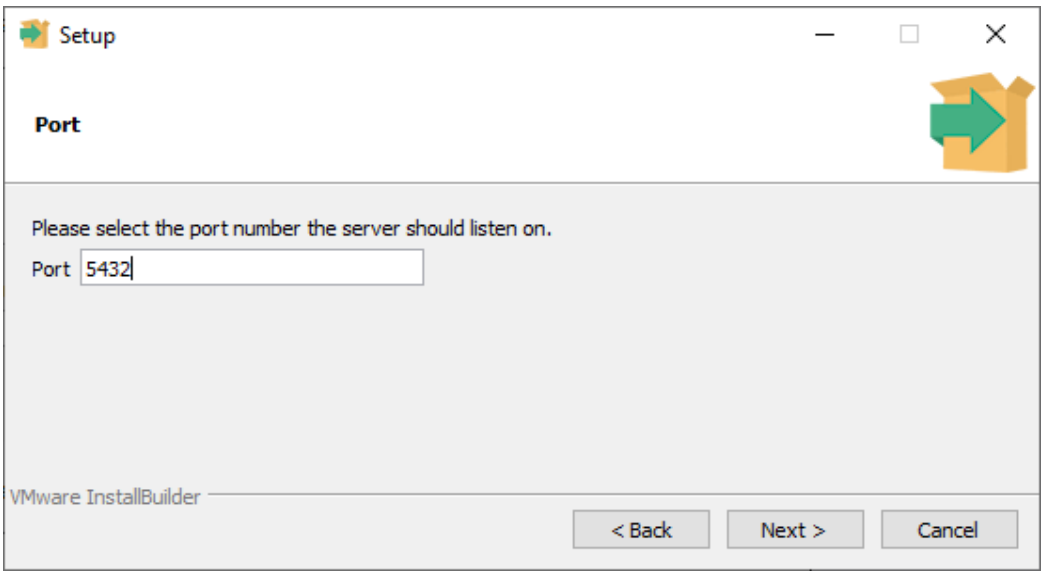
1.5-rasm. Ma’lumotlar bazalari saqlanadigan papkani tanlash

Keling, standart yo‘lni qoldirib, keyingi bosqichga o‘tamiz. Keyin postgres superuser uchun parol o‘rnatishingiz kerak bo‘ladi:



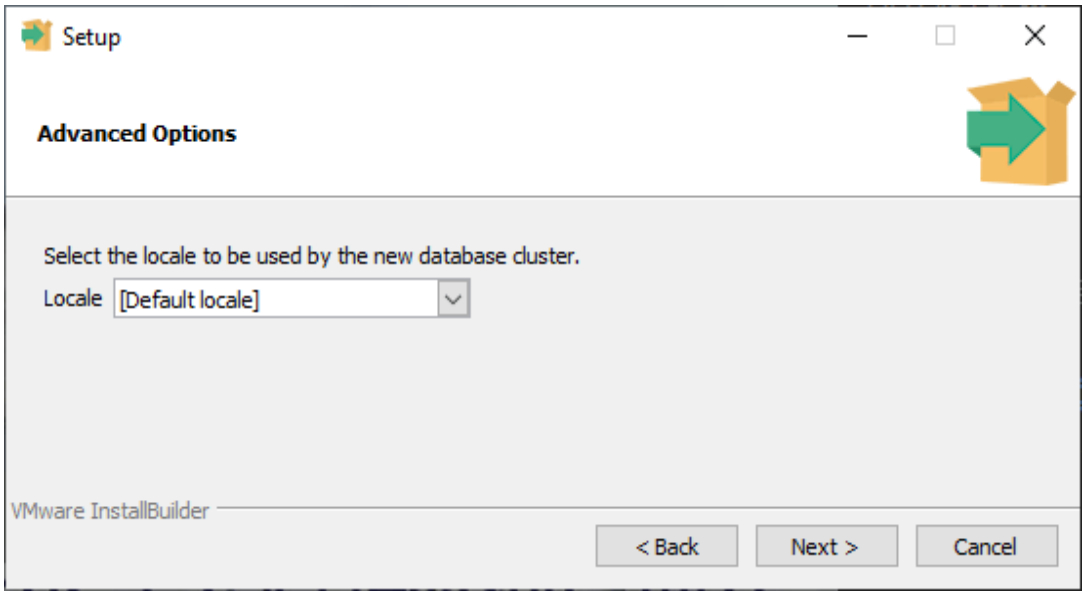
1.6-rasm. Postgres superuser uchun parol o‘rnatish

O‘rnatish vaqtida parolni eslab qoling, chunki u serverga ulanish uchun talab qilinadi. Keyin server ishlaydigan portni o‘rnatishingiz kerak bo‘ladi.



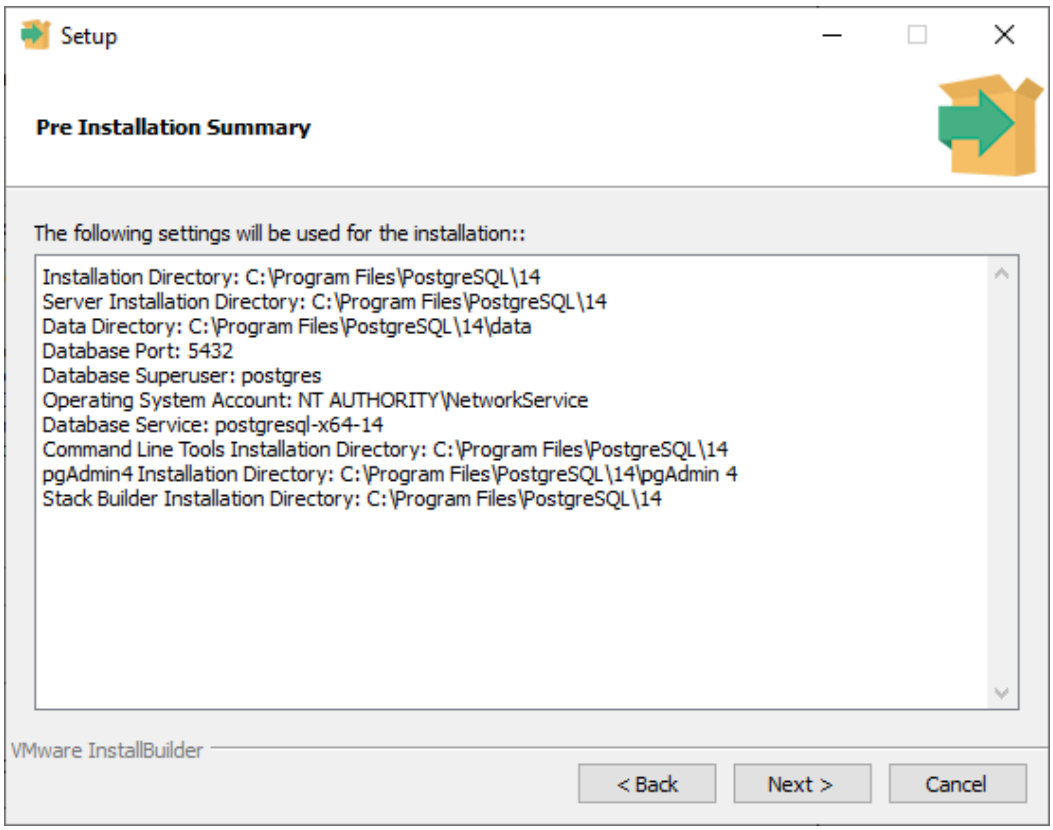
1.7-rasm. Server ishlaydigan portni o‘rnatish

Keyinchalik, siz server tilini o‘rnatishingiz mumkin. Standart sozlamani qoldiramiz:



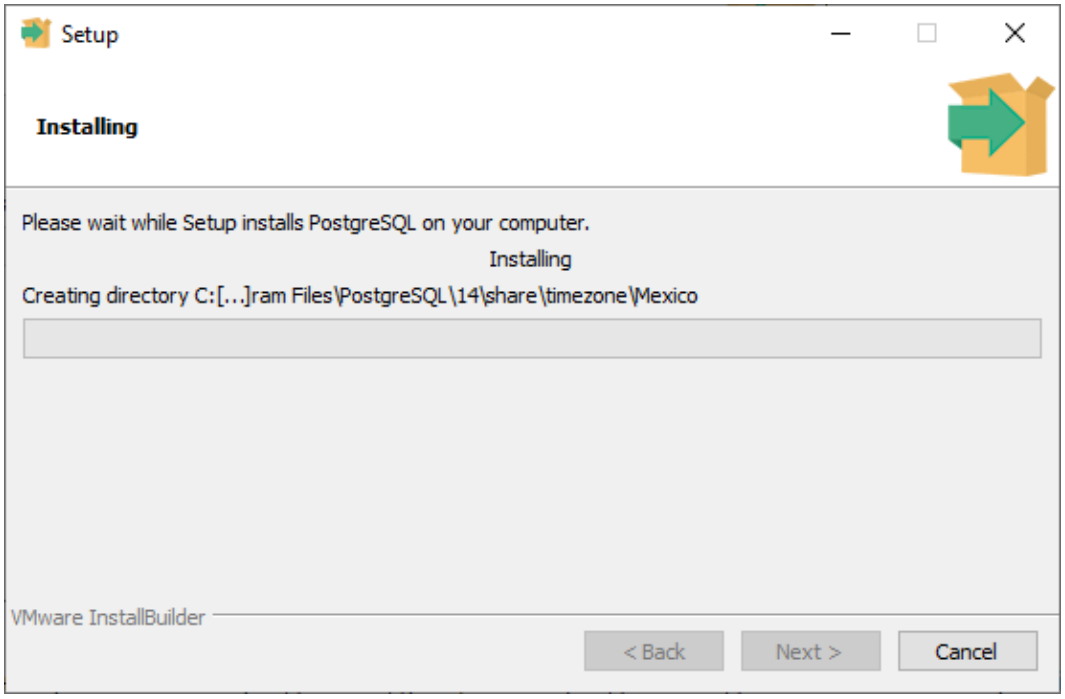
1.8-rasm. Server tilini o‘rnatish oynasi

Shundan so‘ng biz barcha sozlamalarning qisqacha mazmunini ko‘ramiz:



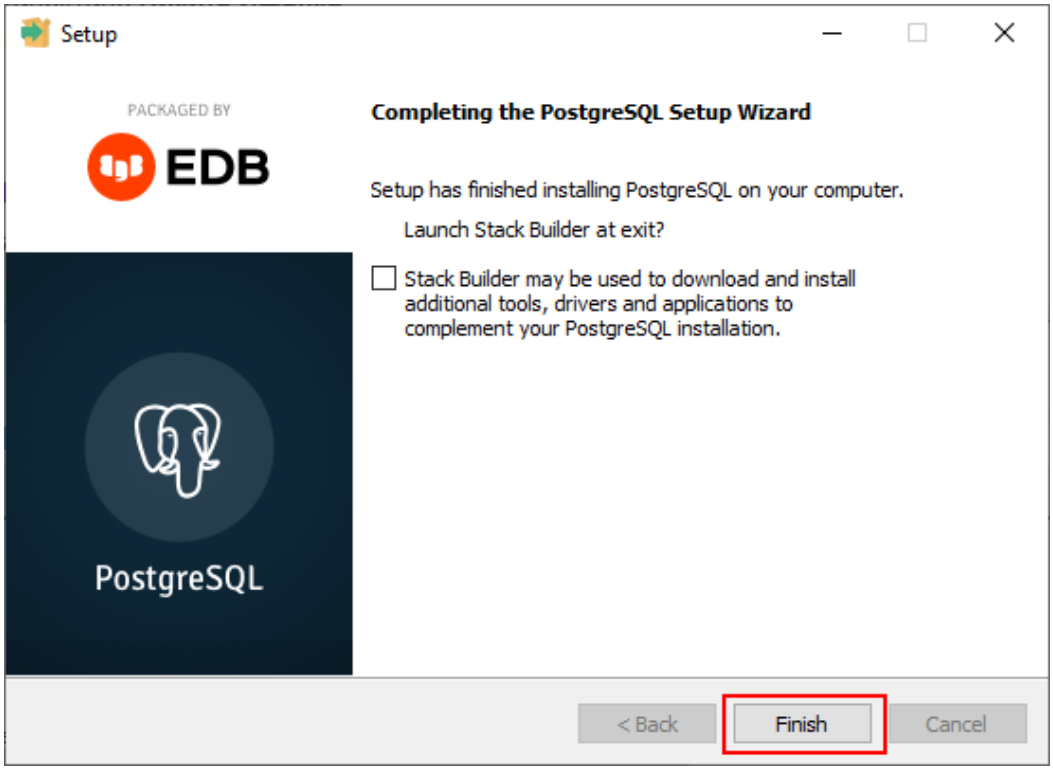
1.9-rasm. Barcha sozlamalarning qisqacha mazmuni

Va agar hamma narsa sizga mos bo‘lsa, unda siz "Next>" tugmasini bosishingiz mumkin va o‘rnatish boshlanadi.



1.10-rasm. PostgreSQL o‘rnatilishi

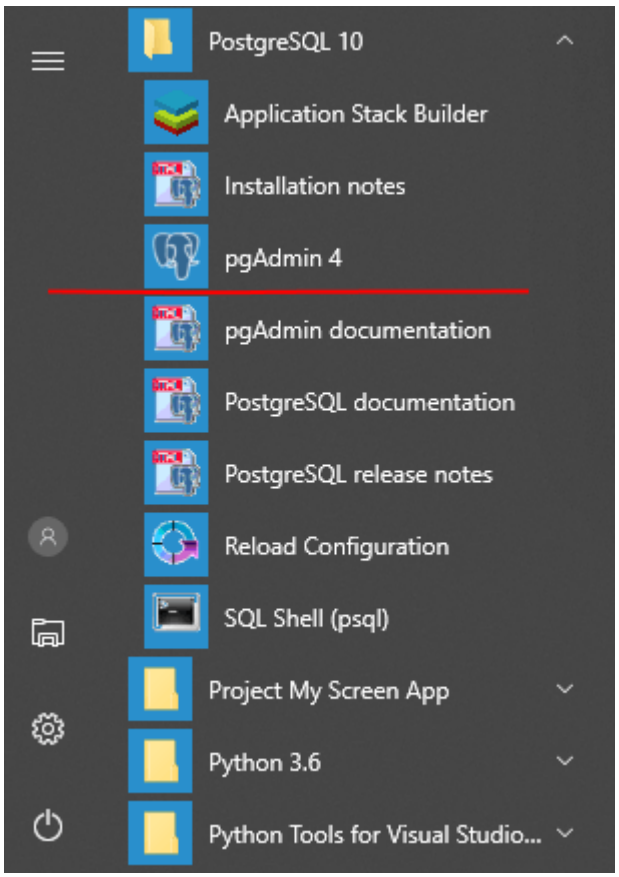
O‘rnatish tugallangandan so‘ng biz quyidagi oynani ko‘ramiz va chiqish uchun “Finish” tugmasini bosing:



1.11-rasm. PostgreSQL o‘rnatishni yakunlash oynasi

Shunday qilib, PostgreSQL serveri o‘rnatildi va biz u bilan ishlashni boshlashimiz mumkin.

Postgresql serverida boshqaruvni soddalashtirish uchun asosiy o‘rnatish to‘plami pgAdmin kabi vositani o‘z ichiga oladi. Bu server bilan ishlash uchun grafik mijozni ifodalaydi, u orqali biz qulaylik bilan ma’lumotlar bazalarini yaratish, o‘chirish, o‘zgartirish va boshqarishimiz mumkin. Shunday qilib, Windowsda o‘rnatishdan so‘ng, biz Start menyusida pgAdmin belgisini topamiz va uni ishga tushiramiz:

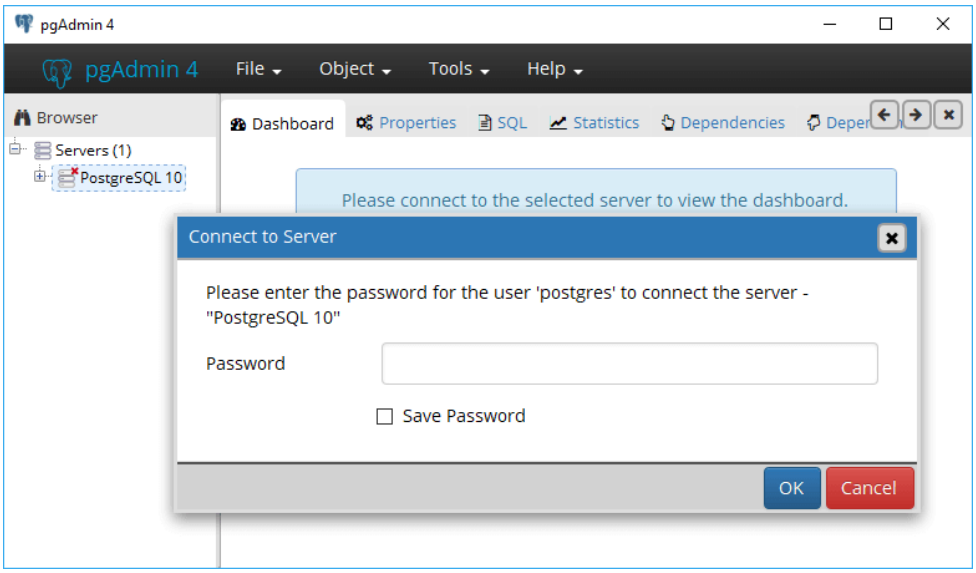


1.12-rasm. PostgreSQL ishga tushirish

Shundan so‘ng biz uchun quyidagi dastur ochiladi:

1.13-rasm. PostgreSQL asosiy oynasi

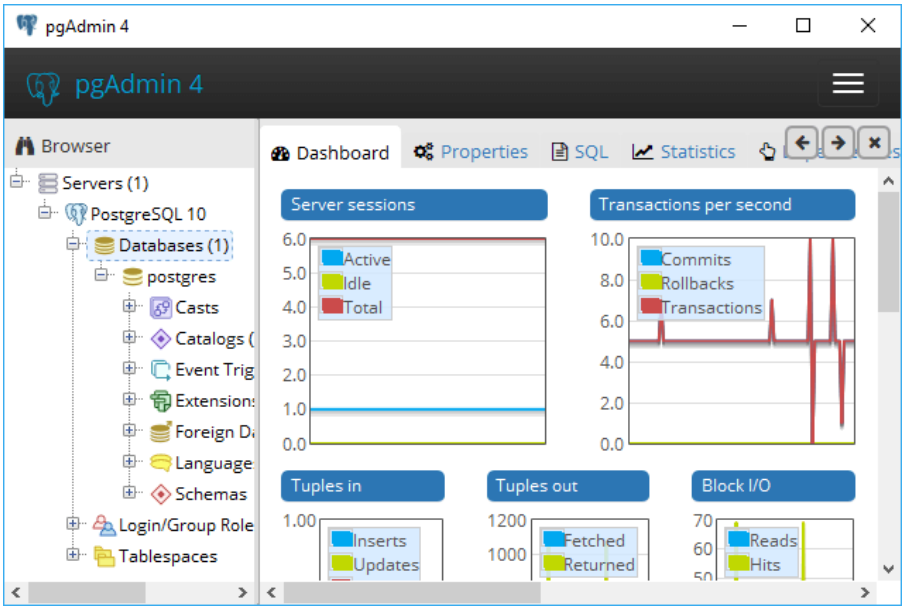
Endi PostgreSQL serveriga ulanamiz. Buning uchun dastur oynasining chap qismida PostgreSQL serverlari to‘plamini o‘z ichiga olgan Serverlar bandini oching. Eng so‘nggi versiyani o‘rnatishda server o‘rnatiladi, u sukut bo‘yicha PostgreSQL 10 nomiga ega. Ushbu elementni bosing va parolni kiritish oynasi ko‘rsatiladi:



1.14-rasm. PostgreSQL serveriga kirish uchun parolni kiritish oynasi

Bu yerda PostgreSQL o‘rnatilishi paytida o‘rnatilgan postgres superuser uchun parolni kiritishingiz kerak.

Muvaffaqiyatli tizimga kirgandan so‘ng, server tarkibi bizga ochiladi:

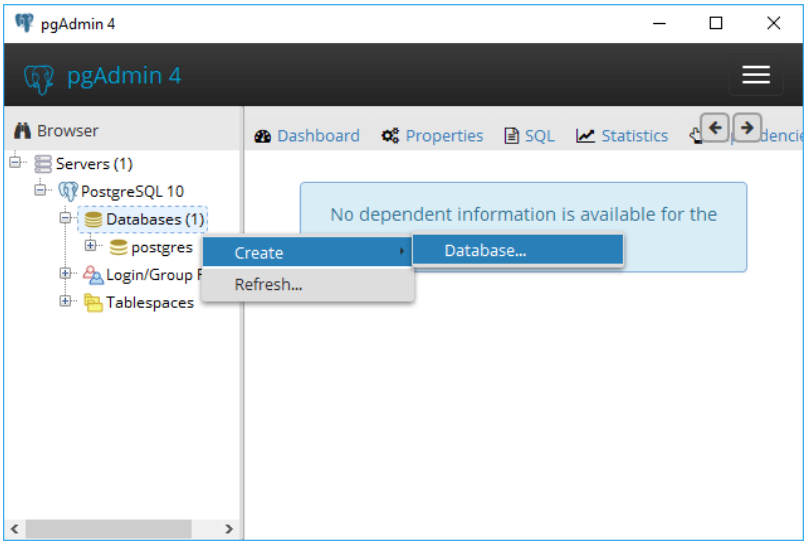


1.15-rasm. PostgreSQL ishchi oynasi

Xususan, Ma’lumotlar bazalari bo‘limida biz barcha mavjud ma’lumotlar bazalarini ko‘rishimiz mumkin. Bu yerda faqat bitta ma’lumotlar bazasini ko‘rishimiz mumkun, postgres.

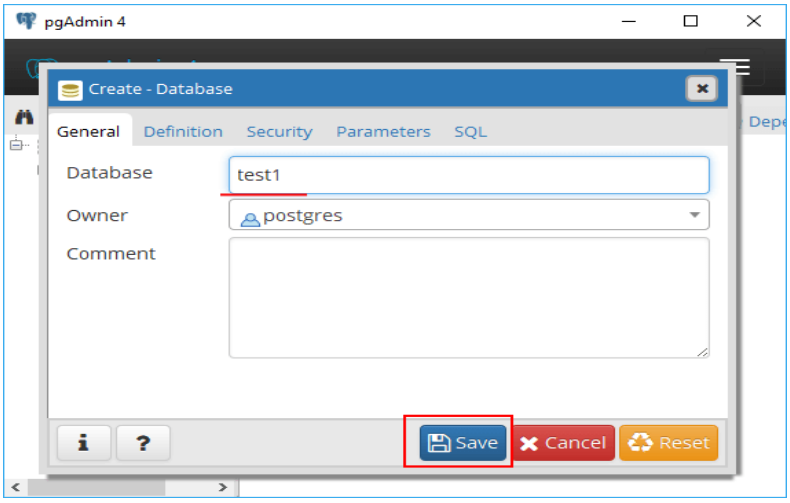
Shuningdek, o‘ng tomonda biz foydalanuvchilar va ularning rollarini boshqarish uchun mo‘ljallangan Login / Group Roles bo‘limini ko‘rishimiz mumkin.

Uchinchi bo‘lim - Tablespaces ma’lumotlar bazasi fayllarining saqlash joyini boshqarish imkonini beradi.



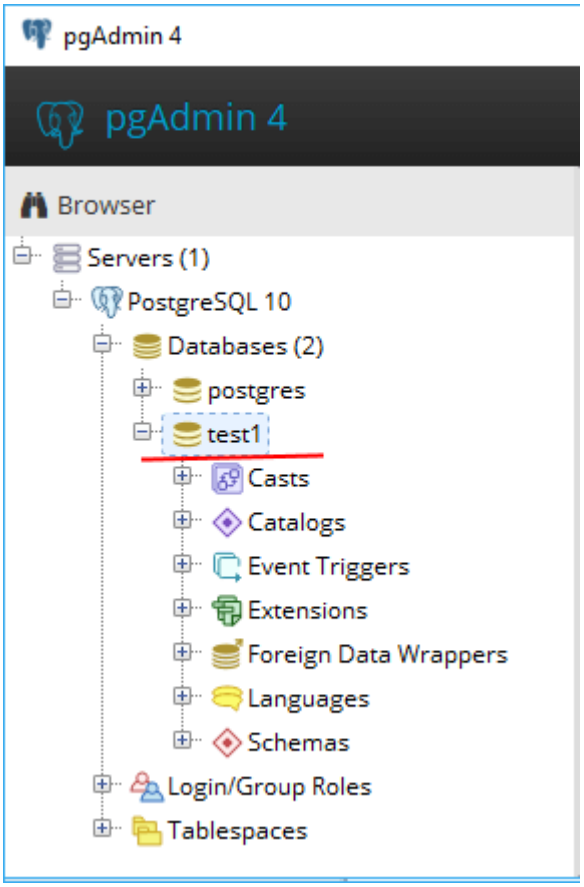
1.16-rasm. PostgreSQL da ma’lumotlar bazasini yaratish oynasi

Shundan so‘ng bizga ma’lumotlar bazasini yaratish oynasi taqdim etiladi. Keling, ma’lumotlar bazasi nomini kiritamiz, masalan, test1 va "Saqlash" tugmasini bosing:



1.17-rasm. PostgreSQL da ma’lumotlar bazasiga nom berish

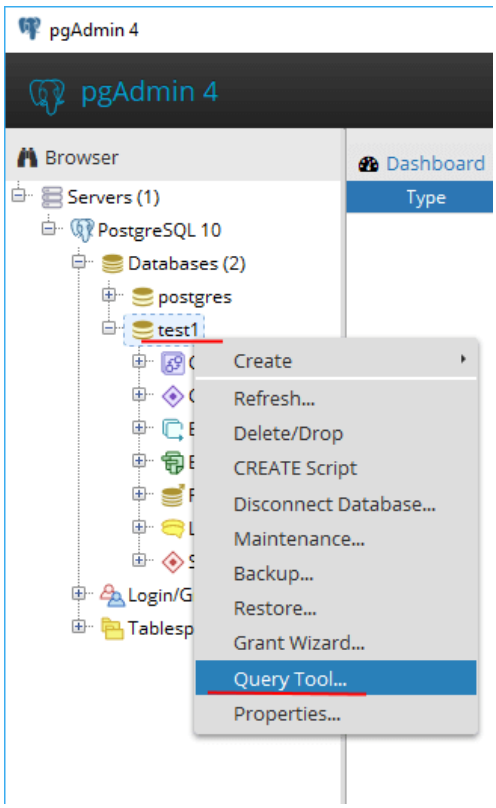
Shundan so‘ng, yaratilgan test1 ma’lumotlar bazasining tarkibi chapdagi daraxt menyusida ko‘rsatiladi:



1.18-rasm. PostgreSQL daraxt menyusi

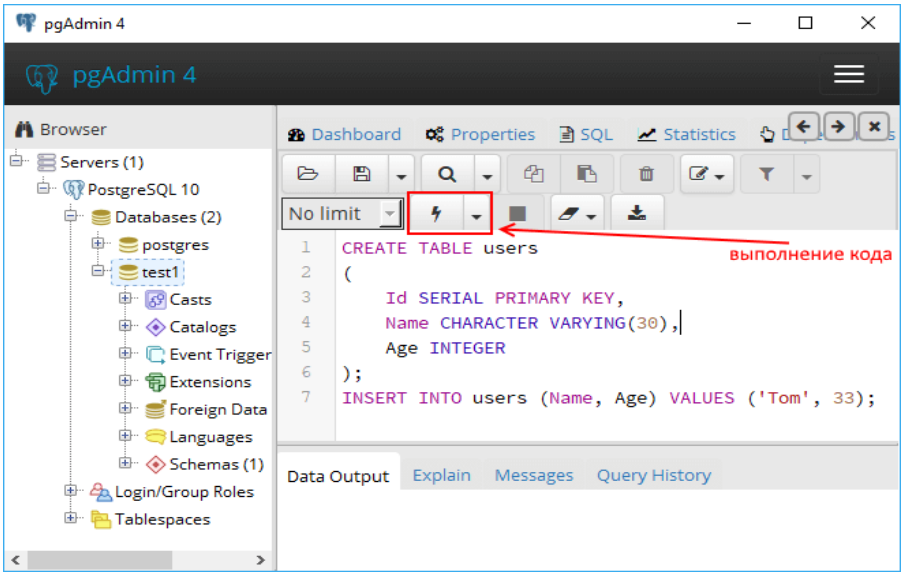
Qoida tariqasida, ma’lumotlar bazasi bilan ishlash maxsus so‘rovlar tili - SQL yordamida amalga oshiriladi. Keling, pgAdmin-da ma’lumotlar bazasiga nisbatan eng oddiy SQL so‘rovlarini qanday bajarishni ko‘rib chiqaylik.

Masalan, oldingi mavzuda yaratilgan test1 ma’lumotlar bazasini olaylik (yoki yangisini yaratamiz) va unga jadval va bir nechta dastlabki ma’lumotlarni qo‘shamiz. Buni amalga oshirish uchun pgAdmin oynasining o‘ng qismidagi ma’lumotlar bazasini sichqonchaning o‘ng tugmasi bilan bosing va paydo bo‘lgan kontekst menyusida So‘rovlar vositasi elementini tanlang:



1.19-rasm. PostgreSQL dasturida so‘rovlar oynasini ishga tushirish

Shundan so‘ng, dasturning markaziy qismida SQL kodini kiritish uchun maydon ochiladi. Keling, quyidagi iboralar to‘plamini kiritamiz:



1.20-rasm. SQL kodini kiritish uchun maydon

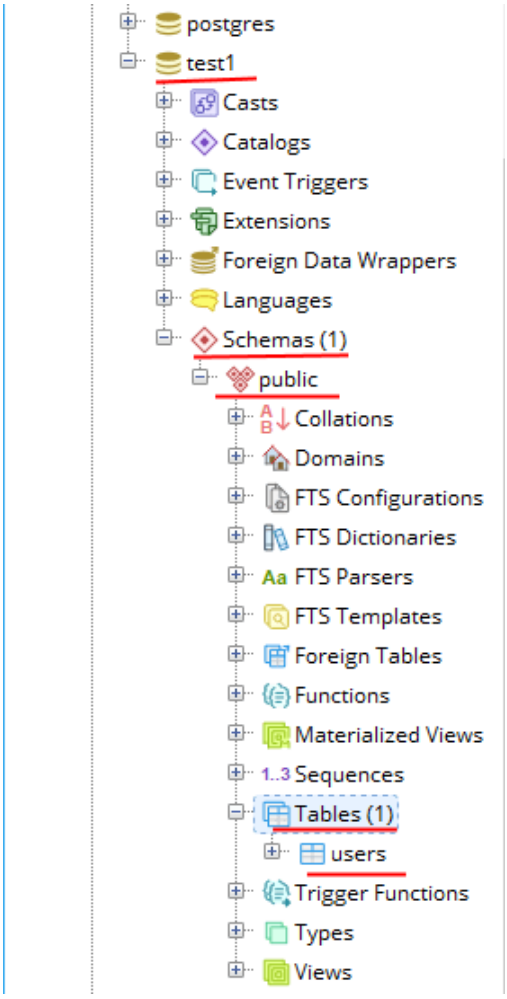
Aslida, barcha kodlar ikki qismga bo‘lingan. Birinchi qism CREATE TABLE operatori bo‘lib, u uchta ustunli Id, Name va Age dan iborat

foydalanuvchilar jadvalini yaratadi. Ikkinchi qism esa bir qatorga bitta jadval qo‘shadigan INSERT operatoridir.

Ushbu ko‘rsatmalarga amal qilish uchun asboblar panelidagi kod ustidagi chaqmoq belgisini bosing. Va shundan so‘ng, foydalanuvchilar jadvali tanlangan ma’lumotlar bazasiga (test1) qo‘shiladi, unga bitta qator qo‘shiladi.

Keyinchalik, ma’lumotlar bazasiga qarshi har qanday boshqa SQL kodi xuddi shunday tarzda bajariladi. Kerakli ma’lumotlar bazasi ham tanlanadi, Query Tool parametri tanlanadi, so‘ngra kiritish maydoniga SQL kodi kiritiladi va u bajariladi.

Shuni ta’kidlash kerakki, har bir jadval uchun sxema aniqlangan. Odatiy bo‘lib, bu “ommaviy” sxema. Shuning uchun, jadvalni topish uchun biz ma’lumotlar bazasi qismiga o‘tishimiz, uni kengaytirishimiz kerak, so‘ngra Sxemalar pastki qismini, undagi umumiy pastki qismi (sxema nomi) va undan keyin jadval bilan bog‘langan barcha jadvallarni ifodalovchi, jadvallar pastki qismini tanlashimiz kerak. umumiy sxema:

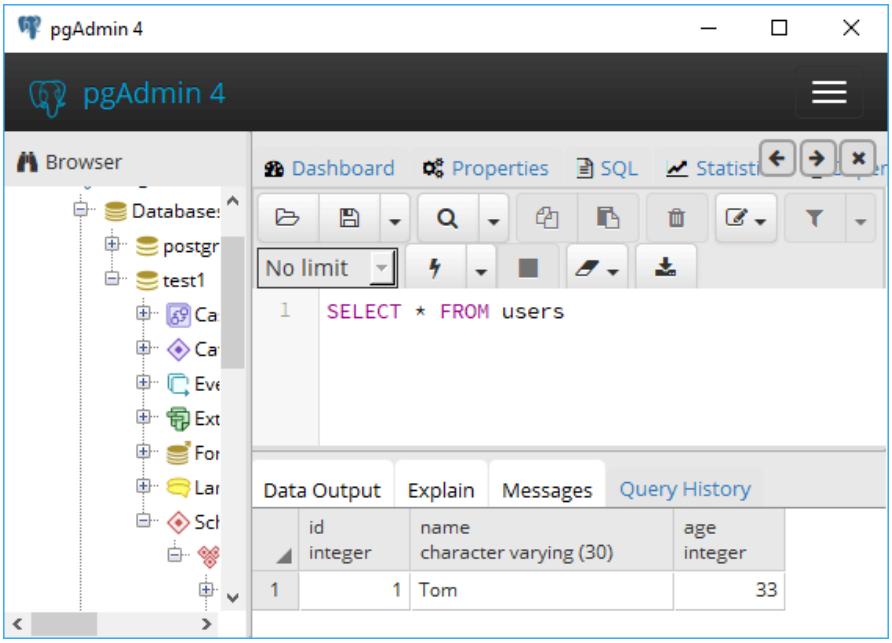


1.21-rasm. Ma’lumotlar bazasining umumiy ko‘rinishi

Endi jadval yaratilganda qo‘shilgan ma’lumotlarni olamiz. Buning uchun quyidagi kodni ishga tushiring:

```
1   SELECT * FROM users
```

Va dasturning pastki qismida, Ma’lumotlarni chiqarish maydonida, biz jadval ko‘rinishida ilgari qo‘shilgan ma’lumotlarni ko‘ramiz.



1.22-rasm. Ma’lumotlarni chiqarish maydoni

Nazorat savollari.

1. Qanday ma’lumotlar bazasini boshqarish tizimlarini bilasiz?
2. Ma’lumotlarni tasvirlash modellari qanday talablari bor?
3. MB modellarining asosiy turlari?
4. PostgreSQL haqida nimalar bilasiz?
5. PostgreSQL dasturida server yaratish?

Topshiriqlar

Har bir amaliy ish uchun topshiriqlar quyidagi jadvalda keltirilgan bo‘lib, talaba keltirilgan obyektlardan birini tanlashi va shu obyekt uchun barcha amaliy vazifalarni bajarishi kerak.

1.	Dorixona
2.	Sug‘urta kompaniyasi
3.	Internet do‘kon
4.	Qurilish mollari do‘koni
5.	Uy oldi sotisi
6.	Kompyter magazin
7.	Salon
8.	Avia kassa
9.	Avtosalon
10.	Maishiy texnika ustaxonasi
11.	Fermer xo‘jaligi
12.	Kutubxona
13.	Kiyim kechak do‘koni
14.	Oziq ovqat do‘koni
15.	Omborxona
16.	Maktab
17.	Hayvonlar do‘koni
18.	O‘yinchoq magazini
19.	Universitet
20.	Yotoqxona
21.	Oshxona
22.	Telefon do‘koni
23.	Mehmonxona

2- laboratoriya ishi
SQL tili asosida ishlaydigan tizimlar muhokamasi. Predmet soha strukturasini yaratish (PostgreSQL asosida)

Ishdan maqsad: ma’lumotlar bazasini boshqarish dasturiy vositalari bilan ishlar, sozlash va tahrirlash ko‘nikmalariga ega bo‘lish.

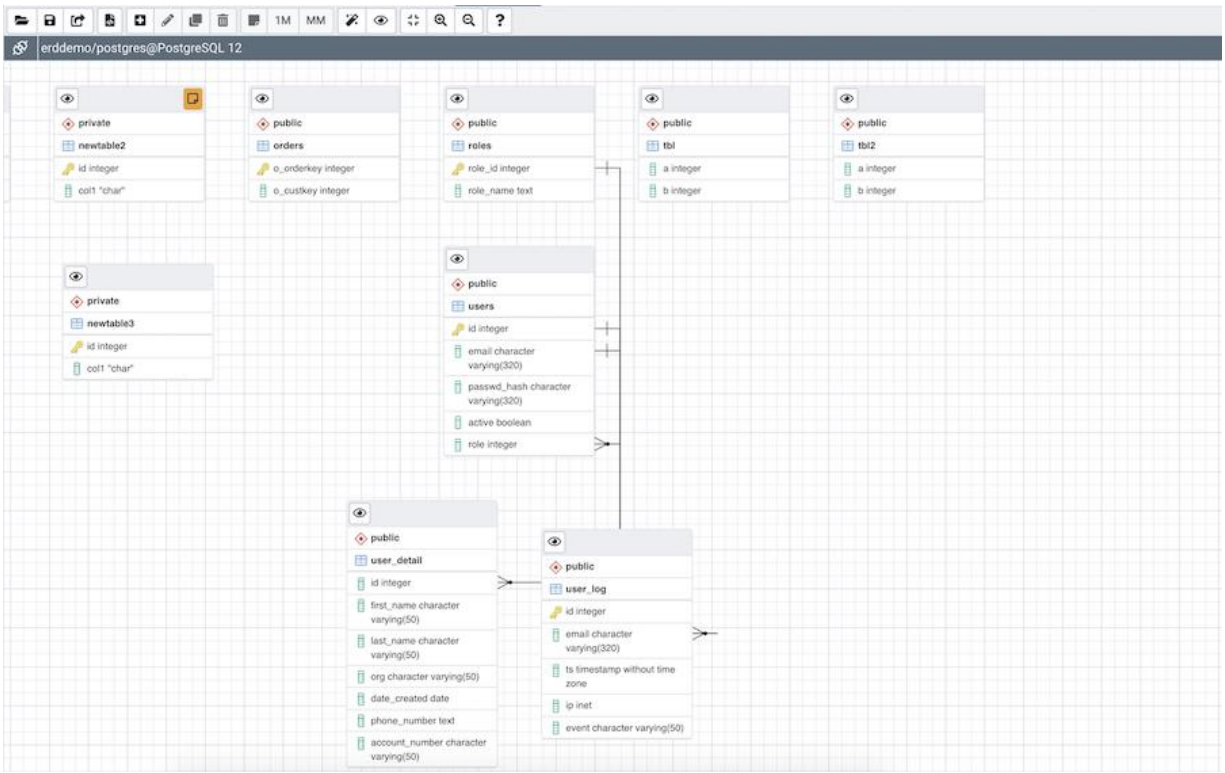
Masalaning qo‘yilishi: ma’lumotlar bazasini boshqarish vositalarini o‘rnatish va sozlashni o‘rganish.

Uslubiy ko‘rsatmalar: Loyihalarning o‘sishi bilan dasturiy qismning murakkabligi oshadi, bir tomonidan ishlov beriladigan ma’lumotlar miqdori, shuningdek ma’lumotlar sxemasining murakkabligi oshadi. PostgreSQL dasturida Entity-Relationship Diagram (ERD) bo‘limidan

foydalanib, ma’lumotlar bazasidagi jadvallarning o‘zaro aloqasini quramiz.

Entity-Relationship Diagram (ERD) vositasi ma’lumotlar bazasi jadvallari, ustunlari va o‘zaro munosabatlarining grafik tasvirini ta'minlovchi ma’lumotlar bazasini loyihalash vositasidir. ERD ma’lumotlar bazasini ishlab chiqish va saqlashda ma’lumotlar bazasi ma’muriga rioya qilish uchun yetarli ma’lumot berishi mumkin. ERD Tool sizga quyidagilarga imkon beradi:

- Ma’lumotlar bazasi jadvallarini va ularning munosabatlarini loyihalash va vizualizatsiya qilish.
- Diagrammaga eslatma qo‘shish.
- Tozaroq vizualizatsiya uchun jadvallar va havolalarni avtomatik tekislash.
- Diagrammani saqlash va u bilan ishlashni davom ettirish uchun uni keyinroq ochish.
- Ma’lumotlar bazasi dizaynidan ishga tushirishga tayyor SQL ni yaratish.
- Mavjud ma’lumotlar bazasi uchun ma’lumotlar bazasi diagrammasini yaratish.
- Jadvallarni brauzer daraxtidan diagrammaga oson olib o‘tish.



2.1-rasm. PostgreSQL da yaratilgan model sxemasi

ERD vositasining bir nechta nusxalarini bir vaqtning o‘zida alohida yorliqlarda ochishingiz mumkin. ERD vositasining nusxasini yopish uchun yorliqlar panelining yuqori o‘ng burchagidagi X belgisini bosing.

ERD asboblar paneli tez-tez bajariladigan vazifalar uchun yorliqlarni taqdim etadigan kontekstga sezgir piktogrammalardan foydalanadi. Variant ajratilgan belgi uchun yoqilgan va kulrang piktogramma uchun o‘chiriladi.

Belgining funksiyasini tavsiflovchi maslahatni ko‘rsatish uchun kursorni asboblar panelidagi belgi ustiga olib boring.

Fayl parametrlari

Open File - Oldindan saqlangan diagrammani yuklash uchun Faylni ochish belgisini bosing - Ctrl + O.

Save - Oldindan saqlangan diagrammani tezda saqlash yoki diagrammani faylga saqlash uchun Saqlash belgisini bosing - Ctrl + S

Save as - Yangi brauzer dialog oynasini ochish va diagrammani saqlash uchun, yangi joyni belgilash uchun Boshqacha saqlash tugmasini bosing - Ctrl + Shift + S.

Eksport opsiyalari

Generate SQL - Diagramma uchun DDL SQL ni yaratish uchun SQL yaratish belgisini bosing va yaratilgan SQL bajarilishiga tayyor bo‘lgan so‘rov vositasini oching - Option + Ctrl + S.

Download image - ERD diagrammasini tasvir formatida saqlash uchun “Rasmni yuklab olish” belgisini bosing - Option + Ctrl + I.

Tahrirlash parametrlari

Add table - Diagrammaga yangi jadval qo‘shish uchun ushbu tugmani bosing. Bosgandan so‘ng, jadval ma’lumotlarini joylashtirishingiz mumkin bo‘lgan jadval dialog oynasi ochiladi - Option/Alt + Ctrl + A.

Edit table - Diagrammadagi jadvalni tahrirlash uchun ushbu tugmani bosing. Bosgandan so‘ng, jadval ma’lumotlarini o‘zgartirishingiz mumkin bo‘lgan jadval dialog oynasi ochiladi. Bu jadval tanlanganda faollashadi - Option/Alt + Ctrl + E.

Clone table - Jadvalning to‘liq tuzilishini klonlash, uni avtomatik yaratilgan nom bilan nomlash va diagrammaga joylashtirish uchun ushbu tugmani bosing - Option/Alt + Ctrl + C.

Drop table/link - Ushbu tugma yordamida jadval yoki havolani tushirishingiz mumkin. Jadval yoki havolani tanlashingiz va uni tushirish uchun ushbu tugmani bosishingiz kerak - Option/Alt + Ctrl + D.

Jadvallar orasidagi aloqa

1M - Ikki jadval o‘rtasidagi munosabatlarni qo‘shish uchun, "birdan ko‘pga" aloqa oynasini ochish uchun ushbu tugmani bosing. Tanlangan jadval mos yozuvlar jadvaliga aylanadi va havolaning ko‘p so‘nggi nuqtasiga ega bo‘ladi - Option/Alt + Ctrl + O.

MM - Ikki jadval o‘rtasidagi munosabatlarni qo‘shish uchun ko‘pdan ko‘pga munosabatlar dialogini ochish uchun ushbu tugmani bosing. Ushbu parametr ikkita tegishli jadval uchun tanlangan ustunlar asosida yangi jadval yaratadi va ularni bog‘laydi - Option/Alt + Ctrl + M.

Utility Options

Add/Edit note - Ma’lumotlar bazasini loyihalashda jadvallar tugunlariga eslatma qilish uchun ushbu tugmani bosing - Option/Alt + Ctrl + N.

Auto align(avtomatik tekislash) - Barcha jadvallar va havolalarni yanada toza ko‘rinishi uchun avtomatik tekislash uchun ushbu tugmani bosing - Option/Alt + Ctrl + L.

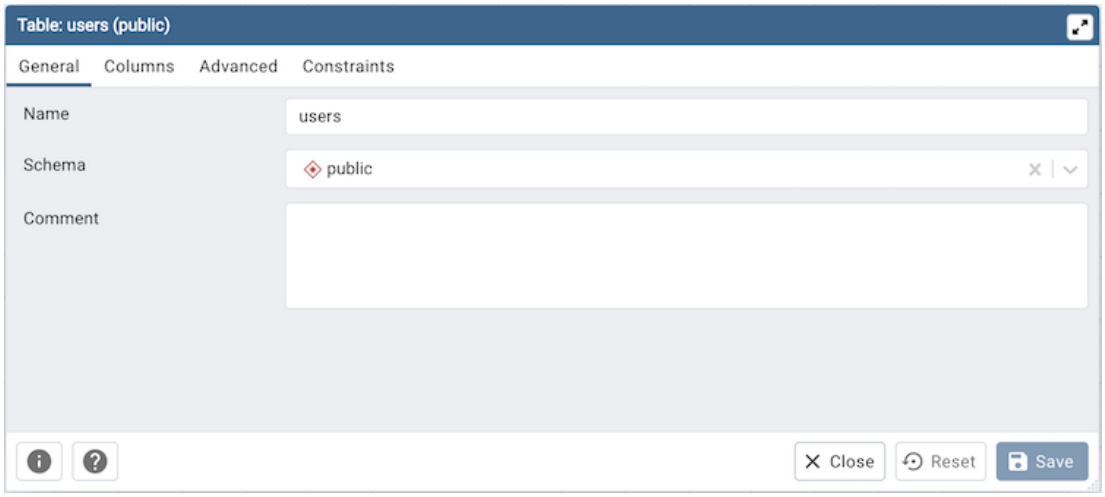
Show details(Tafsilotlarni ko‘rsatish) - Ustun tafsilotlari ko‘rinishini almashtirish uchun ushbu tugmani bosing. Bu sizga bir nechta yoki bir nechta ustun tafsilotlarini ko‘rsatish imkonini beradi - Option/Alt + Shift + D.

Zoom parametrlari

Zoom to fit (Siqish uchun kattalashtiring) - Avtomatik ravishda kattalashtirish/kichraytirish va barcha jadvallarni ko‘rinishga moslashtirish uchun ushbu tugmani bosing - Option/Alt + Shift + F.

Zoom in (Kattalashtirish) - Diagrammani kattalashtirish uchun ushbu tugmani bosing - Option / Alt + Shift + "+".

Zoom out (Kichraytirish) - Diagrammani kichraytirish uchun ushbu tugmani bosing - Option / Alt + Shift + " - ".

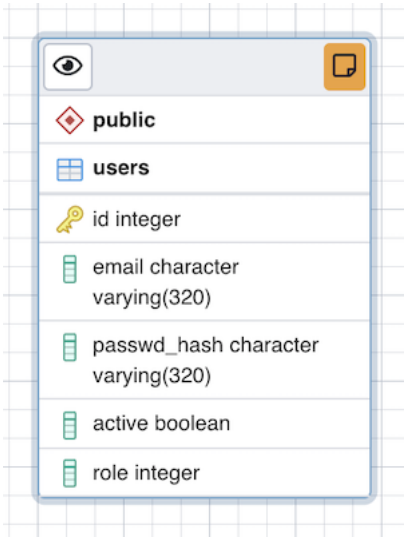


2.2-rasm. Yangi jadval qo‘shish oynasi

Jadval qo‘shish oynasi sizga quyidagilarga imkon beradi:

- Jadval tuzilishi tafsilotlarini o‘zgartiring.
- U mavjud jadvalni tahrirlash yoki yangisini qo‘shish uchun ishlatilishi mumkin.
- Turli sohalar bo‘yicha ma’lumot olish uchun jadval dialog oynasiga qarang.

Jadvalga izoh qo‘shish:



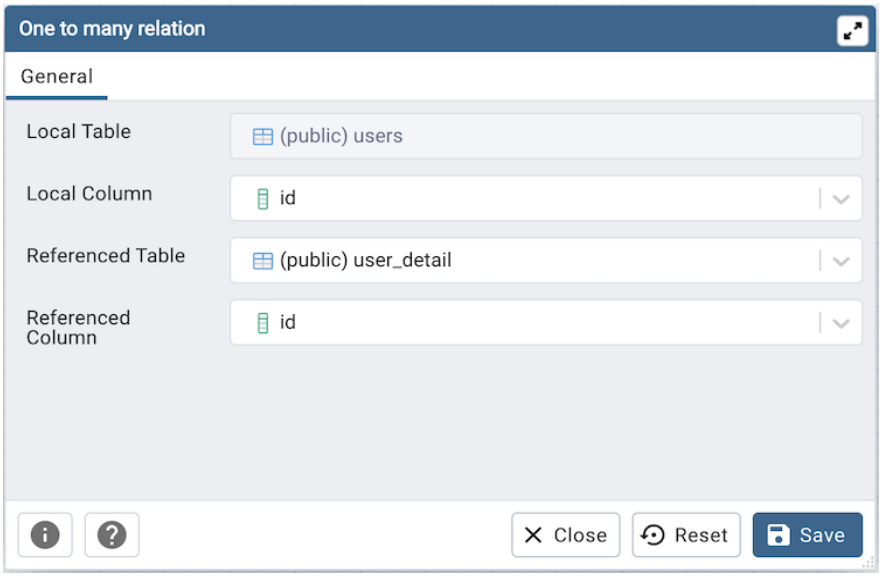
2.3-rasm. Jadval ustunlar ko‘rinishi

Jadval tafsilotlarini grafik tasvirda ko‘rsatadi:

- Yuqori satrda ustun tafsilotlar ko‘rinishini almashtirish uchun foydalaniladigan tafsilotlarni almashtirish tugmasi mavjud. Bundan tashqari, eslatma tugmasi mavjud, u faqat eslatma qo‘shilgan bo‘lsa

koʻrinadi. Eslatmani tezda oʻzgartirish uchun ushbu tugmani bosishingiz mumkin.

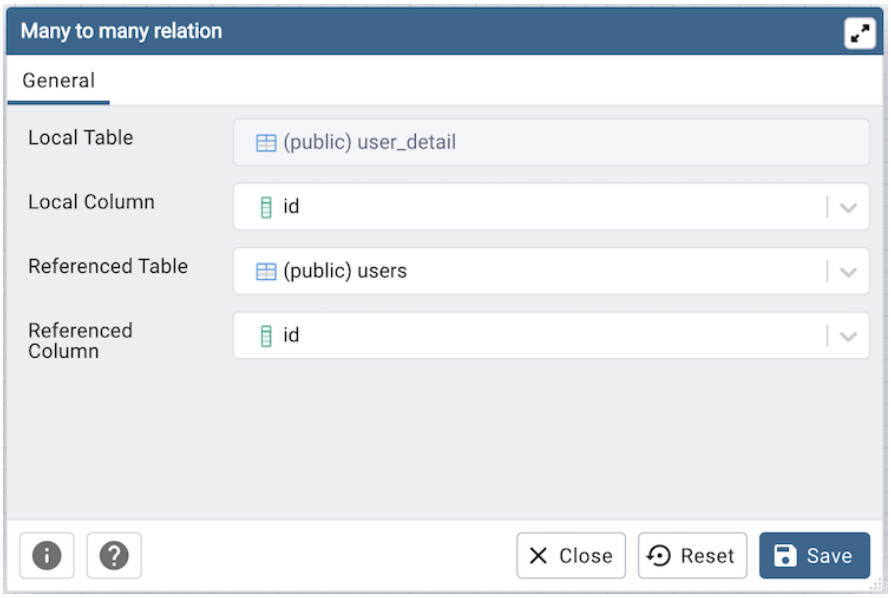
- Birinchi qatorda jadvalning sxema nomi koʻrsatilgan. Masalan. yuqoridagi rasmda ommaviy.
- Ikkinchi qatorda jadval nomi koʻrsatilgan. Masalan, yuqoridagi rasmdagi foydalanuvchilar.
- Jadval nomi ostidagi barcha boshqa qatorlar ma’lumotlar turi bilan birga jadval ustunlaridir. Agar ustun asosiy kalit boʻlsa, unda qulflash tugmasi belgisi boʻladi, masalan. id yuqoridagi rasmdagi asosiy kalitdir. Aks holda, u ustun belgisiga ega boʻladi. Tugunni bosishingiz va harakatlanish uchun sudrab olishingiz mumkin.
- Jadval tugunini ikki marta bosgandan soʻng yoki asboblar panelidagi tahrirlash tugmachasini bosgandan soʻng, jadval ma’lumotlarini oʻzgartirishingiz mumkin boʻlgan jadval dialog oynasi ochiladi. Turli sohalar boʻyicha ma’lumot olish uchun jadval dialog oynasiga qarang.



2.4-rasm. “Birdan koʻpga havola” boʻlimi

Birdan koʻpga havola dialog oynasi sizga quyidagilarga imkon beradi:

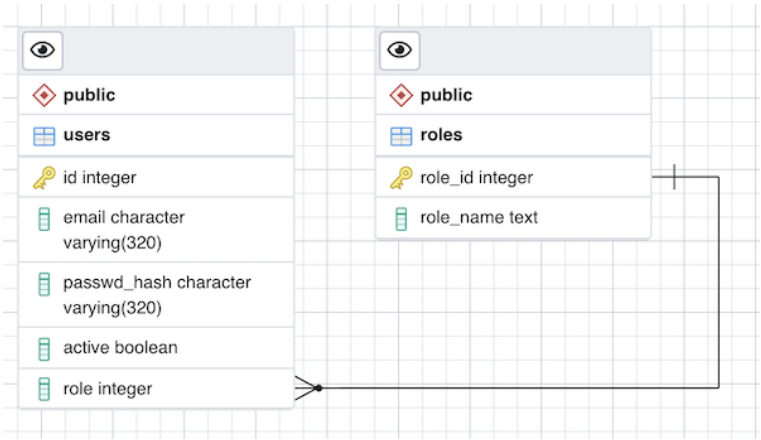
- Ikki jadval oʻrtasida xorijiy kalit aloqasini qoʻshing.
- Mahalliy jadval - bu jadvalga havola qiladigan va koʻplab yakuniy nuqtaga ega boʻlgan jadval.
- Mahalliy ustun havola qilingan ustun.
- Yoʻnaltirilgan jadval - bu havola qilinayotgan va bitta yakuniy nuqtaga ega jadval.
- Havola qilingan ustunga havola qilinayotgan ustun.



2.5-rasm. “Ko‘pdan ko‘pga” aloqasi oynasi

Ko‘pdan ko‘pga havola dialog oynasi sizga quyidagilarga imkon beradi:

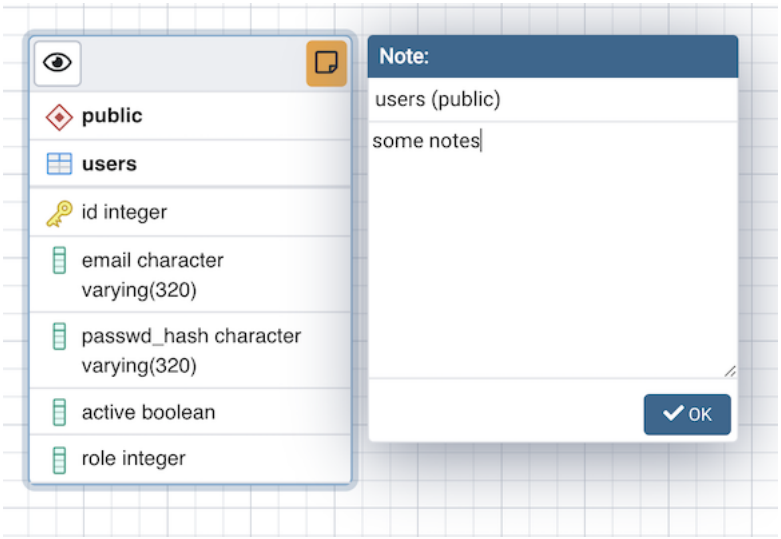
- Ikki jadval o‘rtasida ko‘p va ko‘p munosabatlarni qo‘shing.
- U ikkita jadvaldan olingan ustunlarga ega bo‘lgan munosabatlar jadvallarini yaratadi va ularni jadvallar bilan bog‘laydi.
- Chap jadval bog‘lanishi kerak bo‘lgan birinchi jadvaldir. U yangi munosabatlar jadvali bilan havolaning bitta so‘nggi nuqtasini oladi.
- Chap ustun birinchi jadvalning ustuni bo‘lib, u har doim asosiy kalit bo‘ladi.
- O‘ng jadval - bu bog‘lanishi kerak bo‘lgan ikkinchi jadval. U yangi munosabatlar jadvali bilan havolaning bitta so‘nggi nuqtasini oladi.
- O‘ng ustun ikkinchi jadval ustuni, bu har doim asosiy kalit bo‘ladi.



2.6-rasm. Jadval orasidagi aloqa

Jadval havolasi jadvallar o‘rtasidagi munosabatlarni ko‘rsatadi:

- Havolaning bitta satr so‘nggi nuqtasi havola qilinayotgan ustunni ko‘rsatadi.
- Bog‘lanishning uch qatorli so‘nggi nuqtasi tegishli ustunni ko‘rsatadi.
- Agar havola qilinayotgan yoki tegishli ustunlardan biri jadvaldan olib tashlansa, havola o‘chiriladi.
- havolani bosishingiz va tuvalda harakatlanish uchun sudrab olishingiz mumkin.



2.7-rasm. Jadvalga izoh qo‘shish oynasi

- Ma’lumotlar bazasini loyihalashda ba'zi eslatmalarni belgilash uchun qalqib chiquvchi qayd oynasidan foydalanishingiz mumkin.
- Siz asboblar panelidagi eslatma tugmasi yordamida qalqib chiquvchi oynani ochasiz.

Agar jadvalga biron bir eslatma qo‘shilsa, unda jadval tugunida eslatmalar tugmasi bo‘ladi. Eslatmalarni tekshirish/yangilash uchun tugmani bosishingiz mumkin

Nazorat savollari.

1. PostgreSQL afzalliklari haqida nimalarni bilasiz?
2. Ma‘lumotlarni modellarini tasvirlash qanday amalga oshiriladi?
3. PostgreSQL modelini bazaga sinxronizatsiya qilish tartibi?

3- laboratoriya ishi
Loyihalanayotgan ma‘lumotlar bazasi vositalaridan foydalangan holda misollar yechish (PostgreSQL asoslari)

Ishdan maqsad: ma’lumotlar bazasini boshqarish dasturiy vositasi yordamida serverga ulanishni sozlash va serverdagi ma’lumotlarni qayta ishlash, tahrirlash ko‘nikmalariga ega bo‘lish.

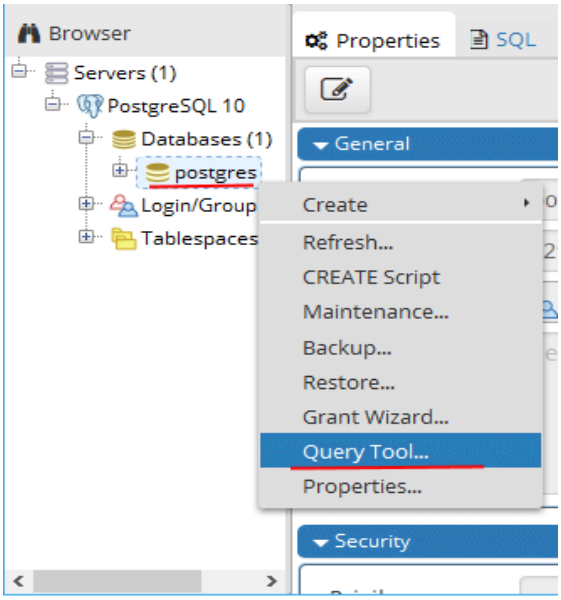
Masalaning qo‘yilishi: ma’lumotlar bazasini boshqarish vositalari yordamida serverdagi ma’lumotlarni qayta ishlash.

Uslubiy ko‘rsatmalar: Oldingi darsda PostgreSQL dasturini o‘rganish, ma’lumotlar modelini ishlab chiqish va tashkilot bilan munosabatlarning ERD diagrammasini yaratishning asosiy prinsiplari misolini ko‘rsatdik. Endigi navbat PostgreSQLni amalda qo‘llashga keldi, shuning uchun bu darsda serverga ulanishni yaratish, mwb modelini serverga yuklash, ish davomida ma’lumotlar sxemasi yangilanishlarini sinxronlashtirish, shuningdek PostgreSQLda ma’lumotlar bazasini yaratishni ko‘rib chiqamiz.

Ma’lumotlar bazasini yaratish uchun CREATE DATABASE buyrug‘idan so‘ng ma’lumotlar bazasi nomidan foydalaning.

So‘rovlarni bajarish uchun pgAdmin grafik mijozidan foydalanamiz, ammo psql konsol mijozidan ham foydalanish mumkin.

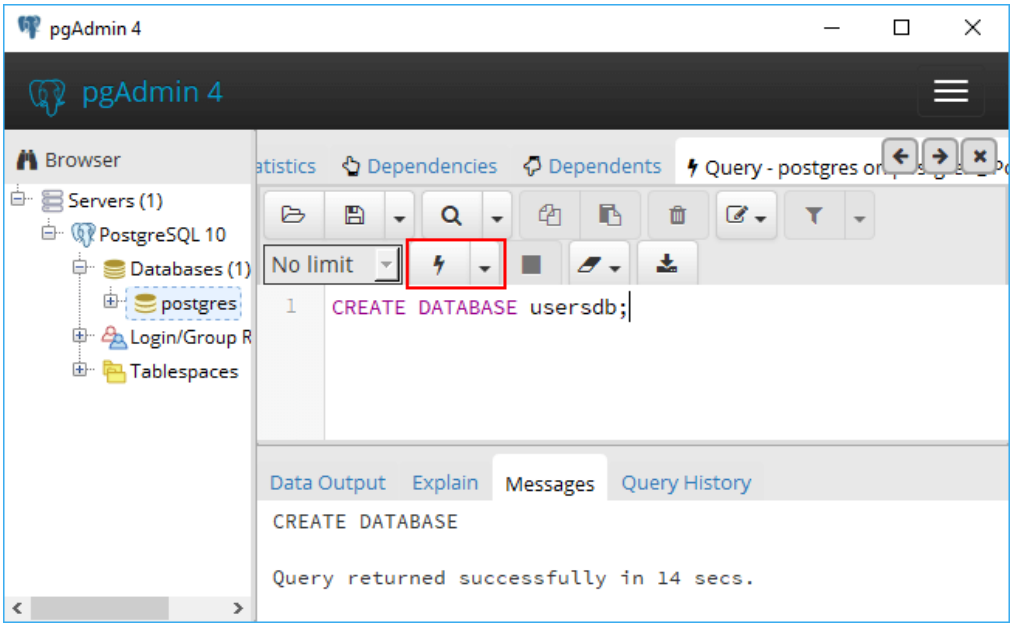
Yangi ma’lumotlar bazasini yaratish uchun pgAdmin-ni oching. Dasturning chap tomonida ma’lumotlar bazasini tanlang, masalan, standart postgres ma’lumotlar bazasi va ustiga o‘ng tugmasini bosing.



3.1-rasm. So‘rovlar oynasini ochish

Ko‘rsatilgan menyuda **Query Tool...** bandini tanlang va dasturning markaziy qismida SQL kodini kiritish maydoni ochiladi. Ushbu maydonga quyidagi kodni kiriting:

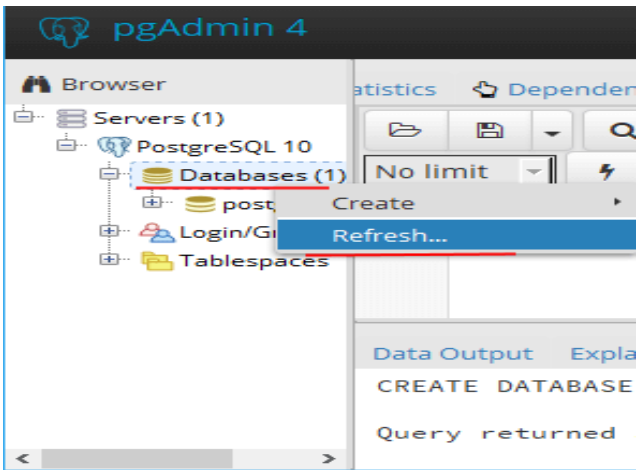
```
1    CREATE DATABASE usersdb;
```



3.2-rasm. So‘rovlar oynasi

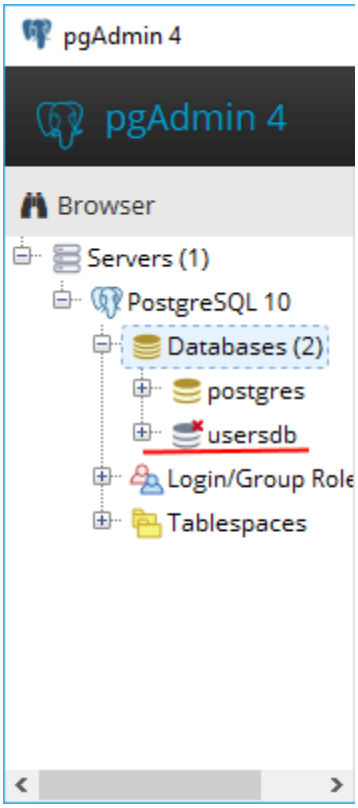
Kodni bajarish uchun chaqmoq ustiga bosing va shundan so‘ng usersdb ma’lumotlar bazasi yaratiladi.

Ma’lumotlar bazamizni ko‘rish uchun chap qismdagi Ma’lumotlar bazalari bo‘limiga sichqonchaning o‘ng tugmachasini bosing va kontekst menyusida **Yangilash...** ni tanlang:



3.3-rasm. Ma’lumotlar bazasini yangilash

Yangilanish sodir bo‘ladi va biz yaratilgan ma’lumotlar bazasini ko‘ramiz.



3.4-rasm. Yangi qo‘shilgan bazani ko‘rish

Odatiy bo‘lib, baza faol emas, shuning uchun uning belgisi kulrang. Lekin unga ulanish uchun uni bosish va uning tugunini ochish kifoya.

Ma’lumotlar bazasini o‘chirish uchun DROP DATABASE buyrug‘idan keyin ma’lumotlar bazasi nomidan foydalaning.

O‘chiriladigan ma’lumotlar bazasi inaktiv bo‘lishi kerak, ya'ni unga ulanish yopiq bo‘lishi kerak.

Masalan, usersdb ma’lumotlar bazasini o‘chirish:

```
1.  DROP DATABASE usersdb;
```

Jadvallarni yaratish uchun CREATE TABLE buyrug‘idan so‘ng jadval nomidan foydalaning. Jadval ustunlari va ularning atributlarini belgilaydigan ushbu buyruq bilan ishlatilishi mumkin bo‘lgan bir qancha operatorlar ham mavjud. Jadval yaratishning umumiy sintaksisi quyidagicha:

CREATE TABLE < table name >

(Ccolumn name> <data type> <column constraint,

Ccolumn name> <data type> <column constraint ...
<table constraint (<column name> [, Ccolumn name>])...)

Barcha ustunlar uchun spetsifikatsiya jadval nomidan keyin qavs ichida keltirilgan. Bundan tashqari, har bir ustun uchun u taqdim etadigan ma’lumotlarning nomi va turi ko‘rsatilishi kerak. Ma’lumotlar turi ustunda qanday ma’lumotlar (raqamlar, satrlar va boshqalar) bo‘lishi mumkinligini aniqlaydi.

Masalan, pgAdmin orqali ma’lumotlar bazasida jadval tuzamiz. Buni amalga oshirish uchun avval pgAdmin-da maqsadli ma’lumotlar bazasini tanlang, ustiga sichqonchaning o‘ng tugmachasini bosing va kontekst menyusidagi So‘rovlar vositasi ... bandini tanlang. So‘rovlar oynasini ochib olamiz (3.1-rasm).

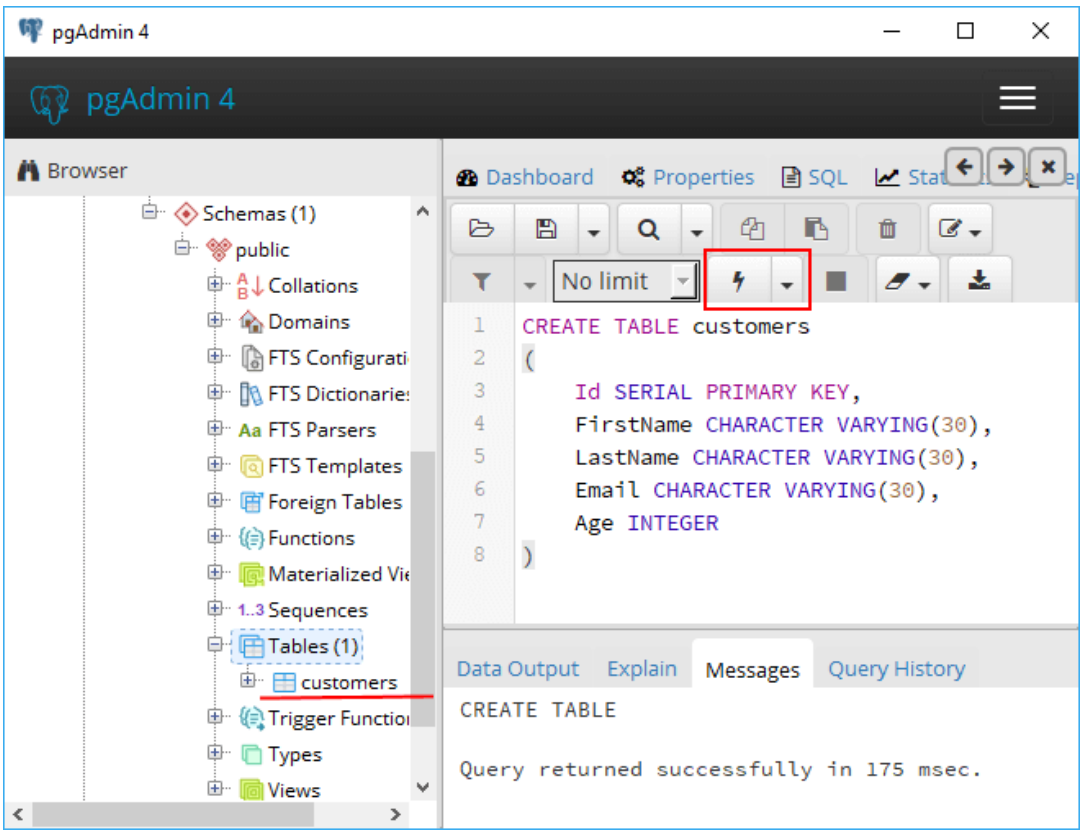
Keyinchalik, dasturning markaziy qismida ochiladigan maydonga quyidagi iboralar to‘plamini kiritamiz:

```
1.  CREATE TABLE customers
2.  (
3.      Id SERIAL PRIMARY KEY,
4.      FirstName CHARACTER VARYING(30),
5.      LastName CHARACTER VARYING(30),
6.      Email CHARACTER VARYING(30),
7.      Age INTEGER
8.  );
```

Bu holda, Mijozlar jadvali beshta ustunni belgilaydi: Id, Ism, Familiya, Yosh, Elektron pochta. Birinchi ustun, Id, mijoz identifikatorini ifodalaydi, u asosiy kalit bo‘lib xizmat qiladi va shuning uchun SERIAL turiga kiradi. Aslida, bu ustunda 1, 2, 3 va hokazo raqamli qiymat saqlanadi, bu har bir yangi qator uchun avtomatik ravishda bittaga ortadi.

Keyingi uchta ustun mijozning ismi, familiyasi va elektron pochta manzilini ifodalaydi va CHARACTER VARYING (30) turiga ega, ya'ni ular 30 ta belgidan ortiq bo‘lmagan qatorni ifodalaydi.

Oxirgi ustun Yosh foydalanuvchining yoshini bildiradi va INTEGER turiga ega, ya'ni u raqamlarni saqlaydi.



3.5-rasm. Kiritilgan SQL so‘rovlarni ishga tushirish oynasi

Va bu buyruqni bajargandan so‘ng, mijozlar jadvali tanlangan ma’lumotlar bazasiga qo‘shiladi.

Jadvallarni o‘chirish

Jadvallarni o‘chirish uchun quyidagi sintaksisga ega DROP TABLE buyrug‘idan foydalaning:

```
1.  DROP TABLE table1 [, table2, ...];
```

Masalan, mijozlar jadvalini o‘chirish:

```
1.  DROP TABLE customers;
```

Maydonga bo‘sh (NULL) qiymatlar kiritilishining oldini olish uchun CREATE TABLE komandasida NOT NULL cheklanishi ishlatiladi. Bu cheklanish faqat har xil ustunlar uchun o‘rnatiladi.

Masalan, shu narsa aniqki, birlamchi kalitlar hech qachon bo‘sh bo‘lmasliklari kerak, shuning uchun **employee** jadvalini quyidagicha yaratish mumkin:

CREATE TABLE Employee

(id integer NOT NULL,
name varchar (45),
birthday date,
seriya varchar(45),
address varchar(255),
phone varchar(15),
staji int,
status tinyint(1),
is_delete tinyint(1))

Ko‘p hollarda ustunga kiritilgan qiymatlar bir-biridan farq qilishi kerak. Agar ustun uchun **UNIQUE** cheklanishi o‘rnatilsa, bu ustunga mavjud qiymatni kiritishga urinish rad etiladi.

Bu cheklanish bo‘sh bo‘lmaydigan (NOT NULL) deb e’lon qilingan maydonlarga qo‘llanishi mumkin.

Masalan:

CREATE TABLE Employee
(id integer NOT NULL UNIQUE,
name varchar (45),
birthday date,
seriya varchar(45),
address varchar(255),
phone varchar(15),
staji int,
status tinyint(1),
is_delete tinyint(1))

Unikalligi talab qilinadigan maydonlar (birlamchi kalitlardan tashqari) kandidat kalitlar yoki unikal kalitlar deyiladi. Jadval cheklanishi ***UNIQUE*** maydonlar guruhiga o‘rnatilishi mumkin. Bu bir necha maydonlar qiymatlari kombinatsiyasi unikalligini ta’minlaydi. Bizning ma’lumotlar bazamizda har bir mijoz bitta xodimga biriktirilgan. Ya’ni Mijozlar jadvalida mijoz nomeri (mid) va xodim nomeri (hid) kombinatsiyasi unikal bo‘lishi kerak. Bu cheklanishni ***UNIQUE*** (mid, hid) yordamida, xaridorlar jadvalini yaratishda kiritish mumkin. Bu ustunlar uchun ***NOT NULL*** cheklanishini kiritish zarurdir.

SQL birlamchi kalitlarni to‘g‘ridan to‘g‘ri birlamchi kalit (***PRIMARY KEY***) cheklanishi orqali ta’riflaydi. ***PRIMARY KEY*** jadvalni yoki ustunlarni cheklashi mumkin. Bu cheklanish ***UNIQUE*** cheklanishi kabi ishlaydi, jadval uchun faqat bitta birlamchi kalit (ixtiyoriy sondagi ustunlar uchun) aniqlanishi mumkin bo‘lgan holdan tashqari. Birlamchi kalitlar

NULL qiymatga ega bo‘lishi mumkin emas.

Misol:

```
CREATE TABLE Employee
(id integer NOT NULL PRIMARY KEY,
name varchar (45),
birthday date,
seriya varchar(45),
address varchar(255),
phone varchar(15),
staji int,
status tinyint(1),
is_delete tinyint(1)
)
```

PRIMARY KEY cheklanishi qiymatlar unikal kombinatsiyasini tashkil qiluvchi bir necha maydonlar uchun qo‘llanishi mumkin. Masalan ***PRIMARY KEY*** cheklanishini juftliklar uchun qo‘llash mumkin:

```
CREATE TABLE Employee
(id integer NOT NULL,
name varchar (45),
birthday date,
seriya varchar(45),
address varchar(255),
phone varchar(15),
staji int,
status tinyint(1),
is_delete tinyint(1),
PRIMARY KEY (id, seriya)
)
```

CHECK cheklanishi jadvalga kiritilayotgan ma’lumot qabul qilinishidan oldin mos kelishi lozim bo‘lgan shart kiritishga imkon beradi. ***CHECK*** cheklanishi ***CHECK*** kalit so‘zi ko‘rsatilgan maydondan foydalanuvchi predikat ifodalaridan iboratdir.

Misol: ***Employee*** jadvali ***status*** ustuniga kiritilayotgan qiymat 1 dan katta yoki teng bo‘lish sharti.

```
CREATE TABLE Employee
(id integer NOT NULL PRIMARY KEY,
name varchar (45),
birthday date,
seriya varchar(45),
```

address varchar(255),
phone varchar(15),
staji int,
status tinyint(1) CHECK(status>=1),
is_delete tinyint(1)
)

CHECK cheklanishidan maydonga ma’lum qiymatlarini kiritishdan himoya qilib, xatolar oldini olish uchun foydalanish mumkin. Masalan xodimlar adresi faqat Toshkent, Buxoro, Samarqand va Guliston shaharlar bo‘lsin.

CREATE TABLE Employee
(id integer NOT NULL PRIMARY KEY,
name varchar (45),
birthday date,
seriya varchar(45),
address varchar(255) CHECK (address in("Toshkent","Buxoro","Samarqand","Guliston"),
phone varchar(15),
staji int,
status tinyint(1) CHECK(status>=1),
is_delete tinyint(1)
)

CHECK jadval cheklanishi sifatida kelishi mumkin. Bu shartga bir necha maydon kiritishga imkon beradi.

Masalan:

CREATE TABLE Employee
(id integer NOT NULL PRIMARY KEY,
name varchar (45),
birthday date,
seriya varchar(45),
address varchar(255),
phone varchar(15),
staji int,
status tinyint(1),
is_delete tinyint(1),
CHECK (staji >=1 or address in("Toshkent","Buxoro","Samarqand","Guliston")
)

Biror bir maydon uchun qiymat ko‘rsatmagan holda jadvalga satr qo‘shsangiz, SQL bunday maydonga kiritish uchun ko‘zda tutilgan qiymatga ega bo‘lishi kerak, aks holda komanda rad etiladi. Eng umumiy ko‘zda tutilgan qiymat ***NULL*** qiymatdir. ***CREATE TABLE*** komandasida ko‘zda tutilgan qiymat ***DEFAULT*** operatori orqali, ustun cheklanishi sifatida ko‘rsatiladi.

Masalan:

```
CREATE TABLE Employee
(id integer NOT NULL PRIMARY KEY,
name varchar (45),
birthday date,
seriya varchar(45),
address varchar(255),
phone varchar(15),
staji int,
status tinyint(1) ) DEFAULT "NULL",
is_delete tinyint(1),
 CHECK (staji >=1 or address
in("Toshkent","Buxoro","Samarqand","Guliston")
)
```

Jadval bir maydonidagi hamma qiymatlar boshqa jadval maydonida aks etsa, birinchi maydon ikkinchisiga ilova qiladi deyiladi. Bu ikki maydon orasidagi bog‘liqlikni ko‘rsatadi. Masalan, buyurtmachilar jadvalida har bir buyurtmachi, sotuvchilar jadvalida o‘ziga biriktirilgan sotuvchiga ilova qiluvchi ***SNum*** maydoniga ega. Bir maydon ikkinchisiga ilova qilsa tashqi kalit, u ilova qilayotgan maydon ajdod kalit deyiladi. Buyurtmachilar jadvalidagi SNum maydoni tashqi kalit, sotuvchilar jadvalidagi ***SNum*** - ajdod kalitdir.

Tashqi kalit bitta maydondan iborat bo‘lishi shart emas. Birlamchi kalit kabi, tashqi kalit bitta modul sifatida qayta ishlanuvchi bir necha maydonlarga ega bo‘lishi mumkin. Maydon tashqi kalit bo‘lsa, ilova qilayotgan jadval bilan ma’lum usulda bog‘liqdir. Tashqi kalit har bir qiymati (satri) ajdod kalitning bitta va faqat bitta qiymatiga (satriga) ilova qilishi kerak. Bu holda tizim ilovali yaxlit holatda deyiladi.

Shu bilan birga ajdod kalit qiymati tashqi kalit bir necha qiymatlariga ilova qilishi mumkin.

Cheklanish **FOREIGN KEY**.

SQL ilovali yaxlitlikni **FOREIGN KEY** yordamida ta’minlaydi. Tashqi kalit vazifasi ajdod kalitda ko‘rsatilmagan qiymatlarni tashqi kalit

maydonlariga kiritmaslikdir. **FOREIGN KEY** cheklanishi sintaksisi:

FOREIGN KEY <column list> REFERENCES
<pktable> [<column list>]

Birinchi ro‘yxat komanda tomonidan o‘zgartiriluvchi ustunlar ro‘yxatidir. **Pktable** - bu ajdod kalitli jadval. Ikkinchi ustunlar ro‘yxati bu ajdod kalitni tashkil qiluvchi ustunlardir.

Misol uchun Xodimlar jadvaliga ilova qiluvchi tashqi kalit sifatida e’lon qilingan **MNum** maydoniga ega bo‘lgan Mijozlar jadvalini yaratamiz:

CREATE TABLE Clients
(id integer NOT NULL PRIMARY KEY,
name varchar (45),
address varchar(255),
phone varchar(15),
MNum *integer*,
status tinyint(1)) DEFAULT "NULL",
is_delete tinyint(1),
FOREIGN KEY (MNum) REFERENCES Employee (MNum))

Tashqi kalitni ustunlar cheklanishi sifatida berish mumkin. Buning uchun ***FOREIGN KEY*** ko‘rinishi — ko‘rsatkichli cheklanish (***REFERENCES***) qo‘llanadi:

FOREIGN KEY cheklanishidan jadval yoki ustun cheklanishi sifatida foydalanganda, agar ular ***PRIMARY KEY*** cheklanishiga ega bo‘lsa, ajdod kalit ustunlarini ko‘rsatmaslik mumkin.

SQL-da jadvallar va ularning ustunlarini belgilashda biz ma’lum cheklovlar qo‘yadigan bir qator atributlardan foydalanishimiz mumkin. Keling, ushbu atributlarni ko‘rib chiqaylik.

PRIMARY KEY ifodasi ustunni asosiy kalit qilish uchun ishlatilishi mumkin.

```
1.  CREATE TABLE Customers
2.  (
3.      Id SERIAL PRIMARY KEY,
4.      FirstName CHARACTER VARYING(30),
5.      LastName CHARACTER VARYING(30),
6.      Email CHARACTER VARYING(30),
7.      Age INTEGER
8.  )
```

Birlamchi kalit jadvaldagi qatorni noyob tarzda aniqlaydi. Birlamchi kalit SERIAL ustunlar bo‘lishi shart emas; ular boshqa har qanday turni ifodalashi mumkin.

Asosiy kalitni jadval darajasida o‘rnatish:

```
1.  CREATE TABLE Customers
2.  (
3.      Id SERIAL,
4.      FirstName CHARACTER VARYING(30),
5.      LastName CHARACTER VARYING(30),
6.      Email CHARACTER VARYING(30),
7.      Age INTEGER,
8.      PRIMARY KEY(Id)
9.  );
```

Asosiy kalit murakkab kalit bo‘lishi mumkin. Jadvaldagi satrni yagona aniqlash uchun bir vaqtning o‘zida ikkita ustunga ega bo‘lsak, bunday kalit talab qilinishi mumkin. Masalan:

```
1.  CREATE TABLE OrderLines
2.  (
3.      OrderId INTEGER,
4.      ProductId INTEGER,
5.      Quantity INTEGER,
6.      Price MONEY,
7.      PRIMARY KEY(OrderId, ProductId)
8.  );
```

Bu yerda OrderId va ProductId maydonlari birgalikda birlamchi asosiy kalit vazifasini bajaradi. Ya'ni, OrderLines jadvalida bu ikkala maydon bir vaqtning o‘zida bir xil qiymatlarga ega bo‘lgan ikkita qator bo‘lishi mumkin emas.

UNIQUE

Agar ustun faqat noyob qiymatlarga ega bo‘lishini istasak, u uchun UNIQUE atributini aniqlash mumkin.

```
1.  CREATE TABLE Customers
2.  (
3.      Id SERIAL PRIMARY KEY,
4.      FirstName CHARACTER VARYING(20),
5.      LastName CHARACTER VARYING(20),
6.      Email CHARACTER VARYING(30) UNIQUE,
```

```
7.        Phone CHARACTER VARYING(30) UNIQUE,
8.        Age INTEGER
9.    );
```

Bunday holda, elektron pochta manzili va telefon raqamini ko‘rsatadigan ustunlar noyob qiymatlarga ega bo‘ladi. Va biz ushbu ustunlar uchun bir xil qiymatlar bilan jadvalga ikkita qator qo‘sha olmaymiz.

Ushbu atributni jadval darajasida ham aniqlashimiz mumkin:

```
1.    CREATE TABLE Customers
2.    (
3.        Id SERIAL PRIMARY KEY,
4.        FirstName CHARACTER VARYING(20),
5.        LastName CHARACTER VARYING(20),
6.        Email CHARACTER VARYING(30),
7.        Phone CHARACTER VARYING(30),
8.        Age INTEGER,
9.        UNIQUE(Email, Phone)
10.   );
```

Yoki shunday:

```
1.    CREATE TABLE Customers
2.    (
3.        Id SERIAL PRIMARY KEY,
4.        FirstName CHARACTER VARYING(20),
5.        LastName CHARACTER VARYING(20),
6.        Email CHARACTER VARYING(30),
7.        Phone CHARACTER VARYING(30),
8.        Age INTEGER,
9.        UNIQUE(Email),
10.       UNIQUE(Phone)
11.   );
```

NULL va NOT NULL

Ustun NULL bo‘lishi mumkinligini ko‘rsatish uchun ustunni belgilashda ustun atributini NULL yoki NOT NULL ga o‘rnatishingiz mumkin. Agar bu atribut aniq ishlatilmasa, ustun sukut bo‘yicha null bo‘ladi. Ustun birlamchi kalit sifatida ishlayotganida istisno - bu holda, sukut bo‘yicha ustun NULL EMAS.

```
1.   CREATE TABLE Customers
2.   (
3.       Id SERIAL PRIMARY KEY,
4.       FirstName CHARACTER VARYING(20) NOT NULL,
5.       LastName CHARACTER VARYING(20) NOT NULL,
6.       Age INTEGER
7.   );
```

DEFAULT

DEFAULT atributi ustun uchun standart qiymatni belgilaydi. Agar ma’lumotlarni qo‘shganda ustun uchun qiymat ko‘rsatilmagan bo‘lsa, u uchun standart qiymat ishlatiladi.

```
1.   CREATE TABLE Customers
2.   (
3.       Id SERIAL PRIMARY KEY,
4.       FirstName CHARACTER VARYING(20),
5.       LastName CHARACTER VARYING(20),
6.       Age INTEGER DEFAULT 18
7.   );
```

Bu yerda Yosh ustunida standart qiymat 18 ga teng.

CHECK

CHECK kalit so‘zi ustunda saqlanishi mumkin bo‘lgan qiymatlar oralig‘ida chegarani belgilaydi. Buning uchun CHECK so‘zidan keyin ustun yoki bir nechta ustunlar mos kelishi sharti qavs ichida ko‘rsatiladi. Masalan, mijozlarning yoshi 0 dan kam yoki 100 dan oshmasligi kerak:

```
1.   CREATE TABLE Customers
2.   (
3.       Id SERIAL PRIMARY KEY,
4.       FirstName CHARACTER VARYING(20),
5.       LastName CHARACTER VARYING(20),
6.       Age  INTEGER DEFAULT 18  CHECK(Age  >0  AND Age  <
7. 100),
8.       Email  CHARACTER  VARYING(30)  UNIQUE  CHECK(Email
9. !=''),
10.      Phone  CHARACTER  VARYING(20)  UNIQUE  CHECK(Phone
11. !='')
12.  );
```

Shuningdek, u E-pochta va Telefon ustunlari qiymat sifatida bo‘sh qatorga ega bo‘lmasligini belgilaydi (bo‘sh satr NULLga teng emas).

AND kalit so‘zi shartlarni birlashtirish uchun ishlatiladi. Shartlar katta (>), kichik (<), teng emas (! =) taqqoslash amallari shaklida ko‘rsatilishi mumkin.

Bundan tashqari, CHECK-dan foydalanib, siz butun jadval uchun cheklov yaratishingiz mumkin:

```
1.   CREATE TABLE Customers
2.   (
3.       Id SERIAL PRIMARY KEY,
4.       Age INTEGER DEFAULT 18,
5.       FirstName CHARACTER VARYING(20),
6.       LastName CHARACTER VARYING(20),
7.       Email CHARACTER VARYING(30) UNIQUE,
8.       Phone CHARACTER VARYING(20) UNIQUE,
9.       CHECK((Age >0 AND Age<100) AND (Email !='') AND
10. (Phone !=''))
11.  );
```

CONSTRAINT bayonoti. Cheklovlar nomini o‘rnatish.

CONSTRAINT kalit so‘zidan cheklovlar nomini belgilash uchun foydalanish mumkin. PRIMARY KEY, UNIQUE, CHECK cheklovlar sifatida ishlatilishi mumkin.

Cheklov nomlari ustunlar darajasida belgilanishi mumkin. Ular CONSTRAINTdan keyin atributlardan oldin belgilanadi:

```
1.   CREATE TABLE Customers
2.   (
3.       Id SERIAL CONSTRAINT customer_Id PRIMARY KEY,
4.       Age   INTEGER   CONSTRAINT   customers_age_check
5. CHECK(Age >0 AND Age < 100),
6.       FirstName CHARACTER VARYING(20) NOT NULL,
7.       LastName CHARACTER VARYING(20) NOT NULL,
8.       Email    CHARACTER    VARYING(30)    CONSTRAINT
9. customers_email_key UNIQUE,
10.      Phone    CHARACTER    VARYING(20)    CONSTRAINT
11. customers_phone_key UNIQUE
12.  );
```

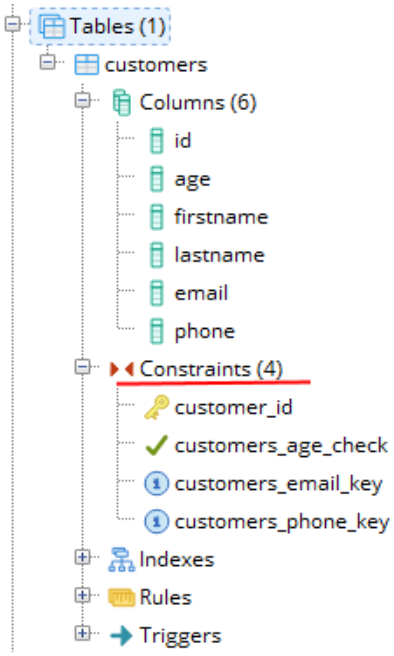
Prinsipial jihatdan cheklovlarning nomlarini ko‘rsatish shart emas, tegishli atributlarni o‘rnatganingizda, SQL Server avtomatik ravishda

ularning nomlarini aniqlaydi. Ammo, cheklovning nomini bilib, biz unga murojaat qilishimiz mumkin, masalan, uni olib tashlash uchun.

Shuningdek, siz cheklovlarning barcha nomlarini jadval atributlari orqali o‘rnatishingiz mumkin:

```
1.   CREATE TABLE Customers
2.   (
3.       Id SERIAL,
4.       Age INTEGER,
5.       FirstName CHARACTER VARYING(20) NOT NULL,
6.       LastName CHARACTER VARYING(20) NOT NULL,
7.       Email CHARACTER VARYING(30),
8.       Phone CHARACTER VARYING(20),
9.       CONSTRAINT customer_Id PRIMARY KEY(Id),
10.      CONSTRAINT customers_age_check CHECK(Age >0 AND
11. Age < 100),
12.      CONSTRAINT customers_email_key UNIQUE(Email),
13.      CONSTRAINT customers_phone_key UNIQUE(Phone)
14.  );
```

CONSTRAINT iborasi cheklovlar yaratish uchun ishlatiladimi yoki yo‘qligidan qat'i nazar (bu holda, cheklovlarni o‘rnatishda PostgreSQL ularni o‘zi nomlaydi), biz pgAdmin-dagi barcha cheklovlarni pastki tugundagi ma’lumotlar bazasi tugunida ko‘rishimiz mumkin:



3.6-rasm. CONSTRAINT iborasi cheklovlar yaratish oynasi.

4- laboratoriya ishi
SQL tilida cheklovlar turlari va o‘rnatish vositalaridan foydalanib masalalar yechish

Ishdan maqsad: ma’lumotlar bazasini boshqarish dasturiy vositasi yordamida ma’lumotlarni qayta ishlash, tahrirlash ko‘nikmalariga ega bo‘lish.

Masalaning qo‘yilishi: ma’lumotlar bazasini boshqarish vositalari yordamida ma’lumotlarni qayta ishlash.

Uslubiy ko‘rsatmalar: SQL tili operatorlarni erkin formatda yozishini ta'minlaydi. Buning ma’nosi, operatorlar elementlarini yozilishi ekrandan fiksirlangan joylarga bog‘liq emas. Komanda strukturasi bir qancha kalit xizmatchi so‘zlar bilan beriladi, masalan:

ALTER(изменять-o‘zgartirish**)**

INSERT (вставка-qo‘yish)

SELECT (выбрать-ajratib olish)

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

Mavjud jadvalni o‘zgartirish, xususan, ustunlarni qo‘shish yoki olib tashlash, ustunlar turini o‘zgartirish va hokazolar odatiy hol emas. Ya'ni jadvalning ta'rifini o‘zgartirishingiz kerak bo‘ladi. Buning uchun quyidagi rasmiy sintaksisga ega ALTER TABLE ifodasidan foydalaning:

```
1.   ALTER TABLE название_таблицы
2.   {     ADD    название_столбца     тип_данных_столбца
3. [ограничения_столбца] |
4.      DROP COLUMN название_столбца |
5.      ALTER COLUMN название_столбца параметры_столбца |
6.      ADD [CONSTRAINT] определение_ограничения |
7.      DROP [CONSTRAINT] имя_ограничения}
```

Keling, jadvalni o‘zgartirishning ba'zi imkoniyatlarini ko‘rib chiqaylik.

Yangi ustun qo‘shish

Mijozlar jadvaliga yangi "Telefon" ustunini qo‘shamiz:

```
1.   ALTER TABLE Customers
2.   ADD Phone  CHARACTER VARYING(20)  NULL;
```

Bu yerda Telefon ustuni CHARACTER VARYING (20) turida bo‘lib, uning uchun NULL atributi aniqlangan, ya’ni ustun null bo‘ladi. Agar null

bo‘lmasligi kerak bo‘lgan ustun qo‘shishimiz kerak bo‘lsa-chi? Agar jadvalda ma’lumotlar mavjud bo‘lsa, quyidagi buyruq bajarilmaydi:

```
1.   ALTER TABLE Customers
2.   ADD Address  CHARACTER VARYING(30)  NOT NULL;
```

Shuning uchun, bu holda, yechim DEFAULT atributi orqali standart qiymatni o‘rnatishdir:

```
1.   ALTER TABLE Customers
2.   ADD  Address  CHARACTER  VARYING(30)  NOT  NULL  DEFAULT
3. 'Неизвестно';
```

Ustunni o‘chirish

Mijozlar jadvalidan Manzil ustunini olib tashlaymiz:

```
1.   ALTER TABLE Customers
2.   DROP COLUMN Address;
```

Ustun turini o‘zgartirish

Turni o‘zgartirish uchun TYPE kalit so‘zi ishlatiladi. Mijozlar jadvalidagi FirstName ustunining ma’lumotlar turini VARCHAR (50) ga o‘zgartiramiz (aka VARYING CHARACTER (50)):

```
1.   ALTER TABLE Customers
2.   ALTER COLUMN FirstName  TYPE  VARCHAR(50);
```

Ustun cheklovlarini o‘zgartirish

Cheklov qo‘shish uchun SET iborasidan keyin cheklovdan foydalaning. Masalan, FirstName ustuni uchun NOT NULL cheklovini o‘rnatamiz:

```
1.   ALTER TABLE Customers
2.   ALTER COLUMN FirstName
3.   SET NOT NULL;
```

Cheklovni olib tashlash uchun DROP operatoridan keyin cheklovdan foydalaning. Masalan, yuqoridagi cheklovni olib tashlaymiz:

```
1.   ALTER TABLE Customers
2.   ALTER COLUMN FirstName
3.   DROP NOT NULL;
```

Jadval cheklovlarini o‘zgartirish

CHECK cheklovini qo‘shish:

```
1.  ALTER TABLE Customers
2.  ADD CHECK (Age > 0);
```

PRIMARY KEY asosiy kalitini qo‘shish:

```
1.  ALTER TABLE Customers
2.  ADD PRIMARY KEY (Id);
```

Bunday holda, jadvalda PRIMARY KEY chekloviga ega bo‘lmagan Id ustuni allaqachon mavjud deb taxmin qilinadi. Va yuqoridagi skript yordamida PRIMARY KEY cheklovi o‘rnatiladi.

UNIQUE cheklovini qo‘shish - elektron pochta ustuni uchun noyob qiymatlarni belgilang:

```
1.  ALTER TABLE Customers
2.  ADD UNIQUE (Email);
```

Cheklov qo‘shsangiz, ularning har biriga ma’lum nom beriladi. Masalan, CHECK uchun yuqorida qo‘shilgan cheklov customer_age_check deb nomlanadi. Cheklovlarning nomlarini pgAdmin orqali jadvalda ko‘rish mumkin.

CONSTRAINT iborasi yordamida cheklovni qo‘shganda, biz ham aniq nom berishimiz mumkin.

```
1.  ALTER TABLE Customers
2.  ADD CONSTRAINT phone_unique UNIQUE (Phone);
```

Bunday holda, cheklov "telefon_unique" deb nomlanadi.

Cheklovni olib tashlash uchun siz uning nomini bilishingiz kerak, bu DROP CONSTRAINT ifodasidan keyin belgilanadi. Masalan, yuqoridagi qo‘shilgan cheklovni olib tashlaymiz:

```
1.  ALTER TABLE Customers
2.  DROP CONSTRAINT phone_unique;
```

Ustun va jadval nomini o‘zgartirish

Manzil ustunining nomini Shaharga o‘zgartiring:

```
1.   ALTER TABLE Customers
2.   RENAME  COLUMN Address  TO City;
```

Mijozlar jadvalini Foydalanuvchilar deb o‘zgartiring:

```
1.   ALTER TABLE Customers
2.   RENAME  TO Users;
```

Ma’lumotlarni qo‘shish uchun quyidagi rasmiy sintaksisga ega bo‘lgan INSERT buyrug‘i ishlatiladi:

```
1.   INSERT  INTO  имя_таблицы  (столбец1,  столбец2,  ...
столбецN)
2.   VALUES (значение1, значение2, ... значениеN)
```

INSERT INTO - jadval nomidan so‘ng, siz ma’lumotlarni qo‘shishingiz kerak bo‘lgan vergul bilan ajratilgan barcha ustunlar qavs ichida ko‘rsatilgan. Va oxirida, VALUES so‘zidan keyin qo‘shilgan qiymatlar qavs ichida keltirilgan.

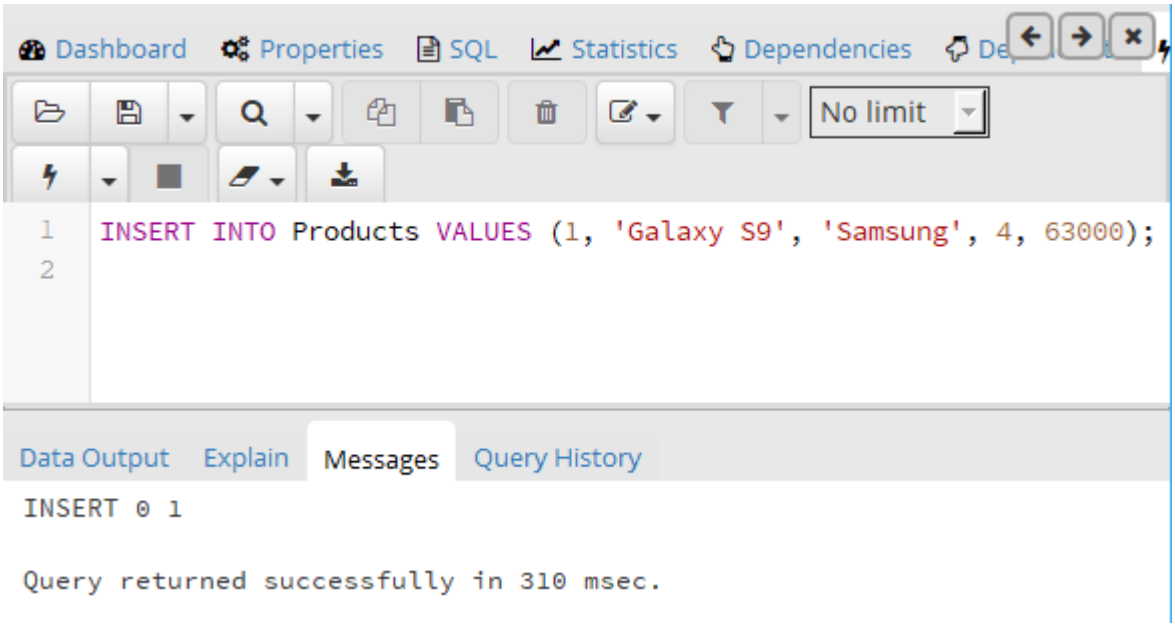
Aytaylik, bizning ma’lumotlar bazamizda quyidagi talitsa mavjud:

```
1.   CREATE TABLE Products
2.   (
3.        Id SERIAL PRIMARY KEY,
4.        ProductName VARCHAR(30) NOT NULL,
5.        Manufacturer VARCHAR(20) NOT NULL,
6.        ProductCount INTEGER DEFAULT 0,
7.        Price NUMERIC
8.   );
```

INSERT buyrug‘i yordamida unga bitta qator qo‘shamiz:

```
1.   INSERT  INTO  Products  VALUES  (1,  'Galaxy  S9',
'Samsung‘, 4, 63000)
```

pgAdmin-da muvaffaqiyatli bajarilgandan so‘ng, xabar maydonida "INSERT 0 1" xabari paydo bo‘lishi kerak:



4.1-rasm. SQL so‘rovlar oynasi

Shuni yodda tutish kerakki, VALUES kalit so‘zidan keyin qavs ichidagi ustunlar uchun qiymatlar e'lon qilingan tartibda uzatiladi. Masalan, yuqoridagi CREATE TABLE bayonotida siz birinchi ustun Id ekanligini ko‘rishingiz mumkin, shuning uchun bu ustunga 1 raqami beriladi. Ikkinchi ustun mahsulot nomi deb ataladi, shuning uchun ikkinchi qiymat, "Galaxy S9" qatori bo‘ladi. bu alohida ustunga o‘tdi va hokazo. Ya'ni, qiymatlar ustunlarga quyidagicha uzatiladi:

Id: 1

Mahsulot nomi: "Galaxy S9"

Ishlab chiqaruvchi: "Samsung"

Mahsulotlar soni: 4

Narxi: 63000

Bundan tashqari, qiymatlarni kiritishda siz qiymatlar qo‘shiladigan bevosita ustunlarni belgilashingiz mumkin:

```
1.  INSERT INTO Products (ProductName, Price,
Manufacturer)
2.  VALUES ('iPhone X', 71000, 'Apple');
```

Bu erda qiymat faqat uchta ustun uchun ko‘rsatilgan. Va endi qiymatlar ustunlar tartibida o‘tkaziladi:

Mahsulot nomi: iPhone X

Ishlab chiqaruvchi: "Apple"

Narxi: 71000

Id ustuni uchun qiymat ma’lumotlar bazasi tomonidan avtomatik ravishda yaratiladi, chunki u Seriya turini ifodalaydi. Ya'ni, oxirgi satrdagi qiymatga bittasi qo‘shiladi.

Qolgan ustunlar uchun DEFAULT atributi (masalan, ProductCount ustuni uchun) NULL o‘rnatilgan bo‘lsa, standart qiymat qo‘shiladi. Bunday holda, aniqlanmagan ustunlar (Serial tipidagilar bundan mustasno) null bo‘lishi yoki DEFAULT atributiga ega bo‘lishi kerak.

Agar birinchi misoldagi kabi maxsus ustunlar ko‘rsatilmagan bo‘lsa, jadvaldagi barcha ustunlar uchun qiymatlarni o‘tkazishimiz kerak.

Biz bir vaqtning o‘zida bir nechta qatorlarni qo‘shishimiz mumkin:

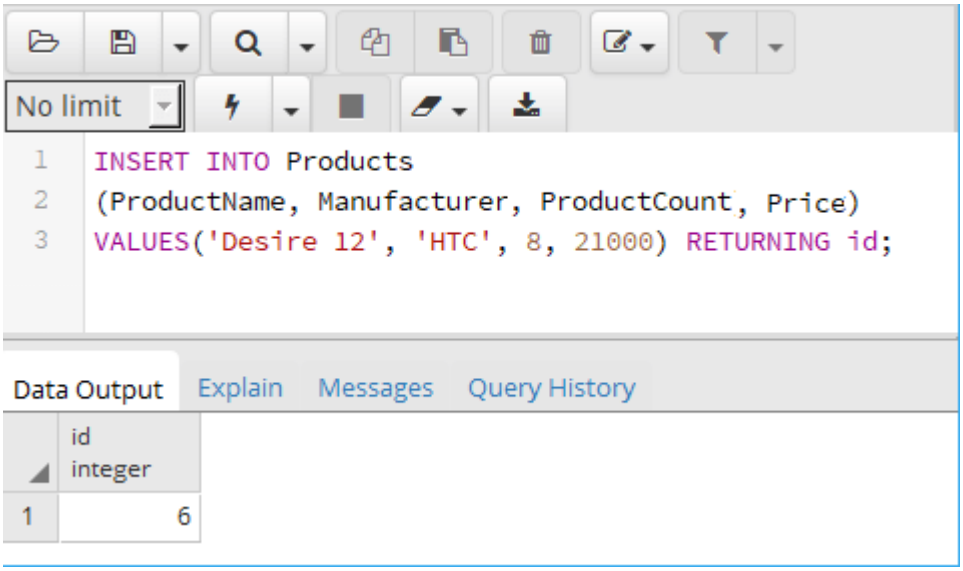
```
1.   INSERT INTO Products   (ProductName,  Manufacturer,
2. ProductCount, Price)
3.   VALUES
4.   ('iPhone 6', 'Apple', 3, 36000),
5.   ('Galaxy S8', 'Samsung‘, 2, 46000),
6.   ('Galaxy S8 Plus', 'Samsung‘, 1, 56000)
```

Bunday holda, jadvalga uchta qator qo‘shiladi.

Qaytariladigan qiymatlar

Agar biz ustunlarning faqat bir qismi uchun qiymatlarni qo‘shsak, boshqa ustunlar qanday qiymatlarga ega bo‘lishini bilmasligimiz mumkin. Masalan, Id ustunni mahsulot uchun qanday qiymatga ega bo‘ladi? RETURNING operatori yordamida biz ushbu qiymatni olishimiz mumkin:

```
1.   INSERT INTO Products
2.   (ProductName,   Manufacturer,   ProductCount,
3. Price)
4.   VALUES('Desire   12',   'HTC',   8,   21000)
5. RETURNING id;
```



4.2-rasm. SQL so‘rovlar oynasida yozilishi

Ma’lumotlar bazasidan ma’lumotlarni olish uchun SELECT buyrug‘idan foydalaning. Soddalashtirilgan shaklda u quyidagi sintaksisga ega:

```
1.  SELECT список_столбцов FROM имя_таблицы
```

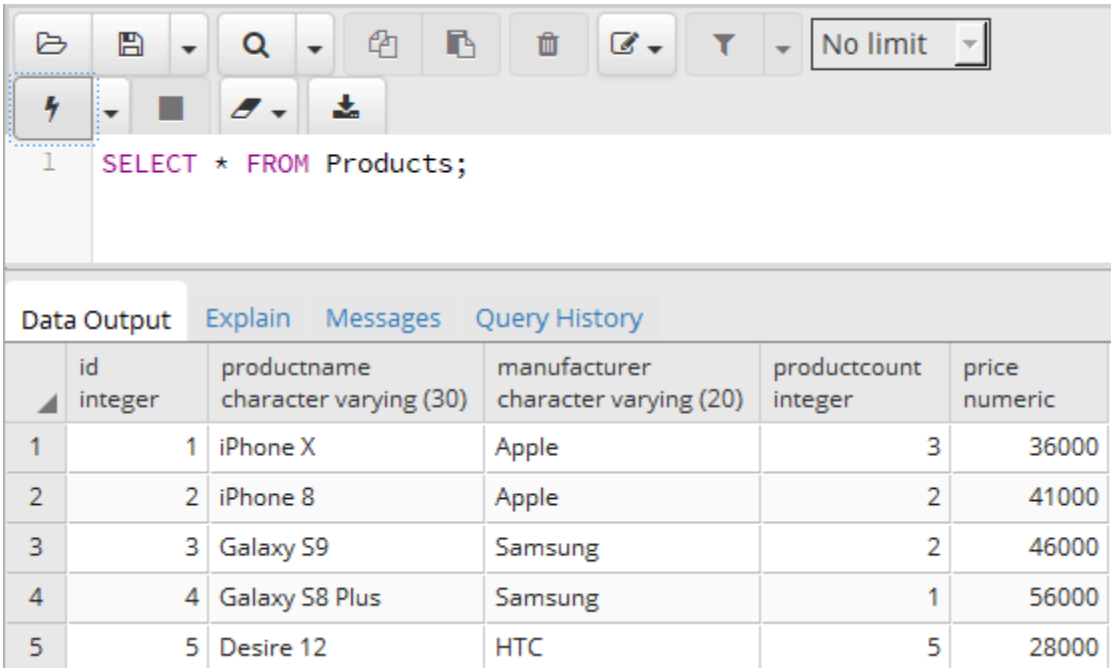
Masalan, siz avvalroq “Mahsulotlar” jadvalini yaratdingiz va unga dastlabki ma’lumotlarni qo‘shdingiz deylik:

```
1.  CREATE TABLE Products
2.  (
3.      Id SERIAL PRIMARY KEY,
4.      ProductName VARCHAR(30) NOT NULL,
5.      Manufacturer VARCHAR(20) NOT NULL,
6.      ProductCount INTEGER DEFAULT 0,
7.      Price NUMERIC
8.  );
9.
10.  INSERT INTO Products (ProductName, Manufacturer,
11. ProductCount, Price)
12.  VALUES
13.  ('iPhone X', 'Apple', 3, 36000),
14.  ('iPhone 8', 'Apple', 2, 41000),
15.  ('Galaxy S9', 'Samsung‘, 2, 46000),
16.  ('Galaxy S8 Plus', 'Samsung‘, 1, 56000),
17.  ('Desire 12', 'HTC', 5, 28000);
```

Keling, ushbu jadvaldagi barcha ob'ektlarni olamiz:

```
1.   SELECT * FROM Products;
```

Yulduzcha * barcha ustunlarni olishni xohlayotganimizni bildiradi.



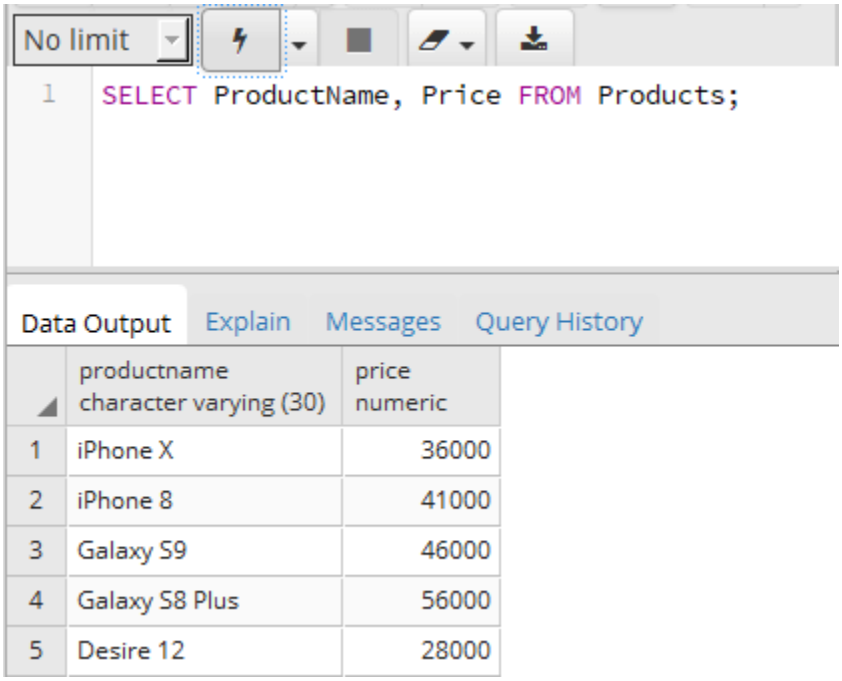
	id integer	productname character varying (30)	manufacturer character varying (20)	productcount integer	price numeric
1	1	iPhone X	Apple	3	36000
2	2	iPhone 8	Apple	2	41000
3	3	Galaxy S9	Samsung	2	46000
4	4	Galaxy S8 Plus	Samsung	1	56000
5	5	Desire 12	HTC	5	28000

4.3-rasm. Select so‘rovining ishlatilishi

Biroq, yulduzcha * belgisidan foydalanish yaxshi amaliyot hisoblanmaydi, chunki odatda barcha ustunlar kerak emas. Va yaxshiroq yondashuv SELECT so‘zidan keyin barcha kerakli ustunlarni sanab o‘tishdir. Jadvalning mutlaqo barcha ustunlari bo‘yicha ma’lumotlarni olishingiz kerak bo‘lgan holatlar bundan mustasno. Bundan tashqari, aniq ustun nomlari noma’lum bo‘lgan holatlarda * belgisidan foydalanish afzalroq bo‘lishi mumkin.

Agar biz hamma uchun emas, balki ba'zi bir ustunlar uchun ma’lumotlarni olishimiz kerak bo‘lsa, unda barcha ustunlar SELECTdan keyin vergul bilan ajratilgan holda keltirilgan:

```
1.   SELECT ProductName, Price FROM Products;
```



	productname character varying (30)	price numeric
1	iPhone X	36000
2	iPhone 8	41000
3	Galaxy S9	46000
4	Galaxy S8 Plus	56000
5	Desire 12	28000

4.4-rasm. Select so‘rovining ishlatilishi

Ustun spetsifikatsiyasi uning nomini ifodalashi shart emas. Bu har qanday ifoda bo‘lishi mumkin, masalan, arifmetik amalning natijasi. Shunday qilib, keling, quyidagi so‘rovni bajaramiz:

```
1.   SELECT ProductCount, Manufacturer, Price * ProductCount
2.   FROM Products;
```

Bu yerda tanlov uchta ustun hosil qiladi. Bundan tashqari, uchinchi ustun Narx ustunining qiymatini ProductCount ustunining qiymatiga, ya'ni mahsulotning umumiy qiymatiga ko‘paytiradi.

AS operatori yordamida siz chiqish ustunining nomini o‘zgartirishingiz yoki uning taxalluslarini belgilashingiz mumkin:

```
1   SELECT ProductCount AS Title,
2   Manufacturer,
3   Price * ProductCount  AS TotalSum
4   FROM Products;
```

Bunday holda, tanlov natijasi 3 ta ustun uchun ma’lumotlardir. Birinchi ustun Sarlavha taxallus bilan belgilanadi, lekin aslida u Mahsulot nomi ustunini ifodalaydi. Ikkinchi ustun o‘z nomini saqlab qoladi - Ishlab chiqaruvchi. Uchinchi ustun, TotalSum, ProductCount va Price ustunlari

mahsulotini saqlaydi.

```
SELECT ProductCount AS Title,
Manufacturer,
Price * ProductCount  AS TotalSum
FROM Products;
```

Data Output | Explain | Messages | Query History

	title integer	manufacturer character varying (20)	totalsum numeric
1	3	Apple	108000
2	2	Apple	82000
3	2	Samsung	92000
4	1	Samsung	56000
5	5	HTC	140000

4.5-rasm. Select so‘rovining ishlatilishi

5- laboratoriya ishi
SQL tilining skalyar ifodalarining maxsus jihatlarining muhokamasi

Ishdan maqsad: ma’lumotlar bazasini boshqarish dasturiy vositasi yordamida masofaviy serverga ulanishni sozlash va serverdagi ma’lumotlarni qayta ishlash, tahrirlash ko‘nikmalariga ega bo‘lish.

Masalaning qo‘yilishi: ma’lumotlar bazasini boshqarish vositalari yordamida masofaviy serverdagi ma’lumotlarni qayta ishlash.

Uslubiy ko‘rsatmalar: SQL so‘rovlar tilida ishlatiladigan so‘rovlarni o‘rganishda davom etamiz. WHERE shart berish buyrug‘ini o‘rganamiz. UPDATE yangilash hamda DELETE o‘chirish buyruqlarini ham ko‘rib chiqamiz.

Ma’lumotlarni filtrlash uchun WHERE bandidan foydalaniladi, shundan so‘ng filtrlash amalga oshiriladigan shart ko‘rsatiladi:

```
1   WHERE shart
```

Agar shart to‘g‘ri bo‘lsa, u holda qator olingan tanlovga kiritiladi. Sifatida taqqoslash operatsiyalaridan foydalanish mumkin. Ushbu

operatsiyalar ikkita ifodani taqqoslaydi. PostgreSQL da quyidagi taqqoslash operatorlaridan foydalanish mumkin:

=: tenglik uchun taqqoslash
<>: tengsizlik uchun solishtirish
<: kamroq
>: dan ortiq
! <: dan kam emas
!>: ortiq emas
<=: kichik yoki teng
> =: kattaroq yoki teng

Masalan, Apple tomonidan ishlab chiqarilgan barcha mahsulotlarni topamiz:

```
1.  SELECT * FROM Products
2.  WHERE Manufacturer = 'Apple';
```

5.1-rasm. SQL so‘rovlar oynasida so‘rovlar ishlatilishi

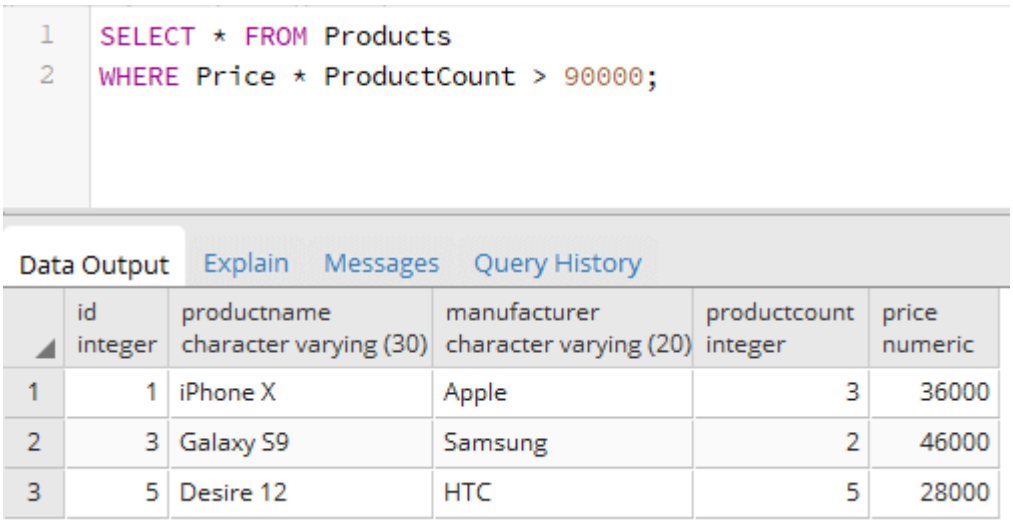
Shuni ta'kidlash kerakki, bu holda belgilarning holati juda muhim, masalan, "Olma" qatori "APPLE" yoki "olma" qatoriga teng emas.

Yana bir misol - narxi 29 000 dan past bo‘lgan barcha mahsulotlarni topamiz:

```
1.  SELECT * FROM Products
2.  WHERE Price < 39000;
```

Shart sifatida murakkabroq ifodalardan foydalanish mumkin. Masalan, umumiy qiymati 90 000 dan ortiq bo‘lgan barcha mahsulotlarni topamiz:

```
1.  SELECT * FROM Products
2.  WHERE Price * ProductCount > 90000;
```



```
1  SELECT * FROM Products
2  WHERE Price * ProductCount > 90000;
```

Data Output | Explain | Messages | Query History

	id integer	productname character varying (30)	manufacturer character varying (20)	productcount integer	price numeric
1	1	iPhone X	Apple	3	36000
2	3	Galaxy S9	Samsung	2	46000
3	5	Desire 12	HTC	5	28000

5.2-rasm. SQL so‘rovlar oynasida so‘rovlar ishlatilishi

Mantiqiy amallar.

Mantiqiy operatorlar PostgreSQL-da bir nechta shartlarni bitta shartga birlashtirish uchun ishlatilishi mumkin:

AND: mantiqiy AND operatsiyasi. U ikkita ifodani birlashtiradi:

```
1.  ifoda1 AND ifoda2
```

Faqat bu ikkala ifoda bir vaqtning o‘zida to‘g‘ri bo‘lsa, AND operatorining umumiy sharti ham to‘g‘ri bo‘ladi. Ya'ni, agar birinchi shart ham, ikkinchisi ham to‘g‘ri bo‘lsa.

OR: mantiqiy YOKI operatsiya. Shuningdek, u ikkita iborani birlashtiradi:

```
1.  ifoda1 OR ifoda2
```

Agar ushbu ifodalardan kamida bittasi to‘g‘ri bo‘lsa, OR operatorining umumiy sharti ham to‘g‘ri bo‘ladi. Ya'ni, birinchi shart yoki ikkinchi shart to‘g‘ri bo‘lsa.

NOT: mantiqiy inkor operatsiyasi. Agar bu amaldagi ifoda noto‘g‘ri bo‘lsa, u holda umumiy shart rost bo‘ladi.

```
1.  NOT ifoda
```

Misol uchun, ishlab chiqaruvchisi Samsung bo‘lgan va bir vaqtning o‘zida narxi 50 000 dan ortiq bo‘lgan barcha mahsulotlarni tanlaymiz:

```
1.  SELECT * FROM Products
2.  WHERE Manufacturer = 'Samsung‘ AND Price > 50000;
```

```
1  SELECT * FROM Products
2  WHERE Manufacturer = 'Samsung' AND Price > 50000;
```

Data Output | Explain | Messages | Query History

	id integer	productname character varying (30)	manufacturer character varying (20)	productcount integer	price numeric
1	4	Galaxy S8 Plus	Samsung	1	56000

5.3-rasm. PostgreSQL-da mantiqiy VA operatori

Endi operatorni OR ga almashtiramiz. Ya'ni, biz ishlab chiqaruvchisi Samsung yoki narxi 50 000 dan yuqori bo‘lgan barcha mahsulotlarni tanlaymiz:

```
1.  SELECT * FROM Products
2.  WHERE Manufacturer = 'Samsung‘ OR Price > 50000;
```

```
1  SELECT * FROM Products
2  WHERE Manufacturer = 'Samsung' OR Price > 50000;
```

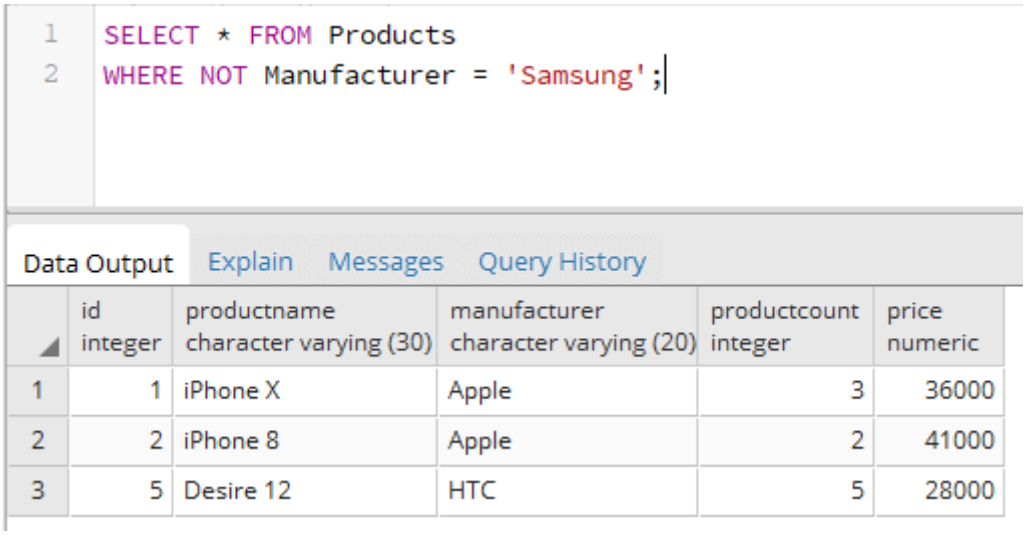
Data Output | Explain | Messages | Query History

	id integer	productname character varying (30)	manufacturer character varying (20)	productcount integer	price numeric
1	3	Galaxy S9	Samsung	2	46000
2	4	Galaxy S8 Plus	Samsung	1	56000

5.4-rasm. PostgreSQL da mantiqiy OR operatori

NO operatorining qo‘llanilishi - biz ishlab chiqaruvchisi Samsung bo‘lmagan barcha mahsulotlarni tanlaymiz:

```
1.   SELECT * FROM Products
2.   WHERE NOT Manufacturer = 'Samsung‘;
```



```
SELECT * FROM Products
WHERE NOT Manufacturer = 'Samsung';
```

Data Output | Explain | Messages | Query History

	id integer	productname character varying (30)	manufacturer character varying (20)	productcount integer	price numeric
1	1	iPhone X	Apple	3	36000
2	2	iPhone 8	Apple	2	41000
3	5	Desire 12	HTC	5	28000

5.5-rasm. PostgreSQL-da mantiqiy NOT operatori

Ammo ko‘p hollarda siz NOT operatorisiz qilishingiz mumkin. Shunday qilib, oldingi misolda biz quyidagicha yozishimiz mumkin:

```
1.   SELECT * FROM Products
2.   WHERE Manufacturer <> 'Samsung‘
```

Bundan tashqari, bitta SELECT buyrug‘ida siz bir vaqtning o‘zida bir nechta operatorlardan foydalanishingiz mumkin:

```
1.   SELECT * FROM Products
2.   WHERE Manufacturer = 'Samsung‘ OR Price > 30000 AND
ProductCount > 2;
```

AND operatori ustuvorlikka ega bo‘lganligi sababli, birinchi navbatda Price> 30000 AND ProductCount> 2 sub ifodasi, keyin esa OR operatori bajariladi. Ya'ni, bu yerda siz 2 dan ortiq zaxiraga ega va bir vaqtning o‘zida narxi 30 000 dan yuqori bo‘lgan tovarlarni yoki ishlab chiqaruvchisi Samsung bo‘lgan tovarlarni tanlaysiz.

Qavslar yordamida amallar tartibini ham qayta belgilashimiz mumkin:

```
1    SELECT * FROM Products
2    WHERE (Manufacturer = 'Samsung‘ OR Price > 30000) AND
ProductCount > 2;
```

IS NULL

Bir qator ustunlar null bo‘lishi mumkin. Bu qiymat bo‘sh satrga teng emas. " NULL har qanday qiymatning to‘liq yo‘qligini bildiradi. Va bunday qiymatni tekshirish uchun IS NULL operatori ishlatiladi.

Misol uchun, ProductCount maydoni o‘rnatilmagan barcha mahsulotlarni tanlaymiz:

```
1.  SELECT * FROM Products
2.  WHERE ProductCount IS NULL;
```

Agar, aksincha, ProductCount maydoni NULL bo‘lmagan qatorlarni olishingiz kerak bo‘lsa, siz NOT operatoridan foydalanishingiz mumkin:

```
1.  SELECT * FROM Products
2.  WHERE ProductCount IS NULL;
```

UPDATE buyrug‘i PostgreSQL ma’lumotlar bazasidagi ma’lumotlarni yangilash uchun ishlatiladi. U quyidagi umumiy rasmiy sintaksisga ega:

```
1.  UPDATE имя_таблицы
2.  SET столбец1 = значение1, столбец2 = значение2, ...
столбецN = значениеN
3.  [WHERE условие_обновления]
```

Masalan, barcha mahsulotlar narxini 3000 ga oshiramiz:

```
1.  UPDATE Products
2.  SET Price = Price + 3000;
```

```
SELECT * FROM Products;
```

Data Output | Explain | Messages | Query History

	id integer	productname character varying (30)	manufacturer character varying (20)	productcount integer	price numeric
1	1	iPhone X	Apple	3	36000
2	2	iPhone 8	Apple	2	41000
3	3	Galaxy S9	Samsung	2	46000
4	4	Galaxy S8 Plus	Samsung	1	56000
5	5	Desire 12	HTC	5	28000

```
UPDATE Products
SET Price = Price + 3000;

SELECT * FROM Products;
```

Data Output | Explain | Messages | Query History

	id integer	productname character varying (30)	manufacturer character varying (20)	productcount integer	price numeric
1	1	iPhone X	Apple	3	39000
2	2	iPhone 8	Apple	2	44000
3	3	Galaxy S9	Samsung	2	49000
4	4	Galaxy S8 Plus	Samsung	1	59000
5	5	Desire 12	HTC	5	31000

5.6-rasm. PostgreSQL-da UPDATE operatori

Bunday holda, yangilanish barcha qatorlar uchun amal qiladi. WHERE bandidan foydalanib, siz yangilanadigan qatorlarni belgilash uchun shartdan foydalanishingiz mumkin - agar qator shartga mos kelsa, u yangilanadi. Misol uchun, ishlab chiqaruvchining nomini "Samsung" dan "Samsung Inc." ga o‘zgartiramiz:

```
1.  UPDATE Products
2.  SET Manufacturer = 'Samsung Inc.'
3.  WHERE Manufacturer = 'Samsung‘;
```

```
1  UPDATE Products
2  SET Manufacturer = 'Samsung Inc.'
3  WHERE Manufacturer = 'Samsung';
4
5  SELECT * FROM Products;
```

Data Output | Explain | Messages | Query History

	id integer	productname character varying (30)	manufacturer character varying (20)	productcount integer	price numeric
1	1	iPhone X	Apple	3	39000
2	2	iPhone 8	Apple	2	44000
3	5	Desire 12	HTC	5	31000
4	3	Galaxy S9	Samsung Inc.	2	49000
5	4	Galaxy S8 Plus	Samsung Inc.	1	59000

5.7-rasm. PostgreSQL-da UPDATE operatori ishlatilishi

Bundan tashqari, bir vaqtning o‘zida bir nechta ustunlarni yangilashingiz mumkin:

```
1.  UPDATE Products
2.  SET Manufacturer = 'Samsung‘,
3.      ProductCount = ProductCount + 3
4.  WHERE Manufacturer = 'Samsung Inc.';
```

Ma’lumotlarni o‘chirish. DELETE buyrug‘i.

PostgreSQL ma’lumotlarni o‘chirish uchun DELETE buyrug‘idan foydalanadi. U quyidagi sintaksisga ega:

```
1.  DELETE FROM имя_таблицы
2.  [WHERE условие_удаления]
```

Masalan, ishlab chiqaruvchisi Apple bo‘lgan qatorlarni olib tashlaymiz:

```
1.  DELETE FROM Products
2.  WHERE Manufacturer='Apple';
```

```
SELECT * FROM Products;
```

Data Output | Explain | Messages | Query History

	id integer	productname character varying (30)	manufacturer character varying (20)	productcount integer	price numeric
1	1	iPhone X	Apple	3	39000
2	2	iPhone 8	Apple	2	44000
3	5	Desire 12	HTC	5	31000
4	3	Galaxy S9	Samsung	5	49000
5	4	Galaxy S8 Plus	Samsung	4	59000

```
DELETE FROM Products
WHERE Manufacturer='Apple';

SELECT * FROM Products;
```

Data Output | Explain | Messages | Query History

	id integer	productname character varying (30)	manufacturer character varying (20)	productcount integer	price numeric
1	5	Desire 12	HTC	5	31000
2	3	Galaxy S9	Samsung	5	49000
3	4	Galaxy S8 Plus	Samsung	4	59000

5.8-rasm. PostgreSQL-da DELETE operatori ishlatilishi

Yoki HTC tomonidan ishlab chiqarilgan va narxi 35 000 dan past bo‘lgan barcha mahsulotlarni olib tashlang:

```
1.  DELETE FROM Products
2.  WHERE Manufacturer='HTC' AND Price < 15000;
```

Agar shartdan qat'i nazar, barcha qatorlarni o‘chirish kerak bo‘lsa, shartni o‘tkazib yuborish mumkin:

```
1.  DELETE FROM Products;
```

Nazorat savollari:

1. CREATE TRIGGER iborasining sintaksisiga izoh bering.

2. INSERTED va DELETED jadvallarining maqsadi nima?
3. Qaysi tizim protsedurasi trigger kodini olishga imkon beradi?
4. Ma‘lumotlar bazasi obyekti ishga tushirilishini qayerdan bilaman?
5. Muayyan jadval uchun triggerlar ro‘yxatini qanday olish mumkin?

6- laboratoriya ishi
Triggerlarning yozilishi bo‘yicha misollar yechish

Ishdan maqsad: ma’lumotlar bazasida dasturlarda triggerlarni ishlab chiqish va ulardan foydalanish bo‘yicha amaliy ko‘nikmalarga ega bo‘lish.

Masalaning qo‘yilishi: ma’lumotlar bazasini boshqarish vositalari yordamida serverdagi ma’lumotlarni qayta ishlash.

Uslubiy ko‘rsatmalar: SQL tilida ***SELECT*** operatori bilan ishlashni davom etamiz. Ushbu darsimizda ***SELECT*** operatori bilan yanada chuqurroq tanishamiz. Darsda ***DISTINCT***, ***LIMIT, OFFSET, ORDER BY, LIKE*** operatorlarini ***SELECT*** so‘rovida ishlatishni o‘rganamiz.

So‘rovlar

DICTINCT. Noyob qiymatlarni olish

DISTINCT operatori ma’lum ustunlar uchun noyob ma’lumotlarni tanlash imkonini beradi.

Misol uchun, mahsulot jadvalida turli xil mahsulotlar bir xil ishlab chiqaruvchilarga ega bo‘lishi mumkin. Masalan, bizda quyidagi jadval mavjud:

```
1.   CREATE TABLE Products
2.   (
3.       Id SERIAL PRIMARY KEY,
4.       ProductName VARCHAR(30) NOT NULL,
5.       Manufacturer VARCHAR(20) NOT NULL,
6.       ProductCount INTEGER DEFAULT 0,
7.       Price NUMERIC
8.   );
9.   INSERT INTO Products   (ProductName, Manufacturer,
10. ProductCount, Price)
11.  VALUES
12.  ('iPhone X', 'Apple', 2, 71000),
13.  ('iPhone 8', 'Apple', 3, 56000),
14.  ('Galaxy S9', 'Samsung‘, 6, 56000),
15.  ('Galaxy S8 Plus', 'Samsung‘, 2, 46000),
16.  ('Desire 12', 'HTC', 3, 26000);
```

Keling, barcha ishlab chiqaruvchilarni tanlaymiz:

```
1.  SELECT DISTINCT Manufacturer FROM Products;
```

```
1  select manufacturer from products;
```

Data Output | Explain | Messages | Query History

	manufacturer character varying (20)
1	Apple
2	Apple
3	Samsung
4	Samsung
5	HTC

```
1  select distinct manufacturer from products;
```

Data Output | Explain | Messages | Query History

	manufacturer character varying (20)
1	HTC
2	Samsung
3	Apple

6.1-rasm. DISTINCT qo‘llanilishi

ORDER BY. Tartiblash

ORDER BY bandi ma’lum bir ustun bo‘yicha qiymatlarni saralashga imkon beradi. Masalan, Mahsulotlar jadvalidan Mahsulotlar soni ustuni bo‘yicha tanlovga buyurtma beraylik:

```
1.  SELECT *  FROM Products
2.  ORDER BY ProductCount;
```

```
1  select * from products
2  order by productcount;
```

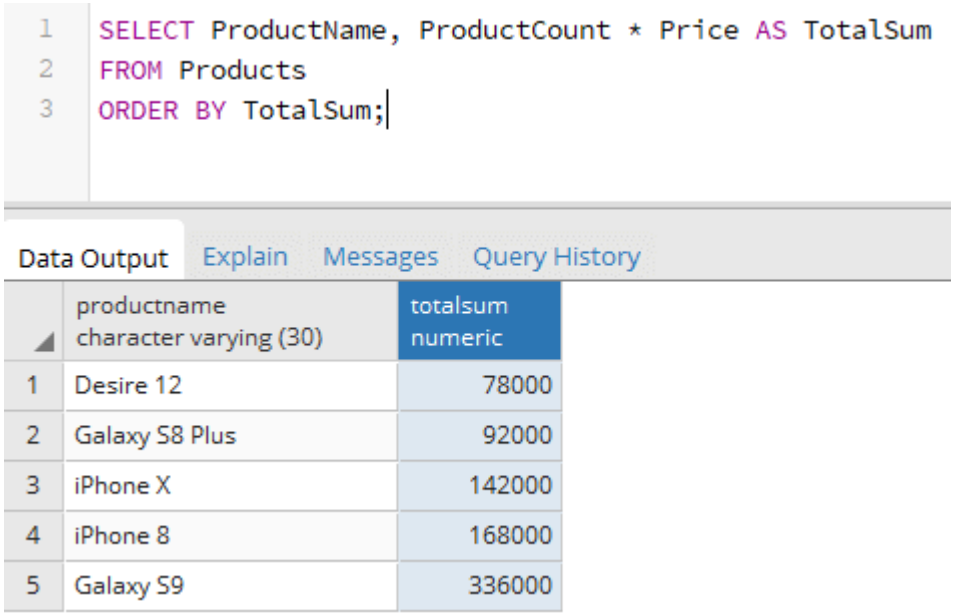
Data Output | Explain | Messages | Query History

	id integer	productname character varying (30)	manufacturer character varying (20)	productcount integer	price numeric
1	11	iPhone X	Apple	2	71000
2	14	Galaxy S8 Plus	Samsung	2	46000
3	12	iPhone 8	Apple	3	56000
4	15	Desire 12	HTC	3	26000
5	13	Galaxy S9	Samsung	6	56000

6.2-rasm. ORDER BY qo‘llanilishi

Shuningdek, siz ma’lumotlarni AS operatori yordamida aniqlanadigan ustun taxalluslari bo‘yicha buyurtma qilishingiz mumkin:

```
1.  SELECT ProductName, ProductCount * Price AS TotalSum
2.  FROM Products
3.  ORDER BY TotalSum;
```



6.3-rasm. ORDER BY qo‘llanilishi.

Saralash mezoni sifatida murakkab ustunga asoslangan ifoda ham ishlatilishi mumkin:

```
1.  SELECT ProductName, Price, ProductCount
2.  FROM Products
3.  ORDER BY ProductCount * Price;
```

```
SELECT ProductName, Price, ProductCount
FROM Products
ORDER BY ProductCount * Price;
```

Data Output | Explain | Messages | Query History

	productname character varying (30)	price numeric	productcount integer
1	Desire 12	26000	3
2	Galaxy S8 Plus	46000	2
3	iPhone X	71000	2
4	iPhone 8	56000	3
5	Galaxy S9	56000	6

6.4-rasm. ORDER BY qo‘llanilish sohalari.

Kamayish bo‘yicha tartiblash.

Odatiy bo‘lib, ma’lumotlar o‘sish tartibida tartiblanadi, lekin siz kamayish tartibida tartiblash uchun DESC operatoridan foydalanishingiz mumkin.

```
SELECT ProductName, Manufacturer
FROM Products
ORDER BY Manufacturer DESC;
```

Data Output | Explain | Messages | Query History

	productname character varying (30)	manufacturer character varying (20)
1	Galaxy S9	Samsung
2	Galaxy S8 Plus	Samsung
3	Desire 12	HTC
4	iPhone X	Apple
5	iPhone 8	Apple

6.5-rasm. ORDER BY qo‘llanilishi

Odatiy bo‘lib, o‘sish tartibida tartiblanadigan DESC o‘rniga ASC operatori ishlatiladi:

```
1.  SELECT ProductName, Manufacturer
2.  FROM Products
3.  ORDER BY Manufacturer ASC;
```

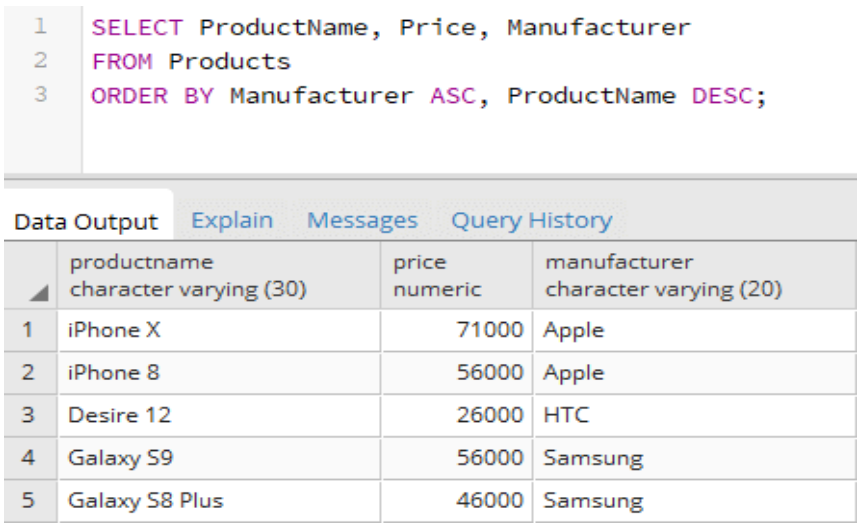
Bir nechta ustunlar bo‘yicha tartiblash.

Agar bir vaqtning o‘zida bir nechta ustunlar bo‘yicha saralash kerak bo‘lsa, ularning barchasi ORDER BY operatoridan keyin vergul bilan ajratiladi:

```
1.  SELECT ProductName, Price, Manufacturer
2.  FROM Products
3.  ORDER BY Manufacturer, ProductName;
```

Bunday holda, satrlar birinchi navbatda Ishlab chiqaruvchi ustuni bo‘yicha o‘sish tartibida tartiblanadi. Keyin, agar ishlab chiqaruvchi ustuni bir xil qiymatga ega bo‘lgan ikkita qator bo‘lsa, ular Mahsulot nomi ustuni bo‘yicha ham o‘sish tartibida tartiblanadi. Ammo yana, ASC va DESC-dan foydalanib, siz turli ustunlar uchun o‘sish va kamayish tartibida tartiblashni alohida aniqlashingiz mumkin:

```
1.  SELECT ProductName, Price, Manufacturer
2.  FROM Products
3.  ORDER BY Manufacturer ASC, ProductName DESC;
```



```
1  SELECT ProductName, Price, Manufacturer
2  FROM Products
3  ORDER BY Manufacturer ASC, ProductName DESC;
```

Data Output | Explain | Messages | Query History

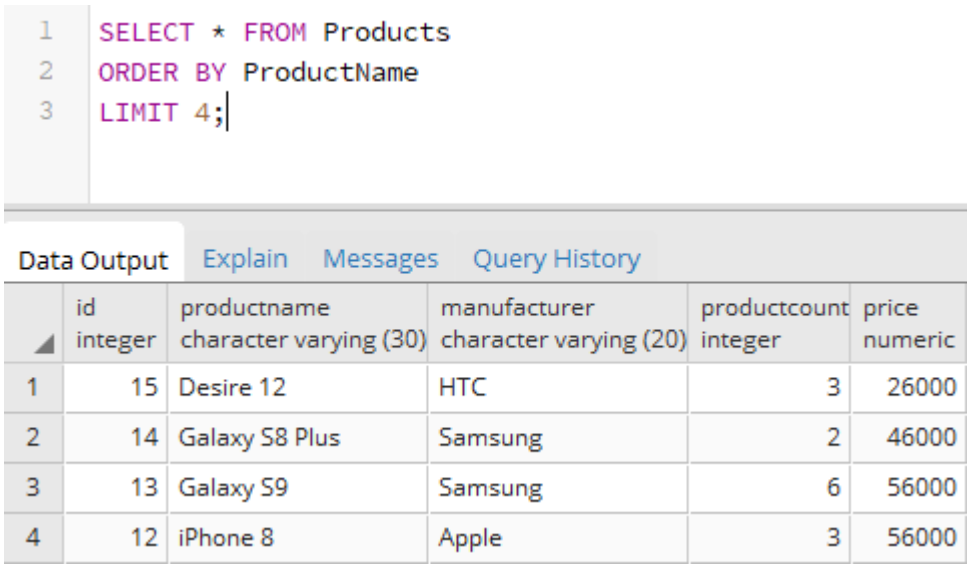
	productname character varying (30)	price numeric	manufacturer character varying (20)
1	iPhone X	71000	Apple
2	iPhone 8	56000	Apple
3	Desire 12	26000	HTC
4	Galaxy S9	56000	Samsung
5	Galaxy S8 Plus	46000	Samsung

6.6-rasm. ORDER BY hamda ASC va DESC qo‘llanilish sohalari

LIMIT va OFFSET.

LIMIT operatori ma’lum bir qatorlarni olish imkonini beradi:

```
1.  SELECT * FROM Products
2.  ORDER BY ProductName
3.  LIMIT 4;
```

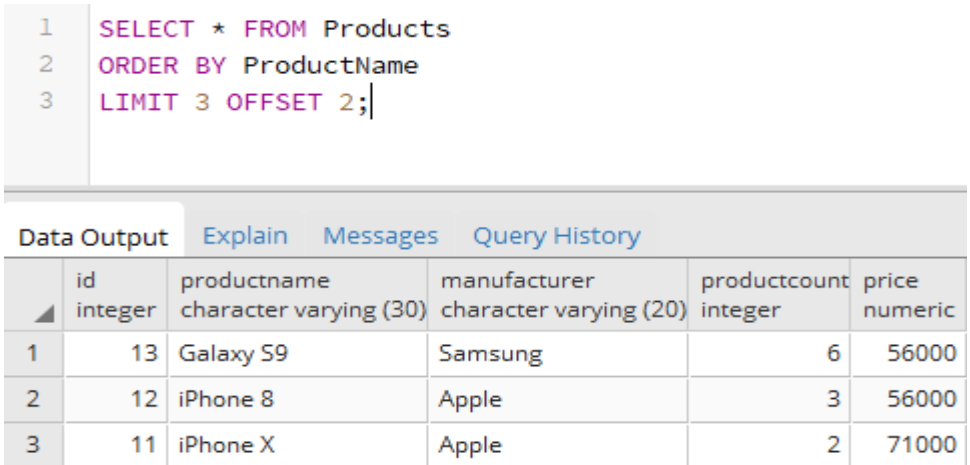


	id integer	productname character varying (30)	manufacturer character varying (20)	productcount integer	price numeric
1	15	Desire 12	HTC	3	26000
2	14	Galaxy S8 Plus	Samsung	2	46000
3	13	Galaxy S9	Samsung	6	56000
4	12	iPhone 8	Apple	3	56000

6.7-rasm. LIMIT operatoridan foydalanish.

OFFSET operatori tanlashni qaysi qatordan boshlashni belgilash imkonini beradi. Masalan, 2-dan boshlab 3 ta qatorni tanlaymiz:

```
1.  SELECT * FROM Products
2.  ORDER BY ProductName
3.  LIMIT 3 OFFSET 2;
```



	id integer	productname character varying (30)	manufacturer character varying (20)	productcount integer	price numeric
1	13	Galaxy S9	Samsung	6	56000
2	12	iPhone 8	Apple	3	56000
3	11	iPhone X	Apple	2	71000

6.8-rasm. LIMIT va OFFSET operatorlari.

Agar ma’lum biridan boshlab barcha qatorlarni tanlash kerak bo‘lsa, LIMIT operatorini o‘tkazib yuborish mumkin:

```
1.  SELECT * FROM Products
2.  ORDER BY ProductName
3.  OFFSET 2;
```

Yoki LIMITdan keyin ALL kalit so‘zini belgilang:

```
1.  SELECT * FROM Products
2.  ORDER BY ProductName
3.  LIMIT ALL OFFSET 2;
```

Filtrlash operatorlari:

IN operatori

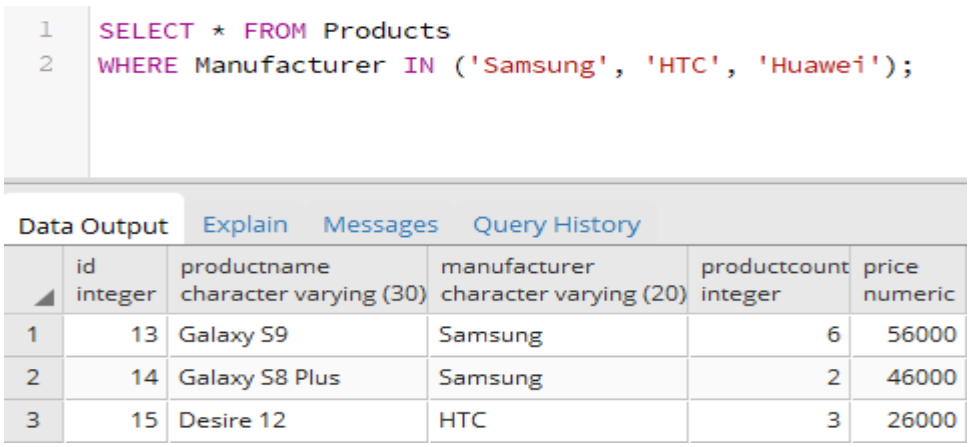
IN operatori ustunlar bo‘lishi kerak bo‘lgan qiymatlar to‘plamini aniqlashga imkon beradi:

```
1.  WHERE выражение [NOT] IN (выражение)
```

IN dan keyingi qavs ichidagi ifoda qiymatlar to‘plamini belgilaydi. Ushbu to‘plam, masalan, boshqa so‘rov asosida dinamik ravishda hisoblanishi mumkin yoki u doimiy qiymatlar bo‘lishi mumkin.

Masalan, ishlab chiqaruvchisi Samsung, Xiaomi yoki Huawei bo‘lgan mahsulotlarni tanlaymiz:

```
1.  SELECT * FROM Products
2.  WHERE Manufacturer IN ('Samsung‘, 'HTC', 'Huawei');
```



	id integer	productname character varying (30)	manufacturer character varying (20)	productcount integer	price numeric
1	13	Galaxy S9	Samsung	6	56000
2	14	Galaxy S8 Plus	Samsung	2	46000
3	15	Desire 12	HTC	3	26000

6.9-rasm. IN operatori

Shu bilan bir qatorda, ushbu qiymatlarning barchasini OR operatori orqali tekshirish mumkin:

```
1.  SELECT * FROM Products
2.  WHERE Manufacturer = 'Samsung‘ OR Manufacturer = 'HTC' OR Manufacturer = 'Huawei';
```

Biroq, IN operatoridan foydalanish, ayniqsa, bunday qiymatlar juda ko‘p bo‘lsa, ancha qulayroqdir.

NOT operatoridan foydalanib, siz, aksincha, qiymatlar to‘plamiga mos kelmaydigan barcha satrlarni topishingiz mumkin:

```
1.  SELECT * FROM Products
2.  WHERE Manufacturer NOT IN ('Samsung‘, 'HTC', 'Huawei');
```

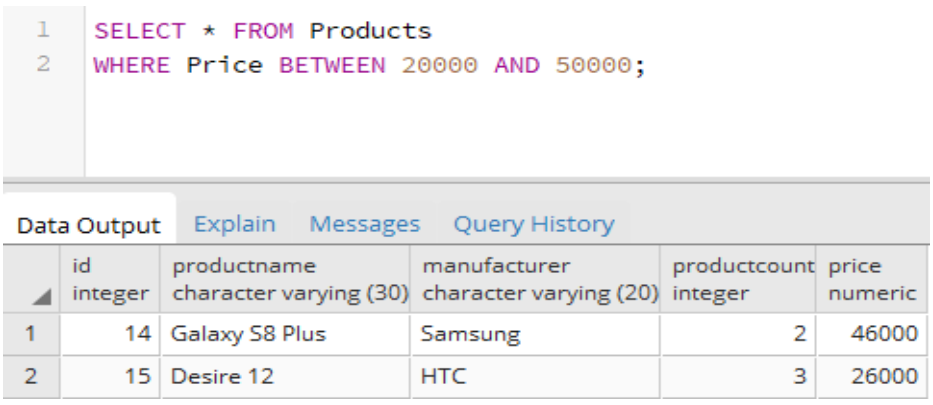
BETWEEN operatori

BETWEEN operatori ifoda mos kelishi kerak bo‘lgan boshlang‘ich va yakuniy qiymatdan foydalangan holda bir qator qiymatlarni belgilaydi:

```
1.  WHERE выражение [NOT] BETWEEN начальное_значение AND конечное_значение
```

Misol uchun, narxlari 20 000 dan 50 000 gacha bo‘lgan barcha mahsulotlarni olaylik (boshlang‘ich va yakuniy qiymatlar ham qatorga kiritilgan):

```
1.  SELECT * FROM Products
2.  WHERE Price BETWEEN 20000 AND 50000;
```



6.10-rasm. BETWEEN operatorining ishlatilishi

Agar, aksincha, ushbu diapazonga kirmaydigan qatorlarni tanlash kerak bo‘lsa, NOT operatori ishlatiladi:

```
1.   SELECT * FROM Products
2.   WHERE Price NOT BETWEEN 20000 AND 50000;
```

Bundan tashqari, murakkabroq ifodalardan foydalanishingiz mumkin. Masalan, biz zaxiralari ma’lum miqdorda bo‘lgan tovarlarni olamiz (narx * miqdor):

```
1.   SELECT * FROM Products
2.   WHERE Price * ProductCount BETWEEN 90000 AND 150000;
```

LIKE operatori

LIKE operatori ifoda mos kelishi kerak bo‘lgan satr nusxasini oladi.

```
1.   WHERE выражение [NOT] LIKE шаблон_строки
```

Shablonni aniqlash uchun bir qancha maxsus belgilardan foydalanish mumkin:

•%: har qanday belgilar soniga ega bo‘lishi mumkin bo‘lgan har qanday pastki qatorga mos keladi va pastki qatorda hech qanday belgilar bo‘lmasligi mumkin

Masalan, "Galaxy%" kabi mahsulot nomi "Galaxy Ace 2" yoki "Galaxy S7" kabi qiymatlarga mos keladi.

• _: har qanday bitta belgiga mos keladi

Masalan, "Galaxy S_" kabi mahsulot nomi "Galaxy S7" yoki "Galaxy S8" kabi qiymatlarga mos keladi.

LIKE operatorini qo‘llaymiz:

```
1.   SELECT * FROM Products
2.   WHERE ProductName LIKE 'iPhone%';
```

```
SELECT * FROM Products
WHERE ProductName LIKE 'iPhone%';
```

Data Output | Explain | Messages | Query History

	id integer	productname character varying (30)	manufacturer character varying (20)	productcount integer	price numeric
1	11	iPhone X	Apple	2	71000
2	12	iPhone 8	Apple	3	56000

6.11-rasm. LIKE operatorining ishlatilishi

Nazorat savollari.

1. Benuqsonlikni cheklash nimani anglatadi?
2. Butunlik uchun cheklash turlarini sanab bering.
3. Triggerlar bilan qanday yaxlitlikni cheklash mumkin?
4. Saqlangan protsedurada trigger tomonidan bajariladigan amallarni kodlash mumkinmi?
5. Triggerlar va saqlanadigan protseduralar o‘rtasidagi farqlar qanday?

7- laboratoriya ishi
Jadvallarni qo‘shish. Murakkab so‘rovlar vositalaridan foydalangan holda masalalar yechish

Ishdan maqsad: ma’lumotlar bazasini boshqarish dasturiy vositasi yordamida serverga ulanishni sozlash va serverdagi ma’lumotlarni qayta ishlash, tahrirlash ko‘nikmalariga ega bo‘lish.

Masalaning qo‘yilishi: ma’lumotlar bazasini boshqarish vositalari yordamida serverdagi ma’lumotlarni qayta ishlash.

Uslubiy ko‘rsatmalar: Jadvallarni jamlashtirish. Jamlashtirish relyatsion ma’lumotlar bazasi operatsiyalaridan biri bo‘lib, jadvallar orasidagi aloqani belgilaydi va ulardan ma’lumotni bitta komanda yordamida ajratishga imkon beradi. Har xil jadvallarda bir xil nomli ustunlar bo‘lishi mumkin bo‘lgani uchun, kerakli ustun uchun jadval nomi prefiksi ishlatiladi. Jamlashda jadvallar FROM ifodasidan so‘ng ro‘yxat sifatida tasvirlanadi. So‘rov predikati ixtiyoriy jadval ixtiyoriy ustuniga tegishli bo‘lishi mumkin. Jamlashning eng soddasi bu dekart ko‘paytmasi bo‘lib, uni quidagicha bajarish mumkin:

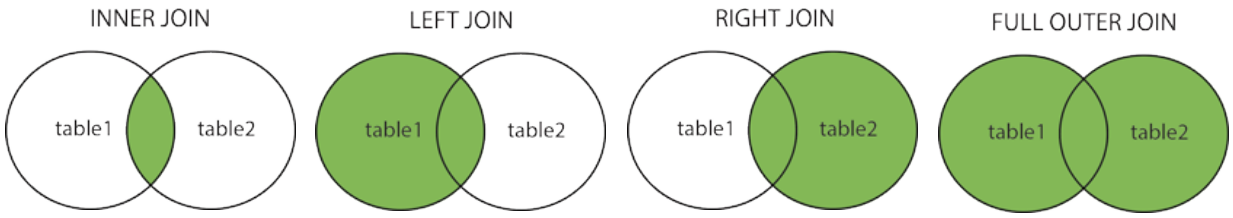
Jadvallarni birlashtirish

Ma’lumotlar bazasidagi bir nechta jadvallarni birlashtirib natijalarni olishimz ham mumkin. Bunda bizga **JOIN** yordamchi so‘zi yordam beradi. Bu operatorimiz asosan to‘rtta ko‘rinishda foydalaniladi:

(INNER) JOIN: Ikkala jadvalda ham mos keladigan qiymatlarga ega bo‘lgan ma’lumotlarni qaytaradi.

LEFT (OUTER) JOIN: Chap jadvaldagi barcha ma’lumotlarni va o‘ng jadvaldagi mos keladigan ma’lumotlarni qaytaradi.

RIGHT (OUTER) JOIN: O‘ng jadvaldagi barcha ma’lumotlarni va chap jadvaldagi mos keladigan ma’lumotlarni qaytaradi.



FULL (OUTER) JOIN: Agar chap yoki o‘ng jadvalda mos keladigan bo‘lsa, barcha ma’lumotlarni qaytaradi.

So‘rovni yozishda bir nechta jadvallarni o‘zaro bo‘ladigan indeks maydonlar ishlatiladi.

Ko‘pincha bir nechta jadvallardan ma’lumotlarni olishimiz kerak bo‘ladi. Turli jadvallardagi ma’lumotlarni birlashtirish uchun SELECT buyrug‘idan foydalanishingiz mumkin. Misol uchun, sizda munosabatlar bilan bog‘langan quyidagi jadvallar mavjud deylik:

```
1.  CREATE TABLE Products
2.  (
3.      Id SERIAL PRIMARY KEY,
4.      ProductName VARCHAR(30) NOT NULL,
5.      Company VARCHAR(20) NOT NULL,
6.      ProductCount INTEGER DEFAULT 0,
7.      Price NUMERIC NOT NULL
8.  );
9.  CREATE TABLE Customers
10. (
11.     Id SERIAL PRIMARY KEY,
12.     FirstName VARCHAR(30) NOT NULL
13. );
14. CREATE TABLE Orders
15. (
        Id SERIAL PRIMARY KEY,
```

```
        ProductId INTEGER NOT NULL REFERENCES
Products(Id) ON DELETE CASCADE,
        CustomerId INTEGER NOT NULL REFERENCES
Customers(Id) ON DELETE CASCADE,
        CreatedAt DATE NOT NULL,
        ProductCount INTEGER DEFAULT 1,
        Price NUMERIC NOT NULL
    );
```

Bunday holda, Mijozlar va Mahsulotlar jadvallari Buyurtmalar jadvaliga birdan ko‘pga bog‘langan. ProductId va CustomerId xorijiy kalitlari ko‘rinishidagi Buyurtmalar jadvali mos ravishda Mahsulotlar va mijozlar jadvallaridagi Id ustunlariga havolalarni o‘z ichiga oladi. Shuningdek, u sotib olingan mahsulot miqdorini (ProductCount) va qanday narxda sotib olinganini (Narx) saqlaydi. Bundan tashqari, jadval xarid sanasini CreatedAt ustunida ham saqlaydi.

Ushbu jadvallar quyidagi ma’lumotlarni o‘z ichiga oladi:

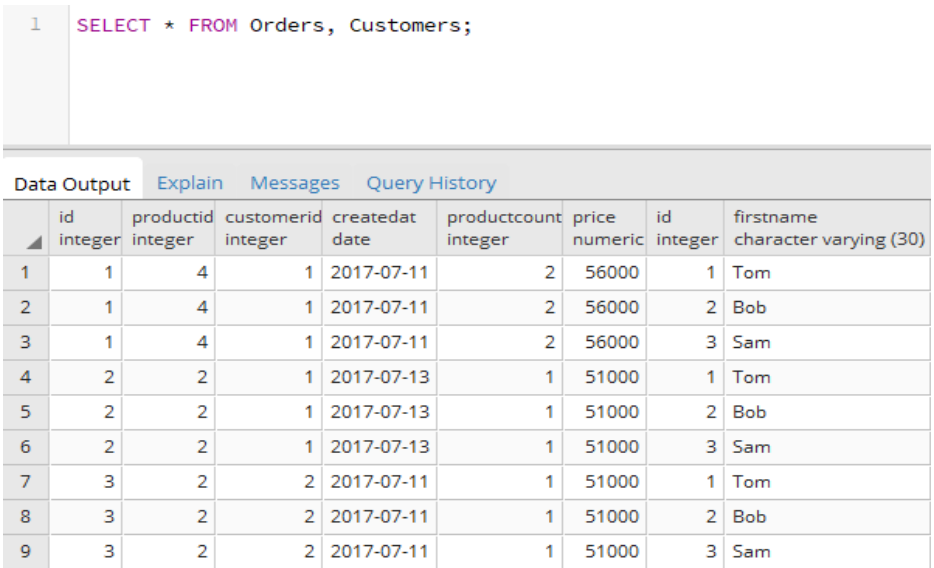
```
    INSERT INTO Products(ProductName, Company,
ProductCount, Price)
    VALUES ('iPhone X', 'Apple', 2, 66000),
    ('iPhone 8', 'Apple', 2, 51000),
    ('iPhone 7', 'Apple', 5, 42000),
    ('Galaxy S9', 'Samsung‘, 2, 56000),
    ('Galaxy S8 Plus', 'Samsung‘, 1, 46000),
    ('Nokia 9', 'HDM Global', 2, 26000),
    ('Desire 12', 'HTC', 6, 38000);

    INSERT INTO Customers(FirstName)
    VALUES ('Tom'), ('Bob'),('Sam');

    INSERT INTO Orders(ProductId, CustomerId, CreatedAt,
ProductCount, Price)
    VALUES
    (
        (SELECT Id FROM Products WHERE
ProductName='Galaxy S9'),
        (SELECT Id FROM Customers WHERE FirstName='Tom'),
        '2017-07-11',
        2,
        (SELECT Price FROM Products WHERE
ProductName='Galaxy S9')
    ),
```

```
    (
        (SELECT Id FROM Products WHERE ProductName='iPhone 8'),
        (SELECT Id FROM Customers WHERE FirstName='Tom'),
        '2017-07-13',
        1,
        (SELECT Price FROM Products WHERE ProductName='iPhone 8')
    ),
    (
        (SELECT Id FROM Products WHERE ProductName='iPhone 8'),
        (SELECT Id FROM Customers WHERE FirstName='Bob'),
        '2017-07-11',
        1,
        (SELECT Price FROM Products WHERE ProductName='iPhone 8')
    );
```

Keling, ikkita jadvalni qo‘shamiz Buyurtmalar va mijozlar:

```
SELECT * FROM Orders, Customers;
```

Ushbu tanlov yordamida Buyurtmalar jadvalidagi har bir satr Mijozlar jadvalidagi har bir qatorga moslashtiriladi. Ya'ni, siz o‘zaro bog‘lanishni olasiz. Masalan, Buyurtmalarda uchta qator, Mijozlarda esa uchta qator mavjud, shuning uchun biz 3 · 3 = 9 qatorni olamiz:



```
1  SELECT * FROM Orders, Customers;
```

Data Output | Explain | Messages | Query History

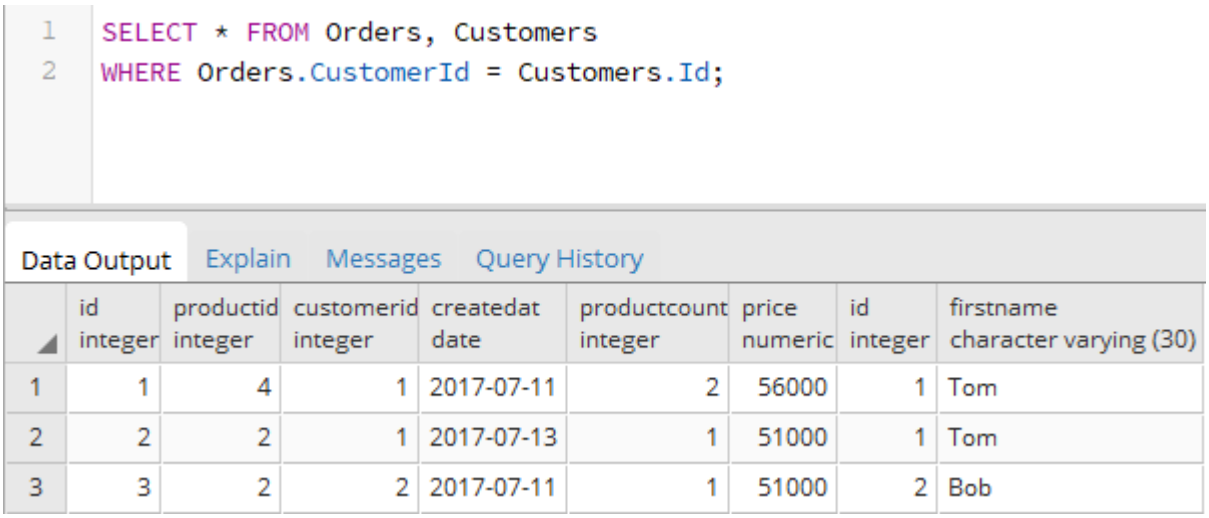
	id integer	productid integer	customerid integer	createdat date	productcount integer	price numeric	id integer	firstname character varying (30)
1	1	4	1	2017-07-11	2	56000	1	Tom
2	1	4	1	2017-07-11	2	56000	2	Bob
3	1	4	1	2017-07-11	2	56000	3	Sam
4	2	2	1	2017-07-13	1	51000	1	Tom
5	2	2	1	2017-07-13	1	51000	2	Bob
6	2	2	1	2017-07-13	1	51000	3	Sam
7	3	2	2	2017-07-11	1	51000	1	Tom
8	3	2	2	2017-07-11	1	51000	2	Bob
9	3	2	2	2017-07-11	1	51000	3	Sam

7.1-rasm. Jadvallarni qo‘shish

Ya'ni, bu holda biz ikki guruhning to‘g‘ridan-to‘g‘ri (kartezian) mahsulotini olamiz. Biroq, bunday natijani istalgan deb atash qiyin. Bundan tashqari, Buyurtmalarning har bir buyurtmasi barcha mumkin bo‘lgan mijozlar bilan emas, balki Mijozlarning ma’lum bir mijozi bilan bog‘liq.

Ushbu muammoni hal qilish uchun siz WHERE bandidan foydalanishingiz va Buyurtmalardagi CustomerId maydoni Mijozlarning Id maydoniga mos kelishi sharti bilan qatorlarni filtrlashingiz kerak:

```
1.   SELECT * FROM Orders, Customers
2.   WHERE Orders.CustomerId = Customers.Id;
```



	id integer	productid integer	customerid integer	createdat date	productcount integer	price numeric	id integer	firstname character varying (30)
1	1	4	1	2017-07-11	2	56000	1	Tom
2	2	2	1	2017-07-13	1	51000	1	Tom
3	3	2	2	2017-07-11	1	51000	2	Bob

7.2-rasm. Jadvallarni qo‘shish

Keling, uchta jadvaldagi Buyurtmalar, Mijozlar va Mahsulotlar uchun ma’lumotlarni birlashtiramiz. Ya'ni, biz barcha buyurtmalarni qabul qilamiz hamda mijoz va tegishli mahsulot haqida ma’lumot qo‘shamiz:

```
1    SELECT Customers.FirstName, Products.ProductName,
Orders.CreatedAt
2    FROM Orders, Customers, Products
3    WHERE Orders.CustomerId = Customers.Id AND
Orders.ProductId=Products.Id;
```

Bu yerda uchta jadval birlashtirilganligi sababli, kamida ikkita shart qo‘llanilishi kerak. Buyurtmalar asosiy jadval bo‘lib qoladi, undan barcha buyurtmalar olinadi va keyin mijoz haqidagi ma’lumotlar unga Orders.

CustomerId = Customers. Id sharti bo‘yicha va mahsulot haqidagi ma’lumotlar Orders. ProductId = Products. Id sharti bo‘yicha ulanadi.

```
SELECT Customers.FirstName, Products.ProductName, Orders.CreatedAt
FROM Orders, Customers, Products
WHERE Orders.CustomerId = Customers.Id AND Orders.ProductId=Products.Id;
```

Data Output | Explain | Messages | Query History

	firstname character varying (30)	productname character varying (30)	createdat date
1	Tom	Galaxy S9	2017-07-11
2	Tom	iPhone 8	2017-07-13
3	Bob	iPhone 8	2017-07-11

7.3-rasm. Jadvallarni qo‘shish

Bu holda jadval nomlari kodni sezilarli darajada oshirganligi sababli, jadval taxalluslari yordamida uni qisqartirishimiz mumkin:

```
1.  SELECT C.FirstName, P.ProductName, O.CreatedAt
2.  FROM Orders AS O, Customers AS C, Products AS P
3.  WHERE O.CustomerId = C.Id AND O.ProductId=P.Id;
```

Agar taxallusdan foydalanganda ma’lum bir jadvaldagi barcha ustunlarni tanlashingiz kerak bo‘lsa, yulduzchadan foydalanishingiz mumkin:

```
1.  SELECT C.FirstName, P.ProductName, O.*
2.  FROM Orders AS O, Customers AS C, Products AS P
3.  WHERE O.CustomerId = C.Id AND O.ProductId=P.Id;
```

INNER JOIN

Jadvallarni birlashtirishning yana bir usuli JOIN yoki INNER JOIN operatorlaridan foydalanishdir. U ichki bog‘lanish deb ataladigan narsani ifodalaydi. Uning rasmiy sintaksisi:

```
1.  SELECT столбцы
2.  FROM таблица1
3.      [INNER] JOIN таблица2
4.      ON условие1
```

```
5.        [[INNER] JOIN таблица3
6.        ON условие2]
```

JOIN operatoridan keyin ikkinchi jadval nomi keladi, uning ma’lumotlari tanlovga qo‘shilishi kerak. JOIN oldidan ixtiyoriy INNER bayonoti bo‘lishi mumkin. Uning mavjudligi yoki yo‘qligi hech narsaga ta'sir qilmaydi. Bundan tashqari, ON kalit so‘zidan keyin qo‘shilish sharti ko‘rsatiladi. Bu shart ikki jadval qanday solishtirilishini belgilaydi. Odatda, birlashma asosiy jadvalning asosiy kalitidan va bog‘liq jadvalning tashqi kalitidan foydalanadi.

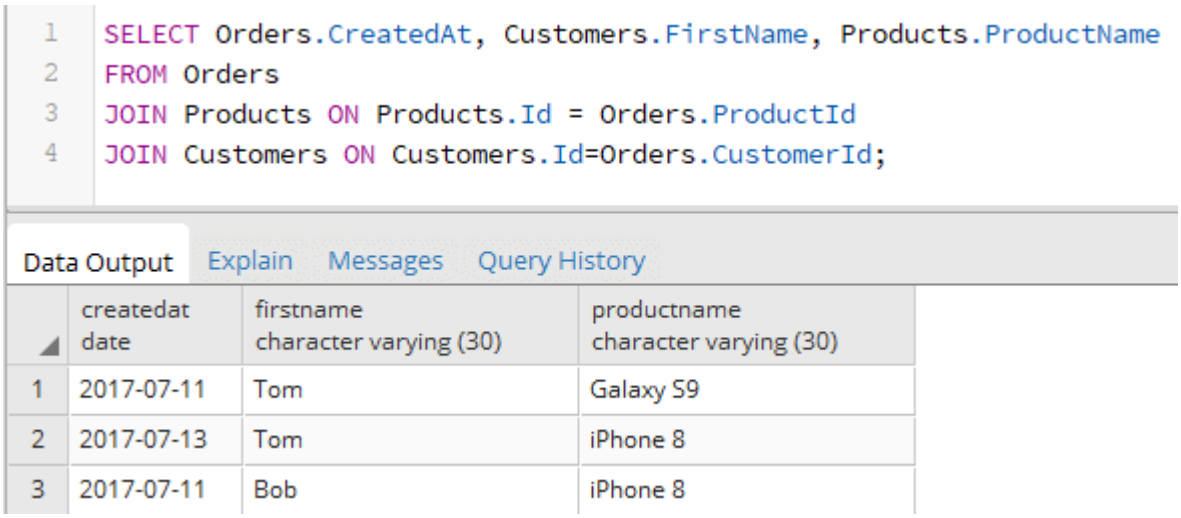
Oldingi yuqoridagi ma’lumotlar jadvallarini olaylik.

JOIN-dan foydalanib, keling, barcha buyurtmalarni tanlaymiz va ularga mahsulot ma’lumotlarini qo‘shamiz:

```
1.   SELECT Orders.CreatedAt, Orders.ProductCount,
Products.ProductName
2.   FROM Orders
3.   JOIN Products ON Products.Id = Orders.ProductId;
```

Xuddi shunday, biz boshqa jadvallarni qo‘shishimiz mumkin. Misol uchun, mijozlar jadvalidagi mijoz ma’lumotlarini buyurtmaga qo‘shamiz:

```
1.   SELECT Orders.CreatedAt, Customers.FirstName,
Products.ProductName
2.   FROM Orders
3.   JOIN Products ON Products.Id = Orders.ProductId
4.   JOIN Customers ON Customers.Id=Orders.CustomerId;
```



7.4-rasm. JOIN operatoridan foydalanish

Jadvallarni birlashtirish tufayli biz tanlovni filtrlash yoki saralash uchun ularning ustunlaridan foydalanishimiz mumkin:

```
1.  SELECT Orders.CreatedAt, Customers.FirstName,
    Products.ProductName
2.  FROM Orders
3.  JOIN Products ON Products.Id = Orders.ProductId
4.  JOIN Customers ON Customers.Id=Orders.CustomerId
5.  WHERE Products.Price > 45000
6.  ORDER BY Customers.FirstName;
```

ON kalit so‘zidan keyingi shartlar murakkabroq bo‘lishi mumkin. Misol uchun, Apple tomonidan ishlab chiqarilgan mahsulotlar uchun barcha buyurtmalarni tanlaymiz.

```
1.  SELECT Orders.CreatedAt, Customers.FirstName,
    Products.ProductName
2.  FROM Orders
3.  JOIN Products ON Products.Id = Orders.ProductId AND
    Products.Company='Apple'
4.  JOIN Customers ON Customers.Id=Orders.CustomerId
5.  ORDER BY Customers.FirstName;
```

OUTER JOIN yoki tashqi birlashmada qatnashgan bir yoki ikkita jadvalning barcha qatorlarini qaytarish imkonini beradi.

Outer Join quyidagi sintaksisga ega:

```
1.  SELECT столбцы
2.  FROM jadval1
3.      {LEFT|RIGHT|FULL} [OUTER] JOIN jadval2 ON shart1
4.      [{LEFT|RIGHT|FULL} [OUTER] JOIN jadval3 ON
    shart2]...
```

JOIN iborasi oldidan LEFT, RIGHT yoki FULL kalit so‘zlardan biri bo‘lib, ular qo‘shilish turini aniqlaydi:

LEFT: tanlov birinchi yoki chap jadvaldagi barcha qatorlarni o‘z ichiga oladi

RIGHT: tanlov ikkinchi yoki o‘ng jadvaldagi barcha qatorlarni o‘z ichiga oladi

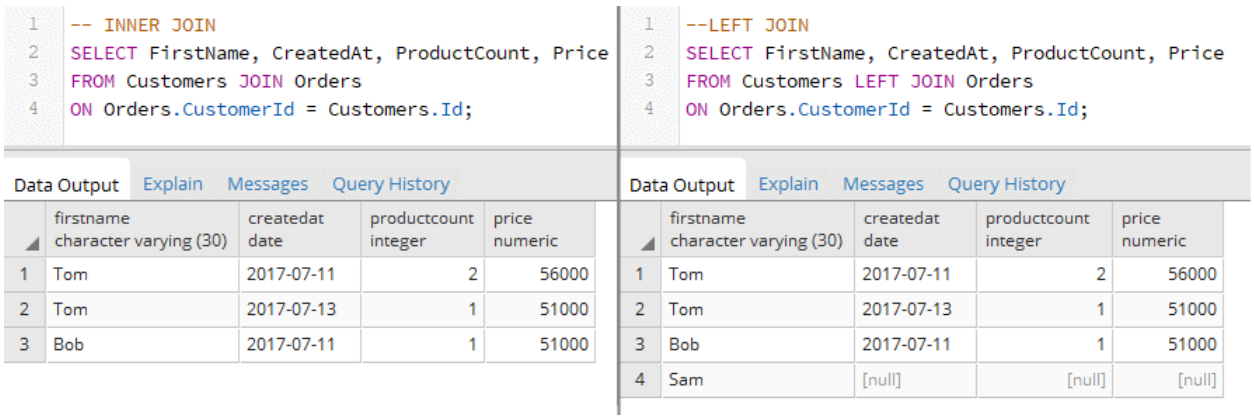
FULL: tanlov ikkala jadvaldagi barcha qatorlarni o‘z ichiga oladi

JOIN operatoridan oldin OUTER kalit so‘zi bo‘lishi mumkin, lekin u ixtiyoriy. JOIN dan keyin qo‘shiladigan jadval, so‘ngra qo‘shilish sharti ON operatoridan keyin keladi.

Buyurtmalar va mijozlar jadvallarini qo‘shish:

```
1.  SELECT FirstName, CreatedAt, ProductCount,
Price, ProductId
2.  FROM Orders LEFT JOIN Customers
3.  ON Orders.CustomerId = Customers.Id;
```

Yuqoridagi natijadan ko‘rinib turibdiki, chap tomondagi birlashma INNER qo‘shilish bilan bir xil, ammo unday emas. Shart bajarilganda Inner Join jadvallari qatorlarini birlashtiradi. Agar jadvallardan birida ushbu shartga mos kelmaydigan qatorlar mavjud bo‘lsa, u holda bu qatorlar chiqish tanloviga kiritilmaydi. Left Join birinchi jadvaldagi barcha satrlarni tanlaydi va keyin o‘ng jadvaldagi qatorlarni ularga birlashtiradi. Masalan, mijozlar jadvalini olaylik va mijozlarga ularning buyurtmalari haqida ma’lumot qo‘shamiz:



7.5-rasm. INNER JOIN va LEFT JOIN.

Yuqoridagi misolda ulanish turini o‘ng tomonga o‘zgartiramiz:

```
1.  SELECT FirstName, CreatedAt, ProductCount, Price,
ProductId
2.  FROM Orders RIGHT JOIN Customers
3.  ON Orders.CustomerId = Customers.Id;
```

Endi Mijozlarning barcha qatorlari tanlanadi va shartlar bo‘yicha Buyurtmalar jadvalidagi qatorlar allaqachon ularga biriktiriladi:

```
SELECT FirstName, CreatedAt, ProductCount, Price, ProductId
FROM Orders RIGHT JOIN Customers
ON Orders.CustomerId = Customers.Id;
```

Data Output | Explain | Messages | Query History

	firstname character varying (30)	createdat date	productcount integer	price numeric	productid integer
1	Tom	2017-07-11	2	56000	4
2	Tom	2017-07-13	1	51000	2
3	Bob	2017-07-11	1	51000	2
4	Sam	[null]	[null]	[null]	[null]

7.6-rasm. RIGHT JOIN

Mijozlar jadvalidagi mijozlardan birining Buyurtmalardan bog‘langan buyurtmalari yo‘qligi sababli, Buyurtmalardagi mos ustunlar NULL bo‘ladi.

FULL JOIN ikkala jadvalni birlashtiradi:

```
1.  SELECT FirstName, CreatedAt, ProductCount, Price,
    ProductId
2.  FROM Orders FULL JOIN Customers
3.  ON Orders.CustomerId = Customers.Id;
```

Nazorat savollari

1. SQL nima va uning yangi standarti qachon qabul qilingan?
2. Qaysi buyruq yordamida jadvaldan ma’lumotlar qidiriladi?
3. Qaysi buyruqlar yordamida jadvaldagi zarur ma’lumotlar olinadi?

ADABIYOTLAR

1. Knuth, D. E. The Art of Computer Programming Vol. I: Fundamental Algorithms, Addison – Wesley, Reading, Mass. (Русский перевод: Кнут Д. Искусство программирования. Том 1: Основные алгоритмы. – М., Вильямс, 2000.)
2. Джон Бентли. Жемчужины программирования. СПб:Питер, 2002.-272 с.
3. Дейт К., Хью Дарвен. Основы будущих систем баз данных. - М: Янус-К, 2004.
4. Нагао М., Катаяма Т., Уэмура С. Структуры и базы данных. - М.: Мир, 2001. - 197 с.
5. Бойко В.В., Савинков В.М. Проектирование баз данных информационных систем. - М.: Финансы и статистика, 2009. - 351 с.
6. Кириллов В.В. Структурированный язык запросов (SQL). - СПб.: ИТМО, 2000. - 80 с.
7. Дейт К. Введение в системы баз данных. 8-е изд., М.: СПб.: Вильяме.- 2005.
8. Хомоненко А.Д. Базы данных. Учебник для высших учебных заведений / Под ред. np(xj). А. Д. Хомопепко. - 6-е изд., доп. - СПб.: КОРОНА-Век, 2009. - 736 с.
9. Гектор Гарсиа-Молина, Джеффри Ульман, Дженифер Уидом. Системы баз данных. Полный курс. - Москва, Санкт-Петербург, Киев, Вильямс, 2003.
10. Система управления базами данных: P:Base System V/ С. В. Горин, Кравченко С. В.,. Кузина Г. Г, Силантьева А. В.. -Москва: МГУ, 2003.
11. Архангельский А.Я. Работа с локальными базами данных в Delphi 5: производственно-практическое издание. – М.: Наука, 2000.
12. Грабер М. Введение в SQL. - М.: Лори, 2001. - 379 с.
13. Ким М. Курс лекции по Транзакционная обработка. - Т., 2001.
14. Когаловский М.Р. Энциклопедия технологий баз данных. - М.: Финансы и статистика, 2002.
15. Боуман Д, Эмерсон С., Дарновски М. Практическое руководство по SQL. - Киев: Диалектика, 1997.
16. Джексон Г. Проектирование реляционных баз данных для использования с микроЭВМ. - М.: Мир, 2004. - 252 с.
17. Диго С.М. Проектирование и использование баз данных. - М.: Финансы и статистика, 1998. - 208 с.

MUNDARIJA

MA’LUMOTLAR BAZASINI BOSHQARISH VA DASTURLASH TEXNOLOGIYALARI

laboratoriya ishlarini bajarish uchun

USLUBIY KO‘RSATMALAR

I qism

Tuzuvchilar:
dots. Sodiqova Sh.Sh.
ass. Ashuraliyev A.A.
ass. Sodiqova F.B.
Muharrir: Miryusupova Z.M.